University of Southampton

Faculty of Engineering and Physical Sciences

Aeronautics, Astronautics, and Computational Engineering

Multimodal Speech and Visual Gesture Control Interface Technique for Small Unmanned Multirotor Aircraft

by

Ayodeji Opeyemi Abioye

ORCID iD: 0000-0003-4637-3278

Thesis for the degree of Doctor of Philosophy

August 2019

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
Aeronautics, Astronautics, and Computational Engineering

Doctor of Philosophy

Multimodal Speech and Visual Gesture Control Interface Technique for Small Unmanned Multirotor Aircraft

by

Ayodeji Opeyemi Abioye

This research conducted an investigation into the use of novel human computer interaction (HCI) interfaces in the control of small multirotor unmanned aerial vehicles (UAVs). The main objective was to propose, design, and develop an alternative control interface for the small multirotor UAV, which could perform better than the standard RC joystick (RCJ) controller, and to evaluate the performance of the proposed interface. The multimodal speech and visual gesture (mSVG) interface was proposed, designed, and developed. This was then coupled to a Rotor_S ROS Gazebo UAV simulator. An experiment study was designed to determine how practical the use of the proposed multimodal speech and visual gesture interface was in the control of small multirotor UAVs by determining the limits of speech and gesture at different ambient noise levels and under different background-lighting conditions, respectively. And to determine how the mSVG interface compares to the RC joystick controller for a simple navigational control task - in terms of performance (time of completion and accuracy of navigational control) and from a human factors perspective (user satisfaction and cognitive workload). 37 participants were recruited. From the results of the experiments conducted, the mSVG interface was found to be an effective alternative to the RCJ interface when operated within a constrained application environment. From the result of the noise level experiment, it was observed that speech recognition accuracy/success rate falls as noise levels rise, with 75 dB noise level being the practical aerial robot (aerobot) application limit. From the results of the gesture lighting experiment, gestures were successfully recognised from 10 Lux and above on distinct solid backgrounds, but the effect of varying both the lighting conditions and the environment background on the quality of gesture recognition, was insignificant ($< 0.5\%$), implying that the technology used, type of gesture captured, and the image processing technique used were more important. From the result of the performance and cognitive workload comparison between the RCJ and mSVG interfaces, the mSVG interface was found to perform better at higher nCA application levels than the RCJ interface. The mSVG interface was 1 minute faster and 25% more accurate than the RCJ interface; and the RCJ interface was found to be 1.4 times more cognitively demanding than the mSVG interface. The main limitation of this research was the limited lighting level range of 10 Lux - 1400 Lux used during the gesture lighting experiment, which constrains the application limit to low lighting indoor environments. Suggested further works from this research included the development of a more robust gesture and speech algorithm and the coupling of the improved mSVG interface on to a practical UAV.

Contents

List of Figures

List of Tables

List of Codes

Declaration of Authorship

Print name:	Ayodeji Opeyemi Abioye

Title of Thesis:	Multimodal Speech and Visual Gesture Control Interface Technique for Small Unmanned Multirotor Aircraft

I declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.
I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as: Abioye et al. (2019a), Abioye et al. (2018b), Abioye et al. (2018a), Abioye et al. (2017), and Abioye et al. (2016).

Signature:		Date:	21-August-2019

Acknowledgment

First of all, I wish to acknowledge all my supervisors: Dr Stephen D. Prior, Dr Glyn T. Thomas, Dr Peter Saddington, and Prof. Sarvapali D. Ramchurn, for their continued advice through the course of this PhD research work. Despite their very busy schedules, they remained consistent and committed to providing me with all the guidance, support, and advice I needed in order to successfully complete this PhD research. I wish to thank Phil Harvey from HantSAR (Hampshire Search and Rescue) who I interviewed in order to understand how search and rescue works, for his time, effort, expertise, experience, knowledge, and encouragement given towards my PhD research. Andre Oliveira from Tekever Lisbon (Portugal), who I interviewed in order to learn more about Tekever's BRAINFLIGHT project; he helped me to understand a bit more on how brain-computer interfaces work. I am thankful for his time, effort, expertise, experience, knowledge, and encouragement. My PhD upgrade examiners: Dr Andrew Chipperfield and Dr David Toal for challenging me to understand my work better and refine my efforts appropriately. Dr David Toal for suggesting the S-Curve technology comparison model used in Section 1.3 of Chapter 1. Dr Chang Liu for making available (open source) his Unicorn indoor navigation platform, although this was not used because the research was eventually simulated using a ROS Gazebo UAV simulator. My PhD examiners, Dr Joel Fischer, from the University of Nottingham's Mixed Reality Laboratory, and Dr András Sóbester, from the University of Southampton's Computational Engineering and Design Group, for using their wide experience, remarkable depth of understanding, and excellent expertise in this area, to examine this PhD research.

My sponsors and financial benefactors, my parents - for my first year funding, the Petroleum Technology Development Fund (PTDF) Nigeria - for funding the remaining years of my PhD under Scholarship Grant PTDF/16PHD052/862, and Faculty of Engineering and the Environment (University of Southampton) - for the Studentship to support my maintenance and living expenses, and the attendance of international conferences in Portugal (2016) and Japan (2018).

Abbreviations

AAP	Altitude, Attitude, and Position
Aerobot	Aerial Robot
ACL	Autonomy Control Level
AFRL	US Air Force Research Lab
Agri-bot	Agricultural Aerobot
AI	Artificial Intelligence
ANOVA	Analysis of Variance
AOV	Angle of View
API	Application Programming Interface
AR	Augmented Reality
ASL	American Sign Language
ASR	Audio Speech Recognition/Recognisers
API	Application Programming Interface
BCI	Brain Computer Interface
BSL	British Sign Language
CAD	Computer Aided Design
COTS	Commercial Off-The-Shelf
CRT	Cathode Ray Tube
DERA	UK Defence Evaluation Research Agency
EEG	Electroencephalography
EMG	Electromyography
EM	Expectation Maximization
EOG	Electrooculography
EP	Evoked Potential
ERP	Event-Related Potential
ESC	Electronic Speed Controller
FC	Flight Controller
fMRI	functional Magnetic Resonance Imaging
fNIRs	functional Near Infrared spectroscopy
FOV	Field of View
FPV	First Person View
GIS	Geographical Information System
Ha3mi	Human Aerobot Tri-Modal Interface
HAI	Human Aerobotic Interaction
HCI	Human Computer Interaction
HHI	Human Human Interaction
HMI	Human Machine Interaction
HMM	Hidden Markov Model
HRI	Human Robot Interaction
HTK	HMM Toolkit speech recogniser
iEMG	intramuscular EMG
IMU	Inertial Measurement Unit
ISR	Intelligence, Surveillance, and Reconnaissance
IR	Infrared
KBD	Keyboard
L4T	“Linux for Tegra” or “Linux4Tegra”
LED	Light Emitting Diode

LS	Lighting Stage
MCPU	Multimodal Control Processing Unit
MMC	Multimodal Communication
MMI	Multimodal Interaction
MMG	Magnetomyography
MR	Mixed Reality
MRI	Magnetic Resonance Imaging
MSL	Mexican Sign Language
mSVG	Multimodal Speech and Visual Gesture
nCA	Navigational Control Autonomy
nHHI	near Human-Human Interaction
NIRS	Near Infrared Spectroscopy
NUI	Natural User Interface
OLED	Organic Light Emitting Diode
OODA	Observe, Orient, Decide, and Act
ONR	Office of Naval Research
OpenCL	Open Computing Language
OpenCV	Open Source Computer vision
OS	Operating System
PACT	Pilot Authority and Control of Task
PET	Positron Emission Tomography
RC	Radio Controlled
RCJ	Radio Controlled Joystick
RF	Radio Frequency
RFDS	RealFlight Drone Simulator
ROC	Receiver Operating Characteristic
ROS	Robot Operating System
SAA	Sense And Avoid
SBC	Single Board Computer
SC	Speech Command
SDK	Software Development Kit
sEMG	surface EMG
SGR	Speech Gesture Ratio
SMC	Single Modal Communication
SMR	Sensorimotor Rhythm
SQUID	Superconducting Quantum Interference Device
SSVEP	Steady State Visual Evoked Potential
SW	Speech Word
TLX	Task Load Index
UAV	Unmanned Aerial Vehicle
UAVM	Unmanned Aerial Vehicle-Manipulator
UBEC	Universal Battery Eliminating Circuit
UDP	User Datagram Protocol
UGV	Unmanned Ground Vehicle
USB	Universal Serial Bus
VAT6	Visual, Auditory, Tactile, & Sixth
VGR	Visual Gesture Recognition
VLL	Varying Lighting Level - Command
VNL-WF	Varying Noise Level - Word Frequency
VNLC	Varying Noise Level - Command
VR	Virtual Reality
VSRi	Visual, Speech, & Radio interface

Chapter 1

Introduction

"Because of the limitations of the human cognitive skills, judgment, decision-making, and tactical understanding in the use of Unmanned Aerial Vehicles (UAV), there is a need to redesign the current human-computer interface to improve the interaction and communication links between operators and the UAVs..."

– Cavett et al. (2007)

1.1 Research Background

Unmanned multirotor aircraft have been gaining a lot of popularity in recent years (Li et al., 2018; Michelson, 2014). These systems have found applications in diverse areas ranging from military to civilian applications (Romell and Karjalainen, 2017; Liu and Foina, 2016). The popularity of these systems, particularly in the civilian domain, has led to several creative applications in agriculture, aerial survey and inspection, surveillance and monitoring, search and rescue, photography and videography, entertainment, sports, health care, law enforcement, and environmental conservation. Table 1.1 lists some of these applications (refer to Table A.1 in Appendix A for a more comprehensive coverage). At the time of this research, the most common application for a small UAV is cost-effective aerial image data acquisition, but this may soon be superseded by drone deliveries of civilian goods, military last mile delivery and resupply, and the delivery of medical supplies. As the popularity and application of unmanned aerial vehicles increases, the number of operators, users, developers, researchers, and other beneficiaries have also increased significantly. This research is interested in how this increasing league of human operators interact with these small multirotor UAVs.

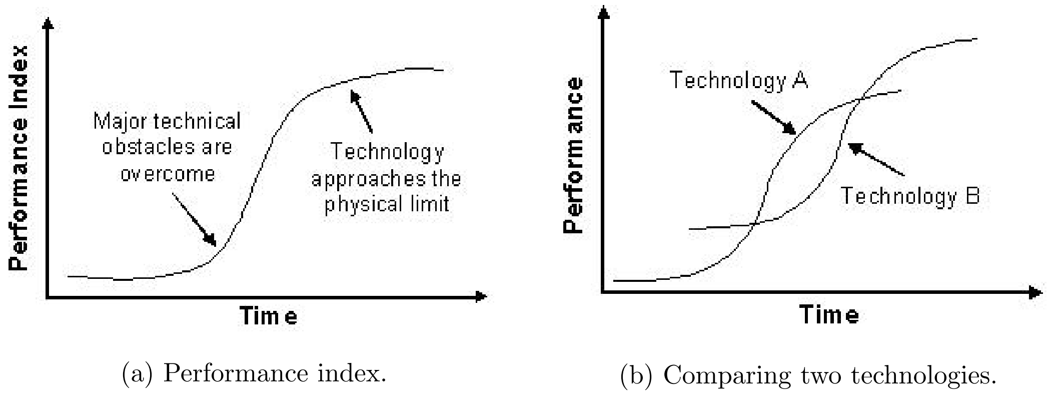
The current designs of human computer interaction (HCI) control interfaces for small UAVs overloads the limited physical and cognitive ability of the UAV operator. Sometimes, two operators may be required to control a single UAV (Aeryon Labs Inc., 2011) particularly in more demanding environments or scenarios, such as a search and rescue mission, where the primary operator (pilot) flies the UAV and a secondary operator (spotter) controls the on-board camera in search of missing persons. Mapping two operators to one UAV is the opposite of the idea portrayed in the DJI (2015a) Phantom X multi-drone control concept, where a single operator simultaneously controls two small multirotor aircraft in a sky-painting application scenario, which seemed to be a simpler, more interesting, and intuitive way of interacting with aerial robots. In a similar manner, researchers and theatre artists from the Texas A&M University adopted the use of small UAVs as little fairies in their theatre adaptation and performance of William Shakespeare's play "A Midsummer Night's Dream" (Murphy et al., 2011). There is also the flying table lamp shade project, codenamed "SPARKED: a live interaction between humans and quadcopters", featuring a choreographed performance by entertainers and researchers from Cirque du Soleil, ETH Zurich, and Verity Studios (Cirque du Soleil et al., 2014; Waibel, 2014a), dreaming of the possibilities of direct human aerobotic interaction with co-located UAVs. Human swarm aerobotic interaction is also a related field of interest (Nagi et al., 2014). The growing presence of aerial robots in civilian airspace raises a social interaction challenge, which if properly done, would allow robots to blend in, share, and participate in the human world.

Table 1.1: Civilian applications of aerial robots.

S/No	CATEGORY	APPLICATIONS
1	Aerial Inspection	Rail Network Inspection; Offshore oil rig maintenance inspection
2	Disaster Relief	Search and rescue operations; Healthcare supply delivery
3	Emergency Services	Law enforcement applications such as portable aerial surveillance; Firefighting applications
4	Entertainment Applications	Commercial Newsgathering; Filmmaking; Theatrical entertainment (Murphy et al., 2011); Drone racing
5	Environmental Applications	Wildlife conservation; Habitat exploration; Flood monitoring; Open water monitoring
6	Field Applications	Aerial Survey & Mapping; Agricultural applications
7	Industrial Applications	Aerial robotic manipulators (Muscio et al., 2016) - flying robotic arms in manufacturing; Warehouse quick light-goods transportation; Indoor inspection
8	Photography	Selfies; Sport Videography; Mountaineering expedition
9	Transportation	Drone delivery of goods e.g. Amazon (2016) & DHL delivery trials; Drone rides transporting humans

1.2 Research Problem

Current human computer interaction control interfaces for small multirotor UAVs may be difficult to learn, require long training hours, and may demand too much of the operator's cognitive effort. There is also the need for intuitive control interaction with aerial robots (aerobots) at higher levels of autonomy, for which the standard RC joystick controller is otherwise arduous. There is also the s-curve model explanation for HCI interfaces. The S-curve framework is particularly useful for analysing technological cycles and predicting the introduction, adoption, growth, and maturation of technological innovations as demonstrated in Griliches (1957). As shown in Figure 1.1a, a lot of research is often required in the early stages of new technologies to break the technological barrier, after which growth and adoption is accelerated to the technological innovation's maturity. Performance for a specific technology peaks as it approaches its maturity or physical limit, beyond which further pushing the performance becomes increasingly difficult. While it may be possible to intensify research at this point to build a new s-curve on the previous, it may be more cost effective and realistic to consider alternative technologies.


Performance Index
Major technical obstacles are overcome
Technology approaches the physical limit
Time
Performance
Technology A
Technology B
Time

(a) Performance index. (b) Comparing two technologies.

Figure 1.1: The S-Curve framework (Scocco, 2006).

The S-curves shown in Figure 1.1b describe a typical variation in the performances of two competing technologies as a function of time and effort. If the RC joystick controller is technology A, and if this technology is considered matured having been around for over half-a-decade, and its performance and adoption rate is peaking, then this research builds on a competing technology B, which is yet to breakthrough its major technical barriers. This research contributes by proposing methods and algorithms that could gradually push the technology towards the breakthrough and exponential growth phase. Therefore this research contributes to a technology in the pipeline, that could improve the human aerobotic interaction experience and offer unprecedented levels of performance.

Since robots are designed to reduce human workload, risk, cost, and human fatigue-driven errors, Fong and Nourbakhsh (2000) suggested that it is crucial to make the human-robot interaction effective, efficient, and natural, in order to achieve this. Green et al. (2007) observed that "people use speech, gesture, gaze and non-verbal cues to communicate in the clearest possible fashion." Therefore a technique that is natural, intuitive, easy to learn, easy to use, and low-cost, is being proposed, designed, and developed by this research work, as shown in Figure 1.2.

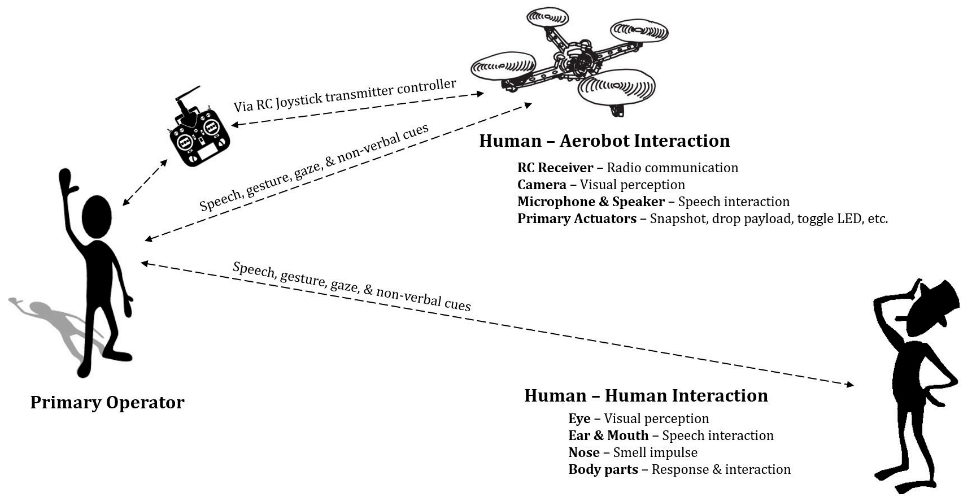

Via RC Joystick transmitter controller
Speech, gesture, gaze, & non-verbal cues
Human – Aerobot Interaction
RC Receiver – Radio communication
Camera – Visual perception
Microphone & Speaker – Speech interaction
Primary Actuators – Snapshot, drop payload, toggle LED, etc.
Speech, gesture, gaze, & non-verbal cues
Primary Operator
Human – Human Interaction
Eye – Visual perception
Ear & Mouth – Speech interaction
Nose – Smell impulse
Body parts – Response & interaction

Figure 1.2: Human-aerobotic interaction emulating human-human interaction.

1.3 Aims and Objectives

The aim of this research was to conduct an investigation into the use of novel human-computer interfaces in the control of a small unmanned multirotor aircraft.

The objectives of this research were to:

1. review current HCI interfaces that can be used for the control of small multirotor UAVs
2. propose an effective alternative interface for the small multirotor UAV that could perform better than the standard RC joystick controller
3. design and develop the alternative interface
4. compare the performance of the alternative interface with the standard RC joystick controller

1.4 Application Scenarios

1.4.1 A case of the Matterhorn

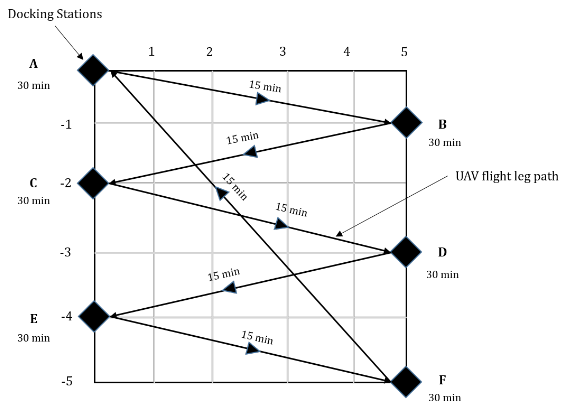
Consider a case in which a small multirotor UAV is being developed to 1) patrol a dangerous region of the Alps, 2) provide signposting information to climbers, 3) alert search and rescue teams in case of any incidence, and 4) support search and rescue efforts or team operations. If such a patrol UAV would be required to interact with climbers when needed, how would the UAV register the climber's requests? As climbers would not normally have the UAV's RC controller, an intangible human-human interaction (HHI)-like multimodal speech and visual gesture interaction method would seem more suitable for such scenarios.

Figure 1.3: Patient awaits pick up by Air Zermatt helicopter Root and Air Zermatt (2016).

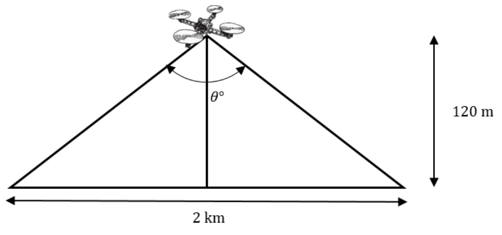
An application of the multimodal speech and visual gesture (mSVG) interaction method on a patrol, search, and rescue robot could be the Alps in Southcentral Europe, where sporting activities such as climbing, mountaineering, hiking, cycling, paragliding, mountain biking, rafting, skiing, snowboarding, curling, and snow shoeing, are quite popular. People visiting these places for the first time could easily get lost if not very careful, climbers could fall when anchored on deceptively rigid surfaces, and people new to certain sports could get hurt, particularly when caught in stormy weathers. Therefore, patrol UAVs could be dispatched to assist local search and rescue operation efforts immediately after a storm. For example in the Matterhorn, about 1,700 rescue missions are conducted annually (Root and Air Zermatt, 2016). These high number of rescues is because "the Matterhorn is home to numerous glaciers, which are laced with countless deep crevasses, many of which are hidden by snow that can give way without warning, swallowing up climbers and skiers in the process" (Root and Air Zermatt, 2016). Figure 1.3 shows a mountaineer being rescued by the Air Zermatt search and rescue helicopter team. With the limited number of resources and man power, how can this operation be run more efficiently? Aerobots could be particularly useful in performing patrol, searching areas, and transporting small supplies. Patrol UAVs could fly along predefine routes, warning climbers of hidden crevasses, because signs on routes are easily covered by snow. The patrol UAVs could also be used to keep track of climbers' progress during its regular flyovers. This could help find climbers in distress even before a call for help is made, which reduces the time between fall and call, potentially shortening rescue time, which in turn increases a rescued climber's survival chances - especially in situations where every second counts. In addition to this, a fallen climber may be unconscious, and therefore may be unable to call for help themselves, the UAV could make the call after failing to establish communication with the climber. Also, climber tracking information collected during patrol could be used to narrow down the search area in the event of an emergency or if a climber goes missing.

The UAV could also be used to provide verbal signposting, alerting climbers of their proximity to deep crevasses hidden by snow. Signs are probably unusable here because they could be easily covered by snow. According to Root and Air Zermatt (2016) "people never know how close they are to the limits, every mistake they make could be their last". A typical signposting interaction could be - a UAV informs a lone climber, *"hello, deep crevasse 400 m ahead"*, and the lone climber could respond with *"ok"* as an affirmative or *"repeat"* to have the UAV repeat itself. Prior to the climb, all climbers and mountaineers would have already been briefed on appropriate UAV responses, and how to ask for help. They may also choose to opt out of being helped by the UAV during its routine fly over, if they think that this may be distracting, in which case the UAV avoids interacting with the climbers. The climber sticks a *"don't disturb"* QR code patch on their backpack, which the UAV scans on approach and flies away instead, except if it is an emergency. Patrol UAVs could also warn climbers of rapidly changing weather conditions and advise them on the nearest refuge points. In the event that the patrol UAV comes across a fallen lone climber, it could potentially alert climbing parties nearby to act as first responders, if they are quite close to the incident site, while also alerting the central control room of the emergency.

An array of UAVs with front and downward facing cameras, thermal cameras, on-board navigational aid (GPS), on-board sense and avoid (proximity sensors), microphones, speakers, etc. could be used to execute this patrol operation as described in Figure 1.4a. Immediately after a storm, such small patrol UAVs augmented with on-board computation abilities via single board computers with microphone arrays, cameras, a speaker, and other sensors, could be dispatched



(a) UAV Patrol Grid



(b) UAV View Pan area

Figure 1.4: $5 \times 5\ km^2$ patrol area grid in six flight legs and with six docking stations.

to patrol and search specific hotspot areas in grids of $5 \times 5\ km^2$. But how should such an mSVG control interface be effectively developed? This thesis describes our approach to developing an mSVG control method for a small multirotor UAV. We developed a mathematical model, converted this into a program algorithm, tested this in MATLAB, and then performed a graphical simulation using a ROS Gazebo firefly UAV simulator. The idea is to run all the computation required for the mSVG on a single board computer, and couple this to the flight controller of a typical UAV, with the SBC communicating with the UAV flight controller via waypoint navigation.

1.4.2 The domestic robot assistant

Consider a case where a small multirotor UAV is being developed to assist elderly or sick persons with limited mobility, in performing small errands around the house, such as fetching a bottle of water from the kitchen downstairs for the user upstairs, or fetching a pair of reading glasses from the bedroom upstairs for a user downstairs, as shown in Figure 1.5. In order for the UAV to successfully complete the task described, the UAV is assumed to be equipped with a multi-joint robotic arm manipulator and gripper mechanism, similar to those described in Kim et al. (2016), Muscio et al. (2016), and Verma (2016) works on augmenting small multirotor UAVs with functional actuators. The small multirotor UAV could also be useful in alerting emergency services in the case of a domestic incident such as a trip-fall, fire, gas leak, etc. observed during it roaming flight around the house.

But how does the user interact with such aerial robot being designed to integrate with the household? The standard RC joystick controller requires the user to reach for a tangible interface each time, which is difficult to learn and use, particularly for elderly or sick persons with reduced

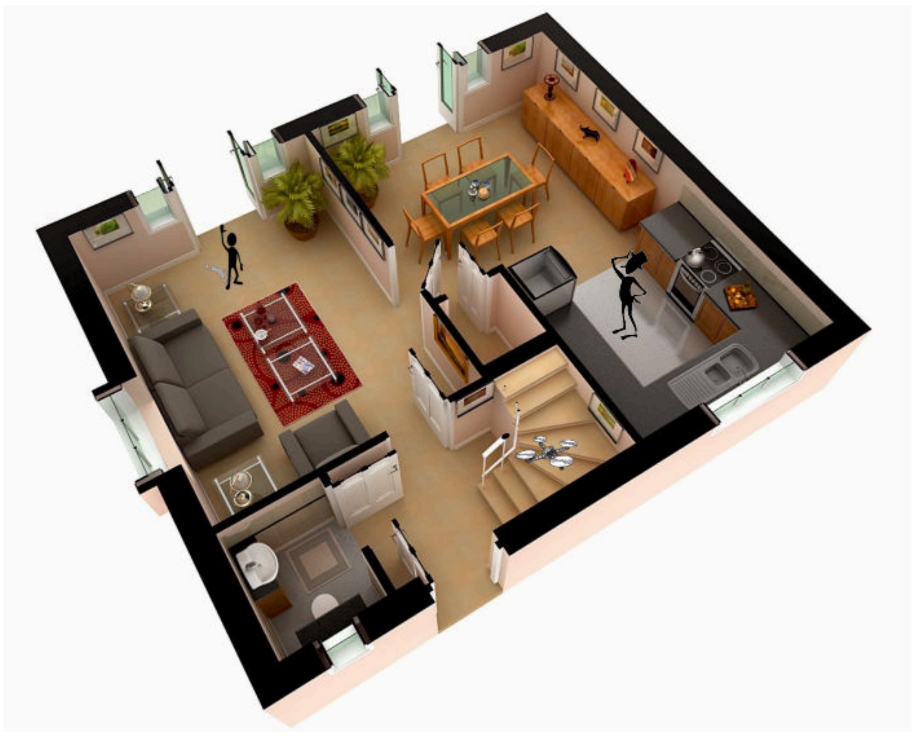


Figure 1.5: Domestic-assist UAV application scenario.

hand dexterity and limited mobility. Again, the multimodal speech and visual gesture interface technique being proposed in the previous scenario (Section 1.4.1) could also be useful in this indoor application scenario. In fact, the UAV microphone could be daisy chained with Amazon echo devices installed around the house, so that the UAV could be issued commands without it being in view. In other words, a user in the bedroom upstairs, could ask *'alexa'*, the Amazon Echo device voice assistant, to ask the UAV in the living room downstairs to fetch a bottle of water from the kitchen (also downstairs), and bring it to them in the bedroom (upstairs). Unlike the previous scenario, this scenario focuses on an indoor application setting within the home, with very low lighting conditions compared to a sunny outdoor environment.

1.5 Research Questions

The research questions were formulated based on the combination of the research aims and objectives, and the proposition of the multimodal speech and visual gesture in communication control signals to small multirotor UAVs. Therefore, the questions addressed in this research were:

1. How practical is the use of the speech and gesture in the control of small multirotor UAVs?
   (a) What is the limit of speech within the context of ambient noise at different dB levels?

(b) What is the limit of gesture within the context of different backgrounds and varying lighting levels?

2. How does an mSVG interface compare to the RC joystick controller for a simple navigational control task?

   (a) In terms of performance - time of completion and accuracy of navigational control?

   (b) From a human factors perspective - user satisfaction and cognitive workload?

1.6 Hypothesis

In addressing the research questions, an hypothesis was proposed that "*the multimodal speech and visual gesture interface was better than the RC joystick controller in terms of physical performance and cognitive workoad*".

This was because the multimodal speech and visual gesture interface was considered to be natural, easy to use, easy to learn, and intuitive. An experiment was designed to test this hypothesis in Chapter 4 and the result was presented, analysed, and discussed in Chapter 8. The analysis of variance (ANOVA) and two-sample t-test statistic test tools were used in the analysis of the results in Chapter 8.

1.7 Research Application Examples

1.7.1 Drone delivery

Figure 1.6: Amazon Prime Air first package delivery Amazon (2016).

Amazon (2016) developed a drone delivery UAV capable of delivering packages weighing up to 2.25 kg, to destinations within 20 minute of their facility, autonomously controlled from take-off, to delivery, return, and landing, being guided by GPS satellite navigation. Figure 1.6 shows an Amazon Prime air drone taking off to deliver a package. The application of this research work in this example is the enabling of an interaction possibility between the customer and the Amazon delivery drone.

1.7.2 Rescue and relief missions

In the wake of a natural disaster, like the Nepal earthquake crisis shown in Figure 1.7, communication and transportation infrastructures are usually damaged, making certain areas inaccessible to rescue or relief missions. Again, the application of this research is in the enabling of an interaction possibility between stranded persons and the search UAV. Stranded persons could be able to communicate their needs and requirement despite not having a tangible UAV control device.

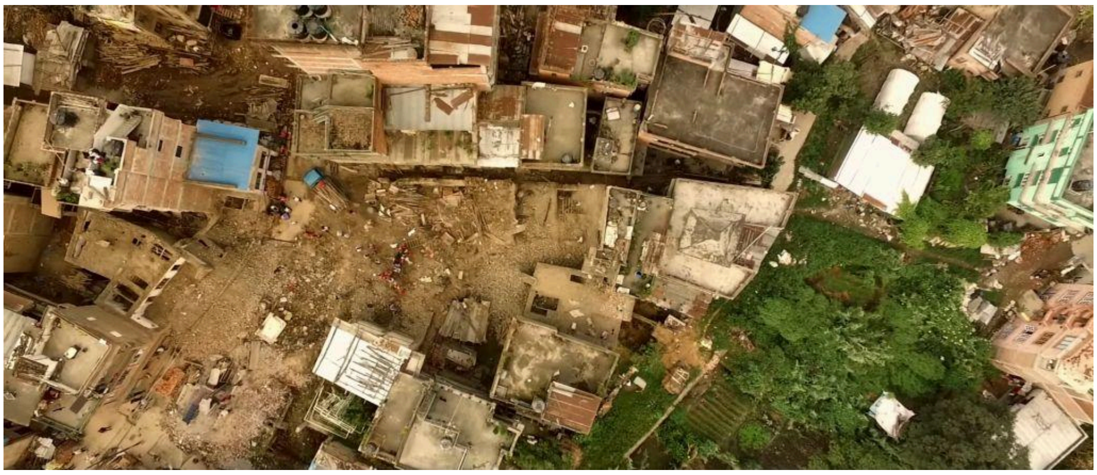


Figure 1.7: Earthquake crisis mapping in Nepal DJI et al. (2015).

1.7.3 Inclusive robotics for persons living with disability

This research makes it possible for persons living with certain disabilities to interact with UAV via speech, intuitive gestures, or sign language. For example, Luis-pérez et al. (2011) investigated the control of a mobile service robot using the Mexican Sign Language (MSL). Liu et al. (2016b) also performed an investigation for astronaut-robot control using the American Sign Language (ASL).

1.8 Research Contributions

The main contributions of this research work have been published in the following peer reviewed outlets as journals, book chapter, and conference proceedings.

1.8.1 Journals

1. Abioye, A. O., Prior, S. D., Saddington, P. and Ramchurn, S. D. (2019) Effects of Varying Noise Levels and Lighting Levels on Multimodal Speech and Visual Gesture Interaction with Aerobots, Applied Sciences. Basel, Switzerland: MDPI, 9(10), pp. 1-29. https://doi.org/10.3390/app9102066

2. Abioye, A. O., Prior, S. D., Saddington, P. and Ramchurn, S. D. (2019) The performance and cognitive workload analysis of a multimodal speech and visual gesture (mSVG) UAV control interface, IEEE ACCESS - Submitted, xx(xx), pp. 1-25.

1.8.2 Conference Papers

3. Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P. and Ramchurn, S. D. (2018) The Multimodal Speech and Visual Gesture (mSVG) Control Model for a Practical Patrol, Search, and Rescue Aerobot, in Giuliani, M., Assaf, T., and Giannaccini, M. E. (eds) Towards Autonomous Robotic Systems - Part of the Lecture Notes in Computer Science book series (LNCS, volume 10965). Bristol, UK: Springer International Publishing, pp. 423-437. doi: https://doi.org/10.1007/978-3-319-96728-8_36

4. Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P. and Ramchurn, S. D. (2018) Quantifying the effects of varying light-visibility and noise-sound levels in practical multimodal speech and visual gesture (mSVG) interaction with aerobots, in Meen, Prior, and Lam (eds) 2018 IEEE International Conference on Applied System Invention (ICASI). Chiba, Tokyo, Japan: IEEE, pp. 842-845. https://doi.org/10.1109/ICASI.2018.8394395

5. Abioye, A. O., Prior, S. D., Thomas, G. T. and Saddington, P. (2016) The Multimodal Edge of Human Aerobotic Interaction, in Blashki, K. and Xiao, Y. (eds) International Conferences Interfaces and Human Computer Interaction. Madeira, Portugal: IADIS Press, pp. 243-248.

1.8.3 Book Chapters

6. Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P. and Ramchurn, S. D. (2017) Multimodal Human Aerobotic Interaction, in Issa, T., Kommers, P., Issa, T., Isaas, P., and

Issa, T. B. (eds) Smart Technology Applications in Business Environments. Pennsylvania, USA: IGI Global, pp. 39-62. Chapter 3.

1.8.4 Posters & Abstracts

7. Abioye, A. O., Prior, S., Thomas, T., Saddington, P. & Ramchurn, S. "A practical mSVG interaction method for patrol, search, and rescue aerobots - Extended Abstract and Poster" In UK-RAS Network Conference (December) 2017, Bristol, UK.

8. Abioye, A. O., Prior, S. & Thomas, T. "Aerial robot control interfaces - Poster" NEC Birmingham, November 2015.

Chapter 2

HCI Control Methods for Aerial Robots

"Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."

– Hewett et al. (1996)

This chapter reviews the unimodal and multimodal HCI control interfaces that are being used in the control of small multirotor UAVs. The unimodal interfaces were classified into five categories - agents, electromechanical, vision, bioelectrical, and speech control interfaces. Different multimodal combination of speech and gesture interfaces were also presented. The limitations of all the interface categories presented were discussed. A concise merit and demerit summary of each interface category was developed and presented at the end of this chapter.

2.1 HCI Control Interfaces

This section introduces the unimodal interfaces discussed in Section 2.2 - Section 2.6. Multimodal interfaces were discussed later in Section 2.7. As shown in Figure 2.1, the unimodal aerial robot interfaces discussed in this section are grouped into five categories: agents, electromechanical, bioelectrical, vision, and speech. As a background to general HCI interfaces, a brief historical perspective was first presented.

The development of most of the notable human computer control interfaces in use today can be traced to the advancement of electronics and computer technology in the mid-twentieth century. Initial human computer interactions were based on mechanical toggle switches. Then came

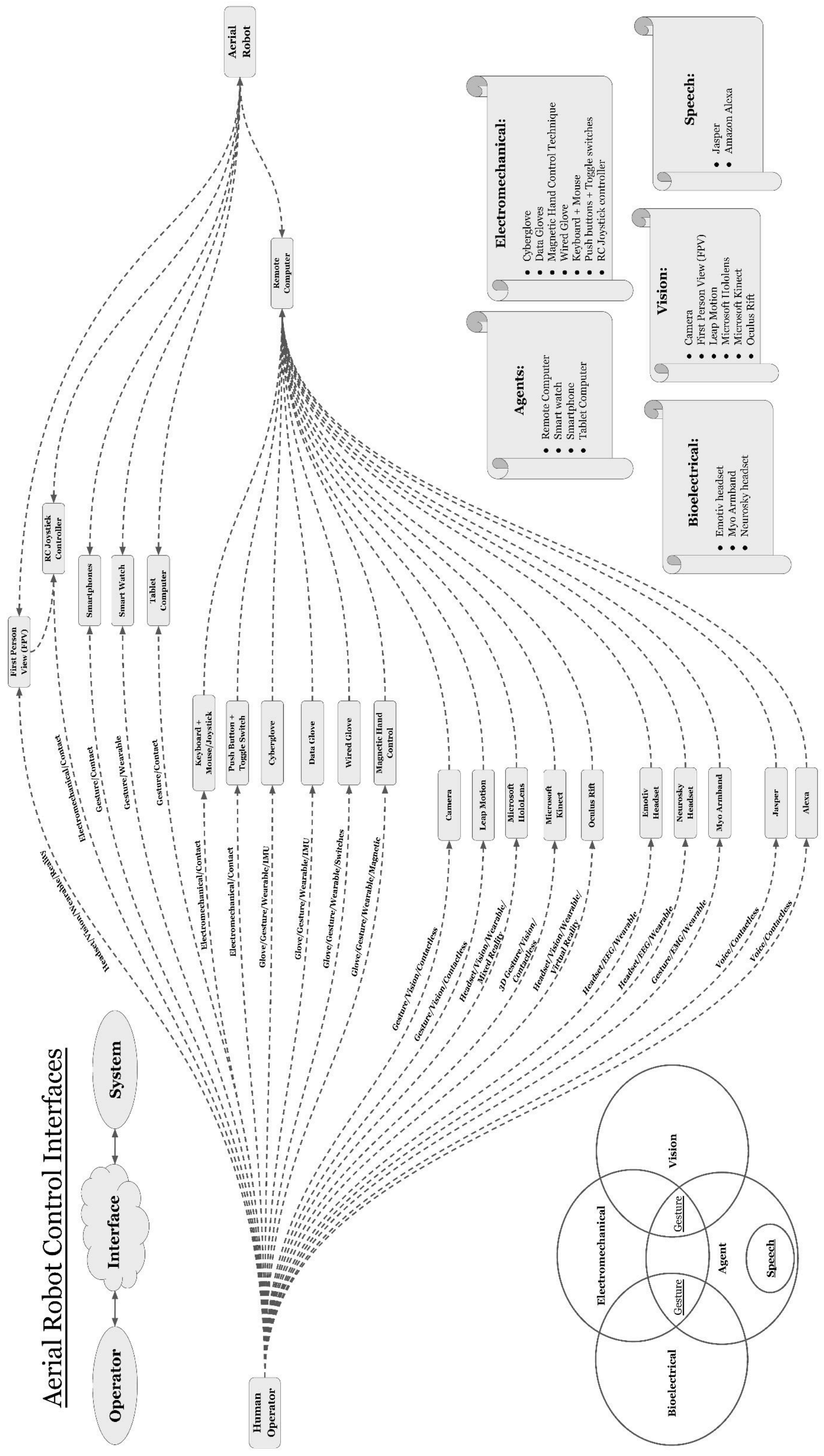


Figure 2.1: Aerial robot control interfaces.

the electromechanical relay switches - from which the electronic keyboard, a set of well-spaced and orderly-arranged electromechanical push-button switches, patterned after the mechanical typewriter keyboard, was developed; and it is still being used today. The light-pen was developed around 1954 to work alongside the keyboard as a graphical input interface (Myers, 1998). It had the form of a light-sensitive wand allowing computer operators to draw or point to displayed objects on a computer's CRT (Cathode Ray Tube) monitor. The first computer joystick was developed around 1962 at MIT along with Slug Russel's video game "Space War", which was considered to be the first graphical video game (Myers, 1998). The mouse was later developed at Stanford Research Laboratory around 1965 as a cheap replacement for the light-pen (Myers, 1998).

Research into gesture recognition and control can be traced back to the light-pen era, where attempts were made at mapping certain light pen gestures to specific Sketchpad program functions (Myers, 1998). Such 2D application of gesture has evolved over the decades. Today, many commercial 3D CAD software, use gestures performed with the aid of the computer mouse, touch screen surface, magnetic and acoustic field manipulation, and visual gesture interaction. Also, virtual reality can be traced to this era, with Ivan Sutherland (Sutherland, 1968) at Harvard in 1965, Tom Furness (Wright-Patterson AFB), and Myron Kruegers (University of Connecticut) pioneering research works; which were later followed by the study of force feedback by Fred Brooks (Brooks, 1977; Brooks et al., 1990) and Henry Fuchs groups at the University of North Carolina in 1971 (Myers, 1998). Natural language and speech recognition for human computer interaction can be traced back to research conducted at CMU, MIT, IBM, AT&T, and Bell Labs in the 1950s, as presented in Anusuya and Katti (2009) historical survey.

Although prior research into the development of human computer interfaces seems to have been largely successful, researchers have continued to investigate the possibilities of developing new interfaces or improving existing interfaces. The aim is often to make these interfaces more natural, intuitive, effective, and efficient. But human-computer interaction is a multi-disciplinary field. It is situated at the intersection of several science and engineering fields such as computer, control, electronics, mechanical, systems design, and bioengineering. It draws on concepts from behavioural science, psychology, ergonomics studies, human factors, and several other fields of study (Preece et al., 2015). Hence the need for these researchers to be either familiar with these fields themselves or work in teams composed of members familiar with the fields of interest. Additionally, the progress in human-robotic interaction and human-machine interaction relies on the successes and progress made in human-computer interaction research. This is because machines and robotic systems rely on computers for their fundamental control operation. In order words, computers translate humans control intention into machine or robotic operations.

2.2 Software-Agent Computing Devices

As presented in the previous section, robotic systems such as the aerial robot, rely on computers for their operation. If these computers could be directly accessed, the control operations of the robot could be modified, eventually altering the behaviour of the robot. In this section, the robot's computer is considered to be connected in a network (through Wi-Fi, Bluetooth, ZigBee, etc.) with another standalone computer (the remote agent), which may be able to modify, update, or specify new control objectives to the robot's computer.

According to Nwana (1996), an agent can be referred to as a software or hardware component that is capable of acting (in an exact manner) on behalf of its user, in order to accomplish a specific task. Agents, in the context of this chapter, refers to intermediate intelligent or smart systems capable of 1) being pre-instructed or programmed by the operator, 2) processing high-level programs into low-level control instructions that can be 3) relayed to an aerial robot, with the aim of providing continuous guidance in order to accomplish the operator's pre-defined task or objective. The most commonly used aerial robot control agent as described in this section is a remote computer system, which could be running a Windows, Linux, or Mac operating system. Other popular agents include the tablet computers (such as Apple's iPad Pro and Samsung's Galaxy Tab S5e), Smartphones (such as the iPhone Xs and Sony Xperia 10), and Smart watches (such as the Apple Watch 4 and Samsung Gear S3).

In addition to the remote computer being an independent agent, it can also act as an intermediate processing/relay terminal for many other control interfaces, as shown in Figure 2.1. The remote computer can be pre-programmed for waypoint navigation. It can also accept electromechanical input and can also provide a visual feedback of the aerial robot collected data and health status. The tablet computer is often powerful enough to operate complex programs like the remote computer. It can also provide waypoint navigation, electromechanical contact switch, touch screen, and accelerometer-based gesture interaction. In addition to this, it can also provide on-screen visual and haptic feedback of telemetry data, status, and other information. The smartphone is similar to the tablet computer but smaller in size and form factor. It can perform similar functions as the tablet computer although it has a smaller screen for visualising telemetry data, status, and other information.

The smart watch is an attempt to miniaturise the smartphone. Control using the smart watch could be relayed through the smartphone as demonstrated by Mark Ven, the PVD+ co-founder, as shown in Figure 2.2. Gesture control is achieved by collecting and processing the smart watch's IMU sensor data. Processing could be relayed through a smartphone, tablet computer,

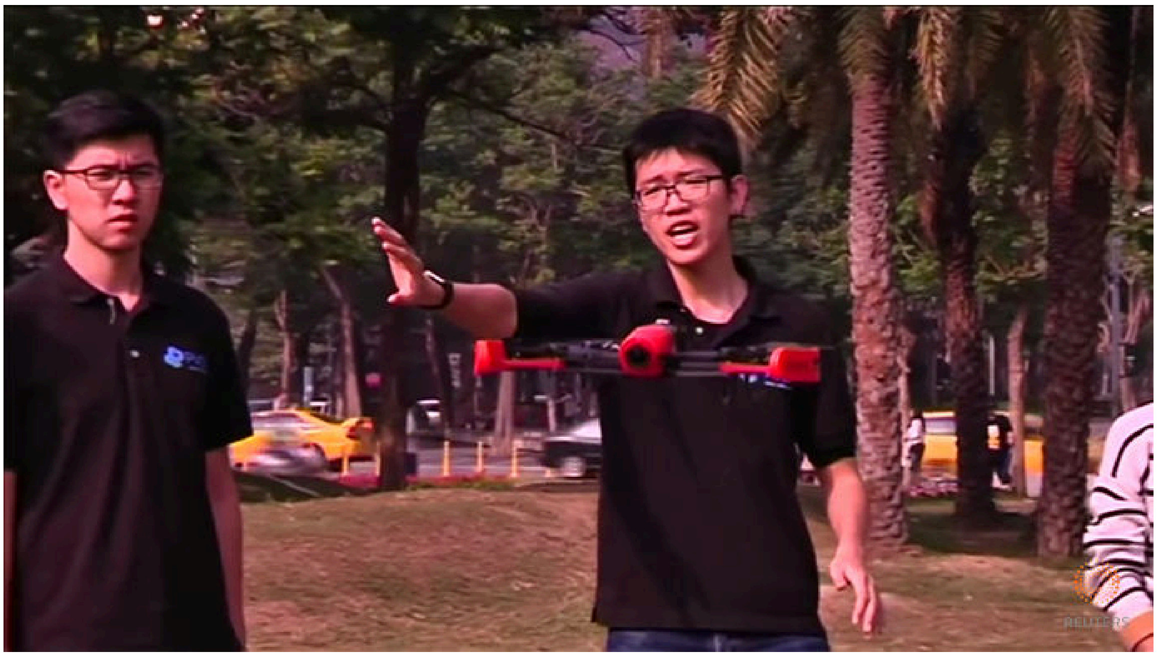


Figure 2.2: Controlling the AR Drone via hand gestures, using an Apple Watch Reuters (2015).

or a remote computer. At the time of writing this chapter, the smartwatch is not capable of processing these data, hence not capable of directly controlling the aerial robot on its own without additional hardware support. However, unlike the other agents, the smart watch has the additional advantage of being a wearable technology.

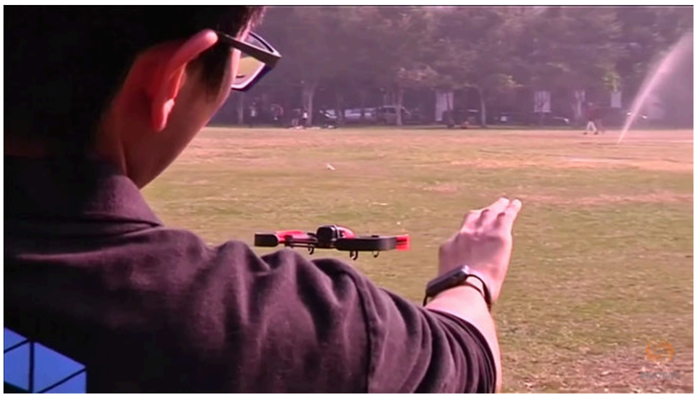
2.2.1 Remote workstation graphical interface example

In Ramchurn et al. (2015) human-agent collaboration with multiple UAV, a supervisory control graphical interface was developed for use on a remote computer workstation. Cavett et al. (2007) graphical interface was also designed to be used for controlling a UAV from a remote computer workstation. These graphical interfaces were designed to lower cognitive workloads and improve performance from planning to task execution. In addition to accepting input controls, which are processed by the remote workstation, and passed on to the UAV for execution, graphical interfaces also provide visual feedback which are useful for non-line-of-sight application scenarios.

2.2.2 The Apple Watch example

The Apple Watch, shown in Figure 2.3a, was developed by Apple Inc. and was first available in 2015, in four versions, each having 8 GB storage capacity. It relies on a wireless connectivity with an iPhone, which could be via Bluetooth or Wi-Fi, to perform many functions. It provides haptic feedback using a linear actuator called the "Taptic Engine", which could be triggered from certain apps. The watch is equipped with IMU sensors, infrared and visible-light LEDs and photodiodes. Its operating system is the Apple's WatchOS.

(a) Apple Watch



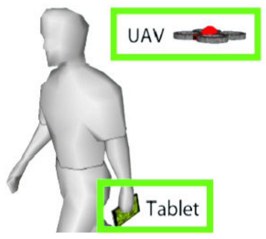
(b) Mark Ven Demo

Figure 2.3: The Apple Watch released April 2015 (Apple, 2015) and Mark Ven demonstrating apple watch hand gesture control on a Parrot AR Drone in December 2015 (Reuters, 2015).

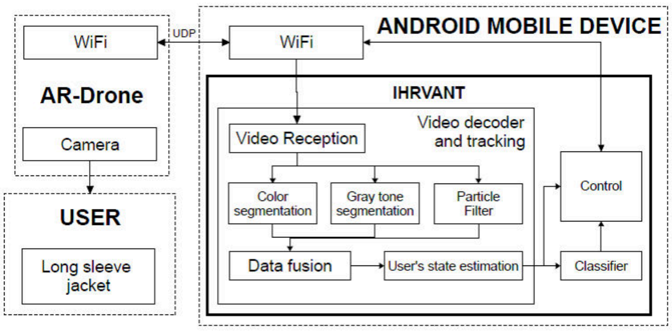
In Figure 2.3b, Mark Ven and his PVD+ team, developers from the National Chung Hsing University, Taiwan, demonstrated the use of the Apple watch in the control of a Parrot AR drone quadcopter. The researchers at PVD+, a 2013 research spin-off of the National Chung Hsing University, Taiwan, developed a software algorithm, which they called 'Dong coding', to convert the Apple Watch into a hand gesture remote controller for a small multi-rotor aircraft (Reuters, 2015). Ven et al successfully demonstrated their control technique at Taichung City, just eight months after the Apple Watch was first available. As Mark Ven explained to Reuters, the Apple Watch interprets his hand's directional gestures, and sends a corresponding direction control signal to the Parrot AR Drone 3.0, which was being used in their demonstration (Reuters, 2015).

2.2.3 Android Tablet device example

Soto-Gerrero and Ramrez-Torres (2013) had also investigated visual gesture control communication for a small UAV. In their work, the authors had utilized a tablet device, running the Android OS, rather than a Windows or Linux based computer, for image processing and other computational workload. In their research, the user carried the tablet as shown in Figure 2.4. The camera on the Parrot AR drone captured the user's image. This was then transferred over Wi-Fi to the Android tablet, where the image was processed and the gesture - extracted, interpreted, and converted to a control signal, which was then communicated back to the UAV. The limitation of this method is that the user would have to carry a control device, the tablet in this case. Could this processing be performed on-board the UAV? Could this be a more convenient, natural, and perhaps intuitive approach? One of the reasons given for this user-UAV-tablet implementation by these researchers was the limitation of the battery power of the tablet. Usually, most small UAVs do not have enough battery power to last more than 20 min, whereas most tablets have enough power to last 8 hours on intensive processing task as performed in their work.



(a) Usage Description



(b) Functional Block Diagram

Figure 2.4: Soto-Gerrero and Ramrez-Torres (2013)'s User-UAV-Tablet interoperability physical & block diagram.

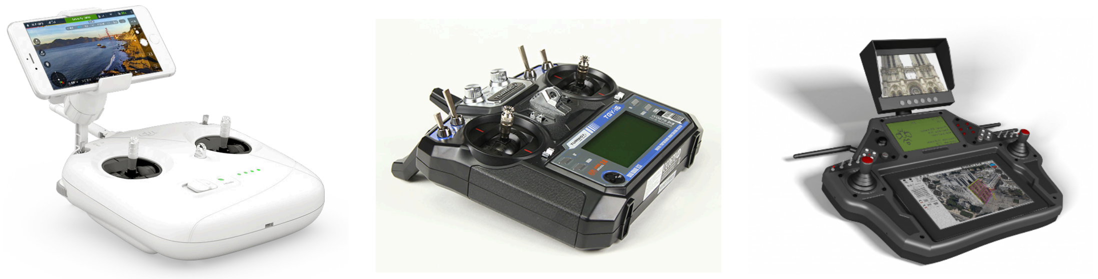
For their hardware implementation, these researchers had used a tablet with an NVidia Tegra 3 quad core mobile processor. For their software implementation, they discarded the Parrot client side API for their IHRVANT java-based native image processing code. Figure 2.4b shows the functional block diagram of their IHRVANT API system.

2.3 Electromechanical

Electromechanical interfaces describe a class of devices that use mechanically actuated switches in controlling the flow of current in an electrical circuit. This regulation of the flow of current is used to signal control intentions to a computer, machine, or robotic system. The most popular electromechanical control device for an aerial robotic system is the RC joystick controller. Other electromechanical devices include the keyboard, mouse, push buttons, toggle switches, Cyber glove, data glove, wired glove, Shift thumb drone controller (Lavars, 2016), IMU HandMagic device (Calella et al., 2016), and magnetic hand gesture device (Ma et al., 2011).

2.3.1 RC Joystick Controller

The most common electromechanical HCI interface controller for small unmanned aircraft is the gamepad-like RC (Radio Controlled) joystick-type controllers shown in in Figure 2.5, which has two control sticks, a few toggle switches, and trimmers. Each of the control sticks perform two of the aircraft's four fundamental operations of throttle, roll, pitch, and yaw control. The trimmers are used to fine tune and balance the two joysticks in their four axes. The toggle switches are used to provide additional channels for other custom control functions such as activating self-stabilizing mode, activating on-board camera recording, and releasing a payload. The RC joystick controller is the most popular multi-rotor control device. It is widely considered to be both efficient and effective, and probably the best controller designed so far, for a multi-rotor aircraft whose autonomy level is within tier-one of the nCA autonomy scale.


(a) DJI (2015b) Phantom 3S remote controller

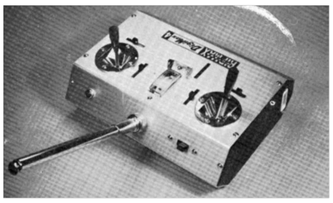
(b) Turnigy (2014) TGY-i6 AFHDS transmitter

(c) VDCI Evo IV ground control station Orzea (2015)

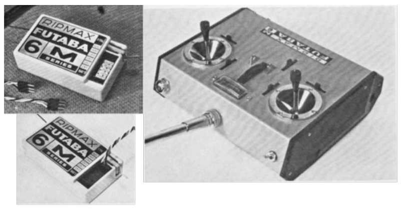
Figure 2.5: Some RC joystick controllers.

RC joystick controllers have been around for over half a century, and in a form similar to what they are today. The RC joystick controller used to be formally called "Radio Modeller (RM)", a term used to describe its original function for flying model aircraft. The Futaba corporation founded in 1948 started manufacturing RC controllers for model planes since 1962 (VRHC, 2013). A line up of their RC controllers from 1970 - 1983 is given in Figure 2.6 and Figure 2.7. Comparing these with their most recent controller in Figure 2.8, may suggest that progress in terms of innovation may be reaching its peak, as the evolution in its physical design form have remained relatively static. However, the underlying technology has continued to improve from analog to digital, AM to FM, PCM, PPM, PWM, FHSS, telemetry feedback, haptics feedback, GPS, etc.

Figure 2.9 shows an image of a Turnigy RC joystick controller, indicating its two joysticks operating in mode 2, toggle switches, push buttons, variable potentiometers (trimmers), and a screen for telemetry information. The joystick can be conceptualised as a two-dimensional

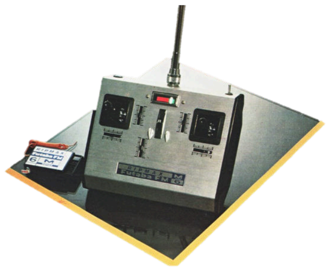


1970 – 4 channel Futaba RC

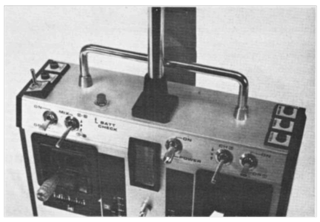


1974 – 6 channel Futaba RC controller

Features
- Amplitude Modulation (AM) radio propagation technology



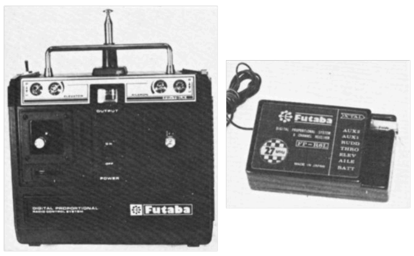
1978 – 6 channel FM Futaba RC controller



1979 – 8 channel FM Futaba RC controller

Features
- Frequency Modulation (FM) radio propagation technology

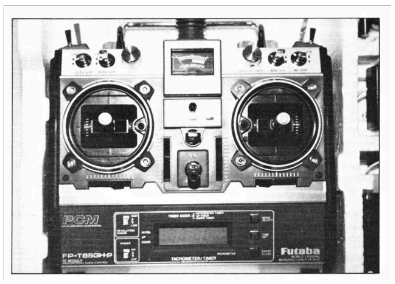
Figure 2.6: Futaba RC Controllers in the 1970's.



1980 – 6 channel 27 MHz Futaba FP5LK RC transmitter and receiver

1982 – 6 channel 35 MHz Futaba RC controller transmitter, receiver, and servos

Features
- Plastic casing
- Light weight
- Aesthetics

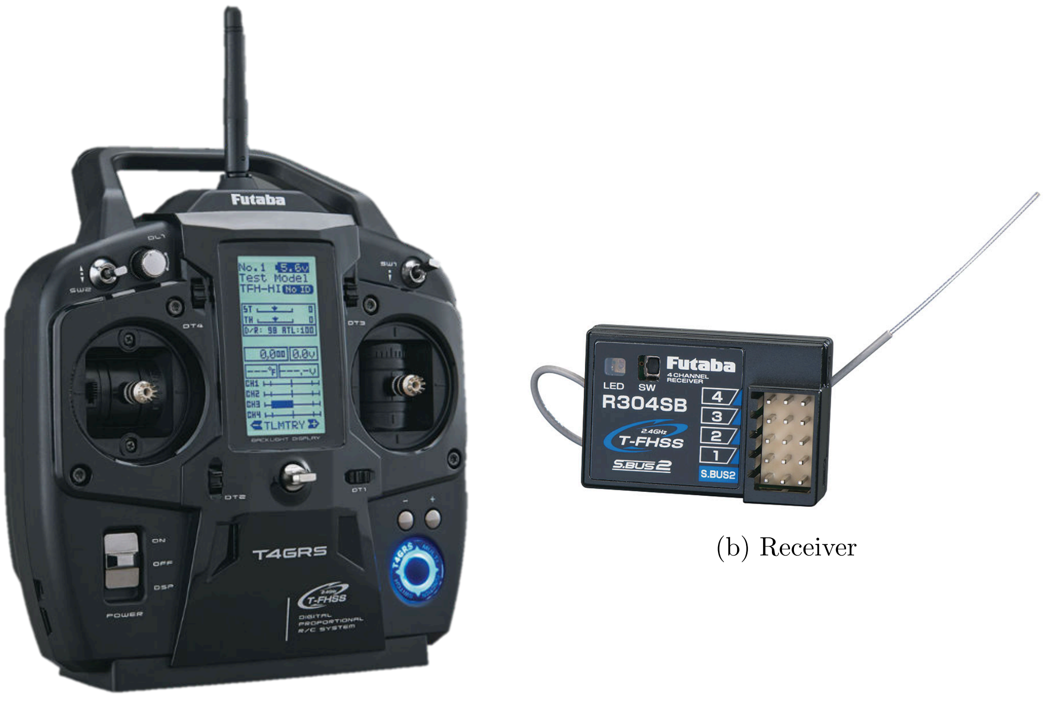


1983 – 8 channel FM Futaba FP-T8SGH RC controller

Features:
- Pulse Code Modulation (PCM) radio technology
- LCD display screen

Figure 2.7: Futaba RC Controllers in the 1980's.

potentiometer (continuous variable resistor) that generates an analogue voltage divider circuit in each dimension. Each dimensional axis is mapped to a control (e.g. thrust) with a continuous range of intensity/values bounded by two extreme levels. The RC joystick controller, unlike most other electromechanical devices, does not require additional or intermediate processing by a remote computer. It is capable of interacting directly with the aerobot's flight controller.



(a) Transmitter

(b) Receiver

Figure 2.8: FUTK4220 4GRS 2.4 GHz 4 Channel Computer Radio System with telemetry (Futaba RC, 2014).

Additionally, it may be augmented with a FPV (First Person View) headset, a smartphone, or a tablet computer to provide visual image data from the aerobot's camera. Telemetry data could also be feedback to the controller via the smartphone, tablet, or a small screen, installed on the controller. It can also be augmented to provide haptic feedback of flight turbulence.

An extension of the portable handheld joystick controller is the cockpit-like workstation that combines both joysticks and keyboards for the control of single, multiple, or large-sized fixed or multi-rotor UAVs, such as the Raytheon's UAV control cockpit shown in Figure 2.10. Although this setup is often used for large fixed winged drones, it could also be configured for controlling, monitoring, or supervising multiple small UAVs. In fact, in this configuration, the electromechanical HCI interface ability can be enhanced by the remote workstation, by communicating operator intentions, receiving feedbacks, logging data for immediate or later analysis. In this configuration, waypoint mapping control method could be an effective method of completing missions. This could be pre-programmed and updated in real-time.

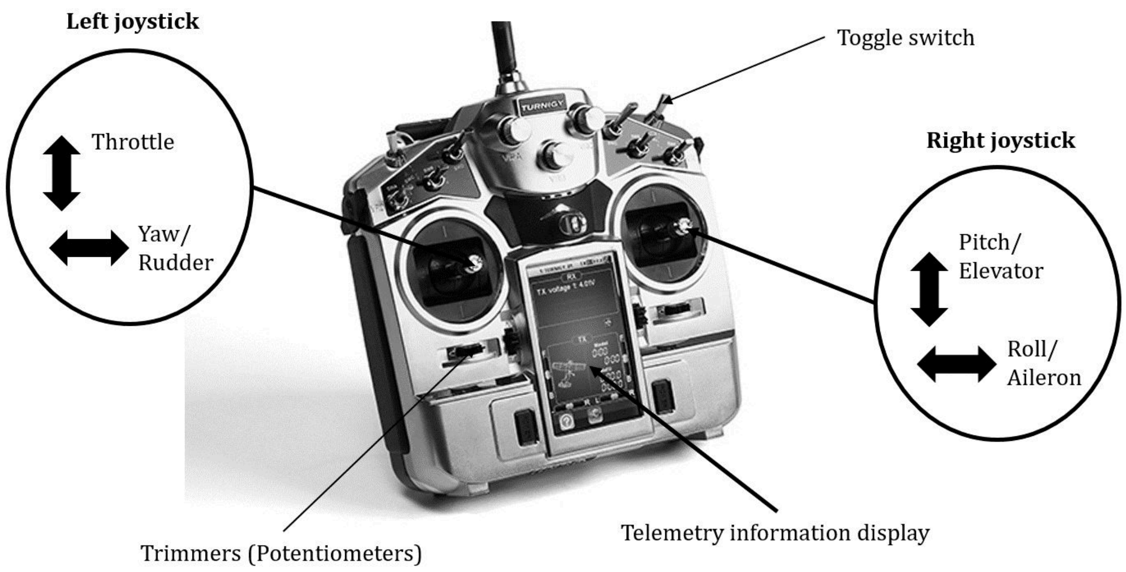


Figure 2.9: 10-Channel Turnigy TGY-i10 (mode 2) telemetry RC joystick controller (HobbyKing and Turnigy, 2015).

2.3.2 Haptics

Haptics is a form of interaction via touch. Haptics "pertains to sensations such as touch, temperature, pressure, etc. mediated by skin, muscle, tendon, or joint" (Brooks et al., 1990). In computer control applications, it occurs via the application or reception of tactile sensation. This touch stimulation results from electromechanically generated forces, vibrations, or motions. Haptic systems often utilize tactile sensors to measure the force applied on an interface by a user, and may provide feedback to alert the user via vibrations of variable intensities. Haptics is also used in providing force feedback to allow operators of heavy machinery 'feel' certain situations such machineries are not able to handled, as in excavators hitting tough large rocks. Such early signals could protect the machines from being damaged. Perhaps this could be applied in providing real time wind gust condition of a UAV to its operator?

Notable research work in haptics interaction includes Noll (1975) early tactile man-machine communication system developed at Bell Telephone Laboratories, Inc. Recent works includes Cirillo et al. (2016) research on new sensorized flexible skin used to enhance safety and intuitiveness in intentional and non-intentional physical human robot interaction (HRI) contacts. Haptics feedback can be found in mobile devices such as smartphones, tablet PCs, and smart watches, as subtle vibrations to alert or notify the user of certain event's occurrence. An example of the application of haptics in robotics is the Shadow Hand developed by Shadow Robot (2013). It was developed to be sensitive to touch, pressure, and position. It was designed to reproduce

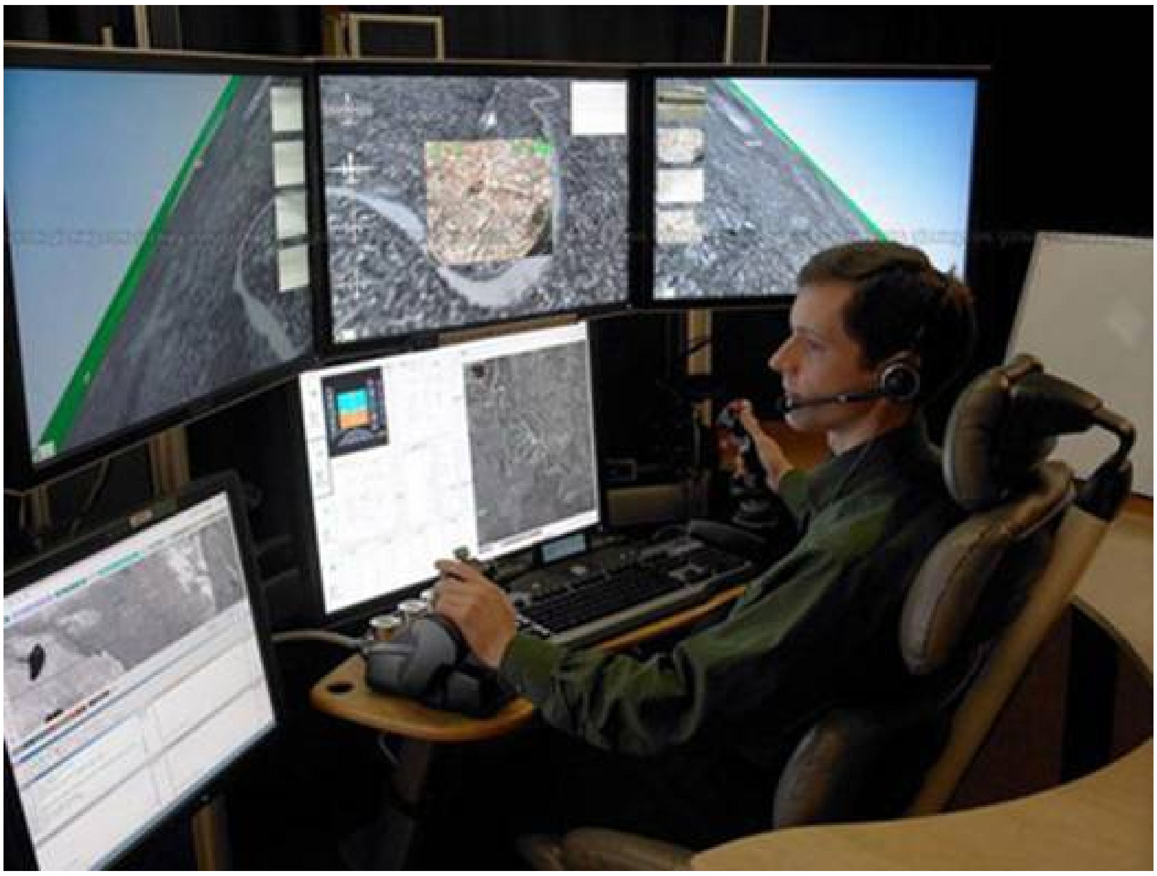
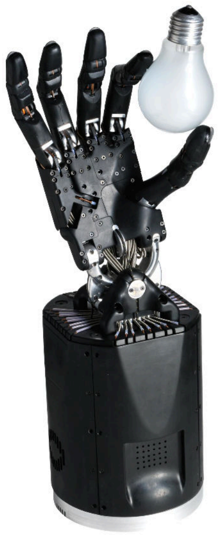


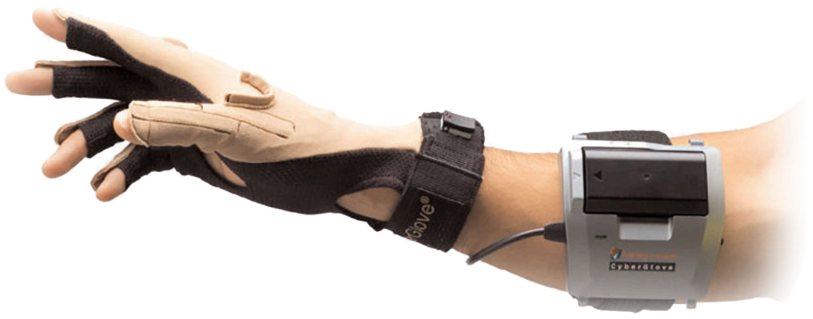
Figure 2.10: Raytheon's UAV control cockpit (Raytheon and Barnard Micro Systems, 2006).

the strength, delicacy, and complexity of the human grip. According to Shadow Robot (2013), the developed tactile sensitive robotic hand can be integrated with the haptic Cyber Glove II (Cyber Glove, 2009), both shown in Figure 2.11

Some researchers are investigating the application of haptic feedback in holographic interaction (Monnai et al., 2014; Makino et al., 2015). These researchers aim to make users interact with and 'feel' virtual objects as though they were real-world objects; without using haptic gloves, by using just their bare hands, via some airborne ultrasound tactile display concept. Tactile interaction could potentially be used in stealthy HRI interaction (Hutchins et al., 2009; Lackey et al., 2011). Haas (2007) suggested the use of multiple tactile cues (such as temporal rhythm, spatial trigger location, frequency of impulse, etc.) in providing multiple feedback information. In addition to the applications of haptics in robotics, it is used specifically in medicine to guide and provide feedback during telepresence surgeries, in entertainment to heighten the user experience as found in gaming and movies, and is a key component of the evolving field of wearable textile technologies.



(a) Tactile Hand



(b) Haptic Feedback Glove

Figure 2.11: Tactile sensitive robot hand (Shadow Robot, 2013) integrate-able with the haptic feedback gloves (Cyber Glove, 2009).

2.3.3 Data Gloves

Data glove, Cyber glove, or Gesture glove are glove-like, hand-wearable, computer input control devices. They are often fitted with electronic inertia sensors used in determining the position and orientation of the fingers and arm. An accelerometer sensor is used to measure hand gesture orientation. A gyroscope sensor may be used to measure the rate at which such hand gestures are formed, hence increasing the vocabulary of possible hand gestures. To further increase the gesture vocabulary, some gesture gloves may have switches that can be triggered by specific hand gestures. These gesture gloves require processing on a remote computer. They could be wirelessly connected to the computer using wireless technologies like Bluetooth or simply connected via wires. Unlike the IMU-based Cyber-glove and Data-glove, Wired-glove is based on finger/hand motions triggering switches embedded along the wearable glove structure (Renitto and Thomas, 2014). Similar to some RC joystick controllers, some of these wearable gesture devices such as the Cyber Glove (2009), are capable of providing haptic feedback.

Figure 2.12 shows the AcceleGlove, an open-source data glove which can be programmed for controlling robotic systems such as a robotic arm, UGV, or a small UAV (Grifantini, 2009). The accelerometer sensors installed on the fingertips of the AcceleGlove is used to determine the position and 3-D orientation of the palm and its fingers, which is then used in controlling a robot. The AcceleGlove was originally developed by Hernandez Jose et al. (2002) at George Washington University, to interpret the American Sign Language. The success of the Accele-

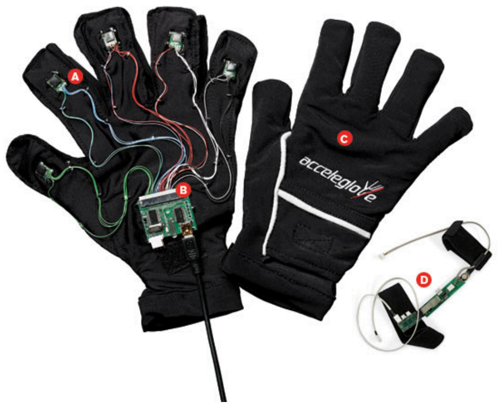


Figure 2.12: AcceleGlove - a programmable open-source data glove (Grifantini, 2009).

Glove arose the interest of both the institute of health and the military; with both organizations sponsoring HCI researchers into investigating the possibility of adapting this glove for physical therapy in healthcare, and for the recognition of military hand signals in robotic warfare systems, respectively (Renitto and Thomas, 2014). Harris and Barber (2014) designed, developed, and used a similar wireless IMU data glove in recording and classify hand gesture commands for a military UGV robot. Other glove systems includes the x-OSC Glove, Key Glove, Clove 2, MIT Glove Mouse, KITTY, Peregrine Gaming Glove, P5 Gaming Glove, Stanford Thumbcode, and Nintendo Power Glove (Renitto and Thomas, 2014). HandMagic is an IMU based wristband for sensing human hand gesture designed and developed by Calella et al. (2016). Although it is similar to the IMU based gesture gloves, it is only being worn on the wrist, hence does not capture finger motions.

2.3.4 Magnetic Hand Control Technique

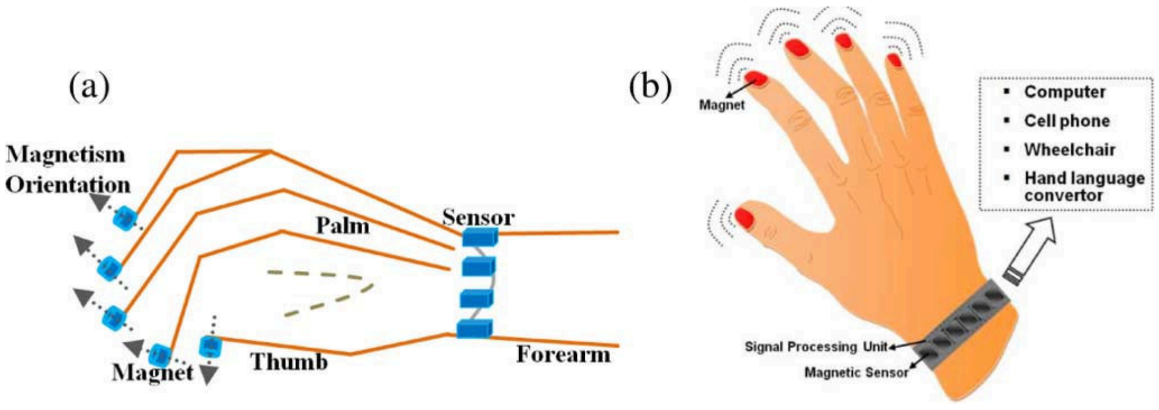
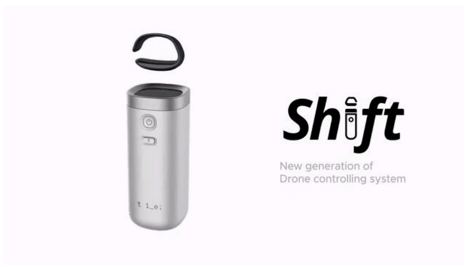


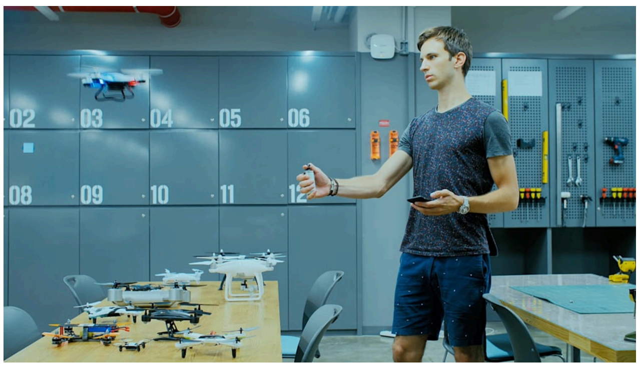
Figure 2.13: Structure of the magnetic hand motion tracking system by Ma et al. (2011).

Another electromechanical control interface of interest is the magnetic hand control method demonstrated by Ma et al. (2011), shown in Figure 2.13. This technique is based on installing small magnets on the fingernails, to generate a magnetic field that can be tracked by a magnetic sensor bracelet worn on the wrist Ma et al. (2011). The location and orientation of the patch of magnets fixed on the fingernail is continuously being tracked by the magnetic sensor on the wrist. A dynamic geometric model of this interaction is developed. The hand posture is then estimated by comparing its computed location and orientation with the developed geometric hand model.

2.3.5 Shift thumb drone controller



(a) Shift thumb controller



(b) Shift thumb steering control

Figure 2.14: Shift device for thumb steering/controlling small UAVs (Lavars, 2016).

The Shift thumb drone controller is a single hand operated interface currently being developed to replace the two hand RC joystick controller. It consists of two major components - a solid cylindrical handheld device and a thumb wearable ring, as shown in Figure 2.14. It was advertised to be more intuitive and compatible with some existing drones (Lavars, 2016).

2.4 Vision

This section describes a set of vision-based HCI interfaces used for communicating control intentions to robots. A camera or some other form of imaging technique is used to capture images, which are then processed (interpreted) to decode the control information. This control information is then passed on to the robot to be executed. Computer vision control techniques often follows the following sequence:

- Capture or acquire image from camera

- Process image: through image segmentation, extraction, perception, recognition, interpretation, etc. in order to decode control information, probably with the aid of a predefined image feature vocabulary or some AI algorithm

- Execute control: through an actuation system that response to the identified control range

This technique is analogous to the human vision interaction system. Its underlying principles emulate the regular human vision principle of operations. An analogy between the human vision system and the computer vision system is presented in Table 2.1.

Vision based control interfaces utilise the techniques of computer vision such as image capture/acquisition, filtering, segmentation, feature extraction, and classification, in identifying the operator's control intentions. Image acquisition is usually performed with the aid of a regular camera, thermal camera, infrared camera, or some other specialised imaging system. Infrared thermal imaging cameras form images using infrared radiation (with wavelengths up to 14 μm), whereas the common video camera uses visible light (with wavelength ranging from $0.4 - 0.7$ μm), for imaging. Due to high processing power requirements, small vision based systems usually require remote processing. The advent of powerful embedded computers are now making it more possible for these processing to be done on small systems nowadays. However, the processing speed of these embedded computers are limited. In the case of remote processing, control throughput is limited by the communication bandwidth for sending captured image data and receiving decoded control information. Therefore, vision based control interfaces usually have low control command throughput, which means they are not suitable for high-rate low-level control function.

Table 2.1: A table of analogy between the human and computer vision systems.

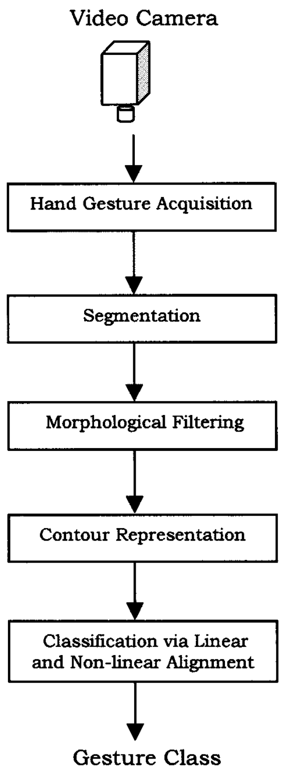
S/No	Human	Computer
1	The human uses the eye for capturing images	The computer uses some type of camera: regular camera, infra-red camera, x-ray camera, radar, etc.
2	The human brain is responsible for processing and interpreting the images captured by the human eye	The computer's CPU processes and interprets the image captured by the camera
3	The brain triggers the human to respond to vision	The computer triggers the system's response via actuating components
4	The human eyes imaging capability is naturally limited to the visible light wavelength range ($0.4 - 0.7 \mu m$)	The computer imaging capacity could be designed to cover the entire electromagnetic spectrum wavelength ($1 \times 10^{-12} - 100 \times 10^{6} m$)

Some devices in this category used for human aerobotic interaction includes regular standalone cameras, web cameras, stereo cameras, Leap Motion device, Microsoft Kinect. It may also include virtual, mixed, and augmented reality headsets such as the Oculus Rift and Microsoft HoloLens amongst others. Some of these devices are discussed further in the following subsections.

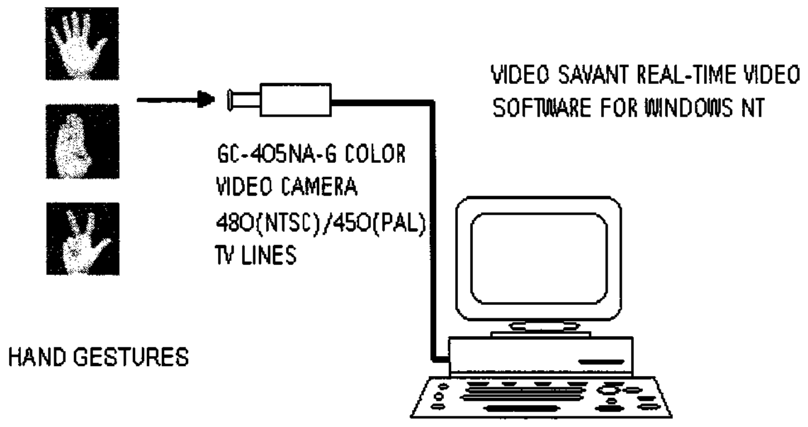
Vision based interfaces, as presented in this research work, may track the human hands, face, or whole-body, in order to identify specific gestures to be mapped to some predefined robotic control operations. Vision based hand gestures are controlled gestures performed specifically with the human hands, in order to communicate the human's intentions. Vision based face gestures are fine controlled movements of facial muscles in order to visually signal or communicate control intentions. Due to the fact that contracting and expanding the facial muscles, over a long period of time, could be tiring and uncomfortable, it is seldom used in robotics by healthy persons. However, this type of HCI interaction finds applications amongst persons with severe disabilities or paralysis of all other parts of their body but their face. However, facial gesture recognition could be integrated into or made to work with other gesture systems. For example, a hand gesture controlled quadcopter used for photography could be queued, to take one's photograph, by a wink-smile. Also, in a search and rescue mission, a multi-rotor aircraft which recognises facial gestures can be able to detect distress and hence perform an appropriate emergency operation. Complex facial attributes can be extracted from 3D facial scans (Blanz, 2007). Certain facial gestures such as smile, neutral face, mouth open, and wink, can be tracked and used in robot control applications. Milanova and Sirakov (2008) and Qing et al. (2008) independently investigated systems to recognise expressions of joy, distress, surprise, interest, frustration, anger, disgust, fear, and a neutral expression among several human observable facial expressions. Vision based whole-body gesture interaction is a method of visually signalling control operations to be executed by a robot, through a set of specified body poses. This gesture control technique differs from the two previously discussed, being that the whole body is used in communication, unlike in the hand and facial gestures - where only a part of the body (hand and face respectively) is used. In this case, the camera captures the whole body pose, processes and interprets the pose, and controls the system's actuators to execute an appropriate response. Soto-Gerrero and Ramrez-Torres (2013) demonstrated this in their upper-body visual gesture control interaction experiment on a Parrot AR drone via an Android-based mobile phone platform.

2.4.1 Single camera systems

An example of a vision based control system designed around a single camera is that developed by Gupta and Ma (2001). These researchers designed and developed a vision-based hand gesture recognition system, which consisted of five steps - hand gesture acquisition, segmentation, filtering, representation, and hand gesture classification, as described in the flowchart in Figure 2.15a. This system basically consisted of a video camera, video capturing software, and a personal computer for capturing, processing, and decoding the hand gesture, as shown in Figure 2.15b.



(a) Operational Flowchart


VIDEO SAVANT REAL-TIME VIDEO SOFTWARE FOR WINDOWS NT
GC-405NA-G COLOR VIDEO CAMERA 480(NTSC)/450(PAL) TV LINES
HAND GESTURES

(b) Gesture Capturing System

Figure 2.15: Vision-based hand gesture system (Gupta and Ma, 2001).

The captured image is first transformed into a grey scale image and the hands are then segmented using a histogram thresholding algorithm. The background and other object noise are then filtered using a morphological filtering approach. After further processing of the resulting gesture contour by comparing with localized contour sequence, and compensating for both similarities and alignment issues, the hand gesture is then classified (Gupta and Ma, 2001). Some vision capture methods may require the human to wear some form of identification markers to aid image feature extraction, segmentation, and detection. But this research work is not interested in those type of methods.

Mohaimenianpour and Vaughan (2018) developed an algorithm to aid the fast detection of users' hands and faces, which was considered robust enough to directly control a UAV in flight, as shown in Figure 2.16.

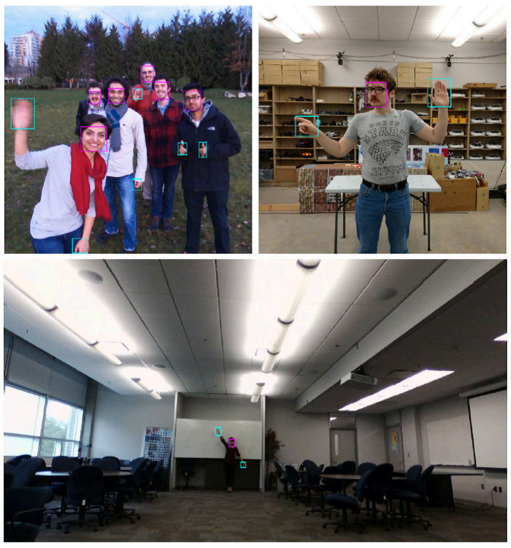


Fig. 1: Typical outputs: hands and faces detected accurately in 15msec per frame in images from a commodity UAV in flight. The user in the lower image is 10m away.

(a) Hand tracking examples

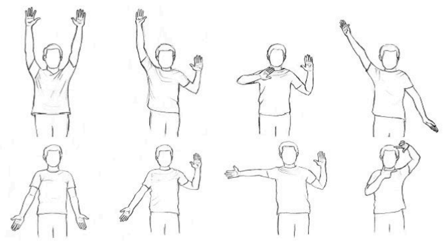


Fig. 2: Our hand gesture vocabulary for Human-UAV interaction. From left to right: ^ Take-off, ↑ Move up, → Move left, ↻ Flip, ∨ Land, ↓ Move down, ← Move right, and 📷 Take a picture.

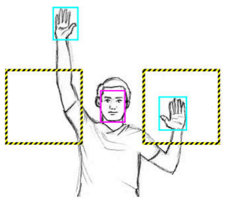


Fig. 3: Our static gesture detection works based on the position of the interaction partner's hands relative to her face. This shows the dead-zone rectangles which are anchored relative to the detected face position and size.

(b) Gesture vocabulary

Figure 2.16: Mohaimenianpour and Vaughan (2018) research on visual gesture control for UAV

Schelle and Stutz (2016) investigated the possibility of visually communicating control to small multi-rotor UAVs via gestures. Their application scenario is illustrated in Figure 2.17. The authors considered that visual communication with aerobots could improve intelligence, surveillance and reconnaissance (ISR) missions, redefine how ground personnel control flying transport vessels, aid search and rescue operations as missing persons could attract attention to themselves and increase their chances of being discovered.

In Schelle and Stutz (2016) scenario: a) an operator on ground tasks the airborne rotary-wing UAV to detect and count persons in a given area by performing a specific set of gestures; b) the UAV perceives those gestures and translates them into gestural command components; c) once the gestural command set reception and processing is complete, the UAV starts its mission and flies to the commanded area to perform the person detection and counting task; d) having

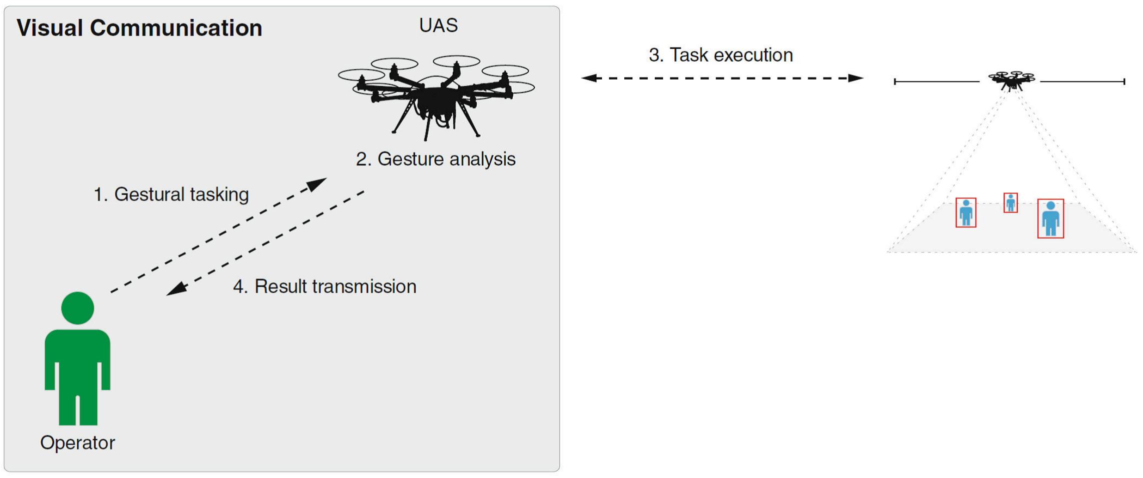
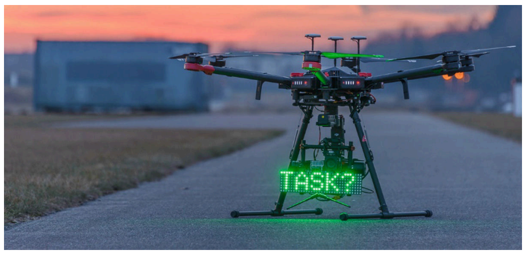
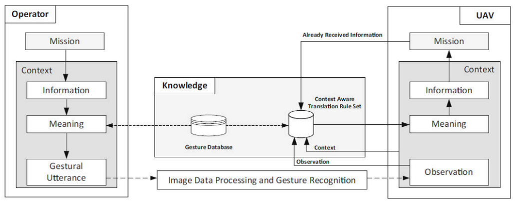


Figure 2.17: Schelle and Stutz (2016) visual communication between an operator and a UAV.

completed this task, the UAV returns to the last known location of the operator and transmits its results visually with light signals or via an LED Matrix (Schelle and Stütz, 2018) as shown in Figure 2.18a, which could also work for this research scenario. By augmenting the UAV speaker output response with the LED matrix display, an on-screen feedback could be provided in the wild. Figure 2.18b shows the block diagram of their visual communication model.



(a) UAV LED matrix feedback



(b) Communication model block diagram

Figure 2.18: A visual communication model (Schelle and Stütz, 2018)

2.4.2 Webcam

A webcam is a video camera designed to broadcast live feeds over the internet. It is basically a single camera system with image frames optimised for efficient communication over the internet. Most webcams are generally designed to operate in low resolution modes such as 480p, in order to reduce the data bandwidth for broadcasting live feeds continuously over the internet. While there are some high resolution web cameras available today, the image quality of webcams are typically lower than dedicated computer vision and broadcast cameras. In Shetty et al. (2016) investigation on controlling a quadcopter via hand gestures, the researchers used a webcam. The captured gesture is processed on a computer system, and the decoded control, relayed to the multi-rotor UAV. Shetty et al. (2016) setup can be described by the Figure 2.19. An Arduino board receives control command from the computer wirelessly and relays this to the KK2.1.5 flight controller board of the multi-rotor UAV.

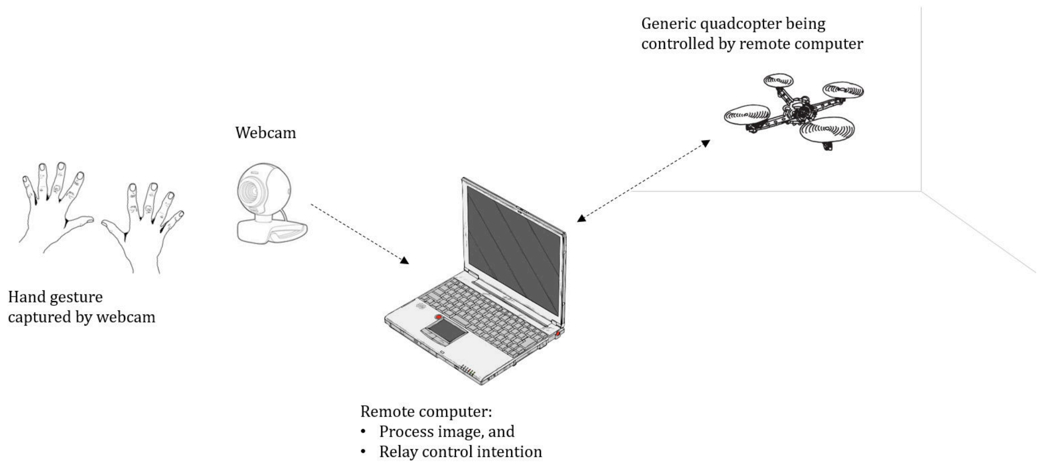

Generic quadcopter being controlled by remote computer
Webcam
Hand gesture captured by webcam
Remote computer:
• Process image, and
• Relay control intention

Figure 2.19: Webcam gesture controlled quadcopter.

2.4.3 Two cameras - orthogonal

The effectiveness of a single camera system could be easily undermined by dynamic environmental objects occluding an operator's gestures. Huang and Chung (2004) developed a method to track and analyse the human body motion gestures using two cameras installed orthogonal to each other as shown in Figure 2.20. By combining images from the two cameras' view, a three-dimensional body gesture can be identified. In addition to this, the occlusion problem in the single camera system is reduced.

In Huang and Chung (2004) method, the extracted human silhouette was pattern matched to a computer generated human model, and then a Hidden Markov Model (HMM) algorithm was used for posture recognition. HMM is a statistical modelling technique which assumes the system to be a Markov process with unobserved hidden states. Such system's outcomes are usually based on states and transition probabilities.

Figure 2.21a shows Huang and Chung (2004)'s skeletal (stick man) model as an approximation of the major human bones structure and joints formations. This was based on the observation that human body part motions result from relative bone movements. Three dimensional (3-D)

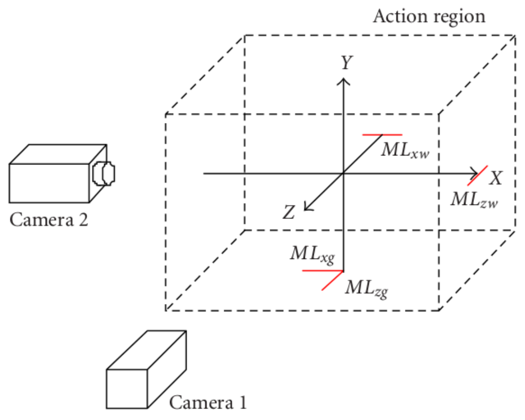
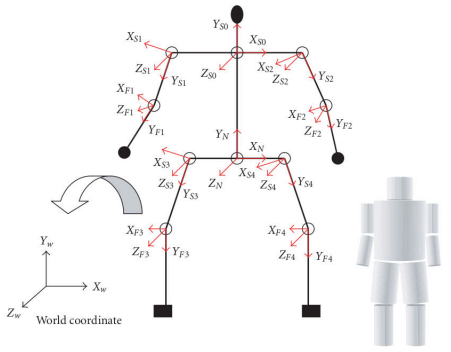
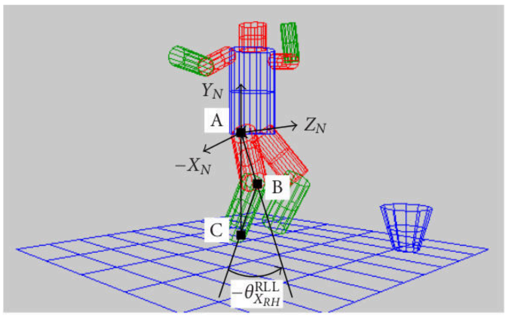


Figure 2.20: Orthogonally arranged gesture capturing cameras (Huang and Chung, 2004).



(a) Skeletal-joint stick-man model



(b) 3-D Computational Model

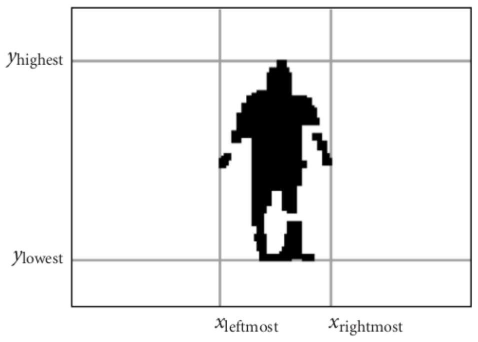
Figure 2.21: Human body models (Huang and Chung, 2004).

shapes or volumes such as the elliptical cylinder shown in Figure 2.21b could also be effective descriptors on which the human model could be based.

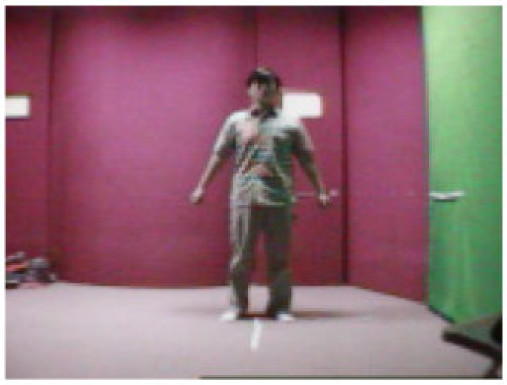
Figure 2.22 shows how Huang and Chung (2004) two camera system works. In this particular test, one camera captures the facade view while the second camera captures the lateral view of the human pose, at the same time. These images can then be processed to extract the whole body control gesture being intended by the human operator for execution by a robotic system.

2.4.4 Microsoft Kinect

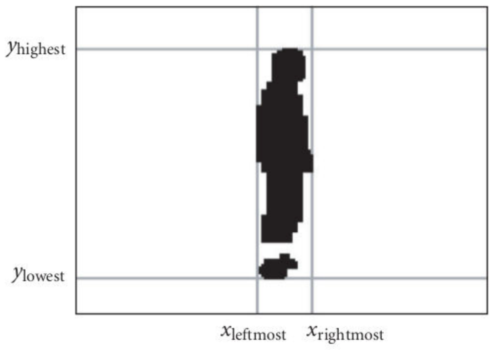
In 2010, Microsoft released a proprietary depth sensing camera system called the Microsoft Kinect, for their Xbox 360 gaming console. In 2012, Microsoft made the Kinect available for the Windows platform. This encouraged the development of robotic systems that rely on the Kinect vision system. Its popularity was centred on its depth sensing capability. Following the success



(a) Facade camera processed image



(b) Facade camera captured image



(c) Lateral camera processed image

(d) Lateral camera captured image

Figure 2.22: Two camera system - capturing and processing image (Huang and Chung, 2004).

of the Kinect for the Xbox 360, Microsoft developed the Kinect v2 (version 2) for the Xbox one in 2013, and also made this available to the Windows platform in 2014. Figure 2.23 shows the Microsoft Kinect version 2. It consists of an RGB camera, an infrared based depth sensor, and a multi-array microphone (Microsoft, 2014). It is capable of full-body 3D motion capture, facial recognition, and voice recognition (Microsoft, 2014). The microphone arrays can also be used for acoustic source localization and ambient noise suppression. The depth sensor works under any ambient lighting conditions and has an adjustable depth sensing range (1.2 - 3.5 m). The Kinect is capable of tracking six persons. It has an 8-bit VGA resolution of 640 $\times$ 480 pixels at 30 Hz frame rate, it is capable of a resolution of 1280 $\times$ 1024 pixel at lower frame rate. The Kinect field of view is 57° horizontal and 43° vertical. The Kinect v2 has a wider field of vision and can track up to 6 human skeletons at once (Microsoft, 2014). It is capable of tracking 25 individual joints of the active player, detect the player's facial expression, heart rate, movement speed, and standard controller gestures.

(a) Kinect v2

(b) Kinect v2 Windows Connector

Figure 2.23: Microsoft Kinect version 2 (Microsoft, 2014).

Sanna et al. (2013) developed a vision-based whole-body gesture capturing system with the aid of the Kinect, as shown in Figure 2.24 and Figure 2.25. In their method, the whole body gesture is captured by the Kinect, processed on a Laptop running the Windows 7 operating system, after which the identified command is communicated to the Parrot AR Drone (a small UAV) via Wi-Fi, as described in Figure 2.24.

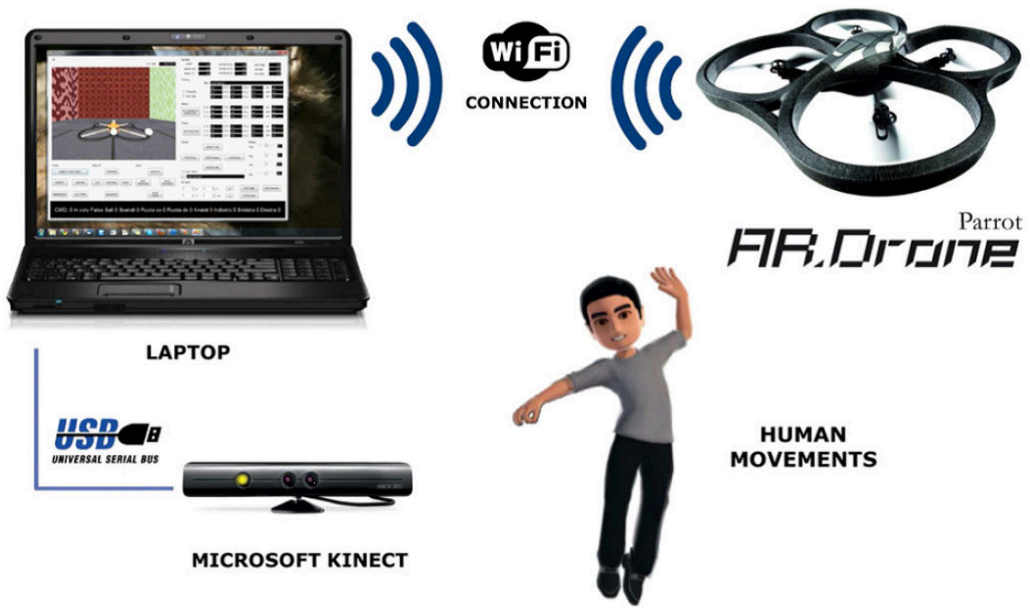


Figure 2.24: Kinect control of a small UAV (Sanna et al., 2013).

Sanna et al. (2013) method requires remote processing by a computer. In their work, they compared the performance of the joystick, iPhone, Kinect, and the AR vertical camera in autonomous mode. They found that the Kinect was more effective at completing tasks. Although, only three commands could be issued in 1 second, due to processing and communication latency, this was found to be better than the AR drone autonomous mode vertical camera's one command in 2 seconds. In other words, they found that remote processing was six times faster than the on-board alternative.

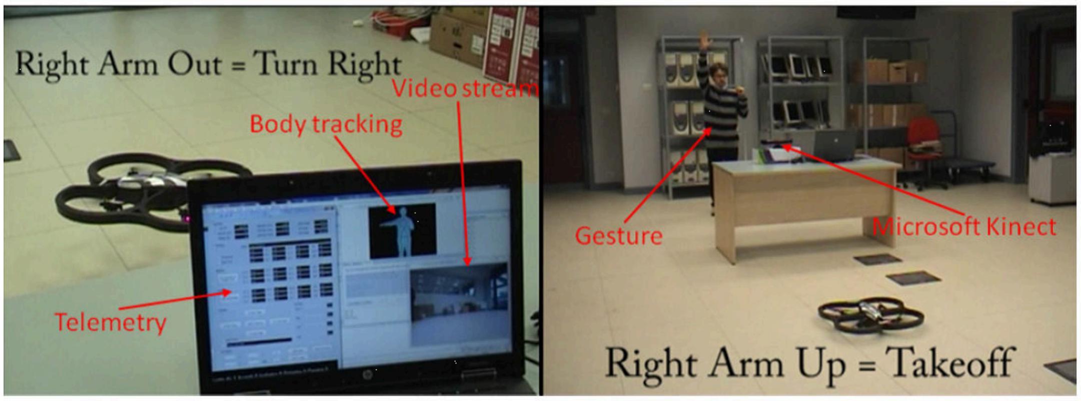


Figure 2.25: Controlling a Parrot AR Drone via the Kinect (Sanna et al., 2013).

Harris and Barber (2014) investigated the use of the Kinect in the control of an unmanned ground vehicle (UGV) called the JRMBot, which is shown in Figure 2.26. The JRMBot consisted of the Kinect, an on-board laptop computer, and other primary control mechanisms. The Kinect was used to capture speech commands, for navigation and obstacle avoidance. It could also have been used to capture and classify hand gesture commands instead of the wireless IMU data glove that they used. The speech and vision information captured by the Kinect was being processed on-board the robot via the laptop computer. In other words, all processing was performed remotely. Although, carrying the Kinect and a laptop computer does not constitute a problem for the UGV, small UAVs are usually designed to fly lighter, longer, and more dynamically; therefore, these additional payload components could make it difficult to achieve such objectives. But with the shrinking size of the computers and vision capturing systems, this may not be an issue for too long.

Ma and Cheng (2016) investigated gesture control on a Parrot AR Drone using the Kinect v2, as described in Figure 2.27. These researchers used NI LabVIEW in programming their system. They suggested that their system yielded a satisfactory performance result, although, the researchers did not provide a measure of their system's performance. In their work, they computed geometrical references using quaternions as shown in Figure 2.28. In their method, they computed the rotational matrix of the arm with respect to the body frame, in order to identify specific control gestures. Their work, just like Sanna et al. (2013), also relies on remote image acquisition and processing.

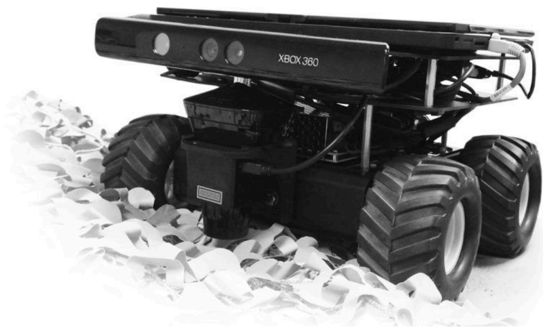


Figure 2.26: JRMBot Platform with the Kinect and a laptop computer installed (Harris and Barber, 2014).

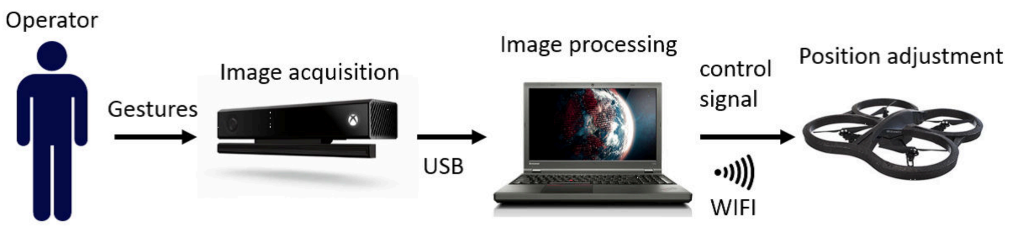


Figure 2.27: An overview of Ma and Cheng (2016) AR Drone gesture control implementation.

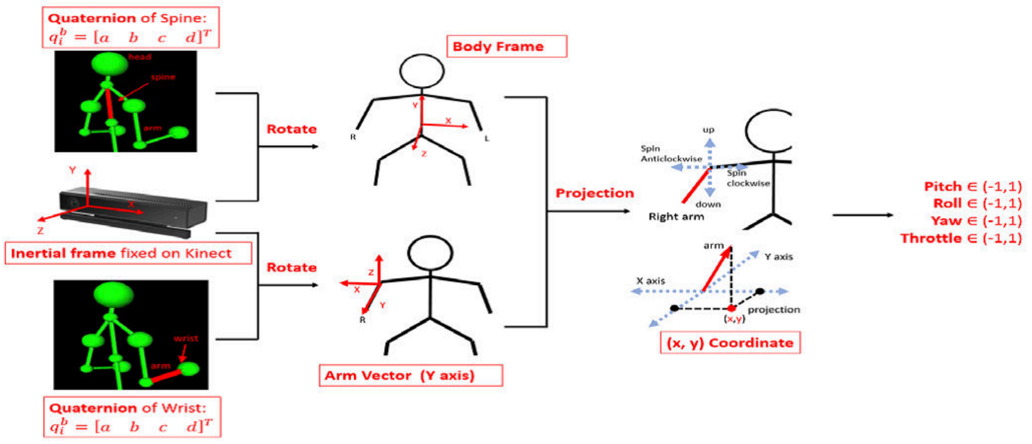


Figure 2.28: Geometric computation and references for visual gestures control (Ma and Cheng, 2016).

In addition to the whole body gesture application of the Kinect, it can also be used for vision-based hand gesture interaction as demonstrated by Nimble VR (2012), formally 3Gear Systems. Nimble VR (2012) developed a finger-precise hand gesture tracking system, which uses two downward facing Microsoft Kinect sensors. This system, which is shown in Figure 2.29, is reported to provide millimetre level accuracy of the hands (Eichhorn, 2012). By using a database of pre-computed 3D images, corresponding to all possible configurations of the hand in the workspace, efficiently sampled and indexed, the hand pose could be determined within milliseconds (Eichhorn, 2012).

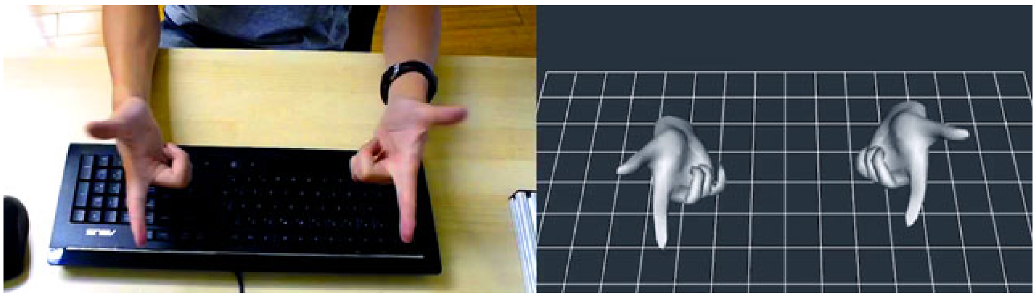
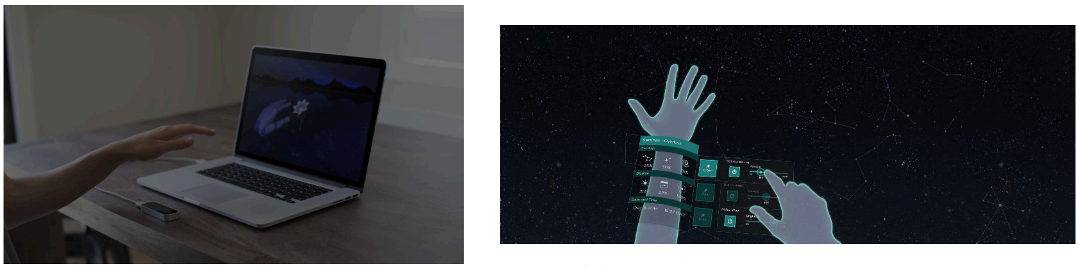


Figure 2.29: 3Gear Systems finger-precise Kinect hand gesture tracking system (Eichhorn, 2012).

2.4.5 Leap Motion Device

The Leap Motion controller, shown in Figure 2.30, is a device that can be used to perform hand gesture interaction within a virtual or augmented reality environment (Leap Motion Inc., 2010a). It consists of three infrared LEDs and two cameras under a black glass (Coelho and Verbeek, 2014).

Figure 2.30: Leap Motion Device (Leap Motion Inc., 2010a).



(a) Leap Motion Setup Connection

(b) Leap Motion Virtual Interaction

Figure 2.31: Leap Motion Device application examples (Leap Motion Inc., 2010a).

By performing some complex spatial mathematical computational analysis of various gesture poses, a program could be written to track the human hand gestures or movements, as shown in Figure 2.31. Coelho and Verbeek (2014) investigated the effectiveness of the Leap Motion device for manipulating three dimensional objects in virtual space. With the advent of mixed reality, the line between virtual reality and real reality is becoming thinner. Hence, what works in virtual space could, in theory, also work in physical space.

In Sarkar et al. (2016) research work, the authors described their implementation of a gesture control system for a small multirotor aircraft. These researchers used commercial-off-the-shelf components: the Leap Motion device and the Parrot AR Drone 2.0 quadcopter. By using the robotic operating system (ROS) framework, installed on both the ARM Cortex A8 processor of the AR Drone and on a Linux based computer, and by utilizing a python script, the researchers were able to perform gesture communication. Figure 2.32 describes their implementation method. The researchers envision scenarios in which such gesture communication could be used in controlling quadcopters during aerial videography or in performing aerial acrobatics stunts. The second scenario on acrobatic stunts may be limited by the image data processing and communication latency for such visual gesture systems. However, a gesture could be used to trigger the execution of a pre-programmed acrobatic stunt.

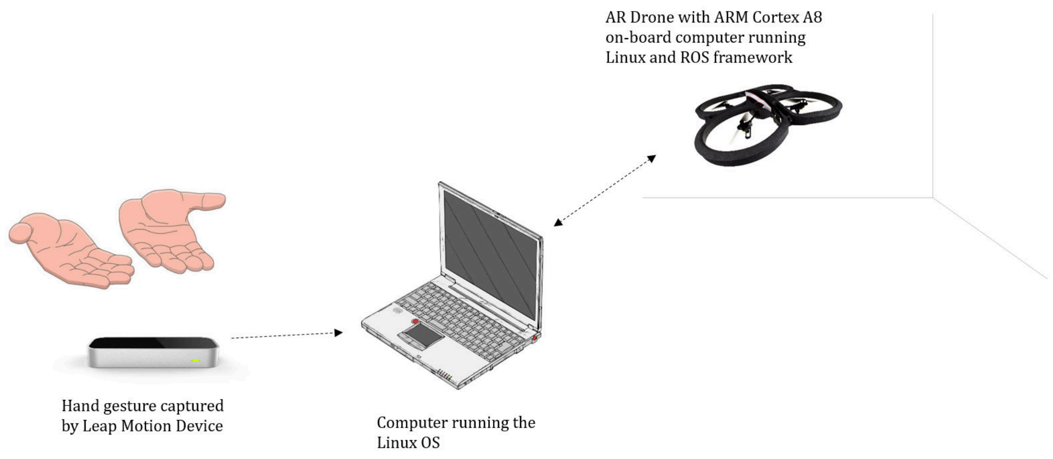
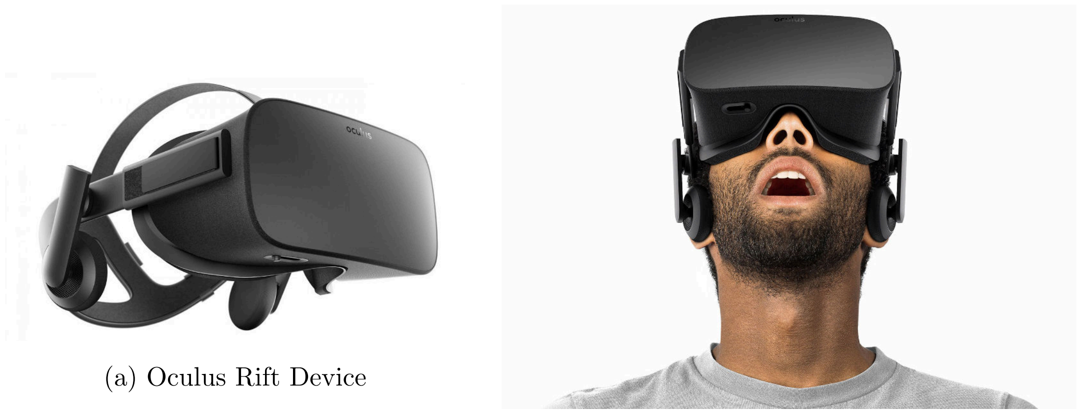


Figure 2.32: Leap Motion Device controlled Quadcopter.

Gubcsi and Zsedrovits (2018) researched ergonomic quadcopter using the Leap Motion controller with the aid of a remote computer relaying control commands to a real IRIS+ drone in an outdoor environment.

2.4.6 Oculus Rift

Oculus Rift, shown in Figure 2.33, is an immersive virtual and augmented reality headset developed by Oculus VR (2016). It is a tethered head-mounted display, originally developed for gaming purpose but is now being integrated into other computer applications. The Rift uses an OLED display technology, with 2160×1200 resolution (that is, 1080×1200 resolution per eye), at 90 Hz refresh rate, and a field of view of 110 deg (Oculus VR, 2016). It has integrated headphones which provides a 3D audio effect, and is also used for rotational and positional tracking. This utilises a positional tracking system, called “Constellation”, performed with the aid of a stationary USB IR LED sensor; used in mapping the room via its infrared and LED lights. Although VR devices such as the Oculus Rift were originally designed to provide feedback of an immersive experience, these devices could also be used for initiating interaction within their virtual environment especially when used in combination with other interfaces such as the Leap Motion device.


(a) Oculus Rift Device

(b) Wearing the Oculus Rift

Figure 2.33: Oculus Rift (Oculus VR, 2016).

2.4.7 Microsoft HoloLens

Microsoft HoloLens is a mixed reality optical head-mounted display smart glass designed to work with the Windows 10 operating system. This is because Windows 10 is the first Microsoft OS platform to support holographic computing with APIs that enable gaze, gesture, voice, and environmental understanding on an untethered device (Furlan, 2016; Microsoft, 2016). Gaze interaction is achieved via gaze tracking built-in sensors that guides the cursor’s hologram selection. Simple gestures can be used to open apps, select and size items, or drag and drop holograms in the real world, as shown in Figure 2.34. And voice commands can be used to navigate, select, open, command, and control apps via Cortana (Microsoft, 2016). This seems to be an extension of the Microsoft Kinect system, as it shares certain underlying technologies such as the 3D depth cameras, microphones, and sensors.

Figure 2.34: Mixed reality HoloLens holographic gesture interaction (Microsoft, 2016).

2.4.8 Other virtual, augmented, or mixed reality devices

Virtual reality (VR) is a computer-generated existence with a rich selection of immersive multimedia experiences often based on a simulation of real world behaviours. Virtual reality offers a visual sense of belonging in an artificial environment; by enabling the user to see and interact in that environment via gaze, touch, or speech. This is often achieved through virtual reality headsets for a more immersive experience, or via computer monitors, as in gaming. Virtual reality finds application in many areas such as education, training, flight simulation, modelling inter-planetary experience, medicine and surgery, video games entertainment, artistic renditions, archaeological exhibitions, architectural designs, and interactive cinema entertainment Kipman (2016). However, there are a few constraints to the wide acceptance of this technology which includes health and safety issues, social issues, conceptual issues, and other philosophical concerns. Motion sickness, user's disorientation in virtual world, computer latency, the complicated nature of headsets, and an unconstrained real world environment are all examples of technical health and safety concerns that could result in a less satisfactory user experience.

In augmented reality, real world objects are overlaid with computer generated graphics. Green et al. (2007) suggested that this could be an effective human-robot collaboration method. While virtual reality involves being completely immersed in a computer simulated reality, augmented reality involves only a partial immersion in the simulated reality. In this case, the users are conscious of and are able to interact with the real world environment.

Mixed reality seems to lie somewhere in-between virtual and augmented reality. It is the "merging of real and virtual worlds to produce new environments and visualizations where physical and digital objects co-exists and interact in real time" (Azuma et al., 2001; Costanza et al., 2009; De Souza E Silva and Sutko, 2009). According to Milgram and Kishino (1994), mixed reality can be considered as a particular subclass of virtual reality that involves a merger of the real and virtual worlds. This seems to be the concept behind the Microsoft (2016) HoloLens. Mixed reality can be considered as the continuum that ranges from reality, through augmented reality, through augmented virtuality, to Virtual reality (Milgram and Kishino, 1994; Azuma et al., 2001).

Some recent VR, AR, and MR devices have been compared and presented in Table 2.2. Virtual, augmented, or mixed reality may also find application in the tele-operation of robotic systems. A multi-rotor aircraft control could potentially be controlled by an operator in a virtual, augmented, or mixed reality. For example, an operator performing control manoeuvres in a virtual cockpit could have real control consequences on a real aircraft via some cross reality design.

Table 2.2: VR headset comparison table.

S/No	**VR Headset**	**Feature**	**Connection**	**Resolution (pixels)**	**Refresh Rate**	**Field of View (degrees)**	**Mass**	**Release Date**
1	Oculus rift (£ 549.00)	Requires PC, Oculus touch controllers, constellation sensors, compatibility with Xbox controller, Tethered to PC, Audio, Oculus Software	HDMI 1.3, USB 2.0, USB 3.0	1080 by 1200 (per eye)	90 Hz	110	470 g	28-Mar-16
2	PlayStation VR (£ 349.99)	Requires Sony PS4 Console, Dual Shock 4 controller, PlayStation Aim, PlayStation Move, PlayStation Camera, Audio + Mic, PlayStation 4 Software	HDMI, USB 2.0	960 by 1080 (per eye)	120 Hz	100	610 g	13-Oct-16
3	HTC Vive (£ 769.99)	Requires PC, Lighthouse tracking system, Audio is not as good as Oculus Rift Audio, SteamVR Software	HDMI 1.4, Display Port 2.0, USB 2.0, USB 3.0	1080 by 1200 (per eye)	90 Hz	110	555 g	5-Apr-16
4	Samsung Gear VR (£ 81.99)	Requires Samsung smartphone, Samsung Gear VR software powered by Oculus	USB 2.0, USB 3.0	1280 by 1440 (per eye)	N/A	N/A	318 g	27-Nov-15
5	Samsung Gear VR 2 (£ 99.00)	Requires Samsung smartphone	USB 2.0, USB 3.0	N/A	N/A	N/A	N/A	N/A
6	Google Daydream View (£ 69.00)	Requires 5-inch or 5.5-inch Android Smartphones, Android 7.0 Nougat	N/A	N/A	60 Hz	N/A	N/A	10-Nov-16
7	LG 360 VR (£ 69.00)	Requires LG G5 smartphone, USB connection to phone	N/A	N/A	N/A	N/A	N/A	N/A
8	Microsoft HoloLens (£ 2,719.00)	Requires PC (Windows 10), 2.3 megapixel widescreen stereoscopic head-mounted display, gestural commands, 2.4 MP Camera, Mixed Reality	IEEE 802.11ac, Bluetooth 4.1 LE	N/A	N/A	N/A	579 g	30-Mar-16 (Development Edition)
9	Huawei VR	Requires Android Smartphones	N/A	N/A	N/A	N/A	N/A	N/A

2.4.9 DJI Spark Gestures

DJI the renowned consumer multirotor UAV manufacturer, in mid-2017, released the DJI Spark drone, a 143 × 143 × 55 *mm* small size, 300 g lightweight, 16 minute flight time, £519 cost UAV (DJI, 2017b,c; Heater, 2017). The Spark was developed to be able to interact with human users via visual hand gestures, iOS and Android based mobile phones/tablets, and the RC joystick controller. The DJI Spark has seven gesture controls - palm launch, start/stop palm control, adjust position, follow, take selfie (picture box symbol), beacon over, and palm land. Figure 2.35 shows a Spark being signalled to take a selfie picture when the primary operator forms a "picture box" gesture.

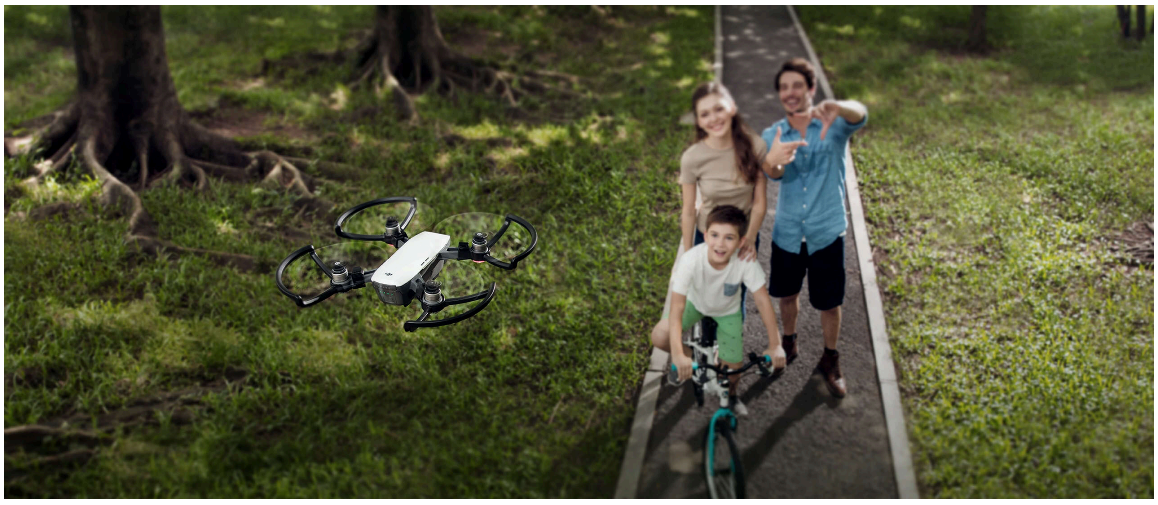


Figure 2.35: Performing the "picture box symbol" gesture to take a selfie using the DJI (2017b) SPARK.

The Spark uses facial recognition in identifying the primary users. A front-facing camera installed on a 2D gimbal, capable of capturing 1080p video @ 30fps and 12 MP photo, is used for capturing gestures, performing the facial recognition, taking selfies, and recording short videos. It uses a combination of IMU, GPS/GLONASS, downward facing camera, and 3D infrared module for positioning and navigation. It is able to navigate in the absence of GPS, common in indoor environment, using only the vision positioning system, which consists of the downward facing camera and 3D infrared module. It uses a 3D sensing system, which consists of a front facing 3D infrared module, for sensing and avoiding obstacles. Its processing system consists of 24 computing cores (DJI, 2017b). The flight controller is built with several safety features, which includes automatically returning "home" when transmission signal is lost or battery level is low, hovering indoors at low altitudes, and sensing and avoiding obstacles on its route (DJI, 2017c).

A 90-minute flying review of the DJI Spark by TechCrunch gives an insight into a typical user's first impression on gesture control drones (Heater, 2017). According to the TechCrunch testers, the gesture control was tough to learn, despite the fact that one of the testers was a seasoned videographer with some drone-flying experience (Heater, 2017); a learning curve that takes more than 16 minutes to get a handle on. According to Heater (2017), there was some noticeable delays between making a palm gesture and the drone performing the corresponding movement. Some erratic behaviours were sometimes observed for simple gestures, with the drone's camera losing track of the pilot in one instance, which then required taking over control through the mobile app interface. The testers also observed that, although the drone size is small, it generates a significant amount of noise similar to the buzz of a lawnmower.

2.5 Bioelectrical

Bioelectrical control interfaces rely on electrical nerve impulses generated by muscular or neurological human activities. These electrical impulses are usually detected with the aid of specialized biomedical instruments and probes. The detected electrical signal may be recorded for immediate analyses on a remote computer, where it is matched against a database of previously recorded activity signals. By matching activities to carefully isolated and repeatable signals, similar future generated signals could be successfully correlated back to the activities generating them. A three way mapping is made between the human's control intentions, the associated bioelectrical activity generated, and the part of the human body from which the bioelectrical activity is observed. From this mapping, a control vocabulary of the human bio-electrical activities can be created. In other words, bioelectrical control of robotic systems could be achieved by using a biomedical instrument to capture a user's control intention, through monitoring the bio-electrically generated electrical signals on a part of the user's body, processing this signal on a remote computer to determine the control operation intended, and relaying such control command to a robotic system for execution. The biomedical equipment used could be considered as the bioelectrical control interface. A few of these is discussed in this subsection. The recorded signal data could also be analysed via advanced AI algorithms in order to infer implicit or more complex controls.

2.5.1 Electroencephalography (EEG)

Electroencephalography is the use of an electroencephalograph device to produce a record of brain activity. An electroencephalograph is a machine that uses electrodes placed on the scalp to monitor the electrical activities of different parts of the brain. Clinically, electroencephalography (EEG) is the recording of the brain's spontaneous electrical activity over time, by multiple electrodes placed on the scalp (Niedermeyer and Lopes Da Silva, 2004). It can be graphically visualized as a plot of the voltage generated by an activity over a period of time. EEG is primarily applied in medical diagnosis. Spectral patterns of neural oscillations (brain waves) are used for medical diagnosis of epilepsy, sleeping disorder, brain death, encephalopathies, and coma (Abou-Khalil and Misulis, 2006). In robotics, these patterns are being studied, analysed, and decoded for HCI control applications. The brain activities are often produced as complex tracings, which are then mapped to predefined human activities, such as various arm control - left, right, up, down, back, forward, and sideways. In practice, most cerebral signal waveforms observed in the scalp EEG ranges from $1 - 20$ Hz. These are subdivided into four bandwidths, which are - delta (< 4 Hz), theta ($4 - 8$ Hz), alpha ($8 - 14$ Hz), and beta (> 14 Hz) bandwidths (International Federation of Clinical Neurophysiology, 1999).

The surface EEG technique is a non-invasive method of measuring brain waves. The invasive brain computer interface (BCI) alternatives often involve surgical procedures to implant electrodes directly on the brain's grey matter surface, in order to collect higher quality signals (Anupama et al., 2012). Although these invasive methods often produce the highest quality EEG signals, it is prone to scar-tissue build-up; that is the development of foreign objects in the brain, which could eventually lead to weaker or even lost EEG signals (Anupama et al., 2012). The non-invasive EEG BCI technique, is easy to implement, low cost, and non-invasive. Of the three EEG based BCI signals: sensorimotor rythms (SMRs), steady state visual evoked potential (SSVEP), and slow cortical potentials, SMRs are preferred due to the ease of being detected and its higher level of control (He et al., 2015). In general, EEG based methods seem to be generally preferred to other brain wave monitoring techniques such as the functional magnetic resonance imaging (fMRI), functional near infrared spectroscopy (fNIRs), positron emission tomography (PET), evoked potentials (EP), and event-related potentials (ERPs). The EEG technique is capable of detecting changes over milliseconds (Anderson, 2015), and it can be combined with other techniques such as near infrared spectroscopy (NIRS) and fMRI to record high spatial resolution and high temporal resolution data, simultaneously, without major technical difficulty.

Magnetoencephalography (MEG) is the magnetic imaging alternative to the electrical EEG method. According to the US National Institute of Health (2011b), magnetoencephalography is the measurement of magnetic fields over the head generated by electric currents in the brain. As in any electrical conductor, electric fields in the brain are accompanied by orthogonal magnetic fields. The measurement of these fields provides information about the localization of brain activity which is complementary to that provided by electroencephalography. Magnetoencephalography may be used alone or together with electroencephalography, for measurement of spontaneous or evoked activity, and for research or clinical purposes (National Institute of

Health, 2011b).

Neurosky - brain computer interface

The Neurosky headset is a commercially available EEG measuring device. It is used by HCI researchers in investigating BCI control of robotic systems. An application of the Neurosky EEG bio-sensor MindWave headset in levitating a small multirotor is as shown in Figure 2.36 (Hammacher Schlemmer, 2015). According to Neurosky, this device measures brainwaves and not thoughts as some persons may misunderstand this to do (NeuroSky, 2011). Neurosky has been used as an HCI control interface for video gaming (NeuroSky, 2015) and for controlling a computer cursor (Ang et al., 2015).

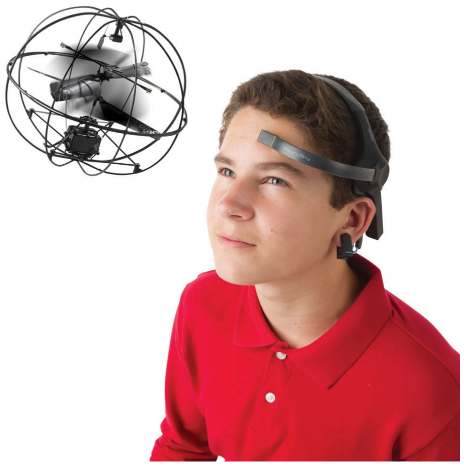
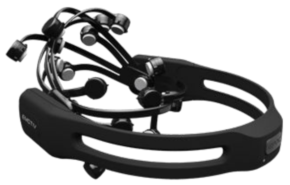


Figure 2.36: Neurosky mind control UFO device available at Hammacher Schlemmer (2015).

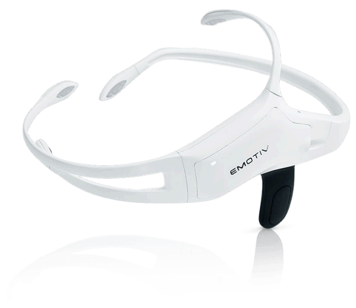
Emotiv - brain computer interface

The Emotiv headset is another commercially available EEG measuring device, which seems more popular among researchers than the Neurosky. There are two Emotiv headsets - the Insight and the Epoc, as shown in Figure 2.37. The Emotiv Insight is a sleek, 5-channel, wireless headset that records brainwaves. It uses proprietary dry polymer sensors to achieve a good level of electrical conductivity, without the usual saline setup requirement of many other EEG devices. It has five channels, which are AF3, AF4, T7, T8, and Pz, and two reference signals (Emotiv, 2014). These are sampled at 128 samples per second per channel. It has a voltage resolution of 0.51 μV and a frequency response ranging from 1 - 43 Hz, covering the four bandwidth range of the scalp EEG signal. Limiting the maximum frequency to 43 Hz ensures the Nyquist criteria is satisfied by the 128 Hz per channel sampling frequency. The Emotiv Insight has been used to investigate BCI levitation control of a small multirotor UAV as shown in Figure 2.37. The Emotiv standard development SDK includes a brainwear detection algorithm, for interpreting signals measured as either mental commands, facial expressions, or brain performance metrics (Emotiv, 2014). Mental commands includes rotate, pull, push, and levitate, and may also include hard to visualize commands like fade or disappear.

(a) Emotive multirotor levitation



(b) Emotive Epoc headset



(c) Emotive Insight headset

Figure 2.37: Emotiv BCI headsets and a robotic control application example (Emotiv, 2014).

On the other hand, the Emotiv EPOC and EPOC+ BCI headset were designed for scientific and research applications such as in neurotherapy. They have a higher performance, smaller resolution, and greater number of channels than the Emotiv Insight. They have 14 EEG channels and 2 references (Emotiv, 2014). The Emotiv Epoc has been used in performing the following research: a) steering a tractor through EPOC collected EMG signals by Gomez-Gil et al. (2011), b) studying biofeedback in virtual reality applications such as gaming by Iancovici et al. (2011), c) robotic arm control by Ranky and Adamovich (2010), and d) non-invasive BCI remote control of a humanoid robot by Thobbi et al. (2010).

University of Minnesota BCI Group

Advances in the field of neuroscience and signal processing have made it possible to develop brain computer interfaces for mind-control applications in robotics. He et al. (2015), a group of BCI researchers from the University of Minnesota, investigated a neuro engineering approach to developing a sensorimotor rhythm (SMR) EEG-based brain computer interface, for the control of robotic devices. By inversely mapping scalp-recorded EEG signals to the cortical source domain, and by integrating BCI with non-invasive neuromodulation strategies, the researchers improved the system's performance and enhanced the operator's BCI control learning process. These researchers also observed that mind-body-awareness-training (MBAT) further enhances mind control of BCI devices. These researchers consider BCI to be a direct alternative means of controlling robotic systems, without the intervention of our natural neuromuscular pathways (moving body parts), by decoding control intents from the brains electrophysiological signal activity (He et al., 2015). By this BCI method, the brain could control devices such as a quadcopter, wheel chair, laptop, or a humanoid directly, via SMR motor imagery, rather than through the use of the human hand, as shown in Figure 2.38. In this case, the human imagines a motor action without actually performing any physical movement. The imagined motor action is BCI decoded, to control a real physical device or a robotic system.

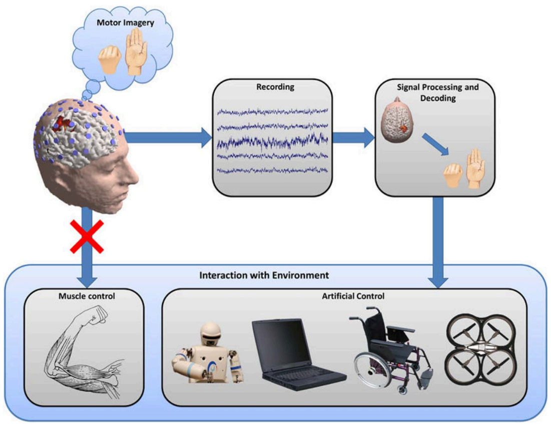
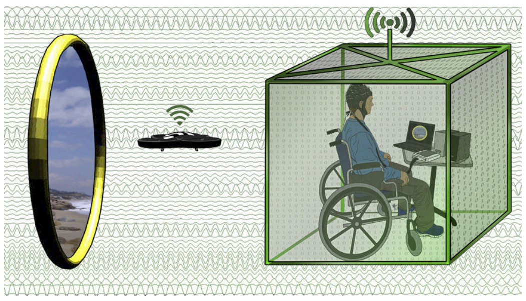


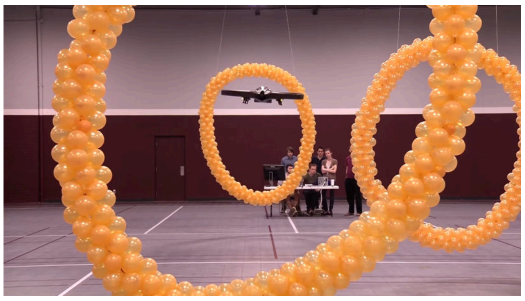
Figure 2.38: BCI control description (He et al., 2015).

In another research, Lafleur et al. (2013) demonstrated the BCI control of a Parrot AR drone quadcopter in three dimensional space, as shown in Figure 2.39. They implemented this using both functional MRI and EEG to map the neural activities of the part of the brain that gets excited by directional movement thoughts.

Tekever's BRAINFLIGHT project

Tekever's BRAINFLIGHT project was an attempt to demonstrate the use of a brain machine interface in the control of a medium-sized fixed-wing UAV, by the Portuguese aerospace firm, using a commercially available off-the-shelf electroencephalography (EEG) cap, normally used in neuro-medical applications for recording signals generated by brain activities. This project was made up of researchers from the Champalimaud Foundation (Portugal), Eagle Science (Netherlands), Technische Universitat Munchen (Germany), and Tekever (Portugal). These researchers combined concepts from the field of aeronautics and neuroscience. By using a high performance non-invasive medical-grade EEG device, they were able to capture high quality brain waves. The team then applied their developed algorithms to analyse, process, and map the acquired brain signals to practical drone control signals, which they successfully demonstrated in Lisbon, Portugal (IMechE PE, 2015; Lee, 2015; Mendes, 2015b; Owano et al., 2015). Figure 2.40 shows the drone pilot wearing the medical-grade EEG cap during the public BCI drone


(a) Emotive multi-rotor levitation


(b) Emotive Epoc headset

(c) Emotive Insight headset

Figure 2.39: The works of the University of Minnesota BCI research group (Lafleur et al., 2013; University of Minnesota, 2013a,b; Nguyen, 2014).

control test. Visual display, audio sounds, and haptic vibrations were used in providing real time feedback to the pilot, who in turn continuously tries to trigger the appropriate brain activities that corresponds to the required adjustment control. Before performing this test, the BRAINFLIGHT researchers had previously tested their system on two flight simulators - a Diamond DA42 aircraft simulator and a UAV simulator. However, the BRAINFLIGHT project does seem to have been temporarily halted, as Mendes (2015b) pointed out that its continued development would require further maturity of all the technologies involved. In other words, the BRAINFLIGHT project was already at the limit of the current available technology.

2.5.2 Electromyography (EMG)

Electromyography is a neuromuscular disorder diagnostic method that uses a biomedical instrument in recording the electrical waves associated with the activities of the skeletal muscles (Kamen, 2004). The electromyogram is the graphical record of electric current (or voltage) associated with muscular contractions. Medically, EMG is an electro-diagnostic technique that uses

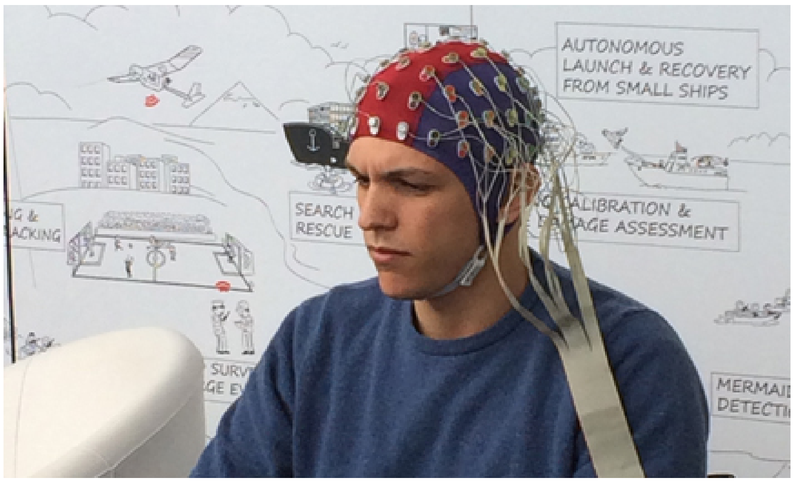


Figure 2.40: Tekever's BRAINFLIGHT specialised COTS EEG cap (IMechE PE, 2015; Mendes, 2015b).

surface or needle electrodes, for recording changes in a muscle's electric potential, when its cells become electrically or neurologically activated (National Institute of Health, 2011a). Instead of directly measuring electrical signals, an alternative technique called magnetomyography (MMG) maps muscle activity by recording the magnetic fields produced by electrical currents, occurring naturally in the muscles, using arrays of SQUIDs (superconducting quantum interference devices). It is considered more suitable than electromyography in detecting slow or direct currents. The development of this technique is influenced by the development of SQUIDs.

There are two EMG acquisition methods: surface EMG (sEMG) and intramuscular EMG (iEMG). In sEMG, a limited assessment of muscle activity is provided by electrodes placed above the muscles' skin surface. Because sEMG relies on voltage potential difference, two or more electrodes are often required to be used. Intramuscular EMG is an invasive but more accurate approach. Different methods, such as monopolar needle electrode approach, could be employed in gaining access to the specific muscles of interest. However, for general HCI robotics control functions, the non-invasive sEMG is preferred. In order to ensure the accurate reception of good control signals, and a reduction in noise and artefacts, a proper preparation routine, such as cleaning and marking points of interest, should be adhered before placing the EMG electrodes. Also, the EMG might need to be calibrated for each use.

EMG often finds applications in the control of prosthetic devices such as prosthetic arms, hands, and lower limbs. It also finds applications in unvoiced speech recognition applications for people with aphasia or without vocal cords. In addition to these applications HCI researchers are investigating EMG control applications for robotics systems, electronic devices, video games, wheel chairs, amongst several other.

2.5.3 Myo - EMG armband device

The Myo armband device, shown in Figure 2.41, is a commercially available EMG capture device, invented by Thalmic Labs in 2013. The Myo armband device is worn on the arm just like an armband. It is able to capture, consistently, a few unique arm gestures, which can then be used for robotic control applications as demonstrated in Nagar and Xu (2015), Thalmic Labs (2015b), and Cacace et al. (2016). Figure 2.42 shows an example of the Myo armband being used to control a small multirotor via hand gestures.

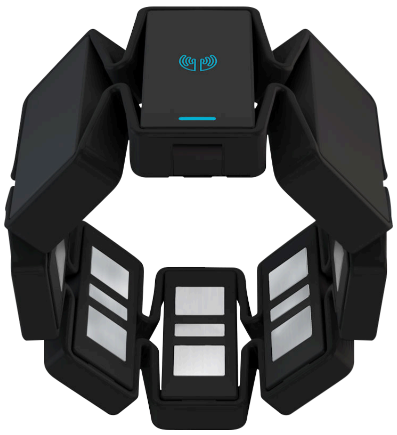


Figure 2.41: Myo armband device (Thalmic Labs, 2013a).

Figure 2.42: Quadcopter hand gesture control using Myo armband device (Thalmic Labs, 2015a).

2.5.4 Electrooculography (EOG)

Electrooculography (EOG) describes a technique used in measuring the corneo-retinal standing potential that exists between the front and the back of the human eye. The resulting signal is called the electrooculogram. It is usually used in ophthalmological diagnosis and in recording eye movements. In order to measure the eye movements, a pair of electrodes is placed either above and below the eye or to the left and right of the eye. By tracking the potential difference between this electrodes, the position of the eye can be precisely estimated. This method is used in gaze tracking systems for medical diagnostics of eye problems, visual system, psychology, cognitive linguistics, and in product design research (Renitto and Thomas, 2014). It could also find application in complementing VR, AR, and MR technologies. The ability to track the eye could be used to focus a cursor on a target rather than the need to rotate the whole head from the neck in order to point a virtual cursor on the target. In fact, Microsoft implemented a gaze tracking system for their MR headset, the Microsoft HoloLens (Microsoft, 2016). Also, Ang et al. (2015) developed a cursor control method for persons with severe disabilities, by using the Neurosky headset as a wearable eye-tracking single-channel EOG HCI equipment.

2.6 Speech

Speech interfaces provide another method of communicating control via voice commands. A microphone is used to detect the sound wave generated by an operator's voice commands, which is then converted to an electrical signal for processing. The operator's speech command may be identified by querying a database of speech command vocabulary with the captured speech signal, for a match. Harris and Barber (2014) investigated verbal speech interaction on a UGV mobile robot. In their experiment, two microphones, a wireless clip-on microphone and the microphone installed on the Microsoft's Kinect, were used to capture the voice input. The captured audio were processed to text by using three separate audio speech recognition (ASR) technologies - Microsoft speech platform SDK, CMU PocketSphinx, and Googles web speech API (Harris and Barber, 2014). These are then compared to a text based vocabulary of commands in order to find a match for an appropriate command execution. In Harris and Barber (2014) investigation, they found out that by specifying a set of searchable command vocabulary, a much higher classification accuracy was achieved. They compared the Microsoft speech SDK and CMU's PocketSphinx, which were dictionary-based language recognisers, with Google's unconstrained grammar speech web API. They found that the dictionary based recogniser yielded better accuracy than the non-dictionary based alternative. Also, they rated the Microsoft Kinect speech platform SDK slightly better than the CMU PocketSphinx.

Anand and Mathiyazaghan (2016) implemented a voice controlled UAV system. In their work, the researchers attempted to guide the navigation of the UAV by using voice commands. In their implementation, the researchers used the Easy VR 3 module to convert spoken words to text. The text is passed through an Arduino microcontroller board, which is then transmitted to their custom-built UAV via RF, as shown in Figure 2.43. Alternatively, the Arduino could be connected to the computer to setup and verify the configuration of the Easy VR 3 module. Anand and Mathiyazaghan (2016) described their voice recognition module operation methods in two steps. In the first step, the acoustic signals were captured and broken into 30 ms segments, for which an acoustic image (the vector of the main signal characteristic) was extracted for each. In the second step, the phoneme (smallest unit of spoken language) of each segment is determined by matching - through a combination of the probability of such phonemes being next to each other, the pronunciation probability, and the probability of the word occurrence in the target language.

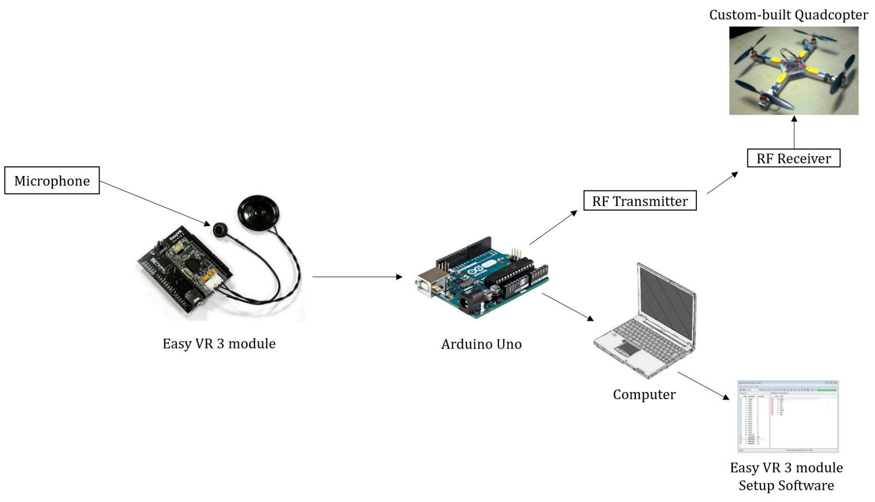


Figure 2.43: Summary of Anand and Mathiyazaghan (2016) voice controlled quadcopter implementation.

Speech or natural language processing is widely considered a natural user interaction modality in HCI and Robotic applications. Redden et al. (2010) observed that speech control is particularly effective during multitasking, as it requires less cognitive resources, especially for short-burst activities; but performance decreases under longer continuous operations (non-discrete tasks). Other benefits include that speech is both hands and eyes free (Redden et al., 2010), it feels more natural, and is not user specific. The development and gradual popularity of commercial voice assistants such as the Apple's Siri, Microsoft's Cortana, Google's Now, and Amazon's Alexa, does indicates a growing trend from tangible to intangible interfaces for consumer household robotic devices. Most research into controlling a UAV via voice communication often include gesture or other communication modalities, and are discussed under the multimodal interfaces (Section 2.7). In this subsection, only works that implement the speech communication mode for UAVs were discussed.

2.7 Multimodal Interfaces

2.7.1 Background

It is often assumed that “Multimodal interfaces can support flexible, efficient, and expressive means of human-computer interaction, that are more akin to the multimodal experiences humans experience in their physical world” (Oviatt, 2003; Preece et al., 2015). This position was supported by Turk (2014), who noted that humans fundamentally interact with the world multimodally. This poses a challenge for HCI researchers to find ways to endow computers with complementary multimodal abilities that are more intuitive and feel more natural, in order to heighten the human experience of interacting with computers, machines, and robotic systems. Turk (2014) attributed the recent rise in the number of multimodal research (any combination of speech, touch, vision, gesture, etc.) to the advancement in non-desktop embedded computing, more powerful mobile devices, and more affordable sensors. According to Shah and Breazeal (2010), humans tend to exhibit more implicit behaviours, utilizing a combination of short verbal and nonverbal gestures in communicating their intentions, when performing tasks under stressed conditions, with resource constraints, and under time pressure, as is often the case in space, military, aviation, and medical domains. They suggested that such efficient form of human to human interaction, could be transferred to human robot interaction in order to improve the efficiency of human robot collaborations over tasks. They further suggested that an effective robot teammate should be able to react to human communications in ways that seems natural to the human, and to have an understanding of how the human teammate may incorporate certain communication cues in their action planning process. Also, Bischoff and Graefe (2002) discovered that HERMES, a humanoid robot assistant, appeared more user-friendly, intelligent, and cooperative, when endowed with the ability to interact via a multimodal combination of speech, vision, and haptics. According to Harris and Barber (2014), soldiers often use a combination of verbal and visual lexicons, to communicate manoeuvres with each other. And that incorporating robots into these existing human ISR teams often presents a human-robot interaction challenge. And with the increasing adoption of semi-autonomous unmanned systems in military operations, intuitive HCI interfaces that use such human-human interaction redundancies, are perhaps essential to both the human-robot interaction and the operation’s success (Harris and Barber, 2014). Hence, the need to further investigate alternative and intuitive HCI control interfaces -

suggesting multimodal interfaces. Furthermore, redundancy in interface development would not only improve the reliability of interaction (Harris and Barber, 2014), but would also improve the effectiveness of the robotic control interaction. This could also contribute to a better overall user experience. Redden et al. (2010) also observed that speech control is particularly effective in brief task usage context, but may perform poorly for task requiring longer continuous control communication; in which case a 'manual' control may be better. Since some task seems to be more efficiently performed by certain type of control interfaces, perhaps, a multimodal interface could ensure that such tasks are performed by the best possible interface for that task.

According to Oviatt (2002), "Multimodal system process two or more combined user input modes - such as speech, pen, gaze, manual gestures, and body movements - in a coordinated manner..." Lackey et al. (2011) defined multimodal communication as "the exchange of information through a flexible selection of explicit and implicit modalities that enables interactions and influences behaviour, thoughts, and emotions." By explicit modalities, the authors meant the use of specific communication methods that clearly conveys control information to the recipient. Whereas, implicit modalities are methods that do not clearly convey the information intended by themselves alone, but often require the recipient to consider other supporting contextual information in order to accurately interpret the intended information. Hence, these implicit modalities may be susceptible to misinterpretation. Social robots are often designed to recognise these implicit cues, which may be relevant in conveying contextual emotional state that might modify the robot's behaviour, actions, thoughts, or perceived interpretation. According to Turk (2014) "Multimodal interfaces describes interactive systems that seek to leverage natural human capabilities to communicate via speech, gesture, touch, facial expression, and other modalities, bringing more sophisticated pattern recognition and classification methods to human-computer interaction." A multimodal interface could be a combination of any two or more compatible, complementary, unimodal interfaces. Multimodal interaction offers the flexibility of switching sequentially across multiple communication modes without breaking the interaction or communicating via parallel modes simultaneously for redundancy and error correction. Also, an interaction started in a sequential multimodal mode, could continue into parallel multimodal modes, and end in a sequential multimodal mode. Multimodal interaction could be particular useful in situations where communicating via only one mode of interaction may be ambiguous, for example in the description of a route on a map, or pointing out an area of interest on an object with several features. As Bolt (1980) suggested, the smooth transition between multiple communication modes could enhance the interaction experience by providing an economy of expression, a briefness in interaction, a more effective interaction, and a greater interaction efficiency. Or as Turk (2014) puts it, "the goal of research in multimodal interaction is to develop technologies, interaction methods, and interfaces that remove existing constraints on what is possible in human-computer interaction, towards the full use of human communication and interaction capabilities in our interactions."

One of the early works on multimodal HCI interaction was Bolt (1980) "Put-that-there", which is shown in Figure 2.44, as was graphically illustrated in Bolt (1980). This early research suggested that speech can be augmented by pointing gestures, and vice versa. And that this had the advantage of being natural, providing an economy of expression, and a briefness in interaction. This challenged later researchers into investigating various multimodal combinations, which could serve as alternatives to the popular keyboard and mouse method being used for computing.

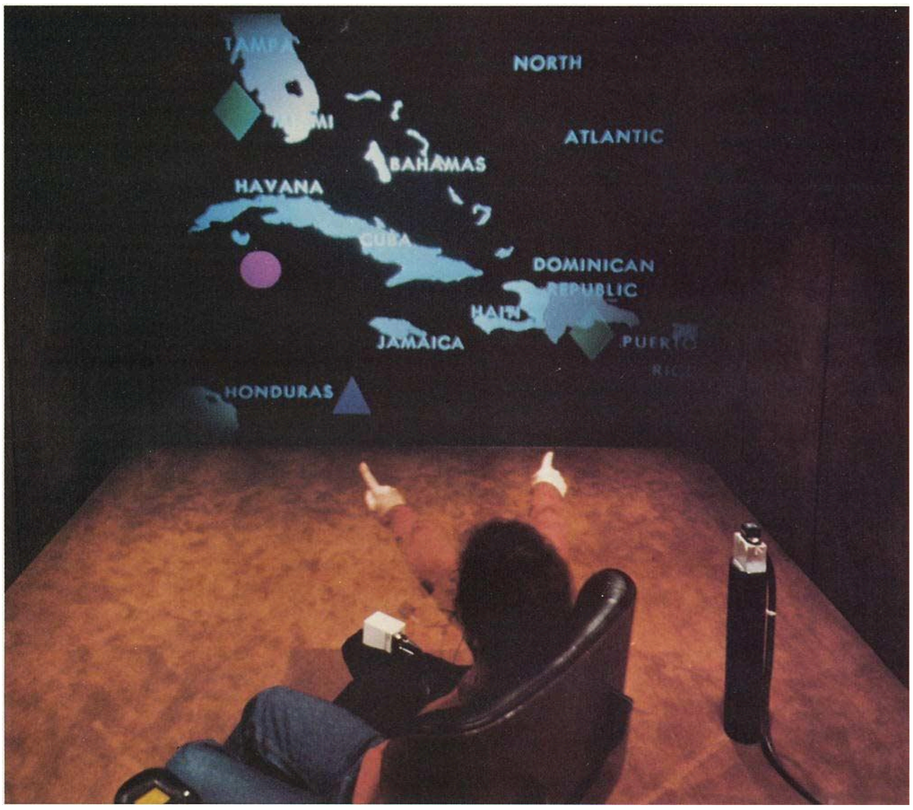


Figure 2.44: "Put that there" multimodal interaction graphical illustration (Bolt, 1980).

In Reeves et al. (2004), a few guidelines that may be considered in developing multimodal interfaces were suggested. These included that interfaces should be consistent, be able to provide feedback, be able to handle or prevent error, and should be able to adapt to different users with different physical abilities or different context of use. Reeves et al. (2004) emphasized that "multimodal interfaces should adapt to the needs and abilities of different users, as well as different contexts of use." Oviatt (1999) presented ten caveat to consider when developing multimodal systems. The ones most relevant to this research were these four: a) users may not necessarily interact multimodally on a multimodal system, b) multimodal interaction primarily offers complementarity and not redundancy, c) individual unimodal system's unreliability do not linearly or geometrically add up to cause greater system unreliability, as operators often tread along strengths of each system to give an overall better system performance, d) heightened user experience, rather than enhanced efficiency, is the main advantage offered by multimodal systems.

Some of the benefits of multimodal interfaces includes repetition, emphasize, briefness, complementarity, naturalness, intuitiveness, inclusiveness, and diversity. An instruction could be repeated over an alternative modality, when the original medium becomes corrupted. For example, visually gesturing could be an effective alternative when speech fails in a noisy environment. Repeating the same command over different modality either simultaneously or in quick succession could be used to emphasize the need for urgency in completing the instructed task. Multimodal interaction provides an economy of expression by making the communication of instruction briefer, efficient, and effective. For example "Go there (pointing where)..." as a combination of verbal and visual gesture command is more effective than through the speech or gesture component alone. According to Haas (2007), speech and haptic interfaces could supplement each other in demanding environments where varying levels of noise and vibration masks cues from a single modality. Rim and Schiaratura (1991) observed the speech and gesture interaction between two persons and noted how these mediums were efficiently utilized by the actors. Humans could switch continuously from one method to the other, in order to avoid a break in interaction, dynamic environments where variables such as lighting and noise levels are continuously being changed, as described in Figure 2.45. Also because humans interact with each other multimodally, multimodal interfaces that emulate the human multimodal methods are generally considered more natural and intuitive, compared to their unimodal counterparts. Multimodal interfaces also tend to be more inclusive than unimodal interfaces, by providing alternative interaction method for persons living with disabilities to interact. According to Lim et al. (2018) the synthesis of multimodal interactions such as touchscreen, gesture recognition, eye tracking, etc could aid the development of context-aware input HCIs.

2.7.2 Speech + Gesture (Myo)

Cacace et al. (2016) investigated multimodal speech and gesture communication with multiple UAVs in a search and rescue mission. They used the Myo armband (shown in Figure 2.46a) for gesture interaction and the open-source Julius framework Lee et al. (2001) for speech recognition. They tested their multimodal deictic speech and Myo based gesture method in the Unity 3D

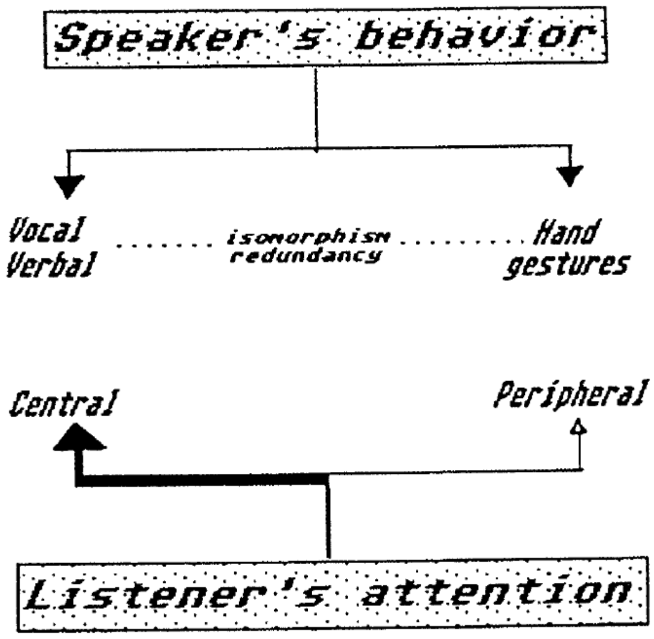
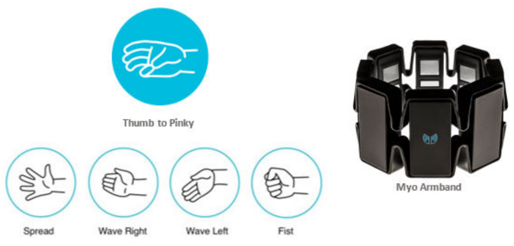
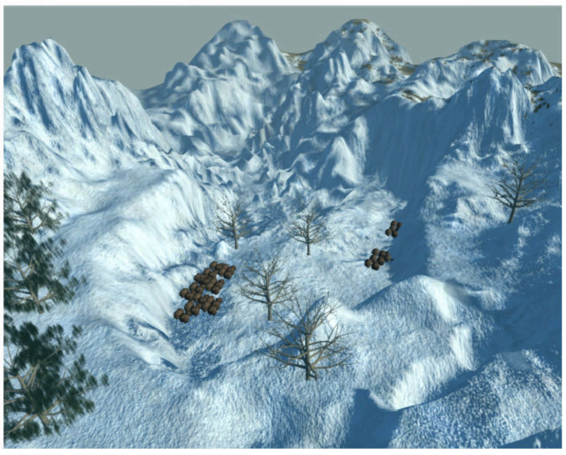


Figure 2.45: Rim and Schiaratura (1991) model of listener's attention to speaker's verbal and non-verbal behaviours.

based simulated search and rescue environment shown in Figure 2.46b.



(a) Myo armband gestures



(b) Simulated search and rescue environment

Figure 2.46: Cacace et al. (2016) gesture device, and search and rescue environment.

In their experiment, a search area of 120×120 m^2 was specified, with six persons missing, which were required to be found within six minutes, by a rescuer with three UAVs. The data collected for their analysis included the number of detected persons, the selection time, operative mode, and interaction type. The result of their simulation showed that a human operator could interact effectively and reliably with a UAV via multiple modalities of speech and gesture, in autonomous, mixed-initiative, or teleoperation mode.

2.7.3 Speech + Gesture (Leap Motion + Camera)

Fernandez et al. (2016) investigated the use of natural user interfaces (NUIs) in the control of small UAVs. They developed a software framework, called Aerostack, to combine several NUI methods and computer vision techniques, as described in Figure 2.47. Fernandez et al. (2016) were motivated by the prospect of aerial robots integrating into the human society, and interacting with humans through natural and intuitive human methods of speech, gesture, and vision. Their project was aimed at studying, implementing, and validating NUI's efficiency in human UAV interaction.

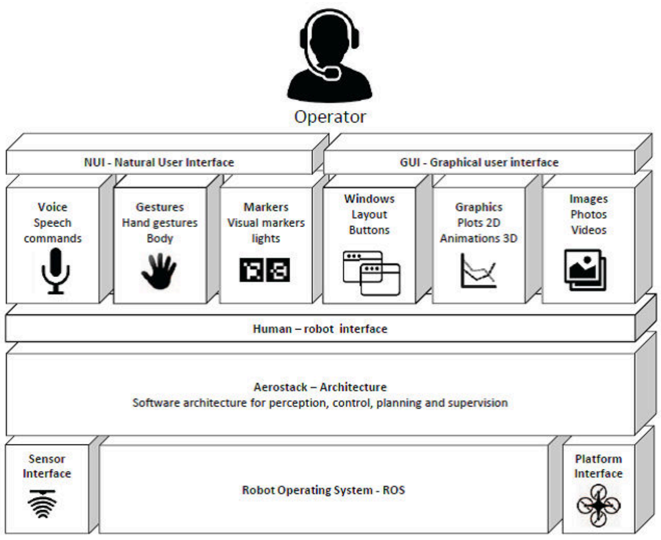

Operator
NUI - Natural User Interface
GUI - Graphical user interface
Voice Speech commands
Gestures Hand gestures Body
Markers Visual markers lights
Windows Layout Buttons
Graphics Plots 2D Animations 3D
Images Photos Videos
Human – robot interface
Aerostack – Architecture
Software architecture for perception, control, planning and supervision
Sensor Interface
Robot Operating System - ROS
Platform Interface

Figure 2.47: Aerostack Architecture - a framework interfacing operator to robot (Fernandez et al., 2016).

In their experiment, they captured whole body gestures and visual markers (for localization and commands) via a Parrot AR Drone 2.0 camera, captured hand gestures via the Leap Motion device, and speech command was captured via the ROS implementation of the CMU PocketSphinx library. A remote computer, running the Aerostack framework, processed all the captured natural communication methods. It then relays control signals to the UAV to execute the operator's control intention. These researchers concluded that natural user interfaces are effective enough for higher level UAV communication, as demonstrated in their work.

2.7.4 Speech + Gesture (Gloves + Kinect)

Barber et al. (2016) investigated the performance of a speech and gesture multimodal interface for a soldier-robot team communication during an ISR mission. The researchers collected performance data on how quickly gestures or speech commands were executed. They attempted to minimize the time between issuing a command and its physical execution. The authors considered that multimodal interfaces could provide redundancy and robustness in interaction. They setup an experiment to evaluate the performance of the multimodal communication interfaces in an experimental outdoor reconnaissance and surveillance mission. They recorded the time it takes the robot to receive, process, and execute an issued command, otherwise called the response time. They also recorded how accurate the commands were decoded, as the interface accuracy. As shown in Figure 2.48, Barber et al. (2016)'s experiment implemented a bi-directional multimodal communication system. The soldier commands the robot vehicle via speech, visual gesture (using the Microsoft Kinect), and non-visual gesture (using the gesture glove). The vehicle communicates back status information via audio and displayed on-screen text on a tablet computer.

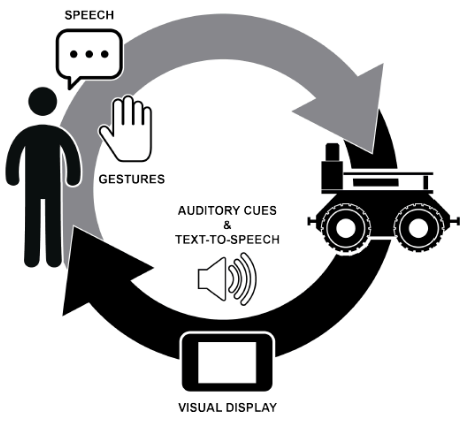


Figure 2.48: Bi-directional multimodal communication description (Barber et al., 2016).

In setting up for their experiments, they integrated several Commercial-off-the-shelf (COTS) components such as Microsoft Kinect sensor used in the Xbox one gaming hardware, with some custom hardware such as the open-finger, Bluetooth based, IMU gesture glove with flex resistors. The researchers implemented a statistical classifier (a machine learning and data-mining algorithm) in their gesture recognition module. This seemed to have improved the gesture processing time to that obtained in their previous work, Harris and Barber (2014). The researchers initiated gesture interaction by holding the fist, a gesture cue to signal gesture start and stop, and a speech call sign "husky" for initiating speech interaction. This was particularly important in order to avoid the robot from responding to gestures that were not directed towards it, and to reduce errors due to random speech utterance not meant for the robot. They also suggested the possibility of developing complex semantic navigation commands such as *"perch over there* (speech + pointing gesture)*, on the tank to the right of the stone monument* (speech)*"* (Borkowski and Siemiatkowska, 2010; Barber et al., 2016). Speech was used to provide contextual information for the pointing gesture and vice versa. The speech SDK utilizes a grammar-based classifier, with a dictionary definition of a few standard military lexical vocabulary. This resulted in a higher classification performance, having faster look-up time and better accuracy, due to its smaller vocabulary set. It also had the advantage of off-line operation, which eliminates the latency required to connect to a remote server for larger speech dictionary definition sets.

In a related research by Hill et al. (2015), the researchers suggested that multimodal speech and gesture communication was a means to achieving an enhanced naturalistic communication, reducing workload, and improving the human-robot communication experience. Kattoju et al. (2016) also investigated the effectiveness of speech and gesture communication in soldier-robot interaction. In their research, they used an updated version of the unmanned ground vehicle (UGV) developed by Harris and Barber (2014). They were motivated by the transition of autonomous robots from being assistant tools to functional operational teammates. In their experiment, they used a Lapel microphone and a commercial-off-the-shelf speech recognition software for capturing and classifying speech, and the Microsoft Kinect in conjunction with a custom-built gesture glove for capturing visual gestures. Based on the UGV's high classification rate for the speech and gesture communication, and the minimal training required to execute this communication by the operator, the researchers concluded that these communication methods have a significant potential in soldier-robot control interaction.

2.7.5 Speech + Gesture (Camera)

As an extension to the Amazon drone delivery project (Amazon, 2016), Amazon now has a patent for a UAV that can interpret gesture and vocal commands, a drone that could in theory be used to deliver packages. The patent describes a drone-like device outfitted with various sensors, cameras and other equipment that could recognize gestures such as a person waving it towards them or someone shooing it away (Locklear and Engadget UK, 2017). Figure 2.49 shows a UAV approaching a human who is waving at it wildly. The idea is to equip the drone with light sensors and cameras, auditory sensors and output devices like speakers or a laser projector (Locklear and Engadget UK, 2017). Clearly, Amazon is considering speech and visual gesture for aerobots, something this research work is contributing towards too.

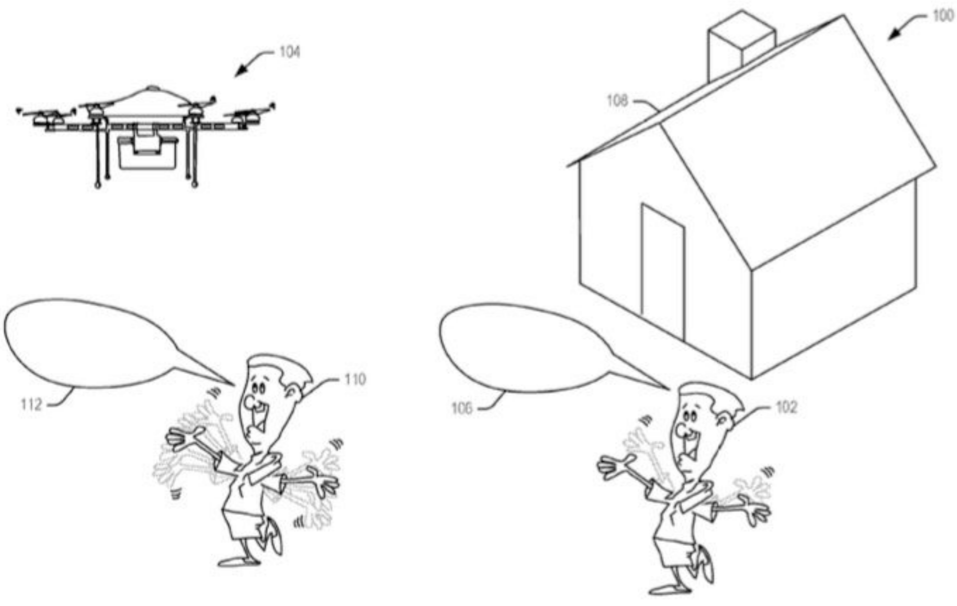


Figure 2.49: Amazon drone delivery gesture patent (Locklear and Engadget UK, 2017).

2.7.6 Wizard of Oz Simulation

Wizard of Oz simulation often involves a drone being controlled by a drone pilot using the RC joystick controller, hidden away from a participant in an experiment. This is usually aimed at elicitation studies on user behaviours. The users are usually made to feel they are in control of the UAV, whereas the drone pilot, hidden away somewhere, is the one in control of the UAV's operation. The drone pilot simply tries to interpret the user's intention and acts on this. A few studies that were based on this type of simulation are described next.

Ng and Sharlin (2011) used wizard of Oz simulation approach to investigate collocated interaction with flying robot, as shown in Figure 2.50. The authors studied the participants behaviour around the UAV. They also studied how the participants' interacted with the UAV. Some of their observations included the users combining speech and two hand gestures in communicating control intentions to the UAV.

Ng and Sharlin (2011) were curious about how gesture control could be compared with the Joystick controller for a UAV in a real world setting, but were not able to perform this investigation due to technical challenge of implementing such a gesture-based system in a real world setting. A problem this research aims to address - a problem of how such a system could be developed. They considered that speech could be used to augment gesture for a more natural

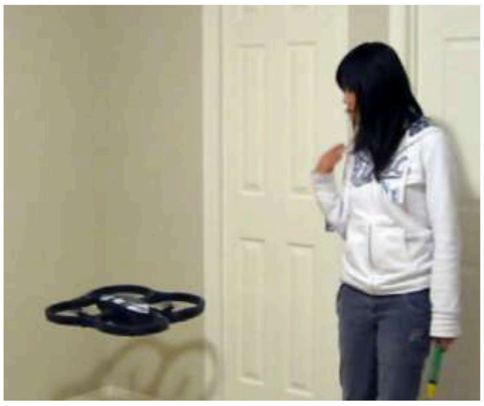


Figure 2.50: A Wizard of Oz UAV collocation experiment (Ng and Sharlin, 2011).

interaction, which is also a fundamental assumption for this research.

Cauchard et al. (2015) performed an elicitation study, with 19 participants, to determine what gestures are considered intuitive for controlling a UAV. Some of the favourite gestures observed are presented in Figure 2.51. Similar to Ng and Sharlin (2011), the main technical challenge faced by Cauchard et al. (2015) was implementing this gesture control method on a real UAV in a real world scenario.

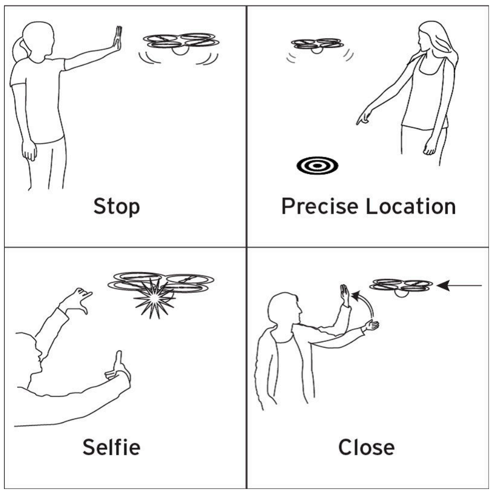
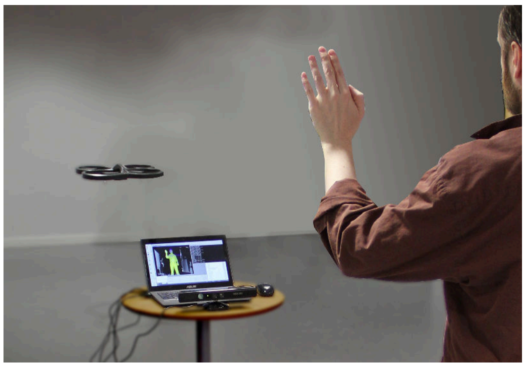
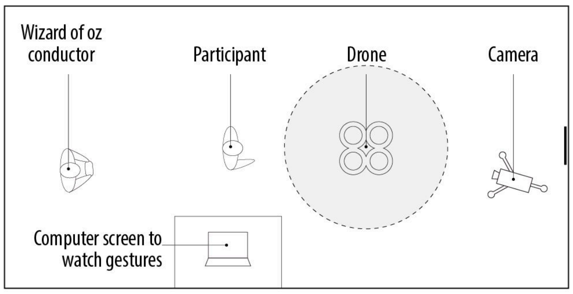


Figure 2.51: Some favourite user-defined gestures for UAV control (Cauchard et al., 2015).

Obaid et al. (2016) extended Cauchard et al. (2015) work further by conducting a similar elicitation study, with 25 participants, to investigate how to control a drone's navigation via gestures - i.e. gesture manoeuvre a drone. Their approach was a user-centred approach, which allowed the participants to use whatever gesture they considered intuitive for performing a set of 12 drone actions of which 10 were navigational commands. Their wizard of Oz experimental setup is as described in Figure 2.52. Figure 2.53 shows a summary of the most popular gestures among the participants for the 12 drone actions.


(a) Experiment conduction



(b) Experiment setup

Figure 2.52: A Wizard of Oz experimental setup (Obaid et al., 2016).

Viallet et al. (2006) describes a real time speech and gesture multimodal interface used in playing chess, shown in Figure 2.54, which was initially developed through a wizard of Oz simulation experiments.

2.8 Limitations of HCI Control Interfaces

2.8.1 Limitations of control agents

Agents used as HCI control interfaces such as remote workstations, laptop computers, tablet computers, and smart phones, have made it possible to pre-plan flight missions and update this in real-time via waypoint mapping. This is usually performed on the agents with the aid of a touch-screen input interface, mouse drag and click, or keyboard input of coordinates. Because of the waypoint pre-plan requirement, this interaction method is not flexible, as it does not support spontaneous control interaction. Missions are executed exactly as scheduled with minimal interruption from the supervisory control operator. Based on some real-time observed feedback, the system operator may be able to change certain objectives, and require the UAV to update its instructions mid-flight and in relative real time. However, this control is neither as instantaneous nor its response as immediate as that produced by the RC joystick controller interface.

2.8.2 Limitations of electromechanical devices

Conventional robotic control interfaces includes manual input devices such as joysticks, touch screens, and trackballs (Redden et al., 2010). Typically, the interaction between humans and

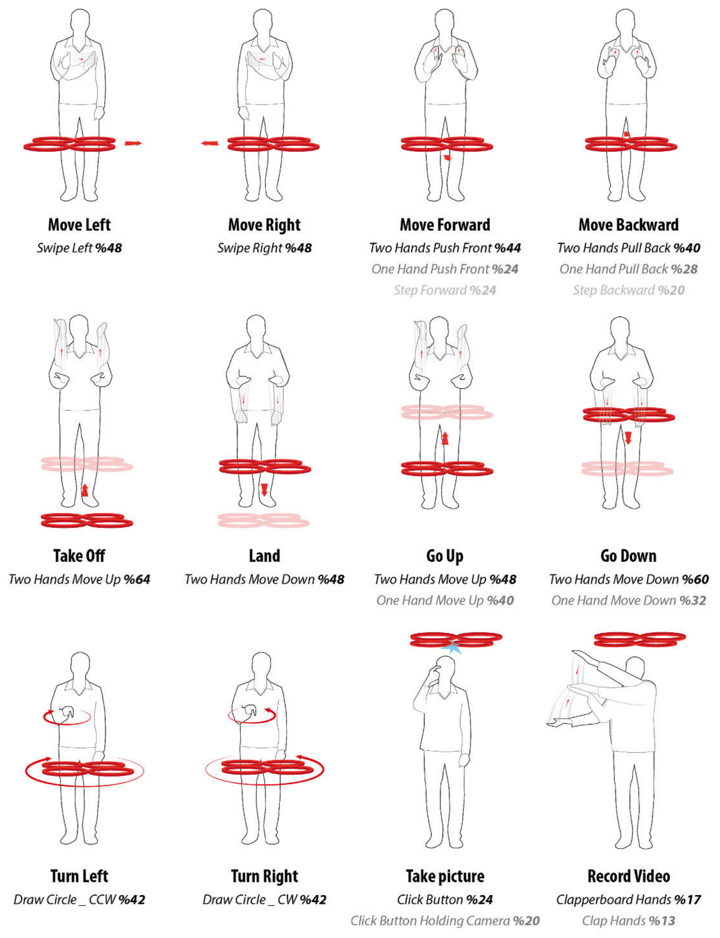


Figure 2.53: Visual representation of popular gesture among the participants of Obaid et al. (2016) drone navigation gesture control elicitation study.

Figure 2.54: Multimodal oral with gesture large display interface (MOWGLI) (Viallet et al., 2006).

aerobots fundamentally consists of a radio controlled joystick controller, augmented with toggle switches, push buttons, and variable potentiometer, for improved control functionality, as presented in Figure 2.9. The RC controller is plagued with a number of limitations, some of which are presented as follows:

1. Dexterity requirement: the operator must have good dexterity in the use of the hands (especially the thumbs) in order to be able to fly the multirotor UAV through the RC joystick controller.

2. Neurological coordination: flying the UAV through the RC controller also requires a good neurological motor response ability - quick mental refresh rate and a fast physical response rate.

3. Hand-eye coordination: this puts a lot of strain on the operator's cognitive effort. Nevertheless, with enough practice, this ability could form a compound-impulse response pathway between the brain and hands, therefore making the control loop feel a bit more natural.

4. Long training hours: due to some control ambiguities, such as dexterity and neuro-coordination requirements, learning to fly a small multi-rotor UAV via the RC joystick controller often requires many training hours (estimated 150 hours to be proficient, and 600 hours to be considered an expert). This helps the operator to overcome these ambiguities by skilfully adapting over time.

5. Potential side effects: this may arise due to the stress accumulated on the thumbs, due to the continuous brisk flickering of both the left and right thumb, placed on the left and right joystick, in order to maintain horizontal level balance, within fractions of a second.

6. Too many controls (Aeryon Labs Inc., 2011): Besides operating the left and right joysticks, the operator might be required to operate switches to perform some other functions such as payload drop or snapshot from an onboard camera, at some specified time, altitude, distance, space, and location. Simultaneously executing this accurately may be difficult.

7. Orientation un-intuitiveness: particularly observable when the UAV's front faces the human operator's front. In this case, the operator's forward command causes the UAV to draw nearer the operator instead of farther away. From the UAV's point of view, this makes perfect sense; but from the operator's point of view, this is a control ambiguity, difficult to quickly process.

8. Multiple UAV control dilemma: due to the control overload on a small unmanned multi-rotor aircraft's operator, the situational awareness of the operator is reduced; making it impossible to simultaneously control multiple aircraft (Cavett et al., 2007).

9. Little or no autonomy: the level of control offered by the RC joystick controller is limited to tier-one of the nCA model; hence, it is not suitable for control operations at higher levels of autonomy.

10. Tangible control device requirement

11. Reduced mobility: the body parts used in actuating this control interface are not available for other functions.

2.8.3 Limitations of vision

Vision based HCI systems are often required to perform the operations of capturing, processing, and responding to control gestures in real time. This is particularly difficult in the absence of good form-sized high-speed hardware with efficient software processing algorithms. Another challenge is the fact that the human body gestures are complexly articulated dynamically fluid, non-rigid, non-discrete motions, which could make building a gesture vocabulary difficult. In addition to these, there is also the possibility of self-occlusion, where some parts of the human body obstruct other parts of the human body being captured, which eventually makes tracking the human body motion more difficult (Huang and Chung, 2004). Vision based systems are also restricted in their range of visibility. The farther away the human operator is from the camera, the more difficult it is for them to be detected.

2.8.4 Limitations of bioelectrical interfaces

Installing the bioelectrical interfaces such as the EEG cap used in Lafleur et al. (2013) UAV BCI control project and Tekever's BRAINFLIGHT (Mendes, 2015a) project, often requires dedicating a good amount of time, which increases the overall setup duration, as opposed to the RC controller "pick and fly" quick setup scenario. Variation in the placement of electrodes on the human bodies could affect bioelectrical activity readings, which probably explains the long setup duration. However, this also reduces the installation flexibility and implies that a drift in electrode position in the middle of a control operation could be potentially dangerous. A good interface should be able to accommodate a wide margin of error, given that humans are prone to many types of errors. In addition, bioelectrical interfaces often require long hours of training and calibration. The cost of setting up, training, and equipment limits the number of pilots as observed in many BCI research projects, where only one or two operators are trained as pilots for such research. Additionally, bioelectrical signals recorded from the same activity may vary from person to person, which suggests the need to observe and calibrate each pilot's bioelectrical activity. In addition to these, bioelectrical signals may vary over time for the same activity being performed by the same person, further making it difficult to create a control vocabulary dictionary for the observed bioelectrical signals. Moreover, if the human operator were to become sick, their bioelectrical activities would probably become erratic and unpredictable, which may result in firing into actions unintended control operations. Finally, a group of researchers suggested the possibility of hacking EEG signals, and wondered that there may even be a $10 - 40\%$ chance of stealing credit card pin details from a carefully observed person (Martinovic et al., 2012). Therefore, this raises a question of security, even though this method is still being researched and largely under development. Closely associated to this security concern is the need for the operator's privacy right protection.

2.8.5 Limitations of speech interfaces

Speech, as an HCI interface, does have a few technical, social, and environmental limitations. Some of these are enumerated below:

1. Lexical ambiguity: unlike written language, spoken language contains some ambiguity. Humans often use the context in which words are spoken to distinguish meanings and interpret the speech appropriately. Some examples of these ambiguities include homophones (words that sound alike), word breaks, abbreviations, numbers, URLs, and punctuations (Kaljurand and Alumäe, 2012). Limiting the speech vocabulary may be a way to reduce this ambiguity. However, this limitation of the speech vocabulary reduces robustness and introduces a requirement for the operator to learn the created control vocabulary.

2. Noise: speech is susceptible to the inherent environmental noise, which often results in classification errors (Kaljurand and Alumäe, 2012; Harris and Barber, 2014). This may be reduced by using an array of microphones to determine and subtract the ambient noise before speech recognition, as performed in the Amazon speech device, Alexa (Amazon, 2016). Directional microphones could also be considered.

3. Speech recogniser's constraint: While speaker dependent speech recognisers may require longer training periods, speaker independent speech recognisers are generally less accurate (Redden et al., 2010). There is also the training time required for speaker-adaptive systems to adapt to the user (Lim et al., 2018).

4. Lower comparative effectiveness: Speech control is often less effective than manual control especially for long continuous control operations, as pointed out in Redden et al. (2010).

5. Dictionary size complexity: the size of the language dictionary could affect the search speed, performance, and accuracy of the speech interface. However, bounding the ASR search space to a set of discrete lexicons, may reduce this complexity (Harris and Barber, 2014) and improve efficiency, although at the expense of a more flexible and robust interaction. That is a reduction in speech vocabulary to improve accuracy limits the number of possible applications of the system (Lim et al., 2018).

6. Context ambiguity: In the case of natural language speech interaction, the robotic system may be unable to differentiate between a real command, sarcasm, joke, or speech not directed towards it (Hirsch and Ehrlicher, 1995; Kaljurand and Alumäe, 2012; Harris and Barber, 2014).

7. Obstruction by other equipment: apparatus such as oxygen or gas mask could obstruct speech clarity, thereby reducing recognition (Redden et al., 2010).

8. Physical user state: the state of the speaker, such as the state of being stressed, in pain, sick, or afraid, could also affect speech clarity; hence affecting speech recognition accuracy (Redden et al., 2010). Similarly, changes in the user's voice under different operational conditions, for example during periods of high or low workload, could also affect the speech recognition accuracy (Lim et al., 2018).

9. Mission constraint: there may also be difficulty in satisfy some mission environmental requirements. For example, the continuum of military operations cuts across many types of environments ranging from stealth, to high noise, high g-force, high vibration, echoes, reverberation, and cross-talk, which may require more robust and resilient interaction features for which the speech interface may be incapable (Redden et al., 2010).

10. Lambard effect: the Lambard effect is the increase in speaker's vocal effort observed when speaking in a noisy environment (Zollinger and Brumm, 2011). This attempt to stretch the vocal limit, during interaction could potentially distort speech, and result in poor speech recognition accuracy (Redden et al., 2010).

11. Inadequate for low-level control: speech may not be suitable for controls below tier 1-III of the nCA model. Control below this level often requires high refresh rate, which speech interfaces are not able to provide, because of the latency involved in issuing, capturing, and processing the speech command.

12. Eavesdropping privacy/security risk: performing authentication via speech command may be challenging because it is easy to be heard when making verbal utterance to a robot. Although, alternative form of authentication such as voice recognition may be considered, privacy remains an issue, especially when the robot is feeding back information such as one's calendar or diary entry.

13. Accidental/inadvertent triggering of the system leading to unintended inputs (Lim et al., 2018).

14. Operator/user differences in speech tone, pitch, accents may also affect system accuracy (Lim et al., 2018)

2.8.6 Limitations of multimodal interfaces

This section discusses three challenges of multimodal interfaces. The first is a problem of the fusion complexity. Integrating two or more imperfect unimodal hardware interfaces could be complicated. As Turk (2014) suggested “different modalities may have different temporal [timing] constraint, and different signal and semantic endurance.” The hardware and software differences of the two or more interfaces need to be carefully considered and the integration performed in such a way that each interface is able to function effectively without getting in the way of the other. How can the interface fusion be optimised in such a way that yields the most effective and efficient performance? In other words, how can one avoid the potential unintended consequences, derived from the sum of individual problems plaguing the unimodal interfaces?

The second problem is that of unimodal dependence. Multimodal interfaces often depend on the maturity of the constituting unimodal technologies. Therefore, the effectiveness and efficiency of the multimodal interfaces may be potentially tied to the constituting unimodal interfaces. Although, Oviatt (1999) argued that multimodal interfaces are more effective and efficient than their constituting unimodal elements, because it is the good bits of the constituting interfaces that are often brought together, in such a way that one interface weakness is complemented by the others strength.

The third problem is that of contradictory commands. What happens when two unimodal interfaces receive contradictory commands? What happens when two contradictory actions result from individual interpretation by the unimodal interfaces of the same control command repeated for different modes? What should be the conflict resolution model for such cases? Avoiding or addressing this control conflict could potentially be an area for further research.

There is also the problem of reduced control command throughput or decreased input speed due to multi-interface capture and processing requirements (Lim et al., 2018). This decreased speed could also arise from control choice overload, resulting in indecision, or slow decision as to which modality should be used for communication in a particular control instance, or even repeating the same control over multiple interfaces. This may have the advantage of providing redundancy of communication channels but it also has the disadvantage of increasing processing time, and hence increasing the time between control command input and execution.

Table 2.3: Human aerobotic interaction (hai) interface comparison.

S/No	Interface Category	Interfaces	Merits	Demerits	Stakeholders/Researchers
1	Electromechanical	RC Joystick Controller Keyboard + Mouse + Joysticks Push buttons + Toggle switches Wired Glove, Data Glove, Cyber Glove; Magnetic Hand Gesture Technique Shift drone thumb controller Google Project Soli	a) Fast response b) Very reliable c) Easier to implement	a) Long hours dedicated to training b) Not natural - not a normal human interaction method c) Not intuitive - especially when re-oriented towards operator d) Demands high cognitive workload	Futaba, Turnigy, DJI, Hobbyking, FrSky, Spektrum, Raytheon, 3D Connection, VDCI; Ma et al. (2010) Lavars (2016) Google (2014)
2	Vision Camera	First Person View (FPV) Leap Motion Microsoft HoloLens Microsoft Kinect Oculus Rift	a) Natural b) Easy to learn	a) Difficult to implement b) Requires higher processing power c) Occlusion	Gupta and Ma (2001) Huang and Chung (2004) Nimble VR (2012) Blanz (2007) Leap Motion Inc. (2010b)
3	Bioelectrical	MYO armband Emotiv headset Neurosky headset	a) Assistive technology b) Gesture interaction c) Stealth interaction	a) Not natural due to wearable device requirement b) High cognitive concentration workload c) Security concerns - snooping EEG signals to potentially steal certain information like login pins, credit card number, etc.	Thalmic Labs (2013b) NeuroSky (2011) Emotiv (2014) Martinovic et al. (2012)
4	Speech	Jasper CMU Sphinx Julius Amazon Alexa	a) Natural b) Easy to learn	a) Difficult to implement b) Not immune to prevailing environmental noise	Bastianelli et al. (2015) Lee et al. (2003) Mu-Chun and Ming-Tsang (2001), Amazon, Google, Microsoft, Apple
5	Agent	Remote Computer Tablet Computer Smartphone Smartwatch	a) Easy to learn b) Touch-screen gestures c) Orientation-tilt feature d) Waypoint programming	a) Relative precision requirement over a small screen area b) Severe fatigue on user's wrist	Huhn and Haewon (2014)
6	Multimodal	HHI-like interaction (Speech + gesture)	a) Natural b) Easy to learn c) Robust - complementing interfaces overcomes unimodal limitations	a) Difficult to implement	Harris and Barber (2014) Oviatt (2003), Bolt (1980) Shah and Breazeal (2010) Lackey et al. (2011) Reeves et al. (2004)

2.9 Summary of HCI Control Interfaces

As already discussed, there is a variety of HCI interfaces available for interacting with robotic systems such as the small, unmanned, multi-rotor aircraft. Table 2.3 presents a summary of these interfaces, their merits, and their demerits.

2.10 Chapter Conclusion

Due to the complexity of the current HCI control interfaces, sometimes two operators may be required to control a single UAV (Aeryon Labs Inc., 2011). For example, in a search and rescue mission, it may be difficult for the pilot to, simultaneously, control the UAV while effectively searching for missing persons. In order for robots to reduce human workload, risk, cost, and human fatigue-driven errors, it is crucial to make the human-robot interaction effective, efficient, and natural, through multiple modalities of contact, dialogue, and gestures (Fong and Nourbakhsh, 2000). The need for intuitive control interaction interfaces for aerobots beyond tier-one components of the nCA autonomy model opens up an opportunity to explore smart novel interaction techniques (Abioye et al., 2017).

Multimodal Interface overview It is often assumed that "Multimodal interfaces can support flexible, efficient, and expressive means of human-computer interaction, that are more akin to the multimodal experiences human experience in their physical world" (Oviatt, 2003; Preece et al., 2015; Turk, 2014). Hence HCI researchers are constantly trying to find ways to endow computers, machines, and robotic systems with intuitive and natural multimodal interaction abilities similar to the human-human experience. This is possible because of the advancement in non-desktop embedded computing, more powerful mobile devices, and more affordable sensors (Turk, 2014). According to Shah and Breazeal (2010), humans tend to exhibit more implicit behaviours, using a combination of short verbal and nonverbal gestures in communicating their intentions, when performing tasks under stressed conditions, with resource constraints, and under time pressure, as is often the case in the space, military, aviation, and medical domains. Bischoff and Graefe (2002) discovered that HERMES, a humanoid robot assistant, appeared more user-friendly, intelligent, and cooperative, when endowed with the ability to interact via a multimodal combination of speech, vision, and haptics. According to Harris and Barber (2014), soldiers often use a combination of verbal and visual lexicons, to communicate manoeuvres with each other, hence incorporating robots into these existing human ISR teams often presents a human-robot interaction challenge.

Multimodal interfaces in aerial robotics A multimodal speech and gesture communication with multiple UAVs in a search and rescue mission, was investigated by Cacace et al. (2016) using the Julius framework (Lee et al., 2001) and Myo device for speech and gesture respectively. The result of their simulation experiment showed that a human operator could interact effectively and reliably with a UAV via multiple modalities of speech and gesture, in autonomous, mixed-initiative, or teleoperation mode. Fernandez et al. (2016) investigated the use of natural user interfaces (NUIs) in the control of small UAVs using the Aerostack software framework. Their project was aimed at studying, implementing, and validating NUIs efficiency in human UAV interaction. In their experiment, they captured whole body gestures and had visual markers (for localization and commands) via a Parrot AR Drone 2.0 camera, captured hand gestures via the Leap Motion device, and speech command was captured via the ROS implementation of the CMU PocketSphinx library. These researchers demonstrated that natural user interfaces are effective enough for higher level UAV communication. Harris and Barber (2014) and Barber et al. (2016) investigated the performance of a speech and gesture multimodal interface for a soldier-robot team communication during an ISR mission, even considering complex semantic navigation commands such as *"perch over there* (speech + pointing gesture)*, on the tank to the right of the stone monument* (speech)*"* Borkowski and Siemiatkowska (2010); Barber et al. (2016). In a related research by Hill et al. (2015), the researchers suggested that multimodal speech and gesture communication was a means to achieving an enhanced naturalistic communication, reducing workload, and improving the human-robot communication experience, especially when factoring in that only a minimal training is required to execute this communication method by the operator. Kattoju et al. (2016) also investigated the effectiveness of speech and gesture communication in soldier-robot interaction. Ng and Sharlin (2011) observed participants' behaviour around UAVs and studied how the participants' interacted with the UAV, particularly how the users combined speech and two hand gestures in communicating control intentions to the UAV. Cauchard et al. (2015) and Obaid et al. (2016) conducted elicitation study to determine intuitive gestures for controlling UAVs. Nagi et al. (2014) investigated human and UAV swarm interaction using spatial gestures.

It is quite clear that there exists a plethora of HCI control interfaces for the control of small multirotor UAVs, as discussed in this chapter. Each of these interfaces is also known to have specific limitations which affects their suitability for a given application. The HCI control interface being proposed, designed, developed, and investigated for the control of small multirotor UAVs (aerobots), particularly for the search and rescue scenario described in Section 1.4.1, is the multimodal combination of speech and visual gesture. This is considered to be an intangible, intuitive, natural, and HHI-like interaction technique that is easy to learn and easy to use. It is also clear that much of the previous research conducted in this area of speech and gesture control for robots often require processing the recorded speech or captured image on a remote computer, which then relays the identified control command to the robot for execution (Waibel et al., 2011; Higuchi et al., 2011; Soto-Gerrero and Ramrez-Torres, 2013; Harris and Barber, 2014; Waibel, 2014b; Anand and Mathiyazaghan, 2016; Barber et al., 2016; Cacace et al., 2016; Fernandez et al., 2016; Ma and Cheng, 2016; Shetty et al., 2016). This research explored a single on-board computer solution, where all the processing of both speech and gesture are being performed in real-time on-board the UAV without remote processing.

Chapter 3

Speech and Gesture Recognition

Due to the multidisciplinary nature of this thesis, this chapter has been written to extend the literature review to help readers unfamiliar with the concept of speech processing via the hidden Markov Model (HMM) and gesture recognition via Haar Cascade and Convex Hull computer vision techniques. Readers familiar with these theories can skip this chapter.

This chapter discusses the theory of speech recognition processing and computer vision object recognition. It focuses on the major techniques used in the development of the multimodal speech and visual gesture component of this research. The first section discusses speech recognition using the hidden Markov model (HMM). The second section discusses the Haar cascade and convex hull computer vision techniques for object recognition.

3.1 Automatic Speech Recognition

In order to discuss the hidden Markov model (HMM) for speech recognition, an understanding of the theory of both the hidden Markov model and the discrete Markov model is required. In this section, the discrete Markov model is first discussed, after which the hidden Markov model is discussed. Speech recognition using the hidden Markov model is discussed last.

3.1.1 Discrete Markov Models

Models can be used to provide the theoretical description of a signal processing system which when used to process the signal provide a desired output. These are important components of prediction systems, identification systems, recognition systems, and several others. Models can be considered to be either deterministic or statistical. In deterministic models, specific properties of the signal are known, for example, it may have a sinusoidal, exponential, polynomial, or linear behaviour. In which case estimating its parameters may be straightforward. In statistical models, an attempt is made to characterise the statistical properties of the signal. These systems include Markov, hidden Markov, Gaussian, and Poisson processes among others. Whilst various deterministic and statistical models have been developed for speech processing, as highlighted in (Rabiner, 1989), for the purpose of this research, only the HMM-based statistical model speech implementation is discussed.

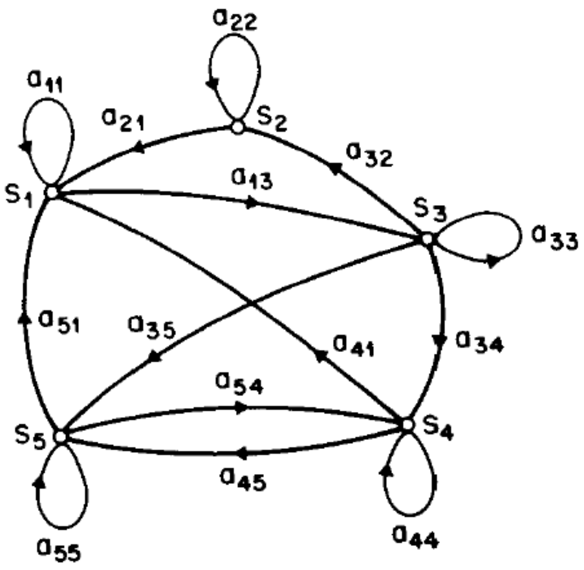


Figure 3.1: A 5-state Markov process (Rabiner, 1989).

A Markov process can be used to describe a system that can only be in one of n distinct states, $s_1, s_2, \ldots, s_n$ at any given point in time. Figure 3.1 shows such a system with five distinct state, undergoing a change of state according to some predefined probabilistic set of rules and at some specific set interval. If the state at time t, defined as q_t, is s_n, then the discrete time Markov chain transition probability can be formulated as being dependent on some previous state as described in Equation 3.1

$$P[q_t = s_n \mid q_{t-1} = s_{n-1}, q_{t-2} = s_{n-2}, \ldots, q_1 = s_1] \tag{3.1}$$

Figure 3.2 show four different model topologies of a 4-state Markov process. In the Ergodic Model, any state can be reached from any state. This model is not ideal for speech recognition because speech consist of an ordered sequence of sounds. The general left-to-right model topology restricts backward transition to previous states. That is state transition occurs in one direction only, left to right. This can be considered as a special case of the Ergodic model with reverse

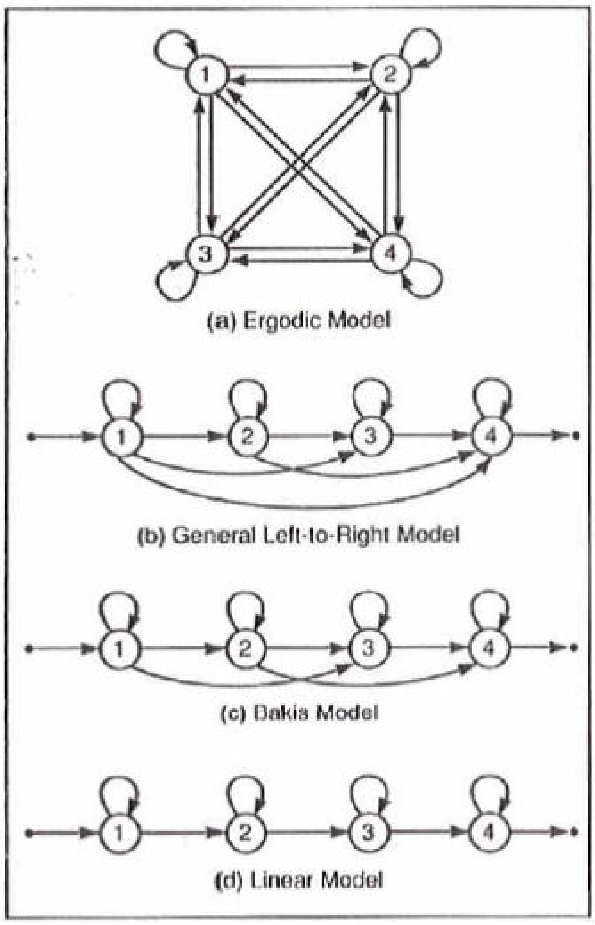


Figure 3.2: Four 4-state Markov chain topologies (Paul, 1990).

transitions probabilities set to zero. The Bakis and Linear model are more constrained forms of the left-to-right models, with transition restricted to a maximum number of state jumps. Generally, the left to right models (Bakis and Linear included) are good for speech recognition, since speech is an ordered sequence of sounds. They also require less training data, are simpler models, and may give better performance than more complex models (Paul, 1990).

For the discrete, first order, Markov chain, in which the current state 'j' is only dependent on the previous state 'i', the transition probabilities a_{ij} described by Equation 3.1 can be truncated to

$$a_{ij} = P[q_t = s_j \mid q_{t-1} = s_i] \tag{3.2}$$

The state transition coefficients have the following properties:

1. Transition probabilities a_{ij} cannot be negative

$$a_{ij} \geq 0 \tag{3.3}$$

2. The sum of the transition probabilities of leaving and entering a particular state equals unity.

$$\sum_{j=1}^{n} a_{ij} = 1 \tag{3.4}$$

Consider a three state Markov model of the weather observed at a particular time of the day, for a week. Figure 3.3 is an Ergodic model of this system.

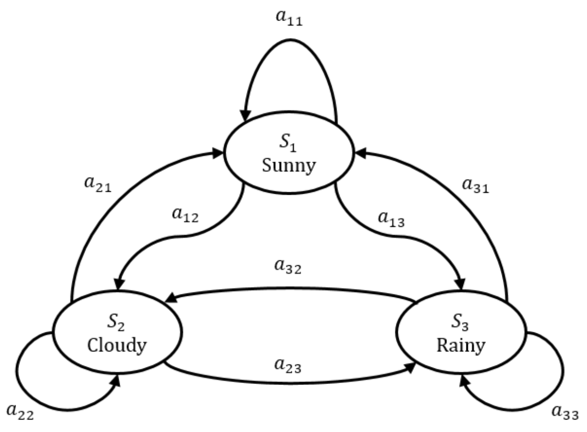


Figure 3.3: Ergodic model of three-state weather Markov process example.

The transition states can be characterised by the following matrix

$$A = a_{ij} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{3.5}$$

According to this model, the probability of observing the sequence of "sunny, sunny, cloudy, rainy, cloudy, sunny, rainy" for the next seven days can be estimated. The observation sequence is given as

$$O = S_1, S_1, S_2, S_3, S_2, S_1, S_3 \tag{3.6}$$

Which corresponds to the time sequence

$$t = 1, 2, 3, 4, 5, 6, 7 \tag{3.7}$$

Therefore, the probability of the observation 'O', given the model 'M', can be evaluated as

$$P(O \mid M) = P[S_1, S_1, S_2, S_3, S_2, S_1, S_3 \mid M] \tag{3.8}$$

$$= P[S_1] \cdot P[S_1 \mid S_1] \cdot P[S_2 \mid S_1] \cdot P[S_3 \mid S_2] \cdot P[S_2 \mid S_3] \cdot P[S_1 \mid S_2] \cdot P[S_3 \mid S_1] \tag{3.9}$$

$$= \pi_1 \cdot a_{11} \cdot a_{12} \cdot a_{23} \cdot a_{32} \cdot a_{21} \cdot a_{13} \tag{3.10}$$

Where π_1 denotes the initial state probability, generally represented as shown in Equation 3.11, usually works out to unity.

$$\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N \tag{3.11}$$

This stochastic process is called an observable Markov model since the output process is the set of states at each instant of time, where each state corresponds to an observable (physical) event (Rabiner, 1989).

3.1.2 Hidden Markov Models

Assuming the Markov model concept is extended to include cases in which the state observation is also a probability, such that the resulting model is a doubly embedded stochastic model, in which the underlying stochastic processes is hidden (not directly observable), although it can be observed through another set of stochastic process that produces the second observation sequence. Such a model is called the Hidden Markov Model (HMM).

To explain this further, consider the classical coin toss example (Rabiner, 1989; Uchat, 2006). Assume a coin is being tossed in a room where a curtain barrier is used to block an observer from directly observing the result of each coin tossed. Assuming an intermediate arbitrator calls out the result of each coin flip without providing any additional information such as "how many coins are being tossed" - single coin repeated, two coins alternately, three coins randomly, etc. Hence a sequence of hidden coin tossing experiment with the observation sequence consisting of a series of heads and tails, is being performed. A typical observation sequence could be

$$O = H, H, T, T, T, H, T, H \tag{3.12}$$

Given this scenario, how does one build an HMM to explain (model) the observed sequence of heads and tails. The first problem in developing the HMM model is the need to define what the states in the model should represent, and then deciding how many states should be in the model. Figure 3.4 describes three possible models. The first assumes that only a single coin is repeatedly tossed each time. In this case, there are only two states with one state representing heads (state 1), and the other, tails (state 2) as shown in Figure 3.4a.

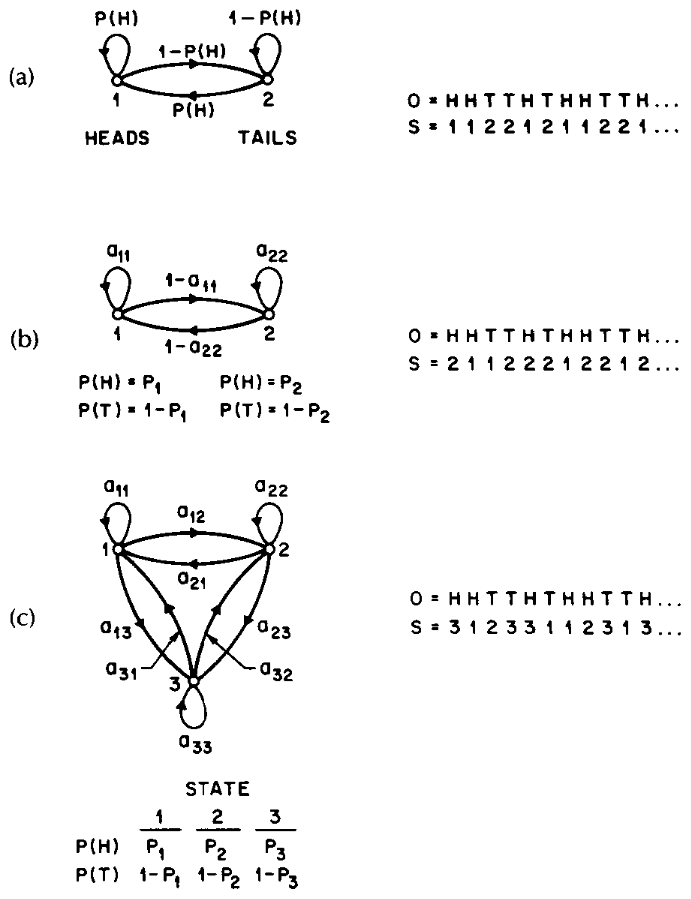


Figure 3.4: Three possible Markov models that can be used to describe the results of the coin tossing experiments (a) 1-coin model, (b) 2-coins model, and (c) 3-coins model. (Rabiner, 1989).

The second model, Figure 3.4b, assumes that there are two coins, with each coin representing a state. How the decision is made between which of the two coins is tossed is unclear. This could be based on a set of coin tosses or other probabilistic events. Hence, the transitions are modelled by some probabilities, a_{ij}, in order to determine whether to remain in the same state 'i' or to transition to the next state 'j'. The third model, Figure 3.4c, is similar to the second model, but assumes that there are three coins, with each coin representing a single state. Similar to the previous case, transitions between states are also being based on either a set of independent coin tosses or some other probabilistic events.

After determining the best model that explains the observation, the next problem is the need to determine the unknown parameters of the model. The first model has one unknown parameter P(H), the second model has four unknown parameters $(P_1, P_2, a_{11}, a_{22})$, and the third model has nine unknown parameters. These are usually estimated through training, from a given set of sample observations.

A hidden Markov model is usually characterised by (Rabiner, 1989):

1. The number of states, N, in the model. The individual states are denoted by $S = \{S_1, S_2, S_3, \ldots, S_N\}$, and the state at time t by q_t. These could correspond to the number of biased coins in the coin toss model of Figure 3.4b and Figure 3.4c.
2. The number of distinct observation symbol, M, which corresponds to the physical output of the system being modelled. For the coin toss model, this is simply heads or tails.
3. The state transition probability distribution, given as $A = \{a_{ij}\}$, where

$$a_{ij} = P[q_{t+1} = s_j \mid q_t = s_i], \quad 1 \leq i,\ j \leq N \tag{3.13}$$

4. The observation symbol probability distribution in state j, given as $B = \{b_j(k)\}$, where

$$b_j(k) = P[V_k\ at\ t \mid q_t = s_j], \quad 1 \leq j \leq N;\ 1 \leq k \leq M; \tag{3.14}$$

5. The initial state distribution $\pi = \{\pi_i\}$, where

$$\pi_i = P[q_1 = s_i], \quad 1 \leq j \leq N \tag{3.15}$$

Hence, given the appropriate values of N, M, A, B, and π, the HMM can be used to generate an observation sequence

$$O = O_1\ O_2\ O_3\ \ldots\ O_T \tag{3.16}$$

Where O_T is one of the symbols from V, and T is the number of observations.

Therefore, a complete specification of an HMM requires two model parameters (N and M), observation symbols V, and the three probability measures A, B, and π. The HMM model can be represented with the following compact notation

$$\lambda = (A, B, \pi) \tag{3.17}$$

In Rabiner (1989), the following three fundamental problems of HMM design were discussed:

1. The evaluation of the probability (or likelihood) of a sequence of observation given a specific HMM
2. The determination of a best sequence of model states
3. The adjustment of model parameters so as to best account for the observed signal

3.1.3 Speech Recognition Using Hidden Markov Model

Automatic Speech Recognition (ASR) focuses on speech recognition and not speech understanding. Speech understanding requires getting the utterance meaning whereas speech recognition is sampling transcribing speech without necessarily getting the meaning of the utterance (Paul, 1990). Practical automatic speech recognition systems based on HMM includes Carnegie Mellon University's Pocket Sphinx project (CMU Sphinx, 2009), Cambridge Universities HTK project (Young et al., 2006; Gales and Young, 2007), and Japan's NIST and KU Julius project (Lee et al., 2001). Template comparison methods of speech recognition, such as dynamic time warping, directly compares the unknown utterance with known samples. HMM creates stochastic models from known utterances, and then compares the probability that the unknown utterance was generated by each stochastic model. Model parameters such as the transition probabilities are estimated through training with the known utterances. Paul (1990) describes the three basic HMM algorithms given below

1. Classification of an unknown observation sequence (recognition)
2. Training the models from a set of training data
   (a) Forward-backward (Baum-Welch) algorithm
   (b) Viterbi training procedure
   (c) Gradient hill climbing
   (d) Simulated annealing
3. Evaluation of the probability of an observation sequence

Some training algorithm emphasises the maximum likelihood training criterion, such as the Viterbi training procedure used in the isolated word recognition of "HISTOGRAM" in Paul (1990), using the waveform, frequency, and time sequence plot as shown in Figure 3.5.

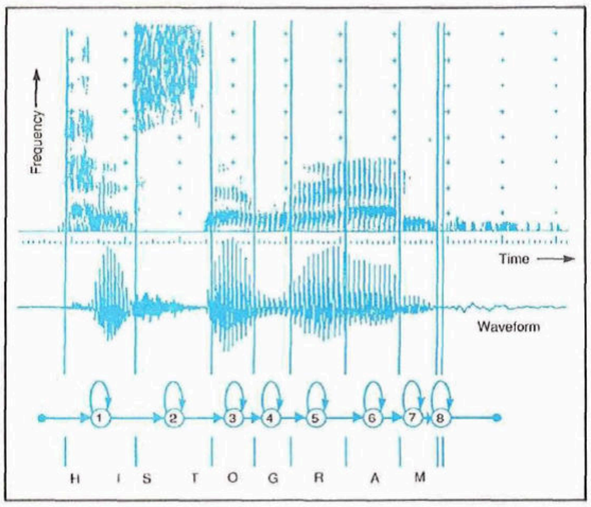


Figure 3.5: Viterbi decoder alignment for the word "histogram" (Paul, 1990).

HMM as used in speech processing complies with a small set of rules. An HMM model $\lambda = (A, B, \pi)$ is used to determine an observation $p(q_t = s_i, q_{(t+1)} = s_j, O|\lambda)$. The hidden Markov model could be based on an ergodic topology model, which has the property that every state can be reached from every other state in a finite number of steps. Therefore, in ergodic or fully connected HMM models, every state of the model could be reached, in a single step, from every other state of the model. However, as previously pointed out, this topology is not ideal for speech recognition, hence the choice often being some form of left-to-right topology. The fundamental property of all left-right HMMs is that the state transition coefficients have the property that

$$a_{ij} = 0, \quad j < i \tag{3.18}$$

That is, no transitions are allowed to states whose indices are lower than the current state, in other words, no backward transitions, hence probability of backward transitions is set to zero. Initial state probability for the left-right HMMs is given by

$$\pi_i = \begin{cases} 0 & i \neq 1 \\ 1 & i = 1 \end{cases} \tag{3.19}$$

In addition, in the left-right HMMs, additional constraints are placed on the state transition coefficient to prevent large state changes, since state sequences must begin in state 1, progressing left-right, and ending in state N.

$$a_{ij} = 0, \quad j > i + \Delta \tag{3.20}$$

Typically Δ is 1 or 2. HMMs for speech recognition often include

(a) Language model

(b) Phone model

(c) Acoustic model

(d) Pronunciation dictionary

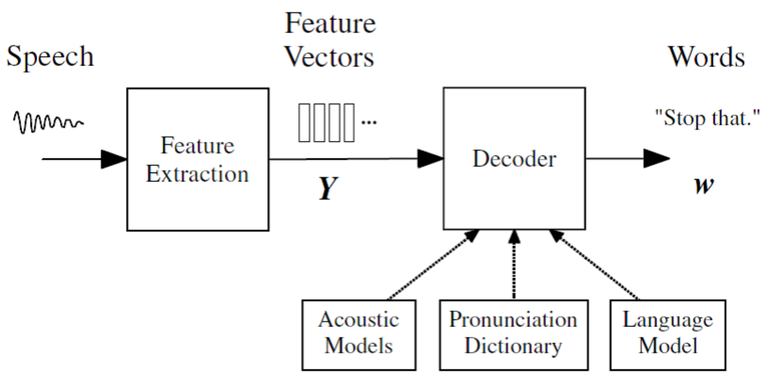


Figure 3.6: Architecture of a HMM-based Recogniser (Gales and Young, 2007).

Figure 3.6 illustrates the main components of a typical large vocabulary continuous speech recogniser - feature extraction and decoder. The decoder consists of the acoustic model, the pronunciation dictionary, and an N-gram Language model. A microphone is used to capture the speech input, converting it into an electrical audio signal waveform, which is then broken into a sequence of 10 ms fixed-size acoustic vectors **Y**, when passed through a feature extraction process Gales and Young (2007), as shown in Figure 3.6.

$$\mathbf{Y}_{1:T} = y_1, y_2, \ldots, y_T \tag{3.21}$$

The vectors **Y** is then fed through a decoder in an attempt to retrieve the word sequence

$$\mathbf{w}_{1:T} = w_1, w_2, \ldots, w_L \tag{3.22}$$

which is most likely to have generated **Y**, that is:

$$\hat{\mathbf{w}} = \arg\max_{x} \{P(\mathbf{w} \mid \mathbf{Y})\} \tag{3.23}$$

Some systems are based on these discriminative models where $P(\mathbf{w} \mid \mathbf{Y})$ is modelled directly. But this is often considered difficult, hence the use of generative models, such as HMMs, where the observation sequence $P(\mathbf{Y} \mid \mathbf{w})$ is modelled instead (Gales and Young, 2007). Given that

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \tag{3.24}$$

And

$$P(A \cap B) = P(B \cap A) = P(B \mid A)P(A) = P(A \mid B)P(B) \tag{3.25}$$

Then

$$P(\mathbf{w} \mid \mathbf{Y}) = \frac{P(\mathbf{Y} \mid \mathbf{w})P(\mathbf{w})}{P(\mathbf{Y})} \qquad \text{(Bayes Rule)} \tag{3.26}$$

Therefore, substituting Equation 3.26 into Equation 3.23, results in

$$\hat{\mathbf{w}} = \arg\max_{x} \left\{ \frac{P(\mathbf{Y} \mid \mathbf{w})P(\mathbf{w})}{P(\mathbf{Y})} \right\} \tag{3.27}$$

Where

$\frac{P(\mathbf{Y}|\mathbf{w})}{P(\mathbf{Y})}$ - is determined by the acoustic model

$P(\mathbf{w})$ - is determined by the language model

The acoustic model represents sounds in phone units. For example, the word "cat" is composed of three phones /k/ /ae/ /t/. The English language is considered to have around 44 of such sound phonemes. For any given word '**w**', the corresponding acoustic model is synthesised by concatenating phone models in order to form words, as defined by a pronunciation dictionary (Gales and Young, 2007). The phonetic model parameters are estimated from training data consisting of their corresponding speech waveforms and orthographic transcriptions. The language model is often patterned after an N-gram model in which the probability of each word is influenced by its preceding word, "N-1". The decoder searching through all possible word sequences, removing unlikely hypotheses through pruning, in order to keep the search tractable. At the end of the utterance, the most likely word sequence is output. Lattices could be used to represent compactly the most likely hypotheses, as shown in Figure 3.7.

Feature Extraction

The feature extraction stage shown in Figure 3.6, breaks the speech waveform captured by the microphone into compact-sized bits. It needs to minimise the loss of information that is used to discriminate between words, and should match the distributional assumptions made by the

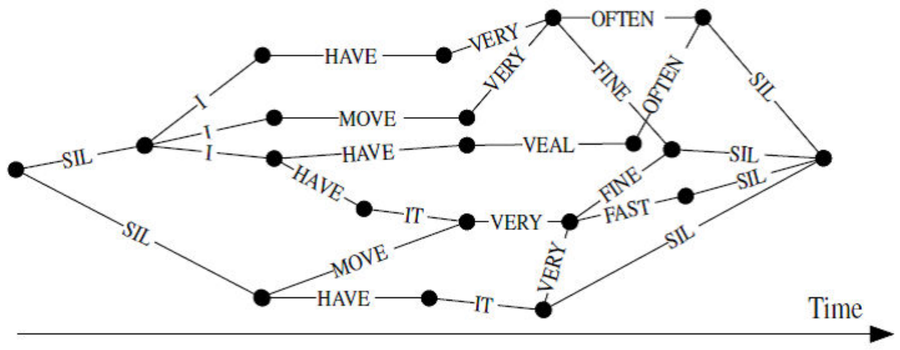


Figure 3.7: Word lattice network (Gales and Young, 2007).

acoustic models (Gales and Young, 2007). Typically, the feature vectors are usually computed every 10 ms using an overlapping analysis window of around 25 ms. The feature parameter y_t is formed from the concatenation of delta parameters as shown below.

$$y_t = \left[y_t^{sT} \quad \Delta y_t^{sT} \quad x^2 y_t^{sT}\right]^T \tag{3.28}$$

Where the delta parameter is given by

$$\Delta y_t^s = \frac{\sum_{i=1}^{n} w_i (y_{t+i}^s - y_{t-i}^s)}{2\sum_{i=1}^{n} w_i^2} \tag{3.29}$$

This results in a partially but not fully decorrelated feature vector with dimensionality typically 44 (number of sound phonemes in the English language).

HMM Acoustic Models

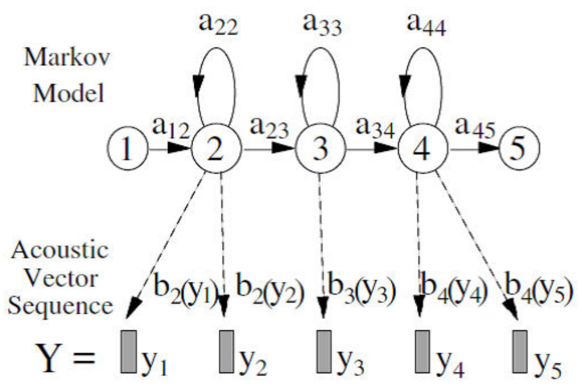


Figure 3.8: HMM-based phone model (Gales and Young, 2007).

Words are decomposed into a sequence of K_w basic sound units or base phones, called pronunciation:

$$q^w_{1:K_w} = q_1,\ q_2,\ \ldots,\ q_{K_w} \tag{3.30}$$

In order to allow for the possibility of slight pronunciation differences (common across multiple speakers), the likelihood $\frac{P(\mathbf{Y}|\mathbf{w})}{P(\mathbf{Y})}$ can be computed over multiple pronunciation training instances, where a summation is carried out over all valid pronunciation sequence of $\mathbf{w}$.

$$\frac{P(\mathbf{Y} \mid \mathbf{w})}{P(\mathbf{Y})} = \sum_{\mathbf{Q}} \frac{P(\mathbf{Y} \mid \mathbf{Q})\ P(\mathbf{Q} \mid \mathbf{w})}{P(\mathbf{Y})} \tag{3.31}$$

Where $\mathbf{Q}$ is a particular sequence of pronunciation,

$$P(\mathbf{Q} \mid \mathbf{w}) = \prod_{l=1}^{L} P(q^{w_l} \mid w_l) \tag{3.32}$$

q^{w_l} is a valid pronunciation for the word w_l. A continuous density HMM with transition probability parameters a_{ij} and output observation distribution b_j is used to represent each base phone, as illustrated in Figure 3.8. The transition probability a_{ij} specifies the probability of making a particular transition from state s_i to state s_j. On entering a state, feature vectors are generated using the distribution $b_j(y_n)$ associated with the state being entered, where $n = 1, 2, \ldots, N$, as shown in Figure 3.8. This follows the standard conditional independence assumptions for HMM as presented in Gales and Young (2007) and Rabiner (1989)

1. States are conditionally independent of all other states given the previous state
2. Observations are conditionally independent of all other observations given the state that generated it

Assuming that the output distribution is a single multivariate Gaussian

$$b_j(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mu^{(j)}, \Sigma^{(j)}) \tag{3.33}$$

Where

$\mu^{(j)}$ - is the mean of state s_j

$\Sigma^{(j)}$ - is the covariance of state s_j

Due to the relatively high dimensionality of the acoustic vector $\mathbf{y}$, the covariance is usually constrained to the diagonal. An alternative approach is to model the output using a mixture Gaussian model. The acoustic likelihood can be determined as

$$\frac{P(\mathbf{Y} \mid \mathbf{Q})}{P(\mathbf{Y})} = \sum_{\mathbf{Q}} \frac{P(\mathbf{\Theta}, \mathbf{Y} \mid \mathbf{Q})}{P(\mathbf{Y})} \tag{3.34}$$

When given the composite HMM $\mathbf{Q}$ formed by concatenating all the constituent based phones $q^{(w_1)}, \ldots, q^{(w_L)}$, where $\Theta = \Theta_0, \ldots, \Theta(T+1)$ is the state sequence through the composite. A more detailed discussion on this is given in Gales and Young (2007). A forward-backward expectation maximization training algorithm is applied to a training set of speech utterances, in order to efficiently estimate the HMM acoustic model's parameters (Baum et al., 1970; Dempster et al., 1977; Rabiner, 1989; Paul, 1990; Gales and Young, 2007).

$$\lambda_{acoustic} = [\{a_{ij}\}, \{b_j(y_n)\}] \tag{3.35}$$

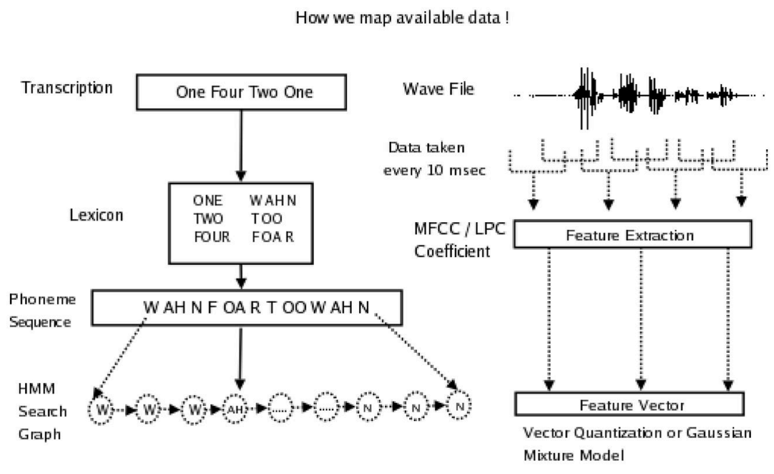


Figure 3.9: Text to lexicon to phoneme sequence to HMM graph Architecture (Uchat, 2006).

The core acoustic models used in the HTK (HMM Toolkit) speech recogniser consists of a set of tied three-state HMMs with Gaussian output distribution, built in an order described in Young et al. (2006), which was also summarised in Gales and Young (2007) as

1. Create a flat-start monophone set based on a single-Gaussian HMM
2. Determine the Gaussian monophone's parameters by performing 3 or 4 EM (Expectation Maximization) iterations
3. Clone each single Gaussian monophone once for each distinct triphone

4. Re-estimate using EM the resulting set of training data triphones

5. Create a decision tree for each state in each base phone

This process results in a tied-state context-dependent acoustic model set. Separating background silence from the input speech can be performed using any of these three techniques (Rabiner, 1989):

1. Explicitly detecting the presence of speech via techniques which discriminate background from speech on the basis of signal energy or signal durations

2. Build a model of the background silence, e.g. a statistical model, and represent the incoming signal as an arbitrary sequence of speech and background i.e.

$$Signal = (silence) - speech - (silence) \tag{3.36}$$

3. Extend the speech unit models so that background silence is included within the first and/or last state of the model. Hence silence inherently gets included within all speech unit models

Limitation of HMMs as applied to speech

1. Assumptions that successive observations (frames of speech) are independent

2. Assumption that the distributions of individual observation parameters can be well represented as a mixture of Gaussian and autoregressive densities.

3. The Markov assumption that the probability of being in a given state at time 't' only depends on the state at time '$t-1$', is clearly inappropriate for speech sounds where dependencies often extend through several states

3.2 Vision Gesture Recognition

In performing the visual gesture recognition for this research, two computer vision object recognition methods were used. These were the Haar cascade object detection and convex hull defects. These methods were chosen because of their popularity in hand gesture recognition applications for robotic systems.

3.2.1 Haar cascade object detection

Viola and Jones (2001), proposed an effective and robust real-time object detection method using Haar feature-based cascade classifiers. Their method is a machine learning based approach where a cascade function is trained from many positive and negative images of an object. The function is then used to detect the object of interest in other images. This method was successfully tested for robust real-time face detection (Viola and Jones, 2004), and was used for hand gesture tracking in this research work. Viola and Jones (2001) object detection procedure classifies images based on the numerical value obtained during the computation of some simple rectangular features/detectors. Feature-based systems have a few advantages over pixel-based systems, such as being able to encode ad-hoc domain knowledge that may be difficult to learn using a finite quantity of training data, in addition to operating much faster (Viola and Jones, 2001). The features used in this method derived from the Haar basis functions. Three kinds of features were used - (a) two-rectangle feature, (b) three-rectangle feature, and (c) four-rectangle feature, as shown in Figure 3.10.

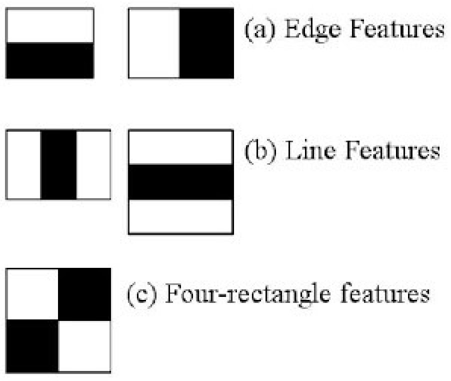


Figure 3.10: Haar features detector window (Mordvintsev and Abid, 2013b) - (a) two-rectangle features, (b) three-rectangle features, and (c) four-rectangle feature.

The sum of the pixels within the white rectangles are subtracted from the sum of the pixels within the black rectangles of the feature detector. For the two-rectangle feature, the difference between the sums of the pixels within the two rectangular regions is computed. For the three-rectangle feature, the sums of the pixels within the two outside rectangles are subtracted from the sums of the pixels within the centre rectangle. For the four-rectangle feature, the sums of the pixels in the diagonal rectangle pairs are subtracted from the other diagonal pair's pixel sums. The computational requirement of this Haar feature extraction from pixel operations could be quite expensive, given that a 24×24 resolution base detector could generate $160,000$ features (Viola and Jones, 2004). Therefore, the authors proposed an integral image rapid computation method that uses intermediate representation for images.

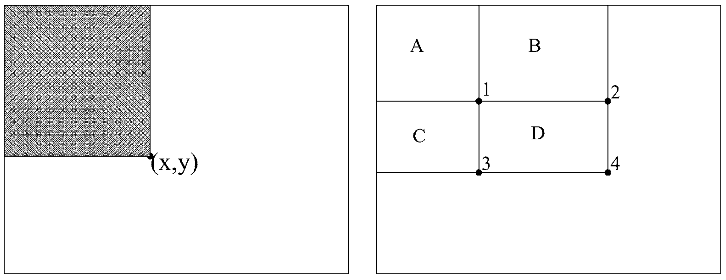
$$ii(x,y) = \sum_{x' \leq x,\ y' \leq y} i(x',y') \tag{3.37}$$

Where

$ii(x,y)$ - is the integral image

$i(x,y)$ - is the original image

The integral image at (x,y), contains the sum of the pixels above and to the left of (x,y) inclusively, as shown in Figure 3.11.



(a) Integral sum (b) Array sum

Figure 3.11: Integral sum at point (x,y) and the four array sum reference (Viola and Jones, 2004).

In order to compute the integral image in just one pass over the original image, the following pair of recurrences are used

$$s(x,y) = s(x,y-1) + i(x,y) \tag{3.38}$$

And

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{3.39}$$

Where $s(x,y)$ is the cumulative row sum, $s(x,-1) = 0$, and $ii(-1,y) = 0$. The integral image computes the sum of any rectangle in four array references as described in Figure 3.11. Given that the integral image at point 1 is given by

$$ii(x_1,y_1) = A \tag{3.40}$$

At point 2 is

$$ii(x_2, y_2) = A + B \tag{3.41}$$

At point 3 is

$$ii(x_3, y_3) = A + C \tag{3.42}$$

And at point 4 is

$$ii(x_4, y_4) = A + B + C + D \tag{3.43}$$

Then the sum of pixels within region D can be computed as

$$[ii(x_1, y_1) + ii(x_4, y_4)] - [ii(x_2, y_2) + ii(x_3, y_3)] \tag{3.44}$$

$$[(A) + (A + B + C + D)] - [(A + B) + (A + C)] \tag{3.45}$$

$$2A + B + C + D - 2A - B - C = D \tag{3.46}$$

From this, it can be shown that the two-feature rectangles can be computed in six array sums of the integral image, the three-feature rectangles in eight array sums, and the four-feature rectangle in nine array sums. Like most object detection systems, this technique scans the input image at many scales. Object detection starts at the base scale in which objects are detected at a size of 24×24 pixels. The image is then scanned at 11 other scales, each at a factor 1.25 larger than the previous image scan.

Given any feature set and a training set of positive and negative images, there exist a number of machine learning approaches, such as Gaussian mixture model, neural network, support vector machine, and winnow learning procedure, that could be used to learn a classification function. However, Viola and Jones (2001) opted for a variant of AdaBoost learning algorithm (Freund and Schapire, 1997) in selecting the features and training the classifier.

The weak learning algorithm was designed to select a single rectangle feature that best separates the positive and negative training examples. For each feature, the weak learner determines the optimal threshold classification function with the least number of examples being misclassified. The weak classifier $h(x, f, p, \theta)$, consists of a feature f, a polarity p, which indicates the direction of the inequality, and a threshold θ. The classifier is described by

$$h(x, f, p, \theta) = \begin{cases} 1, & \text{if} \quad pf(x) < p\theta \\ 0, & \text{otherwise} \end{cases} \tag{3.47}$$

A listing of a single feature boosting algorithm for learning a query online, as described by Viola and Jones (2004), is presented in Table 3.1. 'T' number of hypotheses were constructed during the learning, with each hypothesis using a single feature. The final hypothesis was a weighted linear combination of the 'T' hypotheses where the weights were designed to be inversely proportional to the training errors.

The algorithm presented in Table 3.1 selects key weak classifiers from the set of possible weak classifiers. If there exists one weak classifier per distinct feature/threshold combination, there are effectively KN weak classifiers. K is the number of features while N is the number training image examples. Therefore, given a task with $20,000$ examples and $160,000$ distinct features per example, then there exist 3.2 billion distinct binary weak classifiers. Although the set of weak classifiers is extraordinarily large, the AdaBoost process is quite efficient in selecting the best set of features from the extraordinarily large set of weak classifiers. Viola and Jones (2004) initial work was based on classifiers constructed from 200 features, which yielded a detection rate of 95% with one false positive in $14,084$ testing dataset.

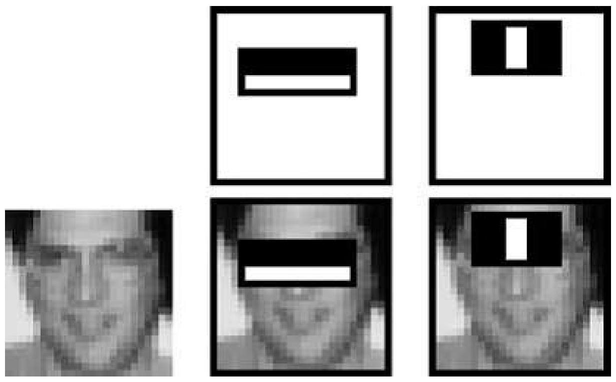


Figure 3.12: The first two features (two and three Haar rectangle features) selected by AdaBoost during classifier training for face recognition (Viola and Jones, 2001).

For the face detection application shown in Figure 3.12, the first feature selected by the AdaBoost algorithm was the horizontal two-feature rectangle, which seemed to focus on the property that the eye region was often darker than the nose and cheeks region. The second feature selected was the longitudinal three-feature rectangle, which seems to focus on the property that the eyes are darker than the bridge of the nose.

In Viola and Jones (2001) Haar feature based object detection method, a cascade of classifiers were used to increase detection performance while radically reducing computation time. Smaller, simpler, efficient, and boosted classifiers were constructed to reject many negative sub-windows

Table 3.1: Viola and Jones (2004) boosting algorithm for learning a query online.

A. Given example images

$$(x_1, y_1), \dots, (x_n, y_n)$$

where $y_i = 0, 1$ respectively represent negative and positive examples

B. Initialize weights

$$w_{1,i} = \tfrac{1}{2m}, \tfrac{1}{2l}$$

for $y_i = 0, 1$ respectively; where m and n respectively represents the number of negative and positive examples

C. For $t = 1, 2, \dots, T$:

1. Normalize the weights,

$$w_{t,i} \rightarrow \tfrac{w_{t,i}}{\sum_{j-1}^{n} w_{t,j}}$$

2. Select the best weak classifier based on the weighted errors of each feature

$$\epsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$$

3. Define

$$h_t(x) = h(x, f_t, p_t, \theta_t)$$

where f_t, p_t, and θ_t are the minimizers of ϵ_t

4. Update the weights

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

Where

$$e_i = \begin{cases} 0, & \text{if example } x_i \text{ is correctly classified} \\ 1, & \text{otherwise} \end{cases}$$

And

$$\beta_t = \tfrac{\epsilon_t}{1-\epsilon_t}$$

D. Determine the final strong classifier

$$c(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

and detect almost all positive instances, before calling on more complex classifiers to process low false positive rates.

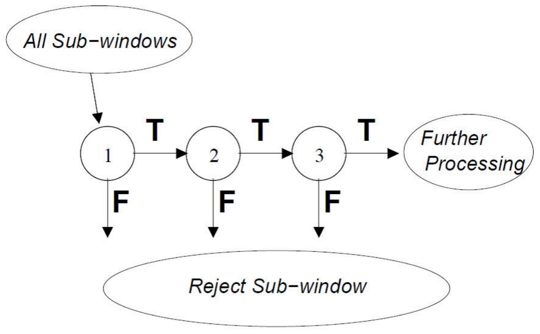


Figure 3.13: Schematic description of how cascaded classifiers work (Viola and Jones, 2001).

Figure 3.13 describes how the cascaded series of classifiers work. The first classifier is adjusted to detect 100% of the object with a false positive rate of 50%. The next classifier stage takes the input from the first state and tries to half or reduce the false positive to around 25%, and so on. The initial classifiers are designed with lower thresholds. Although lower thresholds yields higher detection rate, they also generate higher false positive rates. The aim is to have the initial classifiers eliminate large number of negative examples with very little processing, because subsequent layers require additional computation to detect and eliminate more negatives. This method of rejection by cascade classifiers relies on the assumption that in any given image, an overwhelming majority of the sub-windows are negative, hence the cascade classifier attempts to eliminate as many negatives as possible at the earliest stage possible (Viola and Jones, 2004). Subsequent classifiers in the cascade are trained using examples passed through previous cascade stages; hence, subsequent classifiers face a more difficult task than their preceding classifier stages, which results in them requiring more computational intensiveness. Viola and Jones (2001, 2004) provided a detailed account of the cascade classifier training in their work, which could be referred to for more on this topic. Figure 3.14 shows a ROC (receiver operating characteristic) curve comparison of a 200-feature monolithic classifier and a ten-stage cascade classifier, with each stage having 20 features, from an experiment conducted by Viola and Jones (2001). A ROC curve is a graphical plot of the true positive rate (TPR) or sensitivity against the false positive rate (FPR) or fall-out, at various threshold settings, in order to illustrate the diagnostic ability of a binary classifier system as its discrimination threshold is varied. They concluded that, although the accuracy of the systems were not significantly different, the speed of the cascaded classifier over the monolithic classifier was about ten times faster.

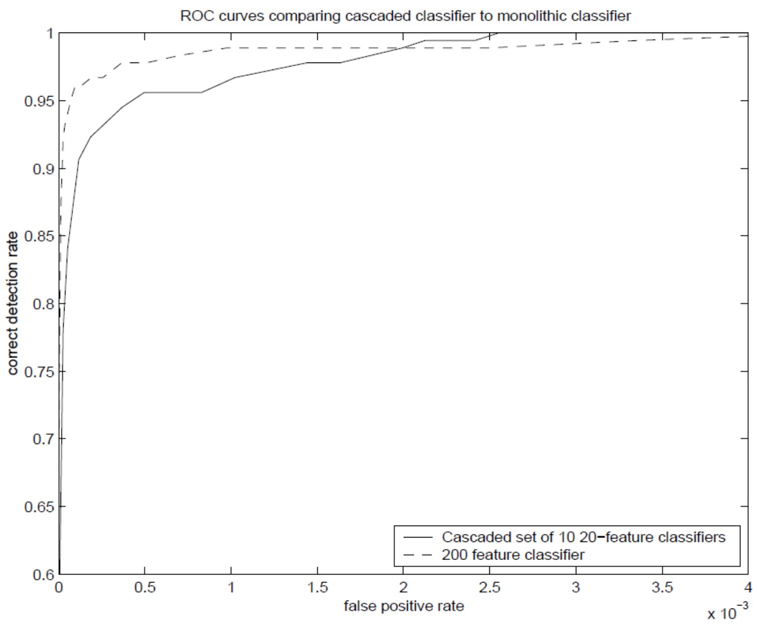


Figure 3.14: ROC curves comparing a 200-feature monolithic classifier with a cascaded classifier containing ten 20-feature classifiers (Viola and Jones, 2001).

In training Viola and Jones (2001) object detection method for face detection, 4, 916 set of cropped faces, scaled and aligned to a base resolution of 24×24 pixels, were used. Each cropped face contained more of the head to include hairline were possible, the chin, and the cheeks; the contours of which were used to improve feature accuracy. Their final detector was a 32-layer classifier cascade with 4, 297 total features. Their feature classifier was trained with 4, 916 faces and 10, 000 non-face, suggesting the ratio of 1:2 positive to negative training examples. That is if 'S' examples were used to train the classifier, then one-third of the training set, $\frac{1}{3}$ S, are positive examples while two-third, $\frac{2}{3}$ S, are negative examples.

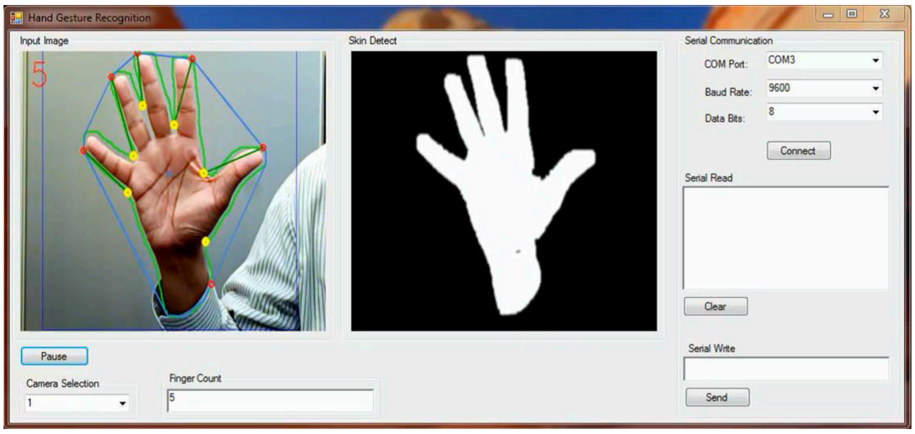
The Haar feature cascade object detection algorithm/method is part of the OpenCV library, which was used in the implementation of the hand fist gesture in this research work. The OpenCV documentation in Mordvintsev and Abid (2013b), provides a Haar feature cascade example for face and eye object. OpenCV already contains many pre-trained classifiers for face, eye, full body, lower body, upper body, smile, and plate number detection, which can be found in the "*opencv/data/haarcascades/*" directory of its GitHub repository (OpenCV, 2010), as XML files. In order to enable anyone to train Haar cascade classifiers for any particular object of interest, a guide on how to train the Haar cascade classifier for object detection in OpenCV has been published by Puttemans (2015).

3.2.2 Convex hull defects hand gesture recognition

In Ganapathyraju (2013) hand gesture recognition through convexity hall defects, the author processed captured images of the hands through a four-stage operation:

(a) Skin detection
(b) Noise elimination
(c) Convex hull algorithm - to get the hand's outline
(d) Convexity hull defects algorithm to determine finger count for control operation

In order to isolate the hand gestures from other environmental components in the image, skin colour detection was performed. This was particularly important for accurately applying the contour detection methods - convex hull and convexity defect algorithms. Ganapathyraju (2013) used an algorithm that was based on the $Y'C_BC_R$ colour space to achieve skin colour detection. In the $Y'C_BC_R$ colour space, Y' represents the luma component, while C_B and C_R represent the blue-difference and red-difference chroma components respectively. In order to achieve robust skin colour detection under varying illumination conditions, the luminance Y' was separated from the chrominance C_B and C_R of the skin. Ganapathyraju (2013) used the following pixel values for skin detection: $60 \leq Y' \leq 255, 100 \leq C_B \leq 255$, and $135 \leq C_R \leq 170$. Figure 3.15 shows Ganapathyraju (2013)'s C# based Windows application of an imaged captured by a webcam, processed with OpenCV convex hull and convexity defect algorithms, to count the number of fingers being held up by a user.


Hand Gesture Recognition

COM Port: COM3
Baud Rate: 9600
Data Bits: 8
Camera Selection: 1
Finger Count: 5

Figure 3.15: Ganapathyraju (2013) convexity defect finger counting hang gesture.

Noise filtering could be achieved by using $cvErode(*)$ and $cvDilate(*)$ OpenCV functions, with the first function eroding or trimming down areas where hands are not detected, and the second function dilating/enlarging the non-eroded areas for further processing. This process effectively removes noisy pixels from the image. An outline of the hand, from which the convex hull was to be computed, was then generated. This was achieved through the application of an OpenCV function, $cvDrawContours(*)$.

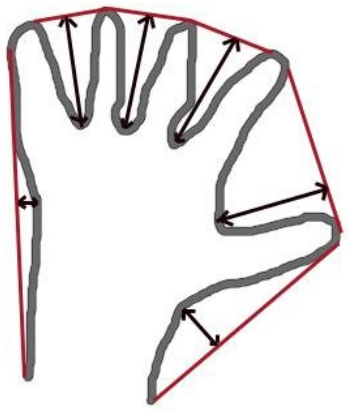


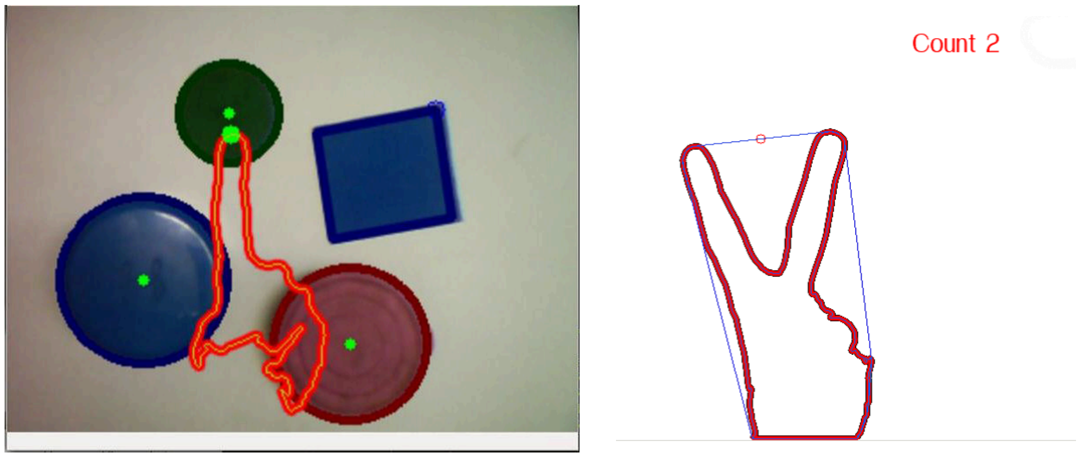
Figure 3.16: Convexity defects Mordvintsev and Abid (2013a).

The $cv2.convexHull(*)$ function checks for convexity defects in curves and tries to correct it. The convex curve usually bulges outward but can also be flat. Convexity defect arises where the curves collapses inward. Figure 3.16 shows the outline of a human hand, with the convex hull represented by the line drawn around the hand, along the convex tips of the fingers, and the convexity defects indicated by the double-sided arrow lines, which are the local maximum deviations of hull from contours, computed using the OpenCV $cv2.isContourConvex(*)$ and $cvConvexityDefects(*)$ functions. The number of defects and the Euclidian distance of the defects from the geometric centre of the contour was used in determining the number of fingers in Ganapathyraju (2013) control investigation. Using a similar method Dhawan and Honrao (2013) developed a convex hull defect based virtual finger pointer and finger counter as shown in Figure 3.17.

Hand gestures may vary across different usage context, geographical region, and cultural backgrounds. Hand gestures could be specific to particular professional fields such as the military, traffic controllers, airfield assistants, etc. Gestures could also be a language such as the American Sign Language (ASL), Mexican Sign Language (MSL), and British Sign Language (BSL).

3.3 Chapter Conclusion

In this chapter, the discrete Markov model was discussed as a first order Markov process explained using an ergodic model. The hidden Markov model was described as a doubly embedded stochastic model in which the underlying stochastic processes is not directly observable,



(a) Finger point (b) Finger count

Figure 3.17: Dhawan and Honrao (2013) convex hull defect based virtual finger pointer and finger counter.

although it can be observed through another set of stochastic process that produces the second observation. For speech recognition using the hidden Markov model, stochastic models from known utterances were first created, and the probability that an unknown utterance was generated by each stochastic model, were then evaluated. Model parameters such as the transition probabilities are estimated through training with the known utterances. Examples of practical HMM-based automatic speech recognition systems included the CMU Pocket Sphinx (CMU Sphinx, 2009), HTK (Young et al., 2006; Gales and Young, 2007), and Julius (Lee et al., 2001). The Haar cascade object detection technique was discussed as a method that uses a machine learning approach for training cascade function, which were then used in object detection. The cascade classifier functions use the edge, line, and four-rectangle Haar feature detectors. The convex hull defect for finger gesture recognition was also discussed. The speech and gesture recognition methods described in this chapter were used in the implementation of the speech and visual gesture components of the mSVG control interface in this research.

Chapter 4

Research Method

This chapter discusses the research method used in conducting this research work. The purpose of this chapter is to present the relevant methods and operational procedures used to complete this research study, and to help ensure that the quality and validity of these procedures are accurate in addressing the research questions. The approach used in this study involves conducting experiments with human participants, using custom hardware systems running custom software programs, based on a novel multimodal speech and visual gesture (mSVG) control interface combination concept, designed and developed in this research, tested within a controlled laboratory environment. The experiment study design, hardware components and setup, and experiment procedure is discussed in this chapter.

4.1 Background

After conducting an extensive literature review (Chapter 2) on HCI control interfaces, as part of the investigation into the use of novel human-computer interfaces in the control of small unmanned multirotor aircraft, it became clear that there exists a plethora of HCI control interfaces to choose from and that the interface selection depends on application scenario and the autonomy level of the UAVs being used. Therefore a number of application scenario of interest were considered, and the search and rescue scenario described in Section 1.4.1, and the domestic application scenario described in Section 1.4.2, were developed. Also, a further research was conducted on UAV navigation autonomy, which resulted in the development of the nCA tier classfication framework described in Chapter 5, as the authors saw the need for an alternative classification system different from the existing limited number of classification schemes available. Many of the interfaces discussed in Chapter 2 have their applications constrained to the tier-one components of the nCA autonomy model. Suitable interfaces for tier-two and tier-three components are generally lacking. In this research, we focused on the development of tier 1-III and tier 2-I nCA autonomy level platforms, as described in Figure 4.1 due to the challenging limitations of current technologies. The original approach was to simulate the nCA Tier 2-I process and develop the nCA Tier 1-III process, as indicated in Figure 4.1.

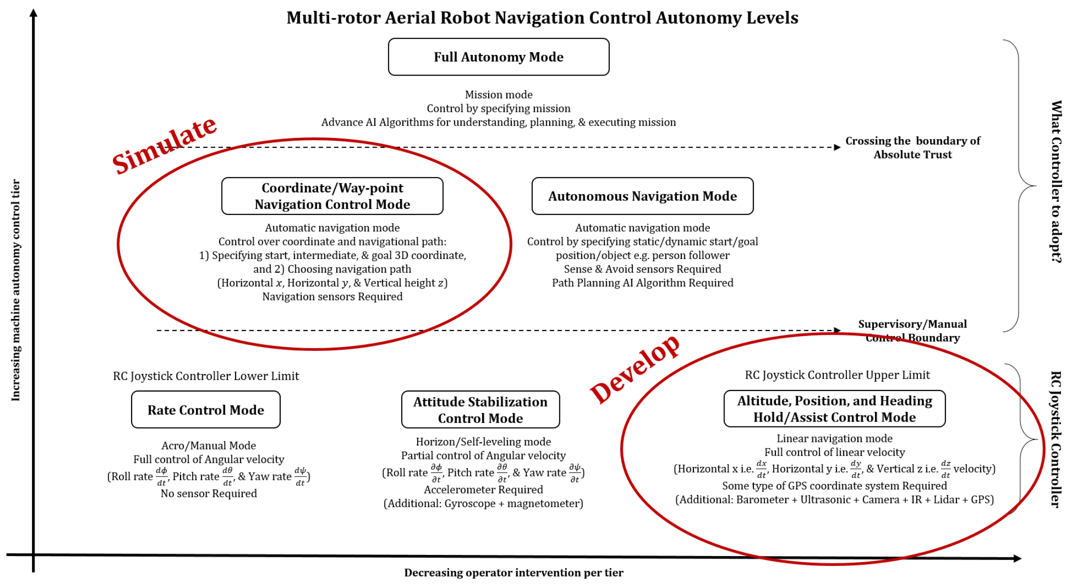

Multi-rotor Aerial Robot Navigation Control Autonomy Levels
Full Autonomy Mode
Mission mode
Control by specifying mission
Advance AI Algorithms for understanding, planning, & executing mission
Simulate
Crossing the boundary of Absolute Trust
Coordinate/Way-point Navigation Control Mode
Automatic navigation mode
Control over coordinate and navigational path:
1) Specifying start, intermediate, & goal 3D coordinate,
and 2) Choosing navigation path
(Horizontal x, Horizontal y, & Vertical height z)
Navigation sensors Required
Autonomous Navigation Mode
Automatic navigation mode
Control by specifying static/dynamic start/goal position/object e.g. person follower
Sense & Avoid sensors Required
Path Planning AI Algorithm Required
What Controller to adopt?
Supervisory/Manual Control Boundary
Increasing machine autonomy control tier
Develop
RC Joystick Controller Lower Limit
Rate Control Mode
Acro/Manual Mode
Full control of Angular velocity
(Roll rate dϕ/dt, Pitch rate dθ/dt, & Yaw rate dψ/dt)
No sensor Required
Attitude Stabilization Control Mode
Horizon/Self-leveling mode
Partial control of Angular velocity
(Roll rate ∂ϕ/∂t, Pitch rate ∂θ/∂t, & Yaw rate ∂ψ/∂t)
Accelerometer Required
(Additional: Gyroscope + magnetometer)
RC Joystick Controller Upper Limit
Altitude, Position, and Heading Hold/Assist Control Mode
Linear navigation mode
Full control of linear velocity
(Horizontal x i.e. dx/dt, Horizontal y i.e. dy/dt, & Vertical z i.e. dz/dt velocity)
Some type of GPS coordinate system Required
(Additional: Barometer + Ultrasonic + Camera + IR + Lidar + GPS)
RC Joystick Controller
Decreasing operator intervention per tier

Figure 4.1: Research Space as described by the nCA model showing two major areas of interests being considered for simulation and development.

This research investigation was conceptualised as a two-sided interaction problem - on one side is the human user and on the other side is the robot being controlled - with the focus being on the middle interface enabling the human to convey control intent to the robot on the other side. This concept is described in Figure 4.2. The developed platform was designed to be able to operate at both tier 1-III and tier 2-I nCA autonomy levels, and to receive control input via multiple input modalities. The nCA API functionally defines the drone autonomy and hence its command capability. Example nCA tier 1-III commands are "go forward half metre", "hover one metre", "take a snapshot", "rotate 270° ", "go [that way] one metre", etc., where commands in square brackets are gestures accompanying a speech command. Example of nCA tier 2-I commands are "go to point D", "return to start point", "go to [that point]", "go to point C, take a snapshot, and then go to goal point G", etc. In nCA tier 1-III, the navigation route is fully specified as a function of direction and distance. Whereas, in the nCA tier 2-I level, navigation was specified in waypoints.

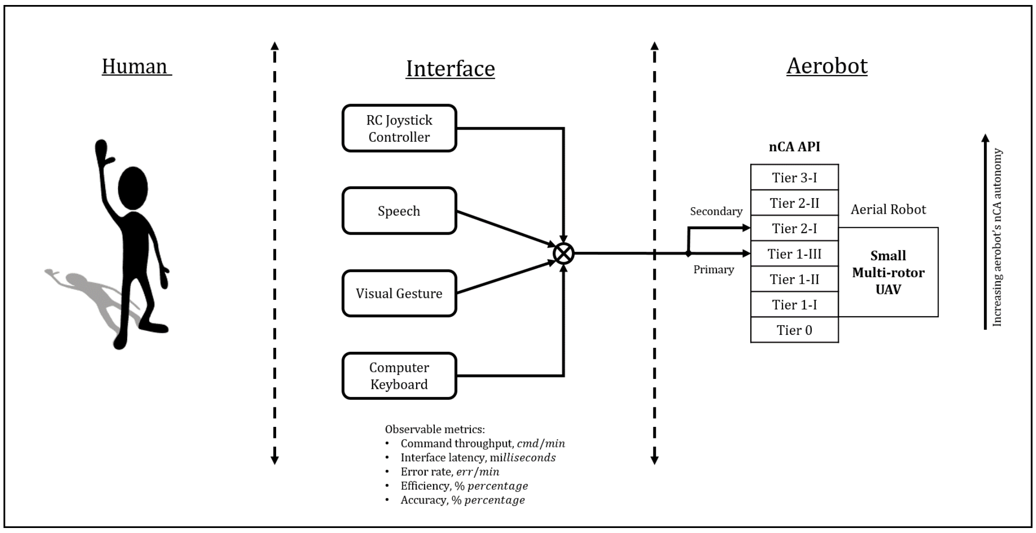


Figure 4.2: A human aerobot interaction interface problem.

4.2 Interface Development

In this section, the development of the speech and visual gesture component was described. However, in order to achieve this, an intermediate step involving the use of the computer keyboard as an interim control symbol generator was first developed. This was then used as the common symbol template on which all control inputs were being translated into before further processing into executable UAV actions. The keyboard was later used as a special case of the RCJ interface with altitude, attitude, and position (AAP) assist during the performance comparison of the RCJ and mSVG interfaces.

4.2.1 Intermediate keyboard control symbol processing

The embedded computer keyboard was used to generate control symbols/commands being processed on the SBC computer and then transferred to the flight controller, thereby bypassing the speech or visual gesture capture and processing stage. This had the advantage of decoupling the complexity associated with developing both the interface capture block and multimodal control processing block simultaneously. The communication between the on-board computers was conducted through the serial USART interface. A set of five control commands - forward, backward, left, right, and stop, was defined. A look-up table was used to map control commands to a unique four-digit number system, able to encode a set of 10,000 unique control commands. This encoding scheme improves medium efficiency, reduces communication bandwidth requirement, and reduces transmission latency. Table 4.1 shows the command-keyboard-symbol mapping.

Table 4.1: Keyboard control command and serial data encoding.

S/N	Command	Keyboard	Serial data
1	Stop	Ctrl + S	1001
2	Forward	Ctrl + Up Arrow Key	1001
3	Backward	Ctrl + Down Arrow Key	1002
4	Right	Ctrl + Right Arrow Key	1003
5	Left	Ctrl + Left Arrow Key	1004

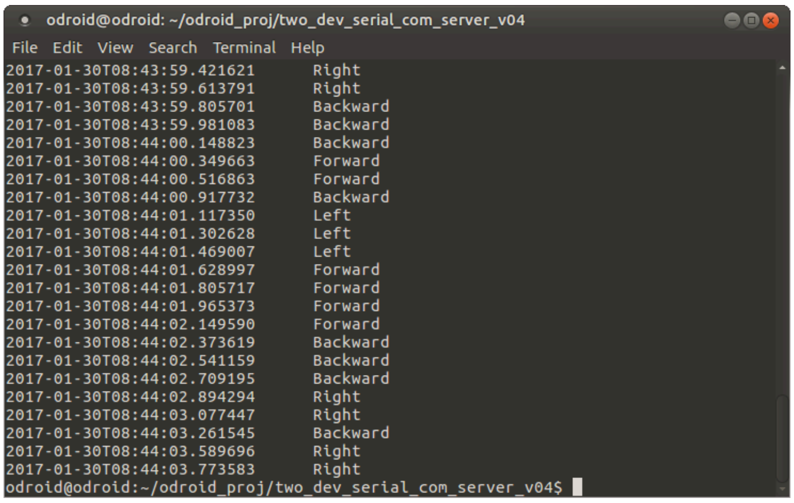
A python script was written to capture and log the keyboard input and capture time. Listing 4.1 is a code snippet for the keyboard input capture and symbol processing. The full source code is listed in Appendix C.6. Figure 4.3 shows the logged output/feedback as displayed on Linux Terminal.

Listing 4.1: Keyboard input capture and symbol processing.

```
def get():
        inkey = _Getch()
        while(1):
            k=inkey()
            if k!='':break
        x = ord(k)
        if x == 3:
                exit()
        elif x == 59:
                        k=inkey()
                        if ord(k) == 53:
                                k=inkey()
                                if ord(k) == 65:
                                        print 'Forward'
                                        ser.write("1001".encode('ascii')+'\n')
                                elif ord(k) == 66:
                                        print 'Backward'
                                        ser.write("1002".encode('ascii')+'\n')
                                elif ord(k) == 67:
                                        print 'Right'
                                        ser.write("1003".encode('ascii')+'\n')
                                elif ord(k) == 68:
                                        print 'Left'
                                        ser.write("1004".encode('ascii')+'\n')

#          print 'you pressed', ord(k)

def main():
        print 'press␣\"ctrl␣+␣Arrow␣Key\"␣for␣Quad␣Navigation␣or␣\"ctrl␣+␣c\"␣to␣Quit'
        while(True):
                get()
```

4.2.2 Speech capture and processing

In developing the speech component of the mSVG interface, speech was captured using the Kinobo USB condenser microphone, and processed using the CMU Pocket Sphinx (CMU Sphinx, 2009) automatic speech recognition (ASR) toolkit. The CMU Sphinx is based on the hidden


odroid@odroid: ~/odroid_proj/two_dev_serial_com_server_v04
File Edit View Search Terminal Help

```
2017-01-30T08:43:59.421621      Right
2017-01-30T08:43:59.613791      Right
2017-01-30T08:43:59.805701      Backward
2017-01-30T08:43:59.981083      Backward
2017-01-30T08:44:00.148823      Backward
2017-01-30T08:44:00.349663      Forward
2017-01-30T08:44:00.516863      Forward
2017-01-30T08:44:00.917732      Backward
2017-01-30T08:44:01.117350      Left
2017-01-30T08:44:01.302628      Left
2017-01-30T08:44:01.469007      Left
2017-01-30T08:44:01.628997      Forward
2017-01-30T08:44:01.805717      Forward
2017-01-30T08:44:01.965373      Forward
2017-01-30T08:44:02.149590      Forward
2017-01-30T08:44:02.373619      Backward
2017-01-30T08:44:02.541159      Backward
2017-01-30T08:44:02.709195      Backward
2017-01-30T08:44:02.894294      Right
2017-01-30T08:44:03.077447      Right
2017-01-30T08:44:03.261545      Backward
2017-01-30T08:44:03.589696      Right
2017-01-30T08:44:03.773583      Right
odroid@odroid:~/odroid_proj/two_dev_serial_com_server_v04$
```

Figure 4.3: iQuad Jade Kebyoard control input log as printed on Odroid XU4 command terminal.

markov model (HMM) for speech recognition described in Chapter 3. Other HMM based ASR considered were HTK (Young et al., 2006; Gales and Young, 2007) and Julius (Lee et al., 2001). Some commercial ASR were also considered such as the Amazon Alexa, Apple Siri, Microsoft Cortana, and Google Now. The CMU Sphinx ASR was chosen because its implementation was well documented, it is open-sourced, supports multiple languages, and can be configured for offline speech processing on a single-board/embedded computer. All speech model is stored offline on the embedded computer and does not require a remote network connection for speech recognition.

The CMU Sphinx installation process for Unix system is well documented under "building an application with pocketsphinx" (CMU Sphinx, 2009). This process is summarised as follows:

1. Download and unpack SphinxBase (support library required by PocketSphinx) and PocketSphinx (recognizer library written in C) packages. At the time of this development, sphinxbase-5prealpha and pocketsphinx-5prealpha were the latest versions available.

2. Build and install SphinxBase using Linux "make install" command from the "sphinxbase" directory via Terminal.

3. Export environment variables for "LD_LIBRARY_PATH" and "PKG_CONFIG_PATH".

4. Switch to the "pocketsphinx" folder via Terminal, and also build and install using "make install" Linux command.

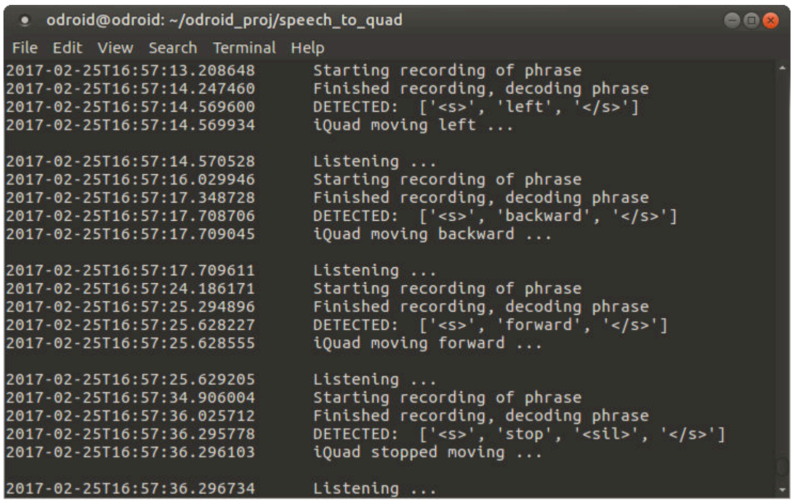
5. Test installation by running "pocketsphinx_continuous -inmic yes" example from Terminal and speaking into connected microphone.

Although, the CMU Sphinx ASR was originally developed in C and Java, a python wrapped version to work with this research's application was developed by combining the C-based basic usage example with Li (2016) Python-based example. The Kinobo USB microphone was connected to the Odroid XU4 SBC. The recording was performed as a single-channel (monaural), little endian, unheadered 16-bit signed PCM audio file sampled at 16 kHz, as required by the Sphinx ASR. This was temporarily saved with a ".wav" extension. The Sphinx ASR engine was then started to process and recognise the uttered speech, performing a speech-to-text translation of the recorded speech command. A keyword search was then performed on the recognised text, in order to identify commands and modifiers of interests. After completing the control operation, the temporarily recorded ".wav" speech command is deleted. However, a text log of the preceding command is kept. The default CMU Sphinx US English language model was used in this implementation. The phonetic dictionary model was based on a selection of 19 command words which made up the speech command vocabulary used in this research. Table 4.2 lists the selections of command vocabulary words and phonemes.

Table 4.2: iQuad phonetics dictionary for speech command.

S/N	Words	Phonemes
1	backward	B AE K W ER D
2	climb	K L AY M
3	drop	D R AA P
4	forward	F AO R W ER D
5	go	G OW
6	half	HH AE F
7	hover	HH AH V ER
8	land	L AE N D
9	left	L EH F T
10	metre	M IY T ER
11	one	W AH N
12	right	R AY T
13	stop	S T AA P
14	step	S T EH P
15	pan	P AE N
16	starboard	S T AA R B ER D
17	larboard	L AA R B ER D
18	down	D AW N
19	up	AH P

The nineteen words in the dictionary were selected to improve accuracy and efficiency by reducing the size of the lookup table for each speech utterance to just a few set of phonetic combinations. Figure 4.4 shows a terminal screen capture of the speech recognition and command processing. The full code for the speech capture and processing into control symbol is listed in Appendix C.4.


odroid@odroid: ~/odroid_proj/speech_to_quad
File Edit View Search Terminal Help

```
2017-02-25T16:57:13.208648      Starting recording of phrase
2017-02-25T16:57:14.247460      Finished recording, decoding phrase
2017-02-25T16:57:14.569600      DETECTED:  ['<s>', 'left', '</s>']
2017-02-25T16:57:14.569934      iQuad moving left ...

2017-02-25T16:57:14.570528      Listening ...
2017-02-25T16:57:16.029946      Starting recording of phrase
2017-02-25T16:57:17.348728      Finished recording, decoding phrase
2017-02-25T16:57:17.708706      DETECTED:  ['<s>', 'backward', '</s>']
2017-02-25T16:57:17.709045      iQuad moving backward ...

2017-02-25T16:57:17.709611      Listening ...
2017-02-25T16:57:24.186171      Starting recording of phrase
2017-02-25T16:57:25.294896      Finished recording, decoding phrase
2017-02-25T16:57:25.628227      DETECTED:  ['<s>', 'forward', '</s>']
2017-02-25T16:57:25.628555      iQuad moving forward ...

2017-02-25T16:57:25.629205      Listening ...
2017-02-25T16:57:34.906004      Starting recording of phrase
2017-02-25T16:57:36.025712      Finished recording, decoding phrase
2017-02-25T16:57:36.295778      DETECTED:  ['<s>', 'stop', '<sil>', '</s>']
2017-02-25T16:57:36.296103      iQuad stopped moving ...

2017-02-25T16:57:36.296734      Listening ...
```

Figure 4.4: Speech recognition logging on Odroid XU4 Linux terminal.

The main limitation of the developed speech interface was it susceptibility to continuous noise corruption from 80 dB onwards. Although speech could still be recognised, the random ambient noise triggers intermittent speech recording even when speech is not being uttered. This was noted as the point of regular speech interference. Above 85 dB, continuous ambient noise level recording occurs until the noise source is muted or lowered to under 80 dB, this was noted as the point of speech drowning. Above this point, uttered speech was wholly drowned by the noise level that the speech system gets stuck in a recording loop until the noise source is turned off or noise level lowered. It was still able to process and interpret the captured speech-noise mix, occasionally recognising some speech utterance spoken during it long recording loop. Therefore, the developed speech interface could not be used beyond 85 dB as implemented.

4.2.3 Visual gesture capture and processing

Hand gesture interpretation varies across different usage context, geographical region, and cultural backgrounds. Hand gestures could be specific to particular professional fields such as the military, traffic controllers, airfield assistants, etc. Gestures could also be a language such as the American Sign Language (ASL) and Mexican Sign Language (MSL). But for the purpose of this study, finger counting gestures (zero-finger/hand-fist, one finger, two fingers, three fingers, four fingers, and five fingers) were used. In developing the visual gesture capture system, two OpenCV techniques were used - Haar cascades and convex hull.

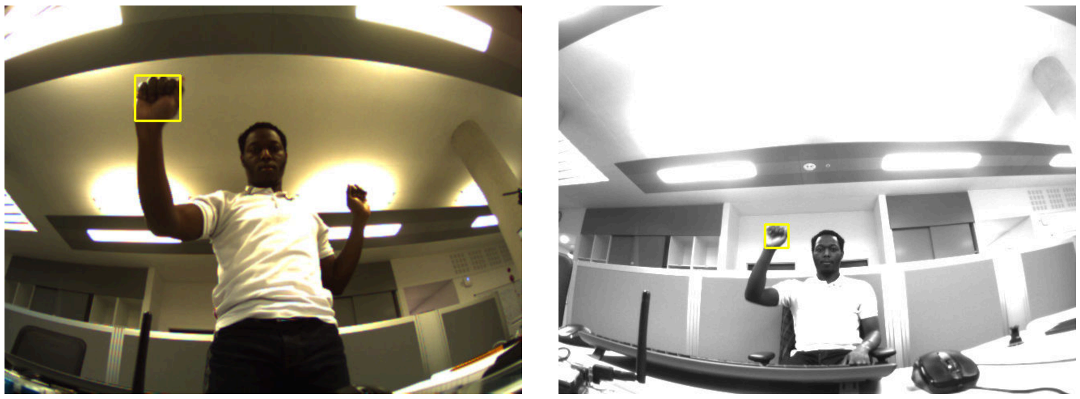
Haar cascades fist tracking

The IDS uEye global shutter colour camera, UI-1221LE-C-HQ (IDS GmbH, 2012), combined with the BM-2118-V2 lens, was used. With the aid of the IDS uEye API manual and libraries (IDS GmbH, 2008), a C program was written to configure and capture a sequence of 200 images on the UI-1221LE-C-HQ uEye camera. The full code is listed in Appendix C.5.2. The hand fist gesture was processed by using the Haar cascade OpenCV algorithm presented in Mordvintsev and Abid (2013c) and the Haar cascade classifier for detecting the letter 'A' in the American Sign Language (ASL) by Wachs (2005), as used in Wachs et al. (2006). Listing 4.2 shows a condensed code listing for applying the "*aGest.xml*" Haar cascade classifier in the hand fist gesture recognition application. The full code is listed in Appendix C.5.1.

Listing 4.2: Haar cascade fist gesture processing.

```
 1 import cv2
 2 import numpy as np
 3 
 4 hand_cascade = cv2.CascadeClassifier('aGest.xml')
 5 
 6 #cap = cv2.VideoCapture(1)
 7 hand_hit = 0
 8 for k in range(0,200):
 9         filename = "C_Img_%d.bmp" %k
10         print filename
11         img = cv2.imread(filename,cv2.IMREAD_UNCHANGED)
12         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
13 
14         #hand = hand_cascade.detectMultiScale(img,scale,neighbours)
15         hand = hand_cascade.detectMultiScale(gray, 1.3, 5)
16         for (x,y,w,h) in hand:
17                 cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,255), 2)
18                 hand_hit = hand_hit + 1;
19                 print "Hand_Hit_-_%d" % hand_hit
20                 #cv2.imwrite('processed_img_hand%d.png'%hand_hit,img)
21 
22         #cv2.imshow('img%d'%k,img)
23         #cv2.imshow('img2',gray)
24 
25 total_images = 200
26 print ("Summary:\tHand_Hit_-_%d/%d"%(hand_hit,total_images))
27 cv2.imshow('img',img)
28 while True:
29         k = cv2.waitKey(30) & 0xff
30         if k == 27:
31                 break
32 
33 #cap.release()
34 cv2.destroyAllWindows()
```

Code line 14, $hand = hand\_cascade.detectMultiScale(img, scale, neighbours)$, calls the classifier loaded in line 4, specifying the image being processed, the iteration scale factor, and the minimum number of neighbours required for positive hand object/gesture detection. For example, a scale factor of 1.05 decreases Haar detector window by 5% in each scale iteration of the image. Smaller scale values increased the number of computations required, while larger scale values reduced the number of computations. Also, specifying smaller number of neighbours resulted in many false positives, while a larger number of neighbours resulted in the failure to detect many true positives. A scale factor of 1.3 and neighbour size of 5 was found to be sufficient for this research implementation.



(a) uEye UI-1221LE-C-HQ camera capture. (b) uEye UI-1221LE-M-GL camera capture.

Figure 4.5: Right-hand fist gesture tracking using colour (UI-1221LE-C-HQ) and monochrome (UI-1221LE-M-GL) camera.

Figure 4.5 shows images of the right hand fist gesture being tracked by the colour (UI-1221LE-C-HQ) and monochrome (UI-1221LE-M-GL) uEye cameras. Because of the unavailability of documentation on how to connect the proprietary uEye camera to OpenCV, the two step strategy was adopted – a) the C program to capture a frame sequence, and b) the Python program to process the capture images and recognise the gesture. However, the global shutter uEye camera was eventually replaced with the rolling shutter Odroid 720p web camera, due to further configuration problems in integrating the uEye camera with the OpenCV libraries, for real-time processing in the experiment study.

Convex Hull Finger gestures

While the Haar cascade technique was used to detect fist hand gesture, the convex hull technique was used to detect finger-count hand gestures. These techniques were combined to form the gesture used in this research. The convex hull method used for the finger-counting gesture was similar to that described in Ganapathyraju (2013), where hand gestures were recognised with the aid of convexity hull defects, as explained in Mordvintsev and Abid (2013a). A four-stage image processing operation of skin detection, noise elimination, convex hull and convexity defect processing, was performed with the aid of OpenCV algorithm libraries, in order to count the number of fingers being held up by a human user. In order to isolate the hand gestures from other environmental components in the image, skin colour detection was performed in the $Y'C_BC_R$ colour space. In order to achieve robust skin colour detection under varying illumination conditions, the luminance Y' was separated from the blue-difference chrominance C_B and red-difference chrominance C_R of the skin. Noise filtering was achieved by using $cvErode(*)$ and $cvDilate(*)$ OpenCV functions, with the first function eroding/trimming down areas were the hand was not detected, and the second function dilating/enlarging the non-eroded areas for further processing. This process effectively removes noisy pixels from the image. An outline of the hand, from which the convex hull was to be computed, was then generated. This was achieved through the application of an OpenCV function, $cvDrawContours(*)$. The $cv2.convexHull(*)$ function checks for convexity defects in curves and tries to correct it. Convex curves are usually bulge out, or at least flat. Convexity defect arises where the curve outline caves inward. The number of defects and the Euclidian distance of the defects from the geometric centre of the contour was used in determining the number of fingers. Using a similar method Dhawan and Honrao (2013) developed a convex hull defect based virtual finger pointer and finger counter.

Listing 4.3: OpenCV contours and convex hull.

```python
import cv2
import numpy as np

img = cv2.imread('Img_4_edit.jpg',cv2.IMREAD_UNCHANGED)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(5,5),0)
blur2 = cv2.medianBlur(img,5)
ret,thresh1=cv2.threshold(blur,170,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
#ret,thresh1=cv2.threshold(blur,190,255,cv2.THRESH_BINARY)
thresh2=cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,11,2)
thresh3=cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,2)

contours, hierarchy = cv2.findContours(thresh1,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
max_area = 0
for i in range(len(contours)):
    cnt = contours[i]
    area = cv2.contourArea(cnt)
    if(area > max_area):
        max_area=area
        ci = i
    cnt = contours[ci]
    hull = cv2.convexHull(cnt)
    drawing = np.zeros(img.shape,np.uint8)
    cv2.drawContours(drawing,[cnt],0,(0,255,0),2)
    cv2.drawContours(drawing,[hull],0,(0,0,255),2)

cv2.imshow('img1',img)
cv2.imshow('img2',gray)
cv2.imshow('img3',blur)
cv2.imshow('img3.1',blur2)
cv2.imshow('img4',thresh1)
cv2.imshow('img5',thresh2)
cv2.imshow('img6',thresh3)
cv2.imshow('img7',drawing)
```

```

while True:
        k = cv2.waitKey(30) & 0xff
        if k == 27:
                break

#cap.release()
cv2.destroyAllWindows()
```

Listing 4.3 is a code listing showing the use of OpenCV for finding convex hull and drawing contours. Different thresholding algorithm were tested in order to determine the most effective threshold for this research application. The result of this observation is shown in Figure 4.6. Binary thresholding was selected and was combined with Gaussian blurring because the combination resulted in a better contour outline, and a more successful gesture recognition.

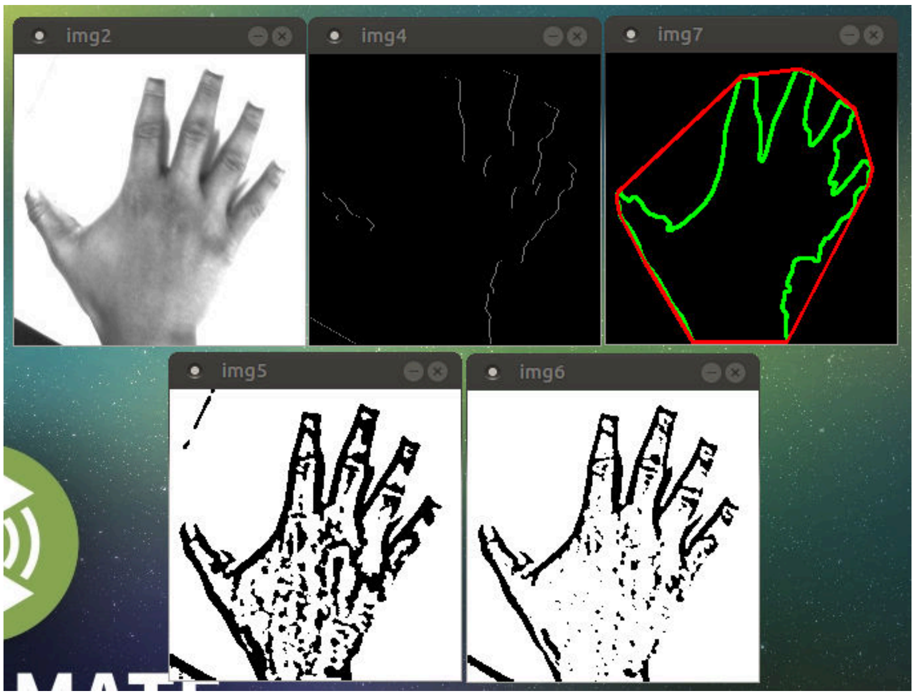
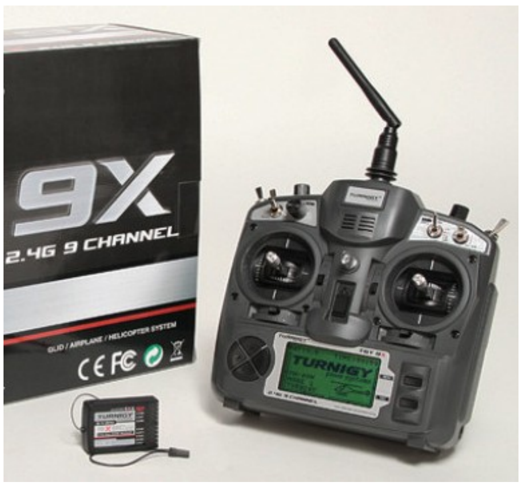


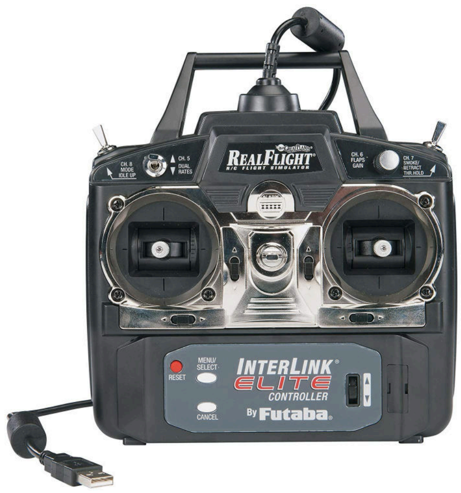
Figure 4.6: Convex hull hand processing - (img2) monochrome grayscale, (img4) binary inversion + Otsu thresholding, (img5) adaptive mean threshold, (img6) binary thresholding, (img7) convex hull contour based on img4.

4.2.4 RC Joystick Controllers

For the joystick controller component of this research, two joystick controllers were used, one for the developed multirotor hardware platforms, and the second for the simulation platform. The 9 channel 2.4 GHz Turnigy 9X transmitter configured for mode 2 operation and reflashed with the v2 firmware update, available from HobbyKing (2010), shown in Figure 4.7a, was used as the RC joystick controller for the hardware multirotor platforms. It was selected because of its popularity, ease of setup, ease of use, and cost effectiveness. It comes with a 128×64 LCD display screen for configuration and information display. The Turnigy 9X 9Ch Transmitter comes with the Turnigy RF9X-V2 module and the Turnigy 9X8C-V2 8-channel receiver.



(a) 9 Ch Turnigy RC joystick (HobbyKing, 2010).

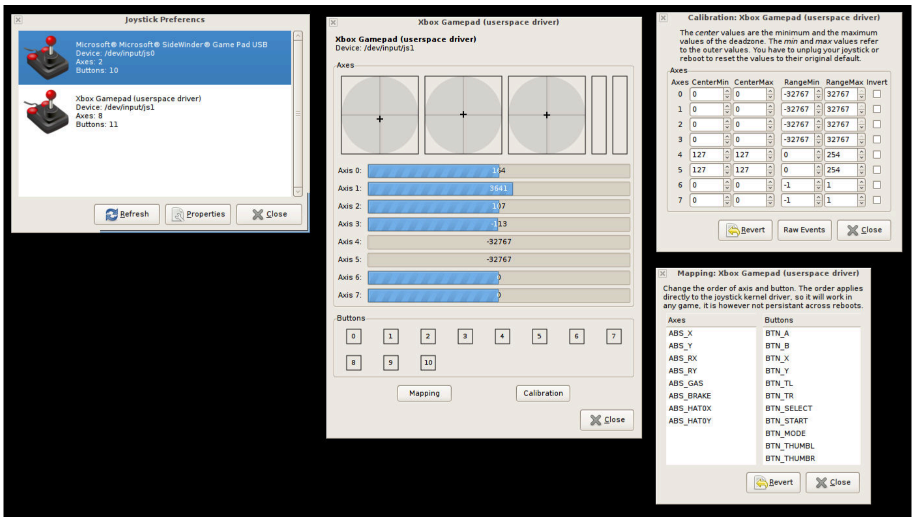


(b) RFDS Int-lnk joystick (Amazon UK, 2016).

Figure 4.7: Turnigy RC and RFDS Futaba Interlink Elite joystick controllers.

The RealFlight Futaba Interlink Elite joystick controller, shown in Figure 4.7b, was used for the hardware-in-the-loop simulation of RC control in the RealFlight Drone simulator (RFDS) and in the ROS Gazebo simulation. The RealFlight Interlink Elite joystick comes with the RFDS software, developed by Hobbico, Inc. et al. (2016) and available from Amazon UK (2016) as "Great Planes RealFlight GPMZ4800 RealFlight Drone with Interlink Elite Mode 2 Edition".

In setting up the RealFlight Futaba Interlink Elite joystick controller to work with the RotorS ROS Gazebo Simulator, the *jstest-gtk* joystick testing and configuration tool for GNU/Linux, developed by Ruhnke (2009), was used. Figure 4.8 shows a screenshot of this utility. The jstest-gtk is based on Gtk+, shows all attached joysticks, visually emulates buttons and axis being pressed, enables axis and buttons remapping, useful for eliminating deadzones, and for joystick calibration. It is recommended to calibrate the joystick limits and check axis inversion before starting the RotorS ROS Gazebo simulation, to avoid abrupt take-off/landing and reverse navigation.



Microsoft® Microsoft® SideWinder® Game Pad USB
Device: /dev/input/js0
Axes: 2
Buttons: 10

Xbox Gamepad (userspace driver)
Device: /dev/input/js1
Axes: 8
Buttons: 11

Xbox Gamepad (userspace driver)
Device: /dev/input/js1

Axis 0: 164
Axis 1: 3641
Axis 2: 107
Axis 3: -113
Axis 4: -32767
Axis 5: -32767
Axis 6: 0
Axis 7: 0

The *center* values are the minimum and the maximum values of the deadzone. The *min* and *max* values refer to the outer values. You have to unplug your joystick or reboot to reset the values to their original default.

Axes	CenterMin	CenterMax	RangeMin	RangeMax	Invert
0	0	0	-32767	32767	☐
1	0	0	-32767	32767	☐
2	0	0	-32767	32767	☐
3	0	0	-32767	32767	☐
4	127	127	0	254	☐
5	127	127	0	254	☐
6	0	0	-1	1	☐
7	0	0	-1	1	☐

Change the order of axis and button. The order applies directly to the joystick kernel driver, so it will work in any game, it is however not persistant across reboots.

Axes	Buttons
ABS_X	BTN_A
ABS_Y	BTN_B
ABS_RX	BTN_X
ABS_RY	BTN_Y
ABS_GAS	BTN_TL
ABS_BRAKE	BTN_TR
ABS_HAT0X	BTN_SELECT
ABS_HAT0Y	BTN_START
	BTN_MODE
	BTN_THUMBL
	BTN_THUMBR

Figure 4.8: jstest-gtk joystick testing and configuration tool for GNU/Linux (Ruhnke, 2009).

4.3 Hardware Platform Development

The development of a physical small multirotor platform for this work was initially considered because the successful realisation of an aerobot controllable via the mSVG technique could be revolutionary. However, due to the difficulty of successfully implementing the physical system within the research time constraint, a hardware-in-the-loop simulation approach was eventually adopted. This section describes the original hardware platform development for the purpose of research continuity and further works. Also the development of a custom platform was preferred to the use of commercial platforms which were either too expensive (like the DJI Matrix 100) or use proprietary flight controller software with limited or no configurable developer options.

4.3.1 ImmersionRC XuGong Multirotor

The first multirotor platform developed was based on the ImmersionRC XuGong V2 Pro quadcopter frame (ImmersionRC, 2014). It was combined with the HKPilot Mega 2.7 flight controller hardware running the Ardupilot APM 2.6 (ArduPilot, 2012) open source flight controller software. The propulsion system consisted of four Turnigy Multistar 20 A electronic speed controllers (ESCs), four 920 KV (RPM/V) MultiStar 2212 motor with 9.4 $\times$ 4.3 inch self-tightening carbon fibre propellers, and a 4S 5200 mAh Multistar Lipo battery. It was setup to be controlled by a 9 channel Turnigy RC joystick controller configured for Mode 2 operation. The developed ImmersionRC based quadcopter is shown in Figure 4.9. The main challenge encountered was in adapting the convoluted APM 2.5 open source flight controller source codes for this research's application particularly with features such as indoor localisation and waypoint navigation.

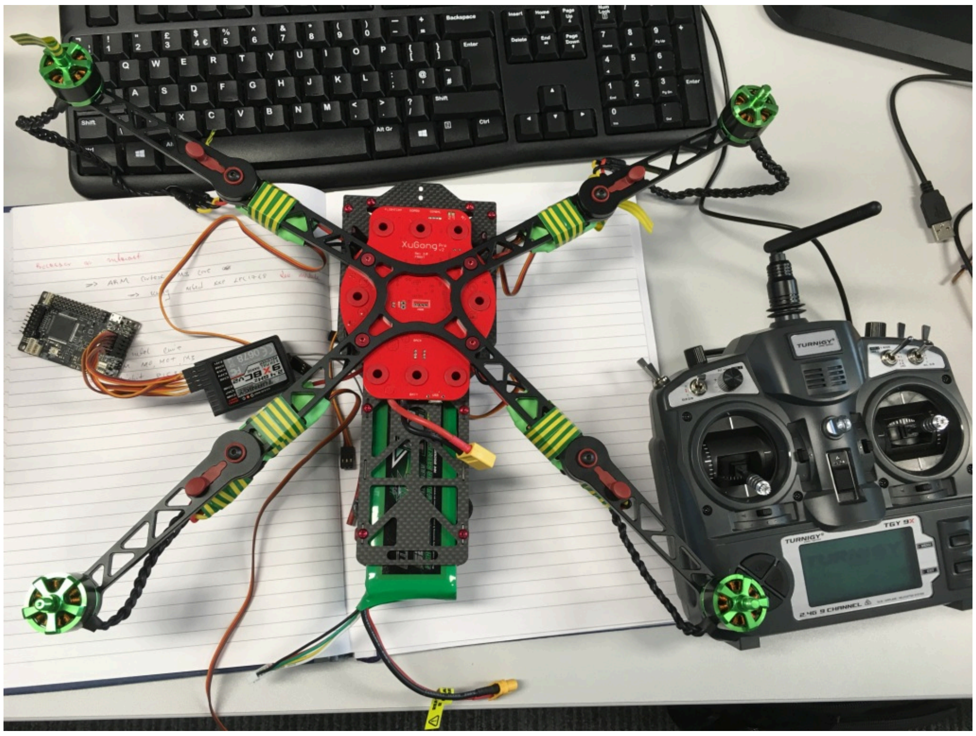


Figure 4.9: Developed ImmersionRC XuGong v2 pro based 450 mm quadcopter.

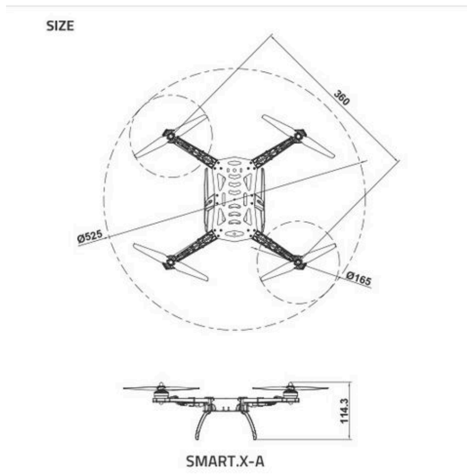
4.3.2 iQuad Jade Multirotor

The second hardware platform considered was the development of a smaller 350 mm frame quadcopter, named the "*iQuad*" project. Unlike the first hardware platform that used the Arduipilot APM 2.6 flight controller firmware, the flight controller firmware for the iQuad project was developed from the ground up.

Frame

For the frame, a range of $250 - 350\,mm$ quad-rotor platform frame with extra compartment space capable of housing two controllers - a microcontroller for flight control and an embedded SBC on-board computer for higher level processing, were considered. The T-Drones Smart X type-A quadcopter frame, shown in Figure 4.10, was selected because it satisfied the space requirement, was low cost, and came with extra useful components. The frame was available from Electricwingman (2016), Goodluckbuy (2016), and Modellbau (2016). The Smart X measures $360\,mm$ from corner to corner and has a maximum take-off weight of $820\,g$. The frame weighs $168\,g$ with no load installed. The arms are $138\,mm$ long and weight about $19\,g$ each.

It comes with four T-Motor AIR 2205 motors, four T-Motor Air $15\,A$ ESCs, and four self-tightening $6.5' \times 3.5'$ inch propellers. The T-Drone frame also comes with a power distribution



(a) Schematics

(b) Frame

Figure 4.10: T-Drones Smart X type-A quadcopter frame schematic (Goodluckbuy, 2016) and top view (Modellbau, 2016).

board for distributing power from the battery to the ESCs and UBEC (Universal Battery Eliminating Circuit). A 1500 mAh 3S Turnigy lithium polymer battery was used. The RC controller used was an eight channel Turnigy TGY 9X RC controller.

Teensy 3.2 Flight Controller Design

Figure 4.11 shows the designed flight controller circuit diagram. Figure 4.12 shows the developed flight controller. The flight controller hardware consisted of the Teensy 3.2 and GY-86 IMU breakout board.

The Teensy 3.2 is an Arduino compatible board which requires the download of a software patch PJRC (2015). The GY-86 IMU breakout board is a 10 DOF (degree of freedom) module with a three-axis gyroscope, a tri-axial accelerometer, a three-axis magnetometer, and an atmospheric pressure barometer. The gyroscope and accelerometer are contain in the MPU6050 chip, the magnetometer is contained in the HMC5883L chip, and the MS5611 chip contains the barometer. Its power supply can range from 3 v to 5 v. It communicates through the I2C serial bus. It is 22 mm long, 17 mm wide. It uses a standard pin spacing of 2.84 mm and has a hole at it top right corner, which is 3 mm wide, for fastening to other surfaces.

The flight controller program was developed in arduino using the C++ program language, designed to be operated via the 9 channel 2.4 GHz Turnigy 9X mode 2 transmitter shown in Figure 4.7a. The full program listing is presented in Appendix C.1. The flight controller program loops at 450 Hz, refreshing the IMU values at the same rate. The flight test video is included as

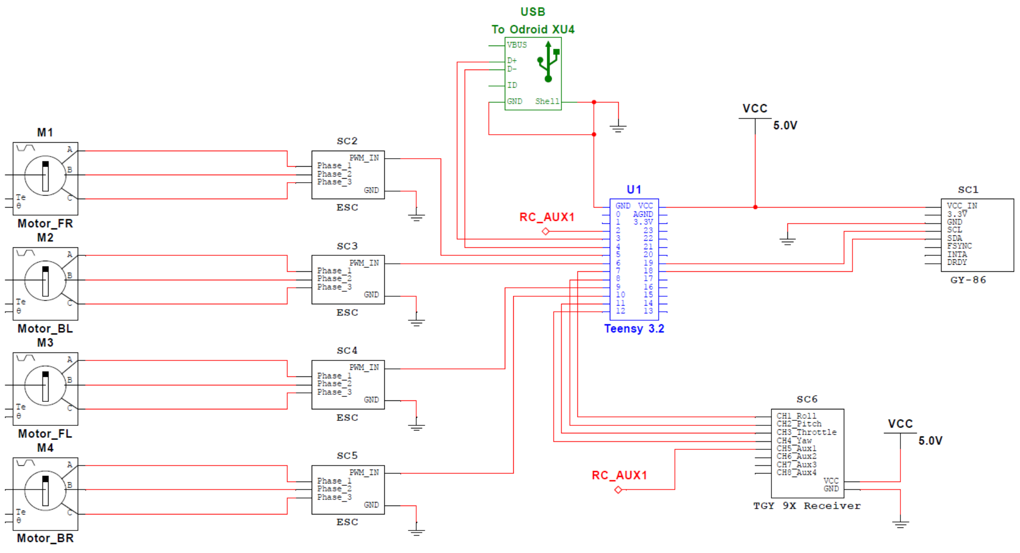


Figure 4.11: A Multisim circuit diagram showing the design of the flight controller.

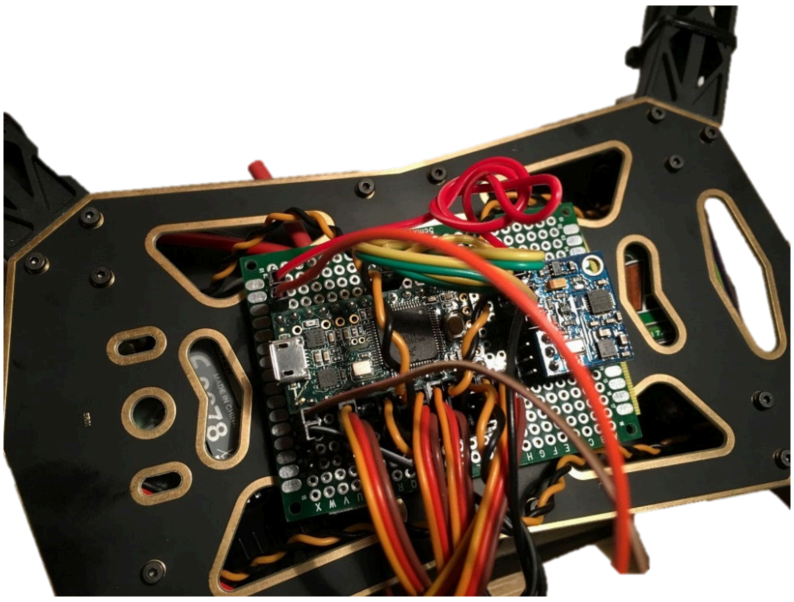
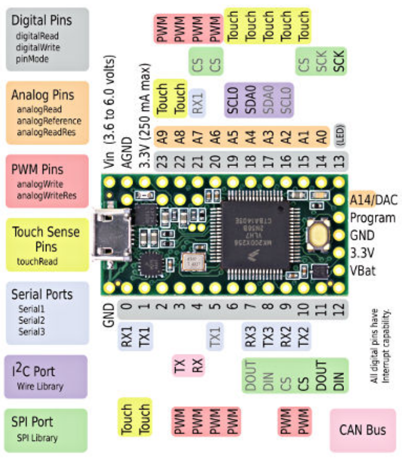
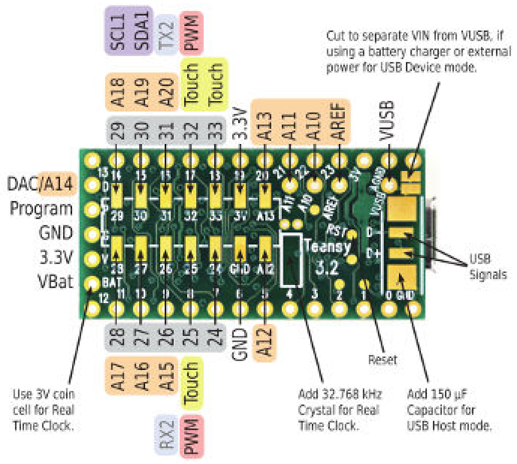


Figure 4.12: The developed and programmed flight controller from the Teensy 3.2 and GY-86 boards.

supplementary multimedia data with this thesis. The program was inspired by the open source Arducopter project with the initial aim of customizing the codes for the ArduPilot Mega 2.5 (APM 2.5) board which uses the AVR Atmega2560 microcontroller and the MPU6000 6-DoF Accel-Gyro and HMC5883L 3-DoF Compass. But due to the complexity of the APM project development, we decided to develop a simpler compact single file flight controller program to work with our custom developed flight controller hardware.

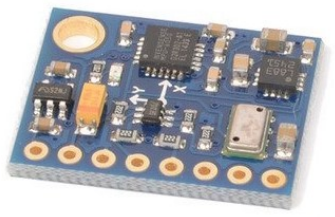


(a) Top pin out

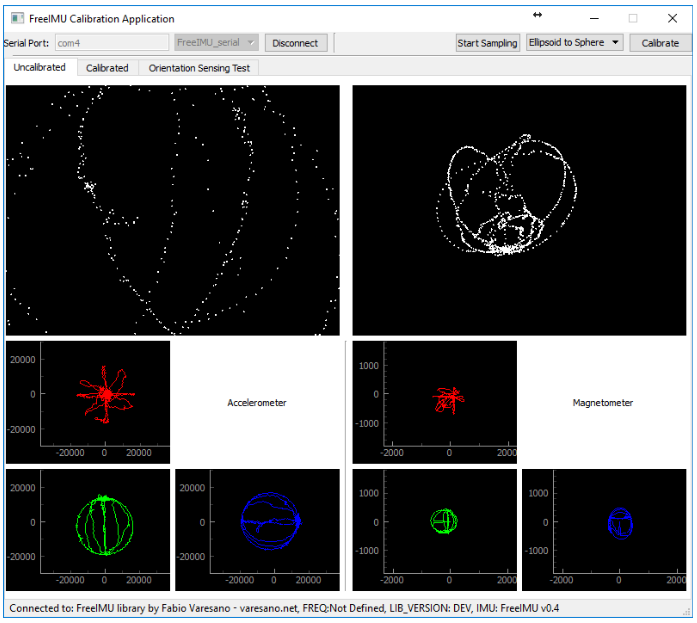

SCL1
SDA1
TX2
PWM
Touch
A18
A19
A20
3.3V
A13
A11
A10
AREF
VUSB
DAC/A14
Program
GND
3.3V
VBat
Cut to separate VIN from VUSB, if using a battery charger or external power for USB Device mode.
USB Signals
Teensy 3.2
A17
A16
A15
GND
A12
RX2
Use 3V coin cell for Real Time Clock.
Add 32.768 kHz Crystal for Real Time Clock.
Reset
Add 150 µF Capacitor for USB Host mode.

(b) Bottom pin out

Figure 4.13: Teensy 3.2 top and bottom pin out diagram (PJRC, 2015).



(a) GY-86 10 DOF module.



Serial Port: com4 FreeIMU_serial
Ellipsoid to Sphere

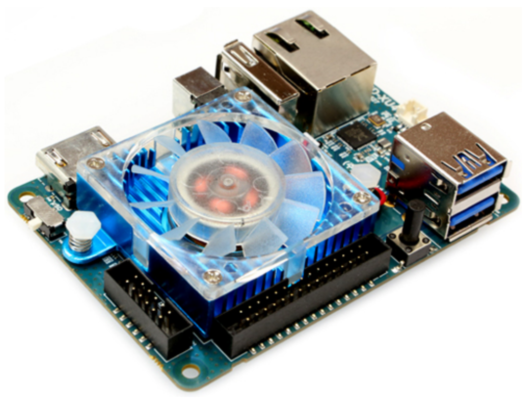
Connected to: FreeIMU library by Fabio Varesano - varesano.net, FREQ:Not Defined, LIB_VERSION: DEV, IMU: FreeIMU v0.4

(b) GY-86 Calibration.

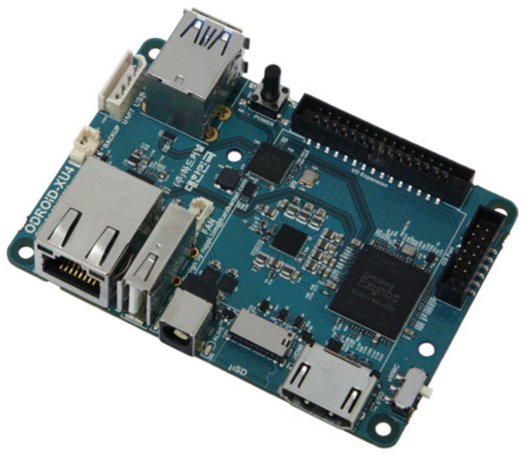
Figure 4.14: GY-86 DOF module accelerometer and magnetometer calibration.

Single Board Computer

A second on-board computer, to support the primary flight controller (Teensy 3.2 microcontroller), in performing higher level functions of speech and gesture processing was needed. The Linux compatible Odroid XU4, shown in Figure 4.15, was used. At the time of the iQuad platform development, the Odroid XU4 was one of the most recent single board computers, others being the NVidia Jetson Tx1, Raspberry Pi 3, and Intel Edison boards. Although, the NVidia Jetson TX1 running Ubuntu 16.04 L4T ("Linux for Tegra" or "Linux4Tegra") version 24.2.1, was considered, the Odroid XU4 was selected because it had more processing power than the Raspberry Pi 3, it was more cost effective than the NVidia Jetson Tx1, and it does not require additional modules like the intel edison boards. The Odroid XU4 supports heterogeneous multi-processing (HMP). It is more powerful and more energy efficient than the XU3 before it, and has a small form factor - approximately $83 \times 58 \times 22\ mm$. It is compatible with Ubuntu 16.04 Mate, and Android 4.4 KitKat, 5.0 Lollipop, and 7.1 Nougat. It can be booted from a regular memory card or from the eMMC 5.0 HS400 Flash Storage (which is faster and hence highly recommended). It has two USB 3.0 port, one USB 2.0 port, and a Gigabit Ethernet interface. It has eight CPUs - four Samsung Exynos5422 Cortex-A15 running at 2 GHz (quad-core) and four Cortex-A7 running at 1.4 GHz (quad-core). It uses the Mali-T628 MP6 (OpenGL ES 3.0/2.0/1.1 and OpenCL 1.1 Full profile) for graphics. It has 2 Gigabyte LPDDR3 RAM PoP stacked. It displays output through an HDMI 1.4a port. It shipped with Linux Kernel 4.14 LTS and the Ubuntu 16.04 MATE installed.



(a) XU4 with cooling fan



(b) XU4 without cooling fan

Figure 4.15: Odroid XU4 embedded computer compatible with Ubuntu 16.04 Mate Linux (Hardkernel, 2015)

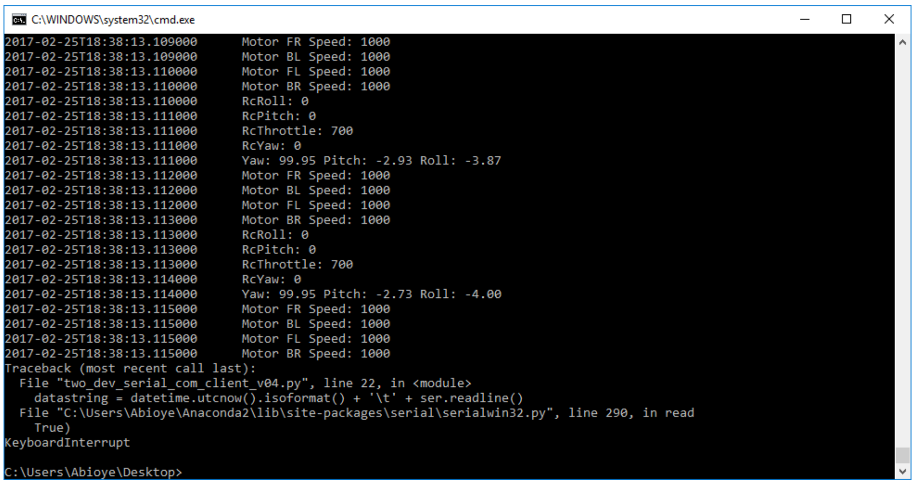
The Odroid XU4 SBC was coupled to the Teensy 3.2 flight controller via USB and serial UART Tx/Rx communication protocol was used to send control symbols from the SBC to the flight controller for differential motor control simulation of horizontal 2D navigation, at 115200 baud rate. A Logitech combo MK270 2.4 GHz wireless keyboard and optical mouse set was connected to Odroid XU4 SBC for controlling applications on the SBC's Linux operating system. The keyboard was also used in generating overriding control symbols, listed in Table 4.1, to remotely control the iQuad Jade multirotor platform. The Kinobo USB microphone was also connected to Odroid XU4 SBC, for capturing speech control commands, which were then processed into the control symbols listed in Table 4.1, and then passed on to the iQuad Jade multirotor for differential motor control emulating the control input.

Listing 4.4: Flight controller modified code with serial speech and keyboard control input symbol.

```
// CODEMARK II:  Keyboard/Serial Input Segment
// CODEMARK II:  Speech/Serial Input Segment via Odroid XU4

// serial bytes available?
int bytesAvail = Serial.available();
if (bytesAvail > 0) {
  while (bytesAvail > 0) { //yes
    char c = (char)Serial.read();  //read next byte

    if (c == '\n') {             // new line reached - process cmd
      buf[buf_offset] = '\0';     // add null terminator
      String str = String(buf);

      if (str == "1000") // halt horizontal motion - level quadcopter by setting roll and pitch
          angles to zero
      {
        rcpit = 0;
        rcroll = 0;
      }

      if (str == "1011") // forward - drive rear motors faster and front motors slower by
          kbd_cursor_constant amount
      {
        //channels[1] += kbd_cursor_constant;        // modify pitch
        //rcpit += kbd_cursor_constant;        // modify pitch
        rcpit = speech_movement_constant;      // banks 30 degrees in the direction of motion
        rcroll = 0;
      }

      if (str == "1012") // back - drive front motors faster and rear motors slower by
          kbd_cursor_constant amount
      {
        //channels[1] -= kbd_cursor_constant;        // modify pitch
        //rcpit -= kbd_cursor_constant;        // modify pitch
        rcpit = -speech_movement_constant;      // banks -30 degrees in the pitch direction
        rcroll = 0;
      }

      if (str == "1013") // right - drive left motors faster and right motors slower by
          kbd_cursor_constant amount
      {
        rcpit = 0;
        rcroll = speech_movement_constant;      // banks roll angle to +30 degrees
      }

      if (str == "1014") // left - drive right motors faster and rear motors slower by
          kbd_cursor_constant amount
      {
        rcpit = 0;
        rcroll = -speech_movement_constant;      // banks roll angle to -30 degrees
      }
      kbd_timeout_start = micros();
      buf_offset = 0;
    }
    else if (c != '\r') {
      buf[buf_offset++] = c;    // store in buffer and continue until newline
    }
    bytesAvail --;
  }
}
```

The original C++ program code listed in Appendix C.1 for RC joystick control, was modified to enable the keyboard and speech serial input control symbol execution on the iQuad Jade UAV platform. Listing 4.4 shows a code snippet of modifications to the original code. The modified full code is listed in Appendix C.1.3.

With the aid of a wireless IEEE 802.15.4 Zigbee module, a remote data-logging and debugging system was developed. Python based command line utility was developed to run on both Windows and Linux operating systems, to enable remote monitoring of the iQuad Jade platform and ease troubleshooting of its operations. Figure 4.16 shows the debugger running on the Windows operating system.

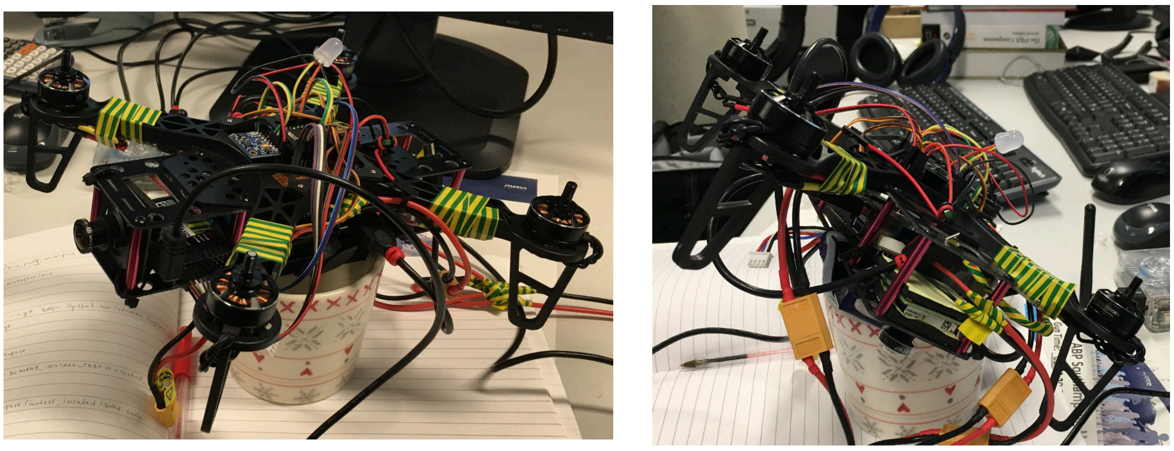


```
2017-02-25T18:38:13.109000      Motor FR Speed: 1000
2017-02-25T18:38:13.109000      Motor BL Speed: 1000
2017-02-25T18:38:13.110000      Motor FL Speed: 1000
2017-02-25T18:38:13.110000      Motor BR Speed: 1000
2017-02-25T18:38:13.110000      RcRoll: 0
2017-02-25T18:38:13.111000      RcPitch: 0
2017-02-25T18:38:13.111000      RcThrottle: 700
2017-02-25T18:38:13.111000      RcYaw: 0
2017-02-25T18:38:13.111000      Yaw: 99.95 Pitch: -2.93 Roll: -3.87
2017-02-25T18:38:13.112000      Motor FR Speed: 1000
2017-02-25T18:38:13.112000      Motor BL Speed: 1000
2017-02-25T18:38:13.112000      Motor FL Speed: 1000
2017-02-25T18:38:13.113000      Motor BR Speed: 1000
2017-02-25T18:38:13.113000      RcRoll: 0
2017-02-25T18:38:13.113000      RcPitch: 0
2017-02-25T18:38:13.113000      RcThrottle: 700
2017-02-25T18:38:13.114000      RcYaw: 0
2017-02-25T18:38:13.114000      Yaw: 99.95 Pitch: -2.73 Roll: -4.00
2017-02-25T18:38:13.115000      Motor FR Speed: 1000
2017-02-25T18:38:13.115000      Motor BL Speed: 1000
2017-02-25T18:38:13.115000      Motor FL Speed: 1000
2017-02-25T18:38:13.115000      Motor BR Speed: 1000
Traceback (most recent call last):
  File "two_dev_serial_com_client_v04.py", line 22, in <module>
    datastring = datetime.utcnow().isoformat() + '\t' + ser.readline()
  File "C:\Users\Abioye\Anaconda2\lib\site-packages\serial\serialwin32.py", line 290, in read
    True)
KeyboardInterrupt

C:\Users\Abioye\Desktop>
```

Figure 4.16: Developed data-logging and debugging serial command line utility for monitoring iQuad Jade platform.

4.3.3 Unicorn Multirotor

Due to slow development progress on the iQuad Jade multirotor platform, an alternative Unicorn multirotor platform was considered. The Unicorn multirotor UAV was an indoor navigation platform being developed at the autonomous systems laboratory at the University of Southampton by Liu et al. (2016a) based on a fast semi-direct monocular visual odometry (SVO) localisation techniques described by Forster et al. (2014). It was a $250 \times 250\,mm$ span UAV that uses the Teensy 3.1 and Odroid XU3 as flight controller and on-board computer, and a global shutter IDS uEye UI-1221LE-M-GL grey scale camera and an ultrasonic sensor for indoor localisation, in order to hover at set position and altitude. We considered adopting the indoor navigation technique proposed by Liu et al. (2016a) in order to augment our custom developed UAV, but this integration proved too challenging and derailing to this research's aim and objectives that we decided to consider the more realistic and practical simulation alternative, which was eventually setup and used in completing this research work.



(a) Isometric front-view showing gesture capture camera (b) Rear-view showing SVO localisation camera

Figure 4.17: Developed Unicorn-based quadcopter UAV.

Figure 4.17 shows the developed Unicorn-derived multirotor platform, which was modified to accommodate this research's requirement. The modified Unicorn platform had a front facing UI-1221LE-C-HQ colour USB camera installed to capture visual gesture images, a downward facing UI-1221LE-M-GL grey scale USB camera for localisation and waypoint navigation, a Kinobo USB microphone for speech capture, and a more powerful Odroid XU4 embedded computer running a more recent Ubuntu 16.04 Mate Linux distribution, for high-level computation processing of the speech, gesture, localisation, and waypoint navigation data input.

4.4 Hardware-in-the-loop software simulation

Two hardware-in-the-loop simulation software were used in this research. The first was the commercial RealFlight Drone Simulator (RFDS), which was based on proprietary software, was used to estimate small multirotor UAV pilots RC joystick flying skill level. The second was the open source RotorS ROS Gazebo simulator, which was used in comparing control interface performances.

4.4.1 RealFlight Drone Simulator (RFDS)

The RealFlight Drone Simulator (RFDS) was developed by Hobbico, Inc. et al. (2016) and was available from Amazon UK (2016). It was derived from the RealFlight 7.5, sold as a limited edition of the 7.5 without the airplanes and helicopter aircraft – just the multirotor drone aircraft. The RealFlight drone edition simulator came with a software CD and an Interlink Elite joystick controller developed by Futaba – the hardware-in-the-loop component. Unlike in a physical UAV platform, the cost of crashing a UAV or the consequences of a UAV accident was completely eliminated in the hardware-in-the-loop RFDS software simulation test of RC joystick controller flying skill level.

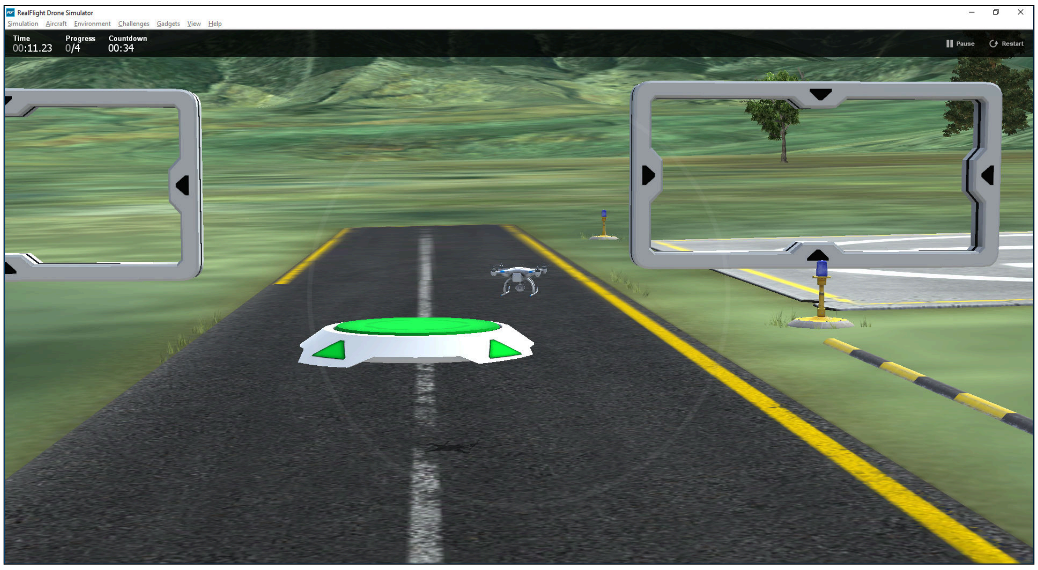

RealFlight Drone Simulator
Simulation Aircraft Environment Challenges Gadgets View Help
Time 00:11.23
Progress 0/4
Countdown 00:34
Pause
Restart

Figure 4.18: RFDS challenge Level 5.

The RFDS level challenge tasks were a combination of countdown-time-based manoeuvres such as flying through a gate, landing on a touchpad, etc. at the end of which a score was assigned based on how quickly the tasks were completed. Figure 4.18 shows a screenshot of an RFDS level 5 challenge being attempted. The maximum number of attempts per level per participant were capped at 7, beyond which the participant does not proceed to subsequent challenge levels, if not successful. The complete RFDS challenge levels were:

1. Fly through the square gate ahead.
2. Fly through the curved path and then land on the circular touchpad ahead.
3. Fly through the gate and then land on the touchpad.
4. Fly through the path, hit the touchpad, fly through the gate, then land on the second touchpad.
5. Fly through the path, hit the touchpad, fly through the left raised gate, then through the right raised gate, and then land back on the starting touchpad.

6. Hit a series of five touchpad in order, starting and finishing on the middle touchpad. Four touchpads arranged in a circle around a middle fifth touchpad.

7. Fly through the four slalom staggered gates in order.

8. Fly through the four gate curve, then downward through the horizontal gate, and land on the touchpad below the horizontal gate.

9. Hit the right touchpad, fly sideways through the raised gate, then hit the left touchpad on the ground, and fly back upwards and sideways through the raised gate, and then land back on the right touchpad.

10. Complete the combine obstacles and courses from challenges 7, 8, and 9 in reduced time.

The RFDS simulator was chosen for testing users RC flying skill level because it had a fully developed set of practical and realistic flight level challenges, which were sufficient for the test in this research. The RotorS ROS Gazebo simulation did not have this flight challenges, and would have required significant design and development efforts to successfully implement this; and this was not a primary focus of this research.

4.4.2 ROS Gazebo Simulation

Due to the fact that the RFDS simulator was a commercial product based on proprietary software, for which the source code was not available to developers, open source ROS Gazebo UAV simulators were considered. The *hector_quadrotor* ROS Gazebo UAV simulator, which was developed by Meyer et al. (2012), and had support for both indoor and outdoor navigation scenarios, as shown in Figure 4.19, was the first simulator considered. However, the hector_quadrotor documentation was lacking in details, which made it implementation and setup challenging.

The second major ROS Gazebo UAV simulator considered was the *RotorS* simulator, which was developed by Furrer et al. (2016). The RotorS Ros Gazebo UAV simulator was more recent than the hector_quadrotor, was currently being supported and updated by the developers, was compatible with the latest Ubuntu 16.04 LTS Linux distribution and the ROS kinetic Kame used at the time of this study, and was better documented. Therefore, the RotorS ROS Gazebo UAV simulator was chosen over the hector_quadrotor, and was successfully developed and used in this research. The RotorS ROS Gazebo simulator has a model of the Ascending Technologies (2013) AscTec Firefly hexacopter UAV, which was used to emulate the small multirotor platform navigation in a custom navigation path developed in this research study. Figure 4.20 shows the RotorS ROS Gazebo simulator executing a developed python-based waypoint navigation

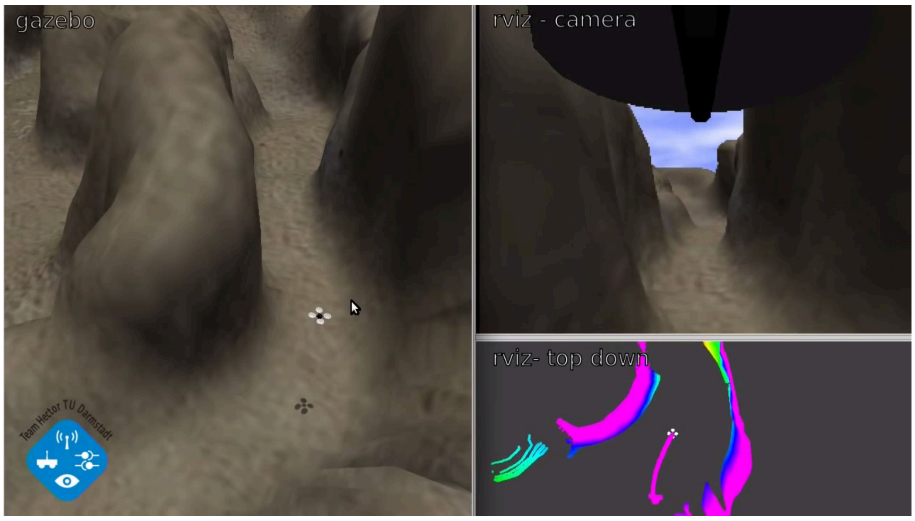


Figure 4.19: Hector_quadrotor outdoor scenario demo (Meyer et al., 2012; Meyer and Kohlbrecher, 2012).

program, which was computing and passing GPS-like navigation coordinates and speed to the RotorS physics framework for simulation.

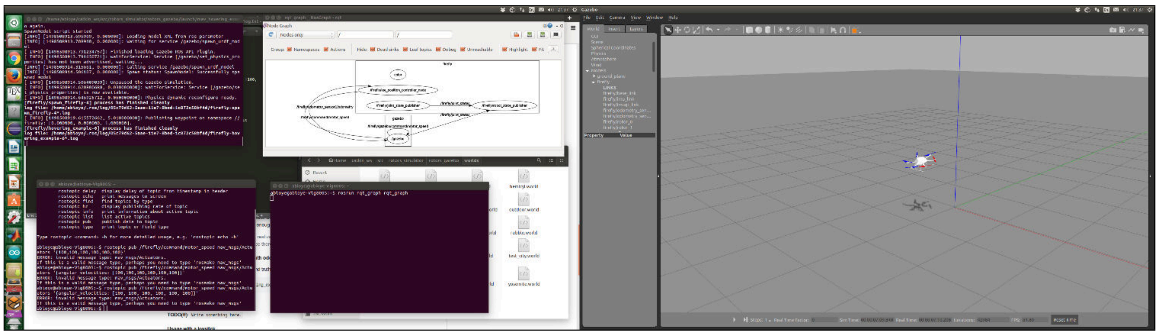


Figure 4.20: Setting up RotorS (Furrer et al., 2016) ROS Gazebo Simulator.

For emulating the joystick controller in Gazebo, the RealFlight Futaba Interlink Elite joystick controller was used as the input joystick controller hardware. The setup/startup procedure is described in Table 4.3.

The hardware for running the ROS Gazebo UAV simulator was a mainstream Viglen desktop computer with two Iiyama-B2282HD monitors. The computer had an Intel Core i5-7500 Quad-core 3.40 GHz Processor, with integrated Intel HD Graphics 630 (1100 MHz), an 8 GB 2400 MHz PC4-19200 DDR4 RAM Memory, and a 1 TB 7200 RPM Hard disk drive. The computer was running the Linux Ubuntu 16.04 LTS desktop operating system, and had the ROS Kinetic Kame distribution for Ubuntu Xenial (16.04 LTS) and Gazebo version 8 installed.

Table 4.3: Joystick control ROS Gazebo simulation startup.

A. Hardware components setup:

1. Startup the Linux desktop computer and boot to the default desktop graphical user interface
2. Connect the RealFlight Futaba Interlink Elite USB joystick controller to the Linux desktop
3. Run jstest-gtk (install if necessary):

```
#if installation is needed
$ sudo apt-get install jstest-gtk

# starts the joystick configurator
$ jstest-gtk
```

B. RotorS simulation:

1. Open a new Terminal, "*Ctrl + Alt + T*"
2. Open "home/catkin_ws/" folder in the Terminal using,

```
$ cd catkin_ws/
```

3. Load setup.bash, and start ROS,

```
$ source devel/setup.bash
$ roscore
```

4. In a new terminal window, start RotorS mavlink based joystick control nodes in basic world with the firefly UAV,

```
$ roslaunch rotors_gazebo mav_with_joy.launch mav_
    name:=firefly world_name:=basic
```

4.5 Experiment Study Design

In order to address the research questions in Section 1.5 and to investigate the proposed hypothesis in Section 1.6, an experiment study was designed. The study was divided into two major parts - study A and study B - with each part addressing one of the two major research questions. Study A was interested in the practical usage limit of the components of the proposed speech and visual gesture interface in the control of small multirotor UAVs. It investigated how the noise levels generated by multirotor UAV propulsion systems affects speech control input. It also investigated how different backgrounds and lighting levels affects gesture control input. Study B was interested in comparing the performance and cognitive workload of the proposed mSVG interface with the standard RC joystick controller via an nCA Tier 1-III navigation control task.

The study was designed to involve human participants, interacting with hardware components whose output responses are simulated and displayed to the study participant via computer monitors. The experiment was conducted with the aid of two computer-based UAV simulators, interacting with other external hardware components such as the joystick controller, single-board computer, web camera, microphone, speaker, and multi-colour (white and yellow) variable LED lighting systems. These additional hardware components were laid out as shown in Figure 4.21. Unlike typical pure simulation experiments, the hardware-in-the-loop experiment enabled practical interaction with real world components, making the results more generalisable to practical real world applications. The study participants were mostly sited in front of the three-screen UAV simulation computer workstation, during which the participant were asked to perform a series of task.

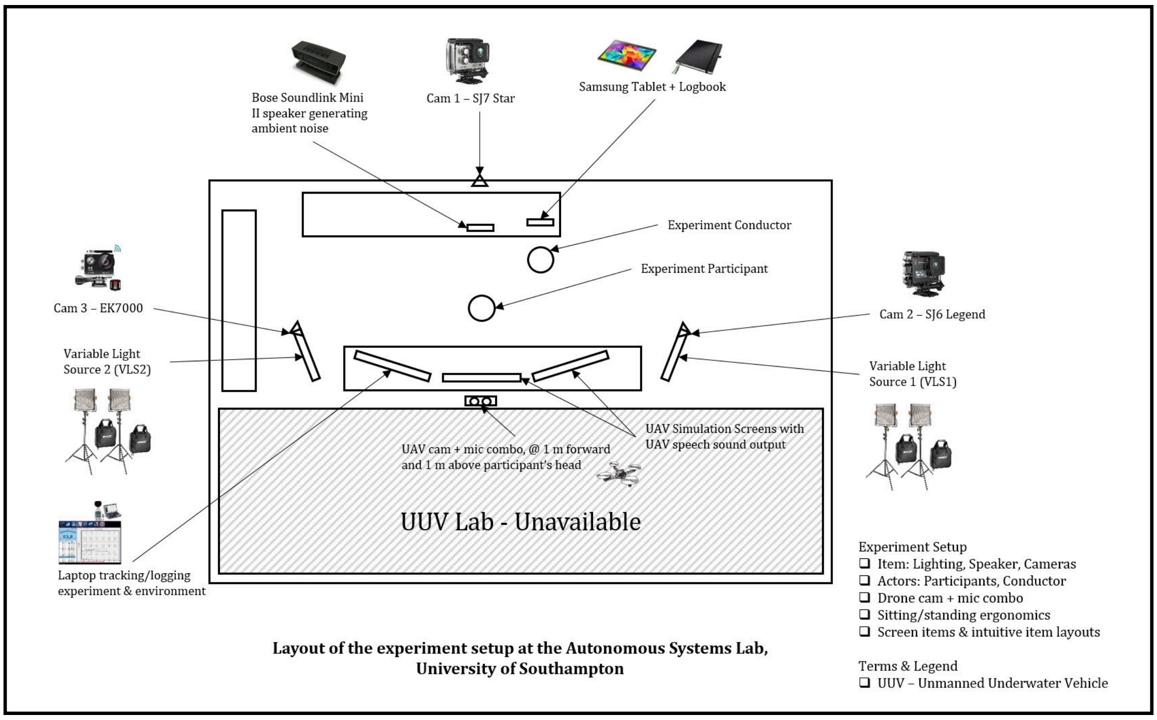

Bose Soundlink Mini II speaker generating ambient noise
Cam 1 – SJ7 Star
Samsung Tablet + Logbook
Experiment Conductor
Experiment Participant
Cam 3 – EK7000
Cam 2 – SJ6 Legend
Variable Light Source 2 (VLS2)
Variable Light Source 1 (VLS1)
UAV Simulation Screens with UAV speech sound output
UAV cam + mic combo, @ 1 m forward and 1 m above participant's head
UUV Lab - Unavailable
Laptop tracking/logging experiment & environment
Experiment Setup
❑ Item: Lighting, Speaker, Cameras
❑ Actors: Participants, Conductor
❑ Drone cam + mic combo
❑ Sitting/standing ergonomics
❑ Screen items & intuitive item layouts
Layout of the experiment setup at the Autonomous Systems Lab, University of Southampton
Terms & Legend
❑ UUV – Unmanned Underwater Vehicle

Figure 4.21: Experiment Setup Layout Diagram.

4.5.1 Study A design

Study A was divided into two sub-parts with the first part focused on speech and the second part on lighting and background conditions.

Varying ambient noise levels

Speech was one of the components of the proposed mSVG aerobot control interface, hence the need to determine it practical limitation for the proposed usage. This was done by introducing noise. Two methods of introducing noise were considered – after or during speech capture. The first was to generate a noise profile using a python program, to corrupt the captured speech recording before it is being passed to the ASR for processing. Alternatively, a pre-recorded noise such as stormy weather and multirotor propulsion noises could be overlaid on captured speech to corrupt it before processing, using audio editing software to vary the noise level before the combination. The second was to physically generate the noise level in the lab, so that the captured speech recording was already corrupted by noise at the point of capture, after which it is passed on to the ASR speech recogniser. The second method was chosen because it was more practical and a better representation of typical real life applications.

Table 4.4: Noise levels generated by some small multirotor UAVs (Levin, 2017).

S/N	Small Multirotor UAV	Noise Levels (dB)
1	DJI Phantom 2	75.8
2	DJI Phantom 3 Pro	76.3
3	DJI Phantom 4 pro	76.9
4	DJI Inspire 2	79.8
5	Hover Cam	72.1
6	DJI Mavic Pro	~ 65
7	DJI Mavic Pro Platinum	~ 60

In order to determine what noise levels was to be used in the experiment, the average noise level generated by practical multirotor UAV propulsion system was considered. Islam et al. (2016) and Levin (2017) conducted experiments to measure the noise level generated by small UAVs. In Levin (2017) experiments, five small UAVs were tested by flying the UAVs to a 1 m altitude, and placing a soundmeter 1 m adjacent to the UAV. Table 4.4 summarises Levin (2017) observation. From this result, and for the purpose of this study, the noise level generated by the small multirotor UAV was assumed to be approximately 80 decibels. Therefore, by considering the practical noise levels generated by real small multirotor UAVs, the limitation of the implemented ASR speech system ($\sim$ 85 dB practical ASR upper limit) and the ambient laboratory noise level ($\sim$ 55 dB quiet laboratory conditions), the noise level range of 55 dB to 85 dB, was selected.

It was also noted that sound reduces at a rate of 6 dB for every doubling of distance from a noise source (Collman, 2014). Therefore, if a DJI phantom 2 generates 75.8 dB of noise at 1 m, then it would be 69.8 dB at 2 m, 81.8 dB at 0.5 m, and 87.8 dB at 0.25 m. This information was used to ensure that the point of noise measurement was at the microphone where both speech and noise were being captured/recorded.

Table 4.5: List of speech commands uttered during noise level experiment.

S/N	**Speech Command**
1	Go Forward
2	Go Backward
3	Step Left
4	Step Right
5	Hover
6	Land
7	Go Forward Half Meter
8	Go Backward One Meter
9	Hover One Meter
10	Step Left Half Meter
11	Step Right One Meter
12	Stop

Thefore, the first task was to measure the effect of varying noise levels from 55 dB to 85 dB in steps of 5 dB, being generated from a Bose Sound link mini speaker system, playing a pre-recorded loop of a multirotor UAV propeller-rotor noise sound. The experiment participants were asked to repeat a series of 12 speech commands made from a selection of words in the iQuad speech vocabulary list in Table 4.2. The commands were *"go forward, go backward, step left, step right, hover, land, go forward half metre, go backward one metre, hover one metre, step left half metre, step right one metre, and stop"*, as shown in Table 4.5. The first six speech commands were selected because they were essential UAV navigation command that specify fundamental horizontal motion in the x and y coordinate direction at a fixed z altitude/vertical coordinate. The seventh to eleventh command were randomly selected commands used to emulate practical x, y, and z coordinate navigation were short distance (less than 3 m) modifiers were applied. The twelfth command was selected based on practical consideration for a fail-safe/emergency/urgent command phrase that halts any current motion action. These commands are issued at quiet laboratory conditions of 55 dB, and then repeated for 60 dB, 65 dB, 70 dB, 75 dB, 80 dB, and 85 dB noisy laboratory conditions. The results are presented and discussed under Section 7.1 and Section 7.2.

Varying lighting levels and background conditions

Visual gesture was the second component of the proposed mSVG aerobot control interface, and similar to the first component (speech), its practical usage limits was also needed to be determined. This was done by varying lighting levels and background conditions to determine how different visibility conditions affects visual gesture capture and recognition. Similar to the speech component, two methods of introducing visual noise were considered – after or during gesture capture. The first was to add Gaussian or other white noise to the originally captured gesture image before processing the image for gesture recognition. Alternatively, an image capture of an outdoor or indoor environment such as poor visibility in a stormy weather or snow precipitation conditions could be edited and made the background of the captured gesture image before being passed on for gesture recognition, by using a photo editing software to vary the combination properties such as clarity level, brightness level, contrast level, transparency level, etc. The second was to physically vary the visibility conditions of the capture environment by varying the room lighting levels and background conditions. In this case, the visual gesture image being captured would naturally include the ambient lighting conditions. Similar to the speech limit experiment design, the second method was chosen for visual gesture image capture visibility corruption, because it was more practical and a better representation of typical real life applications.

From measurement, the limits of the LED lighting system provided generated a maximum visibility of ∼1400 Lux (Overcast Day). Table 4.6 shows the practical lighting level ranges. When the laboratory and LED lighting systems were turned off, the ambient visibility level measured was ∼10 Lux (Twilight). This was due to the ambient stray light rays from the workstation monitors and light reflection bouncing off corridor walls into the lab. Note that the lux meter measurement was held vertically next to seated participants raised right hand (finger gesture hand), facing the camera just like the experiment participant's hand. The LED lighting system had two lighting colour temperatures – white (5500 K) and yellow (3500 K), which were used separately and in combination to additionally vary the room lighting conditions. Three solid background colours – white, green, and navy blue were also used to further expand ambient gesture capture conditions. The solid backgrounds were chosen to simplify gesture processing.

Table 4.6: Lighting levels description.

S/N	Light Levels (Lux)	Description
1	0.0001	Overcast Night
2	0.001	Starlight
3	0.01	Quarter Moon
4	0.1	Full Moon
5	1	Deep Twilight
6	10	Twilight
7	50	Living room, hallway, toilets
8	100	Very Dark Day
9	500	Office lighting
10	1,000	Overcast Day, TV studio
11	10,000	Daylight
12	100,000	Sunlight

The green background was chosen because of its popularity in computer generated graphics, and it is a primary colour. The navy blue background was used because it is a dark shade of the primary colour, blue. White was chosen because of its popularity in photography, and its consideration as a neutral background colour.

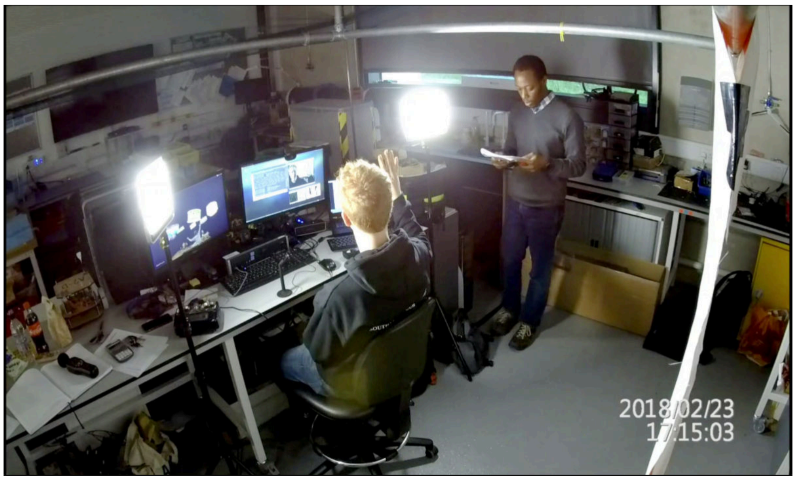


Figure 4.22: Conducting the experiments - visibility testing.

Therefore, for the second part of study A experiment, focusing on how varying ambient lighting conditions and changing background affects gesture recognition quality, the procedure was to divide the experiment into a sequence of nine lighting stages (LS1 - LS9). The default room lighting was turned off, all window blinds closed, and the main lighting source was the LED lamp with two colour LEDs - white (5500 K) and yellow (3500 K), as shown in Figure 4.22. For the first light stage, LS1, the LED Lamp were turned off and ambient stray light rays from workstation monitors, of around 10 Lux was measured and recorded against each of the solid coloured finger gesture capture background surface area. The green background was setup first, and the user asked to hold up the following finger gestures in order: 'one finger', 'two fingers', 'three fingers', 'four fingers', and 'five fingers'. The background qualities are estimated based on how distinct the finger gestures were clearly recognised using a numeric scale of 1-10, with '1' being a complete failure in gesture recognition, '3' being the hand outline was successfully registered, '5' being all finger gestures being successful but with high frequency noise fluctuations, and '7' being all fingers were clearly distinguished but with small low frequency fluctuations (one in 10 seconds), and '10' being perfect steady recognition, with no noise fluctuations within 60 seconds. This was repeated for the blue and white background. And then the second lighting stage, LS2 experiment was performed, turning only the white lighting knob to the first indicator point on the right LED lamp, and repeating the capture procedure described for LS1. Note that all other knobs were reset to zero. For LS3, the white lighting knob was turned to the first indicator point on both the right and left LED lamps. All the white knobs position were then reset back to zero before proceeding to LS4. For LS4, the yellow lighting knob was turned to the first indicator point on the right LED lamp. For LS5, the yellow lighting knobs were turned to the first indicator point on both the right and left LED lamp. For LS6, the white and yellow lighting knobs were turned to the first indicator point on both the right and left LED lamps. After completing LS6 capture, the yellow lighting knobs were reset back to zero. For LS7, the white lighting knob was turned to the maximum position on both the right and left LED lamp. After the LS7 capture, all white lighting knobs were reset back to zero. For LS8, the yellow lighting knobs were turned to the maximum position on both the right and left LED lamp. And finally, for LS9, both the white and yellow lighting knobs were turned to the maximum position on both the right and left LED lamps. For each of these LS settings, the capture procedure described for LS1 was repeated. Note that LS2, LS3, and LS7 were the white lighting capture experiment stages. LS4, LS5, and LS8 were the yellow lighting experiment stages. LS1, LS6, and LS9 were the mixed white and yellow lighting experiment stages. The results were presented and discussed in Section 7.3.

4.5.2 Study B design

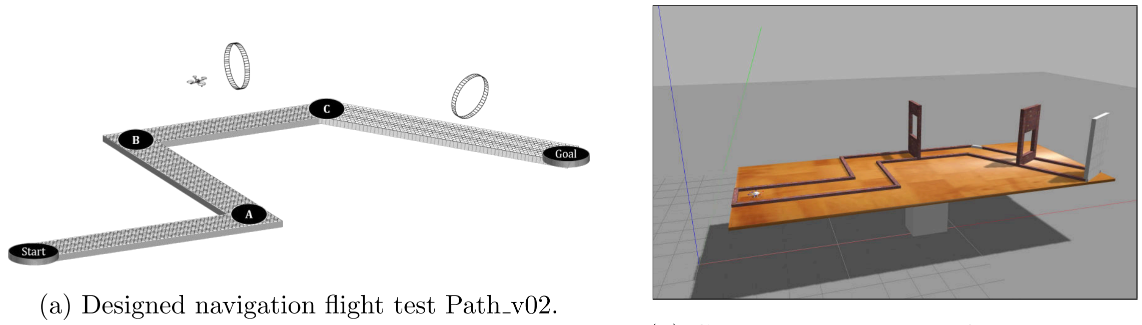
Study B experiment was designed to measure and compare the performance and cognitive workload of the mSVG and RCJ interfaces. The cognitive workload was measured with the aid of the NASA Task Load Index (TLX) survey questionnaire (Hart and Staveland, 1988; Hart, 2006), and the performance compared in terms of how quickly and how accurately the navigation task was completed. The task performed was an nCA Tier 1-III level task, which was considered a balance task for both mSVG and RCJ, because any lower would benefit the RCJ and any higher would benefit the mSVG. The study was conducted on the computer-based ROS Gazebo UAV simulator, augmented with external hardware-in-the-loop components (single-board computer, joystick controller, camera, microphone, speaker, and lighting system), in order to interact with the physical world. The study participants were mostly sited in front of a three-screen UAV simulation computer workstation, during which the participant were asked to perform two major tasks – Task B1 and Task B2. Task B1 measured the participants RCJ skill level using the commercial RealFlight Drone Simulator (RFDS) level challenges. Task B2 measured the time taken to complete the Path_v02 task using both the RCJ and mSVG interfaces. Task B2 also recorded how accurately the navigation task was completed using each of the interfaces, after which the NASA TLX survey questionnaire was administered to the participants to relate their experiences.

Task B1 - RCJ skill level

For Task B1, Futaba Interlink joystick controller was plugged into a computer running the windows operating system, and the RealFlight Drone Simulator software program was loaded. The challenge levels were then queued, and the joystick controller handed to the study participants to begin the flight navigation task. The maximum number of trials allowed per challenge level for each participant was 7. The skill test was ended if the participant failed to progress to the next challenge level within the allowed number of trials, or if the participants completes all challenges through level 10. The scores generated by the RFDS simulator was a function of time, indicating how quickly a participant completed the task regardless of how inaccurately or roughly the task manoeuvre were executed.

Task B2 - RCJ vs mSVG performance comparison

In order to perform the RCJ and mSVG performance comparison, a test navigation path was designed and developed in Gazebo, which was then imported into the basic world environment of the RotorS ROS Gazebo simulator (Furrer et al., 2016), with an instance of the of the Ascending Technologies (2013) AscTec Firefly hexacopter UAV running. Figure 4.23a shows the designed navigation test path. It was designed to emulate a random real world flight navigation experience with 90° right and left turns, obtuse angle turn, climbing and dropping, hovering, and going through narrow openings. Figure 4.23b shows the developed navigation test path in Gazebo. In developing the the Gazebo navigation test path, the dimensions of the Rotor_s gazebo firefly uav was considered. It was found to be 0.665 m wide, which fitted within the square wall windows which were dimensioned 1 m horizontally and 1.5 m vertically. The full dimension specification details of the Astec firefly UAV model was available from Ascending Technologies (2013), and was given 605 × 665 × 165 mm. The height of the first window obstacle from table floor was 1 m, and the second window obstacle was 0.5 m. The table floor was 2.5 m from ground reference (z-axis upward). The entire challenge course structure is positioned $x = 8.0$ m, $y = 0.5$ m, and $z = 0$ m.



(a) Designed navigation flight test Path_v02.

(b) Gazebo developed test flight Path_v02.

Figure 4.23: Designed and developed nCA Tier 1-III Path_v02 flight navigation test path.

For each study participant, the navigation test was first completed using the joystick controller interface, after which it was re-completed using the mSVG interface. Figure 4.24 shows the developed mSVG navigation command syntax which the participants were required to familiarise themselves with, before performing the mSVG segment of the experiments.

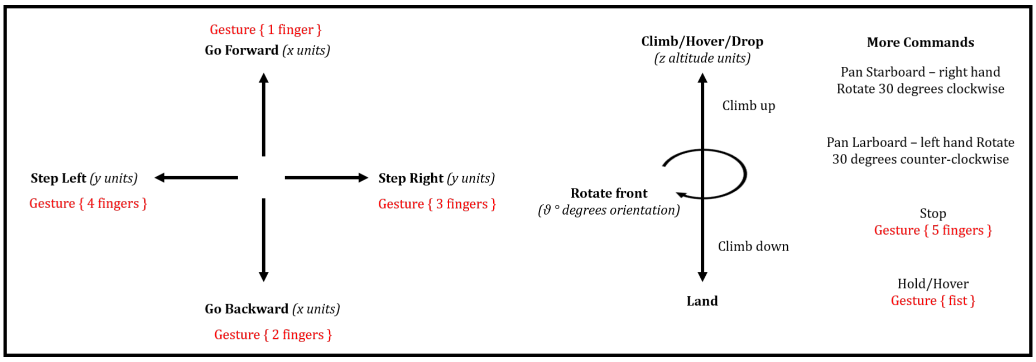


Figure 4.24: mSVG navigation control syntax used in flight Path_v02 navigation.

After conducting some preliminary experiments, the performance margin between the RCJ and mSVG interface was significantly large, the reason for which was due to the fact that the mSVG was endowed with certain features such as altitude, attitude, and position assist. Therefore, the Keyboard (KBD) interface was introduce to simulate the RCJ interface in altitude, atittude, and position (AAP) assist mode, in order to even the advantages the mSVG had over the RCJ, for a more like-to-like comparison. Therefore, three interfaces (RCJ, KBD, and mSVG) were eventually compared in the study, with the KBD interface representing the AAP-assisted RCJ interface, and the conclusions drawn for the KBD interface were applied to the AAP-assisted RCJ interface. In order words, a comparison was made between the mSVG interface and both the standard and AAP-modified RCJ interfaces.

4.5.3 Experiment procedure

This section enumerates the experiment study procedure. For a more detailed step-by-step procedure, refer to the 10-page participant logbook in Appendix D.5. As part of preliminary preparation for each experiments, each participants were provided with a printed copy of the participant information sheet, which was previously emailed to them. If the participant was satisfied with the study conditions and wishes to continue with the tests, then they were required to sign a written consent before the study could begin. In addition to this, health and safety briefings were given. Samples of this documents are available in Appendix D.3 and Appendix D.4. The experiment study procedure was as follows:

1. The participant workspace was setup with the participant comfortably seated in front of the computer workstation. Tested that the speech and gesture system works for participants – calibrating where necessary. Ensured that the participant could interact with the UAV simulator via speech, gestures, and a combination of both under optimum conditions (quiet environment and good lighting) while sitting in front of the UAV simulator. The preliminary measurements of the participants distance to microphone, microphone to speaker, participant to camera, ambient noise level, participant voice level, and ambient lighting level, was then performed and recorded in the participant's logbook. (20 minutes)

2. The first part of study A experiment was then conducted. The Bose link mini speaker noise source was turned on and a recording of a multirotor propulsion noise (included as supplementary multimedia data with this thesis) was played through the speaker. The noise level was being varied from 55 dB in steps of 5 dB until 85 dB. For each of 55 dB, 60 dB, 65 dB, 70 dB, 75 dB, 80 db, and 85 dB, the 12 word commands listed in Section 4.5.1 were repeated, and the result was recorded. (20 minutes)

3. The second part of study A experiment was then conducted. The LED lighting system was setup, the default laboratory lighting was turned off. Using the steps described in Section 4.5.1, the gesture capture quality for each lighting stage was recorded. (20 minutes)

4. The first part of study B (Task B1) was then conducted. The RFDS challenge levels were queued, and the participants were asked to attempt the RFDS challenges starting from

Level 1 through Level 10, as described in Section 4.5.2. The score attained, the completion time, and the number of trials for each level were recorded in the participants logbook. (25 minutes)

5. The second part of study B (Task B2) was then conducted, as described in Section 4.5.2. The ROS Gazebo was then setup for the RCJ navigation test experiment. The participant completion time was recorded in the participant logbook. A screen recording of the simulator navigation was saved for later analysis of the navigation accuracy. The participant was then administered the NASA TLX survey questionnaire to record their experience with the RCJ interface. These were repeated for the keyboard (AAP-assisted RCJ) and the mSVG navigation tests. For the mSVG test segment, an additional step of recording in the participant logbook when speech or gesture was used during the navigation test was performed. (30 minute)

The experiment study duration was 2.5 hours per participant – i.e. 20 minutes participant preliminary + 60 minutes study A + 15 minutes break + 55 minutes study B. Figure 4.25, shows the research experiment conductor preparing the experiment participant.

Figure 4.25: Preparing the participant – conducting the experiment.

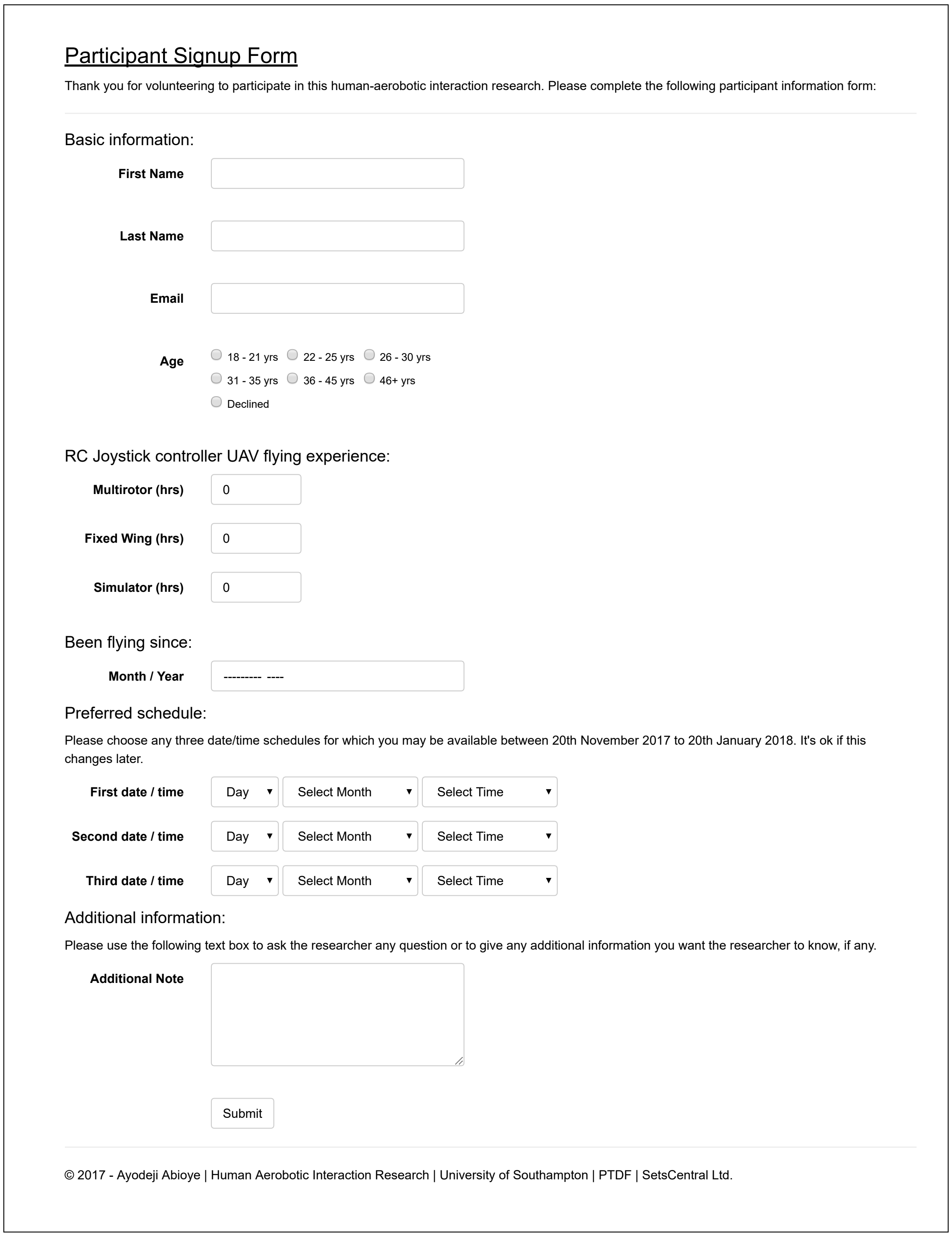
Some of the experiment precautions observed included carefully taking readings to ensure that the collected data was accurate and reliable. Experiment assumptions, constraints, and limitations were clearly stated were applicable, to ensure their repeatability. Standard factory calibrated measuring instruments were used. The Gain Express SLM-25 professional sound level meter with backlight display and a measurement range of 30 dB - 130 dB (with data recording function) was used for measuring sound levels. The YH-THINKING portable handheld professional digital Lux meter with USB and a measurement range of 0 Lux to 200,000 Lux was used for measuring the lighting level. Digital fluctuations were allowed to settle before taking readings.

A risk assessment was completed for this study and the study was approved by the University of Southampton Ethics and Research Governance Community with the ethics approval code: 30377. A copy of the completed risk assessment and the approved university ethics application form is given in Appendix D.1 and Appendix D.2.

4.5.4 Participants recruitment and demographic

For the research study, 37 participants were recruited. The study participants were volunteers and randomly recruited. Most of the participants were university students in their second or third year of studies. Each participant received a £20 Amazon voucher to compensate for time and travel. The selection criteria was good finger dexterity, normal speech articulation ability, English language proficiency, and aged between 18 and 69 years old. A website was designed, developed, and hosted on-line for recruiting and signing up the experiment participants – `https://www.hai-research.com/participate`. The website was developed using ASP.net C# with the aid of Microsoft Visual Studio 2017 IDE. It was hosted on Microsoft Azure UK South servers, on the domain *hai-research.com*. An SSL certificate was used to secure the website by establishing an encrypted link between the web server and user's web browsers. A secure database was setup to securely store participants signup information. The Website was also optimised for mobile usage to ease participant's signup process. Figure 4.26 shows the participants signup page printed from the Google Chrome web browser.

The demographic distribution of the 37 recruited participants is shown in Figure 4.27. Figure 4.27a shows each participants accumulated UAV flying hours using the RC joystick controller. The average flying hours of the participants was 10.9595 hours as indicated in Figure 4.27a. 14 of the 37 participants ($\sim$ 38%) had no previous flying experience. 12 of the 37 participants ($\sim$ 32%) had at least 10 hours of UAV RCJ flying experience. And the remaining 11 of the 37 ($\sim$ 30%) participants had some flying experience under 10 hours. Figure 4.27b shows how long the participants have been flying for. The average number of months the participants have been flying for was 24.1622 months. 11 of the 37 participants ($\sim$ 30%) had been flying for at least 24 months. The pie chart in Figure 4.27c shows the male to female gender distribution of the study participants. Of the 37 participants, there were 14 female and 23 male volunteers. The pie chart in Figure 4.27d shows the age distribution of the participants. 25 of the 37 participants ($\sim$ 68%) were aged between 18 - 21 years of age. The remaining 11 of the 37 participants age were distributed as shown in Figure 4.27d. Figure 4.27e shows the ethnic distribution of the participants. 24 of the 37 participants ($\sim$ 65%) were White-British and the others distributed



Participant Signup Form

Thank you for volunteering to participate in this human-aerobotic interaction research. Please complete the following participant information form:

Basic information:

First Name	
Last Name	
Email	
Age	○ 18 - 21 yrs ○ 22 - 25 yrs ○ 26 - 30 yrs ○ 31 - 35 yrs ○ 36 - 45 yrs ○ 46+ yrs ○ Declined

RC Joystick controller UAV flying experience:

Multirotor (hrs)	0
Fixed Wing (hrs)	0
Simulator (hrs)	0

Been flying since:

Month / Year	--------- ----

Preferred schedule:

Please choose any three date/time schedules for which you may be available between 20th November 2017 to 20th January 2018. It's ok if this changes later.

First date / time	Day	Select Month	Select Time
Second date / time	Day	Select Month	Select Time
Third date / time	Day	Select Month	Select Time

Additional information:

Please use the following text box to ask the researcher any question or to give any additional information you want the researcher to know, if any.

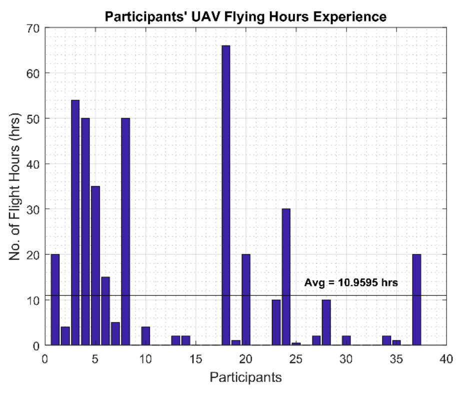
Additional Note

Submit

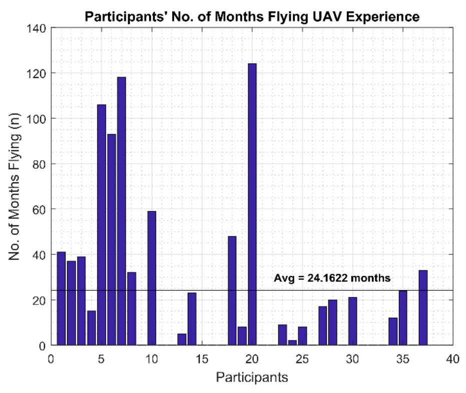
© 2017 - Ayodeji Abioye | Human Aerobotic Interaction Research | University of Southampton | PTDF | SetsCentral Ltd.

Figure 4.26: Participant signup page – https://www.hai-research.com/participate.

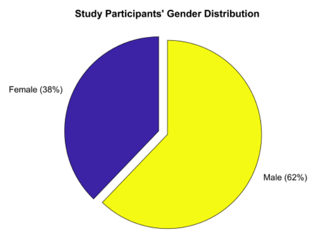
as shown in Figure 4.27e. The demographic data is also presented in the Table in Appendix B.1, for the purpose of specific detail referencing.



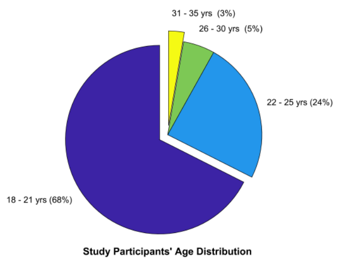
(a) UAV flying hours.



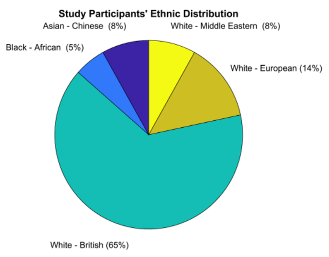
(b) Months flying UAV.



(c) Gender distribution.



(d) Age distribution.



(e) Ethnic distribution.

Figure 4.27: Participants demographic distribution - previous flying experience and unique characteristics.

4.5.5 Experiment Result Analysis

The experiment results were analysed with the aid of regression plots and analysis, the Analysis of Variance (ANOVA), and the two-sample t-tests. Statistical significance testing was perform on the results. Both null and alternative hypothesis were defined to proof or disproof the thesis hypothesis. The null hypothesis followed the pattern of stating that there was no statistically significant difference between the mean performances of the RCJ, KBD, and mSVG interfaces i.e. $\mu_{rcj} = \mu_{kbd} = \mu_{msvg}$, or $MS_{Between}$ roughly equals MS_{Within}. While the alternative hypothesis often argued that there was a significant difference between the mean performances of the interfaces i.e. $\mu_{rcj} \neq \mu_{kbd} \neq \mu_{msvg}$, or $MS_{Between} > MS_{Within}$. Where the null hypothesis test fails to be disproved or rejected, the null hypothesis was accepted, which usually indicates that the sample means were clustered within the 95% confidence interval of the combined sample distribution, for a given $\alpha = 0.05$. But were the null hypothesis was disproved, the null hypothesis was rejected and the alternative hypothesis was accepted. The t-test was used for pairwise comparison where two of the three sample means were considered similar, yet the ANOVA analysis suggests a significant difference because of the simultaneous triple comparison. In most cases, the ANOVA analysis was sufficient and preferred, in order to avoid compounding type I errors with multiple t-tests since more than two sample means were involved. Also specifying low α values helped reduced type I errors (false positive) – cases where the null hypothesis was rejected even though the null hypothesis was actually true.

4.6 Human Factors

According to Sanders and McCormick (1993), human factors "discovers and applies information about human behaviour, abilities, limitations, and other characteristics to the design of tools, machines, tasks, jobs, and environments for productive, safe, comfortable, and effective human use" (Stanton et al., 2005). Human factors is an important consideration in the design and maintenance of unmanned aircraft (Hobbs and Herwitz, 2005). For example, Johnson and Shea (2008) showed that the 2006 Predator B UAV accident, in Arizona, was caused by multiple human factor failures that could have been potentially prevented. In recent years, with the massive proliferation of remotely piloted unmanned aerial systems in many civilian applications, certain concerns are brewing up from several quarters - government, civil aviation agencies, security and intelligent agencies, the military, rights groups, and the public, who have argued that the operator's lack of shared fate with the aircraft could breed complacency. In addition, the operators of these systems could get bored or easily distracted. This could put the safe operation of such aerial vehicles into questions as such momentary lapse in concentration could result in an aerial accident, flying into restricted areas, invasion of privacy, amongst several others.

Therefore, beyond investigating and developing novel HCI interfaces for controlling small UAVs, there is a need to determine the effectiveness, efficiency, intuitiveness, reliability, safety, and suitability of these interfaces. The effectiveness and efficiency of the proposed mSVG interface, was investigated through the performance and cognitive workload analysis of the results from the experiment study.

4.6.1 Performance Measurement

A major consideration in the development of any system is how to measure the system's performance. According to Lindquist (1985), the effort-to-learn and the effort-to-use contributes to the usability of human-computer interaction (HCI) interfaces. Generally, a good interface design improves the overall user experience, makes interaction enjoyable, intuitive, easy to learn, and is functionally effective with an accepted level of precision and accuracy. Chao (2009) also suggested that a good human computer interaction is not about the human adapting to the limitation of the computer (as used to be the case), but about the computer adapting to the natural human tendencies or expectations. According to Shneiderman (1998), the usability of an interface can be measured by the time to learn to use the interface, the interface's performance speed and rate of errors, the user's retention of how to use the interface over time, and the user's subjective satisfaction. The last point, user's satisfaction, is particularly important as it agrees with Coelho and Verbeek (2014) observation that despite the technical poor performance and inaccuracy of a gesture interface, the users still rated the system high because they enjoyed the experience. Considering this implication from an alternative perspective, implies that a very technically good and accurate interface may still offer a poor user experience. According to Oviatt et al. (2004), an interface performances can be measured by the human performance error and the systems response latency, which agrees with Shneiderman (1998) usability measures. The participants of this research study were also asked to rate their navigation task performance with each of the RCJ, KBD, and mSVG interfaces.

4.6.2 Cognitive Workload and Situation Awareness

Cavett et al. (2007) investigated the performance of various human-computer interface (HCI) configurations for the control of unmanned aerial vehicles. The researchers designed a few combinations of graphical interfaces using MATLAB, to test the variation in the human cognitive workload on different interfaces. They tried to identify interfaces that reduces the human operator cognitive workload and enhances the operator's situation awareness. These researchers suggested the need to redesign current HCI interaction and communication links between UAV and operators because of the limitation in human cognition, judgement, decision-making, and tactical understanding. Although in Cavett et al. (2007), the researchers were interested in a redesign of how information is presented from the UAV to the Operator, this research investigates the complementary case of how the information from the operator is communicated to the UAV. In both cases, the idea of reducing the operator workload and improving situation awareness is equally important. Both direction of communication combine to form a close-loop control system, with feedback. Similarly, Lim et al. (2019) investigated the management of the human operator's workload and situational awareness, within the context of multi-UAV surveillance applications, using their developed Cognitive Human-Machine Interfaces and Interactions ($CHMI^2$) framework. They proposed an adaptable and reconfigurable ground control station that responds to the real-time fatique level and situation awareness of the operator in order to improve their decision making capability.

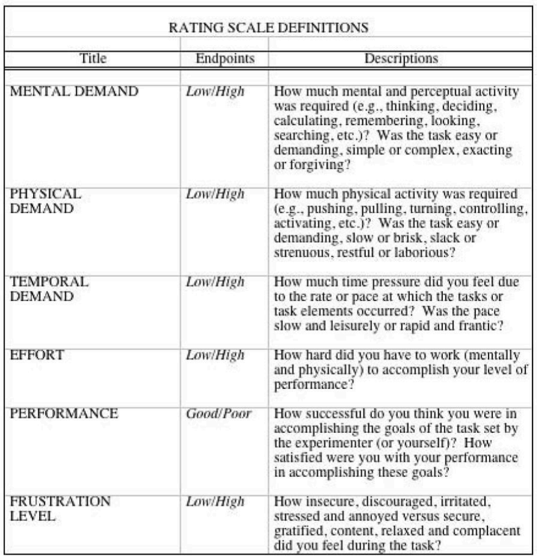
So, what is meant by the operator's cognitive workload and situation awareness? Cognitive workload refers to the amount of concentration required by an operator in performing a task. It can be considered as the amount of 'brain power' dedicated to the performance of a particular activity. Situation awareness describes an awareness of the environment - elements and events

- during task performance. It is the ability to effectively observe and react to other environmental disturbances, while performing the primary task. For example, consider the case of a UAV operator using a joystick controller, to control a UAV without stabilization assist (nCA tier 1-I). The operator's concentration requirement is highly demanded to keep the UAV flying within a safe distance from all obstacles, as well as protect itself from any dangerous manoeuvre due to an application of too much control (thrust, rudder, aileron, or yaw). Because of this, the operators interaction with the environment is bare minimal. A narrow field of visual focus could result in a fast moving flying object approaching from a distance, appearing suddenly, causing a potentially dangerous impulse response from the UAV operator. The mSVG technique proposed and developed in this research was considered to offer a better balance between cognitive workload and situation awareness for the small multirotor UAV operator. This was investigated through the use of the NASA Task Load Index (TLX) components.

4.6.3 Cognitive Workload - NASA Task Load Index

The NASA TLX is a cognitive workload measurement tool developed by Hart and Staveland (1988) for NASA for performing comparative measurements of the cognitive demand designed equipments place on their aviation crew. This tool has since found application in many other fields beyond aviation and aeronautics, most notably robotics and user interface research, such as Fels et al. (2015) predicting cognitive workload for brain-robot interfaces, and Haber and Chung (2016) assessment of UAV operator workload for a ground control workstation design, among several others. In Hart (2006) survey of 550 studies in which the NASA-TLX questionnaire was used, the authors found this tool to have spread far beyond its original application (aviation), focus (crew complement), and language (English).

Figure 4.28 shows the NASA Task Load Index (TLX), a multidimensional scale subjective workload rating method designed to obtain workload estimates from one or more operators while they are performing a task or immediately afterwards (Hart, 2006). In NASA TLX, workload is defined as the "cost incurred by human operators to achieve a specific level of performance." The subjective experience of workload is defined as an integration of weighted subjective responses (emotional, cognitive, and physical) and weighted evaluation of behaviours. The behaviours and subjective responses, in turn, are driven by perceptions of task demand. Task demands can be objectively quantified in terms of magnitude and importance. An experimentally based process of elimination led to the identification of six dimensions for the subjective experience of workload: mental demand, physical demand, temporal demand, perceived performance, effort, and frustration level. Figure 4.29 shows a graphical representation of the sub-scales and overall weight ratings. The overall workload rating was obtained by performing a weighted sum of the



RATING SCALE DEFINITIONS

Title	Endpoints	Descriptions
MENTAL DEMAND	*Low/High*	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
PHYSICAL DEMAND	*Low/High*	How much physical activity was required (e.g., pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
TEMPORAL DEMAND	*Low/High*	How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
EFFORT	*Low/High*	How hard did you have to work (mentally and physically) to accomplish your level of performance?
PERFORMANCE	*Good/Poor*	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
FRUSTRATION LEVEL	*Low/High*	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

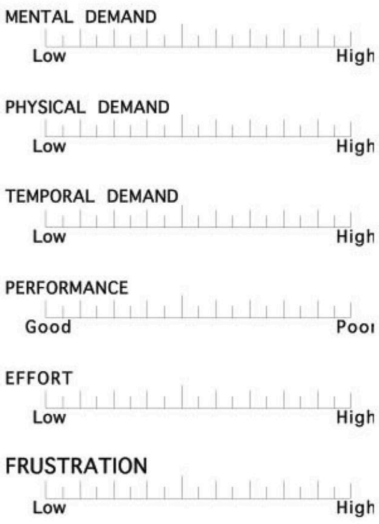


Figure 4.28: Hart (2006) NASA TLX rating scale definition.

six index components.

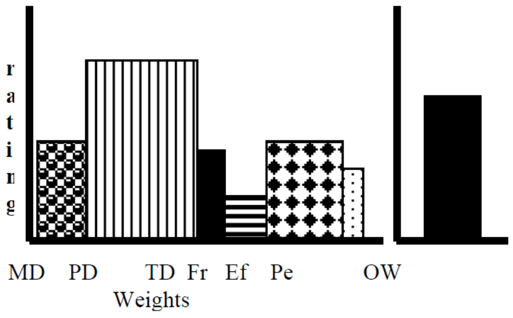


Figure 4.29: Hart (2006) graphical representation of weighted sub-scale ratings and an overall workload value.

In this research study, the administered NASA TLX survey was analysed by comparing the non-weighted bar chart contour of the interfaces with each other. The interface with the lower contour was considered the better interface. In addition to this, element by element bar plots were compared too. And finally, an overall non-weighted mean-sum was used to determine a scaler factor to represent this research study, for the purpose of future numerical/quantitative comparison with other similar research works. A sample of the NASA TLX questionnaire is given in Appendix D.6. Table 4.7 highlights the key question posed by each component of the NASA TLX survey questionnaire, and also shows the symbols used to represent the TLX

Table 4.7: NASA TLX Symbol and Meanings

S/N	**Symbol**	**TLX Index**	**Key Question**
1	md	mental demand	How mentally demanding was the task?
2	pd	physical demand	How physically demanding was the task?
3	td	temporal demand	How hurried or rushed was the pace of the task?
4	p	performance	How successful were you in accomplishing what you were asked to do?
5	e	effort	How hard did you have to work to accomplish your level of performance?
6	f	frustration	How insecure, discouraged, irritated, stressed, and annoyed were you?

component in the analysis of the experiment result.

4.7 Chapter Conclusion

In this chapter, the research method used and the rational for selecting the method and components were given. The interface development section discussed the keyboard interface as a symbol generator, the speech capture and processing, visual gesture recognition, and RC joystick selection. Three hardware platforms were discussed, the ImmersionRC XuGong, iQuad Jade, and Unicorn multirotor platforms. Two hardware in the loop simulation were also discussed, the RealFlight Drone Simulator and the RotorS ROS Gazebo simulator. The method used in this research work was to conduct an experiment study. The details of the experiment design was presented and discussed. The experiment procedure was enumerated. The participant recruitment method and the demography of the recruited participants was discussed. The method considered for the analysis of the experiment result was also presented. The discussions in this chapter was capped with the presentation of the cognitive workload human factor analysis consideration, with a focus on the NASA TLX survey tool.

By comparing the methodology used in this research with other similar research, it was found that Gubcsi and Zsedrovits (2018) 'Ergonomic Quadcopter Control Using The Leap Motion Controller', also used the Odroid XU4 hardware but only recruited 4 experiment participants, whereas this study recruited 37 participants i.e. 33 more participants. However, unlike the hardware-in-the-loop laboratory simulation tests performed in this study, Gubcsi and Zsedrovits (2018) conducted outdoor tests on a real IRIS+ drone. Further, the authors chose to investigate the Leap Motion device for UAV control while in this research, a multimodal combination of speech and gesture was selected for the same UAV control. Schelle and Stutz (2016) work was gesture only, i.e. no speech combination, although they considered using flashlights to augment their gesture interaction. They used whole body gestures, whereas this research used finger gestures only. Also, they did not perform any comparison with other UAV control interfaces, whereas this research compared the mSVG technique with the RCJ interface. They did not perform any experiment involving multiple human operators as was performed in this research, where 37 participant participated in the experiment study. However, their implementation was more practical, unlike this research which was conducted under controlled laboratory conditions, their tests was conducted outdoors (Schelle and Stütz, 2018).

Chapter 5

The Navigational Control Autonomy (nCA) Model

The development of the navigational control autonomy (nCA) model was motivated by the need to compare multiple control interfaces. It was soon observed that the autonomy of the platform affects the choice of human-computer interaction (HCI) control interface used. The nCA model therefore provides a method for mapping HCI control interfaces to an appropriate platform autonomy level.

5.1 Control Autonomy - Autonomy Control Levels

"By Autonomous, it is meant the ability to be self-governing/operating within given constraints or rule set" (Payne, 2007). The meaning of autonomy in robotics slightly differs from the political, philosophical, or biological meanings. This difference has been responsible for several contentions between experts on how this term is used. The argument as to whether robotic systems performing a task without external control commands are really doing so solely through their own ability to make decisions or through a decision-making method that has been pre-programmed into them. Especially when such systems may be considered to have been designed and developed to perform such specific functions and not a random system existing in nature. The designed artificial autonomous system process could be broken down into bits, understood, and hence predictable by the designer. Therefore, there seems to be a thin layer requiring further clarification as to whether:

- Proponents: a robotic system, performing a particular task without being told how to do such task, is simply autonomous or
- Opponents: just executing a pre-defined program on how to make the right decisions that

complete the task.

In other words, a clarification needs to be made as to whether executing a program, not of the task, but of how to decide on key elements that affects accomplishing the task, is autonomous or automatic. Clearly, executing a program of the task would be automatic as described by Clough (2002). In order to avoid being drawn into this debate, what is meant by autonomy in this chapter is defined in simple, clear, and unambiguous terms. Clough (2002) made a distinction between automatic and autonomous systems, "Automatic means that a system will do exactly as programmed, it has no choice. Autonomous means that a system has a choice to make free of outside influence..." According to Pfeifer and Scheier (1999), "autonomy means independence of control. This characterization implies that autonomy is a property of the relation between two agents, in the case of robotics, of the relations between the designer and the autonomous robot. Self-sufficiency, situated-ness, learning or development, and evolution increase an agent's degree of autonomy."

In this chapter, autonomy is conceptualised as being self-operating in a dynamic environment, within a given sets of rules or constraints, without relying on external control. This could be with the aid of a computer program working with sensors to guide decision-making, or some complicated artificial intelligence real-time learning and adapting algorithm. This could be a decision method that the operator may or may not understand, may or may not be able to predict, but is optimal enough to safely complete the assigned task to an acceptable standard. Autonomy, in this case, needs no continuous real-time control interaction during task execution but rather sparse real-time command, or just the initial command instruction may be given, in some cases.

Autonomy is an abstract concept difficult to measure empirically; hence, it is rather described qualitatively by analogically developed subjective hierarchical levels. The autonomy of an aerial robot can be considered in two contexts - navigation and payload. Usually, the payload autonomy level is expected to match the navigation autonomy as observed in Clough (2002) classification, which was adopted by the US Air Force Research Lab (AFRL) as the AFRL's ACL (Autonomous Control Levels). It was also used as a standard classification in other US military and government agencies, and by their contractors (Sholes, 2007). Clough (2002) identified eleven autonomy control levels, listed in order of increasing autonomy:

1. Remotely Piloted Vehicle

2. Execute Pre-planned Mission

3. Changeable Mission

4. Robust Response to Real Time Faults/Events
5. Fault/Event Adaptive Vehicle
6. Real Time Multi-Vehicle Coordination
7. Real Time Multi-Vehicle Cooperation
8. Battlespace Knowledge
9. Battlespace Cognizance
10. Battlespace Swarm Cognizance
11. Fully Autonomous

The ACL incorporated a model of the human OODA (Observe, Orient, Decide, and Act) loop, a loop that seemed popular with the military (Clough, 2002). Sholes (2007) developed a simulation technique for evaluating multiple autonomy algorithms based on Clough (2002) ACL model. In a separate study by the Office of Naval Research Horrigan and ONR's Committee (2000) classified autonomy into the following six levels enumerated below:

1. Human Operated
2. Human Assisted
3. Human Delegated
4. Human Supervised
5. Mixed Initiatives
6. Fully Autonomous

Hill et al. (2007) presented the modified PACT (Pilot Authority and Control of Task) autonomy level system, originally developed by the UK Defence Evaluation Research Agency (DERA), as a better classification of autonomy compared to the Clough (2002) ACL. Firstly, they considered Clough (2002) descriptive autonomy useful for vehicle level classification but inadequate for classification at functional levels. Secondly, Clough (2002) ACL imposed an artificial ceiling on the autonomy level that can be achieved by an individual vehicle by introducing swarm behavioural requirements within it. The PACT classification is as follows:

1. Level 0: Full pilot with no computer autonomy

2. Level 1: Pilot assisted by computer only when requested
3. Level 2: Pilot assisted by computer without pilot necessarily requesting it (computer advice continuously)
4. Level 3: Pilot backed up by computer
5. Level 4a: Computer backed up by pilot - computer acts when authorised
6. Level 4b: Computer backed up by pilot - computer acts unless pilot disapproves
7. Level 5a: Computer monitored by pilot - computer informs human of actions
8. Level 5b: Computer monitored by pilot - computer acts completely autonomously and need not inform the human of its decisions

As may be observed, the PACT levels seemed similar to Horrigan and ONR's Committee (2000) classification, which also closely aligns with the civilian aerial robot navigational control autonomy model presented in this chapter. In addition to the limitations already offered by Hill et al. (2007), Clough (2002) ACL was structured after military missions, therefore, its military-specifics context makes it difficult to adopt it in many emerging civilian applications. In addition, Clough (2002) ACL was modelled as a generalised classification method for all types of military robotic systems - aerial, ground, and underwater.

Since autonomy, as a concept in robotics is a subjective non-quantitative (not measurable) analogical (but comparable) descriptor, perhaps its layers could be considered in fuzzy terms as overlapping strata expressed as a range of percentile rather than as discrete levels. Therefore, the developed autonomy levels presented in Figure 5.1, which is used as the basis of discussing the human aerobotic interfaces, is considered as a continuous, overlapping layer rather than quantized or discrete layers. For this reason, the presentation of the autonomy levels in Figure 5.1 is not presented in levels or vertical columns suggesting equal linear step size increments in autonomy. Sholes (2007) simulation plot and prediction of the growth of autonomy seems to supports this hypothesis of a non-linear step-size increase or change from one successive layer to another. Moreover, the concept of autonomy is a borrowed property from living organisms existing in nature and the ecological system, which is rarely linear.

Figure 5.1 presents a two-dimensional, expandable pyramidal model of Navigational Control Autonomy (nCA) levels for multirotor aerial robotic systems. In this model, autonomy is divided into three tiers: lower, intermediate, and upper tiers. Within each tier, autonomy level increases from left to right, and then across tiers from lower to intermediate to the uppermost tier. For

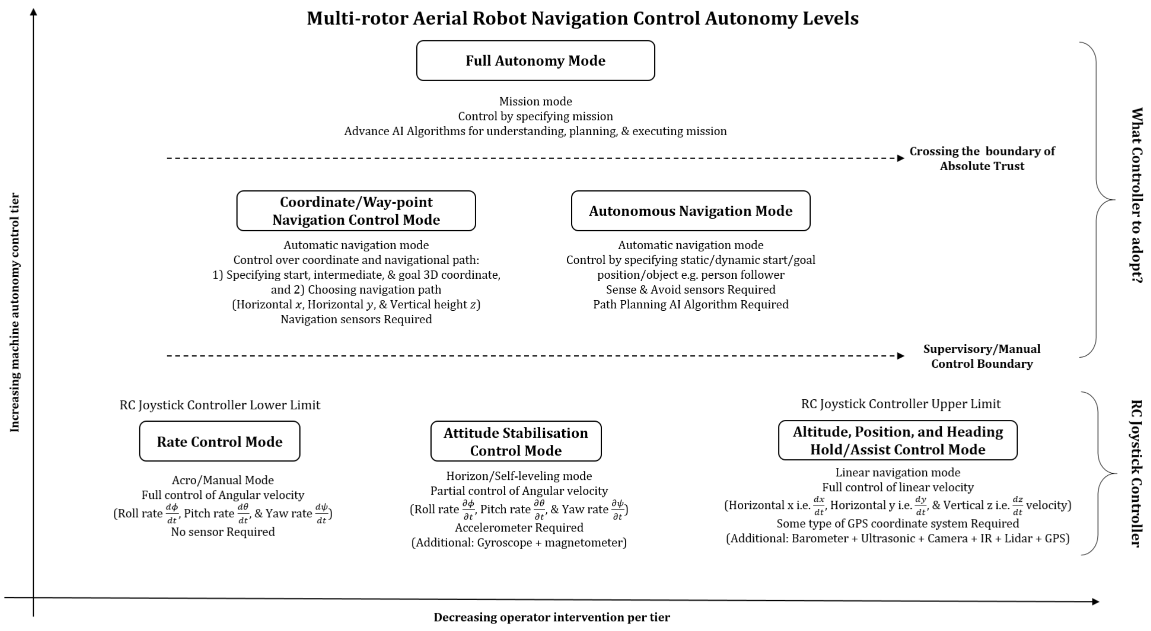

Multi-rotor Aerial Robot Navigation Control Autonomy Levels
Full Autonomy Mode
Mission mode
Control by specifying mission
Advance AI Algorithms for understanding, planning, & executing mission
Crossing the boundary of Absolute Trust
Coordinate/Way-point Navigation Control Mode
Automatic navigation mode
Control over coordinate and navigational path:
1) Specifying start, intermediate, & goal 3D coordinate,
and 2) Choosing navigation path
(Horizontal x, Horizontal y, & Vertical height z)
Navigation sensors Required
Autonomous Navigation Mode
Automatic navigation mode
Control by specifying static/dynamic start/goal
position/object e.g. person follower
Sense & Avoid sensors Required
Path Planning AI Algorithm Required
Supervisory/Manual Control Boundary
What Controller to adopt?
RC Joystick Controller Lower Limit
Rate Control Mode
Acro/Manual Mode
Full control of Angular velocity
(Roll rate $\frac{d\phi}{dt}$, Pitch rate $\frac{d\theta}{dt}$, & Yaw rate $\frac{d\psi}{dt}$)
No sensor Required
Attitude Stabilisation Control Mode
Horizon/Self-leveling mode
Partial control of Angular velocity
(Roll rate $\frac{\partial\phi}{\partial t}$, Pitch rate $\frac{\partial\theta}{\partial t}$, & Yaw rate $\frac{\partial\psi}{\partial t}$)
Accelerometer Required
(Additional: Gyroscope + magnetometer)
RC Joystick Controller Upper Limit
Altitude, Position, and Heading Hold/Assist Control Mode
Linear navigation mode
Full control of linear velocity
(Horizontal x i.e. $\frac{dx}{dt}$, Horizontal y i.e. $\frac{dy}{dt}$, & Vertical z i.e. $\frac{dz}{dt}$ velocity)
Some type of GPS coordinate system Required
(Additional: Barometer + Ultrasonic + Camera + IR + Lidar + GPS)
RC Joystick Controller
Increasing machine autonomy control tier
Decreasing operator intervention per tier

Figure 5.1: Multi-rotor aerial robot navigation control autonomy (nCA) levels.

this reason, it is termed the 'inverted lightning model of autonomy' as shown in Figure 5.2. This two-dimensional pyramidal model of autonomy is expandable rightwards and upwards to accommodate for new or missing autonomy level classes.

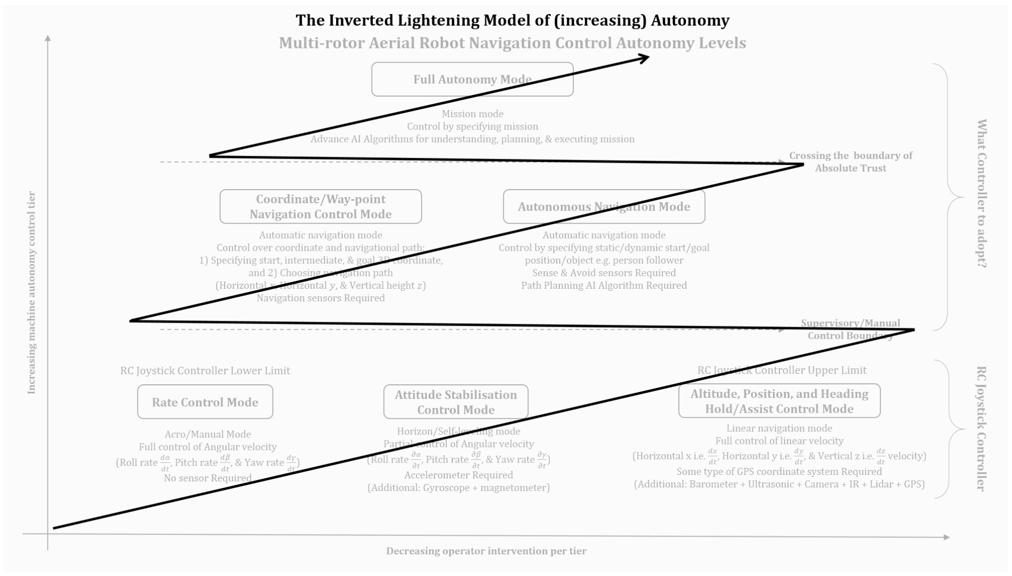

The Inverted Lightening Model of (increasing) Autonomy
Multi-rotor Aerial Robot Navigation Control Autonomy Levels
Full Autonomy Mode
Crossing the boundary of Absolute Trust
Coordinate/Way-point Navigation Control Mode
Autonomous Navigation Mode
Supervisory/Manual Control Boundary
What Controller to adopt?
RC Joystick Controller Lower Limit
Rate Control Mode
Attitude Stabilisation Control Mode
RC Joystick Controller Upper Limit
Altitude, Position, and Heading Hold/Assist Control Mode
RC Joystick Controller
Increasing machine autonomy control tier
Decreasing operator intervention per tier

Figure 5.2: The inverted lightning model of increasing autonomy.

Unlike the Office of Naval Research (ONR) and Clough (2002) levels of autonomy, the nCA model does not take into account the payload autonomy. However, payload autonomy is expected to match the aerial robot's navigation control level. The separation of navigational autonomy from payload autonomy simplifies the interaction relationship between an operator and an aerial robot (as considered in this chapter). Additionally, because autonomy classification may vary slightly across specific domains, the classification method used in this chapter has been developed specifically for multirotor aerial robotic systems' application in the civilian domain; as the AFRL's and ONR's classification were patterned towards military specific applications. Perhaps, with the rapidly evolving civilian applications of the aerobot, a less military objective-based classification may be worth considering.

5.2 Tier-one Control Levels

The nCA model described in Figure 5.1, has six components distributed in three tiers. The first tier consists of the following modes:

1. Rate control
2. Attitude stabilisation control
3. Altitude, position, and heading hold/assist Control

Rate control mode is the minimal level at which a human operator can control a small multirotor aircraft. In this mode, no sensor is needed. Full control of the three angular velocities (Roll rate $\frac{d\phi}{dt}$, Pitch rate $\frac{d\theta}{dt}$, & Yaw rate $\frac{d\psi}{dt}$) are given to the operator. This is usually performed with the aid of the RC joystick controller. A high level of skill is required to operate the multirotor aircraft at this level. This level is also called the Acro (short for Acrobatic) or Manual mode. RC hobbyist often prefer this mode when performing stunt manoeuvres for entertainment purposes. Flying at this level usually requires many training hours. It could take up to 600 hours of training to become an expert (from being a novice who has never flown before).

The Attitude stabilisation control mode is the next level after the rate control mode. In this mode, the operator is assisted with attitude stabilisation. The operator only has partial control over the three angular velocities (Roll rate $\frac{\partial\phi}{\partial t}$, Pitch rate $\frac{\partial\theta}{\partial t}$, & Yaw rate $\frac{\partial\psi}{\partial t}$). This is because the multirotor aircraft's flight controller, in this case, has a stabilisation loop that continuously balances the aerobot horizontally, especially when the operator stops feeding new rate control data. The stabilisation loop in this mode requires, at least, an accelerometer sensor. Additional or optional sensors may include the gyroscope and magnetometer. Data from a gyroscope could be collected and fused with the accelerometer data for a more accurate attitude stabilization control, because the accelerometer is prone to noise and the gyroscope to drift. A magnetometer could also be integrated to minimize yaw drift.

The next level after the attitude stabilisation is the altitude, position, and heading hold/assist control mode. In addition to the attitude stabilisation from the previous mode, the operator is also assisted with the altitude, position, and heading direction. In this mode, the operator no longer has control over three angular velocity parameters. Control is performed by manipulating three linear velocities (Horizontal x i.e. $\frac{dx}{dt}$, Horizontal y i.e. $\frac{dy}{dt}$, & Vertical z i.e. $\frac{dz}{dt}$ velocity). In this mode, more sensors are required, such as GPS, barometer, ultrasonic, camera, infrared, lidar, etc. for localisation within the system specified coordinate system.

5.3 Tier-two Control Levels

The second tier of the nCA model consists of the following two modes:

1. Coordinate/waypoint navigation control mode
2. Autonomous navigational mode

The lowest level of the second tier is the coordinate/waypoint navigation control mode. This may also be called the automatic navigation mode. In this mode, low-level controls are completely abstracted from the operator. Navigation is specified by the user before execution. The start point, intermediate points, and goal point are pre-fed into the aerobot. Once execution starts, the multirotor flight controller is responsible for all low-level control required to achieve the goal point from the start point to the endpoint via the route specified by the operator. Execution would proceed as plan except if the operator decides to abort the mission. In this mode, navigational sensors for location, continuously tracking flight route, are required. The system may or may not have a sense and avoid system and is, therefore, the responsibility of the operator to ensure that the specified route is safe enough for the aerobot to operate. The aerobot at this control level is expected to be able to perform some self-diagnostics check on itself, and relay issues requiring operator advice or intervention and may be robust enough to execute certain fundamental safety measures such as abort mission and return to launch.

The higher level of the second tier of the nCA model is the autonomous navigation mode. In this mode, the operator only needs to specify the goal position or object, which could be static or dynamic, as in the case of a person-following UAV. This mode requires: 1) sense and avoid (SAA) sensors and 2) a path planning AI algorithm.

5.3.1 nCA Tier 2-I Alps search and rescue scenario

In Section 1.4.1, the Alps search and rescue scenario was introduced. The operation described was a high level operation requiring navigation autonomy in the order of nCA Tier-two.

Table 5.1: Example Alps climber's UAV communication vocabulary set.

S/N	Possible Climber's Requests	Possible UAV Response	Possible Climber's Response
1	Speech "Alert!"	a)No Weather or Signposting alert! Is there anything else I can help with? b) Mild/Medium/High Weather/Storm/Precipitation heading this way! Is there anything else I can help with? c) Nearest Crevice is within 0.5 km, 1 km, 2 km, 3 km, 4 km, 5 km, or more than 5 km away. Is there anything else I can help with? d) Mild/Medium/High Weather/Storm/Precipitation heading this way, and the nearest Crevice is within 0.5 km, 1 km, 2 km, 3 km, 4 km, 5 km, or more than 5 km away. Is there anything else I can help with?	a) Speech "Yes" b) Speech "No" c) Gesture "Thumbs up" d) Next Request
2	Speech "Shelter!" or Gesture "time out"	a) No shelter nearby! b) Nearest Crevice is within 0.5 km, 1 km, 2 km, 3 km, 4 km, 5 km, or more than 5 km away. Is there anything else I can help with?	a) Speech "Yes" b) Speech "No" c) Gesture "Thumbs up" d) Next Request
3	Speech "Panoramic selfie!" or Gesture "Picture board"	a) Initializing panoramic event! Please confirm panoramic selfie? [Climber responds to continue] b) [After panoramic selfie event] Panoramic selfie captured! Is there anything else I can help with?	a) Speech "Yes" b) Speech "No" c) Gesture "Thumbs up" d) Next Request
4	Speech "Help!" or Gesture "Pray"	a) Contacting control room operator! Please confirm request? [Climber responds to continue] b) UAV acts as dial up telephone between mountaineer and rescue operator c) [After disconnect] Is there anything else I can help with?	a) Speech "Yes" b) Speech "No" c) Gesture "Thumbs up" d) Next Request
5	Speech "Go away!" or Gesture "Wave away" or Gesture "Goodbye wave" or "ignore UAV for 30 s"	a) UAV returning to patrol! Goodbye?	a) Speech "Yes" b) Speech "No" c) Gesture "Thumbs up" d) Next Request

Table 5.2: Example Alps search and rescue operator’s UAV communication vocabulary set.

S/N	Possible Operator’s Requests	Possible UAV Response	Possible Operator’s Response
1	a) Speech “Hold!” b) Speech “Hold Position!” c) Speech “Hold Altitude!” d) Gesture “Fist!” e) Speech “Release hold!”	a) Holding position and altitude! Please confirm operation? b) Holding position! Please confirm operation? c) Holding altitude! Please confirm operation? d) Holding position and altitude! Please confirm operation? e) Releasing position and altitude hold! Please confirm operation?	a) Speech “Yes” b) Speech “No” c) Gesture “Thumbs up” d) Next Request
2	a) Speech “Hover higher/lower!” b) Speech “Hover at 3 m, 4 m, 5 m, 6 m, or 7 m”	a) Hovering higher/lower! Please confirm operation? b) Hovering at 3m, 4 m, 5 m, 6 m, or 7 m! Please confirm operation?	a) Speech “Yes” b) Speech “No” c) Gesture “Thumbs up” d) Next Request
3	a) Speech “Video!” or b) Speech “Stop Video!”	a) Recording video! Please confirm operation? b) Stopping video recording! Please confirm operation?	a) Speech “Yes” b) Speech “No” c) Gesture “Thumbs up” d) Next Request
4	Speech “Panoramic view/video!”	a) Capturing panoramic video! Please confirm operation? [Operator responds to continue] b) [After capture] Initializing Bluetooth/Wi-Fi video transfer/upload! [Operator preps Rx device or operation times out in 30 sec] c) [After transfer] Panoramic view completed! Is there anything else I can help with?	a) Speech “Yes” b) Speech “No” c) Gesture “Thumbs up” d) Next Request

5	a) Speech “Bird’s eye view!” or b) Gesture “hand salute on forehead long range view”	a) Capturing bird’s eye view! Please confirm operation? [Operator responds to continue] b) [After capture] Initializing Bluetooth/Wi-Fi image transfer-/upload! [Operator preps receiving device or times out in 30 sec] c) [After transfer] Bird’s eye view completed! Is there anything else I can help with?	a) Speech “Yes” b) Speech “No” c) Gesture “Thumbs up” d) Next Request
6	a) Speech “Follow me!” or Gesture “Wave along” b) Speech “Stop following!”	a) Initializing follow mode! Please confirm operation? b) Stop following! Please confirm operation?	a) Speech “Yes” b) Speech “No” c) Gesture “Thumbs up” d) Next Request
7	a) Speech “Find ‘p’ object with ‘q’ colour in ‘r’ place!” b) Speech “Find climbers over there [Pointing Gesture]” c) Speech “Find missing persons over there [Pointing Gesture] using heat/thermal signatures” d) Speech “Track that [Pointing Gesture] brown [colour descriptor] vehicle [object descriptor]!”	a) Searching ‘p’ object with ‘q’ colour in ‘r’ place! Please confirm operation? [recall via controller or handheld computer] b) Searching for climbers North, South, East, or West of here. Please confirm operation? [recall/cancel via controller or handheld computer] c) Searching for missing persons North, South, East, or West of here, using visible and thermal imaging. Please confirm operation? [recall/cancel via controller or handheld computer] d) Tracking the brown [colour descriptor] vehicle [object descriptor] at θ deg [compass bearing]. Please confirm operation? [recall/cancel via controller or handheld computer]	a) Speech “Yes” b) Speech “No” c) Gesture “Thumbs up” d) Next Request
8	a)Speech “Go away/patrol!” b) Speech “Patrol” c) Gesture “Wave away”	a) UAV returning to patrol! Goodbye?	a) Speech “Yes” b) Speech “No” c) Gesture “Thumbs up” d) Next Request

Example Alp search and rescue operator’s UAV communication vocabulary set - END

This section expands on other possible applications within the Alp mountaineering environment. For example, any of these events could trigger the automatic approach of a patrol: a) signposting event to alert of dangerous crevice nearby or approaching stormy weather, b) scenic sight event to suggest a panoramic selfie, c) emergency event such as an unconscious climber to alert the central control for emergency response. However, a climber could trigger a UAV approach by simply waving it over. Examples of the climber control command and interaction possible are listed in Table 5.1. In addition to the standard climber control commands, search and rescue operators may have access to more advance control command operations such as those listed in Table 5.2.

5.3.2 Panoramic selfie

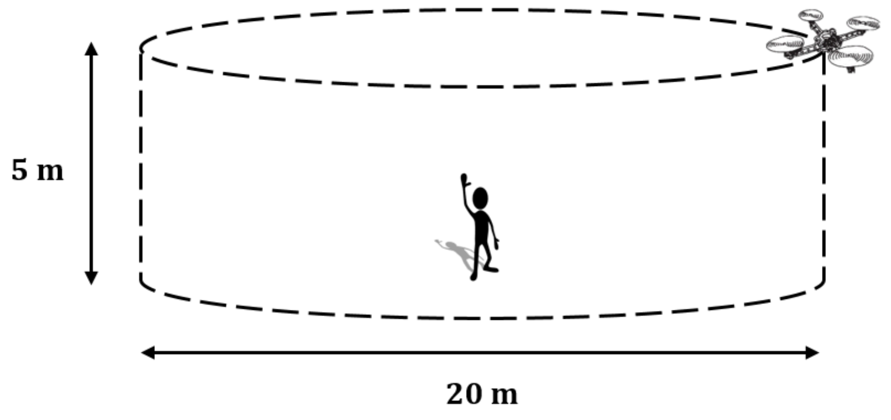


Figure 5.3: Panoramic selfie operation.

In panoramic selfie mode, described in Figure 5.3, the UAV pulls back from the operator to a horizontal distance 10 m away from the operator and climbs to a 5 m altitude. It then focuses the camera on the mountaineer and flies along the fixed geometric path, formed by the top edge of an imaginary cylinder as shown in Figure 5.3, for a complete revolution about the mountaineer, while panning the camera to remain focused on the climber, capturing a short video clip or image sequence. Figure 5.4 provides a real life size references for this application.

5.3.3 Bird's eye view

In bird's eye view mode, the UAV rises vertically to an altitude of 50 m, captures an image, the centre of which is GPS tagged, and then descends back to the user to transfer the image via

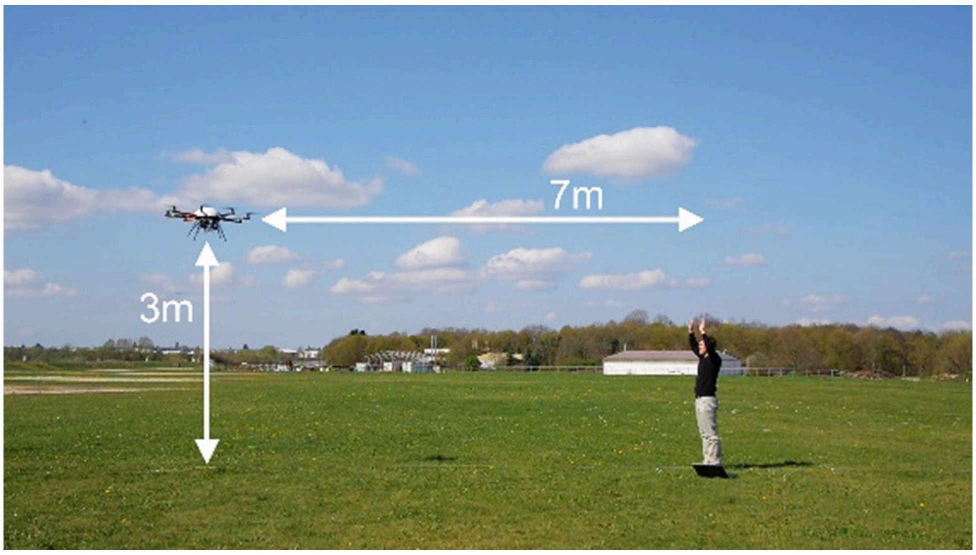


Figure 5.4: Panoramic selfie size reference [image source: Schelle and Stutz (2016)].

Wi-Fi or Bluetooth with the current climber location geo-tagged.

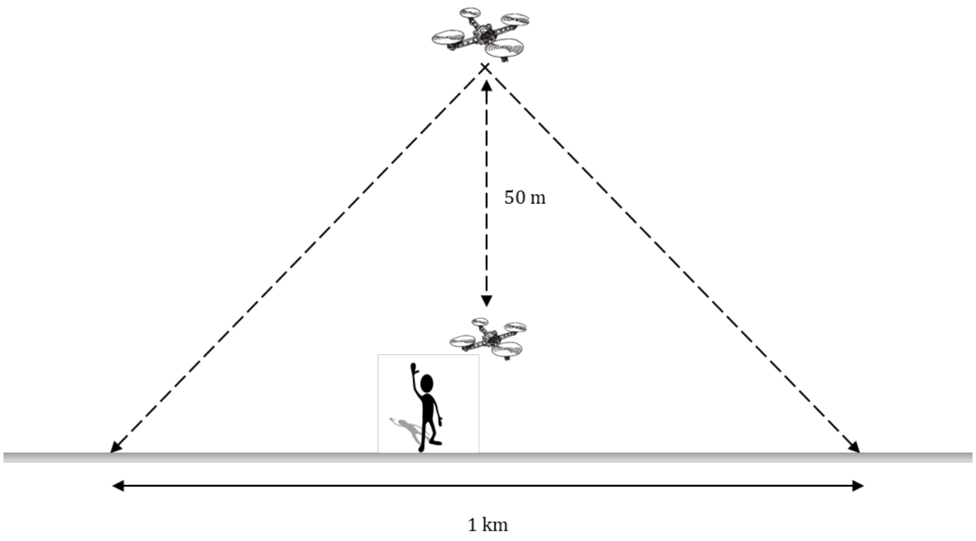


Figure 5.5: Bird's eye view operation.

When a search and rescue team operator requests a bird's eye view, the UAV climbs up 50 m (from its initial 3-5 m altitude), captures an image 1 km wide, and then descends back to the operator, as described in Figure 5.5, where the image is transferred to the operator's handheld computer, with the operator's location geo-tagged and other landmark or persons nearby indicated. Both visible and thermal images are made available to the operator. For the operator in Figure 5.5, the angle of view required for the camera on the UAV to produce the 1 km stretch image, assuming the UAV was initially hovering at 3 m from ground level, would be $\theta = 2 \arctan \frac{500 \text{ m}}{53 \text{ m}} = 168°$.

5.4 Tier-three Control Level

The third tier consists of only one control mode - the full autonomy mode. At this point, the operator interaction with the aerobot is very sparse. The operator only needs to specify a mission and the aerobot takes care of the rest - planning and execution. In this mode, the aerobot must be well equipped with advanced AI algorithms, for understanding the prevailing scenarios or context within which mission has been given, planning the mission, and executing the mission.

5.5 Tier-to-tier Boundary

Between tier one and tier two is the supervisory/manual control boundary. In other words, most of the operation performed in tier one is manual operator control, whereas tier two is mostly supervisory. The boundary separating tier-two from tier-three is the boundary of "absolute trust". For the aerobot autonomy to progress beyond this point, the operator must be willing to trust the aerobot's decisions absolutely.

5.6 AFRL vs nCA vs ONR Autonomy Control Levels

This section relates how the nCA level model maps to the ONR's and Clough (2002) AFRL ACL models. In 2000, a research committee for the Office of Naval Research (ONR), proposed six robotic autonomy level (Horrigan and ONR's Committee, 2000). In 2002, the Airforce Research Laboratory (AFRL) UAV program motivated Clough (2002) research into developing a measurement metrics for autonomous vehicles (Clough, 2002). This was later adopted by the AFRL as a metric method of measuring their autonomous vehicles. Since then, software has been developed by various vendors and researchers based on this measurement metrics for used by government and military agencies (Sholes, 2007). These two widely recognised standard metrics for autonomy control levels were developed particularly for the military domain. The nCA model, which was developed in this research, aimed to serve a similar purpose in the evolving civilian application domain of autonomous vehicles. These three ACL models are compared side-by-side, with similar levels mapped to each other, as shown in Figure 5.6.

The AFRL's eleven-level model maps to the nCA six-level model as shown in Figure 5.6. AFRL's level zero (Remotely piloted vehicle) maps to the three tier one component of the nCA

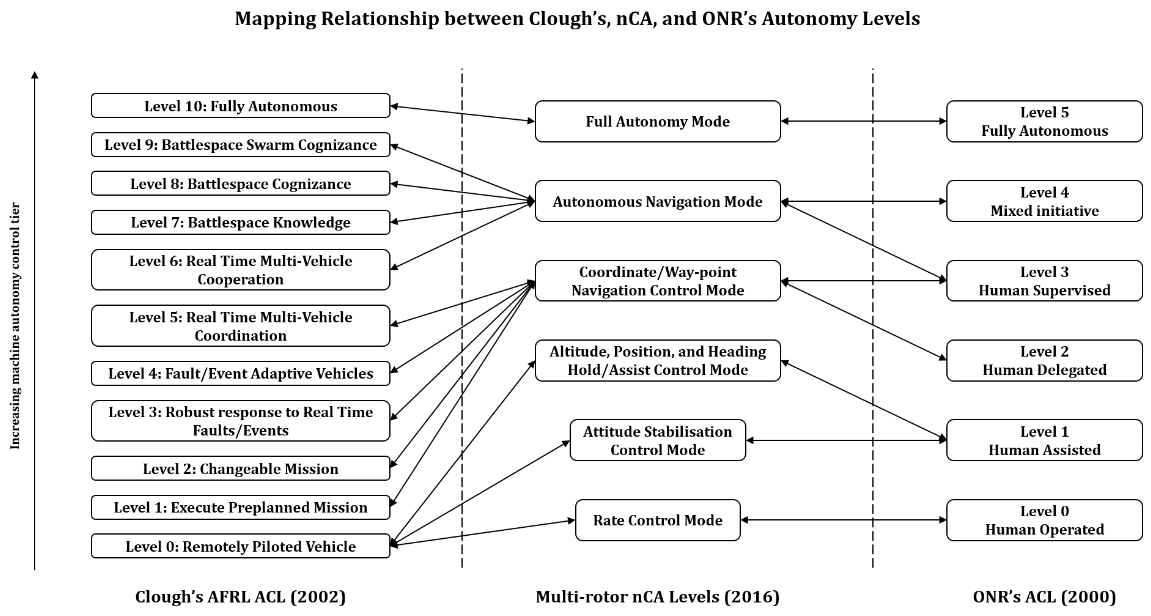

Mapping Relationship between Clough's, nCA, and ONR's Autonomy Levels
Increasing machine autonomy control tier
Level 10: Fully Autonomous
Level 9: Battlespace Swarm Cognizance
Level 8: Battlespace Cognizance
Level 7: Battlespace Knowledge
Level 6: Real Time Multi-Vehicle Cooperation
Level 5: Real Time Multi-Vehicle Coordination
Level 4: Fault/Event Adaptive Vehicles
Level 3: Robust response to Real Time Faults/Events
Level 2: Changeable Mission
Level 1: Execute Preplanned Mission
Level 0: Remotely Piloted Vehicle
Full Autonomy Mode
Autonomous Navigation Mode
Coordinate/Way-point Navigation Control Mode
Altitude, Position, and Heading Hold/Assist Control Mode
Attitude Stabilisation Control Mode
Rate Control Mode
Level 5 Fully Autonomous
Level 4 Mixed initiative
Level 3 Human Supervised
Level 2 Human Delegated
Level 1 Human Assisted
Level 0 Human Operated
Clough's AFRL ACL (2002)
Multi-rotor nCA Levels (2016)
ONR's ACL (2000)

Figure 5.6: Mapping relationship between AFRL, nCA, and ONR autonomy levels.

model. AFRL's level 1-5 maps to the first component of the second tier of the nCA model. AFRL's level 6-9 maps to the second component of the second tier of the nCA model. A one-to-one mapping is observed in the highest column of both hierarchies, as AFRL's level 10 (fully autonomous) maps to the nCA third tiers only component, full autonomy mode.

The ONR's six-level model maps to the nCA six-level model as shown in Figure 5.6. Both the bottom and top most level of both models have a one-to-one mapping. The ONR's level 1 maps to the second and third components of the first tier of the nCA model. The ONR's level 2 maps to the first component of the second tier of the nCA model. The ONR's level 3 maps to both components of the second tier of the nCA model. The ONR's level 4 maps to the second component of the second tier of the nCA model.

The mapping relationships of the nCA model with the AFRL and ONR models describes how robotic systems classified in any of these scales could be re-classified under the nCA scale. It also shows how robust and fully representative the nCA scale is, of previous models and domain application, despite being designed for the emerging multirotor aerial systems in the civilian domain.

5.7 Chapter Conclusion

Because of the limitations of the existing autonomy models (such as the AFRL and ONR), in aerobot navigational control autonomy classification, a new autonomy classification model called the navigational control autonomy (nCA) model was proposed and developed. It was compared with other existing models. The nCA model differs from the other models in that it eliminates step-size increment, favours a fuzzy classification between levels, presents levels in a pyramidal order, and clearly maps the limits of aerobotic control interfaces, such as the RC joystick controller. The nCA model was used as a basis for discussing the unimodal human aerobotic control interfaces, in Chapter 2. For example, the nCA model was used to identify a control void beyond the tier-one components of the nCA model. The popular RC joystick controller is very effective for tier-one interactions, because it has a high refresh rate capable of handling such control. Nevertheless, beyond the nCA tier-one classes, it becomes un-intuitive to use the RC controller. The higher autonomy requires an effective complementing high-level interaction interface. The nCA model was used as the basis for further discussions in subsequent Chapters of this research work.

Chapter 6

The Multimodal Speech and Visual Gesture (mSVG) UAV Control Interface

This chapter describes the design and development of the multimodal speech and visual gesture (mSVG) interface for use in human aerobotic interaction. The functional block diagram of the mSVG operation is presented and discussed. The computations are symbolically described using mathematical notations and logic. In the development of the mSVG system, example applications were developed to demonstrate its operations. These applications were numerically simulated using MATLAB and graphically simulated using a combination of Python, ROS, and the Gazebo Simulator. The examples were based on the nCA (navigation control autonomy) Tier 1-III and nCA Tier 2-I models discussed in Chapter 5.

6.1 Multimodal Speech and Visual Gesture (mSVG) Model

The mSVG technique is fundamentally a multimodal combination of speech and visual gesture, a method that leverages familiar human-to-human type interaction in human aerobotic interaction. This combination could be simultaneous, sequential, or complementary. The underlying architecture of how this technique is designed to work is as described in Figure 6.1. Let the speech and visual gesture input be s and v respectively, and let f and g be the respective processing functions, which generate the control symbols $f(s)$ and $g(v)$ as shown in Figure 6.1.

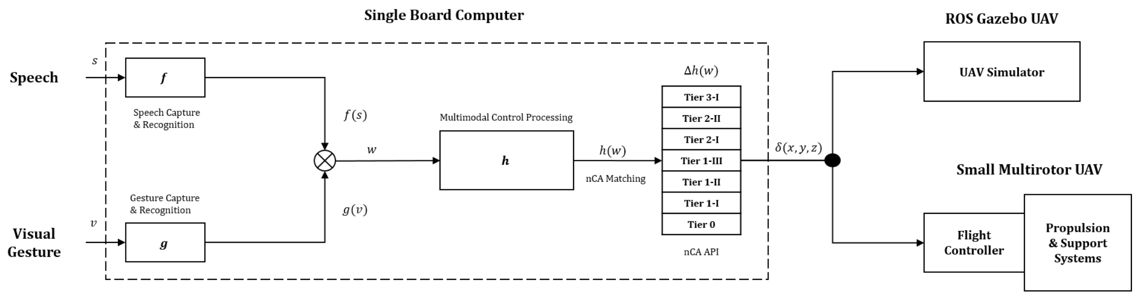


Figure 6.1: mSVG design architecture control capture, processing, and execution.

Then the control command w processed at the multimodal control processing unit (MCPU) is a function of control symbols $f(s)$ and $g(v)$ as shown below:

$$w = f(s) \otimes g(v) \tag{6.1}$$

Where $\otimes$ is the combination function that sequences, orders, and timestamps incoming control symbols, determining if they are simultaneous, sequential, or complementary; after which the combined control input symbol w is generated and passed on to the MCPU. $h(w)$ is the output of the MCPU, containing the step-by-step instructions required by the nCA autonomy model API layer matching $h(w)$ with the coordinate increment/decrement parameters $\delta(x, y, z)$ via a delta Δ parameter, which depends on the nCA autonomy level of the UAV. This is described by the following equation as:

$$\delta(x, y, z) = \Delta h(w) \tag{6.2}$$

The delta parameter Δ is a function generated by the nCA API to modify the MCPU output, $h(w)$, to enable compatibility with different nCA navigational control autonomy levels. For the Tier 1-III nCA model component, $\Delta = 1$. Therefore, $\delta(x, y, z) = h(w)$. The coordinate increment/decrement parameters $\delta(x, y, z)$ specifies the change in the aerobots 3-dimensional position with respect to its current position.

6.1.1 nCA Tier 1-III application model

The processing operation in the multimodal control processing unit (MCPU) stage can be mathematically described through the use of relational set theories. The universal command set used consists of navigational and scenario commands, which are as presented in Table 6.1 and Table 6.3. The symbol "u" is a numeric modifier parameter specifying amount of navigational increment/decrement as used in Table 6.1. Control keyword and modifiers are also highlighted in block letters. The current position of the aerobot in the world environment is represented by

Table 6.1: Navigaitional commands and control expresisions with example usage

S/N	Navigational Command	Control Expression	Command Example	
1	Forward	$x + udx$	Go FORWARD HALF metre	$x + 0.5dx$
2	Backward	$x - udx$	Go BACKWARD ONE metre	$x + dx$
3	Right	$y + udy$	Step RIGHT	$y + 0.5dy$
4	Left	$y - udy$	Step LEFT TWO metres	$y - 2dy$
5	Up	$z + udz$	Climb UP ONE metre	$z + dz$
6	Down	$z - udz$	Go DOWN HALF metre	$z + 0.5dz$
7	Hover	$z = u$	HOVER THREE metres	$z = 3$
8	Land	$z = 0$	Land	$z = 0$

the x, y, and z components of a right hand coordinate system as shown in Figure 6.2. Where dx, dy, and dz are unit conversion parameter from simulation to world environment, for example $dx = dy = dz = 1$ in the simulation test.

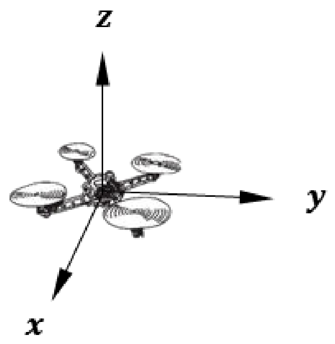


Figure 6.2: Right hand UAV navigational coordinate orientation.

Figure 6.3 is a decision tree showing the mapping relationship between the control command keywords, modifier keyword, and control expression executed. For a given control command, the command keyword is first identified, followed by the modifying keyword. These are then combined to determine the control expression to compute for the new navigation coordinate to be fed in to the UAV flight controller. The 3-D navigation decision tree was used in both the MATLAB numerical simulation program and the Python ROS Gazebo graphical simulation program.

The use of the term “relative 3-D navigation” in Figure 6.3 is a reference to the fact that navigation coordinates are computed with respect to the current UAV position when the command was issued. Also, the modifier keywords are numeric values e.g. $\frac{1}{2}$, 1, 2, etc. accompanied by a unit of distance measure such as metres or feet. But what happens when an operator issues a control command with two or more primary command keywords, such as a speech com-

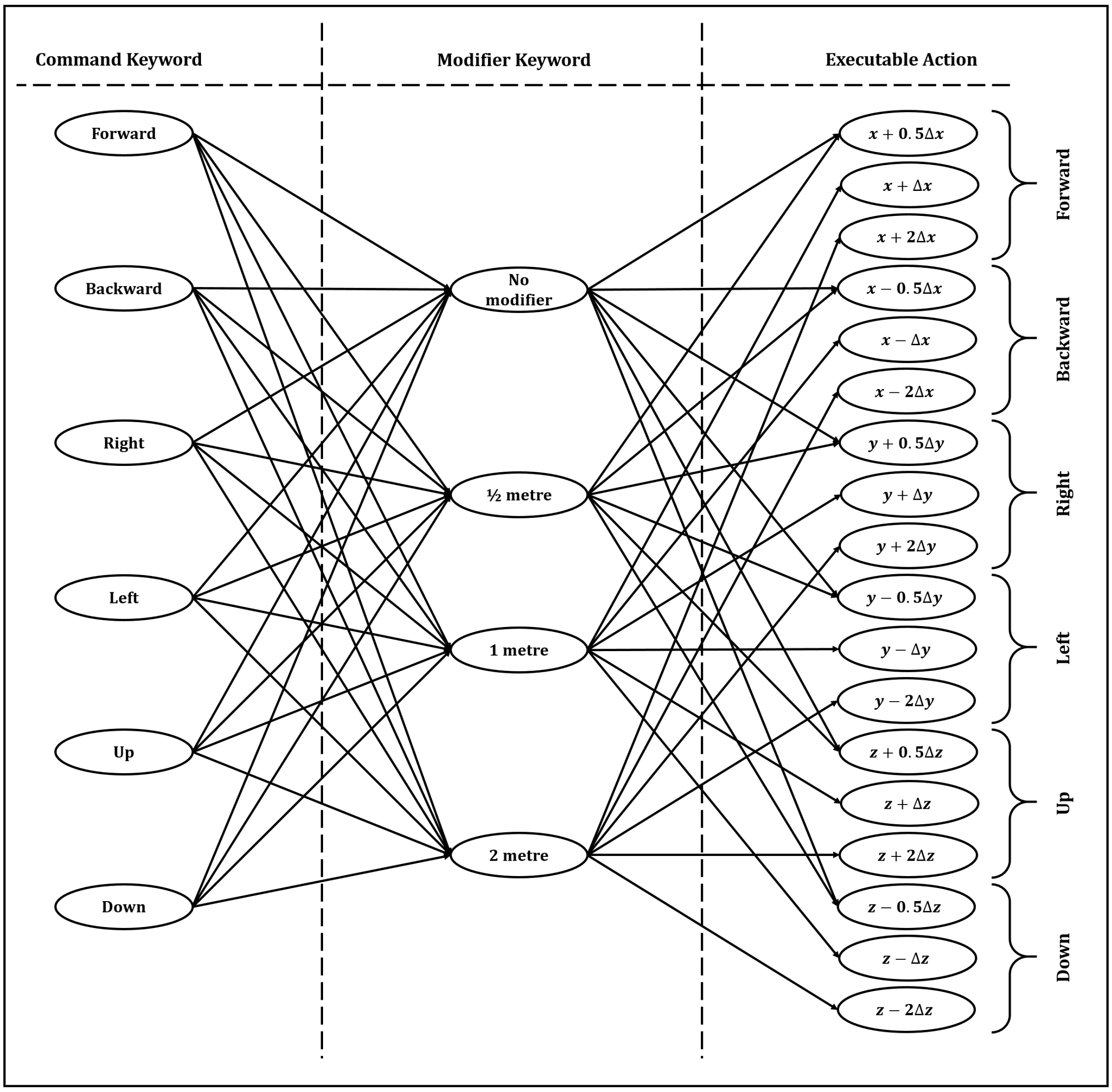


Figure 6.3: Relative 3-D Navigation Symbol Decision Tree.

mand "go forward backward two half metre"? Or contradictory commands such as "go forward [speech]" while "pointing backward [pointing gesture]"? The conflict resolution strategy for this was to disregard all the preceding primary keywords in the control command and execute the last keyword in the command sequence. The case of more than one primary keyword was successfully tested in the MATLAB numerical simulation using this strategy. The MATLAB and Python program code listings for the numerical and graphical test simulation is given in Appendix C.2 and Appendix C.3 respectively. A basic algorithm for computing nCA Tier 1-III task is presented in Table 6.2.

Table 6.2: nCA Tier 1-III navigation computation example algorithm.

A. Order/sequence control symbol as captured in time

B. Log current UAV position at command time

C. From the control symbol, identify if navigation or scenario command:

   1. if navigation command - go to nCA Tier 1-III program loop (step D)
   2. if scenario command - go to nCA Tier 2-I program loop

D. If navigation command, identify which of the eight navigation controls is needed

E. Identify modifying parameters 'u' value

F. Use decision tree to identify control expression, from table, based on navigation command and modifying parameter

G. Compute expression using both the UAV position and modifier parameter e.g. "$x + u * dx$"

H. Update flight controller with new coordinate, specifying velocity, and direction of travel

6.1.2 nCA Tier 2-I application model/library

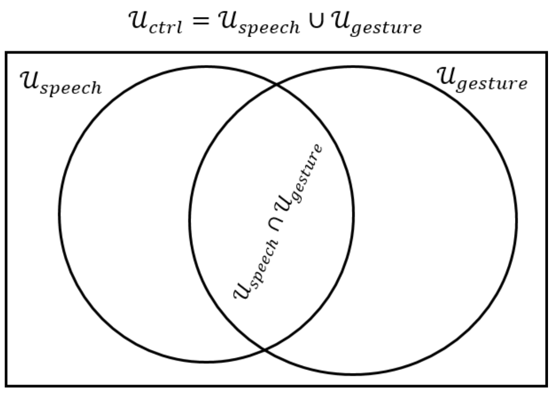

$u_{ctrl} = u_{speech} \cup u_{gesture}$
u_{speech}
$u_{gesture}$
$u_{speech} \cap u_{gesture}$

Figure 6.4: Speech and Gesture control command set model.

Let us consider that the universal set of control commands u_{ctrl}, listed in Table 6.1 and Table 6.3, could be issued as either speech or gestures commands. Then the universal command set can be described as $u_{ctrl} = u_{speech} \cup u_{gesture}$. Where u_{speech} is speech only commands, and $u_{gesture}$ is gesture only commands. Figure 6.4 presents a set model describing this relationship. Commands that can be issued via either speech or gestures are represented as commands found at the intersection of both method $u_{common} = u_{speech} \cap u_{gesture}$.

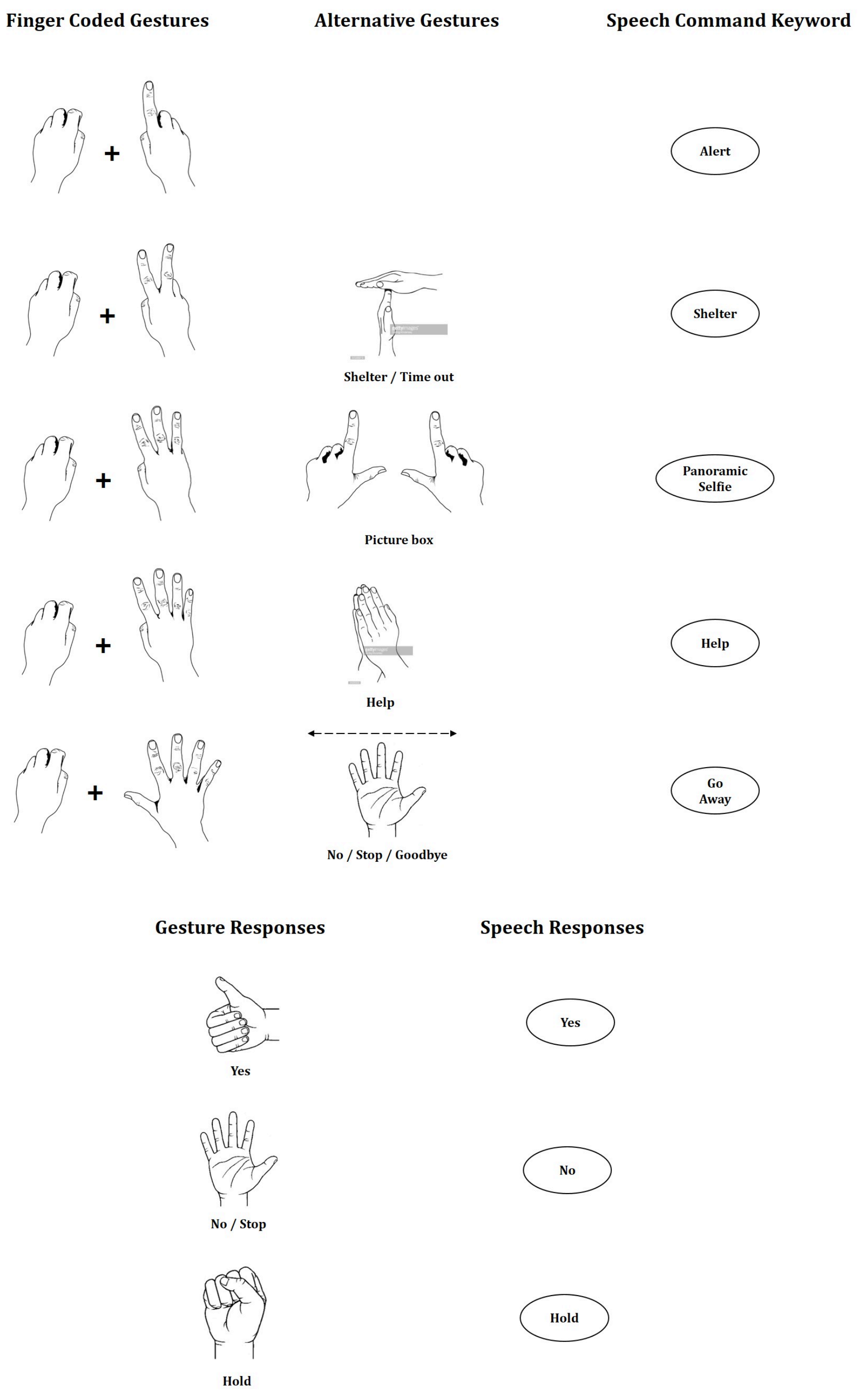


Figure 6.5: Finger coded climber gestures and corresponding speech commands.

For the first phase of this work, all commands were implemented as speech commands, and only a few higher level scenario commands were implemented as gestures as described in Figure 6.5. Lets consider an hypothetical set of speech commands,

$$u_{speech} = [\{ok\}, \{weather\}, \{signpost\}, \{there\}, \{that\}, \{selfie\}, etc.] \tag{6.3}$$

Which can be denoted as

$$u_{speech} = [s_1, s_2, s_3, ..., s_n] \tag{6.4}$$

Where n is the total set of climber-user speech control vocabulary available. Similarly, $u_{gesture}$ could be an hypothetical set of climbers' visual gesture commands,

$$\begin{aligned} u_{gesture} = [&\{\text{ok - thumbs up}\}, \{\text{leave - wave away}\}, \{\text{what - open palm}\}, \\ &\{\text{that - point object}\}, \{\text{there - point place}\}, \\ &\{\text{selfie - picture board symbol}\}, etc.] \end{aligned} \tag{6.5}$$

Which can also be denoted as

$$u_{gesture} = [g_1, g_2, g_3, ..., g_n] \tag{6.6}$$

Where m is the total number of climber-user gesture control vocabulary available. Using these notations, a typical series of control commands could be a sequence of

$$(t_1, G_1), (t_2, G_2), (t_3, S_3), (t_4, G_4 + S_4), (t_5, S_5 + G_5), (t_6, S_6), etc. \tag{6.7}$$

Where t_i is the sequential time component, S_i is the speech command component, and G_i is the gesture command component. These commands could be sequential, for example - (t_1, S_1) followed by (t_2, G_2) and so on; or simultaneous, as in - $(t_4, G_4 + S_4), (t_5, S_5 + G_5), etc.$ While sequential commands consist of only one gesture or speech component in each time component t_i, simultaneous commands consist of both speech and gesture components at the same time component t_i. In spatio-temporal terms, a speech and gesture command is considered simultaneous if the time between capture is no more than $0.5s$, otherwise it is considered a sequential command and one would be executed after the other. In other words

$$\text{Command Selection} = \begin{cases} \text{Simultaneous}(G + S) & \text{if } t \leq 0.5s \\ \text{Sequential}(G, S) & \text{if } t > 0.5s \end{cases} \tag{6.8}$$

Simultaneous command could be emphatic or complementary. A simultaneous command is emphatic if it repeats the same command using the alternative modality, whereas it is complementary when it provides additional information not given in the alternative modality. For example, *"Hold* (Speech) *+ Fist* (Gesture for hold)*"* issued within $0.5s$ apart only emphasises the command for the UAV to "hold" its position. Whereas a command *"Go Forward* (Speech) *+ Two-fingers* (Gesture for numeric modifier two)*"* issued within $0.5s$ of each other, results in the aerobot advancing two metres in the Forward direction. In this case, the gesture command complements the speech command.

Table 6.3: Scenario commands and synonyms

S/N	Scenario Command	Command Synonyms
1	Alert	-
2	Shelter	-
3	Panoramic Selfie	Panoramic, Selfie, Panorama
4	Help	Operator
5	Go Away	Away, Patrol

6.2 mSVG Design and Simulation of nCA Tier 2-I Application Scenarios

The last section discussed navigational control command operations. This section discusses examples of scenario command operations that could be executed by a search and rescue patrol aerobot in the wild. Navigation commands emulate low level nCA interaction while scenario commands emulate higher nCA level models. Table 6.3 presents some scenario commands.

6.2.1 Shelter command computation

The UAV knowledge base includes the UAVs world map shown in Figure 6.6, as a computable lookup table, accessible during multimodal control processing. Figure 6.6 shows the crevice and shelter map/table implemented in the MATLAB and Python based Gazebo Simulation test.

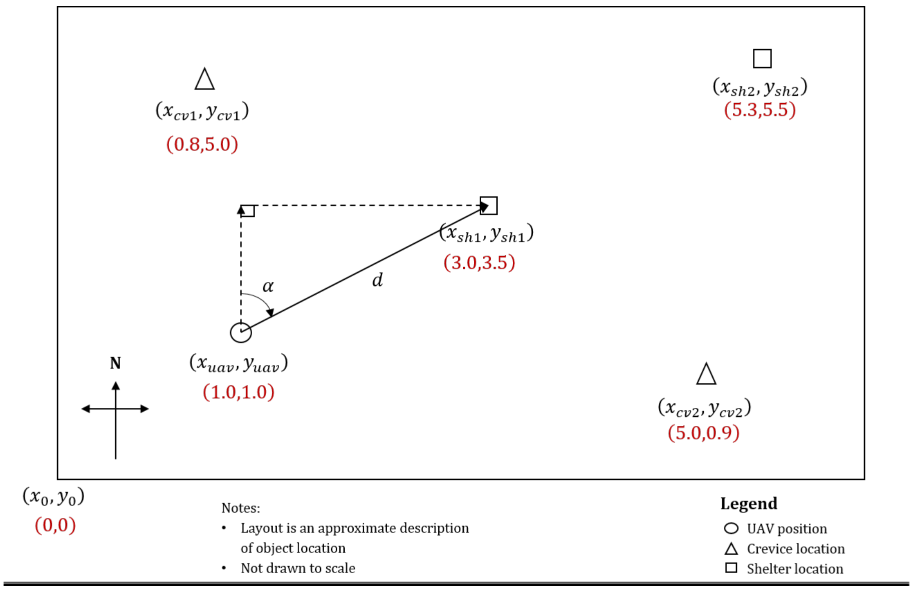


Figure 6.6: UAV world map showing UAV position, crevices, and shelter locations.

From Figure 6.6, the relative North degree direction of the crevices, shelters, and other objects of interests can be computed as follows:

$$\tan\alpha = \frac{y_{sh1} - y_{uav}}{x_{sh1} - x_{uav}} \tag{6.9}$$

Therefore, the bearing angle

$$\alpha = \arctan(\frac{y_{sh1} - y_{uav}}{x_{sh1} - x_{uav}}) \tag{6.10}$$

The distance between the UAV and the shelter shown in Figure 6.6, can be computed as

$$d^2 = (x_{sh1} - x_{uav})^2 + (y_{sh1} - y_{uav})^2 \tag{6.11}$$

Therefore, the nearest shelter to the climber can be said to be *"d km in the α° North direction"*, where x and y components are measured in km. In general, the equations for computing the distance and direction of any object located at point 'x, y' on the map, from the user/-climber/operator/UAV 'x_{uav}, y_{uav}' are

$$\alpha = \arctan(\frac{y - y_{uav}}{x - x_{uav}}) \tag{6.12}$$

$$d = \sqrt{(x - x_{uav})^2 + (y - y_{uav})^2} \tag{6.13}$$

Interpreted as the map object of interest is *"d km in the α° North direction"* from the user. In addition to these, the UAV could beam a *"red arrow"* light on the ground with the arrow pointing in the direction of travel. Also, the UAV could travel the initial 50 metres with the climber, before flying away.

In the MATLAB simulation, the computation for the shelter command scenario is performed by the code snippet in Listing 6.1. The code loop in this snippet gets executed when triggered by the "shelter" keyword in a speech command such as "where is the nearest shelter?" The Python graphical simulation was similar to the MATLAB simulation. The full code listing for both the MATLAB and Python programs are given in Appendix C.2 and Appendix C.3 respectively.

Listing 6.1: MCPU nearest shelter processing

```
        % compute nearest shelter or crevice
        for n = 1:1:length(uav_world_map)
            if char(uav_world_map(n,1)) == 'shelter'
                % retrieving shelter "x,y" coordinates
                x_sh = str2double(uav_world_map(n,2));
                y_sh = str2double(uav_world_map(n,3));

                % retrieving shelter "x,y" coordinates
                x_uav = quad_pos(i+1,1);
                y_uav = quad_pos(i+1,2);

%                 % uncomment to see the result of each shelter location computed
%                 d_sh = sqrt(((x_sh-x_uav)^2)+((y_sh-y_uav)^2))
%                 alpha_sh = atan2d((y_sh-y_uav),(x_sh-x_uav));    % MATLAB four quadrant tan inverse in degress
%                 % converting from trigonometric angles to compass north bearing angles - see logbook V (Fri 25-08-2017) for details
%                 if ((x_sh-x_uav) < 0) && ((y_sh-y_uav) > 0)     % for 2nd quadrant
%                     alpha_sh_bearing = 450 - alpha_sh
%                 else
%                     alpha_sh_bearing = 90 - alpha_sh       % for 1st, 3rd, and 4th quadrant
%                 end

                d_sh = sqrt(((x_sh-x_uav)^2)+((y_sh-y_uav)^2));
                % identifying nearest shelter
                if d_sh < d_sh_nearest

                    alpha_sh = atan2d((y_sh-y_uav),(x_sh-x_uav));    % MATLAB four quadrant tan inverse in degress

                    % converting from trigonometric angles to compass north bearing angles - see logbook V (Fri 25-08-2017) for details
                    if ((x_sh-x_uav) < 0) && ((y_sh-y_uav) > 0)     % for 2nd quadrant
                        alpha_sh_bearing = 450 - alpha_sh;
                        quad_pos(i+1,4) = alpha_sh_bearing;
                    else
                        alpha_sh_bearing = 90 - alpha_sh;       % for 1st, 3rd, and 4th quadrant
                        quad_pos(i+1,4) = alpha_sh_bearing;
                    end

                    % updating distance to nearest shelter, and name of shelter
                    d_sh_nearest = d_sh;
                    scenario_place = uav_world_map(n,4);
                end
            end
```

6.2.2 Panoramic selfie command computation

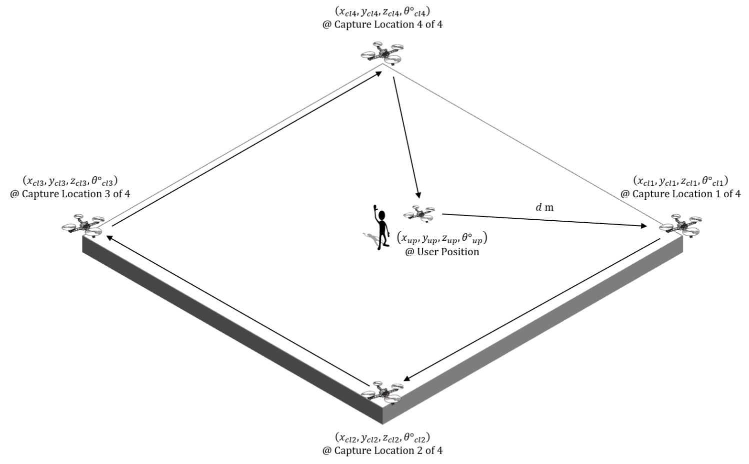
From Figure 6.7a, capture location 1 components can be computed as

$$x_{cl1} = x_{up} - d \sin\theta_{up} \tag{6.14}$$

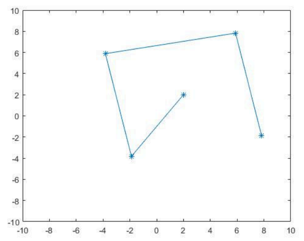
$$y_{cl1} = y_{up} - d \cos\theta_{up} \tag{6.15}$$

For capture location 2 - 4 :

$$x_b = x_a + \sqrt{d^2 + d^2 \sin(\theta_a + 90)} \tag{6.16}$$


$(x_{cl4}, y_{cl4}, z_{cl4}, \theta°_{cl4})$
@ Capture Location 4 of 4
$(x_{cl3}, y_{cl3}, z_{cl3}, \theta°_{cl3})$
@ Capture Location 3 of 4
$(x_{cl1}, y_{cl1}, z_{cl1}, \theta°_{cl1})$
@ Capture Location 1 of 4
d m
$(x_{up}, y_{up}, z_{up}, \theta°_{up})$
@ User Position
$(x_{cl2}, y_{cl2}, z_{cl2}, \theta°_{cl2})$
@ Capture Location 2 of 4

(a) Panoramic selfie capture location description.



(b) Panoramic MATLAB computation location plot with UAV initially facing 33 deg North.

Figure 6.7: Panoramic selfie capture.

$$y_b = y_a + \sqrt{d^2 + d^2 \cos(\theta_a + 90)} \tag{6.17}$$

Where subscript 'a' denoted parameters refers to parameters from the previous location, and 'b' subscripted parameters refers to the parameters for the current location being computed.

In the MATLAB simulation, the computation for the panoramic selfie command scenario is performed by the code snippet in Listing 6.2. The code loop in this snippet gets executed when triggered by the "Panoromic" keyword or any of its command synonyms in a speech or gesture command. This is based on the computation described in this section.

Listing 6.2: MCPU panoramic selfie processing.

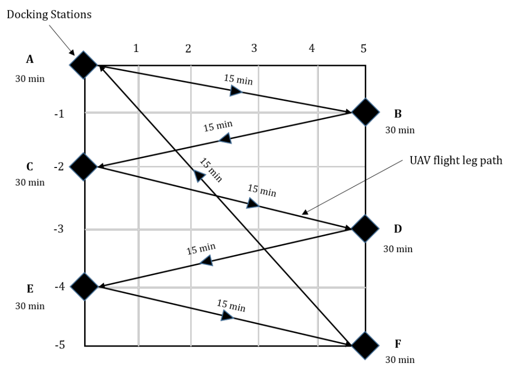
```
    % program codes to guide the execution of 'scenario_3' i.e. panoramic selfie
    if char(execute) == 'scenario_3'

        % start video record
        vid_record = 'true';

        % collecting uav parameters for computation
        % current uav position and orientation
        x_up = quad_pos(i+1,1);
        y_up = quad_pos(i+1,2);
        z_up = quad_pos(i+1,3);
        theta_up = quad_pos(i+1,4);
        % theta_up = 315;

        d_capture = 7;     % image capture distance from user to capture location is 7 meres
        alt_capture = 3;   % defining image capture altitude - 3 metres

        % log scenario nav locations
        pan_selfie_log = {'location','x','y','z','theta';
                          'user_pos.',x_up,y_up,z_up,theta_up
                          };
        x_capt(1)=x_up;
        y_capt(1)= y_up;

        % computing capture location 1
            x_cl1 = x_up - (d_capture * sind(theta_up));
            y_cl1 = y_up - (d_capture * cosd(theta_up));
            z_cl1 = alt_capture;
            theta_cl1 = theta_up;

            % go to computed location 1
            quad_pos(i+1,1:4)=[x_cl1,y_cl1,z_cl1,theta_cl1];

            % camera takes picture - while facing user
            cam_snap = 'true_@_1';

            % log data
            pan_selfie_log(3,:) = {'cap_loc_1',x_cl1,y_cl1,z_cl1,theta_cl1};
            x_capt(2)=x_cl1;
            y_capt(2)= y_cl1;

        % computing capture location 2 (rotating -45 degrees)
            x_cl2 = x_cl1 + (sqrt(2*(d_capture^2)) * sind(theta_cl1 - 45));
            y_cl2 = y_cl1 + (sqrt(2*(d_capture^2)) * cosd(theta_cl1 - 45));
            z_cl2 = alt_capture;
            theta_cl2 = theta_cl1 - 45;

            % go to computed location 2
            quad_pos(i+1,1:4)=[x_cl2,y_cl2,z_cl2,theta_cl2];

            % camera takes picture - while facing user
            cam_snap = 'true_@_2';

            % log data
            pan_selfie_log(4,:) = {'cap_loc_2',x_cl2,y_cl2,z_cl2,theta_cl2};
            x_capt(3)=x_cl2;
            y_capt(3)= y_cl2;

        % computing capture location 3 (rotating +90 degrees)
            x_cl3 = x_cl2 + (sqrt(2*(d_capture^2)) * sind(theta_cl2 + 90));
            y_cl3 = y_cl2 + (sqrt(2*(d_capture^2)) * cosd(theta_cl2 + 90));
            z_cl3 = alt_capture;
            theta_cl3 = theta_cl2 + 90;

            % go to computed location 3
            quad_pos(i+1,1:4)=[x_cl3,y_cl3,z_cl3,theta_cl3];

            % camera takes picture - while facing user
            cam_snap = 'true_@_3';

            % log data
            pan_selfie_log(5,:) = {'cap_loc_3',x_cl3,y_cl3,z_cl3,theta_cl3};
            x_capt(4)=x_cl3;
            y_capt(4)= y_cl3;

        % computing capture location 4 (rotating +90 degrees)
            x_cl4 = x_cl3 + (sqrt(2*(d_capture^2)) * sind(theta_cl3 + 90));
            y_cl4 = y_cl3 + (sqrt(2*(d_capture^2)) * cosd(theta_cl3 + 90));
            z_cl4 = alt_capture;
            theta_cl4 = theta_cl3 + 90;

            % go to computed location 4
            quad_pos(i+1,1:4)=[x_cl4,y_cl4,z_cl4,theta_cl4];

            % camera takes picture - while facing user
            cam_snap = 'true_@_4';

            % log data
            pan_selfie_log(6,:) = {'cap_loc_4',x_cl4,y_cl4,z_cl4,theta_cl4};
            x_capt(5)=x_cl4;
            y_capt(5)= y_cl4;

        % UAV returning to user's location
            % go to UAV's initial position
            quad_pos(i+1,1:4)=[x_up,y_up,z_up,theta_up];

            % log data
            pan_selfie_log(7,:) = {'user_loc.',x_up,y_up,z_up,theta_up}
            %x_capt(6)= x_up;
            %y_capt(6)= y_up;

        figure
        plot(x_capt,y_capt,'-*')
        xlim([-10 10])
        ylim([-10 10])
        % axis equal

        vid_record = 'false';
    end
```

6.2.3 Patrol command computation

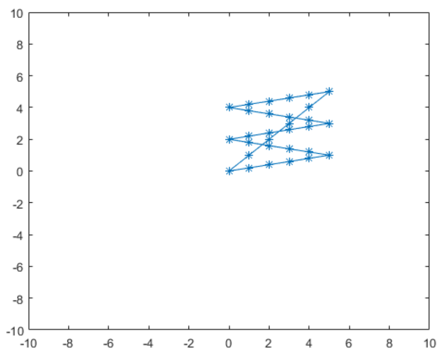
As was shown in Figure 6.8a, the patrol operation can be broken down into the following components: 1) UAV flies at 0.5 m altitude, 2) Briefly stopping at 4 intermediate stop points between two docking stations (e.g. A and B) to pan and scan area under. The stop points between point A and point B are computed using the following expression in both the MATLAB and Python implementation,

$$(x_n, y_n) = (x_p + 0.2(x_b - x_a), y_p + 0.2(y_b - y_a)) \tag{6.18}$$

Note that this expression generates five points because the goal point B is included in the computation. Where p is the previous state and n is the next state. And the 'a' and 'b' subscripted coordinate 'x' and 'y' corresponds to the UAV's location A and B coordinate component. Figure 6.9 shows this python based ROS Gazebo implementation of the UAV patrol operation.

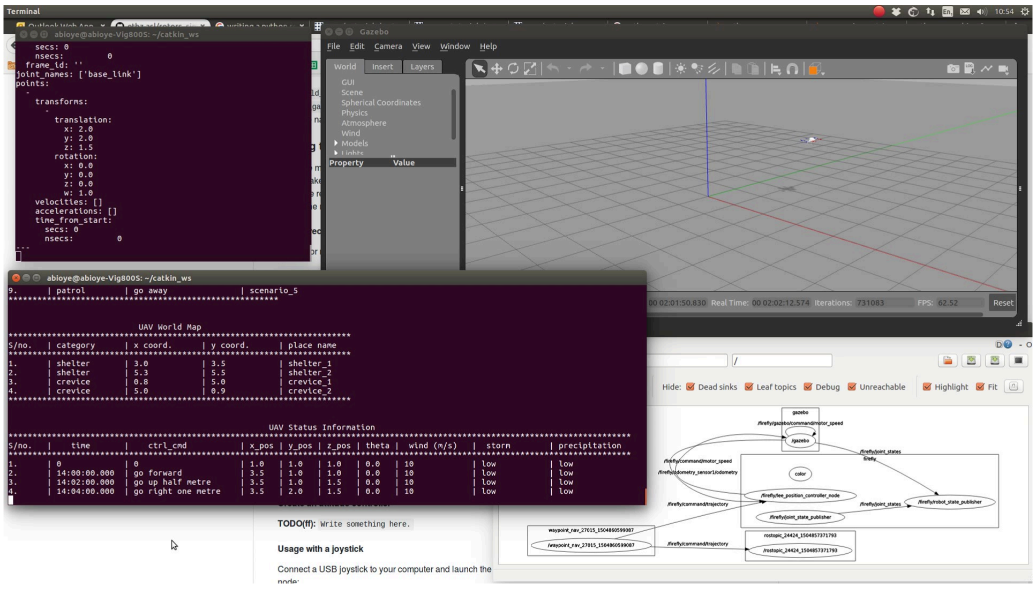


(a) Designed 5×5 km^2 patrol grid scenario



(b) MATLAB patrol operation route plot

Figure 6.8: Patrol control operation simulation in MATLAB.


```
abioye@abioye-Vig800S: ~/catkin_ws
    secs: 0
    nsecs:         0
  frame_id: ''
joint_names: ['base_link']
points:
  -
    transforms:
      -
        translation:
          x: 2.0
          y: 2.0
          z: 1.5
        rotation:
          x: 0.0
          y: 0.0
          z: 0.0
          w: 1.0
    velocities: []
    accelerations: []
    time_from_start:
      secs: 0
      nsecs:         0
---
```

```
abioye@abioye-Vig800S: ~/catkin_ws
9.      | patrol        | go away               | scenario_5
********************************************************


                         UAV World Map
***********************************************************************
S/no.   | category      | x coord.        | y coord.        | place name
***********************************************************************
1.      | shelter       | 3.0             | 3.5             | shelter_1
2.      | shelter       | 5.3             | 5.5             | shelter_2
3.      | crevice       | 0.8             | 5.0             | crevice_1
4.      | crevice       | 5.0             | 0.9             | crevice_2
***********************************************************************


                                                   UAV Status Information
*********************************************************************************************************************************
S/no.   |    time       |    ctrl_cmd            | x_pos | y_pos | z_pos | theta |  wind (m/s)    |  storm          | precipitation
*********************************************************************************************************************************
1.      | 0             | 0                      | 1.0   | 1.0   | 1.0   | 0.0   | 10             | low             | low
2.      | 14:00:00.000  | go forward             | 3.5   | 1.0   | 1.0   | 0.0   | 10             | low             | low
3.      | 14:02:00.000  | go up half metre       | 3.5   | 1.0   | 1.5   | 0.0   | 10             | low             | low
4.      | 14:04:00.000  | go right one metre     | 3.5   | 2.0   | 1.5   | 0.0   | 10             | low             | low
```

TODO(ff): Write something here.

Usage with a joystick

Connect a USB joystick to your computer and launch the node:

Terminal
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time: 00 02:02:12.574 Iterations: 731083 FPS: 62.52 Reset
Hide: Dead sinks Leaf topics Debug Unreachable Highlight Fit
gazebo
/gazebo
/firefly/command/motor_speed
/firefly/odometry_sensor1/odometry
/firefly/lee_position_controller_node
/firefly/joint_state_publisher
/firefly/robot_state_publisher
/firefly/joint_states
/firefly/command/trajectory
waypoint_nav_27015_1504860599087
/waypoint_nav_27015_1504860599087
rostopic_24424_1504857371793
/rostopic_24424_1504857371793

Figure 6.9: Python implementation with RotorS ROS Gazebo UAV simulator.

Listing 6.3 is a code snippet for performing the $5 \times 5\ km^2$ patrol grid scenario command in the MATLAB simulation. This was graphically simulated in the ROS Gazebo UAV simulator using Python. The code loop in this snippet gets executed when triggered by the “Patrol” or “Away” speech keyword or after a user interaction with a UAV has ended. The computation in the code listing is based on Equation 6.18.

Listing 6.3: MCPU $5 \times 5\ km^2$ patrol grid scenario processing.

```
% uav patrol loop navigation goes here
if uav_patrol == true

    % defining intial patrol elements
    % x_a = 0;
    % y_a = 0;
    [x_a, y_a]=deal(0,0);
    [x_b, y_b]=deal(5,1);
    [x_c, y_c]=deal(0,2);
    [x_d, y_d]=deal(5,3);
    [x_e, y_e]=deal(0,4);
    [x_f, y_f]=deal(5,5);
    z = 0.05;

    % computing theta ab direction
    theta_ab = 90 - atan2d((y_b - y_a),(x_b - x_a));      % MATLAB four quadrant tan inverse in degress
                                                           % first quadrant computation

    % log scenario nav locations
    patrol_log = {'uav_loc.','x_(km)','y_(km)','z_(km)','theta';
                  'A',0,0,0.05,theta_ab
                  };

    % intializing loop variables
    x_ab = x_a;
    y_ab = y_a;

    % traveling location A to B
    for ab = 1:1:5
```

```
        x_ab = x_ab + (0.2 * (x_b - x_a));
        y_ab = y_ab + (0.2 * (y_b - y_a));
        uav_loc = ['AB',num2str(ab)];

        if ab == 5
            uav_loc = 'B';
        end

        patrol_log(ab+2,:) = {uav_loc,x_ab,y_ab,z,theta_ab};
    end


    % traveling location B to C
    % computing theta bc direction
    theta_bc = 90 - atan2d((y_c - y_b),(x_c - x_b));    % MATLAB four quadrant tan inverse in degress
                                                         % third quadrant computation
        x_bc = x_b;
    y_bc = y_b;

    for bc = 1:1:5
        x_bc = x_bc + (0.2 * (x_c - x_b));
        y_bc = y_bc + (0.2 * (y_c - y_b));
        uav_loc = ['BC',num2str(bc)];

        if bc == 5
            uav_loc = 'C';
        end

        patrol_log(bc+7,:) = {uav_loc,x_bc,y_bc,z,theta_bc};
    end

    % traveling location C to D
    % computing theta cd direction
    theta_cd = 90 - atan2d((y_d - y_c),(x_d - x_c));    % MATLAB four quadrant tan inverse in degress
                                                         % first quadrant computation
        x_cd = x_c;
    y_cd = y_c;

    for cd = 1:1:5
        x_cd = x_cd + (0.2 * (x_d - x_c));
        y_cd = y_cd + (0.2 * (y_d - y_c));
        uav_loc = ['CD',num2str(cd)];

        if cd == 5
            uav_loc = 'D';
        end

        patrol_log(cd+12,:) = {uav_loc,x_cd,y_cd,z,theta_cd};
    end

    % traveling location D to E
    % computing theta de direction
    theta_de = 90 - atan2d((y_e - y_d),(x_e - x_d));    % MATLAB four quadrant tan inverse in degress
                                                         % first quadrant computation
        x_de = x_d;
    y_de = y_d;

    for de = 1:1:5
        x_de = x_de + (0.2 * (x_e - x_d));
        y_de = y_de + (0.2 * (y_e - y_d));
        uav_loc = ['DE',num2str(de)];

        if de == 5
            uav_loc = 'E';
        end

        patrol_log(de+17,:) = {uav_loc,x_de,y_de,z,theta_de};
    end

    % traveling location E to F
    % computing theta ef direction
    theta_ef = 90 - atan2d((y_f - y_e),(x_f - x_e));    % MATLAB four quadrant tan inverse in degress
                                                        % first quadrant computation
        x_ef = x_e;
    y_ef = y_e;

    for ef = 1:1:5
        x_ef = x_ef + (0.2 * (x_f - x_e));
        y_ef = y_ef + (0.2 * (y_f - y_e));
        uav_loc = ['EF',num2str(ef)];

        if ef == 5
            uav_loc = 'F';
        end

        patrol_log(ef+22,:) = {uav_loc,x_ef,y_ef,z,theta_ef};
    end

    % traveling location F to A
    % computing theta fa direction
    theta_fa = 90 - atan2d((y_a - y_f),(x_a - x_f));    % MATLAB four quadrant tan inverse in degress
                                                        % first quadrant computation
        x_fa = x_f;
    y_fa = y_f;

    for fa = 1:1:5
        x_fa = x_fa + (0.2 * (x_a - x_f));
        y_fa = y_fa + (0.2 * (y_a - y_f));
        uav_loc = ['FA',num2str(fa)];

        if fa == 5
            uav_loc = 'A';
        end

        patrol_log(fa+27,:) = {uav_loc,x_fa,y_fa,z,theta_fa};
    end

    patrol_log

    x_patrol = cell2mat(patrol_log(2:end,2));
    y_patrol = cell2mat(patrol_log(2:end,3));

    figure
    plot(x_patrol,y_patrol,'-*')
    xlim([-10 10])
    ylim([-10 10])
    % axis equal

    % animating patrol
    figure
    h = animatedline;
    xlim([-10 10])
    ylim([-10 10])
    for k = 1:length(x_patrol)
        addpoints(h,x_patrol(k),y_patrol(k));
        drawnow
        if rem(k-1,5) == 0 % stopping for one sec (30 min) at docking station
            pause(1)
        else
            pause(0.1)      % navigating slowly along patrol nav path (dock:path) ratio = (1:0.1)
        end

    end

end
```

6.3 mSVG system component development

The multimodal speech and visual gesture system development was performed in blocks as shown in Figure 6.1. The first block developed was the speech capture and recognition block, indicated as block '*f*' in Figure 6.1. The next block developed was the gesture capture and recognition block, indicated as block '*g*'. This was then followed by the discussion of the development of the multimodal control processing unit (MCPU), block '*h*'. The nCA API was already discussed in Chapter 5 with scenarios presented in the preceding sub-sections. And finally, the ROS Gazebo UAV simulator development was discussed. The mSVG system components are laid out as shown in Figure 6.10.

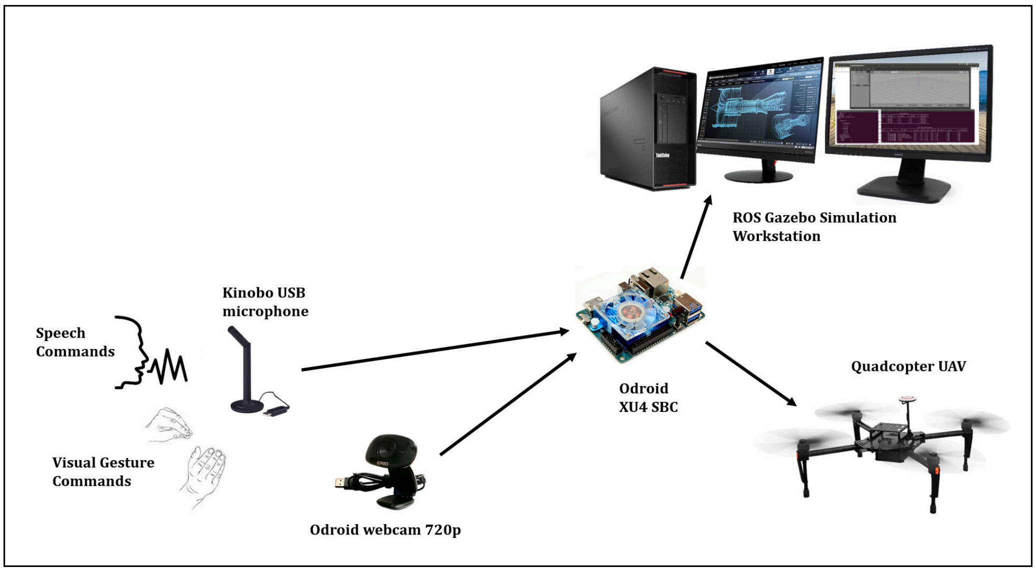


Figure 6.10: mSVG component setup.

6.3.1 Speech capture and recognition

Table 6.4 describes the speech capture and symbol processing algorithm. Speech is captured with the aid of a Kinobo USB microphone, processed and recognised using the CMU Sphinx ASR with a custom-defined phonetic dictionary; which contained a limited set of command vocabulary applicable for this research, in order to increase recognition speed and accuracy. The recognised speech command is encoded into a standard control symbol, which is then passed on to the MCPU, where it is decoded and processed for execution by a UAV. Listing 6.4 shows the serial and data logging initialisation code snippets.

Table 6.4: Speech capture and processing to control symbol algorithm.

A. Program initialisation:

   1. import relevant python libraries
   2. serial port and speech input data log file

B. Initialise the speech detector class

   1. specify audio parameters e.g. sample rate (16 KHz), channel (mono), model directory, speech dictionary, etc.
   2. setup microphone
   3. set noise/speech threshold

C. Listening loop - microphone listening for sound levels to exceed threshold

D. Capture - start recording sound if sound level exceeds threshold

E. Decode - recorded sound using the pocketsphinx asr library

F. Encode - convert decoded command into standardise control symbols

G. Relay - pass control command symbols to MCPU for further processing and command execution

H. Repeat - steps C to G loop continuously until the program is manually ended by the user

Listing 6.4: Speech control - initialising the serial port.

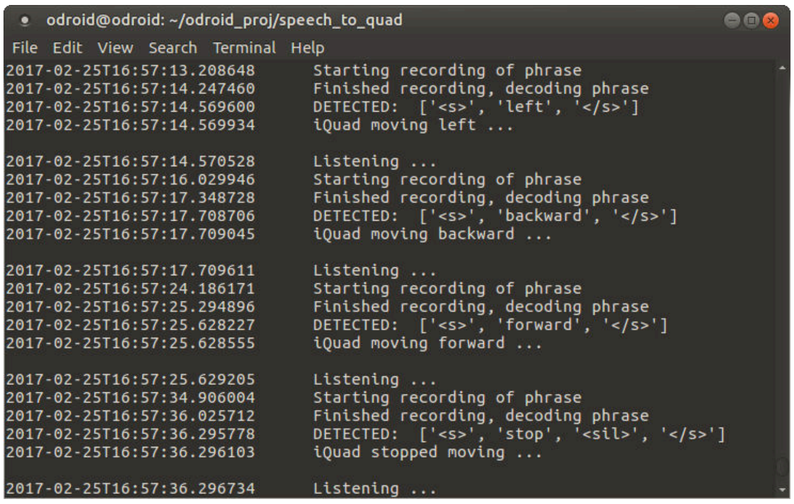
```python
port = "/dev/ttyACM0"
baud = 115200

ser = serial.Serial(port, baud, timeout=1)
filename = 'speech_input_' + datetime.now().strftime("%Y%m%d%H%M%S") + '.txt'
f = open(filename,'a')
f.write(datetime.utcnow().isoformat() + '\n')
f.flush()

#open the serial port
if ser.isOpen():
    #print(ser.name + ' is open...')
    datastring = ser.name + ' is open...' + '\n'
    print datastring
    f.write(datastring)
    f.flush()
```

Listing 6.5 is a code snippet of the listen, capture, decode, encode, relay, and repeat speech processing loop. The complete program code is listed in Appendix C.4.

Listing 6.5: Speech control processing.

```
        while True:
            cur_data = stream.read(self.CHUNK)
            slid_win.append(math.sqrt(abs(audioop.avg(cur_data, 4))))

            if sum([x > self.THRESHOLD for x in slid_win]) > 0:
                if started == False:
                    print datetime.utcnow().isoformat() + "\tStarting recording of phrase"
                    f.write(datetime.utcnow().isoformat() + '\tStarting recording of phrase ...\n')
                    f.flush()
                    started = True
                audio2send.append(cur_data)

            elif started:
                print datetime.utcnow().isoformat() + "\tFinished recording, decoding phrase"
                f.write(datetime.utcnow().isoformat() + '\tFinished recording, decoding phrase ...\n')
                f.flush()
                filename = self.save_speech(list(prev_audio) + audio2send, p)
                r = self.decode_phrase(filename)
                print datetime.utcnow().isoformat() + "\tDETECTED: ", r
                f.write(datetime.utcnow().isoformat() + '\tDETECTED: ' + ', '.join(r) + ' ...\n')
                f.flush()

                for control in r:
                    if control == 'forward':
                        print datetime.utcnow().isoformat() + '\tiQuad moving forward ...\n'
                        ser.write("1001".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad moving forward ...\n\n')
                        f.flush()
                        break
                    if control == 'backward':
                        print datetime.utcnow().isoformat() + '\tiQuad moving backward ...\n'
                        ser.write("1002".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad moving backward ...\n\n')
                        f.flush()
                        break
                    if control == 'right':
                        print datetime.utcnow().isoformat() + '\tiQuad moving right ...\n'
                        ser.write("1003".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad moving right ...\n\n')
                        f.flush()
                        break
                    if control == 'left':
                        print datetime.utcnow().isoformat() + '\tiQuad moving left ...\n'
                        ser.write("1004".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad moving left ...\n\n')
                        f.flush()
                        break
                    if control == 'up':
                        print datetime.utcnow().isoformat() + '\tiQuad climbing upward ...\n'
                        ser.write("1005".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad climbing upward ...\n\n')
                        f.flush()
                        break
                    if control == 'down':
                        print datetime.utcnow().isoformat() + '\tiQuad climbing downward ...\n'
                        ser.write("1006".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad climbing downward ...\n\n')
                        f.flush()
                        break
                    if control == 'starboard':
                        print datetime.utcnow().isoformat() + '\tiQuad panning starboard ...\n'
                        ser.write("1007".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad panning starboard ...\n\n')
                        f.flush()
                        break
                    if control == 'larboard':
                        print datetime.utcnow().isoformat() + '\tiQuad panning larboard ...\n'
                        ser.write("1008".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad panning larboard ...\n\n')
                        f.flush()
                        break
                    if control == 'stop':
                        print datetime.utcnow().isoformat() + '\tiQuad stopped moving ...\n'
                        ser.write("1000".encode('ascii')+'\n')
                        f.write(datetime.utcnow().isoformat() + '\tiQuad stopped moving ...\n\n')
                        f.flush()
                        break
                    if control == '</s>':                                   # end brace detected
                        print datetime.utcnow().isoformat() + '\tNothing useful detected ...\n'
                        ser.write("0000".encode('ascii')+'\n')  # send error code '0000'
                        f.write(datetime.utcnow().isoformat() + '\tNothing useful detected ...\n\n')
                        f.flush()
                        break

                # Removes temp audio file
                os.remove(filename)
                # Reset all
                started = False
                slid_win = deque(maxlen=self.SILENCE_LIMIT * rel)
                prev_audio = deque(maxlen=0.5 * rel)
                audio2send = []
                print datetime.utcnow().isoformat() + "\tListening ..."
                f.write(datetime.utcnow().isoformat() + '\tListening ...\n')
                f.flush()

            else:
                prev_audio.append(cur_data)

        print "* Done listening"
        stream.close()
        p.terminate()
```

Figure 6.11 is a screenshot of the of Odroid XU4 Terminal window showing text output feedback to the user of the speech recognition and the expected UAV 'iQuad' control response. Listing 6.6 is an example of a text file output log data saved for each speech recognition program session. This output log also includes the intialisation data at the beginning of program execution. A new data log file is created for each program session.


odroid@odroid: ~/odroid_proj/speech_to_quad
File Edit View Search Terminal Help

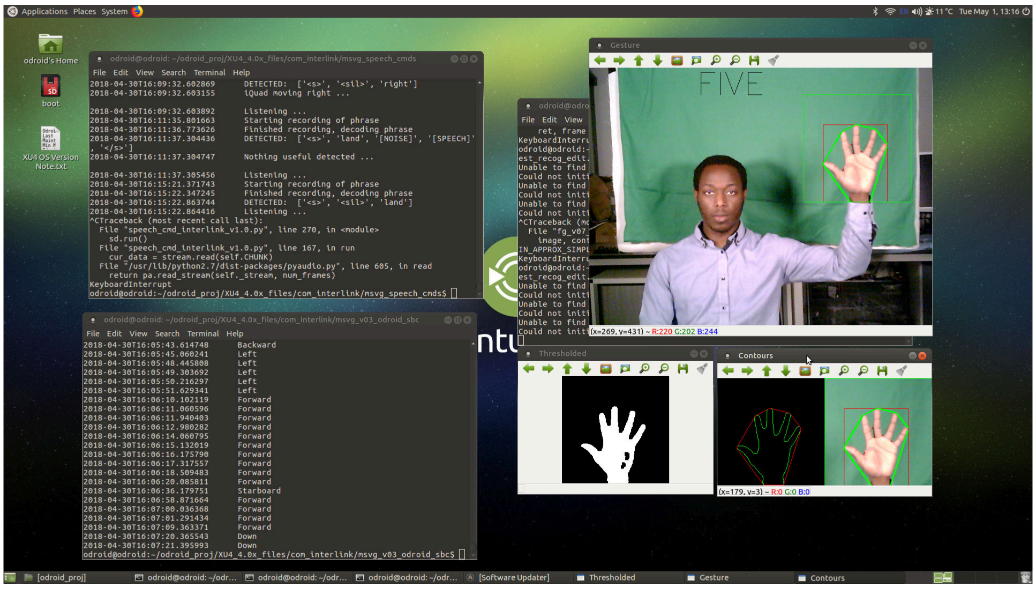
```
2017-02-25T16:57:13.208648      Starting recording of phrase
2017-02-25T16:57:14.247460      Finished recording, decoding phrase
2017-02-25T16:57:14.569600      DETECTED:  ['<s>', 'left', '</s>']
2017-02-25T16:57:14.569934      iQuad moving left ...

2017-02-25T16:57:14.570528      Listening ...
2017-02-25T16:57:16.029946      Starting recording of phrase
2017-02-25T16:57:17.348728      Finished recording, decoding phrase
2017-02-25T16:57:17.708706      DETECTED:  ['<s>', 'backward', '</s>']
2017-02-25T16:57:17.709045      iQuad moving backward ...

2017-02-25T16:57:17.709611      Listening ...
2017-02-25T16:57:24.186171      Starting recording of phrase
2017-02-25T16:57:25.294896      Finished recording, decoding phrase
2017-02-25T16:57:25.628227      DETECTED:  ['<s>', 'forward', '</s>']
2017-02-25T16:57:25.628555      iQuad moving forward ...

2017-02-25T16:57:25.629205      Listening ...
2017-02-25T16:57:34.906004      Starting recording of phrase
2017-02-25T16:57:36.025712      Finished recording, decoding phrase
2017-02-25T16:57:36.295778      DETECTED:  ['<s>', 'stop', '<sil>', '</s>']
2017-02-25T16:57:36.296103      iQuad stopped moving ...

2017-02-25T16:57:36.296734      Listening ...
```

Figure 6.11: Speech recognition feedback on Odroid XU4 Terminal window.

Listing 6.6: Speech control processing - data logging to text file.

```
2018-04-29T21:39:24.669166
/dev/ttyACM0 is open...
press "ctrl + c" to Quit

* Mic set up and listening.

2018-04-29T21:39:39.897660      Starting recording of phrase ...
2018-04-29T21:39:41.210896      Finished recording, decoding phrase ...
2018-04-29T21:39:41.627214      DETECTED: <s>, <sil>, go, forward, <sil>, </s> ...
2018-04-29T21:39:41.628221      iQuad moving forward ...

2018-04-29T21:39:41.628716      Listening ...
2018-04-29T21:39:43.698897      Starting recording of phrase ...
2018-04-29T21:39:45.021239      Finished recording, decoding phrase ...
2018-04-29T21:39:45.405294      DETECTED: <s>, <sil>, go, backward, </s> ...
2018-04-29T21:39:45.405775      iQuad moving backward ...

2018-04-29T21:39:45.406271      Listening ...
2018-04-29T21:39:47.138244      Starting recording of phrase ...
2018-04-29T21:39:48.803298      Finished recording, decoding phrase ...
2018-04-29T21:39:49.201176      DETECTED: <s>, <sil>, go, <sil>, right, </s> ...
2018-04-29T21:39:49.201628      iQuad moving right ...

2018-04-29T21:39:49.202212      Listening ...
2018-04-29T21:39:50.997934      Starting recording of phrase ...
2018-04-29T21:39:52.310841      Finished recording, decoding phrase ...
2018-04-29T21:39:52.671181      DETECTED: <s>, climb, up, <sil>, </s> ...
2018-04-29T21:39:52.671736      iQuad climbing upward ...
```

6.3.2 Gesture capture and recognition



```
odroid@odroid: ~/odroid_proj/XU4_4.0x_files/com_interlink/msvg_speech_cmds
2018-04-30T16:09:32.602869      DETECTED:  ['<s>', '<sil>', 'right']
2018-04-30T16:09:32.603155      iQuad moving right ...

2018-04-30T16:09:32.603892      Listening ...
2018-04-30T16:11:35.801663      Starting recording of phrase
2018-04-30T16:11:36.773626      Finished recording, decoding phrase
2018-04-30T16:11:37.304436      DETECTED:  ['<s>', 'land', '[NOISE]', '[SPEECH]'
, '</s>']
2018-04-30T16:11:37.304747      Nothing useful detected ...

2018-04-30T16:11:37.305456      Listening ...
2018-04-30T16:15:21.371743      Starting recording of phrase
2018-04-30T16:15:22.347245      Finished recording, decoding phrase
2018-04-30T16:15:22.863744      DETECTED:  ['<s>', '<sil>', 'land']
2018-04-30T16:15:22.864416      Listening ...
^CTraceback (most recent call last):
  File "speech_cmd_interlink_v1.0.py", line 270, in <module>
    sd.run()
  File "speech_cmd_interlink_v1.0.py", line 167, in run
    cur_data = stream.read(self.CHUNK)
  File "/usr/lib/python2.7/dist-packages/pyaudio.py", line 605, in read
    return pa.read_stream(self._stream, num_frames)
KeyboardInterrupt
odroid@odroid:~/odroid_proj/XU4_4.0x_files/com_interlink/msvg_speech_cmds$
```

```
odroid@odroid: ~/odroid_proj/XU4_4.0x_files/com_interlink/msvg_v03_odroid_sbc
2018-04-30T16:05:43.614748      Backward
2018-04-30T16:05:45.060241      Left
2018-04-30T16:05:48.445808      Left
2018-04-30T16:05:49.303692      Left
2018-04-30T16:05:50.216297      Left
2018-04-30T16:05:51.629341      Left
2018-04-30T16:06:10.102119      Forward
2018-04-30T16:06:11.060596      Forward
2018-04-30T16:06:11.940403      Forward
2018-04-30T16:06:12.980282      Forward
2018-04-30T16:06:14.060795      Forward
2018-04-30T16:06:15.132019      Forward
2018-04-30T16:06:16.175790      Forward
2018-04-30T16:06:17.317557      Forward
2018-04-30T16:06:18.509483      Forward
2018-04-30T16:06:20.085811      Forward
2018-04-30T16:06:36.179751      Starboard
2018-04-30T16:06:58.871664      Forward
2018-04-30T16:07:00.036368      Forward
2018-04-30T16:07:01.291434      Forward
2018-04-30T16:07:09.363371      Forward
2018-04-30T16:07:20.365543      Down
2018-04-30T16:07:21.395993      Down
odroid@odroid:~/odroid_proj/XU4_4.0x_files/com_interlink/msvg_v03_odroid_sbc$
```

```
odroid@odro
    ret, frame
KeyboardInterru
odroid@odroid:~
est_recog_edit.
Unable to find
Could not initi
Unable to find
Could not initi
Unable to find
Could not initi
^CTraceback (mo
  File "fg_v07_
    image, cont
IN_APPROX_SIMPL
KeyboardInterru
odroid@odroid:~
est_recog_edit.
Unable to find
Could not initi
Unable to find
Could not initi
Unable to find
Could not initi
```

Figure 6.12: Speech recognition feedback on Odroid XU4 Terminal window.

Table 6.5 describes the gesture capture and symbol processing algorithm. Gesture was captured with the aid of an Odroid 720p USB web camera connected to the Odroid XU4 board. The

Table 6.5: Gesture capture and processing to control symbol algorithm.

```
A.  Program initialisation:
        1. import relevant OpenCV and python libraries
        2. start serial port transmission and gesture input data log file
        3. define global variables
        4. Start web camera
B.  Capture and pre-process image:
        1. capture frame
        2. blur image
        3. grey image
        4. filter image background noise
C.  Processing image:
        1. find contours
        2. find convex hull
        3. find convexity defects
D.  Gesture recognition and encoding:
        1. count number of defects, $n_{defects}$
        2. number of fingers, $n_{fingers} = n_{defects} + 1$
        3. map number of fingers to control command symbol
        4. remove high frequency recognition noise by tracking 20 successive cycles
           for 100% accuracy
E.  Relay - pass control command symbols to MCPU for further processing and
    command execution
F.  Repeat - steps B to F loop continuously until the program is manually ended
    by the user
```

captured gestures were finger-counting gestures as shown in Figure 6.12. The finger gestures consisted of one finger, two fingers, three fingers, four fingers, and five fingers gestures, which were mapped to navigational control of forward, backward, right, left, and stop respectively. The processing of the finger gestures was based on the convex hull and convexity defect algorithms. Figure 6.12 shows a five fingers gesture being successfully recognised using this method.

Listing 6.7: Image capture, image processing, and gesture extraction.

```
    # Capture frames from the camera
    ret, frame = capture.read()

    frame = cv2.flip(frame, 1) # vertical image flip for lateral mirror effect

    # Get hand data from the rectangle sub window
    #cv2.rectangle(frame,(100,100),(300,300),(0,255,0),0)
    #crop_image = frame[100:300, 100:300]
    cv2.rectangle(frame, (400,50), (600,250), (0,255,0),0)
    crop_image = frame[50:250, 400:600]
    drawing = np.zeros(crop_image.shape,np.uint8)

    # Apply Gaussian blur
    blur = cv2.GaussianBlur(crop_image, (3,3), 0)

    # Change color-space from BGR -> HSV
    hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)

    # Create a binary image with where white will be skin colors and rest is black
    mask2 = cv2.inRange(hsv, np.array([2,0,0]), np.array([20,255,255]))

    # Kernel for morphological transformation
    kernel = np.ones((5,5))

    # Apply morphological transformations to filter out the background noise
    dilation = cv2.dilate(mask2, kernel, iterations = 1)
    erosion = cv2.erode(dilation, kernel, iterations = 1)

    # Apply Gaussian Blur and Threshold
    filtered = cv2.GaussianBlur(erosion, (3,3), 0)
    ret,thresh = cv2.threshold(filtered, 127, 255, 0)

    # Show threshold image
    cv2.imshow("Thresholded", thresh)

    # Find contours
    image, contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Listing 6.7 is a code snippet of the image capture and pre-processing, in order to extract or recognise the finger gesture. Listing 6.8 is a code snippet of the image processing using convex hull and convexity defects for the finger gesture recognition, and the encoding of the first two finger gestures for further processing at the MCPU. The complete program code is listed in Appendix C.5.

Listing 6.8: Finger gesture command recognition processing.

```
        # Find contour with maximum area
        contour = max(contours, key = lambda x: cv2.contourArea(x))

        # Create bounding rectangle around the contour
        x,y,w,h = cv2.boundingRect(contour)
        cv2.rectangle(crop_image,(x,y),(x+w,y+h),(0,0,255),0)

        # Find convex hull
        hull = cv2.convexHull(contour)

        # Draw contour
        drawing = np.zeros(crop_image.shape,np.uint8)
        cv2.drawContours(drawing,[contour],-1,(0,255,0),0)        # RED contours
        cv2.drawContours(drawing,[hull],-1,(0,0,255),0)          # GREEN hull

        # Find convexity defects
        hull = cv2.convexHull(contour, returnPoints=False)
        defects = cv2.convexityDefects(contour,hull)

        # Use cosine rule to find angle of the far point from the start and end point i.e. the convex points (the finger
        # tips) for all defects
        count_defects = 0

        for i in range(defects.shape[0]):
            s,e,f,d = defects[i,0]
            start = tuple(contour[s][0])
            end = tuple(contour[e][0])
            far = tuple(contour[f][0])

            a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
            b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
            c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
            angle = (math.acos((b**2 + c**2 - a**2)/(2*b*c))*180)/3.14

            # if angle > 90 draw a circle at the far point
            if angle <= 90:
                count_defects += 1
                cv2.circle(crop_image,far,1,[0,0,255],-1)

            cv2.line(crop_image,start,end,[0,255,0],2)

        # Print number of fingers
        if count_defects == 0:
            cv2.putText(frame,"ONE", (200,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
            n_freq_1f = n_freq_1f + 1
            n_freq_2f = 0
            n_freq_3f = 0
            n_freq_4f = 0
            n_freq_5f = 0

            if n_freq_1f >= 20:     # removing high frequency noise, 20 consecutive cycles = success
                n_freq_1f = 0
                print datetime.utcnow().isoformat() + '\tForward␣...\n'
                ser.write("1001".encode('ascii')+'\n')
                f.write(datetime.utcnow().isoformat() + '\tForward␣...\n\n')
                f.flush()

        elif count_defects == 1:
            cv2.putText(frame,"TWO", (200,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
            n_freq_1f = 0
            n_freq_2f = n_freq_2f + 1
            n_freq_3f = 0
            n_freq_4f = 0
            n_freq_5f = 0

            if n_freq_2f >= 20:     # removing high frequency noise, 20 consecutive cycles = success
                n_freq_2f = 0
                print datetime.utcnow().isoformat() + '\tBackward␣...\n'
                ser.write("1002".encode('ascii')+'\n')
                f.write(datetime.utcnow().isoformat() + '\tBackward␣...\n\n')
                f.flush()

        elif count_defects == 2:
            cv2.putText(frame, "THREE", (200,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
            n_freq_1f = 0
            n_freq_2f = 0
            n_freq_3f = n_freq_3f + 1
            n_freq_4f = 0
            n_freq_5f = 0
```

6.3.3 Multimodal control processing unit

Table 6.6 describes the MATLAB MCPU symbol processing algorithm. After developing the input control blocks 'f' and 'g', the next block to be developed was the 'h' block. The approach was to develop the MCPU block in isolation, and then couple the speech and gesture block later on. For this reason, test command symbols were defined for use in simulating the input command being processed into executable UAV actions.

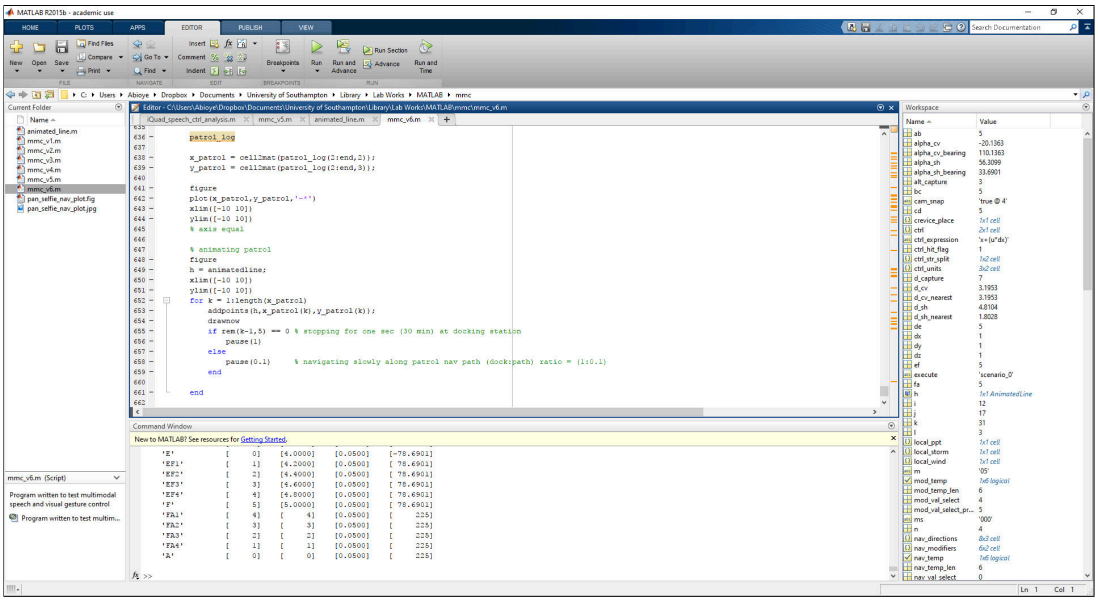
Table 6.6: Multimodal control processing MATLAB operation testing algorithm.

```
A.  Initialise test control symbols/commands
B.  Initialise knowledge base:
        1. nCA Tier 1-III navigation operation - keywords, modifiers, & unit
        2. nCA Tier 2-I scenario commands operation - keyword, execution, & map
C.  Start logger - UAV 3-D position, time, control command, & weather
D.  Loop through test control commands, moving on to the following command
    until the last command has been executed. For each command,
        1. if navigation command - go to nCA Tier 1-III program loop
        2. if scenario command - go to nCA Tier 2-I program loop
```

Listing 6.9: Test command/symbol initialisation.

```
speech_ctrl_time = {'14:00:00.000','14:02:00.000','14:04:00.000','14:05:30.000','14:06:00.000','14:06:30.000','14:07:30.000','14:08:00.000','14:09:00.000','14:12:00.000','14:15:00.000','14:17:00.000'};
speech_ctrl = {'go_forward','go_up_half_metre','go_right_one_metre','hover_at_three_metre','land','hover','go_backward_forward_two_half_metre','go','alert','panoramic_selfie','help', 'go_away'};
```

Listing 6.9 is a code snippet of the control commands used in the developing the MCPU program. The time tag is captured with the control command in order to determine if two contiguous commands were sequential ($t > 0.5s$) or simultaneous ($t \leq 0.5s$). Figure 6.13 shows a screenshot of the MATLAB algorithm during development. Table 6.7 describes the Python MCPU symbol processing algorithm. This was developed from the preceding MATLAB algorithm, optimised as a python program, adapted to work with ROS, interfacing with ROS Gazebo UAV simulator, and receiving control input from the SBC in real time. Listing 6.10 is a code snippet showing the ROS Gazebo initialization routine for the MCPU program. Listing 6.11 is a code snippet showing the ROS Gazebo waypoint publisher subroutine. The full program listing for both the MATLAB and Python MCPU is given in Appendix C.2 and C.3 respectively.


MATLAB R2015b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ Users ▸ Abioye ▸ Dropbox ▸ Documents ▸ University of Southampton ▸ Library ▸ Lab Works ▸ MATLAB ▸ mmc
Current Folder
Name
animated_line.m
mmc_v1.m
mmc_v2.m
mmc_v3.m
mmc_v4.m
mmc_v5.m
mmc_v6.m
pan_selfie_nav_plot.fig
pan_selfie_nav_plot.jpg
mmc_v6.m (Script)
Program written to test multimodal speech and visual gesture control
Program written to test multim...
Editor - C:\Users\Abioye\Dropbox\Documents\University of Southampton\Library\Lab Works\MATLAB\mmc\mmc_v6.m
iQuad_speech_ctrl_analysis.m
mmc_v5.m
animated_line.m
mmc_v6.m
Command Window
New to MATLAB? See resources for Getting Started.
Workspace
Ln 1 Col 1

```
    patrol_log

    x_patrol = cell2mat(patrol_log(2:end,2));
    y_patrol = cell2mat(patrol_log(2:end,3));

    figure
    plot(x_patrol,y_patrol,'-*')
    xlim([-10 10])
    ylim([-10 10])
    % axis equal

    % animating patrol
    figure
    h = animatedline;
    xlim([-10 10])
    ylim([-10 10])
    for k = 1:length(x_patrol)
        addpoints(h,x_patrol(k),y_patrol(k));
        drawnow
        if rem(k-1,5) == 0 % stopping for one sec (30 min) at docking station
            pause(1)
        else
            pause(0.1)     % navigating slowly along patrol nav path (dock:path) ratio = (1:0.1)
        end

    end

```

```
    'E'         [     0]    [4.0000]    [0.0500]    [-78.6901]
    'EF1'       [     1]    [4.2000]    [0.0500]    [ 78.6901]
    'EF2'       [     2]    [4.4000]    [0.0500]    [ 78.6901]
    'EF3'       [     3]    [4.6000]    [0.0500]    [ 78.6901]
    'EF4'       [     4]    [4.8000]    [0.0500]    [ 78.6901]
    'F'         [     5]    [5.0000]    [0.0500]    [ 78.6901]
    'FA1'       [     4]    [     4]    [0.0500]    [      225]
    'FA2'       [     3]    [     3]    [0.0500]    [      225]
    'FA3'       [     2]    [     2]    [0.0500]    [      225]
    'FA4'       [     1]    [     1]    [0.0500]    [      225]
    'A'         [     0]    [     0]    [0.0500]    [      225]

fx >>
```

Name	Value
ab	5
alpha_cv	-20.1363
alpha_cv_bearing	110.1363
alpha_sh	56.3099
alpha_sh_bearing	33.6901
alt_capture	3
bc	5
cam_snap	'true @ 4'
cd	5
crevice_place	1x1 cell
ctrl	2x1 cell
ctrl_expression	'x+(u*dx)'
ctrl_hit_flag	1
ctrl_str_split	1x2 cell
ctrl_units	3x2 cell
d_capture	7
d_cv	3.1953
d_cv_nearest	3.1953
d_sh	4.8104
d_sh_nearest	1.8028
de	5
dx	1
dy	1
dz	1
ef	5
execute	'scenario_0'
fa	5
h	1x1 AnimatedLine
i	12
j	17
k	31
l	3
local_ppt	1x1 cell
local_storm	1x1 cell
local_wind	1x1 cell
m	'05'
mod_temp	1x6 logical
mod_temp_len	6
mod_val_select	4
mod_val_select_pr...	5
ms	'000'
n	4
nav_directions	8x3 cell
nav_modifiers	6x2 cell
nav_temp	1x6 logical
nav_temp_len	6
nav_val_select	0

Figure 6.13: MATLAB Screenshot developing MMC_v6 Algorigthm.

Table 6.7: MCPU Python processing algorithm.

A. Program initialisation:

1. import relevant python libraries
2. define global variables
3. initialise ROS waypoint navigation components

B. Repeat the MATLAB numerical simulation using Python, ROS, and Gazebo for graphical simulation.

1. define control commands to be executed sequentially
2. start UAV data log and printing information in Linux terminal window
3. begin navigation and scenario control command execution
4. continuously publish UAV data log information in terminal
5. continuously send updated position/navigation information to the firefly UAV in the ROS Gazebo simulator
6. end with patrol $5 \times 5\ km^2$ loop operation

Listing 6.10: ROS waypoint navigation initialisation.

```
#*****************************************************************************************
# initializing ros waypoint navigation
#*****************************************************************************************
# probably explore "command/gps_waypoing" rostopic alternative for quicker s/w to h/w deployment
way_pt_pub = rospy.Publisher('/firefly/command/trajectory', MultiDOFJointTrajectory, queue_size=10)
rospy.init_node('waypoint_nav', anonymous=True)        # initializing node 'waypoint_nav'
rospy.loginfo("Started_waypoint_nav")

nav_cmd = MultiDOFJointTrajectory()
nav_points = MultiDOFJointTrajectoryPoint()
nav_transform = geometry_msgs.Transform()
nav_twist = geometry_msgs.Twist()

# creating header
nav_cmd.header.seq = idx
nav_cmd.header.stamp.secs = 0    #rospy.get_time()
nav_cmd.header.stamp.nsecs = 0
nav_cmd.header.frame_id = ''     # "nav_ctrl"

# creating joint_names
nav_cmd.joint_names = ['base_link']

# creating point components
nav_transform.translation.x = 0.0
nav_transform.translation.y = 0.0
nav_transform.translation.z = 1.0
nav_transform.rotation.w = 1.0                  # quaternion rotation transform

#nav_twist.linear = [0, 0, 0]   # [x, y, z]     -       vector3 linear
#nav_twist.angular = [0, 0, 0]  # [x, y, z, w] - vector3 angular
#nav_cmd.points = [nav_transform, nav_twist]

nav_points.transforms = [nav_transform]
nav_cmd.points = [nav_points]

# rospy.loginfo(nav_cmd)
# way_pt_pub.publish(nav_cmd)
#*****************************************************************************************
```

Listing 6.11: ROS Gazebo waypoint publisher.

```
def waypoint_nav(nav_x, nav_y, nav_z, nav_theta):

        rate = rospy.Rate(0.2)  # 0.2 hz i.e. 1 messages per 5 seconds
        if not rospy.is_shutdown():
                nav_transform.translation.x = nav_x
                nav_transform.translation.y = nav_y
                nav_transform.translation.z = nav_z
                # nav_transform.rotation.w = 1.0                  # quaternion rotation transform
                # rospy.loginfo(nav_cmd)
                way_pt_pub.publish(nav_cmd)

                rate.sleep()            # sleeps for 5 seconds
```

Keyboard interface

After converting the MATLAB MCPU test program into Python, but before coupling the speech and gesture input stages, there was a need to pass test control symbols in real time to test how the coupled speech and gesture interface would operate. The keyboard was used to generate this input symbol. Table 6.8 lists all the speech, gesture, and keyboard input control commands used for navigation testing, and their common control symbol. Each command,

Table 6.8: Standardised command control symbols for MCPU input.

S/N	Symbols	Speech	Gesture	Keyboard Input
1	1000	Stop	5 fingers	Ctrl + X
2	1001	Forward	1 finger	Ctrl + Up Arrow Key
3	1002	Backward	2 fingers	Ctrl + Down Arrow Key
4	1003	Right	3 fingers	Ctrl + Right Arrow Key
5	1004	Left	4 finger	Ctrl + Left Arrow Key
6	1005	Up	-	Ctrl + W
7	1006	Down	-	Ctrl + S
8	1007	Starboard	-	Ctrl + D
9	1008	Larboard	-	Ctrl + A

regardless of the source (speech, gesture, or keyboard), is converted into the common control symbol before being passed on to the MCPU for further processing.

Table 6.9: Keyboard control processing to symbol algorithm.

```
A.  Program initialisation:
      1. import relevant python libraries
      2. start serial port transmission and keyboard input data log file
      3. start monitoring keyboard keypress
B.  Loop - wait for keyboard keypress event
C.  Capture keyboard keypress preceeded by “ctrl keyboard command” otherwise
    return to B wait loop
D.  Process keyboard input command via a lookup table system:
      1. if command is found in look up table, then copy corresponding control
         symbol to be passed to MCPU
      2. if not, then return to B wait loop
E.  Relay - pass control command symbols to MCPU for further processing and
    command execution
F.  Repeat - steps B to E loop continuously until the program is manually ended
    by the user
```

Listing 6.12: Processing keyboard control symbol.

```
def get():
        inkey = _Getch()
        while(1):
            k=inkey()
            if k!='':break
        x = ord(k)
        if x == 3:
                exit()
        elif x == 59:
                        # for the arrow keys
                        k=inkey()
                        if ord(k) == 53:
                                k=inkey()
                                if ord(k) == 65:        # decimal value of keyboard 'ctrl + up arrow key'
                                        print datetime.utcnow().isoformat() + '\tForward'
                                        ser.write("1001".encode('ascii')+'\n')
                                        f.write(datetime.utcnow().isoformat() + '\tForward\n')
                                        f.flush()
                                elif ord(k) == 66:      # decimal value of keyboard 'ctrl + down arrow key'
                                        print datetime.utcnow().isoformat() + '\tBackward'
                                        ser.write("1002".encode('ascii')+'\n')
                                        f.write(datetime.utcnow().isoformat() + '\tBackward\n')
                                        f.flush()
                                elif ord(k) == 67:      # decimal value of keyboard 'ctrl + right arrow key'
                                        print datetime.utcnow().isoformat() + '\tRight'
                                        ser.write("1003".encode('ascii')+'\n')
                                        f.write(datetime.utcnow().isoformat() + '\tRight\n')
                                        f.flush()
                                elif ord(k) == 68:      # decimal value of keyboard 'ctrl + left arrow key'
                                        print datetime.utcnow().isoformat() + '\tLeft'
                                        ser.write("1004".encode('ascii')+'\n')
                                        f.write(datetime.utcnow().isoformat() + '\tLeft\n')
                                        f.flush()

        # for the non-arrow keys
        elif ord(k) == 23:      # decimal value of keyboard 'ctrl + w'
                print datetime.utcnow().isoformat() + '\tUp'
                ser.write("1005".encode('ascii')+'\n')
                f.write(datetime.utcnow().isoformat() + '\tUp\n')
                f.flush()
        elif ord(k) == 19:      # decimal value of keyboard 'ctrl + s'
                print datetime.utcnow().isoformat() + '\tDown'
                ser.write("1006".encode('ascii')+'\n')
                f.write(datetime.utcnow().isoformat() + '\tDown\n')
                f.flush()
        elif ord(k) == 4:       # decimal value of keyboard 'ctrl + d'
                print datetime.utcnow().isoformat() + '\tStarboard'
                ser.write("1007".encode('ascii')+'\n')
                f.write(datetime.utcnow().isoformat() + '\tStarboard\n')
                f.flush()
        elif ord(k) == 1:       # decimal value of keyboard 'ctrl + a'
                print datetime.utcnow().isoformat() + '\tLarboard'
                ser.write("1008".encode('ascii')+'\n')
                f.write(datetime.utcnow().isoformat() + '\tLarboard\n')
                f.flush()
        elif ord(k) == 24:      # decimal value of keyboard 'ctrl + x'
                print datetime.utcnow().isoformat() + '\tStop'
                ser.write("1000".encode('ascii')+'\n')
                f.write(datetime.utcnow().isoformat() + '\tStop\n')
                f.flush()
```

Table 6.9 describes the keyboard input capture and symbol processing algorithm. Listing 6.12 is code snippet showing the capture and processing of the keyboard input into control symbols, mapped as shown in Table 6.8. The complete program code is listed in Appendix C.6. Although, the keyboard input was originally developed to pass test control symbols directly to the MCPU before the speech and gesture components were coupled to the MCPU, in the later part of this research work, it was used as the altitude, attitude, and position (AAP) assisted RCJ interface input equivalent, for a more like-to-like comparison of the mSVG and RCJ in the later part of this research work.

6.3.4 ROS Gazebo UAV simulator

Input was received from the Odroid XU4 via USB using serial TX/RX transmission at 115200 Baudrate with the aid of the Teensy 3.2 Arduino compatible microcontroller. Listing 6.13 is a code snippet of the desktop-side (client-side) Teensy 3.2 serial port communication receiver.

Listing 6.13: Client Teensy 3.2 serial interface feeding control symbols to Linux desktop.

```
/*  FileName: msvg_v03_linux_desktop.ino

    Author: Ayo Abioye (abioyeayo@gmail.com)
    Date: Tue 24-Apr-2018
    Version: 3.0
    Target Board: Teensy 3.2 (Arduino Uno compatible code)

    Description: A program to receive ctrl cmds from Odroid XU4 SBC for
                 ROS Gazebo Simulator on Linux Desktop Workstation
*/

char incomingByte;

void setup()
{
  Serial.begin(115200);
  Serial1.begin(115200);

  pinMode(13, OUTPUT);      //ON Status Light
  digitalWrite(13, HIGH);   //
  Serial.println("Starting datalogging...");
}

void loop()
{
  //Serial.print(Serial1.read());
  if (Serial1.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial1.read();

    // say what you got:
    Serial.print(incomingByte);
  }
}
```

Two Teensy 3.2 microcontrollers are used, one connected to the Odroid XU4 via USB - the server, and the second connected to the Linux desktop computer workstation via USB - the client. The client microcontroller writes the serial data received from the server microcontroller via ‘serial1’ to the desktop via ‘serial’. A terminal screen runs on the desktop that captures the serial input and feeds this into the ROS framework, with the aid of the RotorS libraries, processes and passes this to the Gazebo simulator to control the firefly UAV. The server microcontroller has an identical program running but with the ‘serial1’ and ‘serial’ inputs swapped. Also the Tx and Rx pin for serial1 are crossed between the server and client microncontroller interfaces, so that the output of the server can be the input of the client and vice versa.

Listing 6.14: ROS node initialisation for control symbol input processing.

```
def waypoint_nav():
        way_pt_pub = rospy.Publisher('/firefly/command/trajectory', MultiDOFJointTrajectory, queue_size
            =10) # explore "command/gps_waypoing" option
        rospy.init_node('waypoint_nav', anonymous=True)         # initializing node 'waypoint_nav'
        rospy.loginfo("Started_waypoint_nav")

        nav_cmd = MultiDOFJointTrajectory()
        nav_points = MultiDOFJointTrajectoryPoint()
        nav_transform = geometry_msgs.Transform()
        nav_twist = geometry_msgs.Twist()

        # creating header
        nav_cmd.header.seq = idx
        nav_cmd.header.stamp.secs = 0    #rospy.get_time()
        nav_cmd.header.stamp.nsecs = 0
        nav_cmd.header.frame_id = ''     # "nav_ctrl"

        # creating joint_names
        nav_cmd.joint_names = ['base_link']

        # creating points
        #nav_cmd.points = [transforms: [translation: {x: 1, y: 1, z: 1}, rotation:{x: 0, y: 0, z: 0, w:
             1}]]
        #nav_cmd.points.transforms = [[1, 1, 1], [0, 0, 0, 0]]

        #nav_transform.translation = [1, 0, 1]  # [x, y, z]      -        vector3 translation
        #nav_transform.rotation = [0, 0, 0, 0]  # [x, y, z, w] - quaternion rotation


        nav_transform.translation.x = 0.0
        nav_transform.translation.y = 0.0
        nav_transform.translation.z = 1.0
        nav_transform.rotation.w = 1.0

        #nav_twist.linear = [0, 0, 0]    # [x, y, z]      -        vector3 linear
        #nav_twist.angular = [0, 0, 0]   # [x, y, z, w] - vector3 angular

        #nav_cmd.points = [nav_transform, nav_twist]
        nav_points.transforms = [nav_transform]

        nav_cmd.points = [nav_points]

        # rospy.loginfo(nav_cmd)
        # way_pt_pub.publish(nav_cmd)

        # defining initial uav position
        nav_x = 0
        nav_y = 0
        nav_z = 0
        nav_theta = 0
```

Listing 6.14 is a code snippet of the MCPU processing ROS node. Listing 6.15 is a code snippet of the MCPU waypoint navigation ROS node processing the input control symbols into waypoint navigation coordinates, to be passed on to the RotorS framework for executing the navigation control command with the firefly UAV in the Gazebo simulator. The complete program code is listed in Appendix C.7.

Listing 6.15: ROS control symbol input processing.

```
        rate = rospy.Rate(10.0) # 10.0 hz i.e. 10 messages per second
        while not rospy.is_shutdown():
                temp_string = ser.readline()
                if temp_string != "":
                        # record time data
                        datastring = datetime.utcnow().isoformat() + '\t' + temp_string

                        nav_theta_temp = nav_theta


                        # extract command
                        if temp_string == "1001\n":             # forward cmd
                                #nav_x = nav_x + 1

                                tan_sign_mod = 1                                        # tangent sign modifier to alter direction
                                if nav_theta_temp > 90:                                 # added to enable computation of
                                        nav_theta_temp = 360 - nav_theta_temp           # 4th quadrant e.g 315 deg or similar
                                        tan_sign_mod = -1
                                nav_theta_rad = nav_theta_temp/57.2958
                                d_pt = 0.5                      # default modifier is 0.5 m because 1 m is to big a jump

                                nav_x_old = nav_x

                                # compute for new x,y coordinate
                                nav_x = nav_x_old + math.sqrt((d_pt**2) / (1 + math.tan(nav_theta_rad)))
                                nav_y = nav_y + ((nav_x - nav_x_old) * math.tan(nav_theta_rad)*tan_sign_mod)


                        if temp_string == "1002\n":             # backward cmd
                                #nav_x = nav_x - 1

                                tan_sign_mod = 1                                        # tangent sign modifier to alter direction
                                if nav_theta_temp > 90:                                 # added to enable computation of
                                        nav_theta_temp = 360 - nav_theta_temp           # 4th quadrant e.g 315 deg or similar
                                        tan_sign_mod = -1
                                nav_theta_rad = nav_theta_temp/57.2958
                                d_pt = 0.5                      # default modifier is 0.5 m

                                nav_x_old = nav_x

                                # compute for new x,y coordinate
                                nav_x = nav_x_old - math.sqrt((d_pt**2) / (1 + math.tan(nav_theta_rad)))
                                nav_y = nav_y - ((nav_x_old - nav_x) * math.tan(nav_theta_rad)*tan_sign_mod)


                        if temp_string == "1003\n":             # right cmd
                                # nav_y = nav_y - 1

                                tan_sign_mod = 1                                        # tangent sign modifier to alter direction
                                if nav_theta_temp > 90:                                 # added to enable computation of
```

```
                    nav_theta_temp = 360 - nav_theta_temp           # 4th quadrant e.g 315 deg or similar
                    tan_sign_mod = -1
                nav_theta_rad = nav_theta_temp/57.2958
                d_pt = 0.5                  # default modifier is 0.5 m

                nav_y_old = nav_y

                # compute for new x,y coordinate
                nav_y = nav_y_old - math.sqrt((d_pt**2) / (1 + math.tan(nav_theta_rad)))
                nav_x = nav_x + ((nav_y_old - nav_y) * math.tan(nav_theta_rad)*tan_sign_mod)


            if temp_string == "1004\n":         # left cmd
                #nav_y = nav_y + 1

                tan_sign_mod = 1                                    # tangent sign modifier to alter direction
                if nav_theta_temp > 90:                             # added to enable computation of
                    nav_theta_temp = 360 - nav_theta_temp           # 4th quadrant e.g 315 deg or similar
                    tan_sign_mod = -1
                nav_theta_rad = nav_theta_temp/57.2958
                d_pt = 0.5                  # default modifier is 0.5 m

                nav_y_old = nav_y

                # compute for new x,y coordinate
                nav_y = nav_y_old + math.sqrt((d_pt**2) / (1 + math.tan(nav_theta_rad)))
                nav_x = nav_x - ((nav_y - nav_y_old) * math.tan(nav_theta_rad)*tan_sign_mod)


            if temp_string == "1005\n":         # up cmd
                nav_z = nav_z + 0.5             # default modifier is 0.5 m

            if temp_string == "1006\n":         # down cmd
                nav_z = nav_z - 0.5             # default modifier is 0.5 m

            if temp_string == "1007\n":         # starboard cmd
                if nav_theta == 50:
                    nav_theta = 0
                else:
                    nav_theta = 310 #nav_theta + 50


            if temp_string == "1008\n":         # larboard cmd
                if nav_theta == 310:
                    nav_theta = 0
                else:
                    nav_theta = 50 #nav_theta + 50


            # Executing navigation control
            nav_transform.translation.x = nav_x
            nav_transform.translation.y = nav_y
            nav_transform.translation.z = nav_z

            nav_theta_rad = nav_theta / 57.2958     # 180 deg / PI rad = 57.2958
            nav_transform.rotation.x = 0 * math.sin(nav_theta_rad/2)        # yaw has no x-axis components
            nav_transform.rotation.y = 0 * math.sin(nav_theta_rad/2)        # yaw has no y-axis components
            nav_transform.rotation.z = 1 * math.sin(nav_theta_rad/2)        # yaw
```

```
                    has only z-axis components
                nav_transform.rotation.w = math.cos(nav_theta_rad/2)          # yaw
                    angle component in quaternion


                #rospy.loginfo(nav_cmd)
                way_pt_pub.publish(nav_cmd)

                # log data
                print datastring,
                # f.write(datastring)
                # f.flush()                                             # Force system
                    write to disk
                #f.close()

            rate.sleep()
```

mSVG simulation startup procedure

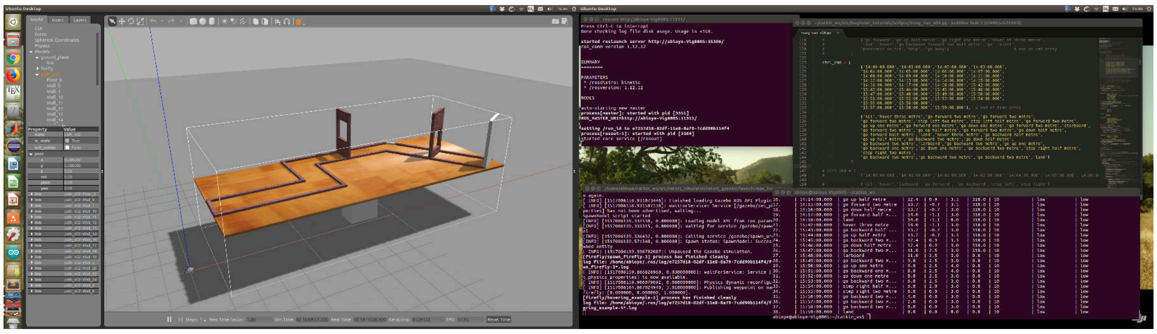


Figure 6.14: Screenshot of the mSVG ROS Gazebo simulator running with the path_v02 loaded.

The procedure for starting up the mSVG simulation is described in Table 6.10. Figure 6.14 shows a successfully started ROS Gazebo simulation with the path_v02 flight path loaded and the firefly UAV awaiting navigation control commands to be executed. Figure 6.15 shows the ROS computation graph “rqt_graph” for the ROS Gazebo UAV simulation. From the rqt_graph, it can be seen that the “waypoint_nav” node (waypoint_nav_18470_1517071665874) developed in this Chapter, publishes the computed coordinate as trajectory messages to the firefly position controller in the RotorS system. This in turn requests the current position of the firefly UAV from the Gazebo simulator, computes the difference, and then updates the firefly UAV’s position in Gazebo.

Table 6.10: ROS Gazebo mSVG simulation startup procedure.

A. Peripheral hardware setup:

1. Teensy 3.2 server and client plugged into the Odroid XU4 SBC and Linux desktop computers respectively and their Tx and Rx pins for 'serial1' crossed
2. Microphone plugged into the Odroid XU4 SBC via USB
3. Odroid web camera plugged into the Odroid XU4 SBC via USB

B. Linux desktop:

1. Startup the Linux desktop computer and boot to the default desktop graphical user interface
2. Open a new Terminal, "*Ctrl + Alt + T*"
3. Open "home/catkin_ws/" folder in the Terminal using,

```
$ cd catkin_ws/
```

4. Load setup.bash, and start ROS,

```
$ source devel/setup.bash
$ roscore
```

5. Start RotorS simulator in basic world with firefly UAV,

```
$ roslaunch rotors_gazebo mav_hovering_example.launch
    mav_name:=firefly world_name:=basic
```

6. Load the path_v02 navigation flight path model, positioning it at $(x, y, z) = (8.0, 0.5, 0.0)$
7. Start mSVG MCPU program to retrieve control command symbol from SBC via the serial port; to compute new waypoint coordinates, direction, and velocities; and to pass this to the RotorS simulator for execution on the firefly UAV,

```
$ rosrun beginner_tutorials msvg_v4.1_linux_desktop.py
```

8. Start rqt graph,

```
$ rqt_graph
```

C. Odroid XU4 single board computer:

1. Startup the Odroid XU4 SBC computer and boot to the default desktop graphical user interface
2. Open Terminal, "*Ctrl + Alt + T*"
3. Start speech capture, process to standardise control symbol, and send to ROS Gazebo UAV simulator on desktop computer via serial port,

```
$ cd ~/odroid_proj/XU4_4.0x_files/com_interlink/msvg_
    speech_cmds
$ python speech_cmd_interlink_v1.0.py
```

4. Open a new Terminal, "*Ctrl + Alt + T*"
5. Start finger gesture capture, process to standardise control symbol, and send to ROS Gazebo UAV simulator on desktop computer via serial port,

```
$ cd ~/odroid_proj/XU4_4.0x_files/com_interlink/msvg_
    finger_gestures
$ python fg_interlink_v3.0.py
```

6. Open a new Terminal, "*Ctrl + Alt + T*"
7. Start keyboard input capture, process to standardise control symbol, and send to ROS Gazebo UAV simulator on desktop computer via serial port,

```
$ cd ~/odroid_proj/XU4_4.0x_files/com_interlink/msvg_v
    03_odroid_sbc
$ python msvg_v03.1_odroid_sbc.py
```

ROS Gazebo mSVG simulation startup procedure - END

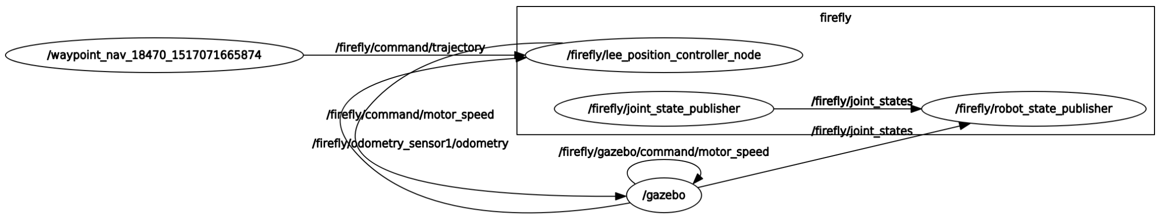

firefly
/waypoint_nav_18470_1517071665874
/firefly/command/trajectory
/firefly/lee_position_controller_node
/firefly/joint_state_publisher
/firefly/joint_states
/firefly/robot_state_publisher
/firefly/joint_states
/firefly/command/motor_speed
/firefly/odometry_sensor1/odometry
/firefly/gazebo/command/motor_speed
/gazebo

Figure 6.15: ROS rqt_graph for the ROS Gazebo UAV simulation.

6.4 Chapter Conclusion

In this chapter, the multimodal speech and visual gesture system model was designed and developed with the aid of nCA Tier 1-III navigation and nCA Tier 2-I scenario application examples. In developing the model, mathematical notations and logic were used. These were then converted into computer programs, from which some useful code snippets were presented. The mSVG development was discussed in functional blocks of speech input unit, gesture input unit, multimodal control processing unit, and ROS Gazebo simulator unit. The operations of the multmodal control processing unit were numerically simulated using MATLAB and then graphically simulated using Python with the aid of ROS and Gazebo. The MATLAB numerical simulation enabled one to completely focus on the development of the core MCPU computation for each scenario without worrying about the demonstration on a UAV simulator or hardware platform. The ROS Gazebo simulator was based on the RotorS framework developed by Furrer et al. (2016). The startup procedure for the mSVG system was clearly outlined, starting with putting the peripheral hardware together, to setting up the ROS Gazebo system on the desktop, and setting up the speech and gesture capture on the SBC. The designed and developed mSVG system was used in performing this research's investigation in the following chapters.

Chapter 7

Effects of varying noise levels and lighting levels on mSVG interaction with aerobots

In this chapter, the result of the experiment study on the effects of varying noise levels and varying lighting levels on speech and gesture control command interfaces for aerobots, is presented, analysed, and discussed. The result consists of around 3,108 speech utterance and 999 gesture quality observations. The result of the varying noise level is presented and analysed in two parts, firstly as the speech command phrase, and secondly as the speech word component. Then the results of the varying lighting level experiment is presented and analysed. Finally, the results are discussed.

As discussed in Section 4.5, the experiment study was designed in Section 4.5.1, 37 participants were recruited in Section 4.5.4, who performed the experiment procedure as described in Section 4.5.3. The 12 speech command phrases used for the speech experiment were *"go forward, go backward, step left, step right, hover, land, go forward half metre, go backward one metre, hover one metre, step left half metre, step right one metre, and stop"*. The following five finger gestures – *'one finger, two fingers, three fingers, four fingers, and five fingers'*, were used in the gesture lighting experiment.

7.1 Varying Noise Level - Speech Command Results

Tables showing the results of the experiment are presented in the figures in Appendix B.2.1. The dataset is included in this thesis CD-R, as part of the result spreadsheet. The blanks indicated by an underscore are points were the data was not available due to lab threshold noise levels being higher than specified, caused by uncontrollable ambient noise conditions during experiments. Also, the whole of participant A1's data in this segment were corrupt due to setup failure at the beginning of the experiments. The implication of this is that total sum of utterances may be fewer than 37, mostly 36 at each noise dB level.

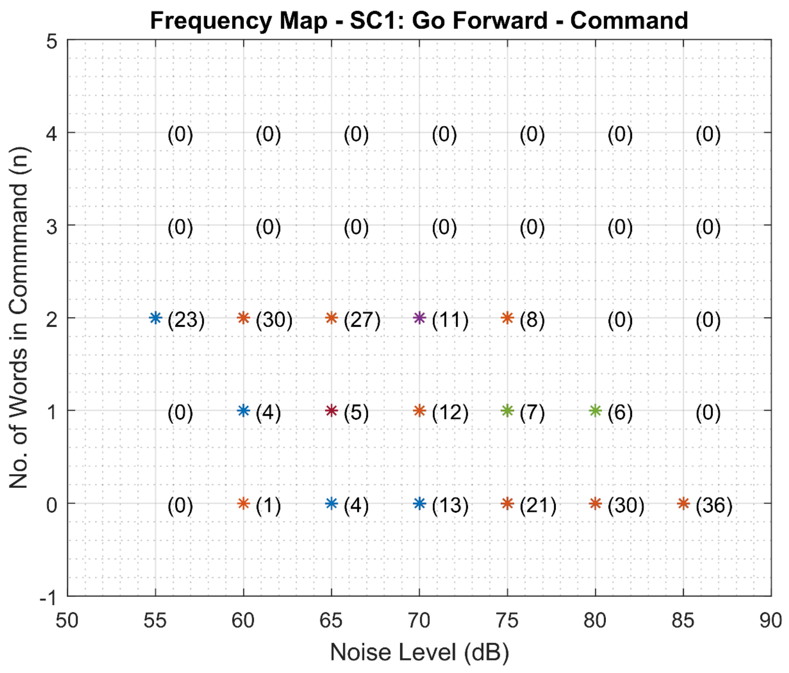


Figure 7.1: SC1 Frequency Map.

Each of the 12 speech commands were collated across the 37 participants, and the number of words successfully recognised were plotted against the noise levels with the number of times the number of words were recognised for the particular speech command by different participants, that is the hit frequency of each point on the plot was being indicated in brackets. This was called the frequency map. A MATLAB program was written to collate and plot the data from the result table. From the frequency map shown in Figure 7.1, it can be observed for the two-word speech command "Go Forward" that all the 23 utterances successfully made at 55 dB were successfully recognised as two-word commands, as indicated in brackets next to the point on the frequency map plot. Also, at 60 dB, it can be observed that of the 35 successfully registered commands, 30 were two-word (full recognition - success), 4 were one-word (partial recognition - partial success), and 1 was no-word (recognition failure). The distribution of the partial success is presented in the next section on "variable noise level - word frequency." At 65 dB, 36 commands were successfully registered/uttered/recorded, but only 27 were two-worded (success), 5 one-worded (partial success), and 4 failures (no-word). At 70 dB, 11 were successfully recognised as two-worded, 12 partial success (one-word), and 13 failures (no-word). At 75 dB, 36 commands were registered, 8 two-word successes, 7 one-word partial successes, and 21 failures (no word recognised). At 80 dB, 36 commands were registered, zero two-word success, 6 one-word successes, and 30 failures (no-word). At 85 dB, 36 commands were registered, but all 36 failured, that is no word was recognised at 85 dB by the custom UAV-Speech interface.

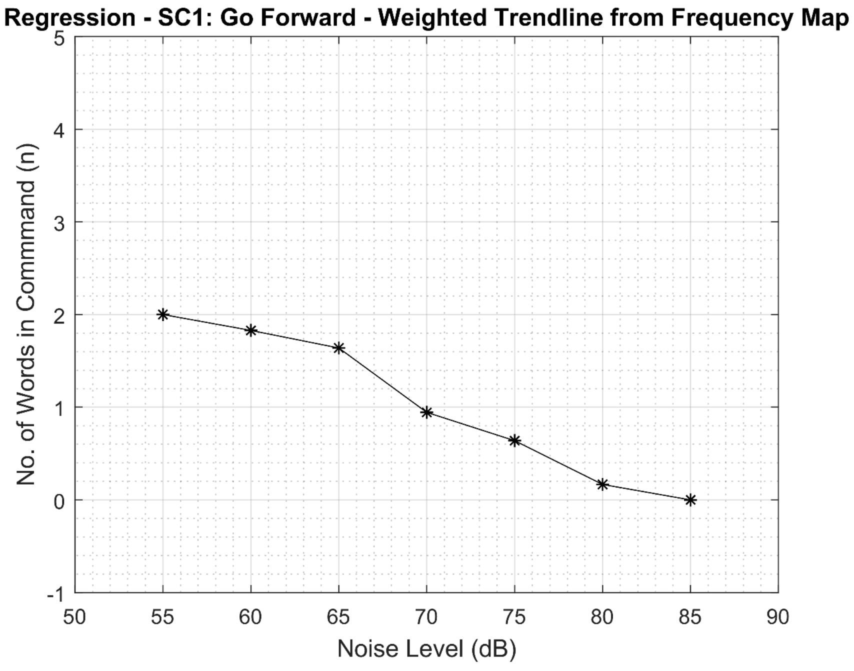


Figure 7.2: SC1 Trendline.

A trend can be observed here, that at lower dB noise level, the two-word speech command “Go Forward” was successfully recognised, whereas recognition fails at higher dB noise levels. This trend is graphically shown by the trendline plot in Figure 7.2. The points on the trendline were computed by taking the vertical weighted average from the frequency map as follows:

$$y(x) = \frac{\sum_{i=0}^{n} d_i(x) f_i(x)}{\sum_{i=0}^{n} f_i(x)} \tag{7.1}$$

Where $n = 4$ - is the maximum number of speech command words used in the experiment. $d_i(x)$ - is the specific number of speech command words being registered, for the given x dB noise level. Note that this corresponds to the i^{th} value. $f_i(x)$ - is the frequency of the $d_i(x)$ point, as indicated on the frequency map, for the given x dB noise level.

For example, the points on the trendline for the first speech command (SC1), “Go Forward” was computed as follows:

When $x = 55$ dB

$$y(55) = \frac{\sum_{i=0}^{4} d_i(55) f_i(55)}{\sum_{i=0}^{n} f_i(55)} \tag{7.2}$$

$$= \frac{d_0(55)f_0(55) + d_1(55)f_1(55) + d_2(55)f_2(55) + d_3(55)f_3(55) + d_4(55)f_4(55)}{f_0(55) + f_1(55) + f_2(55) + f_3(55) + f_4(55)} \tag{7.3}$$

$$= \frac{0 \cdot 0 + 1 \cdot 0 + 2 \cdot 23 + 3 \cdot 0 + 4 \cdot 0}{0 + 0 + 23 + 0 + 0} = 2 \tag{7.4}$$

When $x = 60$ dB

$$y(60) = \frac{\sum_{i=0}^{4} d_i(60) f_i(60)}{\sum_{i=0}^{n} f_i(60)} \tag{7.5}$$

$$= \frac{d_0(60)f_0(60) + d_1(60)f_1(60) + d_2(60)f_2(60) + d_3(60)f_3(60) + d_4(60)f_4(60)}{f_0(60) + f_1(60) + f_2(60) + f_3(60) + f_4(60)} \tag{7.6}$$

$$= \frac{0 \cdot 1 + 1 \cdot 4 + 2 \cdot 30 + 3 \cdot 0 + 4 \cdot 0}{1 + 4 + 30 + 0 + 0} = 1.8286 \tag{7.7}$$

When $x = 65$ dB

$$y(65) = \frac{\sum_{i=0}^{4} d_i(65) f_i(65)}{\sum_{i=0}^{n} f_i(65)} = \frac{0 \cdot 4 + 1 \cdot 5 + 2 \cdot 27 + 3 \cdot 0 + 4 \cdot 0}{4 + 5 + 27 + 0 + 0} = 1.6389 \tag{7.8}$$

When $x = 70$ dB

$$y(70) = \frac{\sum_{i=0}^{4} d_i(70) f_i(70)}{\sum_{i=0}^{n} f_i(70)} = \frac{0 \cdot 13 + 1 \cdot 12 + 2 \cdot 11 + 3 \cdot 0 + 4 \cdot 0}{13 + 12 + 11 + 0 + 0} = 0.9444 \tag{7.9}$$

When $x = 75$ dB

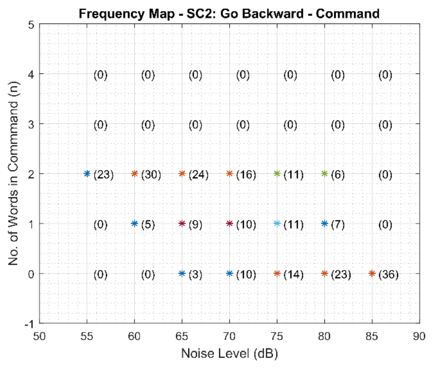
$$y(75) = \frac{\sum_{i=0}^{4} d_i(75) f_i(75)}{\sum_{i=0}^{n} f_i(75)} = \frac{0 \cdot 21 + 1 \cdot 7 + 2 \cdot 8 + 3 \cdot 0 + 4 \cdot 0}{21 + 7 + 8 + 0 + 0} = 0.6389 \tag{7.10}$$

When $x = 80$ dB

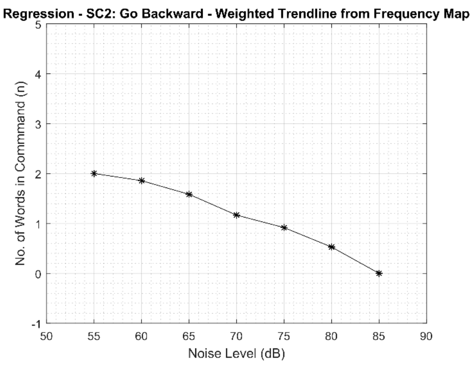
$$y(80) = \frac{\sum_{i=0}^{4} d_i(80) f_i(80)}{\sum_{i=0}^{n} f_i(80)} = \frac{0 \cdot 30 + 1 \cdot 6 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 0}{30 + 6 + 0 + 0 + 0} = 0.1667 \tag{7.11}$$

When $x = 85$ dB

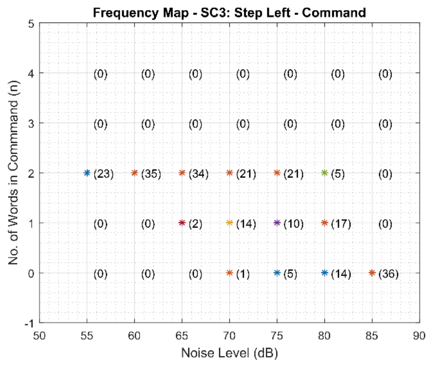
$$y(85) = \frac{\sum_{i=0}^{4} d_i(85) f_i(85)}{\sum_{i=0}^{n} f_i(85)} = \frac{0 \cdot 36 + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 0}{36 + 0 + 0 + 0 + 0} = 0 \tag{7.12}$$



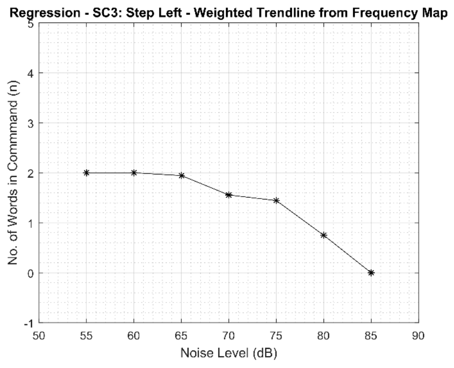
(a) SC2 Frequency Map.



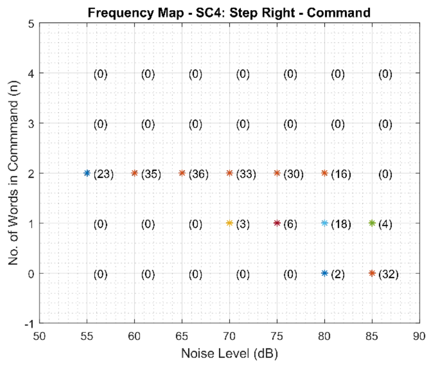
(b) SC2 Trendline.



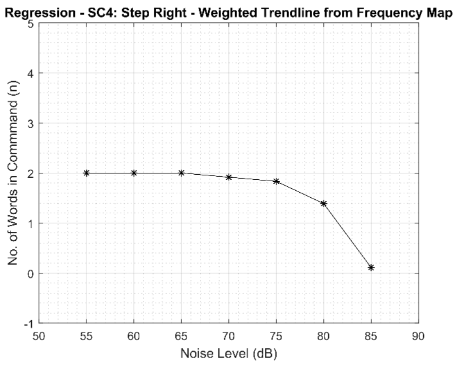
(c) SC3 Frequency Map.



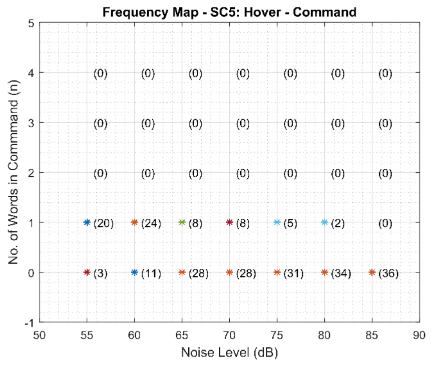
(d) SC3 Trendline.



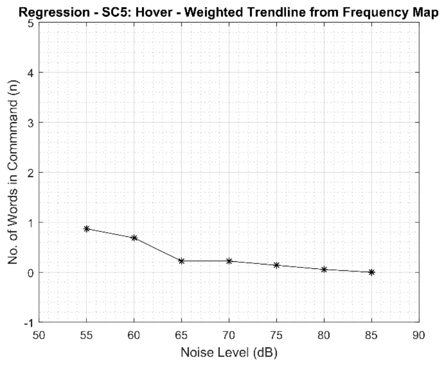
(e) SC4 Frequency Map.



(f) SC4 Trendline.

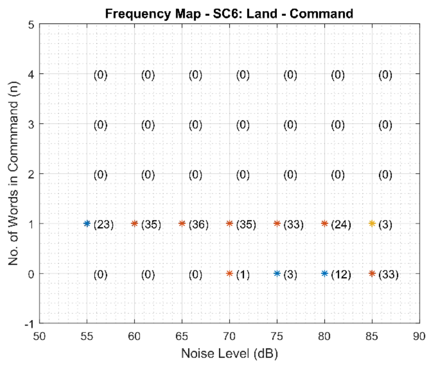


(g) SC5 Frequency Map.

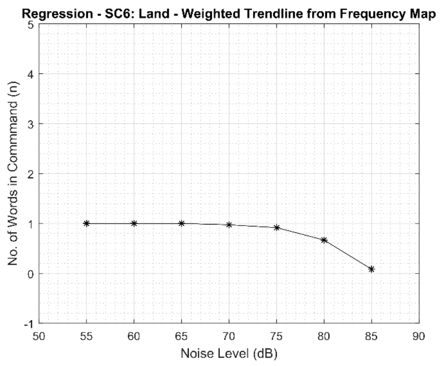


(h) SC5 Trendline.

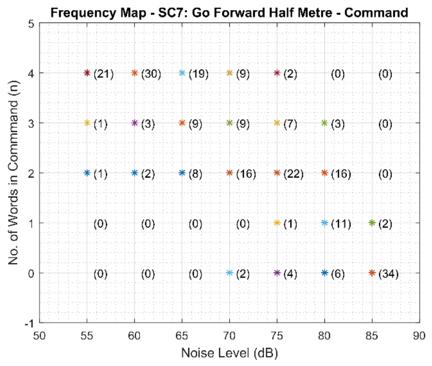
Figure 7.3: Frequency Map and Trendlines I.



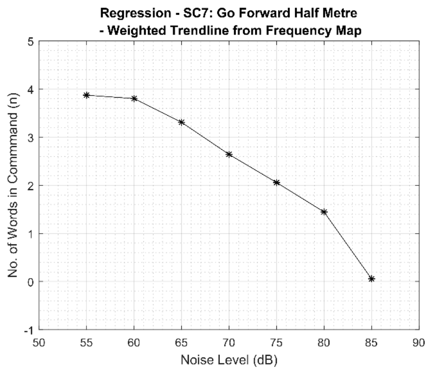
(a) SC6 Frequency Map.



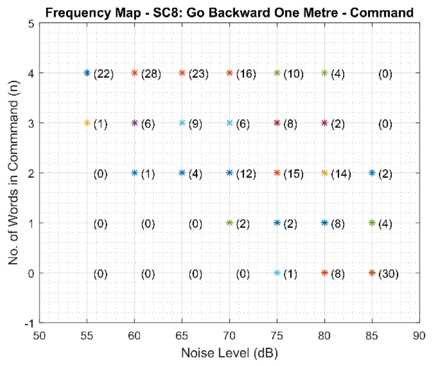
(b) SC6 Trendline.



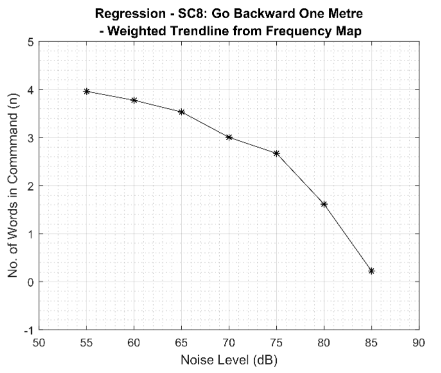
(c) SC7 Frequency Map.



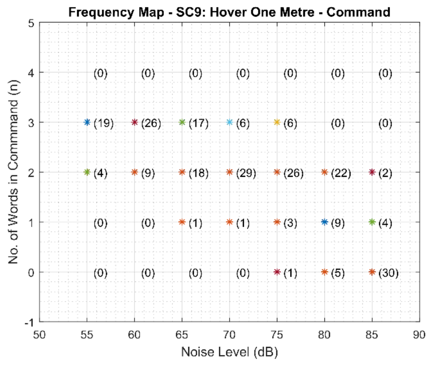
(d) SC7 Trendline.



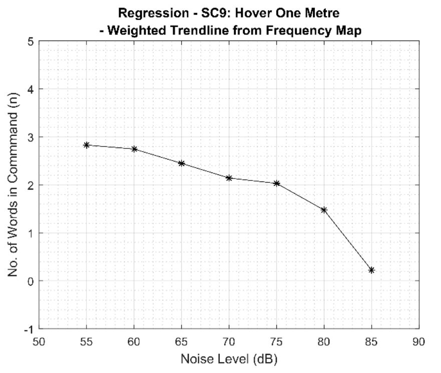
(e) SC8 Frequency Map.



(f) SC8 Trendline.



(g) SC9 Frequency Map.

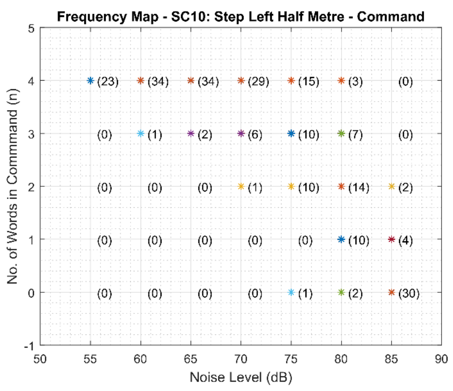


(h) SC9 Trendline.

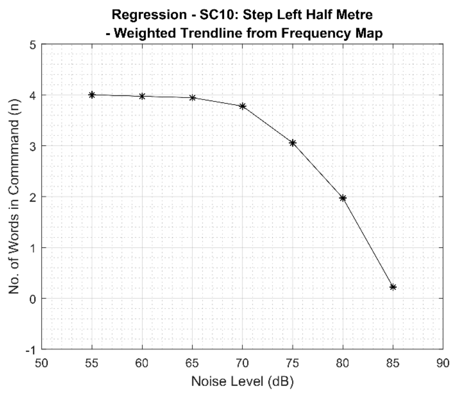
Figure 7.4: Frequency Map and Trendlines II.

Similarly, the frequency map and trendline of the other 11 speech command phrases are presented in Figure 7.3, Figure 7.4, and Figure 7.5, after performing a similar analysis. Figure 7.3 shows the results of four speech commands. In particular, Figure 7.3a and Figure 7.3b show the frequency map and trendline of the second speech command (SC2), “Go Backward”. This is a two-word command which seems to have a slightly better performance across the experiment participants than the first speech command (SC1), “Go Forward”, particularly between 70 dB and 80 dB. Figure 7.3c and Figure 7.3d show the frequency map and trendline of the third speech command (SC3), “Step Left”, another two-word command. Figure 7.3e and Figure 7.3f show the frequency map and trendline of the fourth speech command (SC4), “Step Right”, a two-word command which seems to have been the most successful of all the other two-word commands previously presented. SC4 has a recognition accuracy of over 90% at 75 dB and about 70% at 80 dB. Figure 7.3g and Figure 7.3h show the frequency map and trendline of the fifth speech command (SC5), “Hover”, a one-word command, which had the poorest performance of all the speech command set in the experiment. Its performance was observed to be as low as 22% at 65 dB. This was mainly attributed to its subtle articulation, which leaves it easily prone to noise corruption even at low noise levels. In addition, the SC5 failure was also partly attributed to the speech ASR implementation not being robust enough. Commercial or industrial speech ASR interfaces, such as the Amazon Echo, Apple Siri, and Microsoft Cortana, may offer an improved performance due to their use of more advance and online AI learning algorithms, whereas the custom CMU Sphinx ASR which was used in this application was based on offline hidden markov models (HMM).

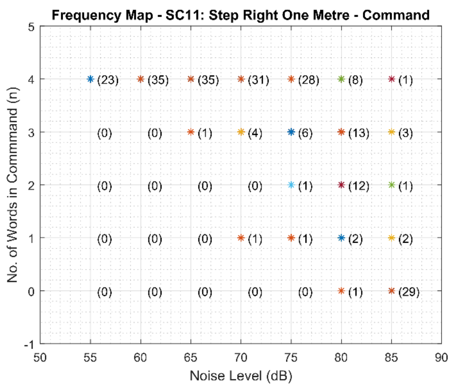
Figure 7.4 shows the results of four more speech commands, SC6 - SC9. Figure 7.4a and Figure 7.4b show the frequency map and trendline of the sixth speech command (SC6), “Land”, which is also a one-word speech command. But unlike SC5: Hover, SC6: Land, had a better performance, with recognition accuracy of over 90% at 75 dB and about 67% at 80 dB. Figure 7.4c and Figure 7.4d show the frequency map and trendline for the seventh speech command (SC7), “Go Forward Half Metre”, which is a four-word command. The result, as presented here, does not give additional information on which of the four words are failing, and whether these are primary keywords, primary modifiers, secondary keywords, or secondary modifiers parameters. Note that the failure of the two primary parameter/words could be considered as the failure of the control command. However a more general approach is used to address this, by investigating the overall word failure frequency in Section 7.2. Figure 7.4e and Figure 7.4f show the frequency map and trendline for the eighth speech command (SC8), “Go Backward One Metre”, another four-word speech command. Figure 7.4g and Figure 7.4h show the frequency map and trendline for the ninth speech command (SC9), “Hover One Metre”, a three-word command.



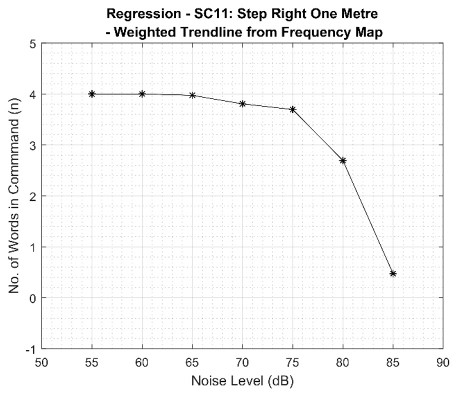
(a) SC10 Frequency Map.



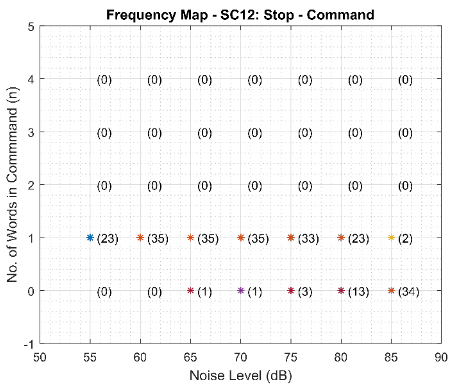
(b) SC10 Trendline.



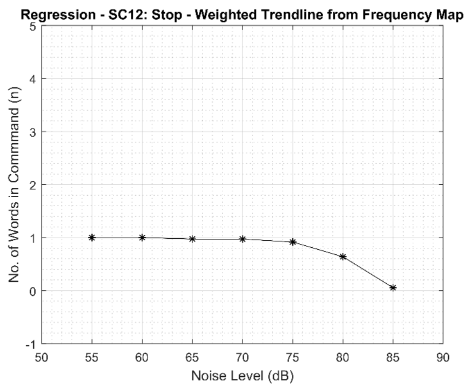
(c) SC11 Frequency Map.



(d) SC11 Trendline.



(e) SC12 Frequency Map.



(f) SC12 Trendline.

Figure 7.5: Frequency Map and Trendlines III.

Figure 7.5 shows the results of the remaining three speech commands, SC10 - SC12. Figure 7.5a and Figure 7.5b show the frequency map and trendline of the tenth speech command (SC10), "Step Left Half Metre", a four-word speech command with over 75% and about 50% recognition accuracy rate at 75 dB and 80 dB respectively. Figure 7.5c and Figure 7.5d show the frequency map and trendline of the eleventh speech command (SC11), “Step Right One Metre”, another four-word speech command. Similar to SC4 “Step Right”, SC11 had a good performance with a recognition accuracy of 92% and 67% at 75 dB and 80 dB respectively. Both SC4 and SC11 share the same base keywords of “Step Right” the recognition of which seems to be highly successful in all cases. This would be investigated further when breaking down the speech command constituents, in Section 7.2, for speech command keyword selection. Figure 7.5e and Figure 7.5f show the frequency map and trendline of the tenth speech command (SC12), “Stop”, a one-word speech command with 91% and 64% recognition accuracy rate at 75 dB and 80 dB respectively.

7.1.1 Speech Command Performance Comparison

In order to compare the performance of each of the 12 speech commands using their trendline characteristic, each weighted trendline was nomalize so they can all be plotted on to the same y-axis, overlaid on the same graph and visually compared. This was done by dividing the weighted trendline $y(x)$ values previously computed by the number of words in the speech command. Mathematically, normalised

$$Y_N(x) = \frac{Y(x)}{n} = \frac{1}{n}\begin{bmatrix} y(55) & y(60) & y(65) & y(70) & y(75) & y(80) & y(85) \end{bmatrix} \tag{7.13}$$

where n is number of words in the specific speech command being normalised. The resulting comparison plot is as shown in Figure 7.6. Note that the poorest performance was observed in the speech recognition of SC5, ‘Hover’, followed by SC1 ‘Go Forward’ and SC2 ‘Go Backward’ as indicated in the combined trendline in Figure 7.6. The best speech recognition performance was observed in SC4 ‘Step Right’, SC6 ‘Land’, SC11 ‘Step Right One Metre’, and SC12 ‘Stop’. Both single-word and multi-word speech commands were found in each performance category. Also, the fact that significant fail safe commands such as SC6 ‘land’ and SC12 ‘stop’, were among the most resilient to noise corruption, having a high recognition success rate at higher noise level, is very important in UAV applications, where fail safe commands are expected to be very reliable, otherwise it may be impractical to use.

7.1.2 Experiment speech command (SC) ASR characteristic

In order to determine the characteristic curve describing the speech command performance within a UAV type application environment, given the custom CMU Sphinx ASR implementation, and the set of speech control commands, an average of the normalised trendline plotted in

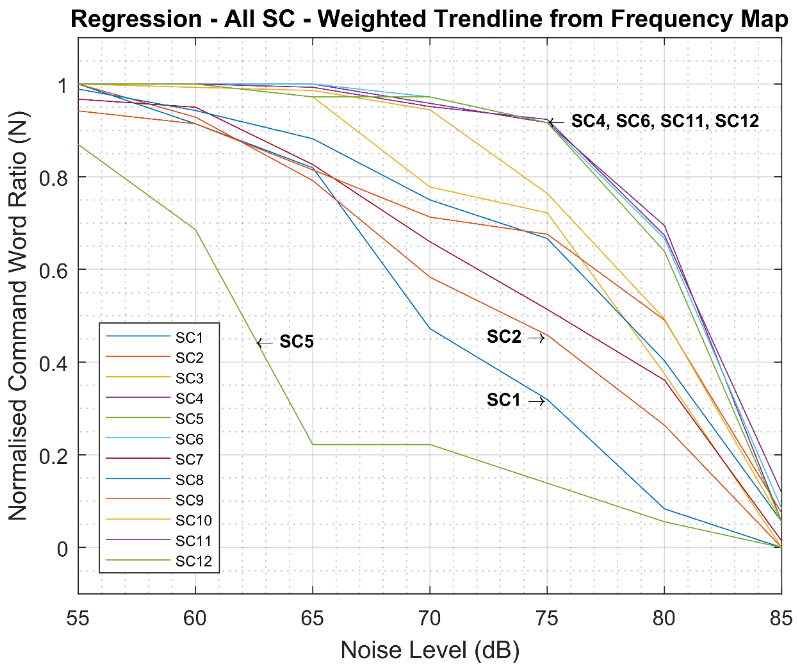


Figure 7.6: Comparing All SC Trendlines.

Figure 7.6 is computed and plotted. The average trendline characteristics,

$$y_c(x) = \frac{y_{N_1}(x) + y_{N_2}(x) + \cdots + y_{N_n}(x)}{n} \tag{7.14}$$

$$= \frac{\sum_{i=1}^{n} y_{N_i}(x)}{n} \tag{7.15}$$

where $y_{N_i}(x)$ - is the normalize value of y at x dB for the i_{th} speech command. Also, $n = 12$ since there are 12 speech commands in this case. The result is the performance characteristic curve shown in Figure 7.7. This curve can be used in predicting the response of the developed speech control interface for the small multirotor UAV, setting the practical limit of the current speech control interface implementation, quantifying the effects of performance modification to the implementation. Also, other speech control methods that use other ASR engines with different underlying theory other than hidden Markov Model (HMM) as is the case with the CMU Sphinx, such as Amazon Echo, Microsoft Cortana, and Apple Siri, could be effectively compared with this implementation using the characteristic performance curve. However, unlike many of the alternative, the current implementation is low-cost and works off-line without relying on network connectivity to function effectively.

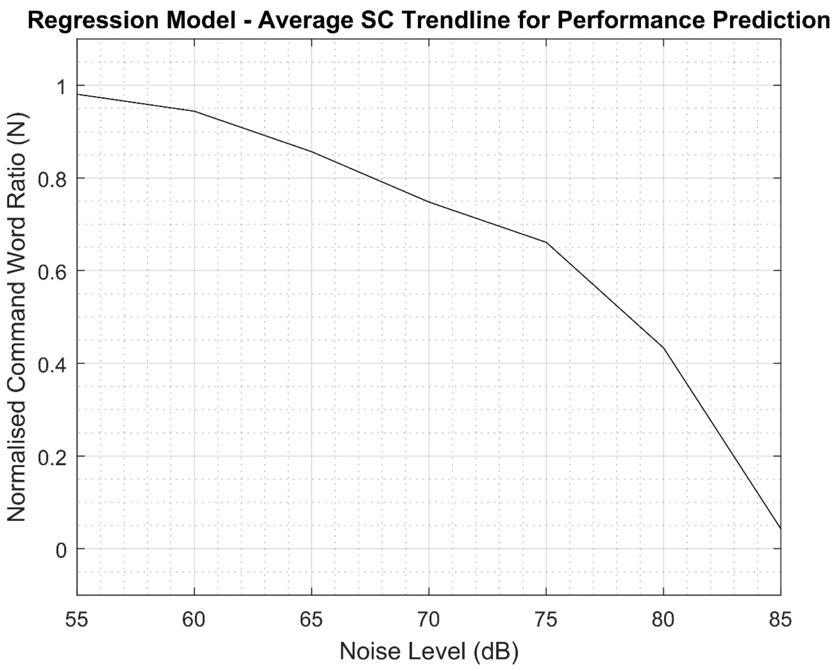


Figure 7.7: SC Trendline Model - Normalised.

7.1.3 Characteristic curve fitting

In order to make the UAV speech command interface performance characteristic easily exportable for other application purposes, the curve was fitted to a polynomial line with three degrees of freedom,

$$y = ax^3 + bx^2 + cx + d \tag{7.16}$$

Where $a = -0.00005147$, $b = 0.00965375$, $c = -0.61458338$, and $d = 14.15673367$, then

$$y = -0.00005147x^3 + 0.00965375x^2 - 0.61458338x + 14.15673367 \tag{7.17}$$

Where $x \in \Re \mid 55 \leq x \leq 85$. The degrees of freedom was considered sufficient as polynomials with higher degree of freedom values had coefficients that were near zero ($\ll 10^{-4}$). Also, higher degree of freedom values risks characteristic curve over-fitting. In addition, because of the nature of the curve, other curve fitting such as linear or exponential were considered unsuitable. The curve fit was generated using MATLAB. The resulting line of best fit equation presented in Equation 7.17, was plotted over the curve generated in Figure 7.7 to give the characteristic curves shown in Figure 7.8. The original characteristic curve is the *black solid line* in the plot, while the fitted characteristic curve is the *red dash-dot line.*

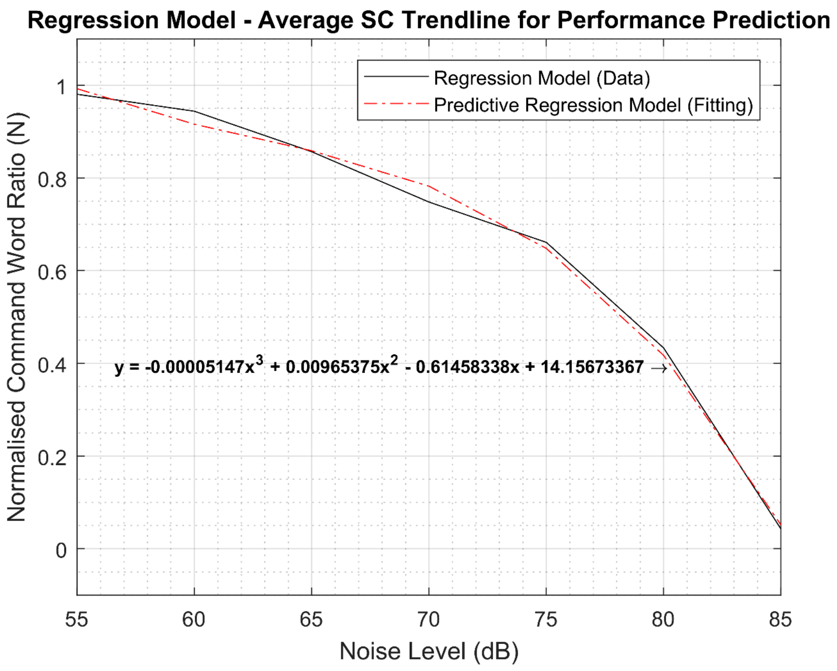


Figure 7.8: SC Trendline Predictive Model (fitting) - Normalised.

7.2 Varying Noise Level - Word Frequency Results

In this section, the experiment results were analysed based on the individual word frequency rather than the speech command phrase as performed in Section 7.1. There are a total of twelve (12) unique words that make up the twelve (12) UAV speech command phrases. Similar to the speech command phrase analysis in the previous section, the total number of each of the 12 words (contained in the 12 speech command phrases) that were successfully recognised were collated across the 37 participants, and presented in Appendix B.2.2. For each word, a frequency map and a trendline plot was generated as presented in Figure 7.9, Figure 7.10, and Figure 7.11.

The frequency map is a plot of the total number of times (zero, one, two, three, four, and five) each word was recorded across all participants for each noise level. For example, Figure 7.9a shows the observation for the first speech word (SW1) ‘Go’ which appeared a total of four times across all 12 speech command phrase. At 55 dB, of the twenty-three participants whose data were successfully captured and processed, twenty-one had all 4 of the 4 ‘Go’ word instances successful, one had 3 of 4 success, and another one had 2 of 4 success. At 60 dB, 4 of 4 ‘Go’ word results were successfully recorded for twenty-five participants, 3 of 4 for five participants, 2 of 4 for two participants, 1 of 4 for two participants, and 0 of 4 for one participant. At 65 dB, of the 36 participants successfully captured and processed, the 4 of 4 ‘Go’ word recognition was successful for 21 participants, 3 of 4 for 3 participants, 2 of 4 for 4 participants, 1 of 4 for

2 participants, and 0 of 4 for 6 participants. At 70 dB, the 4 of 4 'Go' word recognition was thirteen times, 3 of 4 was one time, 2 of 4 was one time, 1 of 4 was five times, and 0 of 4 was sixteen times. At 75 dB, the 4 of 4 'Go' word recognition was seven times, 3 of 4 was three times, 2 of 4 was one time, 1 of 4 was five times, and 0 of 4 was sixteen times. At 80 dB, the 4 of 4 'Go' word recognition was two times, 3 of 4 was two times, 2 of 4 was five times, 1 of 4 was two times, and 0 of 4 was twenty-five times. At 85 dB, the 4 of 4 'Go' word recognition was zero, 3 of 4 was zero, 2 of 4 was zero, 1 of 4 was zero, and 0 of 4 was thirty-six times. From this result, a decresase can be observed in the total number of times each word was recognised as the noise level was increased from 55 dB to 85 dB. This was represented by the trendline shown in Figure 7.9b, which was computed using a similar equation to the weighted trendline Equation 7.1 in the speech command analysis,

$$y(x) = \frac{\sum_{i=0}^{n} d_i(x) f_i(x)}{\sum_{i=0}^{n} f_i(x)} \tag{7.18}$$

Where n - is the total number of each word present in the 12 speech commands combined, as used in the experiment. Note that n is different for each word, for example $n = 4$ for SW1 'Go', $n = 2$ for SW2 'Forward', $n = 1$ for SW8 'Land', $n = 3$ for SW10 'One', and $n = 5$ for SW11 'Metre'. $d_i(x)$ - is a coefficient less than or equal to n specifying the number of recognitions out of n word repetitions in total set of 12 speech commands, for the given x dB noise level. Note that this corresponds to the i^{th} value. $f_i(x)$ - is the frequency (number of times) of the $d_i(x)$ of n recognition for the given word at the given x dB noise level, as indicated on the frequency map.

For example, the points on the trendline for the first speech word (SW1), 'Go' were computed as follows. When $x = 55$ dB,

$$y(55) = \frac{\sum_{i=0}^{4} d_i(55) f_i(55)}{\sum_{i=0}^{n} f_i(55)} = \frac{0 \cdot 0 + 1 \cdot 0 + 2 \cdot 1 + 3 \cdot 1 + 4 \cdot 21}{0 + 0 + 1 + 1 + 21} = 3.8696 \tag{7.19}$$

When $x = 60$ dB

$$y(60) = \frac{\sum_{i=0}^{4} d_i(60) f_i(60)}{\sum_{i=0}^{n} f_i(60)} = \frac{0 \cdot 1 + 1 \cdot 2 + 2 \cdot 2 + 3 \cdot 5 + 4 \cdot 25}{1 + 2 + 2 + 5 + 25} = 3.4571 \tag{7.20}$$

When $x = 65$ dB

$$y(65) = \frac{\sum_{i=0}^{4} d_i(65) f_i(65)}{\sum_{i=0}^{n} f_i(65)} = \frac{0 \cdot 6 + 1 \cdot 2 + 2 \cdot 4 + 3 \cdot 3 + 4 \cdot 21}{6 + 2 + 4 + 3 + 21} = 2.8611 \tag{7.21}$$

When $x = 70$ dB

$$y(70) = \frac{\sum_{i=0}^{4} d_i(70) f_i(70)}{\sum_{i=0}^{n} f_i(70)} = \frac{0 \cdot 16 + 1 \cdot 5 + 2 \cdot 1 + 3 \cdot 1 + 4 \cdot 13}{16 + 5 + 1 + 1 + 13} = 1.7222 \tag{7.22}$$

When $x = 75$ dB

$$y(75) = \frac{\sum_{i=0}^{4} d_i(75) f_i(75)}{\sum_{i=0}^{n} f_i(75)} = \frac{0 \cdot 22 + 1 \cdot 3 + 2 \cdot 1 + 3 \cdot 3 + 4 \cdot 7}{22 + 3 + 1 + 3 + 7} = 1.1667 \tag{7.23}$$

When $x = 80$ dB

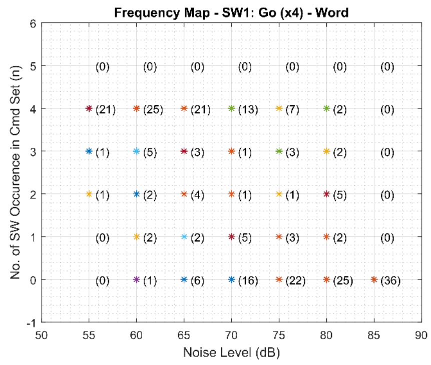
$$y(80) = \frac{\sum_{i=0}^{4} d_i(80) f_i(80)}{\sum_{i=0}^{n} f_i(80)} = \frac{0 \cdot 25 + 1 \cdot 2 + 2 \cdot 5 + 3 \cdot 2 + 4 \cdot 2}{25 + 2 + 5 + 2 + 2} = 0.7222 \tag{7.24}$$

When $x = 85$ dB

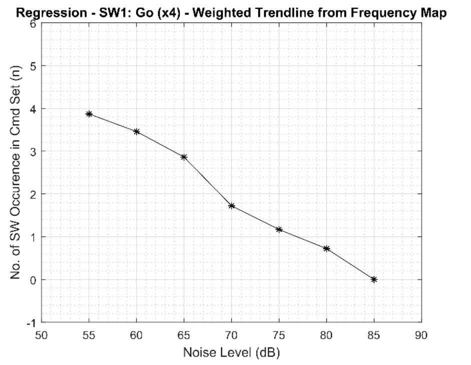
$$y(85) = \frac{\sum_{i=0}^{4} d_i(85) f_i(85)}{\sum_{i=0}^{n} f_i(85)} = \frac{0 \cdot 36 + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 0}{36 + 0 + 0 + 0 + 0} = 0 \tag{7.25}$$

Figure 7.9c and Figure 7.9d show the frequency map and trendline of the second speech word (SW2), ‘Forward’, which appeared only two times in the 12 speech command set. Observe that about half of this command word fails at 70 dB, which is poor for a key command word for which a higher resilience is needed at higher levels of 75 dB and perhaps 80 dB. Figure 7.9e and Figure 7.9f show the frequency map and trendline of the third speech word (SW3), ‘Backward’, which appeared only two times in the 12 speech command set. It had a better performance than SW2, notably at both 75 dB and 80 dB. Figure 7.9g and Figure 7.9h show the frequency map and trendline of the fourth speech word (SW4), ‘Right’, which also appeared only two times in the 12 speech command set, was the most noise resilient and hence the most successful speech command word with a high recognition accuracy of about 90% at 80 dB.

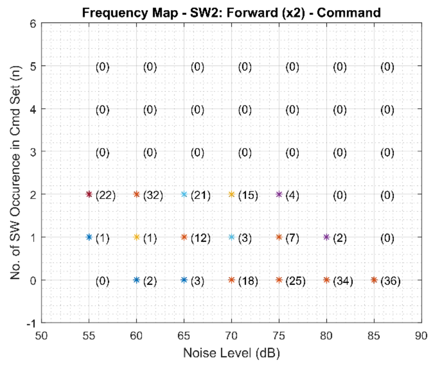
Figure 7.10 shows the results of four speech words, SW5 - SW8. Figure 7.10a and Figure 7.10b show the frequency map and trendline of the fifth speech word (SW5), ‘Left’, which appears twice in the 12 speech command set. Figure 7.10c and Figure 7.10d show the frequency map and trendline for the sixth speech word (SW6), ‘Step’, which appears four times in the 12 speech command set. Figure 7.10e and Figure 7.10f show the frequency map and trendline for the seventh speech word (SW7), ‘Hover’, which appears twice in the 12 speech command set. This was the least successfully recognised speech command word across all participants, with a recognition rate of 36% at 65 dB. Figure 7.10g and Figure 7.10h show the frequency map and trendline for the eighth speech word (SW8), ‘Land’, which appeared only once in the 12 speech command set. Note that this has exactly the same frequency map and trendline characteristic as speech command SC6 ‘Land’ in Figure 7.4a and Figure 7.4b, because it is a single word command that appears only once in the speech command set, therefore both speech command and individual word dimensions of analysis yields the same result.



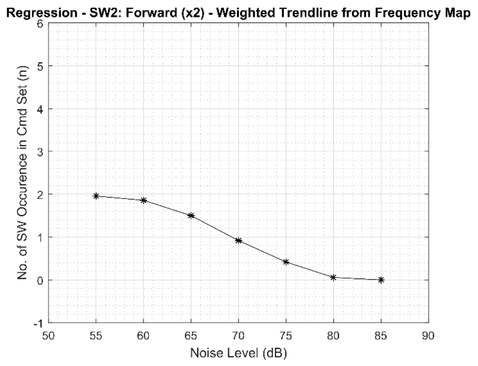
(a) SW1 Frequency Map.



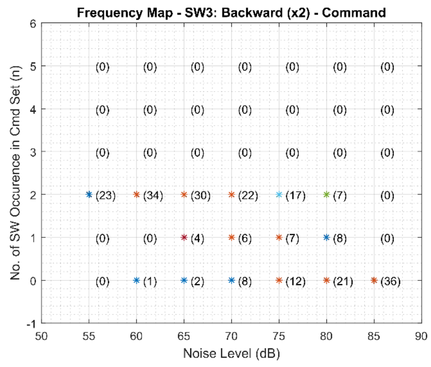
(b) SW1 Trendline.



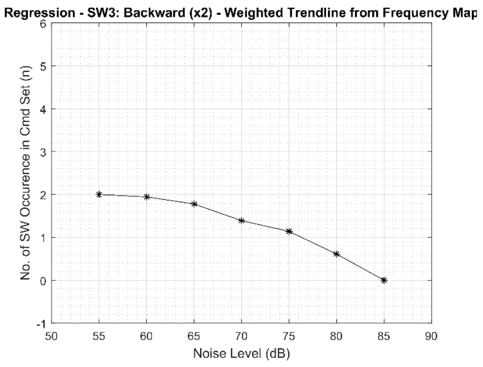
(c) SW2 Frequency Map.



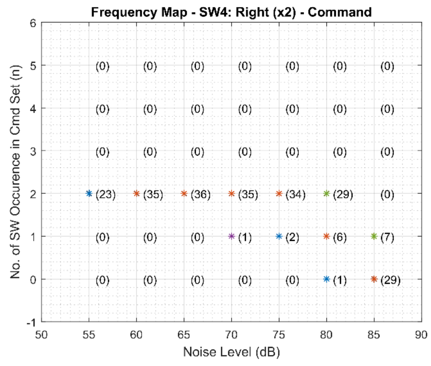
(d) SW2 Trendline.



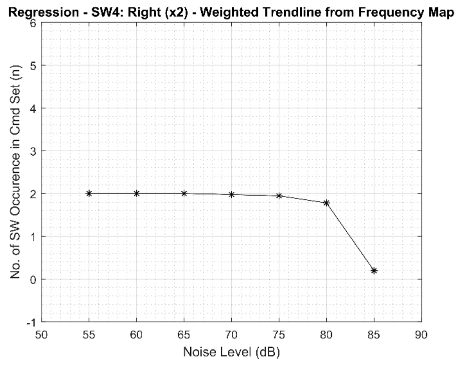
(e) SW3 Frequency Map.



(f) SW3 Trendline.

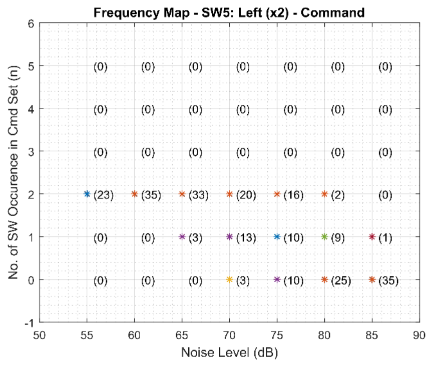


(g) SW4 Frequency Map.

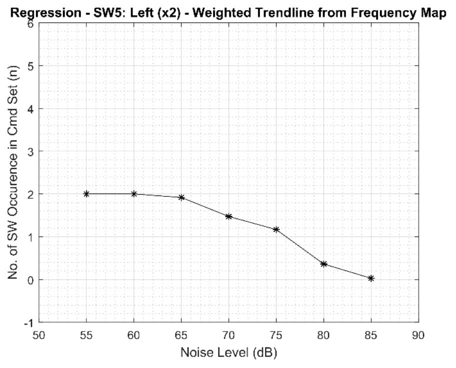


(h) SW4 Trendline.

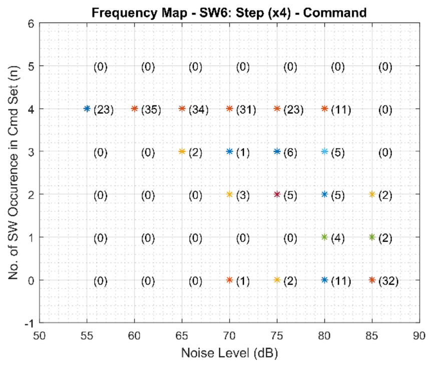
Figure 7.9: Speech Word Frequency Map and Trendlines I.



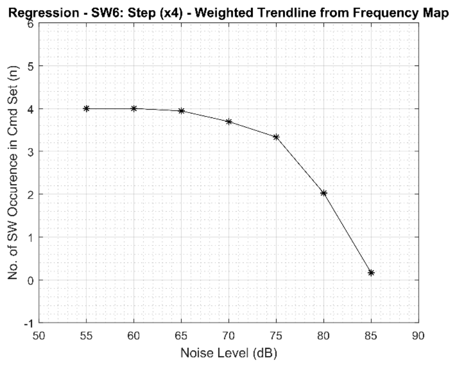
(a) SW5 Frequency Map.



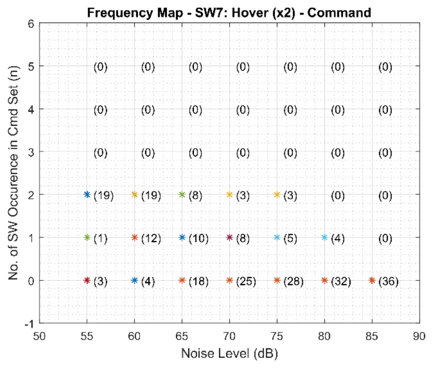
(b) SW5 Trendline.



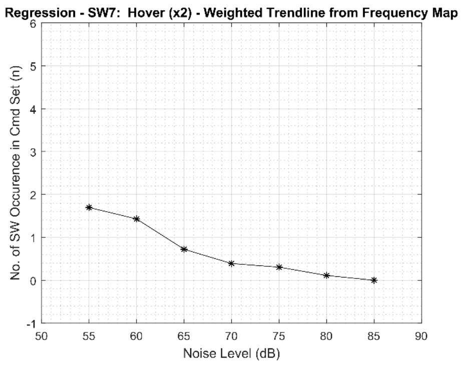
(c) SW6 Frequency Map.



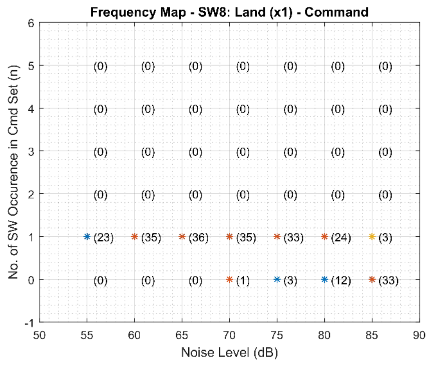
(d) SW6 Trendline.



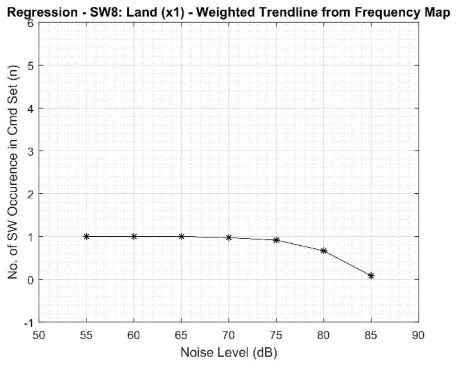
(e) SW7 Frequency Map.



(f) SW7 Trendline.

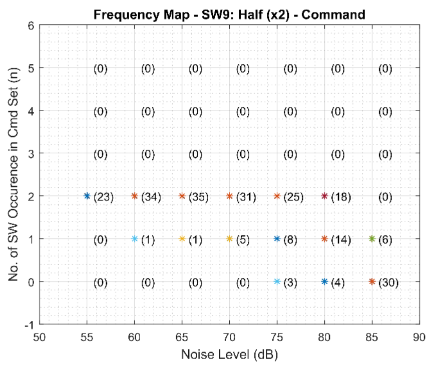


(g) SW8 Frequency Map.

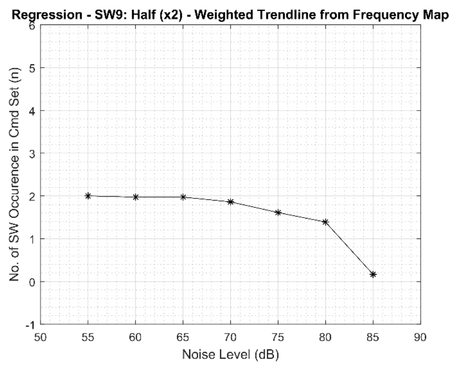


(h) SW8 Trendline.

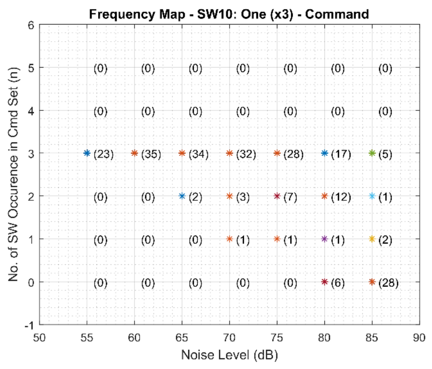
Figure 7.10: Speech Word Frequency Map and Trendlines II.



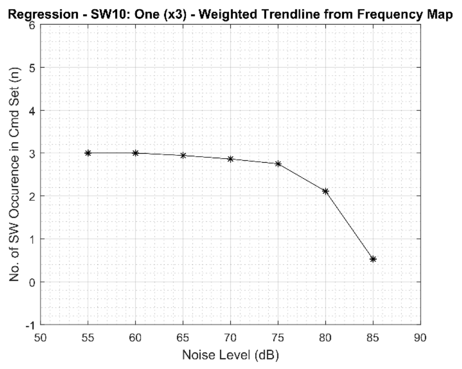
(a) SW9 Frequency Map.



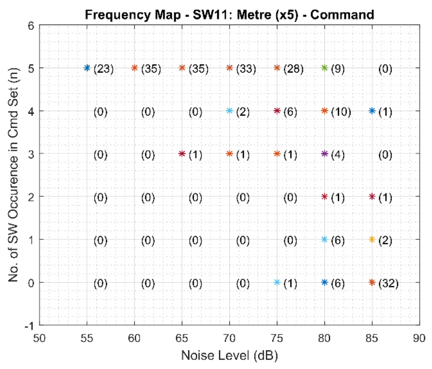
(b) SW9 Trendline.



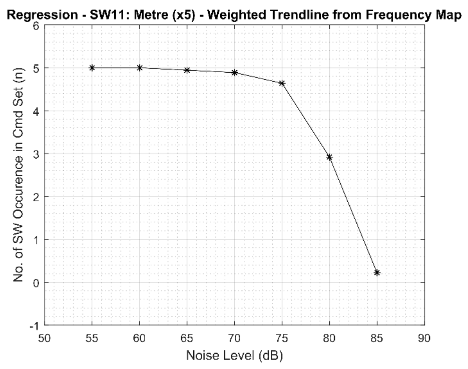
(c) SW10 Frequency Map.



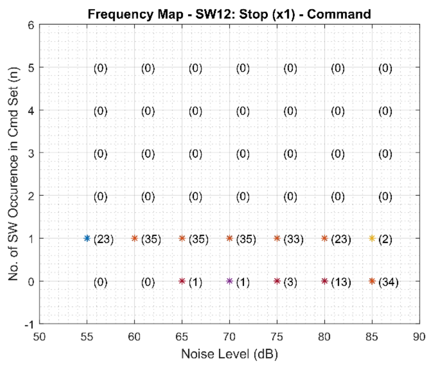
(d) SW10 Trendline.



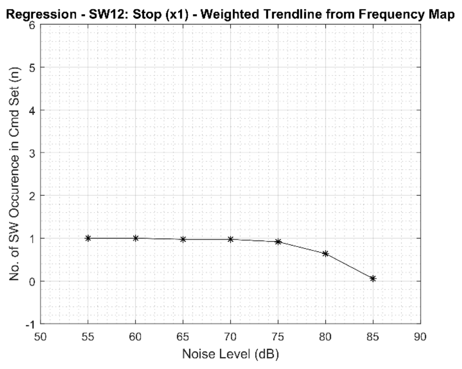
(e) SW11 Frequency Map.



(f) SW11 Trendline.



(g) SW12 Frequency Map.



(h) SW12 Trendline.

Figure 7.11: Speech Word Frequency Map and Trendlines III.

Figure 7.11 shows the results of the remaining four speech words, SW9 - SW12. Figure 7.11a and Figure 7.11b show the frequency map and trendline of the ninth speech word (SW9), 'Half', which appears twice in the 12 speech command set. Figure 7.11c and Figure 7.11d show the frequency map and trendline for the tenth speech word (SW10), 'One', which appears three times in the 12 speech command set. Figure 7.11e and Figure 7.11f show the frequency map and trendline for the eleventh speech word (SW11), 'Metre', which appears five times in the 12 speech command set. This had the highest frequency of occurrence because it is a modifier specifying the unit of movement in any direction being given by the keyword. Figure 7.11g and Figure 7.11h show the frequency map and trendline for the twelfth speech word (SW12), 'Stop', which appeared only once in the 12 speech command set. This has exactly the same frequency map and trendline characteristic as speech command SC12 'Stop' in Figure 7.5e and Figure 7.5f, because it is a single word command that appears only once in the speech command set, therefore both speech command and individual word dimensions of analysis yields the same result.

7.2.1 Speech Word Performance Comparison

The performance of each of the speech words was compared using the same method described in Section 7.1.1 with the aid of Equation 7.13. The resulting comparison plot is shown in Figure 7.12. Note that the poorest performance was observed in the speech word recognition of SW7, 'Hover', followed by SW2 'Forward' and SW1 'Go' as indicated in the combined trendline in Figure 7.12. The best speech recognition performance was observed in SW4 'Right', SW8 'Land', SW10 'One', SW11 'Metre', and SW12 'Stop'.

7.2.2 Experiment speech word (SW) ASR characteristic

Similar to the SC ASR Characteristic curve presented in Section 7.1.2, the average of the normalised trendline plotted in Figure 7.12 is computed and plotted to give the SW ASR Characteristic shown in Figure 7.13, which was computed with the aid of Equation 7.15.

7.2.3 SW characteristic curve fitting

Fitting the SW ASR characteristic curve to a three degree of freedom polynomial curve of the form,

$$y = ax^3 + bx^2 + cx + d \tag{7.26}$$

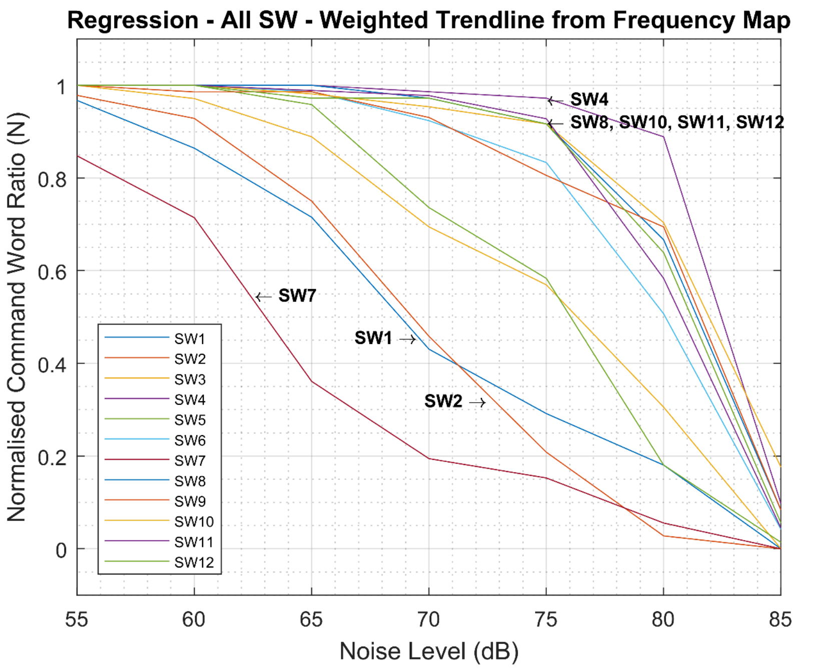


Figure 7.12: Comparing All SW Trendlines.

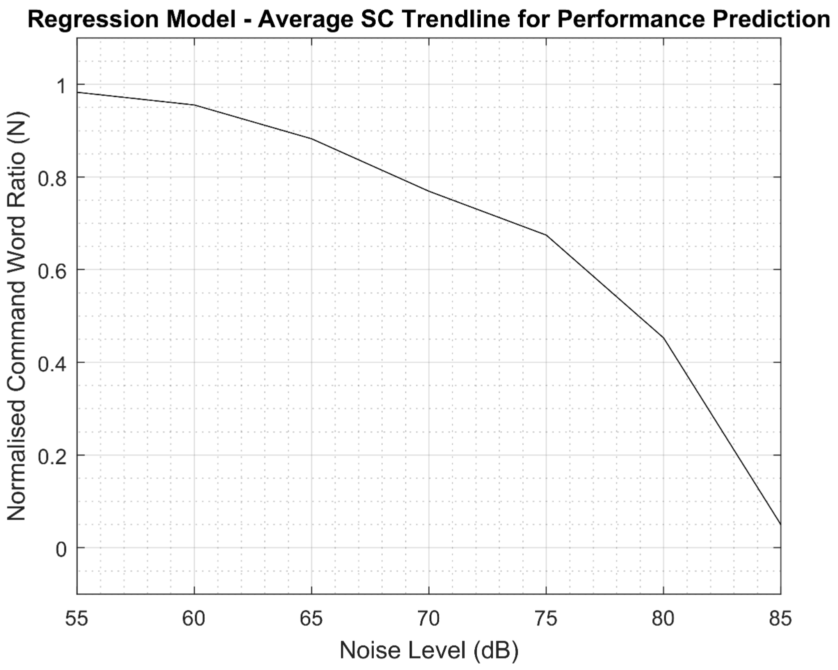


Figure 7.13: SW Trendline Model - Normalised.

Where $a = -0.00004949$, $b = 0.00916273$, $c = -0.57521151$, and $d = 13.14811872$, yielded

$$y = -0.00004949x^3 + 0.00916273x^2 - 0.57521151x + 13.14811872 \tag{7.27}$$

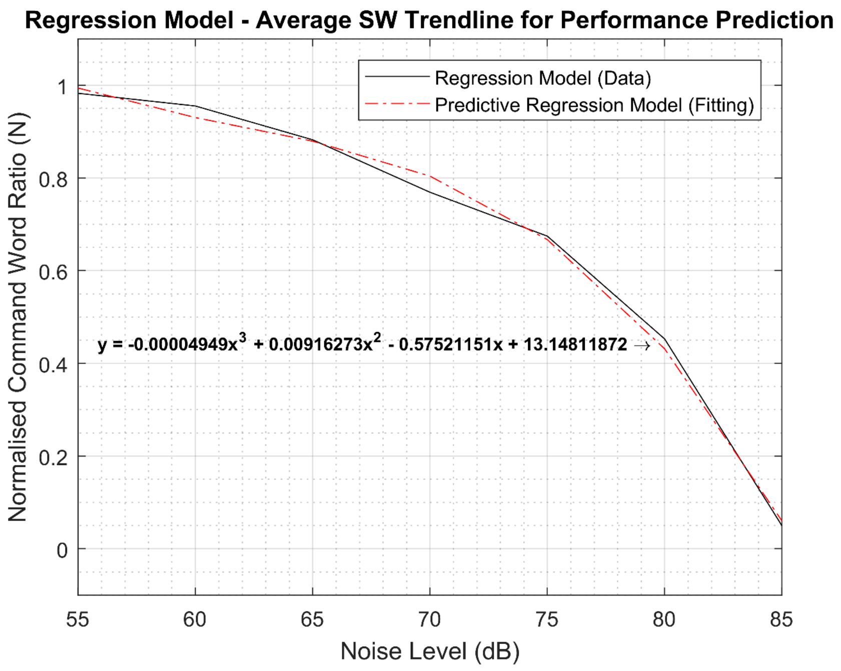
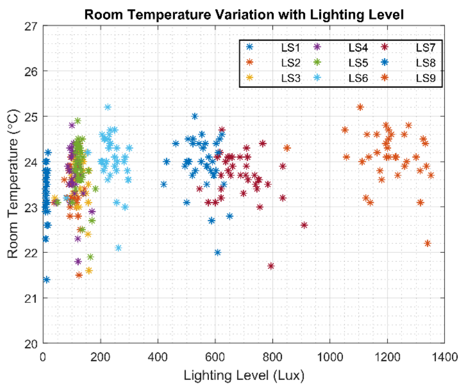


Figure 7.14: SW Trendline Predictive Model (fitting) - Normalised.

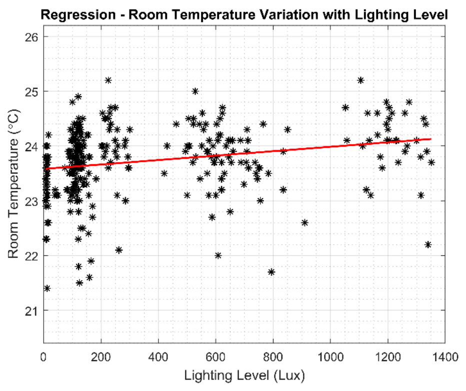
Where $x \in \Re \mid 55 \leq x \leq 85$. This is plotted as shown in Figure 7.14, where the original characteristic curve is the *black solid line* in the plot, and the fitted characteristic curve is the *red dash-dot line*.

7.3 Varying Lighting Level Results

This section presents the result from the varying lighting level (VLL) experiments. The complete result from this segment of the experiment is tabulated in the figures in Appendix B.2.3, which consists of about 999 gesture observations, from nine lighting stages, three background quality experiment per stage, and 37 experiment participants. The blanks indicated by an hyphen are points were the data was not available due to later improvement in experimental condition after preliminary testing. For example, the blue and green background were not used during preliminary testing (participant 1 - 5) but were then made available for the significant remainder of the test (participant 6 - 37). Also, the whole of participant A10's data in this segment could not be captured because the equipment calibration failed for the participant. The implication of this is that the total number of observations for all participants for most of the lighting stage parameters is 36, and 31 for the green and blue background parameter. Also, although the same variable knob setting was used for all participants, slightly different lighting level values were measured up to a span of around 200 lux at higher lighting levels, which also accounts for the scatter observed in the plots. This was due to 1) stray light rays from corridors due to people and lab equipment movements, 2) weather-dependent daylight level penetration via shut windows, and 3) reflection from workstation computer monitors and screens within the experiment lab. However, this was not a problem for the result analysis, since the aim is to observe the quality trend from low to high intensity on different lighting background and with different lighting source. They were considered as white noise whose persistent presence throughout the experiment evenly cancels out their effect eventually. For analysis where the scatter could be a problem, the mean lighting level value could be computed across all participants and use as the lighting stage lighting level value. Figure 7.15 shows the result of room temperature variation for each participant as they progress through different lighting levels going from lighting stage 1 (LS1) to lighting stage 9 (LS9), and it helps validate the fact that the slight differences in the lighting level for each participant does not affect the general trend of rising temperature during experiment.



(a) Scatter plot



(b) Regression

Figure 7.15: Room temperature change during the varying light level experiment.

Figure 7.15a is a scatter plot of the room temperature against lighting level during the experiment progression through the lighting stages. The scatter plot has been colour coded to collate lighting level reading at the same lighting stage together. In Figure 7.15b, a line of best fit was drawn over the scatter plot data in Figure 7.15a, using MATLAB. It can be observed from Figure 7.15b that temperature was gradually rising during the course of the experiment, at a gradient of $a = 0.00040479$, given that the equation of the line is

$$y = ax + b = 0.00040479x + 23.58151862 \tag{7.28}$$

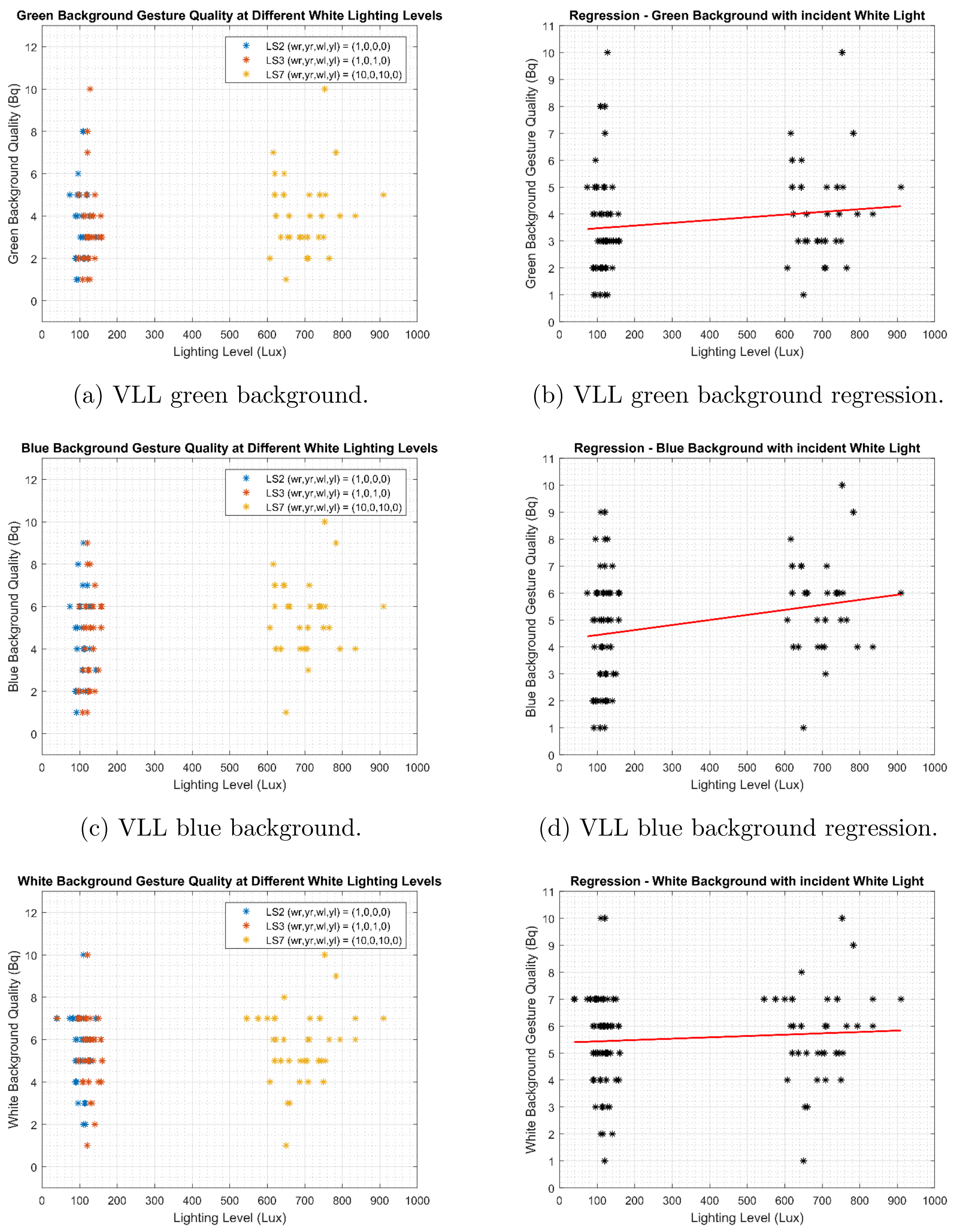
There are nine lighting stages (LS1-LS9). At each lighting stage, each of the different background qualities are estimated based on how distinct the finger gestures were clearly recognised using a numeric scale of 1-10, with '1' being a complete failure in gesture recognition, '3' being the hand outline was successfully registered, '5' being all finger gestures being successful but with high frequency noise fluctuations, and '7' being all fingers were clearly distinguished but with small low frequency fluctuations (one in 10 seconds), and '10' being perfect steady recognition, with no noise fluctuations within 60 seconds. The results from the varying light level experiment would be presented in three sections according to the incident lighting colour of white, yellow, and mixed.

7.3.1 White lighting on different background

Figure 7.16 shows the result of the varying lighting level experiment when only the white LED lighting was used on the different backgrounds. These were the lighting stages 2, 3 and 7 (LS2, LS3, and LS7) steps of the experiment. Figure 7.16a shows the scatter plot of the result of the finger gesture recognition quality on the green background against the white LED lighting level in Lux (luminous flux incident on the background surface per unit-area/square-metre). Figure 7.16b shows the line of best fit for the scatter plot shown in Figure 7.16a. The equation of the line of best fit was $y = 0.00101495x + 3.37095643$ with a gradient of about 0.10%. Figure 7.16c shows the scatter plot of the result of the finger gesture recognition quality on the blue background against the white LED lighting level in Lux. Figure 7.16d shows the line of best fit for the scatter plot shown in Figure 7.16c. The equation of the line of best fit in Figure 7.16d was $y = 0.00186688x + 4.25523286$ with a gradient of about 0.19%. Figure 7.16e shows the scatter plot of the result of the finger gesture recognition quality on the white background against the white LED lighting level in Lux. Figure 7.16f shows the line of best fit for the scatter plot shown in Figure 7.16e. The equation of the line of best fit in Figure 7.16f was $y = 0.00049768x + 5.38390575$ with a gradient of about 0.05%.

7.3.2 Yellow lighting on different background

Figure 7.17 shows the result of the varying lighting level experiment when only the yellow LED lighting was used on the different backgrounds. These were the lighting stages 4, 5 and 8 (LS4, LS5, and LS8) steps of the experiment. Figure 7.17a shows the scatter plot of the result of the finger gesture recognition quality on the green background against the yellow LED lighting level in Lux. Figure 7.17b shows the line of best fit for the scatter plot shown in Figure 7.17a. The equation of the line of best fit was $y = 0.00335540x + 3.17575769$ with a gradient of about 0.34%. Figure 7.17c shows the scatter plot of the result of the finger gesture recognition quality on the blue background against the yellow LED lighting level in Lux. Figure 7.17d shows the line of best fit for the scatter plot shown in Figure 7.17c. The equation of the line of best fit in Figure 7.17d was $y = 0.00296837x + 3.93542753$ with a gradient of about 0.30%. Figure



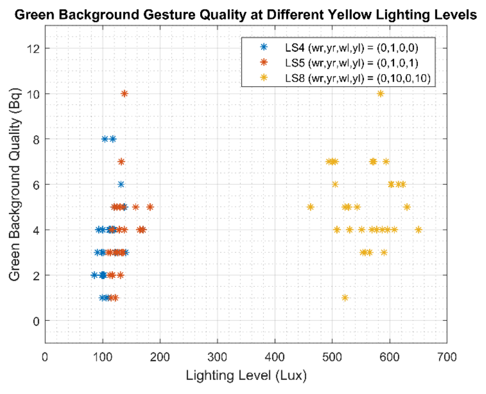
(a) VLL green background. (b) VLL green background regression.

(c) VLL blue background. (d) VLL blue background regression.

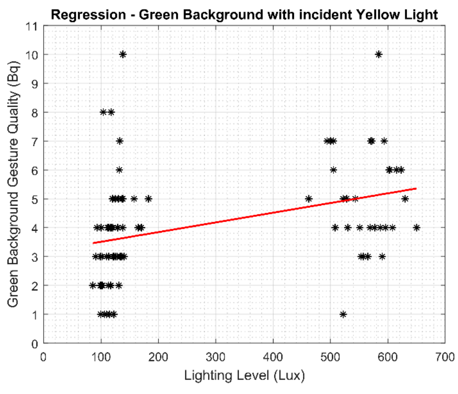
(e) VLL white background. (f) VLL white background regression.

Figure 7.16: VLL white incident LED light on green, blue, and white background.

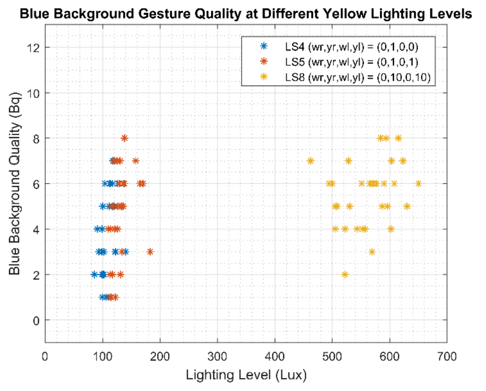
7.17e shows the scatter plot of the result of the finger gesture recognition quality on the white background against the yellow LED lighting level in Lux. Figure 7.17f shows the line of best fit for the scatter plot shown in Figure 7.17e. The equation of the line of best fit in Figure 7.17f was $y = 0.00035590x + 4.32313692$ with a gradient of about 0.04%.



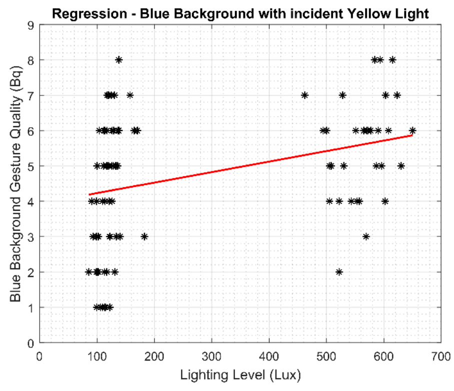
(a) VLL green background.



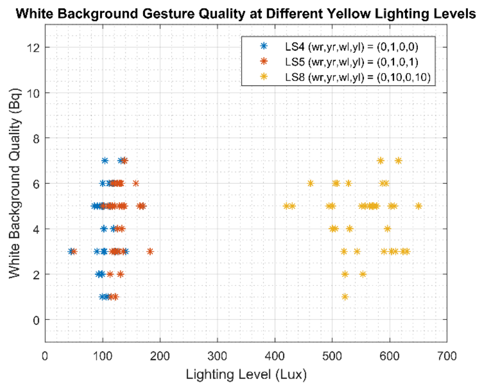
(b) VLL green background regression.



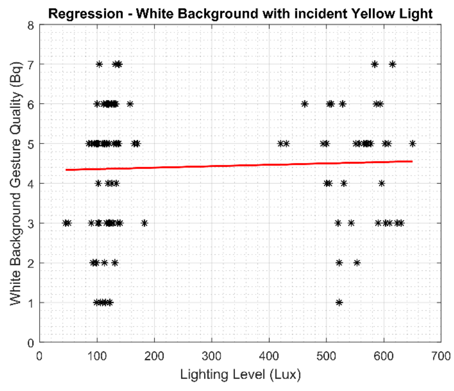
(c) VLL blue background.



(d) VLL blue background regression.



(e) VLL white background.

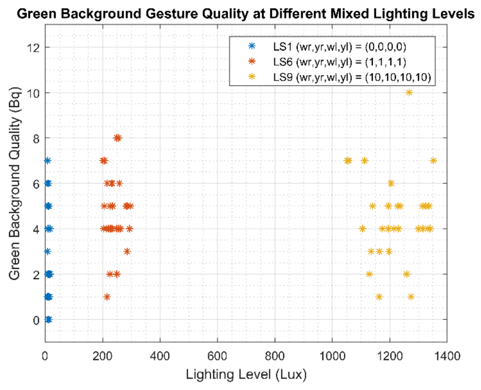


(f) VLL white background regression.

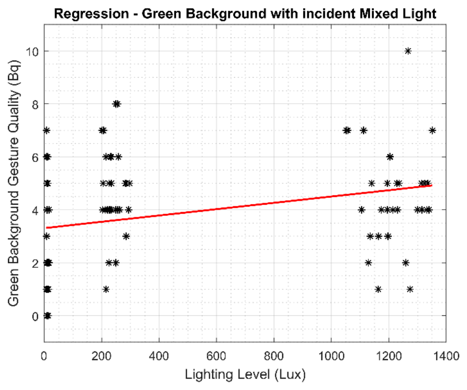
Figure 7.17: VLL yellow incident LED light on green, blue, and white background.

7.3.3 Mixed white and yellow lighting on different background

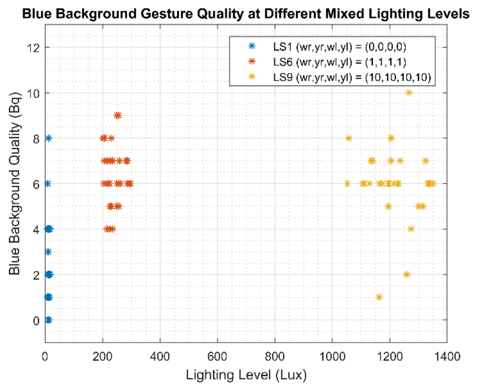
Figure 7.18 shows the result of the varying lighting level experiment when the white and yellow LED lighting were combined on the different backgrounds. These were the lighting stages 1, 6 and 9 (LS1, LS6, and LS9) steps of the experiment. Figure 7.18a shows the scatter plot of the



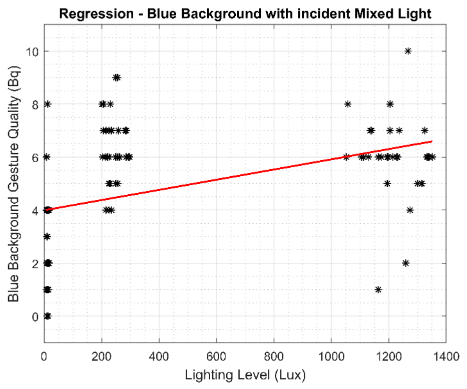
(a) VLL green background.



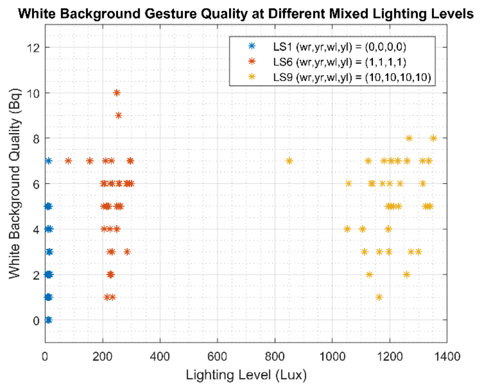
(b) VLL green background regression.



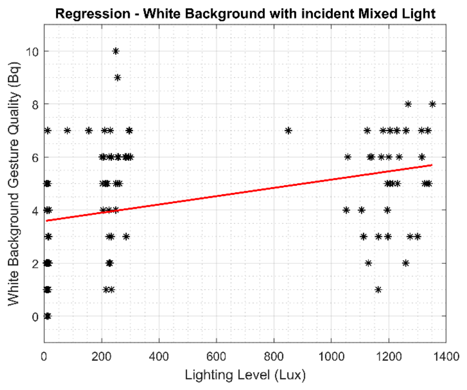
(c) VLL blue background.



(d) VLL blue background regression.



(e) VLL white background.



(f) VLL white background regression.

Figure 7.18: VLL mixed white and yellow incident LED light on green, blue, and white background.

result of the finger gesture recognition quality on the green background against the mixed white and yellow LED lighting level in Lux. Figure 7.18b shows the line of best fit for the scatter plot shown in Figure 7.18a. The equation of the line of best fit was $y = 0.00119177x + 3.30883877$

with a gradient of about 0.12%. Figure 7.18c shows the scatter plot of the result of the finger gesture recognition quality on the blue background against the mixed white and yellow LED lighting level in Lux. Figure 7.18d shows the line of best fit for the scatter plot shown in Figure 7.18c. The equation of the line of best fit in Figure 7.18d was $y = 0.00192255x + 3.99397086$ with a gradient of about 0.19%. Figure 7.18e shows the scatter plot of the result of the finger gesture recognition quality on the white background against the mixed white and yellow LED lighting level in Lux. Figure 7.18f shows the line of best fit for the scatter plot shown in Figure 7.18e. The equation of the line of best fit in Figure 7.18f was $y = 0.00156143x + 3.58726852$ with a gradient of about 0.16%.

7.4 Discussion

7.4.1 Speech command phrase

The result of the experiment shows that speech recognition accuracy/success rate falls as noise levels rises. A regression model was developed from the empirical observation of nearly $3,108$ speech command utterances - that is 12 speech commands, repeated for each of 7 noise levels, by 37 different experiment participants. The polynomial curve fitting characteristic generated for the custom CMU Sphinx ASR can be used to predict speech recognition performance for aerial robotic systems where the CMU Sphinx ASR engine is being used, as well as in the performance comparison with other ASR engines. Also, it was not clear how the length of speech (the number of speech words in phrase) affects the recognition accuracy, because while there are evidence supporting single-word poor performance like ‘hover’, there are alternative evidence supporting single-word good performance for words like ‘land’ and ‘stop’, in the experiment. However, multi-word speech commands may be more reliable and effective than single-word commands due to the possibility of introducing a syntax error checking stage in the recognition process to validate the control command. The composition of multi-word speech commands consists of keywords and modifiers. For example in the SC7 command “Go Forward Half Metre”, the primary keyword is ‘Forward’, the secondary keyword is ‘Go’, the primary modifier is ‘Half’, and the secondary modifier is the unit ‘Metre’. In multi-word speech commands, the primary keyword and the primary modifier is the most significant, as the failure of these primary parameters would result in the command execution failure. Secondary modifiers aids system usability particularly for human operators, and could be used for error checking within the UAV system to ensure fidelity of command communication.

7.4.2 Speech command word

From the results of the experiments, it was observed that speech command selection affects the speech recognition accuracy, as some speech command words were found to be more resilient to corruption at higher noise levels, maintaining over 90% success rate at 75 dB whereas the average rate at 75 dB was just over 65%. In other words, the success of speech command words such as SW4 'right', SW8 'land', and SW12 'stop', at higher noise levels suggests that some speech command words are more noise resistant than others and that a careful selection of these as part of the speech command phrase could improve the robustness and accuracy of speech recognition in spite of high noise levels, thereby contributing to a more successful human aerobotic speech control interaction.

7.4.3 Aerobot speech interaction

Figure 7.19 shows the comparison of the speech word (SW) and speech command (SC) trend-line characteristic curve. The *red solid line* represents the SC curve while the *black dash-dot line* represents the SW curve. Observe that the two curve trends are very similar with the SW curve being below the SC curve, although the SW curve seems to be a replica of the SC curve. The result of the characteristic curve comparison suggests that the speech command phrase slightly outperforms the speech command word, thereby supporting the argument in favour of multi-word commands over single-word commands.

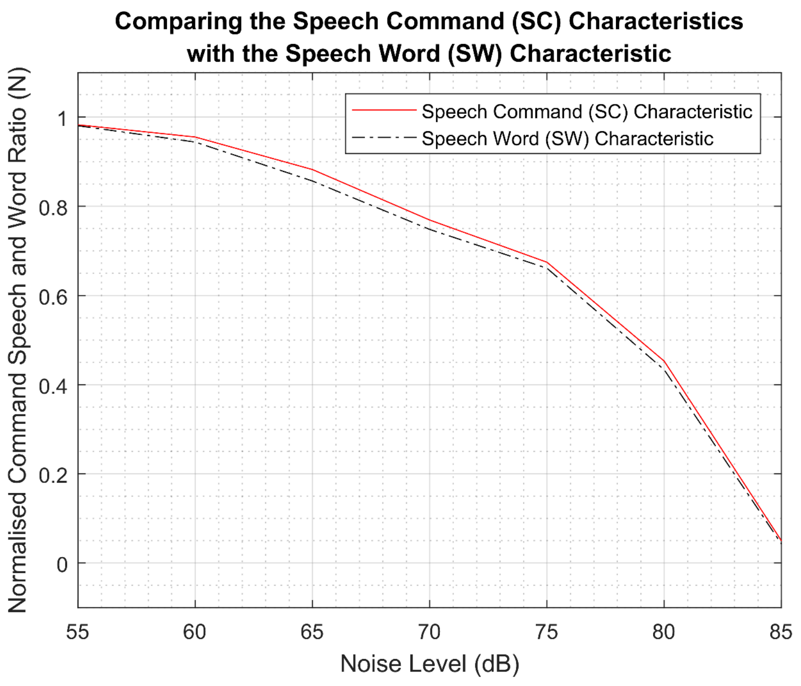


Figure 7.19: SC-SW Trendline characteristic comparison.

Based on this analysis and the observations in this research, it is recommended that the upper limit of 75 dB noise application level should be set for practical aerobot speech interaction. This was because at 75 dB, the speech recognition accuracy/success was about 65%, falling below average at 80 dB, and below 5% at 85 dB. In other words, speech is not an effective means of control interaction with an aerobot beyond 75 dB noise level, as the speech control interaction becomes very unreliable due to poor recognition. However, based on the typical UAV noise level data presented in Table 4.4, it is clear that this limit is below the application range of most small multirotor UAVs. Therefore, there is a need to push the upper limit beyond 75 dB to atleast 80 dB or more. Therefore, the following suggestions are given in order to improve the aerobot speech application:

1. Alternative ASR engine: Consider the use of alternative speech ASR technology with a different learning model from the CMU Sphinx's hidden markov model, such as artificial neural network or deep learning, for improved noise performance.
2. Low noise design: Improve small multirotor UAV propulsion system design to be low noise.
3. Application environment: Deploy application only in sub-75 dB noise conditions.
4. Speech capture hardware: Consider the use of directional, noise canceling, or array microphones.
5. Speech command selection: Optimise the selection of multi-word and single-word speech command in favour of more resilient command variant, as observed in this particularly study.

In Abioye et al. (2018a), it was observed that ambient noise level above 80 dB significantly affected speech capture. While this conclusion still holds true for the current research work, the current research work had more experiment data to analyse in order to more precisely determine the practical application limits, as presented in this section.

Also, note that even as propeller noise level decreases, as in the DJI Mavic pro platinum drone DJI (2017a), there are other environmental noise factors that may not be controllable, such as the noise generated by crowds, weather storms, car traffic, heavy duty vehicles on sites, industrial machines, etc. Therefore, robust speech recognition, in spite of noise, not only in the absence of noise, is needed for practical real world UAV applications.

7.4.4 Lighting level and background effect on gesture recognition quality

The white lighting experiment emulates outdoor daylight at 5500 K colour temperature, while the yellow and mixed lighting experiments emulates indoor lighting conditions of 3500 K - 5500

K colour temperatures. The idea is to consider the aerobot gesture application in both indoor and outdoor (field application) environments. From the varying lighting level experiment results, it was observed that there was only a little improvement in the quality of gesture recognition going from lower light levels to higher lighting levels, as indicated by the gradients from the lines of best fit. Low gradient implies low gesture performance improvement over increasing Lux lighting levels. This observation was consistent across all lighting source (white, yellow, and mixed) and for all three backgrounds of green, blue, and white. The line gradients indicated how much the recognition quality improved from low lighting to higher lighting level, while the intercept indicated the minimum quality threshold. For the white lighting experiment, the blue background had the best improvement with a gradient of 0.19% while the white background had the better minimum quality threshold of 5.38, which means that the finger gestures were clearly recognised but with high frequency noise fluctuations. The green background had the least improvement. For the yellow lighting experiment, the green background had the best improvement with a gradient of 0.34% while the white background had the better minimum quality threshold of 4.32, which means that the finger gestures were barely distinctly recognised. But the blue background had the better balance combination of gradient and intercept of 0.30% and 3.94. For the mixed white and yellow lighting experiment, the blue background had the best performance with the best improvement gradient of 0.19% and the best minimum quality threshold of 3.99. The green and white background were similar in their performance.

From these results, the effects of both lighting conditions and the environment background on the quality of gesture recognition, was almost insignificant, less than 0.5%. Therefore, other factors such as the gesture capture system design and technology (camera and computer hardware), type of gesture being captured (upper body, whole body, hand, fingers, or facial gestures), and the image processing technique (gesture classification algorithms) are more important in successfully recognising gesture commands. However, the setup of the gesture capture system and the recognition processing system would still need to take into account the application environmental conditions in order to develop an optimum gesture command interface.

7.5 Chapter Conclusion

In this chapter, the result of the experiment study on the effects of varying noise levels and varying lighting levels on speech and gesture control command interfaces for aerobots, was presented, analysed, and discussed. It was observed from the results of the experiment that speech recognition accuracy/success rate falls as noise levels rise. A regression model was developed from the empirical observation of nearly $3,108$ speech command utterances by 37 participants. Multi-word speech commands with primary and secondary modifier parameters, were thought to be more reliable and effective than single-word commands due to the possibility of syntax error checking. Also, it was observed that some speech command words were more noise resistant than others, even at higher noise levels, and that a careful selection of these as part of the speech command phrase could improve the robustness and accuracy of speech recognition at such high noise levels. Speech, based on the custom CMU Sphinx ASR system developed, is not an effective means of control interaction with an aerobot beyond 75 dB noise level, as the speech control interaction becomes very unreliable due to poor recognition. There is a need to push the upper limit beyond 75 dB to atleast 80 dB or more, based on current UAV noise ratings (Table 4.4). Suggestions were made on how to push this limit. The importance of robust speech recognition techniques that perform excellently, in spite of noise, in practical UAV applications, was highlighted. From the results of the gesture-lighting experiment, the effects of both lighting conditions and the environment background on the quality of gesture recognition, was almost insignificant, less than 0.5%, which means that other factors such as the gesture capture system design and technology (camera and computer hardware), type of gesture being captured (upper body, whole body, hand, fingers, or facial gestures), and the image processing technique (gesture classification algorithms), are more important in developing a successful gesture recognition system.

7.5.1 Limitation

The main limitation of the varying lighting experiment study in this research was that the lighting level range was constrained between twilight (10 Lux) conditions and an overcast day condition (1400 Lux), typical for indoor lighting levels, whereas outdoor lighting conditions could vary up to 100,000 Lux on a very sunny day with direct sunlight, as shown in Table 4.6. Therefore, the gesture-lighting application described in this study is more applicable to indoor scenarios such as the domestic application scenario described in Section 1.4.2. The gesture-lighting application for the outdoor search and rescue scenario described in Section 1.4.1 is limited to cloudy overcast day conditions.

7.5.2 Further work

In order to improve the application limit of the aerobot speech interface, alternative automatic speech recognisers with a different learning model from the CMU Sphinx's hidden markov model, should be considered. ASRs based on artificial neural network or deep learning such as the IBM Watson speech to text cloud AI service (IBM, 2010), are suggested. A hardware upgrade to directional, noise cancelling, and array microphones should also be considered. Multi and single word speech command selection should be optimised in favour of the more resilient command variant as observed in this particularly study.

In order to improve the gesture performance, better hardware (cameras and single board computers) capable of capturing high resolution images and performing complex graphic computation processing in real time, should be used. In addition, instead of the appearance-based 2D-model algorithm used in this research, a more advance 3D-model based AI gesture algorithm should be developed for capturing and recognising hand, arm, and upper body gestures.

Chapter 8

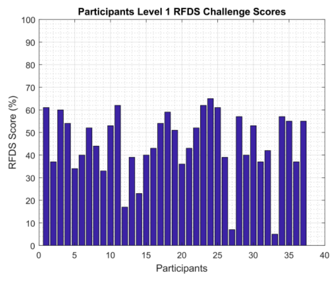
The performance and cognitive workload analysis of the mSVG UAV control interface

In this chapter, the results of the performance and cognitive workload comparison experiment study between the RC joystick (RCJ) and mSVG (multimodal speech and visual gesture) interfaces are presented, analysed, and discussed. The RFDS (Real Flight Drone Simulator) results are first presented, then the Path_v02 completion time and navigaiton accuracy results are presented next. After which the cognitive workload results are presented, and then the SGR usage ratio of the mSVG interface during the Path_v02 flight path navigation. This is followed by the statistical analysis of the Path_v02 flight path navigation result using both the analysis of variance (ANOVA) and the two-sample t-test for significance testing to validate the result. Finally, the implications of these results were also discussed.

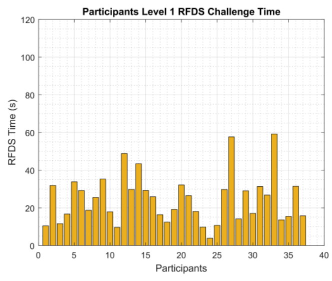
As discussed in Section 4.5, the experiment study was designed in Section 4.5.2, 37 participants were recruited in Section 4.5.4, who performed the experiment procedure as described in Section 4.5.3. The Keyboard (KBD) interface was introduce to simulate the RCJ interface in altitude, atittude, and position (AAP) assist mode, such that a comparison was made between the mSVG interface and both the standard and AAP-modified RCJ interfaces.

8.1 Task B Experiment Results

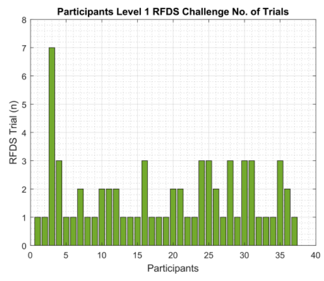
A digital spreadsheet copy of the Task B experiment result is submitted with this thesis. These results are also presented in the Tables in Appendix B.3 for ease of referencing.



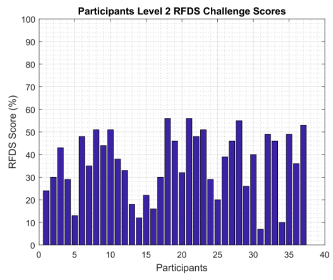
(a) Level 1 RFDS scores.



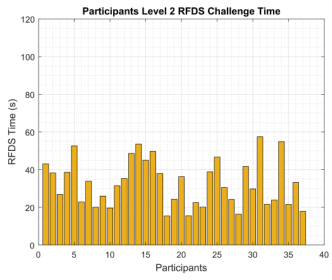
(b) Level 1 RFDS time.



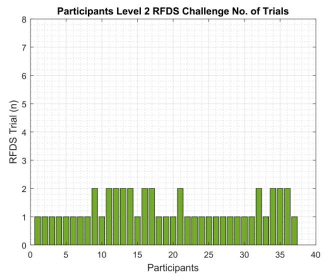
(c) Level 1 RFDS trials.



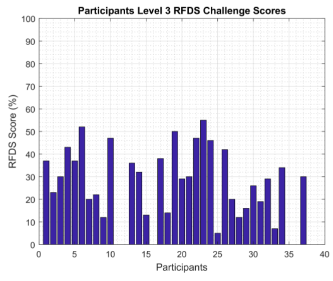
(d) Level 2 RFDS scores.



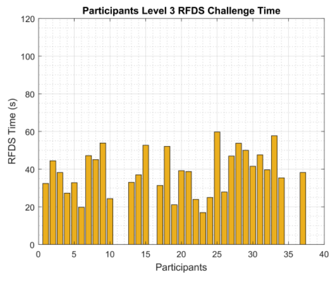
(e) Level 2 RFDS time.



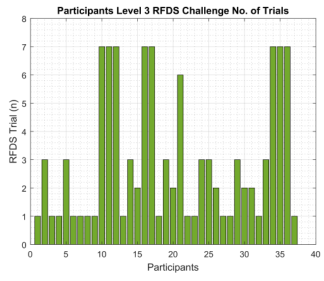
(f) Level 2 RFDS trials



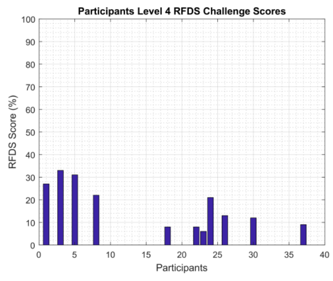
(g) Level 3 RFDS scores.



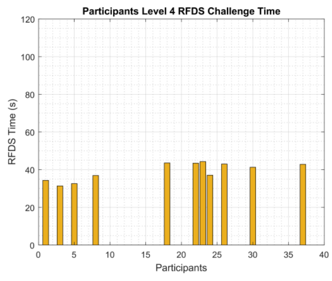
(h) Level 3 RFDS time.



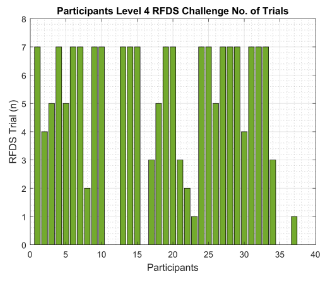
(i) Level 3 RFDS trials



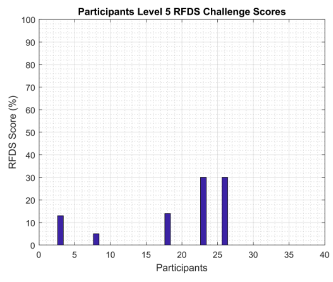
(j) Level 4 RFDS scores.



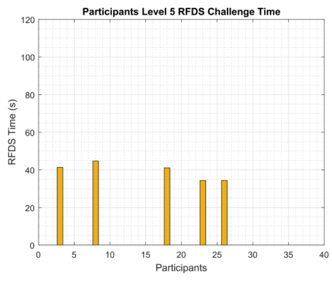
(k) Level 4 RFDS time.



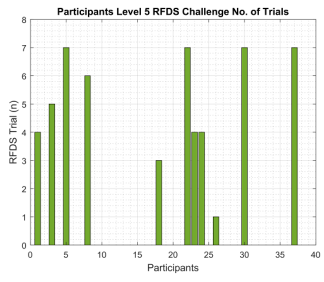
(l) Level 4 RFDS trials



(m) Level 5 RFDS scores.



(n) Level 5 RFDS time.



(o) Level 5 RFDS trials

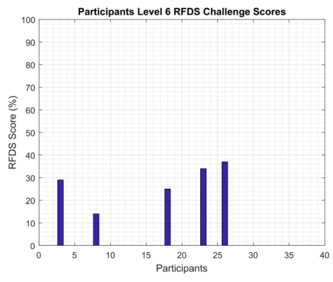
Figure 8.1: RFDS Level 1 to 5 results.

8.1.1 RFDS result

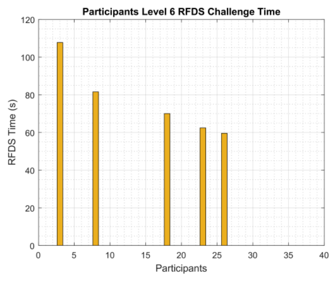
Figure 8.1 shows the results of the 37 participants' RFDS challenge performance for level 1 to 5. Each participant's result consists of three components - level score, time of completion, and the number of trials to pass the level. The level score is a function of the completion time. In the RFDS software, a default upper and lower time threshold are set, beyond which the participants scores a 100%, below which a participant scores zero, and in between, a linear interpolation is performed to determine the scores. Figure 8.1a shows the scores of each of the 37 participants for the RFDS level 1. Figure 8.1b shows the completion time of the RFDS level 1 challenge. It can be observed from Figure 8.1b and Figure 8.1a that the completion time is inversely proportional to the score attained, i.e. the shorter the completion time, the higher the score attain, and vice versa. Figure 8.1c shows the number of trials of each of the 37 participants for the RFDS level 1 challenge. All participants successfully completed the RFDS level 1 challenge within 7 attempts. Figure 8.1d shows the scores of each of the 37 participants for the RFDS level 2 challenge. Figure 8.1e shows the completion time of the RFDS level 2 challenge. Figure 8.1f shows the number of trials of each of the 37 participants for the RFDS level 2 challenge. All participants successfully completed the RFDS level 2 challenge in 2 attempts.

Figure 8.1g shows the scores of each of the 37 participants for the RFDS level 3 challenge. Figure 8.1h shows the completion time of the RFDS level 3 challenge. Figure 8.1i shows the number of trials of each of the 37 participants for the RFDS level 3 challenge. Only 32 of the 37 participants successfully completed the RFDS level 3. This accounts for the blank bars in Figure 8.1g and Figure 8.1h, where 5 participants did not complete the level, and therefore had no score or completion time data to plot. Figure 8.1j shows participants' scores for the RFDS level 4 challenge. Figure 8.1k shows the completion time of the RFDS level 4 challenge. Figure 8.1l shows the number of trials of the participants for the RFDS level 4 challenge. The blank bars in Figure 8.1l are representative of the participants who failed to complete the previous RFDS level (level 3) and therefore did not attempt the RFDS level 4 and hence have no number of trial data to indicate. Only 11 participants successfully completed the RFDS level 4. The remaining 21 participants dropped out at this level. Figure 8.1m shows participants' scores for the RFDS level 5 challenge. Figure 8.1n shows the completion time of the RFDS level 5 challenge. Figure 8.1o shows the number of trials of the participants for the RFDS level 5 challenge. Only 5 participants successfully completed this level, as 6 participants dropped out.

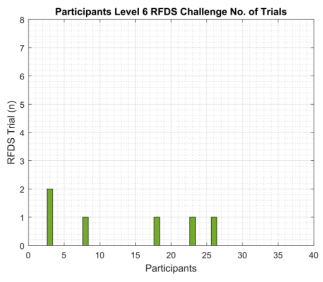
Figure 8.2 shows the results of the 37 participants' RFDS challenge performance for level 6 to 10. Figure 8.2a shows participants' scores for the RFDS level 6 challenge. Figure 8.2b shows the completion time of the RFDS level 6 challenge. Figure 8.2c shows the number of trials



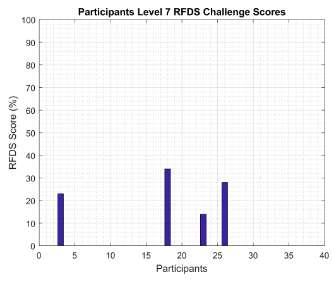
(a) Level 6 RFDS scores.



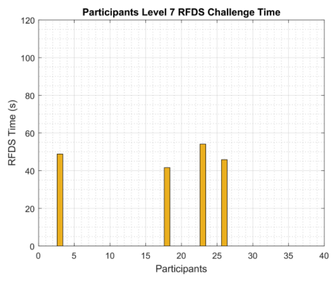
(b) Level 6 RFDS time.



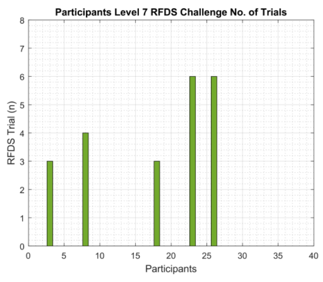
(c) Level 6 RFDS trials.



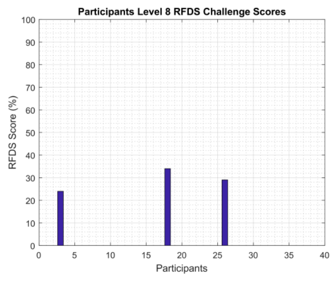
(d) Level 7 RFDS scores.



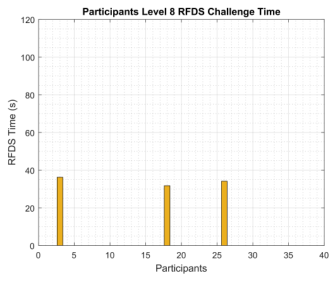
(e) Level 7 RFDS time.



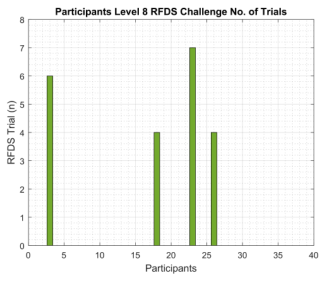
(f) Level 7 RFDS trials



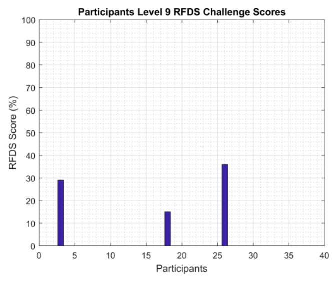
(g) Level 8 RFDS scores.



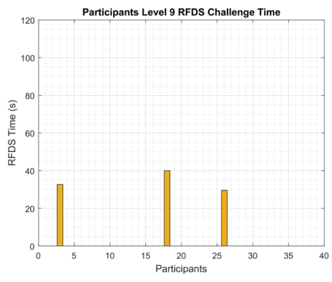
(h) Level 8 RFDS time.



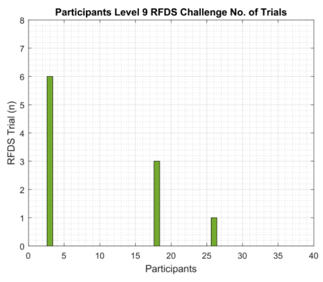
(i) Level 8 RFDS trials



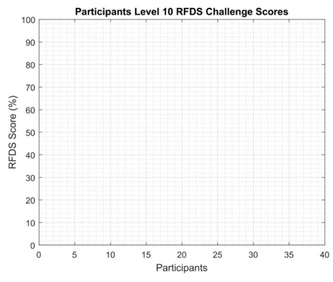
(j) Level 9 RFDS scores.



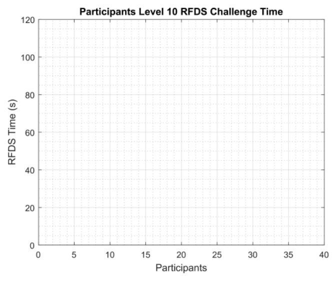
(k) Level 9 RFDS time.



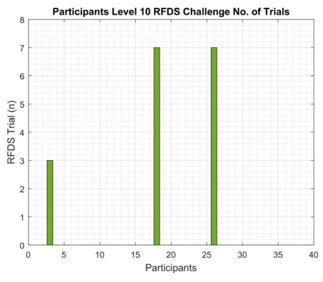
(l) Level 9 RFDS trials



(m) Level 10 RFDS scores.



(n) Level 10 RFDS time.



(o) Level 10 RFDS trials

Figure 8.2: RFDS Level 6 to 10 results.

of the participants for the RFDS level 6 challenge. All 5 remaining participants successfully completed this level. Figure 8.2d shows participants' scores for the RFDS level 7 challenge. Figure 8.2e shows the completion time of the RFDS level 7 challenge. Figure 8.2f shows the number of trials of the participants for the RFDS level 7 challenge. Only 4 of the remaining 5 participants successfully completed this level. Figure 8.2g shows participants' scores for the RFDS level 8 challenge. Figure 8.2h shows the completion time of the RFDS level 8 challenge. Figure 8.2i shows the number of trials of the participants for the RFDS level 8 challenge. Only 3 of the remaining 4 participants successfully completed this level. Figure 8.2j shows participants' scores for the RFDS level 9 challenge. Figure 8.2k shows the completion time of the RFDS level 9 challenge. Figure 8.2l shows the number of trials of the participants for the RFDS level 9 challenge. All 3 remaining participants successfully completed this level. Figure 8.2m shows participants' scores for the RFDS level 10 challenge. Figure 8.2n shows the completion time of the RFDS level 10 challenge. Figure 8.2o shows the number of trials of the participants for the RFDS level 10 challenge. None of the remaining participants successfully completed this level, it was the most difficult challenge, particularly due to the very short time constraint.

Figure 8.3 shows the average performance (score, completion time, and number of trials) for each level, for the participants who completed those levels. The purple bars shows the average score performance, the green bar shows the completion time performance, and the yellow bars shows the number of trials. Note that level 10 has no score nor completion time average due to the fact that none of the participants completed this, despite the attempted number of trials.

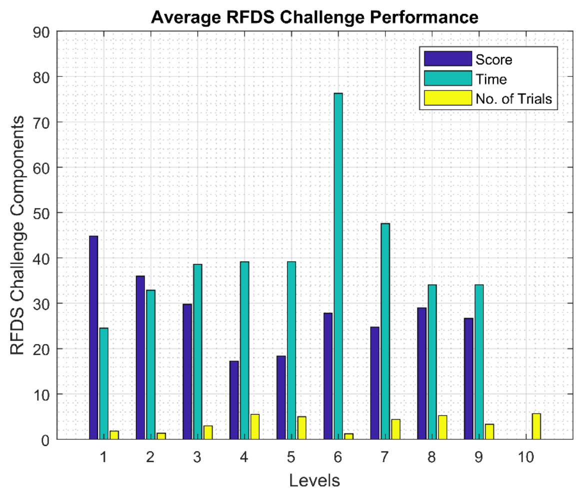
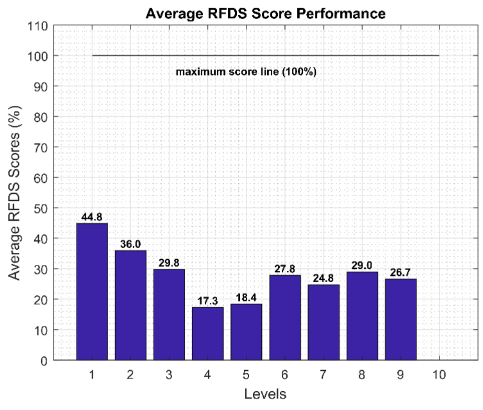
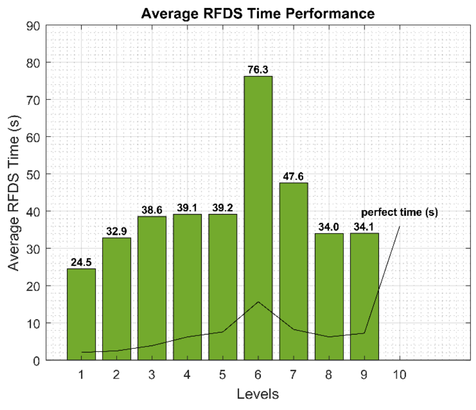


Figure 8.3: RFDS average performances.

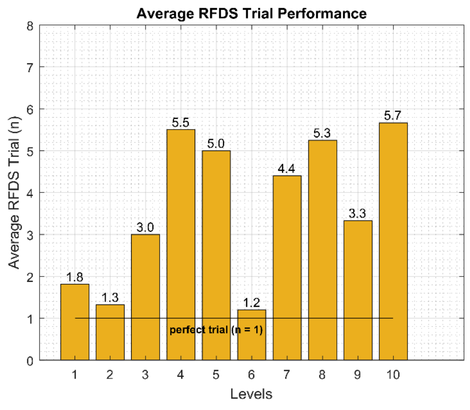
Figure 8.4a shows the average score data for each level. This was only computed for participants who completed each level. The average score values are printed on top of each bars in Figure 8.4a. The maximum score line (at 100%) is shown for comparison. A score of zero in any level indicates a failure to complete that level by any participants, as observed in level 10.



(a) RFDS average scores.



(b) RFDS average time.



(c) RFDS average trial.

Figure 8.4: RFDS average performances shown seperately.

Figure 8.4b shows the average completion time data for each level. This was also only computed for participants who completed each level. The average completion time values are printed on top of each bars in Figure 8.4b. The perfect time line shown in Figure 8.4b indicates the fastest time to beat in order to attain a perfect level score of 100%. This aids in the comparison of the average participant's performance with the maximum possible performance on the RFDS simulator. A completion time of zero in any level indicates a failure to complete that level by any participants, hence the lack of time data as observed in level 10.

Figure 8.4c shows the average number of trial data for each level. This was only computed for participants who attempted each level. This included level 10 which was attempted by three participants but not successfully completed, hence the presence of a plot for level 10 in this case, although Figure 8.4a and Figure 8.4b has no plot for level 10. The ideal number of trial to complete any level is one, and is indicated by the perfect trial line in Figure 8.4c, for the purpose of comparing with the average participant's performance.

8.1.2 Path_v02 performance - completion time and navigation accuracy

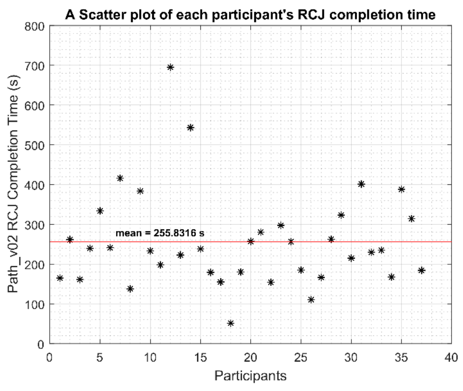
In this section, the results of the participants performance in controlling the navigation of the firefly multirotor UAV through the designed Path_v02 environment, is presented. A screen video clip capture of the UAV navigation along the flight Path_v02 environment, using each of the RCJ, KBD, and mSVG control interface, is included in this thesis CD-R as supplementary multimedia content. In order to determine the navigation accuracy of the multirotor UAV on the Path_v02 flight path for each of the control methods, the following rules were applied for consistency.

1. **-5%** : for each veering off of the UAV from the guided path or touching of the windows walls/frame.
2. **85% cap** : if the UAV glided on the navigation path instead of hovering with a clear separation from the flight path base layer.
3. **60% cap** : if the UAV veers off the elevated navigation area completely onto the ground.
4. **40% cap** : for incomplete navigation along the Path_v02 flight path i.e. did not finish.

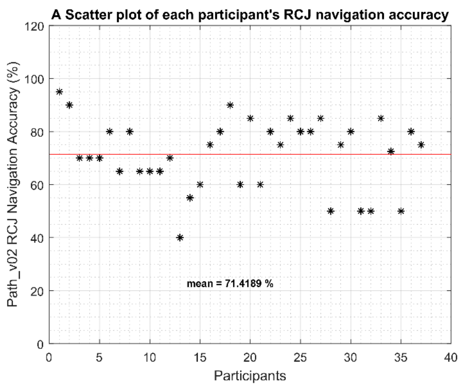
The Path_v02 performance results are presented in Figure 8.5. Figure 8.5a is a scatter plot of each participant's completion time when using the RCJ interface to complete the Path_v02 navigation task. The participants mean completion time is indicated by the red line $y = 255.8316\ s$. Figure 8.5c shows each participant's completion time performance for the KBD interface, with a mean completion time of $y = 62.9721\ s$. Figure 8.5e shows each participants completion time performance for the mSVG interface, with a mean completion time of $y = 179.0984\ s$.

From this result, it can be observed that for the same task, the completion time of the mSVG interface is much faster than the RCJ interface but slower than the KBD interface. In other words, the best time performing interface is KBD, followed by the mSVG, which is then followed by the RCJ, the slowest of the three interfaces. But the KBD interface emulates a special case of the RCJ interface with altitude, attitude, and position (AAP) assist, therefore, the completion time performance of the AAP-assisted RCJ was better (faster) than the mSVG which was equally

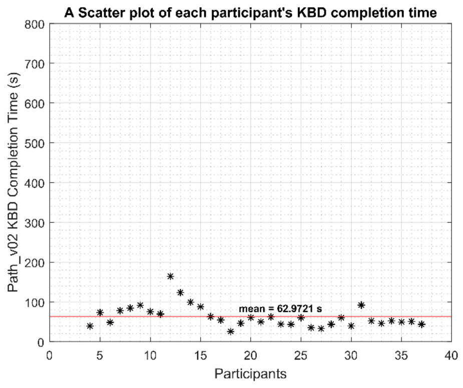
AAP assisted by default because the AAP is essential to its practical operation. In this study, the AAP-assist RCJ was found to complete tasks three times faster, hence perform three times better, than the AAP-assist mSVG interface in the designed nCA Tier 1-III task.



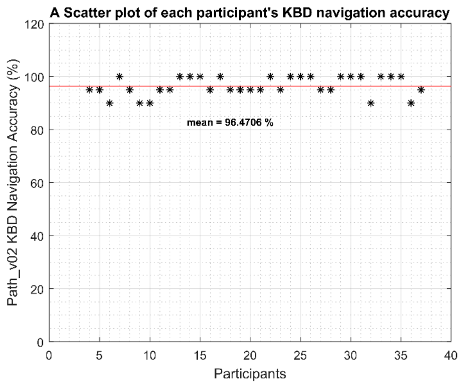
(a) RCJ completion time.



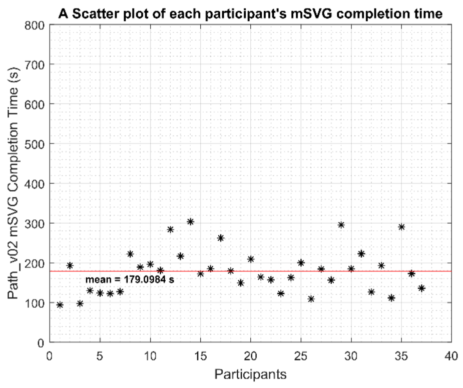
(b) RCJ navigation accuracy.



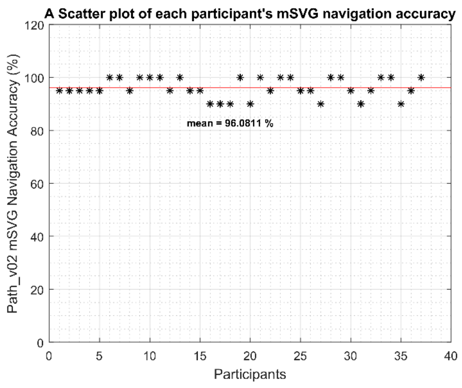
(c) KBD completion time.



(d) KBD navigation accuracy.



(e) mSVG completion time.



(f) mSVG navigation accuracy.

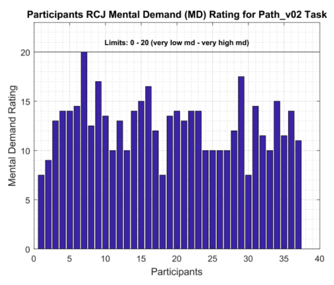
Figure 8.5: Path_v02 navigation task performance.

Figure 8.5b is a scatter plot of each participant's navigation accuracy when using the RCJ interface to complete the Path_v02 navigation task. The participants mean navigation accuracy is indicated by the red line $y = 71.42\%$. Figure 8.5d shows each participants navigation accuracy performance for the KBD interface, with a mean navigation accuracy of $y = 96.47\%$. Figure 8.5f shows each participants navigation accuracy performance for the mSVG interface, with a mean navigation accuracy of $y = 96.08\%$. From this result, it can be observed that the navigation accuracy of the KBD and mSVG is much better than the RCJ, and that the navigational accuracy of the KBD is similar to the mSVG. But since the KBD emulates a special case of the RCJ when hover assist and position hold is introduced, it can be concluded that the performance of the modified RCJ with navigation assist is similar to the mSVG which already has the navigation assist mode by default, since this is required for it practical operation. Therefore, there is no difference in the navigational accuracy between the RCJ and mSVG interface for an nCA Tier 1-III task where altitude and position assist is implemented.

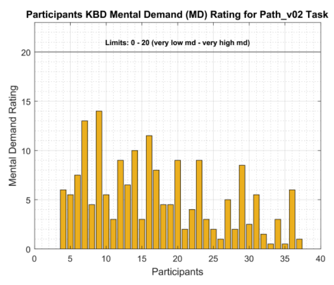
8.1.3 Cognitive workload - Nasa TLX

This section presents the results of the NASA TLX survey questionnaire used to estimate the cognitive workload of the RCJ, KBD, and mSVG interfaces in the Path_v02 flight navigation task. The six NASA TLX components - mental demand (md), physical demand (pd), temporal demand (td), performance (p), effort (e), and frustration (f) - each asks a question for which the participant gives a rating from 0 - 20 on a linear scale with low values representing positive/desirable experiences and high values representing negative/undesirable experiences. Note that the KBD interface TLX ratings for participants 1-3 are not presented because these data were not available, as the KBD interface was not part of the preliminary test, but was introduced afterwards, for the significant part of the experiment, as a modified version of the RCJ with altitude, attitude, and position assist features, for a better like-to-like comparison with the mSVG interface.

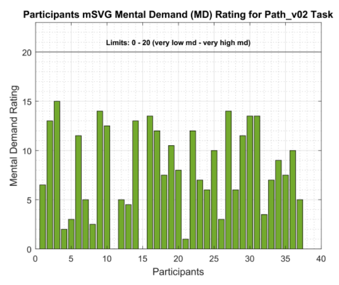
Figure 8.6 presents the results of the TLX survey for the first three components - mental demand (md), physical demand (pd), and temporal demand (td) - for the RCJ, KBD, and mSVG components. Figure 8.6a shows the 37 participants rating of how mentally demanding the task was, when performed using the RCJ interface. Figure 8.6b shows the mental demand component results for the KBD interface. And Figure 8.6c shows the mental demand TLX component results for the mSVG interface. The RCJ interface had the higher rating followed by the mSVG, and then the KBD interface. Figure 8.6d shows the 37 participants rating of how physically demanding the task was, when performed using the RCJ interface. Figure 8.6e shows the physical demand component results for the KBD interface. And Figure 8.6f shows the physical demand TLX component results for the mSVG interface. Although the rating for the physical demand component was generally lower than the mental demand component rating, the trend observed was similar with the RCJ interface being rated higher followed by the mSVG interface and then the KBD interface. Figure 8.6g shows the 37 participants rating of how temporally demanding the task was, when performed using the RCJ interface. By temporal demand, it is meant that how quickly the participant had to react in response to the control interface action on task, in other words the reaction time pressure on participants. Figure 8.6h shows the temporal demand component results for the KBD interface. And Figure 8.6i shows the temporal demand TLX component results for the mSVG interface. Again, the RCJ interface



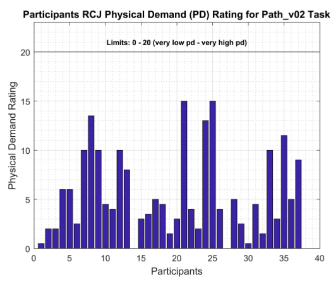
(a) RCJ mental demand rating.



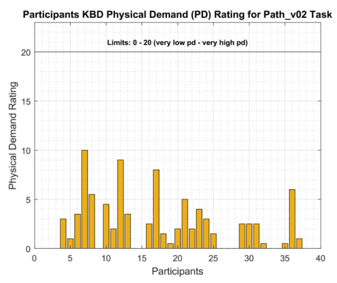
(b) KBD mental demand rating.



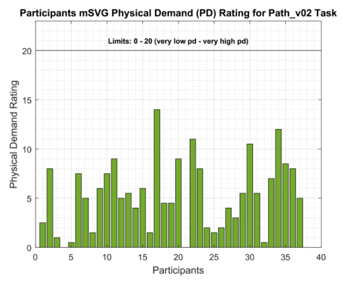
(c) mSVG mental demand rating.



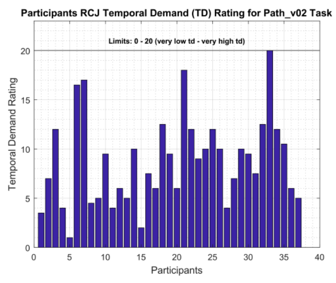
(d) RCJ physical demand rating.



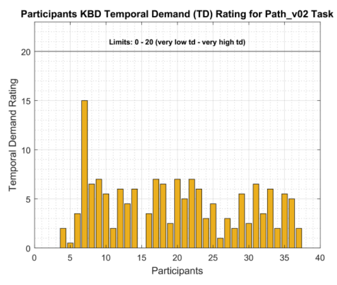
(e) KBD physical demand rating.



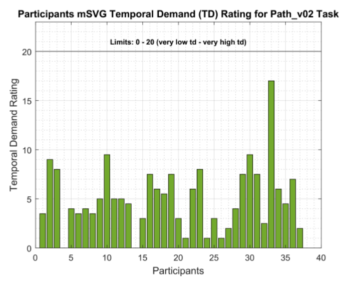
(f) mSVG physical demand rating.



(g) RCJ temporal demand rating.



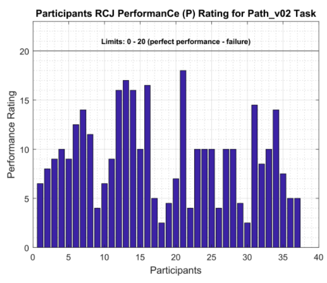
(h) KBD temporal demand rating.



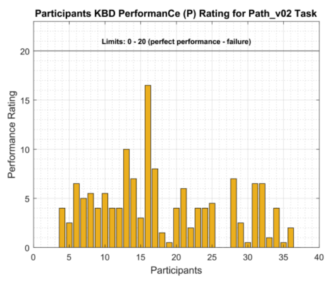
(i) mSVG temporal demand rating.

Figure 8.6: NASA TLX survey questionnaire results for mental demand, physical demand and temporal demand.

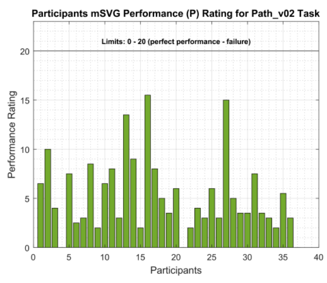
was rated to be more temporally demanding than the KBD and mSVG interfaces.



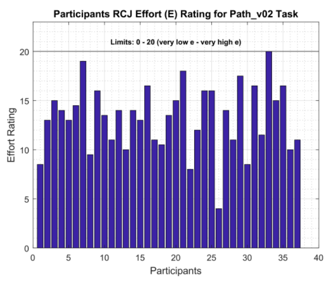
(a) RCJ performance rating.



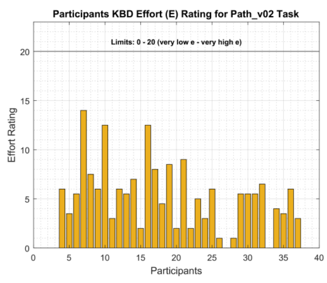
(b) KBD performance rating.



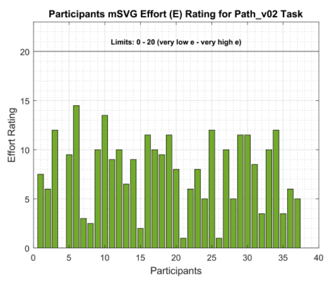
(c) mSVG performance rating.



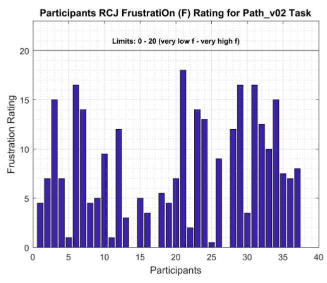
(d) RCJ effort rating.



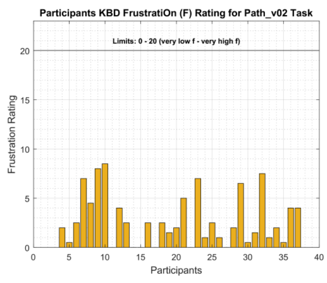
(e) KBD effort rating.



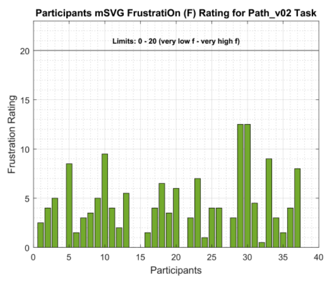
(f) mSVG effort rating.



(g) RCJ frustration rating.



(h) KBD frustration rating.



(i) mSVG frustration rating.

Figure 8.7: NASA TLX survey questionnaire results for performance, effort and frustration.

Figure 8.7 presents the results of the TLX survey for the remaining three components - performance (p), effort (e), and frustration (f) - for the RCJ, KBD, and mSVG interfaces. Figure 8.7a shows the 37 participants' rating of their performance on task, when using the RCJ interface. Figure 8.7b shows the performance component results for the KBD interface. And Figure 8.7c shows the performance TLX component results for the mSVG interface. The participants rated their performance with the KBD and mSVG interfaces better than their performance with the RCJ interface. Figure 8.7d shows the 37 participants rating of how much effort they invested in order to achieve their perfomance level in the navigation task, when using the RCJ interface. Figure 8.7e shows the effort component results for the KBD interface. And Figure 8.7f shows the effort TLX component results for the mSVG interface. From the effort rating results, the participants clearly invested the most effort on the RCJ interface, followed by the mSVG interface, and then the KBD interface which required the least effort. Figure 8.7g shows the 37 participants rating of how frustrated they felt in using the RCJ interface to achieve the navigation task. Figure 8.7h shows the frustration component results for the KBD interface. And Figure 8.7i shows the frustration TLX component results for the mSVG interface. From the frustration level ratings, the participants generally found the RCJ interface more frustrating to use, followed by the mSVG interface, and then the KBD interface which was the least frustrating to use.

In order to compare the cognitive workload for each of the three interfaces, a TLX distribution was first generated for each of the interfaces, and then compared. This distribution was the average of the 37 participants' rating for each TLX survey component, per interface, and was computed using:

$$\left[\frac{\sum_{i=1}^{n} y_{md,i}}{n}, \frac{\sum_{i=1}^{n} y_{pd,i}}{n}, \frac{\sum_{i=1}^{n} y_{td,i}}{n}, \frac{\sum_{i=1}^{n} y_{P,i}}{n}, \frac{\sum_{i=1}^{n} y_{e,i}}{n}, \frac{\sum_{i=1}^{n} y_{f,i}}{n}\right]$$

$$= \left[\bar{y}_{md}, \quad \bar{y}_{pd}, \quad \bar{y}_{td}, \quad \bar{y}_{p}, \quad \bar{y}_{e}, \quad \bar{y}_{f}\right] = TLX \tag{8.1}$$

Where $1 \leq i \leq n$ is a pointer to the corresponding components rating for each of the n participants. $n = 37$ for RCJ and mSVG interfaces, and $n = 34$ for the KBD interface because of the non-available participants 1-3 TLX data for the KBD interface.

In addition to comparing the TLX distribution of each of the control interfaces, the non-weighted mean of each interface's TLX distribution can also be compared. This is computed using:

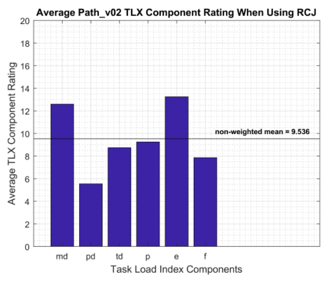
$$Mean = \frac{TLX}{n} = \frac{\bar{y}_{md} + \bar{y}_{pd} + \bar{y}_{td} + \bar{y}_{p} + \bar{y}_{e} + \bar{y}_{f}}{n} \tag{8.2}$$

Where $\bar{y}_{md}, \bar{y}_{pd}, \bar{y}_{td}, \bar{y}_{p}, \bar{y}_{e},$ and $\bar{y}_{f}$ represents the average rating of the corresponding TLX components, and $n = 6$ is the number of TLX components.

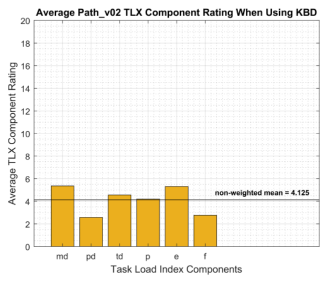
The results for each of the interfaces is presented in Figure 8.8. Figure 8.8a shows the participants average rating for each TLX component for the RCJ interface. The non-weighted mean of the RCJ TLX distribution was computed as:

$$Mean_{rcj} = \frac{TLX_{rcj}}{n_{rcj}} = \frac{\bar{y}_{md} + \bar{y}_{pd} + \bar{y}_{td} + \bar{y}_{p} + \bar{y}_{e} + \bar{y}_{f}}{n}$$

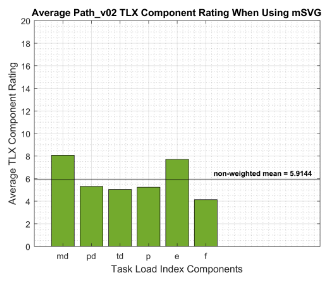
$$= \frac{12.5946 + 5.5405 + 8.7432 + 9.2432 + 13.2432 + 7.8514}{6}$$



(a) Average RCJ TLX component rating.



(b) Average KBD TLX component rating.



(c) Average mSVG TLX component rating.

Figure 8.8: The mean of the 37 participants' TLX rating for each of the TLX component with the non-weighted average of the distribution indicated.

$$= \frac{57.2162}{6} = 9.5360 \tag{8.3}$$

The non-weighted mean of the RCJ TLX distribution is indicated by the line $y = 9.5360$ in Figure 8.8a. Figure 8.8b shows the participants average rating for each TLX component for the KBD interface. The non-weighted mean of the KBD TLX distribution was:

$$Mean_{kbd} = \frac{\bar{y}_{md} + \bar{y}_{pd} + \bar{y}_{td} + \bar{y}_{p} + \bar{y}_{e} + \bar{y}_{f}}{n} = 4.1250$$

The non-weighted mean of the KBD TLX distribution is indicated by the line $y = 4.1250$ in Figure 8.8b. Figure 8.8c shows the participants average rating for each TLX component for the mSVG interface. The non-weighted mean of the mSVG TLX distribution was:

$$Mean_{mSVG} = \frac{\bar{y}_{md} + \bar{y}_{pd} + \bar{y}_{td} + \bar{y}_{p} + \bar{y}_{e} + \bar{y}_{f}}{n} = 5.9144$$

The non-weighted mean of the mSVG TLX distribution is indicated by the line $y = 5.9144$ in Figure 8.8c. From the non-weighted mean results, the least cognitive demanding interface was the KBD interface (cognitive workload ≈ 4). This was followed by the mSVG interface (cognitive workload ≈ 6), which was about 2/3 more demanding than the KBD interface and also about 3/5 times less demanding than the RCJ interface. The RCJ interface (cognitive workload ≈ 10) had the most cognitive workload, being about 2.5 times more demanding than the RCJ interface.

Figure 8.9 provides a side-by-side control interface comparison of the average TLX result per index component. It can be observed from the participants' average rating in Figure 8.9 that the RCJ interface was consistently rated the highest (most negative/undesirable), while the KBD interface was consistently rated the lowest (most positive/desirable), whereas the mSVG interface was consistently in the middle. While the average participants results suggested an order preference of KBD, then mSVG, and then RCJ, which is supported by the ratings of a majority of the participants, there exist a number of participants with different order of preference as indicated in their ratings. For example, participant 32 rated the mSVG interface effort better (lower effort requirement) than the KBD interface, and participant 27 rated the RCJ interface mental demand better (less mentally demanding) than the mSVG interface.

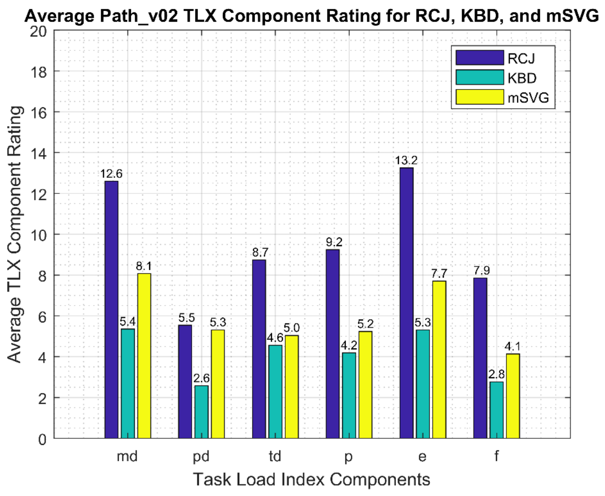


Figure 8.9: Side by side comparison of the RCJ, KBD, and mSVG cognitive workload.

The implication of the cognitive workload result presented in this section is that although the RCJ interface is cognitively more demanding than the mSVG interface, the altitude, attitude, and position assisted RCJ interface (the KBD interface) affords a lower cognitive workload/demand for an nCA Tier 1-III level UAV. However, the mSVG interface cognitive workload rating may be better than the AAP assisted RCJ at higher nCA autonomy levels, because of the intuitive and natural control properties of the mSVG over the RCJ. This is suggested as a future research topic.

8.1.4 Speech Gesture Ratio (SGR)

This section presents the result of the analysis of the speech-gesture usage ratio for the Path_v02 navigation task. In order to do this, the Path_v02 navigation flight path was divided into 24 navigation segments, the navigation of which could either be by speech or by gesture. The participants' speech-gesture usage dataset for the Path_v02 navigation task using mSVG is included in the results Tables in Appendix B.3 for ease of referencing.

Figure 8.10 shows the usage distribution of speech and gesture for each of the 24 Path_v02 navigation segments. The blue bars represent the speech component for each navigation segment. The yellow bars, which are stacked on the blue bars, represent the gesture components. Each stacked bar gives visual representation of the distribution of speech and gesture for each of the 24 navigation segments. From Figure 8.10, it can be observed that navigation stages 1 'hover', 14 'climb up', 18 'pan starboard', 21 'climb down', and 24 'land' were speech only navigation stages. This was because these stages were only implemented as speech commands and had no gesture command alternative implemented. The speech-gesture usage ratio based on the navigation stages was estimated as,

$$SGR_{NavStage} = 13.9167 : 23.0833 \tag{8.4}$$

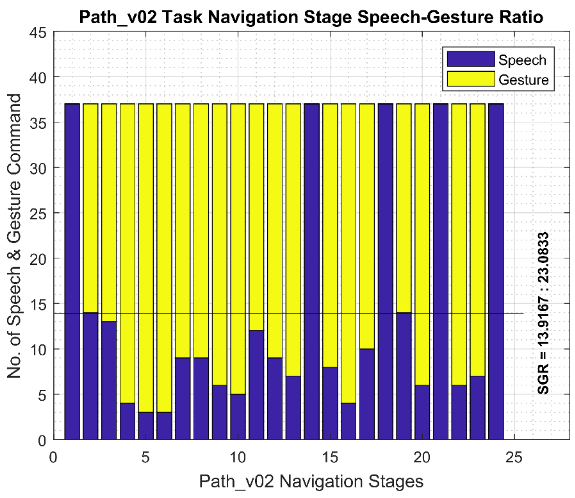


Figure 8.10: Speech Gesture Ratio (SGR) from navigation stage speech-gesture control distribution

Figure 8.11 shows the speech-gesture usage ratio, in the Path_v02 navigation task, for each of the 37 participants. Similar to Figure 8.10, blue bars represent speech, and yellow bars represent gestures, and the stacked combination gives a visual ratio of the speech-gesture distribution for each participant. From this result, it can be observed that participant 1, 16, 17, 21, 22, 31, and 37 used the minimum number of speech and maximum number of gesture possible (SGR 5:19) in completing the mSVG navigation task. Note that there are 5 navigation commands instances where gesture was not implemented and control could only be issued via speech. It can also be observed that participant 4 and 23 used the most amount of speech command and the least amount of gesture (SGR 18:4) in completing the mSVG navigation stage. Although, the relationship between the SGR ratio and the completion time is inconclusive from this result, it was considered that higher gesture ratios could lead to faster task completion time, since the gesture control commands were being captured and processed 3 times faster than the speech control commands in this research's mSVG implementation. The speech-gesture ratio based on the participant's usage was estimated as,

$$SGR_{Ptcpt} = 9.027 : 14.973 \tag{8.5}$$

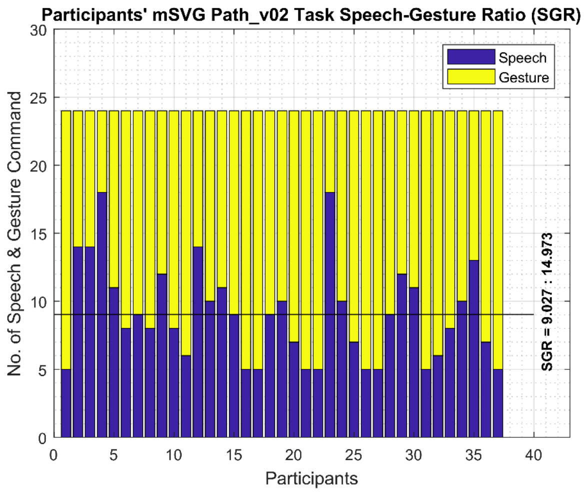


Figure 8.11: Speech Gesture Ratio (SGR) from participants speech-gesture usage distribution

In order to compare the two SGR ratios estimated from the analysis of the navigation stages and the participants usage, both ratios were normalized,

$$nSGR_{NavStage} = \frac{13.9167}{37} : \frac{23.0833}{37} \approx 0.38 : 0.62 \tag{8.6}$$

Similarly,

$$nSGR_{Ptcpt} = \frac{9.027}{24} : \frac{14.973}{24} \approx 0.38 : 0.62 \tag{8.7}$$

From Equation 8.6 and Equation 8.7, the speech gesture ratio was estimated as 0.32 : 0.62, for the developed mSVG interface, based on its usage for the Path_v02 navigation task. Also, Equation 8.6 and Equation 8.7 shows that the SGR ratio was the same regardless of how the data was analysed - either per participant or per navigation stage basis.

8.1.5 Statistical significance of the results

From the Path_v02 navigation task performance result for each of the three interfaces, it was observed that the mSVG interface performed better than the RCJ interface, and the KBD interface was better than the mSVG interface. However, how statistically significant was this result? Therefore, in order to determine how significant the results were, and hence the validity of the conclusion, the analysis of variance (ANOVA) statistical test was performed on the results. The first ANOVA test was conducted on the completion time result, and the second ANOVA test was performed on the Path_v02 navigation accuracy. A normal approximation was assumed, sample size $n > 30$ Ross (2014), the results were drawn from independent events, the variance of which were homogeneous.

Completion Time

First we define the null hypothesis which states that the mean completion time of the RCJ ($\mu_{rcj} = \mu_1$), KBD ($\mu_{kbd} = \mu_2$), and mSVG ($\mu_{msvg} = \mu_3$) are similar, therefore there is no difference between any of the interfaces, their completion time performances are the same or very similar. This is stated as

$$H_0 : \mu_1 = \mu_2 = \mu_3 \tag{8.8}$$

The alternative hypothesis argues that at least one of the means differs from the others significantly enough to consider some interface performance better than some others. This is stated as

$$H_1 : \text{not all the means are equal} \tag{8.9}$$

Assuming a confidence interval of 95% is required, then a significance level of 0.05 is implied and defined as

$$\alpha = 0.05 \tag{8.10}$$

Given that the sum of squares within (Error) is

$$SS_w = \sum_{i=1}^{m} \sum_{j=1}^{n} (X_{ij} - \bar{X}_i)^2 \tag{8.11}$$

Where

$i = 1, 2, 3, \ldots, m$ is the number of independent samples/columns

$j = 1, 2, 3, \ldots, n$ is the ith sample size

$X_{ij} \sim N(\mu_i, \sigma^2)$ is a random variable with unknown mean μ_i and unknown variance σ^2

$\bar{X}_i$ is the average of the ith sample

Given that the sum of squares between (Groups) is

$$SS_b = n\sum_{i=1}^{m}(\bar{X}_i - \bar{\bar{X}})^2 \tag{8.12}$$

Where $\bar{\bar{X}}$ is the average of all the data values, given as

$$\bar{\bar{X}} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij}}{mn} = \frac{\sum_{i=1}^{m}\bar{X}_i}{m} \tag{8.13}$$

But due to the unevenness of the sample size in each sample column, the standard formula is broken down into its constituent parts, in order to accurately represent the current data being analysed by this study.

Therefore, the sum of squares within is given as

$$SS_w = \sum_{j=1}^{n_{rcj}}(X_{rcj,j} - \bar{X}_{rcj})^2 + \sum_{j=1}^{n_{kbd}}(X_{kbd,j} - \bar{X}_{kbd})^2 + \sum_{j=1}^{n_{msvg}}(X_{msvg,j} - \bar{X}_{msvg})^2$$

$$= 670429.1892 \tag{8.14}$$

And the sum of squares between is given as

$$SS_b = n_{rcj}(\bar{X}_{rcj} - \bar{\bar{X}}) + n_{kbd}(\bar{X}_{kbd} - \bar{\bar{X}}) + n_{msvg}(\bar{X}_{msvg} - \bar{\bar{X}})$$

$$= 664965.4309 \tag{8.15}$$

The mean square sum within samples is given as

$$MS_w = \frac{SS_w}{df_w} = \frac{670429.1892}{105} = 6385.0399 \tag{8.16}$$

Where df_w is the degree of freedom within the sample columns, given as

$$df_w = (n_{rcj} + n_{kbd} + n_{msvg}) - 1 \tag{8.17}$$

Similarly, the mean square sum between samples is given by

$$MS_b = \frac{SS_b}{df_b} = \frac{664965.4309}{2} = 332482.7155 \tag{8.18}$$

Where df_b is the degree of freedom between the sample columns, given as

$$df_b = m - 1 \tag{8.19}$$

Therefore the test statistic F value needed to reject or validate the null hypothesis H_0 is given as

$$F = \frac{MS_b}{MS_w} = \frac{332482.7155}{6385.0399} = 52.0721 \tag{8.20}$$

If the value of $F \leq F_{critical}$, where $F_{critical}$ is retrieved from the F-distribution table using the df_w, df_b and α values, the null hypothesis H_0 is accepted. However, if $F > F_{critical}$, the null hypothesis H_0 is rejected, and the alternative hypothesis H_1 is validated. Given that (with the aid of a four-figure table)

$$F_{critical} = 3.0915 \tag{8.21}$$

Therefore, since $F > F_{critical}$, the null hypothesis H_0, which proposed that there was no difference between the completion time performances between the three interfaces, is rejected. And the alternative hypothesis H_1, which proposes a significant difference between the completion time performances of each of the three interfaces, is validated. The ANOVA analysis for the Path_v02 completion time results is as summarised in Table 8.1.

Table 8.1: ANOVA Table - RCJ, KBD, and mSVG completion time result analysis.

Sources	SS	df	MS	F	Prob > F
Groups	664965.4309	2	332482.7155	52.0721	1.9455×10^{-16}
Error	670429.1892	105	6385.0399		
Total	1335394.6201	107			

Figure 8.12 is a MATLAB boxplot from the one-way ANOVA analysis of the RCJ, KBD, and mSVG interface completion time result. From the boxplot, it can be observed that the mean completion time of the RCJ is higher than the mSVG, and that the KBD interface had the least completion time. Also, the outlier data are indicated by the red cross, as observed on the RCJ ($\sim$ 700 and $\sim$ 540) and the KBD ($\sim$ 160 and $\sim$ 120) boxplots. The mSVG data had no distinct outliers.

Navigation Accuracy

For the statistical significance analysis of the navigation accuracy result, the null hypothesis states that the mean navigation accuracy of the RCJ ($\mu_{rcj} = \mu_1$), KBD ($\mu_{kbd} = \mu_2$), and mSVG ($\mu_{msvg} = \mu_3$) are similar, in other words, there is no difference between any of the interfaces' navigation accuracy performance, they are the same or very similar. This is stated as

$$H_0 : \mu_1 = \mu_2 = \mu_3 \tag{8.22}$$

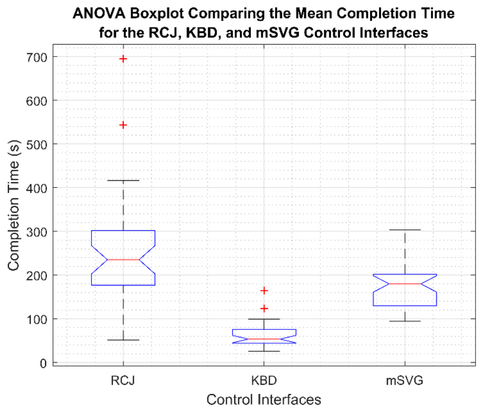


Figure 8.12: ANOVA Boxplot - RCJ, KBD, and mSVG completion time result analysis.

The alternative hypothesis argues that at least one of the means differ from the others significantly enough to consider some interface performance better than some others. This is stated as

$$H_1 : \text{not all the means are equal} \tag{8.23}$$

Assuming a confidence interval of 95% is required, then a significance level of 0.05 is implied and defined as

$$\alpha = 0.05 \tag{8.24}$$

The sum of squares within is given as

$$SS_w = \sum_{j=1}^{n_{rcj}} (X_{rcj,j} - \bar{X}_{rcj})^2 + \sum_{j=1}^{n_{kbd}} (X_{kbd,j} - \bar{X}_{kbd})^2 + \sum_{j=1}^{n_{msvg}} (X_{msvg,j} - \bar{X}_{msvg})^2$$

$$= 7214.9841 \tag{8.25}$$

The sum of squares between is given as

$$SS_b = n_{rcj}(\bar{X}_{rcj} - \bar{\bar{X}}) + n_{kbd}(\bar{X}_{kbd} - \bar{\bar{X}}) + n_{msvg}(\bar{X}_{msvg} - \bar{\bar{X}})$$

$$= 15021.7636 \tag{8.26}$$

The mean square sum within samples is given as

$$MS_w = \frac{SS_w}{df_w} = \frac{7214.9841}{105} = 68.7141 \tag{8.27}$$

Similarly, the mean square sum between samples is given by

$$MS_b = \frac{SS_b}{df_b} = \frac{15021.7636}{2} = 7510.8818 \tag{8.28}$$

Therefore the test statistic F value needed to reject or validate the null hypothesis H_0 is given as

$$F = \frac{MS_b}{MS_w} = \frac{7510.8818}{68.7141} = 109.3062 \tag{8.29}$$

Given that $F_{critical} = 3.0915$, therefore $F > F_{critical}$, again, the null hypothesis H_0, which proposed that there was no difference between the navigation accuracy performances between the three interfaces, is rejected. And the alternative hypothesis H_1 is validated. The ANOVA analysis for the Path_v02 navigation accuracy results is as summarised in Table 8.2.

Table 8.2: ANOVA Table - RCJ, KBD, and mSVG navigation accuracy result analysis.

Sources	SS	df	MS	F	Prob > F
Groups	15021.7636	2	7510.8818	109.3062	2.1683×10^{-26}
Error	7214.9841	105	68.7141		
Total	22236.7477	107			

Figure 8.13 is a MATLAB boxplot from the one-way ANOVA analysis of the RCJ, KBD, and mSVG interface navigation accuracy result. From the boxplot, it can be observed that the accuracy for the RCJ interface was lower than the KBD and mSVG interfaces, and that the KBD and mSVG interface had very similar navigation accuracy performance.

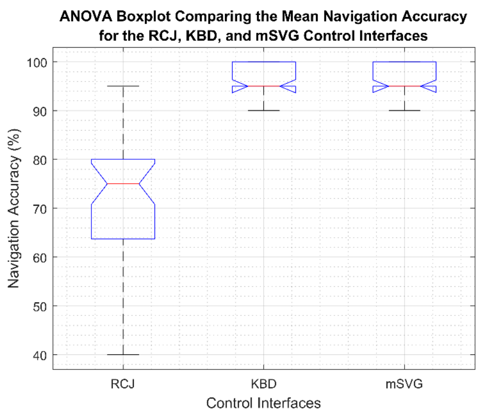


Figure 8.13: ANOVA Boxplot - RCJ, KBD, and mSVG navigation accuracy result analysis.

Two-sample t-test analysis for KBD and mSVG interfaces navigation accuracy

From the navigation accuracy ANOVA analysis, it was clear that the RCJ is significantly different from the KBD and mSVG interfaces. This was clearly observed in the one way ANOVA boxplot of Figure 8.13. However, the navigational accuracy performance of the KBD and mSVG interfaces seemed quite similar and indistinguishable in the MATLAB ANOVA boxplot in Figure 8.13. The variances were similar. Therefore, an unpaired two-sample t-test with equal variance was conducted to confirm if KBD and mSVG interface performance were indeed similar.

The null hypothesis states that the mean navigation accuracy of the KBD (μ_{kbd}) and mSVG (μ_{msvg}) interfaces are similar, in other words, there is no difference between their navigation accuracy performance, they are the same or very similar. This is stated as

$$H_0 : \mu_{kbd} - \mu_{msvg} = 0 \tag{8.30}$$

The alternative hypothesis simply argues that their navigation performances are not similar.

$$H_1 : \mu_{kbd} - \mu_{msvg} \neq 0 \tag{8.31}$$

Assuming a confidence interval of 95% is required, then a significance level of 0.05 is implied and defined as

$$\alpha = 0.05 \tag{8.32}$$

Sum of squares for the KBD interface is given as

$$SS_{kbd} = \sum_{i=1}^{n_{kbd}} (X_{kbd,i} - \bar{X}_{kbd})^2 = 426.4706 \tag{8.33}$$

Where $\bar{X}_{kbd} = 96.4706$ is the mean, $n_{kbd} = 34$ is the sample size, and $1 \leq i \leq n_{kbd}$ points to the elements in the sample.

Sum of squares for the mSVG interface is given as

$$SS_{msvg} = \sum_{i=1}^{n_{msvg}} (X_{msvg,i} - \bar{X}_{msvg})^2 = 506.7568 \tag{8.34}$$

Where $\bar{X}_{msvg} = 96.0811$ is the mean, $n_{msvg} = 37$ is the sample size, and $1 \leq i \leq n_{msvg}$ points to the elements in the sample.

The pooled variance is given as

$$S_p^2 = \frac{SS_{kbd} + SS_{msvg}}{df_{kbd} + df_{msvg}} = 13.5250 \tag{8.35}$$

Where the degrees of freedom, $df_{kbd} = n_{kbd} - 1 = 33$ and $df_{msvg} = n_{msvg} - 1 = 36$. The standard error of the mean is given as

$$SE_{\bar{X}_{kbd} - \bar{X}_{msvg}} = \sqrt{\frac{S_p^2}{n_{kbd}} + \frac{S_p^2}{n_{msvg}}} = 0.8737 \tag{8.36}$$

The test statistic t value

$$t = \frac{\bar{X}_{kbd} - \bar{X}_{msvg}}{SE_{\bar{X}_{kbd} - \bar{X}_{msvg}}} = 0.4458 \tag{8.37}$$

The effect size is determined using Cohen's d as

$$d = \frac{\bar{X}_{kbd} - \bar{X}_{msvg}}{\sqrt{S_p^2}} = 0.1059 \tag{8.38}$$

Since $d < 0.2$, the effect size is considered small. By using $df = df_{kbd} + df_{msvg} = 69$, $\alpha = 0.05$, and a two-tail t-table from a four-figure table, the critical t value is determined as

$$t_{critical} = 1.9997 \tag{8.39}$$

Therefore, since $t < t_{critical}$, the null hypothesis H_0, which proposed that there was no significant difference between the navigation accuracy performance of the KBD and the mSVG interface, is valid. By replacing the KBD interface with the altitude, attitude, and position (AAP) assisted RCJ interface, it can be concluded that: the navigation accuracy performance of the AAP-assisted RCJ interface ($M = 96.47, SD = 3.59$) is similar to the performance of the mSVG interface ($M = 96.08, SD = 3.75$) ($t(66) = 2.00), p < 0.05, \text{Cohen's } d = 0.11$).

8.2 Discussion

8.2.1 RC skill classification

The RFDS performance result showed that 30% of the study population were skilled and experienced RC UAV pilots. The criteria being the ability to successfully complete Level 1 - 4 of the RFDS challenge. It can also be observed from the participants demographic that 30% of the population had a previous UAV flying experience of more than 10 hours. In addition to this, only 30% of the participants have been flying for more than 24 months. From this analysis, it is therefore proposed that: *the flying hours, the number of months flying, and the RFDS Level 4 challenge performance, can be used to estimate the skill and experience level of a* *remote controlled UAV pilot*. For example, a classification threshold of 10 flying hours, over a 24 months period, and the successful completion of the RFDS Level 4 challenge, could be used as a skill benchmark for remote controlled UAV pilots performing typical navigation control task with a small UAV, in a non-congested area, carrying a small payload such as a camera, medical supply, etc.

8.2.2 RCJ vs mSVG performance comparison

From the result of the performance analysis in Section 8.1.2, later verified by the statistical analysis in Section 8.1.5, it was observed that the mSVG control interface performed better than the RCJ UAV control interface, completing the Path_v02 flight path 30% faster than the RCJ interface and 25% more accurately than the RCJ interface. Clearly, the mSVG is a better interface in this context.

However, by endowing the RCJ interface with some of the special features of the mSVG interface, which included altitude, attitude, and position (AAP) assist, the results were interpreted differently. The KBD interface was used to emulate this special case of AAP-assisted RCJ interface. From the KBD interface performance results, it was observed that the AAP-assisted RCJ interface was three times faster than the mSVG interface for completing the Path_v02 flight path, and the AAP-assisted RCJ interface navigation accuracy matched the mSVG interface navigation accuracy. In this context, the AAP-assisted RCJ interface is a better interface than the mSVG interface.

In order to understand the implication of this two-way result, the navigational control autonomy (nCA) model presented in Chapter 5 was used. The nCA model consists of six nCA levels divided into three tiers in a pyramidal structure. The RC joystick controller usability was constrained to the bottom tier levels of rate control mode (nCA Tier 1-I), attitude stabilised control mode (nCA Tier 1-II), and attitude position and heading assist control mode (nCA Tier 1-III). The mSVG interface was more suitable for tier two and tier three applications such as waypoint navigation (nCA Tier 2-I), autonomous navigation (nCA Tier 2-II), and full autonomy mode (nCA Tier 3-I). The Path_v02 flight path was an nCA Tier 1-III task. Therefore, the AAP-assist RCJ was expected to operate effectively at this level, as was observed in the results. However, at higher nCA autonomy levels, the mSVG interface is recommended.

8.2.3 Human factors - cognitive workload and usability

From the result of the cognitive workload analysis, it was observed that the RCJ interface was cognitively more demanding than the mSVG interface. However, similar to the performance analysis, the AAP-assisted RCJ was also found to be less cognitively demanding than the mSVG interface, for the nCA Tier 1-III Path_v02 navigation task. But the mSVG interface is considered to be natural, intuitive, easy to learn, and does not require the use of a tangible device. These soft qualities are considered more valuable at higher UAV autonomy level applications, such as the Alps scenario described in Section 1.4.1.

8.2.4 The bias in mSVG speech-gesture ratios

From the SGR ratio result and analysis, the usage ratio of speech to gesture for the mSVG interface was found to be 0.38 : 0.62. But it was then clarified that the mSVG implementation had constrained the minimum speech threshold to a ratio of 5 : 19, because of the 5 speech command with no gesture command equivalent, requiring all participants to use speech only for such navigational manoeuvre. The implication of this is that the SGR ratios upper and lower thresholds could be intentionally or unintentionally designed into the mSVG implementation to skew the SGR outcome, such as showing a balanced or an unbalanced speech-gesture usability ratio. Biased SGR may result from the need to constrain speech or gesture component in order to eliminate noise, to regulate control speed, or for a quicker deployment of the mSVG interface as in the case of this study.

The ideal/unbiased mSVG implementation is one where all commands are available as both speech and gesture, are formed with equal ease, and are processed at an equal rate. But this may be impractical because equal rate of processing could be an elusive target, due to the inherent nature of the speech and gesture components. Speech consists of a string of utterances captured over a time lapse period, whereas gesture can be processed from a single-instant-image capture, often resulting in different processing rates.

However, the SGR result shows that there was an overall preference for gesture usage over speech. This was due to the difference in their processing rate, as the gestures were being processed three times faster than the alternative speech command. Perhaps without the minimum speech threshold of 5 : 19 of the developed mSVG interface, the ratio for gesture usage may have been much higher.

8.3 Chapter Conclusion

This chapter presented the result of the experiment study comparing the performance and cognitive workload of the RC joystick and mSVG UAV control interfaces. The RFDS test was used to estimate the UAV flying skill level of the participants. It was shown that the flying hours, the number of months flying, and the RFDS Level 4 challenge performance was a good estimator for this. A two-way result was obtained in the comparison of the RCJ and mSVG interfaces. While the mSVG interface was considered better than the standard RCJ interface, the AAP-assisted RCJ interface was often found to be better or at least as effective as the mSVG interface. Therefore, the conclusion of which interface performed better or which interface required more cognitive workload was found to be context dependent. It was also shown that for an mSVG interface, gesture is likely to be the predominantly used mode.

8.3.1 Further work

In this study, the research was constrained to laboratory experiments with a ROS Gazebo simulated UAV. Therefore this research could be advanced further by coupling the mSVG interface onto a real UAV and by performing a field test to demonstrate the practical effectiveness of the mSVG interface in a real world UAV application.

In addition to this, it was considered that at higher nCA levels, the mSVG interface should perform better than the AAP-RCJ interface. Therefore, an nCA Tier 2-I task could be designed to test this hypothesis in a future study. Also, the relationship between the SGR ratio and the task completion time for the mSVG interface could be investigated further using an improved mSVG implementation.

Chapter 9

Conclusion

This chapter provides an overarching summary of this research's contribution by providing a chapter-by-chapter summary, highlighting the major research findings/contributions, presenting the research limitations, and making suggestions for further research works. The main objective of this research was to review current HCI interfaces that can be used for the control of small multirotor UAVs and to propose an effective alternative interface for the small multirotor UAV that could perform better than the standard RC joystick controller, by designing and developing the alternative interface and comparing the performance of the alternative interface with the standard RC joystick (RCJ) controller. The literature review in Chapter 2 reviewed the HCI interfaces, aiding in the proposition of the multimodal speech and visual gesture interface (mSVG) proposed in Chapter 1, which was supported by the Alps search and rescue scenario and the domestic robot assistant scenario. The mSVG interface was designed and developed in Chapter 6 and the comparison was conducted in Chapter 8 using methods described in Chapter 4.

From the research questions, it was found that the mSVG interface was an effective alternative to the RCJ interface, although its effectiveness is constrained by the application context and environment. The effective application limit of the speech component of the mSVG implementation was found to be 75 dB; beyond which the speech recognition deteriorates below 65% becoming unreliable for practical control application. Also, gestures were found to be clearly recognisable for 10 Lux (twilight conditions) and higher Lux levels on distinct backgrounds. The mSVG interface was found to perform better than the RCJ interface at higher navigation control autonomy (nCA) application levels. The mSVG interface was found to be 1 minute faster and 25% more accurate than the RCJ interface. Also, the RCJ interface was found to be 1.4 times more demanding than the mSVG interface.

9.1 Chapter-by-Chapter Summary

Chapter 1 begins by introducing this research work and placing it within the context of current and evolving UAV applications. Some research problems were presented from which this research's aim and objectives were formulated. An alpine search and rescue application scenario was developed as a reference application for this research. The concept of the aerobot was presented. The research questions and hypothesis were clearly defined. Additional application areas were highlighted.

Due to the expanded nature of the literature review, this was split into two chapters. The first literature review chapter, Chapter 2, was more descriptive and it discusses HCIs for small multirotor UAVs under various unimodal and multimodal themes with their limitations. It was quite clear that there exists a plethora of HCI control interfaces for the control of small multirotor UAVs. Although each of these interfaces were also known to have their limitations for any given application. It identified the multimodal combination of speech and visual gesture as being an intangible, intuitive, natural, and HHI-like interaction technique that was easy to learn and easy to use. It highlighted the fact that humans tend to combine verbal and non-verbal gestures in their HHI communication, the techniques of which were considered adaptable and useful in human robot interaction. Other researchers working at different layers/levels of aerial robotics multimodal speech and gesture interaction, were also identified. These researchers included Ng and Sharlin (2011), Harris and Barber (2014), Nagi et al. (2014), Cauchard et al. (2015), Hill et al. (2015), Cacace et al. (2016), Fernandez et al. (2016), Barber et al. (2016), Kattoju et al. (2016), Obaid et al. (2016), Schelle and Stutz (2016), Gubcsi and Zsedrovits (2018), Schelle and Stütz (2018), and Cauchard et al. (2019).

The second literature review chapter, Chapter 3, was more theoritical and focused on the mathematics behind the speech and gesture processing techniques used in this research. In this chapter, the discrete Markov model was discussed as a first order Markov process explained using an ergodic model. The hidden Markov model was described as a doubly embedded stochastic model in which the underlying stochastic processes is not directly observable, although it can be observed through another set of stochastic process that produces the second observation. For speech recognition using the hidden Markov model, stochastic models from known utterances were first created, and the probability that an unknown utterance was generated by each stochastic model, were then evaluated. Examples of practical HMM-based automatic speech recognition systems included the CMU Pocket Sphinx (CMU Sphinx, 2009), HTK (Young et al., 2006; Gales and Young, 2007), and Julius (Lee et al., 2001). The Haar cascade object detection technique was discussed as a method that uses a machine learning approach for training cascade functions, which were then used in object detection. The convex hull defect for finger gesture recognition was also discussed.

Chapter 4 discusses the research method used and the rationale for selecting the method and components. The interface development section discussed the keyboard interface (as a symbol generator), the speech capture and processing operation, visual gesture recognition, and RC joystick selection. Three hardware platforms were discussed, the ImmersionRC XuGong, iQuad Jade, and Unicorn multirotor platforms. Two hardware in the loop simulation were also discussed, the RealFlight Drone Simulator and the RotorS ROS Gazebo simulator. The method used in this research work was to conduct an experiment study. The details of the experiment design were presented and discussed. The experiment procedure was enumerated. The participant recruitment method and the demography of the recruited participants were discussed. The method considered for the analysis of the experiment result was also presented. It also described the NASA TLX survey tool used for cognitive workload anlysis. The chapter concluded by comparing this research's method with other researchers in this area.

In Chapter 5, the limitations of the existing autonomy models (such as the AFRL and ONR), in aerobot navigational control autonomy classification, led to the development of a new autonomy classification model called the navigational control autonomy (nCA) model. The nCA model was used to identify a control void beyond the tier-one components of the nCA model.

In Chapter 6, the multimodal speech and visual gesture system model was designed and developed with the aid of the nCA Tier 1-III navigation and nCA Tier 2-I scenario application examples. In developing the model, mathematical notations and logic were used. These were then converted into computer programs, from which some useful code snippets were presented. The mSVG development was discussed in functional blocks of speech input unit, gesture input unit, multimodal control processing unit, and ROS Gazebo simulator unit. The operations of the multmodal control processing unit was simulated using both MATLAB and Python-ROS-Gazebo.

In Chapter 7, the result of the experiment study on the effects of varying noise levels and varying lighting levels on speech and gesture control command interfaces for aerobots, was presented, analysed, and discussed. It was observed from the results of the experiment that speech recognition accuracy/success rate falls as noise levels rises. A regression model was developed from the empirical observation of nearly $3,108$ speech command utterances by 37 participants. Speech, based on the custom CMU Sphinx ASR system developed, is not an effective means of control interaction with an aerobot beyond 75 dB noise level, as the speech control interaction becomes very unreliable due to poor recognition. From the results of the gesture-lighting experiment, the effects of both lighting conditions and the environment background on the quality of gesture recognition, was almost insignificant, less than 0.5%, which means that other factors such as the gesture capture system design and technology (camera and computer hardware), type of gesture being captured (upper body, whole body, hand, fingers, or facial gestures), and the image processing technique (gesture classification algorithms), are more important in developing a successful gesture recognition system.

In chapter 8, the result of the experiment study comparing the performance and cognitive workload of the RC joystick and mSVG UAV control interfaces, were presented, anlysed, and discussed. It was shown that the flying hours, the number of months flying, and the RFDS Level 4 challenge performance was a good estimator for UAV RC flying skill level. A two-way result was obtained in the comparison of the RCJ and mSVG interfaces. Even though the mSVG interface performed better than the standard RCJ interface, the AAP-assisted RCJ interface was better or at least as effective as the mSVG interface. It was concluded that the application context would determine what interface is the most effective for any given scenario.

9.2 Major Research Contributions

The major contribution of this research are highlighted as follows:

1. The development of the nCA model discussed in Chapter 5 (Abioye et al., 2017).
2. The design and development of a functional mSVG system as presented in Chapter 6 (Abioye et al., 2018b).
3. The development of an understanding of the limitations of the mSVG technique in terms of noise level effects on both speech and gesture within the context of an aerobot type application, and its implication for future research (Chapter 7) (Abioye et al., 2019a, 2018a).
4. Performing a comparison between the mSVG interface with the standard RCJ interface in terms of performance and human factors – Chapter 8 (Abioye et al., 2019b).
5. Unimodal aerial robot HCI classification into software-agents computing devices, electromechanical, vision, bioelectrical, and speech in Chapter 2 (Abioye et al., 2017).

9.3 Research Limitations

The main limitations of this research work are highlighted below:

1. The main limitation of the varying lighting experiment study in this research was that the lighting level range was constrained between twilight (10 Lux) conditions and an overcast day condition (1400 Lux), typical for indoor lighting levels, whereas outdoor lighting conditions could vary up to 100,000 Lux on a very sunny day with direct sunlight, as shown in Table 4.6. Therefore, the gesture-lighting application described in this study is more applicable to indoor scenarios such as the domestic application scenario described in Section 1.4.2. The gesture-lighting application for the outdoor search and rescue scenario described in Section 1.4.1 is limited to cloudy overcast day conditions.

2. From the SGR ratio result and analysis, the usage ratio of speech to gesture for the mSVG interface was found to be 0.38 : 0.62. But it was then clarified that the mSVG implementation had constrained the minimum speech threshold to a ratio of 5 : 19, because of the 5 speech command with no gesture command equivalent, requiring all participants to use speech only for such navigational manoeuvre, as explained in Section 8.2.4. But the SGR result shows that there was an overall preference for gesture usage over speech, which agrees with the conclusion that for the developed mSVG system, the gesture component was preferred, even though the result of the SGR ratio in favour of gesture may have been under-reported due to the mSVG implementation constraint.

9.4 Further Research

Further research work suggested from this research are as highlighted below:

1. The problem of contradictory commands as presented in Section 2.8.6, could potentially be an area for further research. For example, what happens when two unimodal interfaces receive contradictory commands, or when two contradictory actions result from individual interpretation by the unimodal interfaces of the same control command repeated for different modes? What should be the conflict resolution model for such cases - avoid or address the control conflict?

2. The development of a natural multimodal human-like speech and gesture communication method for UAV is a challenging task. One of which could be, how the UAV determines (and learn to ignore) gesture or speech communications that are not directed towards it. Acting on such un-directed communications could potentially lead to accidents hurting the operator, damaging the UAV, or destroying other objects in the environment. An operator was demonstrating natural gesture interaction with the aid of the Kinect in Waibel (2011), when he switched his attention to the audience, and mistakenly directed a gesture at the audience while the UAV was still flying and active. The UAV responded

to this gesture, which was not meant for it, and it nearly hurt the demonstrator. Another obvious challenge is how to eliminate ambient noise for the speech interface, both that due to the multirotor propulsion system and that due to other environment variables. In addition to these is this projects goal of achieving on-board capture and processing of the multimodal control elements (speech and visual gesture) through single board computers. This frees the operator from wearing, holding, or handling any control device.

3. User acceptance: depending on the adoption rate of technologies in this area and users expectation of new technologies, or the prevailing minimum functionality threshold. While this research has focused on determining it's functionality level, there is the question of how close this is to the minimum expectations of stakeholders in this space, for the adoption of this technique.

4. The continued development of the custom UAV developed in this research, and the integration of the mSVG interface as deployed in the HITL ROS Gazebo simulation experiments, can be continued to the realisation of a fully practical operational mSVG controlled multirotor aerobot.

5. In this study, the research was constrained to laboratory experiments with a ROS Gazebo simulated UAV. The next phase of this research could couple the mSVG interface onto a real UAV and a field test could be conducted to demonstrate the practical effectiveness of the mSVG interface in real world UAV applications.

6. In addition to this, it was considered that at higher nCA levels, the mSVG interface should perform better than the AAP-RCJ interface. Therefore, an nCA Tier 2-I task could be designed to test this hypothesis in a future study. Also, the relation between the SGR ratio and the task completion time for the mSVG interface could be conducted alongside this.

7. In order to improve the application limit of the aerobot speech interface, alternative automatic speech recognisers with different learning model from the CMU Sphinx's hidden markov model, could be considered next. ASR's based on artificial neural network or deep learning are of particular interests, such as the IBM Watson speech to text cloud AI service IBM (2010). An hardware upgrade to directional, noise canceling, and array microphones could be considered. Multi and single word speech command selection could be optimised in favour of the more resilient command variant as observed in this particularly study.

8. In order to improve the gesture performance, better alternative hardware (cameras and single board computers) capable of capturing high resolution images and performing complex graphic computation processing in real time, could be used. In addition, instead of

the appearance-based 2D-model algorithm use in this paper, a more advance 3D-model based AI gesture algorithm could be developed for capturing and recognising hand, arm, and upper body gestures.

References

Abioye, A. O., Prior, S. D., Saddington, P., and Ramchurn, S. D. (2019a). 'Effects of Varying Noise Levels and Lighting Levels on Multimodal Speech and Visual Gesture Interaction with Aerobots'. *Applied Sciences* **9**(10):1–29.

Abioye, A. O., Prior, S. D., Thomas, G. T., and Saddington, P. (2016). 'The Multimodal Edge of Human Aerobotic Interaction'. In K. Blashki & Y. Xiao (eds.), *International Conferences Interfaces and Human Computer Interaction*, pp. 243–248, Madeira, Portugal. IADIS Press.

Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P., and Ramchurn, S. D. (2017). 'Multimodal Human Aerobotic Interaction'. In T. Issa, P. Kommers, T. Issa, P. Isaías, & T. B. Issa (eds.), *Smart Technology Applications in Business Environments*, chap. 3, pp. 39–62. IGI Global, Pennsylvania, USA.

Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P., and Ramchurn, S. D. (2018a). 'Quantifying the effects of varying light-visibility and noise-sound levels in practical multimodal speech and visual gesture (mSVG) interaction with aerobots'. In Meen, Prior, & Lam (eds.), *2018 IEEE International Conference on Applied System Invention (ICASI)*, pp. 842–845, Chiba, Tokyo, Japan. IEEE.

Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P., and Ramchurn, S. D. (2018b). 'The Multimodal Speech and Visual Gesture (mSVG) Control Model for a Practical Patrol, Search, and Rescue Aerobot'. In M. Giuliani, T. Assaf, & M. E. Giannaccini (eds.), *Towards Autonomous Robotic Systems - Part of the Lecture Notes in Computer Science book series (LNCS, volume 10965)*, vol. 10965, pp. 423–437, Bristol, UK. Springer International Publishing.

Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P., and Ramchurn, S. D. (2019b). 'The performance and cognitive workload analysis of a multimodal speech and visual gesture (mSVG) UAV control interface – submitted'. *IEEE Access Journal* **xx**(xx):1–19.

Abou-Khalil, B., and Misulis, K. E. (2006). *Atlas of EEG & Seizure Semiology*. Butterworth-Heinemann/Elsevier, Michigan, 1st edn.

Aeryon Labs Inc. (2011). 'Whitepaper - Intuitive Control of a Micro UAV'. https://aeryon.com/whitepaper/ituitivecontrol. First Available: 2011-02-07; Accessed: 2016-01-22.

Amazon (2016). 'Alexa Voice Service (AVS) API'. https://developer.amazon.com/alexa

Amazon (2016). 'First Prime Air Delivery'. https://www.amazon.com/b?node=8037720011. Available: 2016-12-07; Accessed: 2017-02-20.

Amazon (2016). 'First Prime Air Delivery'. https://www.amazon.com/b?node=8037720011

Amazon UK (2016). 'Great Planes RealFlight GPMZ4800 RealFlight Drone with Interlink Elite Mode 2 Edition'. https://www.amazon.co.uk/Great-Planes-RealFlight-GPMZ4800-Interlink/dp/B01724HVD0. Available: 2016-06-01; Accessed: 2017-01-03.

Anand, S. S., and Mathiyazaghan, R. (2016). 'Design and Fabrication of Voice Controlled Unmanned Aerial Vehicle'. *Journal of Aeronautics & Aerospace Engineering* **5**(2):1–5.

Anderson, J. (2015). *Cognitive Psychology and Its Implications*. Worth Publishers, New York, 8th edn.

Ang, A. M. S., Zhang, Z. G., Hung, Y. S., and Mak, J. N. F. (2015). 'A user-friendly wearable single-channel EOG-based human-computer interface for cursor control'. In *7th International IEEE/EMBS Conference on Neural Engineering, NER 2015, April 22, 2015 - April 24, 2015*, vol. 2015-July, pp. 565–568, Montpellier, France. IEEE Computer Society.

Anupama, H. S., Cauvery, N. K., and Lingaraju, G. M. (2012). 'Brain Computer Interface and Its Types - A Study'. *International Journal of Advances in Engineering & Technology* **3**(2):739–745.

Anusuya, M., and Katti, S. (2009). 'Speech recognition by machine: A review'. *International Journal of Computer Science and Information Security* **6**:181–205.

Apple (2015). '42mm Space Grey Aluminium Case with Black Sport Band'. http://www.apple.com/uk/shop/buy-watch/apple-watch-sport/42mm-space-grey-aluminium-case-black-sport-band?product=MJ3T2B/A{&}step=detail

ArduPilot (2012). 'Archived: APM 2.5 and 2.6 Overview'. http://ardupilot.org/copter/docs/common-apm25-and-26-overview.html. Available: 2012-08-22; Accessed: 2016-08-08.

Ascending Technologies (2013). 'AscTec Firefly'. http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-firefly/. Available: 2013; Accessed: 2018-08-19.

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. (2001). 'Recent advances in augmented reality'. *IEEE Computer Graphics and Applications* **21**(6):34–47.

Barber, D. J., Howard, T. M., and Walter, M. R. (2016). 'A multimodal interface for real-time soldier-robot teaming' **9837**:98370M.

Bastianelli, E., Nardi, D., Aiello, L. C., Giacomelli, F., and Manes, N. (2015). 'Speaky for robots: the development of vocal interfaces for robotic applications'. *Applied Intelligence* .

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). 'A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains'. *The Annals of Mathematical Statistics* **41**(1):164–171.

Bischoff, R., and Graefe, V. (2002). 'Dependable multimodal communication and interaction with robotic assistants'. In *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 300–305.

Blanz, V. (2007). 'A learning-based high-level human computer interface for face modeling and animation'. In *20th International Joint Conference on Artificial Intelligence, IJCAI 2007 - Workshop on Artifical Intelligence for Human Computing, January 6, 2007 - January 6, 2007*, vol. 4451 LNAI, pp. 296–315, Hyderabad, India. Springer Verlag.

Bolt, R. a. (1980). 'Put-that-there: Voice and Gesture at the Graphics Interface'. *Proceedings of the 7th annual conference on Computer graphics and interactive techniques - SIGGRAPH '80* pp. 262–270.

Borkowski, A., and Siemiatkowska, B. (2010). 'Towards semantic navigation in mobile robotics'. *Graph Transformations and Model-Driven Engineering* pp. 719–748.

Brooks, F. P., Ouh-Young, M., Batter, J. J., and Kilpatrick, P. J. (1990). 'Project GROPE Haptic displays for scientific visualization'. *ACM SIGGRAPH Computer Graphics* **24**(4):177–185.

Brooks, F. P. J. (1977). 'The Computer 'Scientist' as Toolsmith: Studies in Interactive Computer Graphics'. *Information Processing* **Internatio**(August):pp. 625—-634.

Cacace, J., Finzi, A., and Lippiello, V. (2016). 'Multimodal Interaction with Multiple Co-located Drones in Search and Rescue Missions'. *CoRR* **abs/1605.0**:1–6.

Calella, J. C., Ortega, F. R., Rishe, N., Bernal, J. F., and Barreto, A. (2016). 'HandMagic: Towards User Interaction with Inertial Measuring Units'. In *IEEE Sensors 2016*, pp. 1–3, Orlando, FL, USA. IEEE.

Cauchard, J. R., Jane, L. E., Zhai, K. Y., and Landay, J. A. (2015). 'Drone & me: an exploration into natural human-drone interaction'. *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* pp. 361–365.

Cauchard, J. R., Tamkin, A., Wang, C. Y., Vink, L., Park, M., Fang, T., and Landay, J. A. (2019). 'Drone.io: A Gestural and Visual Interface for Human-Drone Interaction'. *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* pp. 153–162.

Cavett, D., Coker, M., Jimenez, R., and Yaacoubi, B. (2007). 'Human-computer interface for control of unmanned aerial vehicles'. In *2007 IEEE Systems and Information Engineering Design Symposium, SIEDS, April 27, 2007 - April 27, 2007*, Charlottesville, VA, United states. Inst. of Elec. and Elec. Eng. Computer Society.

Chao, G. (2009). 'Human-computer interaction: Process and principles of human-computer interface design'. In *2009 International Conference on Computer and Automation Engineering, ICCAE 2009, March 8, 2009 - March 10, 2009*, pp. 230–233, Bangkok, Thailand. Inst. of Elec. and Elec. Eng. Computer Society.

Cirillo, A., Ficuciello, F., Natale, C., Pirozzi, S., and Villani, L. (2016). 'A Conformable Force/-Tactile Skin for Physical Human-Robot Interaction'. *Robotics and Automation Letters, IEEE* **1**(1):41–48.

Cirque du Soleil, ETH Zurich, and Verity Studios (2014). 'SPARKED: A Live Interaction Between Humans and Quadcopters'. https://www.youtube.com/watch?v=6C8OJsHfmpI. Available: 2014-09-22; Accessed: 2016-02-22.

Clough, B. T. (2002). 'Metrics, schmetrics! How the heck do you determine a UAV's autonomy anyway?'. In *Proceedings of the 2002 Performance Metrics for Intelligent Systems Workshop*, no. 990, pp. 313–319, Gaithersburg, MD.

CMU Sphinx (2009). 'CMU Sphinx Project'. http://cmusphinx.sourceforge.net/. Available: 2009; Accessed: 2017-03-20.

Coelho, J., and Verbeek, F. (2014). 'Pointing Task Evaluation of Leap Motion Controller in 3D Virtual Environment'. In *Creating the Difference, Proceedings of the Chi Sparks 2014 Conference*, pp. 78–85.

Collman, R. A. (2014). 'Is this too Noisy (or perhaps too Quiet)?'. In *CIBSE London*, p. 96, London, UK. CIBSE London.

Costanza, E., Kunz, A., and Fjeld, M. (2009). 'Mixed reality: A survey'. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **5440 LNCS**:47–68.

Cyber Glove (2009). 'CyberGlove II Wireless Glove'. http://www.cyberglovesystems.com/cyberglove-ii/

De Souza E Silva, A., and Sutko, D. M. (2009). *Digital Cityscapes: Merging Digital and Urban Playspaces*, vol. 1. Peter Lang Publishing Inc, New York, first edn.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). 'Maximum Likelihood from Incomplete Data via the EM Algorithm'. *Journal of the Royal Statistical Society. Series B* **39**(1):1–38.

Dhawan, A., and Honrao, V. (2013). 'Implementation of Hand Detection based Techniques for Human Computer Interaction'. *International Journal of Computer Applications* **72**(17):975–8887.

DJI (2015a). 'DJI - Phantom X Concept'. https://www.youtube.com/watch?v=ec1EF2UaQ4U

DJI (2015b). 'Phantom 3s Remote Controller'. http://www.dji.com/product/phantom-3-standard/remote-controller. First Available: 2015; Accessed: 2016-01-11.

DJI (2017a). 'Mavic Pro Platinum Specs'. https://www.dji.com/mavic-pro-platinum/info. Available: 2017-01-03; Accessed: 2019-03-14.

DJI (2017b). 'SPARK Specifications'. http://www.dji.com/spark/info#specs. Available: 2017-05-25; Accessed: 2017-06-08.

DJI (2017c). 'SPARK User Manual V1.0'. https://dl.djicdn.com/downloads/Spark/20170525/Spark+User+Manual+V1.0.pdf. Available: 2017-05-25; Accessed: 2017-06-08.

DJI, UAViators, Pix4D, Kathmandu Living Labs, and Smartisan (2015). 'DJI Stories - Crisis Mapping in Nepal'. https://www.youtube.com/watch?v=QXkcvzxBsrY. Available: 2015-11-04; Accessed: 2017-02-20.

Eichhorn, A. (2012). 'Finger precise hand gesture tracking'. http://remotepresence.org/?p=394

Electricwingman (2016). 'T-Drones Smart X Quadcopter - ARTF (A)'. https://www.electricwingman.com/tdrones-smartx-quadcopter-a-artf

Emotiv (2014). 'Wearables for your brain — EEG Emotiv Insight: Mind for research'. https://emotiv.com/

Fels, M., Bauer, R., and Gharabaghi, A. (2015). 'Predicting workload profiles of brain-robot interface and electromygraphic neurofeedback with cortical resting-state networks: Personal trait or task-specific challenge?'. *Journal of Neural Engineering* **12**(4).

Fernandez, R. A. S., Sanchez-lopez, J. L., Sampedro, C., Bavle, H., Molina, M., and Campoy, P. (2016). 'Natural User Interfaces for Human-Drone Multi-Modal Interaction'. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1013–1022, Arlington, VA USA. IEEE.

Fong, T., and Nourbakhsh, I. (2000). 'Interaction challenges in human-robot space exploration'. In *Proceedings of the Fourth International Conference and Exposition on Robotics for Challenging Situations and Environments*, no. January 2004, pp. 340–346.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). 'SVO: Fast semi-direct monocular visual odometry'. *Proceedings - IEEE International Conference on Robotics and Automation* pp. 15–22.

Freund, Y., and Schapire, R. E. (1997). 'A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting'. *Journal of Computer and System Sciences* **55**(1):119–139.

Furlan, R. (2016). 'The Future of Augmented Reality: Hololens - Microsoft's AR headset shines despite rough edges'. *IEEE Spectrum* (June):21.

Furrer, F., Burri, M., Achtelik, M., and Siegwart, R. (2016). 'RotorSA modular gazebo MAV simulator framework'. *Studies in Computational Intelligence* **625**(October):595–625.

Futaba RC (2014). 'Futaba 4GRS 4 Channel 2.4 GHz Computer Radio System'. http://www.futabarc.com/systems/futk4220-4grs/index.html

Gales, M., and Young, S. (2007). 'The Application of Hidden Markov Models in Speech Recognition'. *Foundations and Trends® in Signal Processing* **1**(3):195–304.

Ganapathyraju, S. (2013). 'Hand gesture recognition using convexity hull defects to control an industrial robot'. In *2013 3rd International Conference on Instrumentation Control and Automation (ICA)*, pp. 63–67, Bali, Indonesia. IEEE.

Gomez-Gil, J., San-Jose-Gonzalez, I., Nicolas-Alonso, L. F., and Alonso-Garcia, S. (2011). 'Steering a tractor by means of an EMG-based human-machine interface'. *Sensors (Basel, Switzerland)* **11**(7):7110–7126.

Goodluckbuy (2016). 'T-Drones Smart.X Type-A 4-Axis Quadcopter Frame with Landing Gear for FPV'. http://www.goodluckbuy.com/

t-drones-smart-x-type-a-4-axis-quadcopter-frame-with-landing-gear-for-fpv.html

Google (2014). 'Google Project Soli: Your Hands Are the Only Interface You Will Need'. https://atap.google.com/soli/

Green, S., Chen, X., Billinnghurst, M., and Chase, J. G. (2007). 'Human Robot Collaboration: an Augmented Reality Approach a Literature Review and Analysis'. *Mechatronics* **5**(1):1–10.

Grifantini, K. (2009). 'Open Source Data Glove'. http://www.technologyreview.com/article/414021/open-source-data-glove/

Griliches, Z. (1957). 'Hybrid Corn: An Exploration in the Economics of Technological Change'. *Econometrica* **25**(4):501–522.

Gubcsi, G., and Zsedrovits, T. (2018). 'Ergonomic Quadcopter Control Using The Leap Motion Controller'. *2018 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)* pp. 1–5.

Gupta, L., and Ma, S. (2001). 'Gesture-based interaction and communication: Automated classification of hand gesture contours'. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews* **31**(1):114–120.

Haas, E. C. (2007). 'Integrating Auditory Warnings with Tactile Cues in Multimodal Displays for Challenging Environments'. In *Proceedings of the 13th International Conference on Auditory Display (ICAD2007)*, pp. 126–130.

Haber, J., and Chung, J. (2016). 'Assessment of UAV Operator Workload in A Reconfigurable Multi-Touch Ground Control Station Environment'. *Journal of Unmanned Vehicle Systems* p. 21.

Hammacher Schlemmer (2015). 'The Mind Controlled UFO'. http://www.hammacher.com/Product/Default.aspx?sku=84249{&}promo=Toys-Games-Remote-Control-Toys{&}catid=247

Hardkernel (2015). 'Odroid-XU4 Product Detail'. http://www.hardkernel.com/main/products/prdt{_}info.php?g{_}code=G143452239825

Harris, J., and Barber, D. (2014). 'Speech and Gesture Interfaces for Squad Level Human Robot Teaming'. In R. E. Karlsen, D. W. Gage, C. M. Shoemaker, & G. R. Gerhart (eds.), *Unmanned Systems Technology Xvi*, vol. 9084. SPIE.

Hart, S. G. (2006). ‘NASA-task load index (NASA-TLX); 20 years later’. *Human Factors and Ergonomics Society Annual Meting* pp. 904–908.

Hart, S. G., and Staveland, L. E. (1988). ‘Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research’. *Advances in Psychology* **52**(C):139–183.

He, B., Baxter, B., Edelman, B. J., Cline, C. C., and Ye, W. W. (2015). ‘Noninvasive brain-computer interfaces based on sensorimotor rhythms’. *Proceedings of the IEEE* **103**(6):907–925.

Heater, B. (2017). ‘The DJI Spark is fun, but not the mainstream drone we were promised’. https://techcrunch.com/2017/06/01/the-dji-spark-is-fun-but-not-the-mainstream-drone-we-were-promised/. Available: 2017-06-08; Accessed: 2017-06-08.

Hernandez Jose, L., Kyriakopoulos, N., and Lindeman, R. (2002). ‘The AcceleGlove a Hole-Hand Input Device for Virtual Reality’. *ACM SIGGRAPH Conference Abstracts and Applications* p. 259.

Hewett, T. T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G., and Verplank, W. (1996). *ACM SIGCHI Curricula for Human-Computer Interaction.* Association for Computing Machinery, Inc., first edn.

Higuchi, K., Shimada, T., and Rekimoto, J. (2011). ‘Flying sports assistant: external visual imagery representation for sports training’. *Proceedings of the 2nd Augmented ...* pp. 1–4.

Hill, A. F., Cayzer, F., and Wilkinson, P. R. (2007). ‘Effective Operator Engagement with Variable Autonomy’. In *2nd SEAS DTC Technical Conference*, p. 7.

Hill, S. G., Barber, D., and Evans, A. W. (2015). ‘Achieving the Vision of Effective Soldier-Robot Teaming : Recent Work in Multimodal Communication’. *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts* pp. 177–178.

Hirsch, H. G., and Ehrlicher, C. (1995). ‘Noise estimation techniques for robust speech recognition’. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, pp. 153 –156.

Hobbico, Inc., Great Planes Model Mfg., and Knife Edge Software (2016). ‘RealFlight Drone Edition’. http://www.realflight.com/products/rfdrone/index.php?moreinfo=drones. Available: 2016-11-12; Accessed: 2017-09-28.

Hobbs, A., and Herwitz, S. R. (2005). ‘Human Factors in the Maintenance of Unmanned Aircraft’. *Small* **15**(100).

HobbyKing (2010). 'Turnigy 9X 9Ch Transmitter w/ Module and 8ch Receiver (Mode 2) (v2 Firmware)'. https://hobbyking.com/en_us/turnigy-9x-9ch-transmitter-w-module-8ch-receiver-mode-2-v2-firmware.html?___store=en_us. Available: 2006-09-01; Accessed: 2016-01-07.

HobbyKing, and Turnigy (2015). 'Turnigy TGY-i10 10ch 2.4GHz Digital Proportional RC System with Telemetry (Mode 2)'. https://hobbyking.com/en{_}us/turnigy-tgy-i10-10ch-2-4ghz-digital-proportional-rc-system-with-telemetry-mode-2.html

Horrigan, F. A., and ONR's Committee (2000). *Review of ONR's Uninhabited Combat Air Vehicles Program.* National Academy press, Washington, D.C.

Huang, C.-L., and Chung, C.-Y. (2004). 'A real-time model-based human motion tracking and analysis for human-computer interface systems'. *Eurasip Journal on Applied Signal Processing* **2004**(11):1648–1662.

Huhn, K., and Haewon, S. (2014). 'Evaluation of the safety and usability of touch gestures in operating in-vehicle information systems with visual occlusion'. *Applied Ergonomics* **45**(3):789–798.

Hutchins, S., Cosenzo, K. A., McDermott, P. L., Feng, T.-D., Barnes, M., and Gacy, M. (2009). 'An Investigation of the Tactile Communications Channel for Robotic Control'. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* **53**(4):182–186.

Iancovici, T. C., Osorio, S., and Bonie Rosario, J. (2011). 'Biofeedback in Virtual Reality applications and Gaming'. *University of Massachusetts Lowell. Introduction to Biosensors. Spring 2011* .

IBM (2010). 'Enterprise-ready AI for your industry'. https://www.ibm.com/watson/about. Available: 2019-02-08; Accessed: 2019-03-14.

IDS GmbH (2008). 'Programming Interfaces - Manuals - IDS Imaging Development Systems GmbH'. https://en.ids-imaging.com/manuals-ueye-api.html. Available: 2008; Accessed: 2017-02-15.

IDS GmbH (2012). 'IDS UI-1221LE USB 2.0 Global Shutter Camera'. https://en.ids-imaging.com/store/ui-1221le-rev-2.html. Available: 2012; Accessed: 2017-02-15.

IMechE PE (2015). 'Pilot's Brain Control Drone'. http://www.imeche.org/news/engineering/pilot's-brain-control-drone. Article in *IMechE Professional Engineering Magazine.* First Available: 2015-03-04; Accessed: 2016-01-14.

ImmersionRC (2014). ‘XuGong v2 Pro – Foldable FPV Quadcopter’. https://www.immersionrc.com/fpv-products/xugong-v2pro/. Available: 2014-12-01; Accessed: 2016-08-08.

International Federation of Clinical Neurophysiology (1999). ‘Recommendations for the Practice of Clinical Neurophysiology: Guidelines of the International Federation of Clinical Physiology’. In G. Deuschl & A. Eisen (eds.), *EEG Suppl. 52*. Elsevier Science B.V.

Islam, R., Kelly, S., and Stimpson, A. (2016). *Small UAV Noise Analysis Design of Experiment*. Master thesis, Duke University.

Johnson, C., and Shea, C. (2008). ‘The Hidden Human Factors in Unmanned Aerial Vehicles’. *Proceedings of the 26th International Conference on Systems Safety* (October):0–9721385.

Kaljurand, K., and Alumäe, T. (2012). ‘Controlled Natural Language in Speech Recognition Based User Interfaces’. In T. Kuhn & N. E. Fuchs (eds.), *Controlled Natural Language*, pp. 79–94, Zurich, Switzerland. Springer.

Kamen, G. (2004). ‘Electromyographic Kinesiology’. In D. G. E. Robertson, G. E. Caldwell, & J. Hamill (eds.), *Research Methods in Biomechanics*, chap. Eight, p. 309. Human Kinetics, Champaign, IL, second edn.

Kattoju, R. K., Barber, D. J., Abich, J., and Harris, J. (2016). ‘Technological evaluation of gesture and speech interfaces for enabling dismounted soldier-robot dialogue’ **9837**:98370N.

Kim, S., Seo, H., Choi, S., and Kim, H. J. (2016). ‘Vision-guided aerial manipulation using a multirotor with a robotic arm’. *IEEE/ASME Transactions on Mechatronics* **4435**(c):1–1.

Kipman, A. (2016). ‘A futuristic vision of the age of holograms’. In *TED Talk*, pp. 1–5, Vancouver, British Columbia, Canada. TED.

Lackey, S., Barber, D., Reinerman, L., Badler, N. I., and Hudson, I. (2011). ‘Defining Next-Generation Multi-Modal Communication in Human Robot Interaction’. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* **55**(1):461–464.

Lafleur, K., Cassady, K., Doud, A., Shades, K., Rogin, E., and He, B. (2013). ‘Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain-computer interface’. *Journal of Neural Engineering* **10**(4).

Lavars, N. (2016). ‘Thumbsteered drone leaves you with a free hand’. http://newatlas.com/shift-drone-thumb/46188/

Leap Motion Inc. (2010a). ‘Leap Motion Controller’. https://www.leapmotion.com/

Leap Motion Inc. (2010b). ‘Leap Motion Controller’. https://www.leapmotion.com/. Available: 2010; Accessed: 2016-01-24.

Lee, a., Kawahara, T., and Shikano, K. (2001). ‘Julius an Open Source Real-Time Large Vocabulary Recognition Engine’. *Eurospeech* pp. 1691–1694.

Lee, D. (2015). ‘Brain-controlled drone shown off by Tekever in Lisbon’. http://www.bbc.co.uk/news/technology-31584547. in *the British Broadcasting Corporation (BBC) Technology News.* First Available: 2015-02-25; Accessed: 2016-01-14.

Lee, T., Meng, H., Lo, W. K., and Ching, P. C. (2003). ‘The State of the Art in Human-computer Speech-based Interface Technologies’. *HKIE Transactions Hong Kong Institution of Engineers* **10**(4):50–61.

Levin, T. (2017). ‘How loud is your drone? - The drone noise test of P2, P3P, P4P, and I2’. https://www.wetalkuav.com/dji-drone-noise-test/. Available: 2017-02-18; Accessed: 2017-10-12.

Li, S. (2016). ‘Python speech to text with PocketSphinx’. http://blog.justsophie.com/python-speech-to-text-with-pocketsphinx/. Available: 2016-03-25; Accessed: 2017-02-06.

Li, S., Aloimonos, Y., Qu, G., and Fermuller, C. (2018). *Human Robot Interaction on Gesture Controlled Drone: Methods of Gesture Action Recognition.* Master thesis, University of Maryland, College Park.

Lim, Y., Gardi, A., Sabatini, R., Ramasamy, S., Kistan, T., Ezer, N., Vince, J., and Bolia, R. (2018). ‘Avionics Human-Machine Interfaces and Interactions for manned and unmanned aircraft’. *Progress in Aerospace Sciences* pp. 1–49.

Lim, Y., Samreeloy, T., Chantaraviwat, C., Ezer, N., Gardi, A., and Sabatini, R. (2019). ‘Cognitive Human-Machine Interfaces and Interactions for Multi-UAV Operations’. In *Australian International Aerospace Congress (AIAC18)*, pp. 1–7, Melbourne, Australia. AIAC.

Lindquist, T. E. (1985). ‘Assessing the Usability of Human-Computer Interfaces’. *IEEE Software* **2**(1):74–82.

Liu, C., Prior, S. D., Teacy, W. L., and Warner, M. (2016a). ‘Computationally efficient visualinertial sensor fusion for Global Positioning Systemdenied navigation on a small quadrotor’. *Advances in Mechanical Engineering* **8**(3):168781401664099.

Liu, J., Luo, Y., and Ju, Z. (2016b). ‘An interactive astronaut-robot system with gesture control’. *Computational Intelligence and Neuroscience* **2016**.

Liu, Z., and Foina, A. G. (2016). 'An Autonomous Quadrotor Avoiding a Helicopter in Low-Altitude Flights' (10).

Locklear, M., and Engadget UK (2017). 'Amazon dreams up a drone that will understand your hand signals - It could be used to deliver packages'. https://www.engadget.com/2018/03/22/amazon-drone-understand-hand-signals/

Luis-pérez, F. E., Trujillo-romero, F., and Martínez-velazco, W. (2011). 'Control of a Service Robot Using the Mexican Sign Language' pp. 419–430.

Ma, L., and Cheng, L. L. (2016). 'Studies of AR Drone on Gesture Control'. In *3rd International Conference on Materials Engineering, Manufacturing Technology and Control (ICMEMTC 2016)*, pp. 1869–1873. Atlantis Press.

Ma, Y., Mao, Z.-H., Jia, W., Li, C., Yang, J., and Sun, M. (2010). 'Magnetic hand tracking for human-computer interface'. In *14th Biennial IEEE Conference on Electromagnetic Field Computation, CEFC2010, May 9, 2010 - May 12, 2010*, p. IEEE Magnetics Society, Chicago, IL, United states. IEEE Computer Society.

Ma, Y., Mao, Z.-H., Jia, W., Li, C., Yang, J., and Sun, M. (2011). 'Magnetic Hand Tracking for Human-Computer Interface'. *IEEE Transactions on Magnetics* **47**(5):970–973.

Makino, Y., Furuyama, Y., and Shinoda, H. (2015). 'HaptoClone (Haptic-Optica l Clone): Mid-a air Haptic-Optical Human-Human Interaction with Perfect Synchronization'. In *SUI '15 Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, pp. 139–139, Los Angeles, California, USA. ACM.

Martinovic, I., Davies, D., Frank, M., Perito, D., Ros, T., and Song, D. (2012). 'On the Feasibility of Side-Channel Attacks with Brain-Computer Interfaces'. *Usenixorg* pp. 1–16.

Mendes, R. (2015a). 'Tekever BRAINFLIGHT project'. http://tekevernews.blogspot.co.uk/2015/02/using-just-your-brain-to-control-drone.html

Mendes, R. (2015b). 'Tekever BRAINFLIGHT project - Blogpost'. http://tekevernews.blogspot.co.uk/2015/02/using-just-your-brain-to-control-drone.html. in *Tekever News blog.* First Available: 2015-02-24; Accessed: 2016-01-14.

Meyer, J., and Kohlbrecher, S. (2012). 'hector_quadrotor - ROS Wiki'. http://wiki.ros.org/hector{_}quadrotor

Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., and Von Stryk, O. (2012). 'Comprehensive simulation of quadrotor UAVs using ROS and Gazebo'. *Lecture Notes in Com-*

puter Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7628 LNAI**:400–411.

Michelson, S. (2014). 'Survey of the Human-Centered Approach to Micro Air Vehicles'. *Handbook of Unmanned Aerial Vehicles* pp. 1311–1327.

Microsoft (2014). 'Meet Kinect for Windows'. https://developer.microsoft.com/en-us/windows/kinect

Microsoft (2016). 'Microsoft Hololens Development Edition'. https://www.microsoft.com/microsoft-hololens/en-us. Available: 2016-03-01; Accessed: 2016-06-01.

Milanova, M., and Sirakov, N. (2008). 'Recognition of emotional states in natural human-computer interaction'. In *8th IEEE International Symposium on Signal Processing and Information Technology, ISSPIT 2008, December 16, 2008 - December 19, 2008*, pp. 186–191, Sarajevo, Bosnia and herzegovina. Inst. of Elec. and Elec. Eng. Computer Society.

Milgram, P., and Kishino, F. (1994). 'Taxonomy of mixed reality visual displays'. *IEICE Transactions on Information and Systems* **E77-D**(12):1321–1329.

Modellbau (2016). 'T-Drones Smart.X Quadrocopter (Rahmen) Typ A, TD-SX-A02'. https://www.mhm-modellbau.de/part-TD-SX-A02.php

Mohaimenianpour, S., and Vaughan, R. (2018). 'Hands and Faces , Fast : Mono-Camera User Detection Robust Enough to Directly Control a UAV in Flight'. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 8, Madrid, Spain. IEEE.

Monnai, Y., Hasegawa, K., Fujiwara, M., Yoshino, K., Inoue, S., and Shinoda, H. (2014). 'HaptoMime : Mid-Air Haptic Interaction with a Floating Virtual Screen'. In *Uist '14*, pp. 663–667, Honolulu, HI, USA. ACM.

Mordvintsev, A., and Abid, R. K. (2013a). 'Contour Features - OpenCV 3.2.0 dev'. http://docs.opencv.org/trunk/dd/d49/tutorial{_}py{_}contour{_}features.html

Mordvintsev, A., and Abid, R. K. (2013b). 'Face Detection using Haar Cascades - OpenCV 3.2.0 dev'. http://docs.opencv.org/trunk/d7/d8b/tutorial{_}py{_}face{_}detection.html

Mordvintsev, A., and Abid, R. K. (2013c). 'Face Detection using Haar Cascades - OpenCV 3.2.0 dev'. https://docs.opencv.org/3.2.0/d7/d8b/tutorial_py_face_detection.html. Available: 2013; Accessed: 2017-06-27.

Mu-Chun, S., and Ming-Tsang, C. (2001). 'Voice-controlled human-computer interface for the disabled'. *Computing & Control Engineering Journal* **12**(5):225–230.

Murphy, R., Shell, D., Guerin, A., Duncan, B., Fine, B., Pratt, K., and Zourntos, T. (2011). 'A Midsummer Night's Dream (with flying robots)'. *Autonomous Robots* **30**(2):143–156.

Muscio, G., Pierri, F., Trujillo, M. A., Cataldi, E., Giglio, G., Antonelli, G., Caccavale, F., Viguria, A., Chiaverini, S., and Ollero, A. (2016). 'Experiments on coordinated motion of aerial robotic manipulators'. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1224–1229, Stockholm, Sweden. IEEE Robotics & Automation Society.

Myers, B. A. (1998). 'A Brief History of Human Computer Interaction Technology'. *ACM Interactions* **5**(December):44–54.

Nagar, A., and Xu, Z. (2015). 'Gesture control by wrist surface electromyography'. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops*, pp. 556–561.

Nagi, J., Giusti, A., Gambardella, L. M., and Di Caro, G. A. (2014). 'Human-swarm interaction using spatial gestures'. *IEEE International Conference on Intelligent Robots and Systems* (Iros):3834–3841.

National Institute of Health (2011a). 'Electromyography'. https://www.nlm.nih.gov/cgi/mesh/2011/MB{_}cgi?index=4387

National Institute of Health (2011b). 'Magnetoencephalography'. https://www.nlm.nih.gov/cgi/mesh/2011/MB{_}cgi?index=14595

NeuroSky (2011). 'Brainwaves. Not Thoughts.'. http://neurosky.com/biosensors/eeg-sensor/

NeuroSky (2015). 'A Real Game-Changer : BCI & EEG Use Cases for Video Games'. http://download.neurosky.com/docs/whitepapers/Video-Game-Use-Case.pdf

Ng, W. S., and Sharlin, E. (2011). 'Collocated interaction with flying robots'. *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication* pp. 143–149.

Nguyen, C. (2014). 'U . S . Scientists Have Taken A Breakthrough Step Into Mind Control'. http://techdrive.co/2014/12/u-s-sciencetists-taken-breakthrough-step-mind-control/

Niedermeyer, E., and Lopes Da Silva, F. H. (2004). *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincott Williams and Wilkins, Philadelphia, 5th edn.

Nimble VR (2012). 'Nimble VR is joining Oculus'. http://nimblevr.com/index.html

Noll, A. M. (1975). 'Tactile man-machine communication system'. https://www.google.com/patents/US3919691

Nwana, H. S. (1996). 'Software Agents : An Overview'. *Knowledge Engineering Review* **11**(3):205–244.

Obaid, M., Kistler, F., Kasparaviciute, G., Yantaç, A. E., and Fjeld, M. (2016). 'How Would You Gesture Navigate a Drone? A User-Centered Approach to Control a Drone'. In *Proceedings of the 20th International Academic Mindtrek Conference*, pp. 113–121, Tampere, Finland. ACM New York, NY, USA.

Oculus VR (2016). 'Oculus rift: Next-generation virtual reality'. https://www.oculus.com/en-us/rift/

OpenCV (2010). 'Open Source Computer Vision Library'. https://github.com/opencv/opencv

Orzea, E. (2015). 'VDCI unveils its all in all compact Ground Control Station for any UAV like DJI and 3D Robotics'. http://www.suasnews.com/2015/02/34514/. in *sUAS News.* First Available: 2015; Accessed: 2016-01-11.

Oviatt, S. (1999). 'Ten myths of multimodal interaction'. *Communications of the ACM* **42**(11):74–81.

Oviatt, S. (2002). 'Breaking the robustness barrier: Recent progress on the design of robust multimodal systems'. *Advances in Computers* **56**(C):305–341.

Oviatt, S. (2003). 'Multimodal interfaces'. In J. A. Jacko & A. Sears (eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*, chap. 14, pp. 286–304. Lawrence Erlbaum Associates, Incorporated, London, first edn.

Oviatt, S., Coulston, R., and Lunsford, R. (2004). 'When do we interact multimodally?: cognitive load and multimodal communication patterns'. *... conference on Multimodal interfaces* pp. 129–136.

Owano, N., Phys.org, and TechXplore (2015). 'Brain signals turn into drone commands in Lisbon presentation'. http://phys.org/pdf344230897.pdf. First Available: 2015-02-27; Accessed: 2016-01-14.

Paul, D. B. (1990). 'Speech Recognition Using Hidden Markov Models'. *The Lincoln Laboratory Journal* **3**(1):41–62.

Payne, K. (2007). 'Autonomy - Proposal for a UK Route Map'.

Pfeifer, R., and Scheier, C. (1999). 'Embodied Cognitive Science: Basic Concepts'. In R. Pfeifer & C. Scheier (eds.), *Understanding Intelligence*, chap. 4, pp. 81–138. MIT Press.

PJRC (2015). 'Teensy 3.2 & 3.1 New Features'. https://www.pjrc.com/teensy/teensy31.html

Preece, J., Sharp, H., and Rogers, Y. (2015). *Interaction Design: Beyond Human-Computer Interaction.* John Wiley & Sons Ltd, Glasgow, fourth edn.

Puttemans, S. (2015). 'Cascade Classifier Training - OpenCV 3.2.0 dev'. http://docs.opencv.org/trunk/dc/d88/tutorial{_}traincascade.html

Qing, C., Cordea, M. D., Petriu, E. M., Whalen, T. E., Rudas, I. J., and Varkonyi-Koczy, A. (2008). 'Hand-Gesture and Facial-Expression Human-Computer Interfaces for Intelligent Space Applications'. In *Medical Measurements and Applications, 2008. MeMeA 2008. IEEE International Workshop on*, pp. 1–6.

Rabiner, L. (1989). 'A tutorial on hidden Markov models and selected applications in speech recognition'. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=18626{%}0Apapers3://publication/doi/10.1109/5.18626

Ramchurn, S. D., Fischer, J. E., Ikuno, Y., Wu, F., Flann, J., and Waldock, A. (2015). 'A study of human-agent collaboration for multi-UAV task allocation in dynamic environments'. *IJCAI International Joint Conference on Artificial Intelligence* **2015-Janua**:1184–1192.

Ranky, G. N., and Adamovich, S. (2010). 'Analysis of a commercial EEG device for the control of a robot arm'. In *Bioengineering Conference, Proceedings of the 2010 IEEE 36th Annual Northeast*, pp. 1–2.

Raytheon, and Barnard Micro Systems (2006). 'Raytheon improves unmanned aircraft pilot's awareness'. http://www.barnardmicrosystems.com/UAV/features/ground{_}control.html

Redden, E. S., Carstens, C. B., and Pettitt, R. A. (2010). 'Intuitive Speech-based Robotic Control'. *U.S. Army Research Laboratory* (Technical Report ARL-TR-5175).

Reeves, L. M., Martin, J.-C., McTear, M., Raman, T., Stanney, K. M., Su, H., Wang, Q. Y., Lai, J., Larson, J. a., Oviatt, S., Balaji, T. S., Buisine, S., Collings, P., Cohen, P., and Kraal,

B. (2004). 'Guidelines for multimodal user interface design'. *Communications of the ACM* **47**(1):57.

Renitto, J. E., and Thomas, N. K. (2014). 'A Survey on Gesture Recognition Technology'. *international Journal for Technological Research in Engineering* **1**(10):1058–1060.

Reuters (2015). 'Using The Force? No, it's an Apple Watch flying this drone'. http://www.reuters.com/article/us-apple-watch-drone-idUSKBN0UE14Q20160101. First Available: 2015-12-31; Accessed: 2016-01-21.

Rim, B., and Schiaratura, L. (1991). 'Gesture and speech'. In R. S. Feldman & B. Rime (eds.), *Fundamentals of Nonverbal Behavior*, chap. 7, pp. 239–281. Cambridge University Press, New York.

Romell, A., and Karjalainen, K. (2017). *Human-Drone Interaction: Drone as a companion?* Master thesis, Chalmers University of Technology, Gothenburg, Sweden.

Root, S., and Air Zermatt (2016). 'The Matterhorn 101 - This is all you need to know about the Matterhorn'. https://www.redbull.com/int-en/the-horn-air-zermatt-matterhorn-rescue-team. Available: 2016-10-17; Accessed: 2017-06-07.

Ross, S. M. (2014). 'Chapter 6 - Distributions of Sampling Statistics'. In *Introduction to Probability and Statistics for Engineers and Scientists (Fifth Edition)*, chap. 6, pp. 207–233. Academic Press, Boston, 5th edn.

Ruhnke, I. (2009). 'jstest-gtk: A joystick testing and configuration tool for GNU/Linux'. https://jstest-gtk.gitlab.io/. Available: 2009; Accessed: 2019-05-31.

Sanders, M. S., and McCormick, E. J. (1993). *Human factors in engineering and design.* McGraw-Hill Education, New York, seventh edn.

Sanna, A., Lamberti, F., Paravati, G., and Manuri, F. (2013). 'A Kinect-based natural interface for quadrotor control'. *Entertainment Computing* **4**(3):179–186.

Sarkar, A., Patel, K. A., Ganesh, R. R. K., and Capoor, G. K. (2016). 'Gesture Control of Drone Using a Motion Controller'. In *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)*, pp. 1–5, Sharjah. IEEE.

Schelle, A., and Stutz, P. (2016). 'Modelling and Simulation for Autonomous Systems'. In J. Hodicky (ed.), *Modelling and Simulation for Autonomous Systems. MESAS 2016. Lecture Notes in Computer Science*, vol. 9991, pp. 81–98, Cham. Springer.

Schelle, A., and Stütz, P. (2018). 'Gestural Transmission of Tasking Information to an Airborne UAV'. In S. Yamamoto & H. Mori (eds.), *Human Interface and the Management of Information. Interaction, Visualization, and Analytics*, pp. 318–335, Cham. Springer International Publishing.

Scocco, D. (2006). 'Innovation management theory part 4: S-Curves'. http://innovationzen.com/blog/2006/08/17/innovation-management-theory-part-4/. Available: 2006-08-17; Accessed: 2017-07-03.

Shadow Robot (2013). 'Shadow Dexterous Hand E1 Series (E1M3R, E1M3L, E1P1R, E1P1L)'. http://www.shadowrobot.com/wp-content/uploads/shadow{_}dexterous{_}hand{_}technical{_}specification{_}E1{_}20130101.pdf

Shah, J., and Breazeal, C. (2010). 'An Empirical Analysis of Team Coordination Behaviors and Action Planning With Application to Human-Robot Teaming'. *Human Factors: The Journal of the Human Factors and Ergonomics Society* **52**(2):234–245.

Shetty, A., Shinde, A., Patel, J., Panchal, N., and Jha, M. (2016). 'Gesture Controlled Quadcopter'. *Imperial Journal of Interdisciplinary Research (IJIR)* **2**(5):1289–1291.

Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, third edn.

Sholes, E. (2007). 'Evolution of a UAV autonomy classification taxonomy'. *IEEE Aerospace Conference Proceedings* .

Soto-Gerrero, D., and Ramrez-Torres, J. G. (2013). 'A human-machine interface with unmanned aerial vehicles'. In *Electrical Engineering, Computing Science and Automatic Control (CCE), 2013 10th International Conference*, vol. 37, pp. 307 – 312, Mexico City. IEEE.

Stanton, N., Hedge, A., Brookhuis, K., Salas, E., and Hendrick, H. (2005). *Handbook of Human Factors and Ergonomics Methods.* CRC Press, New York, first edn.

Sutherland, I. E. (1968). 'A head-mounted three dimensional display'. *Proceedings of the AFIPS '68 (Fall, part I)* p. 757.

Thalmic Labs (2013a). 'Homepage - Gesture Control Has Arrived'. https://www.myo.com/. First Available: 2013; Accessed: 2016-01-19.

Thalmic Labs (2013b). 'Homepage - Gesture Control Has Arrived'. https://www.myo.com/

Thalmic Labs (2015a). 'Myo - Real Life Applications of the Myo Armband'. https://www.youtube.com/watch?v=te1RBQQlHz4. First Available: 2015-08-19; Accessed: 2016-01-19.

Thalmic Labs (2015b). ‘Myo - Real Life Applications of the Myo Armband’. https://www.youtube.com/watch?v=te1RBQQlHz4

Thobbi, A., Kadam, R., and Sheng, W. (2010). ‘Achieving Remote Presence using a Humanoid Robot Controlled by a Non-Invasive BCI Device’. *ICGST International Journal on Automation, Robotics and Autonomous Systems* **10**(1):5.

Turk, M. (2014). ‘Multimodal interaction: A review’. *Pattern Recognition Letters* **36**(1):189–195.

Turnigy (2014). ‘Turnigy TGY-i6 AFHDS Tx and 6CH Rx’. http://www.hobbyking.com/hobbyking/store/__62710__Turnigy_TGY_i6_AFHDS_Transmitter_and_6CH_Receiver_Mode_2_.html. Av 2014; Accessed: 2016-01-11.

Uchat, N. S. (2006). ‘Hidden Markov Model and Speech Recognition’. https://www.cse.iitb.ac.in/{~}nirav06/i/HMM{_}Report.pdf

University of Minnesota (2013a). ‘Mind over mechanics - Discover’. http://discover.umn.edu/news/science-technology/brain-computer-interface-allows-mind-control-robots

University of Minnesota (2013b). ‘Mind Over Mechanics - YouTube Video’. https://www.youtube.com/watch?v=rpHy-fUyXYk

Verma, P. (2016). ‘Flying User Interface’. In *UIST ’16 Adjunct Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 203–204, Tokyo, Japan. ACM New York, NY, USA.

Viallet, J. E., Carbini, S., Delphin-Poulat, L., and Perron, L. (2006). ‘From a Wizard of Oz experiment to a real time speech and gesture multimodal interface’. *Signal Processing* **86**(12):3559–3577.

Viola, P., and Jones, M. (2001). ‘Robust real-time object detection’. *International Journal of Computer Vision* **57**(2):137–154.

Viola, P., and Jones, M. J. (2004). ‘Robust Real-Time Face Detection’. *International Journal of Computer Vision* **57**(2):137–154.

Wachs, J. (2005). ‘Haar-based Detectors For Hand Gesture Letter ‘A’’. https://github.com/codeflows/bordai/blob/master/resources/aGest.xml. Available: 2011-05-18; Accessed: 2019-05-27.

Wachs, J., Stern, H., Edan, Y., Gillam, M., Feied, C., Smith, M., and Handler, J. (2006). ‘A real-time hand gesture interface for medical visualization applications’. In A. Tiwari, R. Roy,

J. Knowles, E. Avineri, & K. Dahal (eds.), *Advances in Intelligent and Soft Computing*, vol. 36, pp. 153–162, Berlin, Heidelberg. Springer.

Waibel, M. (2014a). 'A future for flying machines in entertainment'. `http://robohub.org/new-quadrocopter-video-points-to-a-future-for-flying-machines-in-entertainment/`. in *Robohub's Online Newsletter*. Available: 2014-09-22; Accessed: 2016-02-22.

Waibel, M. (2014b). 'New quadrocopter video points to a future for flying machines in entertainment'. `http://robohub.org/new-quadrocopter-video-points-to-a-future-for-flying-machines-in-entertainment/`

Waibel, M., Ambühl, A., Lupashin, S., and D'Andrea, R. (2011). 'Controlling a Quadrotor Using Kinect'. `http://spectrum.ieee.org/automaton/robotics/robotics-software/quadrotor-interaction`

Young, S. J., Evermann, G., Gales, M. J. F., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. C. (2006). 'The HTK book (v3. 4)'. Tech. rep.

Zollinger, S. A., and Brumm, H. (2011). 'The Lombard effect'. *Current Biology* **21**(16):R614–R615.

Appendices

Appendix A

Aerial Robot Applications

Table A.1 presents some detailed application areas of the multi-rotor UAV.

Table A.1: Applications of Autonomous and Remotely Piloted Unmanned Aerial Systems

S/No	UAV/RPAS Application	Examples	Benefits	Stakeholders
1	Aerial Inspection	• Remotely Operated Aerial Vehicles (ROAV) inspection for onshore oil, gas & chemical industries • ROAV inspection for the offshore oil & gas industry • UAV inspection for the Power and Utility industries • Maersk Oil North Sea gruelling environment aerial inspection	• Save money • De-risk planned shutdowns and maintenance • No plant outage • Minimum Health & Safety issues • High quality information • Close flare tip inspection	• CYBERHAWK Innovations Ltd • Maersk Oil
2	Aerial Survey	• UAV land surveys for Utility industries: Topographic surveys, Environmental impact assessments, As-built surveys, Construction progress photography, Flood risk surveys, Volumetric analysis, Line of sight surveys, Site design, Route surveys, Marketing images • UAV land surveys for Civil Engineering industries: Topographic surveys, Volumetric surveys, Construction progress photography, Flood risk surveys, Quarry/Landfill Surveys, Route surveys, As-built surveys, Marketing images, Site design • UAV land surveys for Government & Public Bodies: Flood risk surveys, Storm damage report, Archaeological site recording, Historic building surveys, Coastal erosion surveys, Ecological surveys, Forestry management surveys, 3D modelling	• Reduce costs • Save time • Improve Safety • Better information • Improve decision making	• CYBERHAWK Innovations Ltd
3	Agriculture Application	• Drone set to tackle pest problem: Embention programme working on eradication of tsetse fly • Quantum VRT drones used for detailed monitoring and evaluation of crop conditions	• Low cost delivery technique • Less Fertiliser • Increased crop yield • Reduce pollution of ground water	• Embention & IAEA • Florian Seibel, Quantum Systems
4	Environmental Application	• Detecting nuclear radiation with drones	• Prevents humans from radiation exposure & the resulting health risk such as radiation poisoining, cancer, etc. • Real time radioactive incident report on nuclear sites • Confirming safe zones	• Bristol-Oxford Nuclear Research Centre
5	Field Application	• Rescue drones for off-roaders: Drone flare to search for network and send SOS messages with GPS coordinates	• Emergency SOS comapanion	• Kenneth Wong et al

Applications of Autonomous and Remotely Piloted Unmanned Aerial Systems - Continued on the next page...

S/No	UAV/RPAS Application	Examples	Benefits	Stakeholders
6	Open-Water Applications	• Loon set to take on air and sea: Loon copter that can fly, swim, and dive • Shark spotter • Air aid: The Migrant Offshore Aid Station (MOAS), M.Y. Phoenix, an emergency rescue vessel in the Mediterranean, uses drones to save lives • Eye in the sky set to save lives: Search and rescue drone reduces risk for NZ Coastguard crews • Save the Whales with Drone technology	• Search-and-rescue operations • Bridge foundation inspections • Underwater pipeline inspections • Tracking of oil spills at different depths Marine life studies • Shark monitoring and Shark spotters • Prevent whale culling	• Osamah A. Rawashdeh, Oakland University • Australia Government Shark Strategy • Christopher and Regina Catrambone • New Zealand Coastguard • Sea Shepherd Conservation Society
7	Mapping	• Crisis Mapping in Nepal	• Aids re-building efforts	• DJI • UAViators • Pix4D
8	Firefighting Applications	• Drones to the rescue: via Aerial Imagery Reconnaissance (AIR) units	• Monitor the safety of firefighters • Better information about fast-moving situations such as fires, burned out buildings, floods and road accidents • Helps commander's plan strategy • Find missing persons • Useful incident footage for training • Improving safety • Providing firefighters with real-time information	• GMFRS - The Greater Manchester Fire and Rescue Service
9	Flood Monitoring	• Flood protection targeted by Malaysian drone	• Easy access to flood hurt areas • Relay real-time data • Supplement satellite flood monitoring • Monitoring fire hotspots • Monitoring Illegal logging and farms	• Malaysian Government • MSTIRSA, Malaysia • Multimedia University • Unmanned Systems Technology
10	Entertainment Applications	• SPARKED: A Live Interaction Between Humans and Quadcopters • Robots Perform in Shakespeare's "A Midsummer Nights Dream" • Flying Robot Dance	• Sophisticated real time play effects • Complex spectacular aerobic choreography	• Cirque du Soleil • ETH Zurich • Verity Studios • Texas A&M University • KMel Robotics • Yuneec International

Applications of Autonomous and Remotely Piloted Unmanned Aerial Systems - Continued on the next page...

S/No	UAV/RPAS Application	Examples	Benefits	Stakeholders
11	Commercial Newsgathering	• Major news network trials drone coverage • Sport coverage	• Report file footage • Broadcast live footage • Live event coverage • Natural disaster, riots, and war zone coverage • Live sports video feeds	• CNN • NFL • Royal Ascot horse racing
12	Drone Delivery	• Amazon prime air • DHL's Parcelcopter • Airmail: Drones looking to deliver across tricky Swiss terrain • On demand delivery could mark new future for retail • VertiKUL: A Delivery Drone capable of delivering packages to a destination 30 Kilometers away	• Rural or remote delivery • High priority consignment delivery • Medical supplies • Reduced carbon emissions • Cheaper delivery alternatives • High-speed delivery	• Amazon • DHL • Swiss Post • Google Australia • University of Leuven, Belgium
13	Rail Network Applications	• Railway track inspection • Monitoring rail track safety • UAV inspection for the Power and Utility industries • Maersk Oil North Sea gruelling environment aerial inspection	• Potentially quicker inspection • Reduces rail downtime due to maintenance • Continuous monitoring • Deterrent to rail-crossing offenders	• BNSF Railway • GCC-wide rail network
14	Exam Invigilation	• Chinese government uses high-tech drones to catch cheating students	• Detect radio communication signals • Highlight suspicious radio signals • Dynamic video recording	• Chinese Government
15	Photography	• Aerial Views: Photography contest showcases creativity through drones	• Rare aerial shots • Magical photos • Rare views of nature	• Professional and Amateur Photographers
16	Health care applications	• Dronlife: making organ delivery for transplant easier • Medical drone delivers • Drones help save lives • Drone rural healthcare services via Matternets network of flying drones ferrying lightweight medical supplies and information	• Quick organ transportation • Medicines delivery to the elderly or special needs patients • Blood delivery to hospitals • Medical kits transportation • Medical supplies to rural and remote communities • Tests and diagnostic information transportation in rural areas of developing countries	• University of A Coruña, Spain • Mishaal Almarzouqui, Dubai Health Authority • Main Hospital, Bhutan • Andreas Raptopoulos, Matternet

Applications of Autonomous and Remotely Piloted Unmanned Aerial Systems - Continued on the next page...

S/No	UAV/RPAS Application	Examples	Benefits	Stakeholders
17	Wild life conservation	• Wildlife conservation tool: drones track poaching and populations, and for filming wildlife unobtrusively • Drones help the Endangered List: used by Sumatra researchers in protecting elephants and beleaguered orangutuans • Preventing Poaching with domestic Drones: by monitoring the health and security of the rare tigers, elephants and rhinoceri that roam their national parks	• Track animal poaching • Track wildlife population • Filming wildlife • Track wildlife behaviour and movements • Monitor illegal deforestation in wildlife areas • Stamp out poaching of endangered species • Create awareness evidence of poaching	• World Wildlife Fund (WWF) • Conservation Drones • Sumatra Wildlife Researchers • Nepalese and Indian conservationists
18	Drone Rides	• Ride on drone gets lift off • Will we see Hover-bikes in the near Future?	• Pilotless aerial travel • Quadrotor Hover-bike motorcycle	• Thorstin Crijns • Malloy Aeronautics
19	Indoor Inspection	• Collision tolerant drone gimbal used in Bridges, tunnels, power plants, boats, ships and vessels inspection	• Reduced accident impacts • Reduced health risks & hazards • Access through little holes or crevices • Access to confined spaces • Safely fly close to humans • Collision tolerant	• Patrick Thevoz, Flyability
20	Mountaineering expedition	• Sprite: the world's most portable and rugged unmanned aerial vehicle Sprite challenges drone design with new approach	• Portability • Rugged	• Ascent Aero Systems
21	Search and Rescue	• Nepal quake disaster puts drones to the test	• Locating victims • Surveying damages	• NGOs
22	Habitat exploration	• Henri A Waterproof Drone for surveying habitats above and below water	• Map reefs and coastal zones • Survey small underwater habitats	• Haiti
23	Sport Videography	• Air Dog: follows and films your every move • A Pet Drone That Follows You Everywhere	• Capture personal sporting moments • Amateur sport videography • Follow athletic autonomously • Captures Pro-athletes training video for improvement analysis	• Helico Aerospace Industries • Falkor Systems

Applications of Autonomous and Remotely Piloted Unmanned Aerial Systems - Continued on the next page...

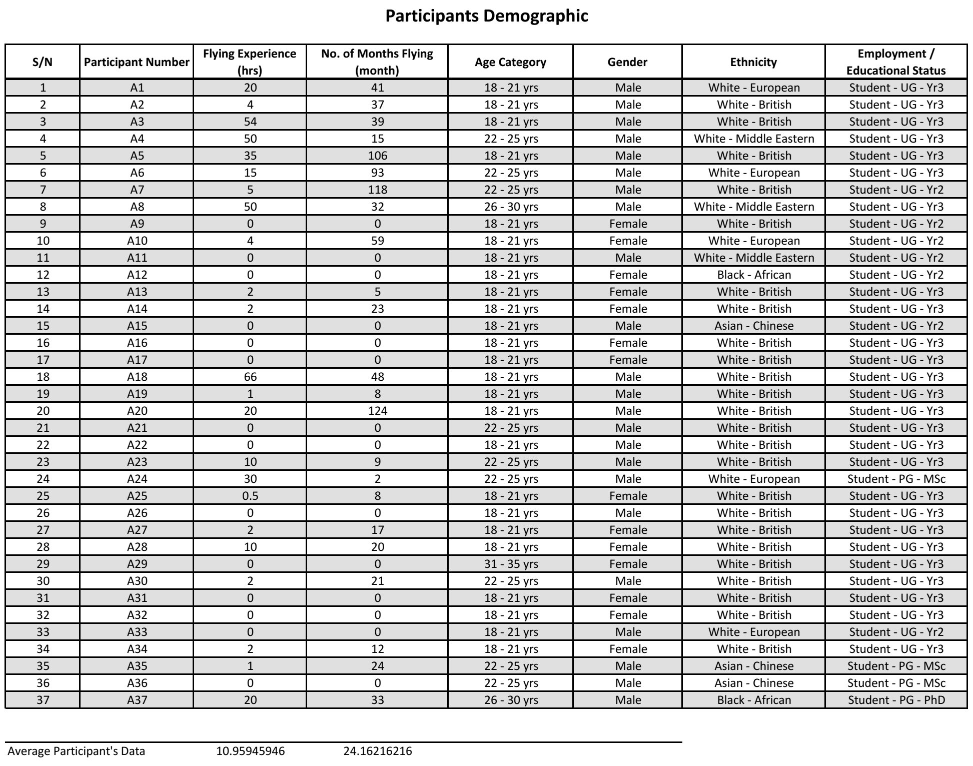
S/No	UAV/RPAS Application	Examples	Benefits	Stakeholders
24	Law enforcement applications	• Thermal-Imaging Drones to catch vandals on German Railways	• Deterrents to law breaking • Tailing law breakers • Providing video evidence to prosecute offenders	• Deutsche Bahn, Germanys national railways company
25	Disaster Relief	• Disaster relief aided by drone support in India Uttarakhand region's flood	• Direct delivery of food and medical supplies • Mapping affected areas • Delivering first aid kits to the more isolated areas	• Social Drones

Applications of Autonomous and Remotely Piloted Unmanned Aerial Systems - End

Appendix B

Experiment Result Tables

B.1 Participants Demographic



Participants Demographic

S/N	Participant Number	Flying Experience (hrs)	No. of Months Flying (month)	Age Category	Gender	Ethnicity	Employment / Educational Status
1	A1	20	41	18 - 21 yrs	Male	White - European	Student - UG - Yr3
2	A2	4	37	18 - 21 yrs	Male	White - British	Student - UG - Yr3
3	A3	54	39	18 - 21 yrs	Male	White - British	Student - UG - Yr3
4	A4	50	15	22 - 25 yrs	Male	White - Middle Eastern	Student - UG - Yr3
5	A5	35	106	18 - 21 yrs	Male	White - British	Student - UG - Yr3
6	A6	15	93	22 - 25 yrs	Male	White - European	Student - UG - Yr3
7	A7	5	118	22 - 25 yrs	Male	White - British	Student - UG - Yr2
8	A8	50	32	26 - 30 yrs	Male	White - Middle Eastern	Student - UG - Yr3
9	A9	0	0	18 - 21 yrs	Female	White - British	Student - UG - Yr2
10	A10	4	59	18 - 21 yrs	Female	White - European	Student - UG - Yr2
11	A11	0	0	18 - 21 yrs	Male	White - Middle Eastern	Student - UG - Yr2
12	A12	0	0	18 - 21 yrs	Female	Black - African	Student - UG - Yr2
13	A13	2	5	18 - 21 yrs	Female	White - British	Student - UG - Yr3
14	A14	2	23	18 - 21 yrs	Female	White - British	Student - UG - Yr3
15	A15	0	0	18 - 21 yrs	Male	Asian - Chinese	Student - UG - Yr2
16	A16	0	0	18 - 21 yrs	Female	White - British	Student - UG - Yr3
17	A17	0	0	18 - 21 yrs	Female	White - British	Student - UG - Yr3
18	A18	66	48	18 - 21 yrs	Male	White - British	Student - UG - Yr3
19	A19	1	8	18 - 21 yrs	Male	White - British	Student - UG - Yr3
20	A20	20	124	18 - 21 yrs	Male	White - British	Student - UG - Yr3
21	A21	0	0	22 - 25 yrs	Male	White - British	Student - UG - Yr3
22	A22	0	0	18 - 21 yrs	Male	White - British	Student - UG - Yr3
23	A23	10	9	22 - 25 yrs	Male	White - British	Student - UG - Yr3
24	A24	30	2	22 - 25 yrs	Male	White - European	Student - PG - MSc
25	A25	0.5	8	18 - 21 yrs	Female	White - British	Student - UG - Yr3
26	A26	0	0	18 - 21 yrs	Male	White - British	Student - UG - Yr3
27	A27	2	17	18 - 21 yrs	Female	White - British	Student - UG - Yr3
28	A28	10	20	18 - 21 yrs	Female	White - British	Student - UG - Yr3
29	A29	0	0	31 - 35 yrs	Female	White - British	Student - UG - Yr3
30	A30	2	21	22 - 25 yrs	Male	White - British	Student - UG - Yr3
31	A31	0	0	18 - 21 yrs	Female	White - British	Student - UG - Yr3
32	A32	0	0	18 - 21 yrs	Female	White - British	Student - UG - Yr3
33	A33	0	0	18 - 21 yrs	Male	White - European	Student - UG - Yr2
34	A34	2	12	18 - 21 yrs	Female	White - British	Student - UG - Yr3
35	A35	1	24	22 - 25 yrs	Male	Asian - Chinese	Student - PG - MSc
36	A36	0	0	22 - 25 yrs	Male	Asian - Chinese	Student - PG - MSc
37	A37	20	33	26 - 30 yrs	Male	Black - African	Student - PG - PhD

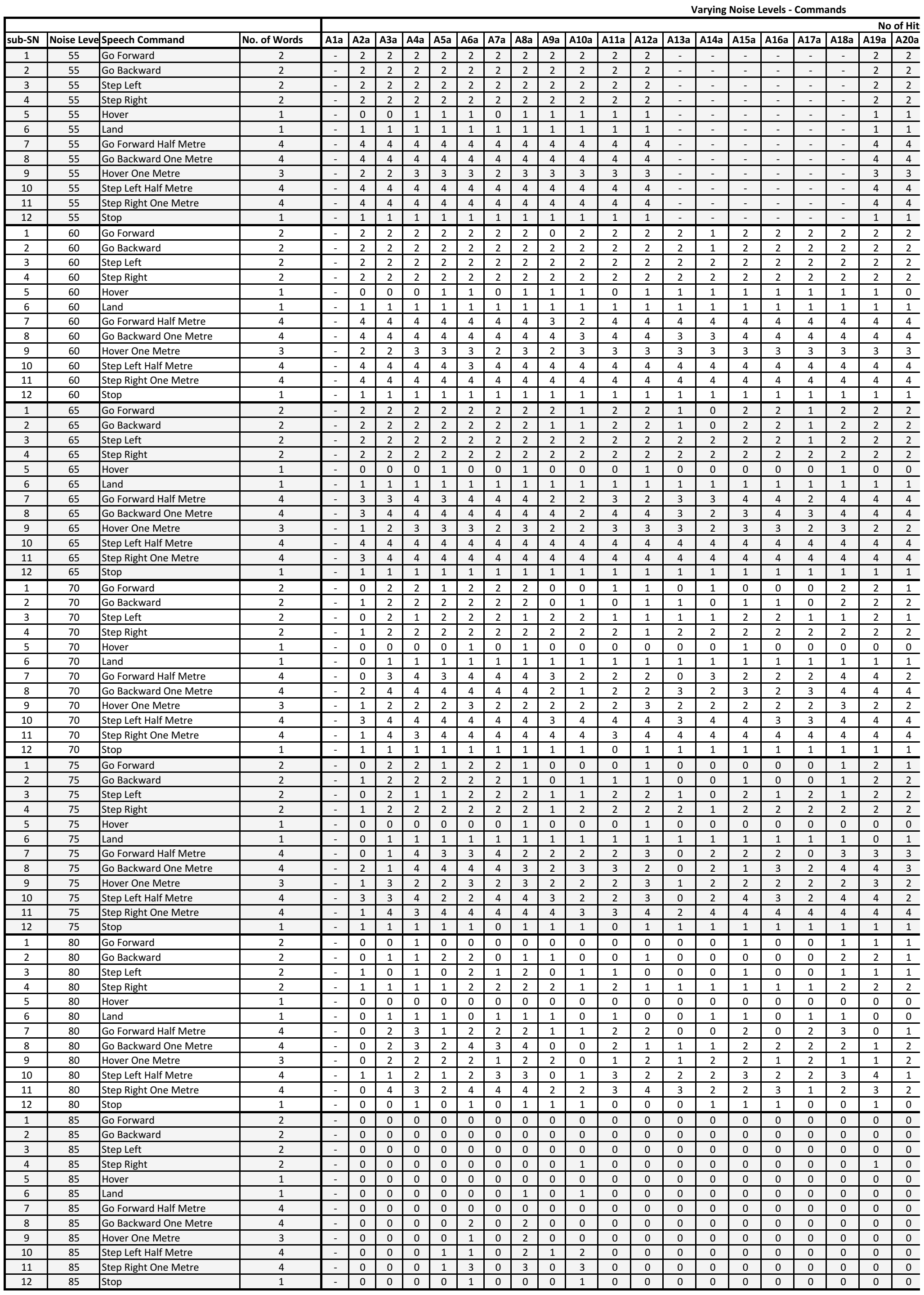
Average Participant's Data		10.95945946	24.16216216				

Figure B.1: Participants demographics.

This page is intentionally left blank.

B.2 Task A Results Table

B.2.1 Speech Command

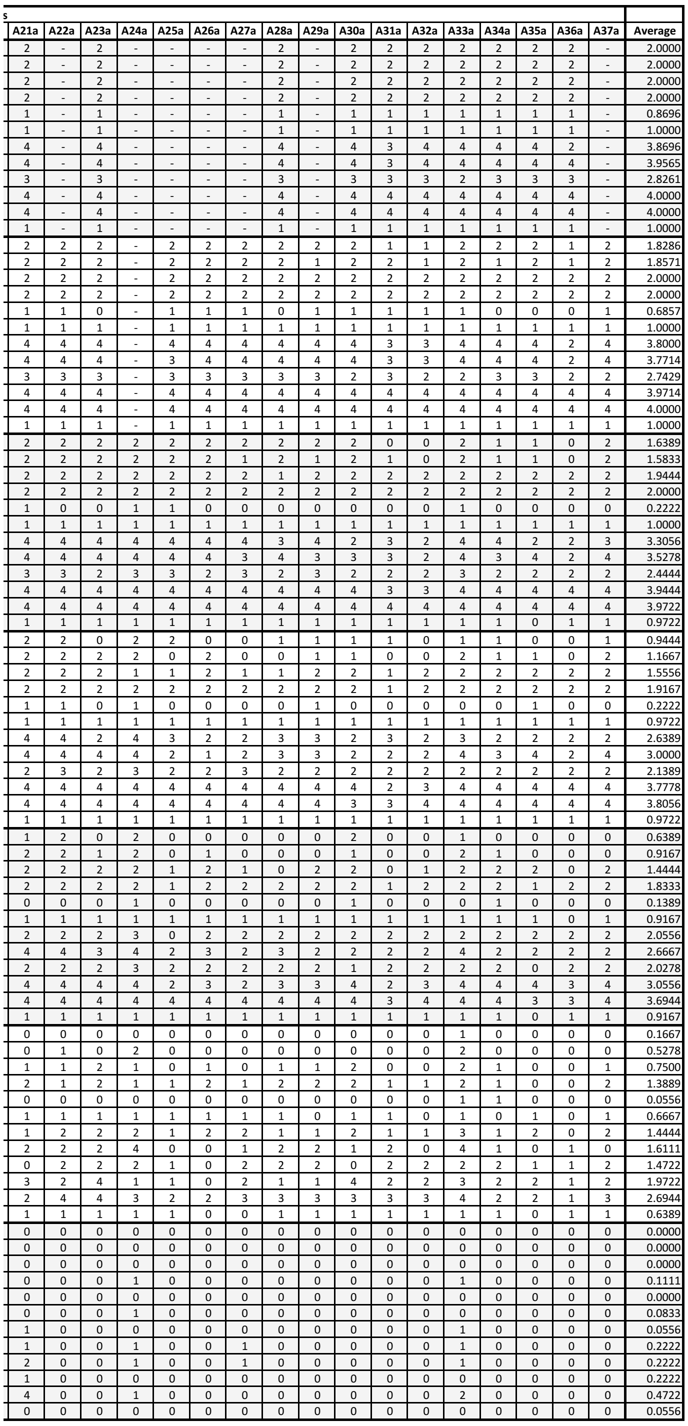


Varying Noise Levels - Commands

																							No of Hit
sub-SN	Noise Leve	Speech Command	No. of Words	A1a	A2a	A3a	A4a	A5a	A6a	A7a	A8a	A9a	A10a	A11a	A12a	A13a	A14a	A15a	A16a	A17a	A18a	A19a	A20a
1	55	Go Forward	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2
2	55	Go Backward	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2
3	55	Step Left	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2
4	55	Step Right	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2
5	55	Hover	1	-	0	0	1	1	1	0	1	1	1	1	1	-	-	-	-	-	-	1	1
6	55	Land	1	-	1	1	1	1	1	1	1	1	1	1	1	-	-	-	-	-	-	1	1
7	55	Go Forward Half Metre	4	-	4	4	4	4	4	4	4	4	4	4	4	-	-	-	-	-	-	4	4
8	55	Go Backward One Metre	4	-	4	4	4	4	4	4	4	4	4	4	4	-	-	-	-	-	-	4	4
9	55	Hover One Metre	3	-	2	2	3	3	3	2	3	3	3	3	3	-	-	-	-	-	-	3	3
10	55	Step Left Half Metre	4	-	4	4	4	4	4	4	4	4	4	4	4	-	-	-	-	-	-	4	4
11	55	Step Right One Metre	4	-	4	4	4	4	4	4	4	4	4	4	4	-	-	-	-	-	-	4	4
12	55	Stop	1	-	1	1	1	1	1	1	1	1	1	1	1	-	-	-	-	-	-	1	1
1	60	Go Forward	2	-	2	2	2	2	2	2	2	0	2	2	2	2	1	2	2	2	2	2	2
2	60	Go Backward	2	-	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2
3	60	Step Left	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	60	Step Right	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	60	Hover	1	-	0	0	0	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	0
6	60	Land	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	60	Go Forward Half Metre	4	-	4	4	4	4	4	4	4	3	2	4	4	4	4	4	4	4	4	4	4
8	60	Go Backward One Metre	4	-	4	4	4	4	4	4	4	4	3	4	4	3	3	4	4	4	4	4	4
9	60	Hover One Metre	3	-	2	2	3	3	3	2	3	2	3	3	3	3	3	3	3	3	3	3	3
10	60	Step Left Half Metre	4	-	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
11	60	Step Right One Metre	4	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
12	60	Stop	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	65	Go Forward	2	-	2	2	2	2	2	2	2	2	1	2	2	1	0	2	2	1	2	2	2
2	65	Go Backward	2	-	2	2	2	2	2	2	2	1	1	2	2	1	0	2	2	1	2	2	2
3	65	Step Left	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2
4	65	Step Right	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	65	Hover	1	-	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0
6	65	Land	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	65	Go Forward Half Metre	4	-	3	3	4	3	4	4	4	2	2	3	2	3	3	4	4	2	4	4	4
8	65	Go Backward One Metre	4	-	3	4	4	4	4	4	4	4	2	4	4	3	2	3	4	3	4	4	4
9	65	Hover One Metre	3	-	1	2	3	3	3	2	3	2	2	3	3	3	2	3	3	2	3	2	2
10	65	Step Left Half Metre	4	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
11	65	Step Right One Metre	4	-	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
12	65	Stop	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	70	Go Forward	2	-	0	2	2	1	2	2	2	0	0	1	1	0	1	0	0	0	2	2	1
2	70	Go Backward	2	-	1	2	2	2	2	2	2	0	1	0	1	1	0	1	1	0	2	2	2
3	70	Step Left	2	-	0	2	1	2	2	2	1	2	2	1	1	1	1	2	2	1	1	2	1
4	70	Step Right	2	-	1	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2
5	70	Hover	1	-	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
6	70	Land	1	-	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	70	Go Forward Half Metre	4	-	0	3	4	3	4	4	4	3	2	2	2	0	3	2	2	2	4	4	2
8	70	Go Backward One Metre	4	-	2	4	4	4	4	4	4	2	1	2	2	3	2	3	2	3	4	4	4
9	70	Hover One Metre	3	-	1	2	2	2	3	2	2	2	2	2	3	2	2	2	2	2	3	2	2
10	70	Step Left Half Metre	4	-	3	4	4	4	4	4	4	3	4	4	4	3	4	4	3	3	4	4	4
11	70	Step Right One Metre	4	-	1	4	3	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4
12	70	Stop	1	-	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
1	75	Go Forward	2	-	0	2	2	1	2	2	1	0	0	0	1	0	0	0	0	0	1	2	1
2	75	Go Backward	2	-	1	2	2	2	2	2	1	0	1	1	1	0	0	1	0	0	1	2	2
3	75	Step Left	2	-	0	2	1	1	2	2	2	1	1	2	2	1	0	2	1	2	1	2	2
4	75	Step Right	2	-	1	2	2	2	2	2	2	1	2	2	2	2	1	2	2	2	2	2	2
5	75	Hover	1	-	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
6	75	Land	1	-	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
7	75	Go Forward Half Metre	4	-	0	1	4	3	3	4	2	2	2	2	3	0	2	2	2	0	3	3	3
8	75	Go Backward One Metre	4	-	2	1	4	4	4	4	3	2	3	3	2	0	2	1	3	2	4	4	3
9	75	Hover One Metre	3	-	1	3	2	2	3	2	3	2	2	2	3	1	2	2	2	2	2	3	2
10	75	Step Left Half Metre	4	-	3	3	4	2	2	4	4	3	2	2	3	0	2	4	3	2	4	4	2
11	75	Step Right One Metre	4	-	1	4	3	4	4	4	4	4	3	3	4	2	4	4	4	4	4	4	4
12	75	Stop	1	-	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1
1	80	Go Forward	2	-	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1
2	80	Go Backward	2	-	0	1	1	2	2	0	1	1	0	0	1	0	0	0	0	0	2	2	1
3	80	Step Left	2	-	1	0	1	0	2	1	2	0	1	1	0	0	0	1	0	0	1	1	1
4	80	Step Right	2	-	1	1	1	1	2	2	2	2	1	2	1	1	1	1	1	1	2	2	2
5	80	Hover	1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	80	Land	1	-	0	1	1	1	0	1	1	1	0	1	0	0	1	1	0	1	1	0	0
7	80	Go Forward Half Metre	4	-	0	2	3	1	2	2	2	1	1	2	2	0	0	2	0	2	3	0	1
8	80	Go Backward One Metre	4	-	0	2	3	2	4	3	4	0	0	2	1	1	1	2	2	2	2	1	2
9	80	Hover One Metre	3	-	0	2	2	2	2	1	2	2	0	1	2	1	2	2	1	2	1	1	2
10	80	Step Left Half Metre	4	-	1	1	2	1	2	3	3	0	1	3	2	2	2	3	2	2	3	4	1
11	80	Step Right One Metre	4	-	0	4	3	2	4	4	4	2	2	3	4	3	2	2	3	1	2	3	2
12	80	Stop	1	-	0	0	1	0	1	0	1	1	1	0	0	0	1	1	1	0	0	1	0
1	85	Go Forward	2	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	85	Go Backward	2	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	85	Step Left	2	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	85	Step Right	2	-	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
5	85	Hover	1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	85	Land	1	-	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
7	85	Go Forward Half Metre	4	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	85	Go Backward One Metre	4	-	0	0	0	0	2	0	2	0	0	0	0	0	0	0	0	0	0	0	0
9	85	Hover One Metre	3	-	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0
10	85	Step Left Half Metre	4	-	0	0	0	1	1	0	2	1	2	0	0	0	0	0	0	0	0	0	0
11	85	Step Right One Metre	4	-	0	0	0	1	3	0	3	0	3	0	0	0	0	0	0	0	0	0	0
12	85	Stop	1	-	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure B.2: VNLC result table.

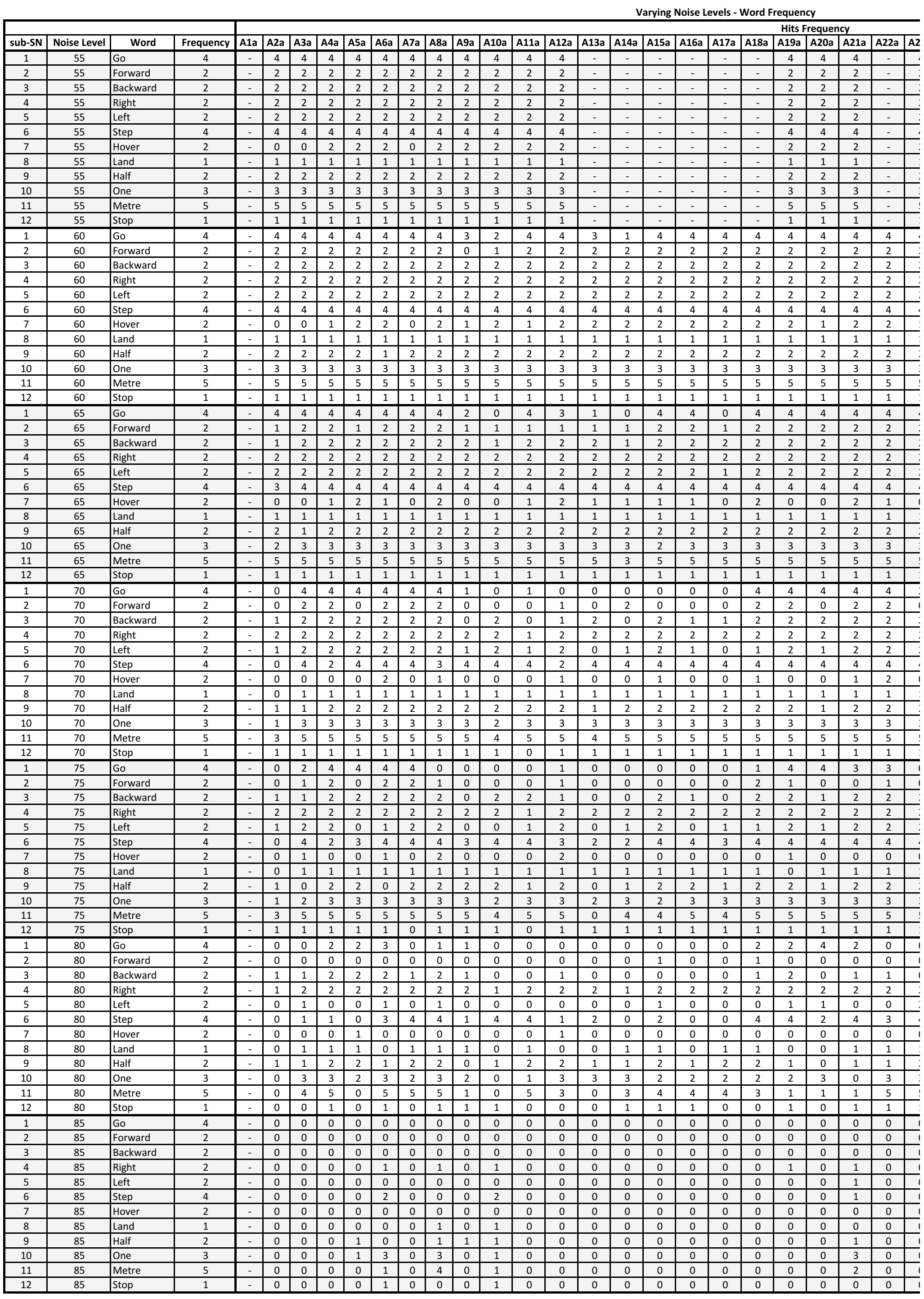
Speech Command - Continues



s																	
A21a	A22a	A23a	A24a	A25a	A26a	A27a	A28a	A29a	A30a	A31a	A32a	A33a	A34a	A35a	A36a	A37a	Average
2	-	2	-	-	-	-	2	-	2	2	2	2	2	2	2	-	2.0000
2	-	2	-	-	-	-	2	-	2	2	2	2	2	2	2	-	2.0000
2	-	2	-	-	-	-	2	-	2	2	2	2	2	2	2	-	2.0000
2	-	2	-	-	-	-	2	-	2	2	2	2	2	2	2	-	2.0000
1	-	1	-	-	-	-	1	-	1	1	1	1	1	1	1	-	0.8696
1	-	1	-	-	-	-	1	-	1	1	1	1	1	1	1	-	1.0000
4	-	4	-	-	-	-	4	-	4	3	4	4	4	4	2	-	3.8696
4	-	4	-	-	-	-	4	-	4	3	4	4	4	4	4	-	3.9565
3	-	3	-	-	-	-	3	-	3	3	3	2	3	3	3	-	2.8261
4	-	4	-	-	-	-	4	-	4	4	4	4	4	4	4	-	4.0000
4	-	4	-	-	-	-	4	-	4	4	4	4	4	4	4	-	4.0000
1	-	1	-	-	-	-	1	-	1	1	1	1	1	1	1	-	1.0000
2	2	2	-	2	2	2	2	2	2	1	1	2	2	2	1	2	1.8286
2	2	2	-	2	2	2	2	1	2	2	1	2	1	2	1	2	1.8571
2	2	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2.0000
2	2	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2.0000
1	1	0	-	1	1	1	0	1	1	1	1	1	0	0	0	1	0.6857
1	1	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
4	4	4	-	4	4	4	4	4	4	3	3	4	4	4	2	4	3.8000
4	4	4	-	3	4	4	4	4	4	3	3	4	4	4	2	4	3.7714
3	3	3	-	3	3	3	3	3	2	3	2	2	3	3	2	2	2.7429
4	4	4	-	4	4	4	4	4	4	4	4	4	4	4	4	4	3.9714
4	4	4	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4.0000
1	1	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
2	2	2	2	2	2	2	2	2	2	0	0	2	1	1	0	2	1.6389
2	2	2	2	2	2	1	2	1	2	1	0	2	1	1	0	2	1.5833
2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	1.9444
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2.0000
1	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0.2222
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
4	4	4	4	4	4	4	3	4	2	3	2	4	4	2	2	3	3.3056
4	4	4	4	4	4	3	4	3	3	3	2	4	3	4	2	4	3.5278
3	3	2	3	3	2	3	2	3	2	2	2	3	2	2	2	2	2.4444
4	4	4	4	4	4	4	4	4	4	3	3	4	4	4	4	4	3.9444
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3.9722
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0.9722
2	2	0	2	2	0	0	1	1	1	1	0	1	1	0	0	1	0.9444
2	2	2	2	0	2	0	0	1	1	0	0	2	1	1	0	2	1.1667
2	2	2	1	1	2	1	1	2	2	1	2	2	2	2	2	2	1.5556
2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	1.9167
1	1	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0.2222
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.9722
4	4	2	4	3	2	2	3	3	2	3	2	3	2	2	2	2	2.6389
4	4	4	4	2	1	2	3	3	2	2	2	4	3	4	2	4	3.0000
2	3	2	3	2	2	3	2	2	2	2	2	2	2	2	2	2	2.1389
4	4	4	4	4	4	4	4	4	4	2	3	4	4	4	4	4	3.7778
4	4	4	4	4	4	4	4	4	3	3	4	4	4	4	4	4	3.8056
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.9722
1	2	0	2	0	0	0	0	0	2	0	0	1	0	0	0	0	0.6389
2	2	1	2	0	1	0	0	0	1	0	0	2	1	0	0	0	0.9167
2	2	2	2	1	2	1	0	2	2	0	1	2	2	2	0	2	1.4444
2	2	2	2	1	2	2	2	2	2	1	2	2	2	1	2	2	1.8333
0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0.1389
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0.9167
2	2	2	3	0	2	2	2	2	2	2	2	2	2	2	2	2	2.0556
4	4	3	4	2	3	2	3	2	2	2	2	4	2	2	2	2	2.6667
2	2	2	3	2	2	2	2	2	1	2	2	2	2	0	2	2	2.0278
4	4	4	4	2	3	2	3	3	4	2	3	4	4	4	3	4	3.0556
4	4	4	4	4	4	4	4	4	4	3	4	4	4	3	3	4	3.6944
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0.9167
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0.1667
0	1	0	2	0	0	0	0	0	0	0	0	2	0	0	0	0	0.5278
1	1	2	1	0	1	0	1	1	2	0	0	2	1	0	0	1	0.7500
2	1	2	1	1	2	1	2	2	2	1	1	2	1	0	0	2	1.3889
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0.0556
1	1	1	1	1	1	1	1	0	1	1	0	1	0	1	0	1	0.6667
1	2	2	2	1	2	2	1	1	2	1	1	3	1	2	0	2	1.4444
2	2	2	4	0	0	1	2	2	1	2	0	4	1	0	1	0	1.6111
0	2	2	2	1	0	2	2	2	0	2	2	2	2	1	1	2	1.4722
3	2	4	1	1	0	2	1	1	4	2	2	3	2	2	1	2	1.9722
2	4	4	3	2	2	3	3	3	3	3	3	4	2	2	1	3	2.6944
1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	1	1	0.6389
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0.1111
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0833
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0.0556
1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0.2222
2	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0.2222
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2222
4	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	0.4722
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0556

Figure B.3: VNLC result table continues.

B.2.2 Word Freqency

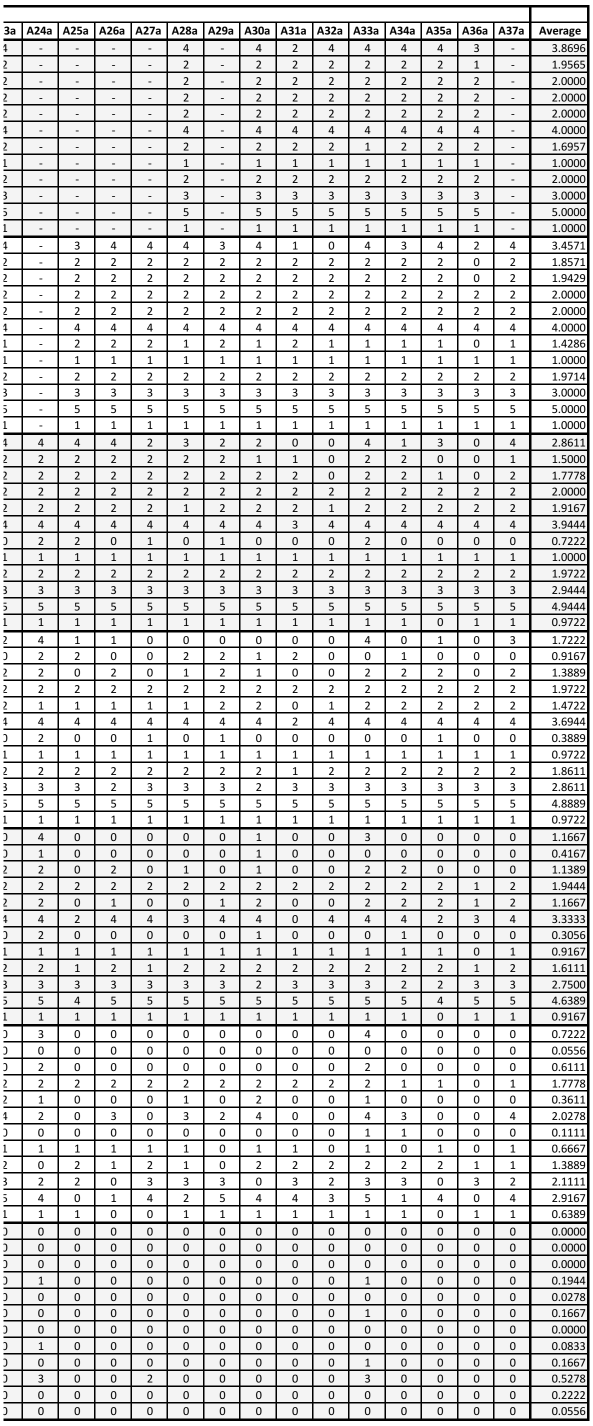


Varying Noise Levels - Word Frequency

				Hits Frequency																					
sub-SN	Noise Level	Word	Frequency	A1a	A2a	A3a	A4a	A5a	A6a	A7a	A8a	A9a	A10a	A11a	A12a	A13a	A14a	A15a	A16a	A17a	A18a	A19a	A20a	A21a	A22a
1	55	Go	4	-	4	4	4	4	4	4	4	4	4	4	4	-	-	-	-	-	-	4	4	4	-
2	55	Forward	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2	2	-
3	55	Backward	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2	2	-
4	55	Right	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2	2	-
5	55	Left	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2	2	-
6	55	Step	4	-	4	4	4	4	4	4	4	4	4	4	4	-	-	-	-	-	-	4	4	4	-
7	55	Hover	2	-	0	0	2	2	2	0	2	2	2	2	2	-	-	-	-	-	-	2	2	2	-
8	55	Land	1	-	1	1	1	1	1	1	1	1	1	1	1	-	-	-	-	-	-	1	1	1	-
9	55	Half	2	-	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	2	2	2	-
10	55	One	3	-	3	3	3	3	3	3	3	3	3	3	3	-	-	-	-	-	-	3	3	3	-
11	55	Metre	5	-	5	5	5	5	5	5	5	5	5	5	5	-	-	-	-	-	-	5	5	5	-
12	55	Stop	1	-	1	1	1	1	1	1	1	1	1	1	1	-	-	-	-	-	-	1	1	1	-
1	60	Go	4	-	4	4	4	4	4	4	4	3	2	4	4	3	1	4	4	4	4	4	4	4	4
2	60	Forward	2	-	2	2	2	2	2	2	2	0	1	2	2	2	2	2	2	2	2	2	2	2	2
3	60	Backward	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	60	Right	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	60	Left	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
6	60	Step	4	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
7	60	Hover	2	-	0	0	1	2	2	0	2	1	2	1	2	2	2	2	2	2	2	2	1	2	2
8	60	Land	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	60	Half	2	-	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
10	60	One	3	-	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
11	60	Metre	5	-	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
12	60	Stop	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	65	Go	4	-	4	4	4	4	4	4	4	2	0	4	3	1	0	4	4	0	4	4	4	4	4
2	65	Forward	2	-	1	2	2	1	2	2	2	1	1	1	1	1	1	2	2	1	2	2	2	2	2
3	65	Backward	2	-	1	2	2	2	2	2	2	2	1	2	2	2	1	2	2	2	2	2	2	2	2
4	65	Right	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	65	Left	2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2
6	65	Step	4	-	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
7	65	Hover	2	-	0	0	1	2	1	0	2	0	0	1	2	1	1	1	1	0	2	0	0	2	1
8	65	Land	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	65	Half	2	-	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
10	65	One	3	-	2	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3
11	65	Metre	5	-	5	5	5	5	5	5	5	5	5	5	5	5	3	5	5	5	5	5	5	5	5
12	65	Stop	1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	70	Go	4	-	0	4	4	4	4	4	4	1	0	1	0	0	0	0	0	0	4	4	4	4	4
2	70	Forward	2	-	0	2	2	0	2	2	2	0	0	0	1	0	2	0	0	0	2	2	0	2	2
3	70	Backward	2	-	1	2	2	2	2	2	2	0	2	0	1	2	0	2	1	1	2	2	2	2	2
4	70	Right	2	-	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	2
5	70	Left	2	-	1	2	2	2	2	2	2	1	2	1	2	0	1	2	1	0	1	2	1	2	2
6	70	Step	4	-	0	4	2	4	4	4	3	4	4	4	2	4	4	4	4	4	4	4	4	4	4
7	70	Hover	2	-	0	0	0	0	2	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	2
8	70	Land	1	-	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	70	Half	2	-	1	1	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	1	2	2
10	70	One	3	-	1	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3	3	3	3	3
11	70	Metre	5	-	3	5	5	5	5	5	5	5	4	5	5	4	5	5	5	5	5	5	5	5	5
12	70	Stop	1	-	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	75	Go	4	-	0	2	4	4	4	4	0	0	0	0	1	0	0	0	0	0	1	4	4	3	3
2	75	Forward	2	-	0	1	2	0	2	2	1	0	0	0	1	0	0	0	0	0	2	1	0	0	1
3	75	Backward	2	-	1	1	2	2	2	2	2	0	2	2	1	0	0	2	1	0	2	2	1	2	2
4	75	Right	2	-	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	2
5	75	Left	2	-	1	2	2	0	1	2	2	0	0	1	2	0	1	2	0	1	1	2	1	2	2
6	75	Step	4	-	0	4	2	3	4	4	4	3	4	4	3	2	2	4	4	3	4	4	4	4	4
7	75	Hover	2	-	0	1	0	0	1	0	2	0	0	0	2	0	0	0	0	0	0	1	0	0	0
8	75	Land	1	-	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
9	75	Half	2	-	1	0	2	2	0	2	2	2	2	1	2	0	1	2	2	1	2	2	1	2	2
10	75	One	3	-	1	2	3	3	3	3	3	3	2	3	3	2	3	2	3	3	3	3	3	3	3
11	75	Metre	5	-	3	5	5	5	5	5	5	5	4	5	5	0	4	4	5	4	5	5	5	5	5
12	75	Stop	1	-	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	80	Go	4	-	0	0	2	2	3	0	1	1	0	0	0	0	0	0	0	0	2	2	4	2	0
2	80	Forward	2	-	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
3	80	Backward	2	-	1	1	2	2	2	1	2	1	0	0	1	0	0	0	0	0	1	2	0	1	1
4	80	Right	2	-	1	2	2	2	2	2	2	2	1	2	2	2	1	2	2	2	2	2	2	2	2
5	80	Left	2	-	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0
6	80	Step	4	-	0	1	1	0	3	4	4	1	4	4	1	2	0	2	0	0	4	4	2	4	3
7	80	Hover	2	-	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
8	80	Land	1	-	0	1	1	1	0	1	1	1	0	1	0	0	1	1	0	1	1	0	0	1	1
9	80	Half	2	-	1	1	2	2	1	2	2	0	1	2	2	1	1	2	1	2	2	1	0	1	1
10	80	One	3	-	0	3	3	2	3	2	3	2	0	1	3	3	3	2	2	2	2	2	3	0	3
11	80	Metre	5	-	0	4	5	0	5	5	5	1	0	5	3	0	3	4	4	4	3	1	1	1	5
12	80	Stop	1	-	0	0	1	0	1	0	1	1	1	0	0	0	1	1	1	0	0	1	0	1	1
1	85	Go	4	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	85	Forward	2	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	85	Backward	2	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	85	Right	2	-	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0
5	85	Left	2	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
6	85	Step	4	-	0	0	0	0	2	0	0	0	2	0	0	0	0	0	0	0	0	0	0	1	0
7	85	Hover	2	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	85	Land	1	-	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	85	Half	2	-	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0
10	85	One	3	-	0	0	0	1	3	0	3	0	1	0	0	0	0	0	0	0	0	0	0	3	0
11	85	Metre	5	-	0	0	0	0	1	0	4	0	1	0	0	0	0	0	0	0	0	0	0	2	0
12	85	Stop	1	-	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure B.4: VNL-WF result table.

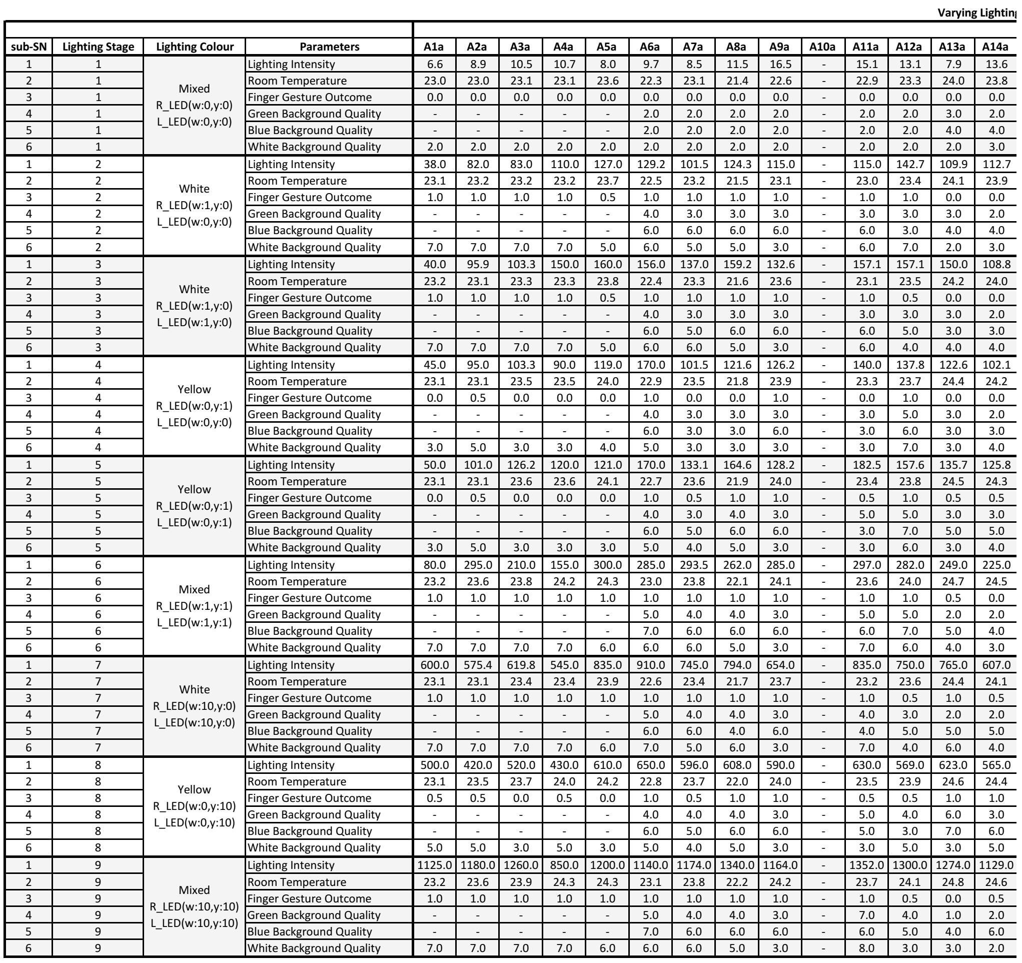
Word freqency result - continues



3a	A24a	A25a	A26a	A27a	A28a	A29a	A30a	A31a	A32a	A33a	A34a	A35a	A36a	A37a	Average
4	-	-	-	-	4	-	4	2	4	4	4	4	3	-	3.8696
2	-	-	-	-	2	-	2	2	2	2	2	2	1	-	1.9565
2	-	-	-	-	2	-	2	2	2	2	2	2	2	-	2.0000
2	-	-	-	-	2	-	2	2	2	2	2	2	2	-	2.0000
2	-	-	-	-	2	-	2	2	2	2	2	2	2	-	2.0000
4	-	-	-	-	4	-	4	4	4	4	4	4	4	-	4.0000
2	-	-	-	-	2	-	2	2	2	1	2	2	2	-	1.6957
1	-	-	-	-	1	-	1	1	1	1	1	1	1	-	1.0000
2	-	-	-	-	2	-	2	2	2	2	2	2	2	-	2.0000
3	-	-	-	-	3	-	3	3	3	3	3	3	3	-	3.0000
5	-	-	-	-	5	-	5	5	5	5	5	5	5	-	5.0000
1	-	-	-	-	1	-	1	1	1	1	1	1	1	-	1.0000
4	-	3	4	4	4	3	4	1	0	4	3	4	2	4	3.4571
2	-	2	2	2	2	2	2	2	2	2	2	2	0	2	1.8571
2	-	2	2	2	2	2	2	2	2	2	2	2	0	2	1.9429
2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2.0000
2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2.0000
4	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4.0000
1	-	2	2	2	1	2	1	2	1	1	1	1	0	1	1.4286
1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	1.9714
3	-	3	3	3	3	3	3	3	3	3	3	3	3	3	3.0000
5	-	5	5	5	5	5	5	5	5	5	5	5	5	5	5.0000
1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
4	4	4	4	2	3	2	2	0	0	4	1	3	0	4	2.8611
2	2	2	2	2	2	2	1	1	0	2	2	0	0	1	1.5000
2	2	2	2	2	2	2	2	2	0	2	2	1	0	2	1.7778
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2.0000
2	2	2	2	2	1	2	2	2	1	2	2	2	2	2	1.9167
4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	3.9444
0	2	2	0	1	0	1	0	0	0	2	0	0	0	0	0.7222
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1.9722
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2.9444
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4.9444
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0.9722
2	4	1	1	0	0	0	0	0	0	4	0	1	0	3	1.7222
0	2	2	0	0	2	2	1	2	0	0	1	0	0	0	0.9167
2	2	0	2	0	1	2	1	0	0	2	2	2	0	2	1.3889
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1.9722
2	1	1	1	1	1	2	2	0	1	2	2	2	2	2	1.4722
4	4	4	4	4	4	4	4	2	4	4	4	4	4	4	3.6944
0	2	0	0	1	0	1	0	0	0	0	0	1	0	0	0.3889
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.9722
2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	1.8611
3	3	3	2	3	3	3	2	3	3	3	3	3	3	3	2.8611
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4.8889
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.9722
0	4	0	0	0	0	0	1	0	0	3	0	0	0	0	1.1667
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0.4167
2	2	0	2	0	1	0	1	0	0	2	2	0	0	0	1.1389
2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1.9444
2	2	0	1	0	0	1	2	0	0	2	2	2	1	2	1.1667
4	4	2	4	4	3	4	4	0	4	4	4	2	3	4	3.3333
0	2	0	0	0	0	0	1	0	0	0	1	0	0	0	0.3056
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0.9167
2	2	1	2	1	2	2	2	2	2	2	2	2	1	2	1.6111
3	3	3	3	3	3	3	2	3	3	3	2	2	3	3	2.7500
5	5	4	5	5	5	5	5	5	5	5	5	4	5	5	4.6389
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0.9167
0	3	0	0	0	0	0	0	0	0	4	0	0	0	0	0.7222
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0556
0	2	0	0	0	0	0	0	0	0	2	0	0	0	0	0.6111
2	2	2	2	2	2	2	2	2	2	2	1	1	0	1	1.7778
2	1	0	0	0	1	0	2	0	0	1	0	0	0	0	0.3611
4	2	0	3	0	3	2	4	0	0	4	3	0	0	4	2.0278
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0.1111
1	1	1	1	1	1	0	1	1	0	1	0	1	0	1	0.6667
2	0	2	1	2	1	0	2	2	2	2	2	2	1	1	1.3889
3	2	2	0	3	3	3	0	3	2	3	3	0	3	2	2.1111
5	4	0	1	4	2	5	4	4	3	5	1	4	0	4	2.9167
1	1	1	0	0	1	1	1	1	1	1	1	0	1	1	0.6389
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0.1944
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0278
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0.1667
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0833
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0.1667
0	3	0	0	2	0	0	0	0	0	3	0	0	0	0	0.5278
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2222
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0556

Figure B.5: VNL-WF result table continues.

B.2.3 Gesture result



																	Varying Lightin
sub-SN	Lighting Stage	Lighting Colour	Parameters	A1a	A2a	A3a	A4a	A5a	A6a	A7a	A8a	A9a	A10a	A11a	A12a	A13a	A14a
1	1	Mixed R_LED(w:0,y:0) L_LED(w:0,y:0)	Lighting Intensity	6.6	8.9	10.5	10.7	8.0	9.7	8.5	11.5	16.5	-	15.1	13.1	7.9	13.6
2	1		Room Temperature	23.0	23.0	23.1	23.1	23.6	22.3	23.1	21.4	22.6	-	22.9	23.3	24.0	23.8
3	1		Finger Gesture Outcome	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-	0.0	0.0	0.0	0.0
4	1		Green Background Quality	-	-	-	-	-	2.0	2.0	2.0	2.0	-	2.0	2.0	3.0	2.0
5	1		Blue Background Quality	-	-	-	-	-	2.0	2.0	2.0	2.0	-	2.0	2.0	4.0	4.0
6	1		White Background Quality	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	-	2.0	2.0	2.0	3.0
1	2	White R_LED(w:1,y:0) L_LED(w:0,y:0)	Lighting Intensity	38.0	82.0	83.0	110.0	127.0	129.2	101.5	124.3	115.0	-	115.0	142.7	109.9	112.7
2	2		Room Temperature	23.1	23.2	23.2	23.2	23.7	22.5	23.2	21.5	23.1	-	23.0	23.4	24.1	23.9
3	2		Finger Gesture Outcome	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	-	1.0	1.0	0.0	0.0
4	2		Green Background Quality	-	-	-	-	-	4.0	3.0	3.0	3.0	-	3.0	3.0	3.0	2.0
5	2		Blue Background Quality	-	-	-	-	-	6.0	6.0	6.0	6.0	-	6.0	3.0	4.0	4.0
6	2		White Background Quality	7.0	7.0	7.0	7.0	5.0	6.0	5.0	5.0	3.0	-	6.0	7.0	2.0	3.0
1	3	White R_LED(w:1,y:0) L_LED(w:1,y:0)	Lighting Intensity	40.0	95.9	103.3	150.0	160.0	156.0	137.0	159.2	132.6	-	157.1	157.1	150.0	108.8
2	3		Room Temperature	23.2	23.1	23.3	23.3	23.8	22.4	23.3	21.6	23.6	-	23.1	23.5	24.2	24.0
3	3		Finger Gesture Outcome	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	-	1.0	0.5	0.0	0.0
4	3		Green Background Quality	-	-	-	-	-	4.0	3.0	3.0	3.0	-	3.0	3.0	3.0	2.0
5	3		Blue Background Quality	-	-	-	-	-	6.0	5.0	6.0	6.0	-	6.0	5.0	3.0	3.0
6	3		White Background Quality	7.0	7.0	7.0	7.0	5.0	6.0	6.0	5.0	3.0	-	6.0	4.0	4.0	4.0
1	4	Yellow R_LED(w:0,y:1) L_LED(w:0,y:0)	Lighting Intensity	45.0	95.0	103.3	90.0	119.0	170.0	101.5	121.6	126.2	-	140.0	137.8	122.6	102.1
2	4		Room Temperature	23.1	23.1	23.5	23.5	24.0	22.9	23.5	21.8	23.9	-	23.3	23.7	24.4	24.2
3	4		Finger Gesture Outcome	0.0	0.5	0.0	0.0	0.0	1.0	0.0	0.0	1.0	-	0.0	1.0	0.0	0.0
4	4		Green Background Quality	-	-	-	-	-	4.0	3.0	3.0	3.0	-	3.0	5.0	3.0	2.0
5	4		Blue Background Quality	-	-	-	-	-	6.0	3.0	3.0	6.0	-	3.0	6.0	3.0	3.0
6	4		White Background Quality	3.0	5.0	3.0	3.0	4.0	5.0	3.0	3.0	3.0	-	3.0	7.0	3.0	4.0
1	5	Yellow R_LED(w:0,y:1) L_LED(w:0,y:1)	Lighting Intensity	50.0	101.0	126.2	120.0	121.0	170.0	133.1	164.6	128.2	-	182.5	157.6	135.7	125.8
2	5		Room Temperature	23.1	23.1	23.6	23.6	24.1	22.7	23.6	21.9	24.0	-	23.4	23.8	24.5	24.3
3	5		Finger Gesture Outcome	0.0	0.5	0.0	0.0	0.0	1.0	0.5	1.0	1.0	-	0.5	1.0	0.5	0.5
4	5		Green Background Quality	-	-	-	-	-	4.0	3.0	4.0	3.0	-	5.0	5.0	3.0	3.0
5	5		Blue Background Quality	-	-	-	-	-	6.0	5.0	6.0	6.0	-	3.0	7.0	5.0	5.0
6	5		White Background Quality	3.0	5.0	3.0	3.0	3.0	5.0	4.0	5.0	3.0	-	3.0	6.0	3.0	4.0
1	6	Mixed R_LED(w:1,y:1) L_LED(w:1,y:1)	Lighting Intensity	80.0	295.0	210.0	155.0	300.0	285.0	293.5	262.0	285.0	-	297.0	282.0	249.0	225.0
2	6		Room Temperature	23.2	23.6	23.8	24.2	24.3	23.0	23.8	22.1	24.1	-	23.6	24.0	24.7	24.5
3	6		Finger Gesture Outcome	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-	1.0	1.0	0.5	0.0
4	6		Green Background Quality	-	-	-	-	-	5.0	4.0	4.0	3.0	-	5.0	5.0	2.0	2.0
5	6		Blue Background Quality	-	-	-	-	-	7.0	6.0	6.0	6.0	-	6.0	7.0	5.0	4.0
6	6		White Background Quality	7.0	7.0	7.0	7.0	6.0	6.0	6.0	5.0	3.0	-	7.0	6.0	4.0	3.0
1	7	White R_LED(w:10,y:0) L_LED(w:10,y:0)	Lighting Intensity	600.0	575.4	619.8	545.0	835.0	910.0	745.0	794.0	654.0	-	835.0	750.0	765.0	607.0
2	7		Room Temperature	23.1	23.1	23.4	23.4	23.9	22.6	23.4	21.7	23.7	-	23.2	23.6	24.4	24.1
3	7		Finger Gesture Outcome	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-	1.0	0.5	1.0	0.5
4	7		Green Background Quality	-	-	-	-	-	5.0	4.0	4.0	3.0	-	4.0	3.0	2.0	2.0
5	7		Blue Background Quality	-	-	-	-	-	6.0	6.0	4.0	6.0	-	4.0	5.0	5.0	5.0
6	7		White Background Quality	7.0	7.0	7.0	7.0	6.0	7.0	5.0	6.0	3.0	-	7.0	4.0	6.0	4.0
1	8	Yellow R_LED(w:0,y:10) L_LED(w:0,y:10)	Lighting Intensity	500.0	420.0	520.0	430.0	610.0	650.0	596.0	608.0	590.0	-	630.0	569.0	623.0	565.0
2	8		Room Temperature	23.1	23.5	23.7	24.0	24.2	22.8	23.7	22.0	24.0	-	23.5	23.9	24.6	24.4
3	8		Finger Gesture Outcome	0.5	0.5	0.0	0.5	0.0	1.0	0.5	1.0	1.0	-	0.5	0.5	1.0	1.0
4	8		Green Background Quality	-	-	-	-	-	4.0	4.0	4.0	3.0	-	5.0	4.0	6.0	3.0
5	8		Blue Background Quality	-	-	-	-	-	6.0	5.0	6.0	6.0	-	5.0	3.0	7.0	6.0
6	8		White Background Quality	5.0	5.0	3.0	5.0	3.0	5.0	4.0	5.0	3.0	-	3.0	5.0	3.0	5.0
1	9	Mixed R_LED(w:10,y:10) L_LED(w:10,y:10)	Lighting Intensity	1125.0	1180.0	1260.0	850.0	1200.0	1140.0	1174.0	1340.0	1164.0	-	1352.0	1300.0	1274.0	1129.0
2	9		Room Temperature	23.2	23.6	23.9	24.3	24.3	23.1	23.8	22.2	24.2	-	23.7	24.1	24.8	24.6
3	9		Finger Gesture Outcome	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-	1.0	0.5	0.0	0.5
4	9		Green Background Quality	-	-	-	-	-	5.0	4.0	4.0	3.0	-	7.0	4.0	1.0	2.0
5	9		Blue Background Quality	-	-	-	-	-	7.0	6.0	6.0	6.0	-	6.0	5.0	4.0	6.0
6	9		White Background Quality	7.0	7.0	7.0	7.0	6.0	6.0	6.0	5.0	3.0	-	8.0	3.0	3.0	2.0

Figure B.6: VLL result table.

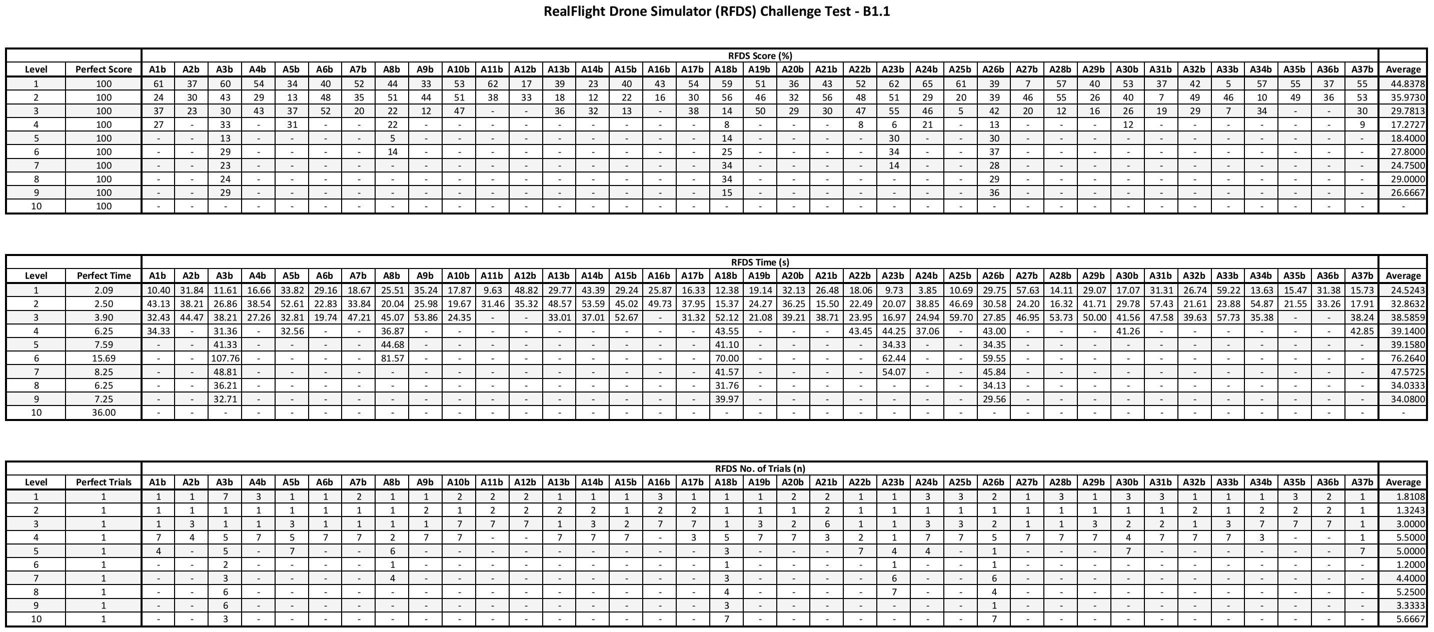
Gesture result - continues

g Levels - Finger Gestures

Parameter values																							
A15a	**A16a**	**A17a**	**A18a**	**A19a**	**A20a**	**A21a**	**A22a**	**A23a**	**A24a**	**A25a**	**A26a**	**A27a**	**A28a**	**A29a**	**A30a**	**A31a**	**A32a**	**A33a**	**A34a**	**A35a**	**A36a**	**A37a**	**Average**
11.8	15.6	7.2	12.2	14.5	16.5	10.0	8.7	12.6	10.5	13.5	7.0	6.5	9.5	11.2	9.2	13.7	13.5	7.7	8.0	9.5	8.5	11.3	10.8139
23.6	23.8	23.8	24.0	23.8	24.2	23.7	23.8	22.6	23.5	23.1	23.4	23.3	23.4	23.1	23.6	23.5	23.7	22.3	22.9	23.2	23.2	23.1	23.2722
0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.5	0.5	0.0	0.5	0.0	0.0	0.0	0.5	0.5	0.0	0.5	1.0	0.0	0.5	1.0	0.1806
1.0	2.0	1.0	1.0	1.0	4.0	4.0	1.0	5.0	5.0	1.0	1.0	1.0	1.0	1.0	4.0	2.0	0.0	6.0	7.0	0.0	5.0	6.0	2.4839
1.0	2.0	1.0	1.0	1.0	4.0	4.0	1.0	4.0	3.0	1.0	1.0	1.0	1.0	1.0	3.0	2.0	0.0	4.0	6.0	0.0	6.0	8.0	2.4516
1.0	3.0	1.0	1.0	1.0	4.0	4.0	1.0	3.0	4.0	3.0	5.0	1.0	1.0	1.0	5.0	5.0	0.0	5.0	5.0	0.0	4.0	7.0	2.5556
114.9	95.6	110.0	95.6	129.2	125.9	120.0	90.4	121.5	73.4	121.2	99.2	88.8	88.7	96.6	88.8	106.5	91.3	92.8	93.3	100.5	105.0	108.5	104.1111
23.7	23.9	23.9	24.1	23.9	24.4	23.9	23.9	22.8	23.6	23.2	23.5	23.5	23.5	23.2	23.8	23.6	23.8	22.8	23.0	23.5	23.3	23.3	23.4222
0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.5	0.5	1.0	0.5	1.0	0.5	0.0	0.5	1.0	0.5	0.5	0.5	0.5	0.5	0.5	1.0	0.6944
2.0	6.0	8.0	1.0	4.0	4.0	5.0	2.0	3.0	5.0	2.0	5.0	2.0	2.0	2.0	4.0	3.0	1.0	4.0	5.0	2.0	4.0	8.0	3.4839
2.0	8.0	9.0	2.0	6.0	4.0	7.0	2.0	3.0	6.0	2.0	6.0	2.0	2.0	2.0	5.0	3.0	1.0	4.0	5.0	2.0	5.0	7.0	4.3871
2.0	7.0	10.0	3.0	5.0	6.0	6.0	5.0	5.0	7.0	5.0	7.0	5.0	4.0	5.0	6.0	5.0	4.0	6.0	7.0	5.0	6.0	6.0	5.4722
140.1	120.5	120.4	127.3	135.7	136.0	140.4	123.7	124.6	97.5	127.4	117.2	96.4	110.0	114.4	110.8	122.0	107.6	117.1	114.7	119.8	128.7	127.3	124.6278
23.8	24.0	24.0	24.2	24.0	24.5	24.0	24.0	23.0	23.7	23.4	23.8	23.6	23.6	23.3	23.9	23.7	24.1	23.0	23.3	23.7	23.4	23.4	23.5500
0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.5	1.0	1.0	1.0	0.5	0.5	0.5	1.0	0.5	0.5	0.5	0.5	0.0	0.5	1.0	0.6806
2.0	7.0	8.0	1.0	4.0	4.0	5.0	2.0	3.0	5.0	3.0	5.0	2.0	2.0	2.0	4.0	3.0	1.0	3.0	4.0	1.0	4.0	10.0	3.5161
2.0	8.0	9.0	2.0	6.0	4.0	7.0	2.0	3.0	6.0	5.0	6.0	2.0	2.0	4.0	5.0	3.0	1.0	5.0	6.0	1.0	5.0	8.0	4.5806
2.0	6.0	10.0	3.0	5.0	6.0	6.0	4.0	5.0	7.0	6.0	7.0	5.0	5.0	5.0	6.0	5.0	4.0	6.0	7.0	1.0	6.0	7.0	5.4167
98.7	93.0	117.3	105.8	112.0	100.0	120.1	102.0	109.2	90.6	131.8	85.5	100.4	111.6	98.7	98.9	118.4	99.1	120.4	113.5	111.3	103.8	117.6	109.2722
24.1	24.3	24.3	24.4	24.2	24.8	24.3	24.2	23.3	23.8	23.7	23.9	23.8	23.8	23.6	24.1	24.0	24.3	22.3	23.5	23.7	23.5	23.6	23.7333
0.0	0.0	0.5	0.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.5	0.5	0.5	1.0	1.0	0.4306
2.0	4.0	4.0	1.0	4.0	4.0	5.0	2.0	1.0	3.0	6.0	2.0	2.0	4.0	3.0	2.0	4.0	1.0	4.0	4.0	4.0	8.0	8.0	3.4839
2.0	3.0	6.0	1.0	6.0	5.0	7.0	2.0	1.0	4.0	5.0	2.0	2.0	5.0	3.0	4.0	5.0	1.0	5.0	6.0	5.0	6.0	7.0	4.0645
2.0	2.0	5.0	1.0	5.0	6.0	6.0	5.0	1.0	5.0	7.0	5.0	5.0	6.0	5.0	5.0	6.0	1.0	6.0	5.0	6.0	7.0	6.0	4.3611
130.9	116.1	124.8	122.0	129.0	120.0	125.9	116.7	112.8	109.5	132.4	116.2	112.5	116.4	120.8	113.7	137.7	114.6	136.0	130.1	128.4	133.4	137.9	126.4861
24.2	24.4	24.4	24.5	24.3	24.9	24.3	24.3	23.4	23.8	23.8	24.0	23.9	23.8	23.7	24.2	24.1	24.4	22.5	23.6	23.9	23.6	23.7	23.8167
0.0	0.5	1.0	0.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.5	1.0	0.5	0.5	1.0	0.5417
2.0	4.0	5.0	1.0	4.0	5.0	3.0	2.0	3.0	3.0	7.0	2.0	2.0	4.0	3.0	3.0	4.0	1.0	5.0	5.0	4.0	3.0	10.0	3.7097
2.0	5.0	7.0	1.0	6.0	7.0	4.0	2.0	1.0	4.0	5.0	5.0	2.0	5.0	4.0	4.0	6.0	1.0	6.0	7.0	5.0	3.0	8.0	4.6129
2.0	3.0	6.0	1.0	5.0	6.0	6.0	5.0	2.0	5.0	6.0	5.0	5.0	6.0	5.0	5.0	5.0	1.0	5.0	6.0	6.0	5.0	7.0	4.3889
225.0	205.0	230.0	234.0	212.0	225.0	233.0	234.0	225.0	215.0	258.0	201.0	203.0	220.0	206.0	229.0	256.0	215.0	256.0	233.0	249.0	255.0	249.0	235.7361
24.5	24.6	24.6	24.7	24.5	25.2	24.4	24.4	23.6	23.9	24.0	24.0	24.0	24.0	24.0	24.3	24.3	24.5	23.1	23.7	24.1	23.8	23.9	24.0278
0.5	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.5	1.0	0.5	0.5	0.5	0.5	1.0	0.5	1.0	1.0	0.8194
2.0	5.0	6.0	4.0	4.0	4.0	6.0	5.0	5.0	6.0	6.0	7.0	4.0	4.0	7.0	4.0	4.0	1.0	4.0	5.0	4.0	8.0	8.0	4.6129
5.0	7.0	8.0	4.0	6.0	6.0	7.0	7.0	7.0	7.0	7.0	8.0	6.0	6.0	8.0	5.0	6.0	4.0	5.0	7.0	6.0	9.0	9.0	6.3548
2.0	4.0	7.0	1.0	5.0	4.0	3.0	6.0	6.0	5.0	6.0	6.0	5.0	5.0	6.0	2.0	5.0	1.0	6.0	6.0	5.0	9.0	10.0	5.2500
660.0	616.0	712.0	709.0	658.0	623.0	644.0	686.0	620.0	620.0	714.0	645.0	688.0	706.0	698.0	636.0	708.0	650.0	755.0	740.0	737.0	783.0	753.0	694.3389
23.9	24.2	24.1	24.3	24.1	24.7	24.1	24.1	23.2	23.7	23.6	23.8	23.7	23.7	23.3	24.0	23.9	24.1	23.0	23.5	23.7	23.5	23.6	23.6500
1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.0	0.5	1.0	0.5	1.0	1.0	0.8056
3.0	7.0	5.0	2.0	4.0	4.0	5.0	3.0	5.0	6.0	4.0	6.0	3.0	2.0	3.0	3.0	3.0	1.0	5.0	5.0	3.0	7.0	10.0	4.0645
6.0	8.0	7.0	3.0	6.0	4.0	7.0	5.0	6.0	7.0	6.0	7.0	4.0	4.0	4.0	4.0	5.0	1.0	6.0	6.0	6.0	9.0	10.0	5.5484
3.0	6.0	6.0	4.0	5.0	6.0	6.0	4.0	7.0	5.0	7.0	8.0	5.0	5.0	5.0	5.0	6.0	1.0	5.0	7.0	5.0	9.0	10.0	5.7222
522.0	505.0	615.0	553.0	462.0	528.0	603.0	543.0	570.0	494.0	572.0	500.0	505.0	508.0	530.0	557.0	577.0	522.0	587.0	593.5	551.0	602.0	584.0	552.6250
24.4	24.5	24.5	24.6	24.4	25.0	24.4	24.4	23.5	23.9	23.9	24.0	24.0	24.0	23.8	24.3	24.1	24.5	22.7	23.7	24.0	23.8	23.8	23.9333
0.5	1.0	1.0	0.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.5	1.0	0.5	0.5	1.0	0.6944
5.0	6.0	6.0	3.0	5.0	5.0	6.0	5.0	7.0	7.0	7.0	7.0	7.0	4.0	4.0	3.0	4.0	1.0	4.0	7.0	4.0	6.0	10.0	5.0323
2.0	4.0	8.0	4.0	7.0	7.0	7.0	4.0	6.0	6.0	6.0	6.0	5.0	5.0	5.0	4.0	6.0	4.0	5.0	8.0	6.0	4.0	8.0	5.5161
2.0	4.0	7.0	2.0	6.0	6.0	3.0	3.0	5.0	5.0	5.0	4.0	6.0	6.0	4.0	5.0	5.0	1.0	6.0	6.0	5.0	5.0	7.0	4.5000
1259.0	1052.0	1204.0	1195.0	1214.0	1105.0	1204.0	1325.0	1236.0	1112.0	1228.0	1057.0	1135.0	1230.0	1205.0	1195.0	1332.0	1163.0	1315.0	1314.0	1197.0	1336.0	1267.0	1204.6667
24.6	24.7	24.7	24.8	24.6	25.2	24.5	24.5	23.7	24.0	24.1	24.1	24.1	24.1	24.1	24.4	24.4	24.6	23.1	23.8	24.1	23.9	24.0	24.1083
0.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	0.5	0.0	0.5	0.5	0.5	0.5	1.0	0.7917
2.0	7.0	6.0	5.0	4.0	4.0	6.0	5.0	5.0	7.0	5.0	7.0	3.0	4.0	6.0	4.0	4.0	1.0	4.0	5.0	3.0	5.0	10.0	4.5806
2.0	6.0	8.0	5.0	6.0	6.0	7.0	7.0	7.0	6.0	6.0	8.0	7.0	6.0	8.0	6.0	6.0	1.0	5.0	5.0	6.0	6.0	10.0	6.0000
2.0	4.0	7.0	4.0	5.0	4.0	5.0	5.0	6.0	3.0	7.0	6.0	6.0	5.0	7.0	5.0	5.0	1.0	6.0	7.0	3.0	7.0	8.0	5.2222

Figure B.7: VLL result table continues.

B.3 Task B Results Tables



RealFlight Drone Simulator (RFDS) Challenge Test - B1.1

		RFDS Score (%)																																				
Level	**Perfect Score**	**A1b**	**A2b**	**A3b**	**A4b**	**A5b**	**A6b**	**A7b**	**A8b**	**A9b**	**A10b**	**A11b**	**A12b**	**A13b**	**A14b**	**A15b**	**A16b**	**A17b**	**A18b**	**A19b**	**A20b**	**A21b**	**A22b**	**A23b**	**A24b**	**A25b**	**A26b**	**A27b**	**A28b**	**A29b**	**A30b**	**A31b**	**A32b**	**A33b**	**A34b**	**A35b**	**A36b**	**A37b**	**Average**
1	100	61	37	60	54	34	40	52	44	33	53	62	17	39	23	40	43	54	59	51	36	43	52	62	65	61	39	7	57	40	53	37	42	5	57	55	37	55	44.8378
2	100	24	30	43	29	13	48	35	51	44	51	38	33	18	12	22	16	30	56	46	32	56	48	51	29	20	39	46	55	26	40	7	49	46	10	49	36	53	35.9730
3	100	37	23	30	43	37	52	20	22	12	47	-	-	36	32	13	-	38	14	50	29	30	47	55	46	5	42	20	12	16	26	19	29	7	34	-	-	30	29.7813
4	100	27	-	33	-	31	-	-	22	-	-	-	-	-	-	-	-	-	8	-	-	-	8	6	21	-	13	-	-	-	12	-	-	-	-	-	-	9	17.2727
5	100	-	-	13	-	-	-	-	5	-	-	-	-	-	-	-	-	-	14	-	-	-	-	30	-	-	30	-	-	-	-	-	-	-	-	-	-	-	18.4000
6	100	-	-	29	-	-	-	-	14	-	-	-	-	-	-	-	-	-	25	-	-	-	-	34	-	-	37	-	-	-	-	-	-	-	-	-	-	-	27.8000
7	100	-	-	23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	34	-	-	-	-	14	-	-	28	-	-	-	-	-	-	-	-	-	-	-	24.7500
8	100	-	-	24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	34	-	-	-	-	-	-	-	29	-	-	-	-	-	-	-	-	-	-	-	29.0000
9	100	-	-	29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	15	-	-	-	-	-	-	-	36	-	-	-	-	-	-	-	-	-	-	-	26.6667
10	100	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

		RFDS Time (s)																																				
Level	**Perfect Time**	**A1b**	**A2b**	**A3b**	**A4b**	**A5b**	**A6b**	**A7b**	**A8b**	**A9b**	**A10b**	**A11b**	**A12b**	**A13b**	**A14b**	**A15b**	**A16b**	**A17b**	**A18b**	**A19b**	**A20b**	**A21b**	**A22b**	**A23b**	**A24b**	**A25b**	**A26b**	**A27b**	**A28b**	**A29b**	**A30b**	**A31b**	**A32b**	**A33b**	**A34b**	**A35b**	**A36b**	**A37b**	**Average**
1	2.09	10.40	31.84	11.61	16.66	33.82	29.16	18.67	25.51	35.24	17.87	9.63	48.82	29.77	43.39	29.24	25.87	16.33	12.38	19.14	32.13	26.48	18.06	9.73	3.85	10.69	29.75	57.63	14.11	29.07	17.07	31.31	26.74	59.22	13.63	15.47	31.38	15.73	24.5243
2	2.50	43.13	38.21	26.86	38.54	52.61	22.83	33.84	20.04	25.98	19.67	31.46	35.32	48.57	53.59	45.02	49.73	37.95	15.37	24.27	36.25	15.50	22.49	20.07	38.85	46.69	30.58	24.20	16.32	41.71	29.78	57.43	21.61	23.88	54.87	21.55	33.26	17.91	32.8632
3	3.90	32.43	44.47	38.21	27.26	32.81	19.74	47.21	45.07	53.86	24.35	-	-	33.01	37.01	52.67	-	31.32	52.12	21.08	39.21	38.71	23.95	16.97	24.94	59.70	27.85	46.95	53.73	50.00	41.56	47.58	39.63	57.73	35.38	-	-	38.24	38.5859
4	6.25	34.33	-	31.36	-	32.56	-	-	36.87	-	-	-	-	-	-	-	-	-	43.55	-	-	-	43.45	44.25	37.06	-	43.00	-	-	-	41.26	-	-	-	-	-	-	42.85	39.1400
5	7.59	-	-	41.33	-	-	-	-	44.68	-	-	-	-	-	-	-	-	-	41.10	-	-	-	-	34.33	-	-	34.35	-	-	-	-	-	-	-	-	-	-	-	39.1580
6	15.69	-	-	107.76	-	-	-	-	81.57	-	-	-	-	-	-	-	-	-	70.00	-	-	-	-	62.44	-	-	59.55	-	-	-	-	-	-	-	-	-	-	-	76.2640
7	8.25	-	-	48.81	-	-	-	-	-	-	-	-	-	-	-	-	-	-	41.57	-	-	-	-	54.07	-	-	45.84	-	-	-	-	-	-	-	-	-	-	-	47.5725
8	6.25	-	-	36.21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	31.76	-	-	-	-	-	-	-	34.13	-	-	-	-	-	-	-	-	-	-	-	34.0333
9	7.25	-	-	32.71	-	-	-	-	-	-	-	-	-	-	-	-	-	-	39.97	-	-	-	-	-	-	-	29.56	-	-	-	-	-	-	-	-	-	-	-	34.0800
10	36.00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

		RFDS No. of Trials (n)																																				
Level	**Perfect Trials**	**A1b**	**A2b**	**A3b**	**A4b**	**A5b**	**A6b**	**A7b**	**A8b**	**A9b**	**A10b**	**A11b**	**A12b**	**A13b**	**A14b**	**A15b**	**A16b**	**A17b**	**A18b**	**A19b**	**A20b**	**A21b**	**A22b**	**A23b**	**A24b**	**A25b**	**A26b**	**A27b**	**A28b**	**A29b**	**A30b**	**A31b**	**A32b**	**A33b**	**A34b**	**A35b**	**A36b**	**A37b**	**Average**
1	1	1	1	7	3	1	1	2	1	1	2	2	2	1	1	1	3	1	1	1	2	2	1	1	3	3	2	1	3	1	3	3	1	1	1	3	2	1	1.8108
2	1	1	1	1	1	1	1	1	1	2	1	2	2	2	2	1	2	2	1	1	1	2	1	1	1	1	1	1	1	1	1	1	2	1	2	2	2	1	1.3243
3	1	1	3	1	1	3	1	1	1	1	7	7	7	1	3	2	7	7	1	3	2	6	1	1	3	3	2	1	1	3	2	2	1	3	7	7	7	1	3.0000
4	1	7	4	5	7	5	7	7	2	7	7	-	-	7	7	7	-	3	5	7	7	3	2	1	7	7	5	7	7	7	4	7	7	7	3	-	-	1	5.5000
5	1	4	-	5	-	7	-	-	6	-	-	-	-	-	-	-	-	-	3	-	-	-	7	4	4	-	1	-	-	-	7	-	-	-	-	-	-	7	5.0000
6	1	-	-	2	-	-	-	-	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	1.2000
7	1	-	-	3	-	-	-	-	4	-	-	-	-	-	-	-	-	-	3	-	-	-	-	6	-	-	6	-	-	-	-	-	-	-	-	-	-	-	4.4000
8	1	-	-	6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	-	-	-	7	-	-	4	-	-	-	-	-	-	-	-	-	-	-	5.2500
9	1	-	-	6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	3.3333
10	1	-	-	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	-	-	-	-	5.6667

Figure B.8: RFDS result table.

Path_v02 Flight Task Performance

		Completion Time (s)																																				
sub-SN	**Task**	**A1b**	**A2b**	**A3b**	**A4b**	**A5b**	**A6b**	**A7b**	**A8b**	**A9b**	**A10b**	**A11b**	**A12b**	**A13b**	**A14b**	**A15b**	**A16b**	**A17b**	**A18b**	**A19b**	**A20b**	**A21b**	**A22b**	**A23b**	**A24b**	**A25b**	**A26b**	**A27b**	**A28b**	**A29b**	**A30b**	**A31b**	**A32b**	**A33b**	**A34b**	**A35b**	**A36b**	**A37b**	**Average**
1	B1.2 (RCJ)	165.00	261.56	161.34	239.69	333.81	241.26	415.96	137.88	383.80	233.30	198.05	694.99	222.84	543.33	238.20	179.31	155.16	51.28	180.40	257.12	280.43	154.39	297.27	256.36	185.02	110.60	166.57	262.34	323.28	215.07	401.10	229.84	235.02	167.52	388.27	314.16	184.25	255.8316
2	B1.3 (KBD)	-	-	-	39.28	73.04	48.68	77.98	84.39	91.57	75.4	69.57	164.24	123.37	99.15	87.77	62.9	54.36	25.5	46.45	60.57	49.78	62.3	43.92	43.46	60.25	35.08	32.76	43.68	60.15	39.52	91.88	52.25	45.73	52.4	49.35	50.7	43.62	62.9721
3	B2.2 (mSVG)	94	193.09	97.39	129.97	123.72	122.27	127.3	222.14	189.08	196.47	180.69	283.79	216.74	303.5	172.19	184.99	262.34	179.76	149.24	209.09	164.09	157.24	122.67	162.59	199.89	109	184.27	156.34	295.25	184.87	222.84	126.59	192.89	111.49	290.16	172.79	135.91	179.0984

		Path Navigation Accuracy (%)																																				
sub-SN	**Task**	**A1b**	**A2b**	**A3b**	**A4b**	**A5b**	**A6b**	**A7b**	**A8b**	**A9b**	**A10b**	**A11b**	**A12b**	**A13b**	**A14b**	**A15b**	**A16b**	**A17b**	**A18b**	**A19b**	**A20b**	**A21b**	**A22b**	**A23b**	**A24b**	**A25b**	**A26b**	**A27b**	**A28b**	**A29b**	**A30b**	**A31b**	**A32b**	**A33b**	**A34b**	**A35b**	**A36b**	**A37b**	**Average**
1	B1.2 (RCJ)	95	90	70	70	70	80	65	80	65	65	65	70	40	55	60	75	80	90	60	85	60	80	75	85	80	80	85	50	75	80	50	50	85	72.5	50	80	75	71.4189
2	B1.3 (KBD)	-	-	-	95	95	90	100	95	90	90	95	95	100	100	100	95	100	95	95	95	95	100	95	100	100	100	95	95	100	100	100	90	100	100	100	90	95	96.4706
3	B2.2 (mSVG)	95	95	95	95	95	100	100	95	100	100	100	95	100	95	95	90	90	90	100	90	100	95	100	100	95	95	90	100	100	95	90	95	100	100	90	95	100	96.0811

NASA Task Load Index (TLX) Workload Survey

		B1.2 (RCJ) - TLX Rating (0-20)																																				
SN	**Workload Index**	**A1b**	**A2b**	**A3b**	**A4b**	**A5b**	**A6b**	**A7b**	**A8b**	**A9b**	**A10b**	**A11b**	**A12b**	**A13b**	**A14b**	**A15b**	**A16b**	**A17b**	**A18b**	**A19b**	**A20b**	**A21b**	**A22b**	**A23b**	**A24b**	**A25b**	**A26b**	**A27b**	**A28b**	**A29b**	**A30b**	**A31b**	**A32b**	**A33b**	**A34b**	**A35b**	**A36b**	**A37b**	**Average**
1	Mental Demand (md)	7.5	9.0	13.0	14.0	14.0	14.5	20.0	12.5	17.0	13.5	10.0	13.0	10.0	14.0	15.0	16.5	12.0	7.5	13.5	14.0	13.0	14.0	14.0	10.0	10.0	10.0	10.0	12.0	17.5	7.5	14.5	11.5	10.0	15.0	11.5	14.0	11.0	12.5946
2	Physical Demand (pd)	0.5	2.0	2.0	6.0	6.0	2.5	10.0	13.5	10.0	4.5	4.0	10.0	8.0	0.0	3.0	3.5	5.0	4.5	1.5	3.0	15.0	4.0	2.0	13.0	15.0	4.0	0.0	5.0	2.5	0.5	4.5	1.5	10.0	3.0	11.5	5.0	9.0	5.5405
3	Temporal Demand (td)	3.5	7.0	12.0	4.0	1.0	16.5	17.0	4.5	5.0	9.5	4.0	6.0	5.0	10.0	2.0	7.5	6.0	12.5	9.5	6.0	18.0	12.0	9.0	10.0	12.0	10.0	4.0	7.0	10.0	9.5	7.5	12.5	20.0	12.0	10.5	6.0	5.0	8.7432
4	Performance (p)	6.5	8.0	9.0	10.0	9.0	12.5	14.0	11.5	4.0	6.5	9.0	16.0	17.0	16.0	10.0	16.5	5.0	2.5	4.5	7.0	18.0	4.0	10.0	10.0	10.0	4.0	10.0	10.0	4.5	2.5	14.5	8.5	10.0	14.0	7.5	5.0	5.0	9.2432
5	Effort (e)	8.5	13.0	15.0	14.0	13.0	14.5	19.0	9.5	16.0	13.5	11.0	14.0	10.0	14.0	13.0	16.5	11.0	10.5	13.5	15.0	18.0	8.0	12.0	16.0	16.0	4.0	14.0	11.0	17.5	8.5	16.5	11.5	20.0	15.0	16.5	10.0	11.0	13.2432
6	Frustration (f)	4.5	7.0	15.0	7.0	1.0	16.5	14.0	4.5	5.0	9.5	1.0	12.0	3.0	0.0	5.0	3.5	0.0	5.5	4.5	7.0	18.0	2.0	14.0	13.0	0.5	9.0	0.0	12.0	16.5	3.5	16.5	12.5	10.0	15.0	7.5	7.0	8.0	7.8514

		B1.3 (KBD) - TLX Rating (0-20)																																				
SN	**Workload Index**	**A1b**	**A2b**	**A3b**	**A4b**	**A5b**	**A6b**	**A7b**	**A8b**	**A9b**	**A10b**	**A11b**	**A12b**	**A13b**	**A14b**	**A15b**	**A16b**	**A17b**	**A18b**	**A19b**	**A20b**	**A21b**	**A22b**	**A23b**	**A24b**	**A25b**	**A26b**	**A27b**	**A28b**	**A29b**	**A30b**	**A31b**	**A32b**	**A33b**	**A34b**	**A35b**	**A36b**	**A37b**	**Average**
1	Mental Demand (md)	-	-	-	6.0	5.5	7.5	13.0	4.5	14.0	5.5	3.0	9.0	6.5	10.0	3.0	11.5	8.0	4.5	4.5	9.0	2.0	4.0	9.0	3.0	2.0	1.0	5.0	2.0	8.5	2.5	5.5	1.5	0.5	3.0	0.5	6.0	1.0	5.3529
2	Physical Demand (pd)	-	-	-	3.0	1.0	3.5	10.0	5.5	0.0	4.5	2.0	9.0	3.5	0.0	0.0	2.5	8.0	1.5	0.5	2.0	5.0	2.0	4.0	3.0	1.5	0.0	0.0	0.0	2.5	2.5	2.5	0.5	0.0	0.0	0.5	6.0	1.0	2.5735
3	Temporal Demand (td)	-	-	-	2.0	0.5	3.5	15.0	6.5	7.0	5.5	2.0	6.0	4.5	6.0	0.0	3.5	7.0	6.5	2.5	7.0	5.0	7.0	6.0	3.0	4.5	1.0	3.0	2.0	5.5	2.5	6.5	3.5	6.0	2.0	5.5	5.0	2.0	4.5588
4	Performance (p)	-	-	-	4.0	2.5	6.5	5.0	5.5	4.0	5.5	4.0	4.0	10.0	7.0	3.0	16.5	8.0	1.5	0.5	4.0	6.0	2.0	4.0	4.0	4.5	0.0	0.0	7.0	2.5	0.5	6.5	6.5	1.0	4.0	0.5	2.0	0.0	4.1912
5	Effort (e)	-	-	-	6.0	3.5	5.5	14.0	7.5	6.0	12.5	3.0	6.0	5.5	7.0	2.0	12.5	8.0	4.5	8.5	2.0	9.0	2.0	5.0	3.0	6.0	1.0	0.0	1.0	5.5	5.5	5.5	6.5	0.0	4.0	3.5	6.0	3.0	5.3088
6	Frustration (f)	-	-	-	2.0	0.5	2.5	7.0	4.5	8.0	8.5	0.0	4.0	2.5	0.0	0.0	2.5	0.0	2.5	1.5	2.0	5.0	0.0	7.0	1.0	2.5	1.0	0.0	2.0	6.5	0.5	1.5	7.5	1.0	2.0	0.5	4.0	4.0	2.7647

		B2.2 (mSVG) - TLX Rating (0-20)																																				
SN	**Workload Index**	**A1b**	**A2b**	**A3b**	**A4b**	**A5b**	**A6b**	**A7b**	**A8b**	**A9b**	**A10b**	**A11b**	**A12b**	**A13b**	**A14b**	**A15b**	**A16b**	**A17b**	**A18b**	**A19b**	**A20b**	**A21b**	**A22b**	**A23b**	**A24b**	**A25b**	**A26b**	**A27b**	**A28b**	**A29b**	**A30b**	**A31b**	**A32b**	**A33b**	**A34b**	**A35b**	**A36b**	**A37b**	**Average**
1	Mental Demand (md)	6.5	13.0	15.0	2.0	3.0	11.5	5.0	2.5	14.0	12.5	0.0	5.0	4.5	13.0	0.0	13.5	12.0	7.5	10.5	8.0	1.0	12.0	7.0	6.0	10.0	3.0	14.0	6.0	11.5	13.5	13.5	3.5	7.0	9.0	7.5	10.0	5.0	8.0676
2	Physical Demand (pd)	2.5	8.0	1.0	0.0	0.5	7.5	5.0	1.5	6.0	7.5	9.0	5.0	5.5	4.0	6.0	1.5	14.0	4.5	4.5	9.0	0.0	11.0	8.0	2.0	1.5	2.0	4.0	3.0	5.5	10.5	5.5	0.5	7.0	12.0	8.5	8.0	5.0	5.3108
3	Temporal Demand (td)	3.5	9.0	8.0	0.0	4.0	3.5	4.0	3.5	5.0	9.5	5.0	5.0	4.5	0.0	3.0	7.5	6.0	5.5	7.5	3.0	1.0	6.0	8.0	1.0	3.0	1.0	2.0	4.0	7.5	9.5	7.5	2.5	17.0	6.0	4.5	7.0	2.0	5.0405
4	Performance (p)	6.5	10.0	4.0	0.0	7.5	2.5	3.0	8.5	2.0	6.5	8.0	3.0	13.5	9.0	2.0	15.5	8.0	5.0	3.5	6.0	0.0	2.0	4.0	3.0	6.0	3.0	15.0	5.0	3.5	3.5	7.5	3.5	3.0	2.0	5.5	3.0	0.0	5.2297
5	Effort (e)	7.5	6.0	12.0	0.0	9.5	14.5	3.0	2.5	10.0	13.5	9.0	10.0	6.5	9.0	2.0	11.5	10.0	9.5	11.5	8.0	1.0	6.0	8.0	5.0	12.0	1.0	10.0	5.0	11.5	11.5	8.5	3.5	10.0	12.0	3.5	6.0	5.0	7.7027
6	Frustration (f)	2.5	4.0	5.0	0.0	8.5	1.5	3.0	3.5	5.0	9.5	4.0	2.0	5.5	0.0	0.0	1.5	4.0	6.5	3.5	6.0	0.0	3.0	7.0	1.0	4.0	4.0	0.0	3.0	12.5	12.5	4.5	0.5	9.0	3.0	1.5	4.0	8.0	4.1351

Figure B.9: NASA TLX survey result table.

mSVG Speech to Gesture Ratio

		Speech Stages																																					
SN	Navigation Stage	A1b	A2b	A3b	A4b	A5b	A6b	A7b	A8b	A9b	A10b	A11b	A12b	A13b	A14b	A15b	A16b	A17b	A18b	A19b	A20b	A21b	A22b	A23b	A24b	A25b	A26b	A27b	A28b	A29b	A30b	A31b	A32b	A33b	A34b	A35b	A36b	A37b	Average
1	Start - hovering	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
2	Start-A (1/4)	0	1	0	0	1	0	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0.3784
3	Start-A (2/4)	0	1	0	1	0	0	1	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0.3514
4	Start-A (3/4)	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0.1081
5	Start-A (4/4)	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0.0811
6	A - Orient/step	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.0811
7	A-B (1/4)	0	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0.2432
8	A-B (2/4)	0	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0.2432
9	A-B (3/4)	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0.1622
10	A-B (4/4)	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0.1351
11	B - Orient/step/none	0	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	1	0	0	1	1	0	1	0	0	0.3243
12	B-C (1/6)	0	0	1	1	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0.2432
13	B-C (2/6)	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0.1892
14	B-C (3/6) Window ascent/descent	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
15	B-C (4/6) Window go through	0	0	1	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0.2162
16	B-C (5/6)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0.1081
17	B-C (6/6)	0	0	0	1	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0.2703
18	C - Orient	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
19	C-Goal (1/6)	0	1	1	1	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0.3784
20	C-Goal (2/6)	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1622
21	C-Goal (3/6) Window ascent/descent	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
22	C-Goal (4/6) Window go through	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0.1622
23	C-Goal (5/6)	0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0.1892
24	C-Goal (6/6) - landing	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0000
		5	14	14	18	11	8	9	8	12	8	6	14	10	11	9	5	5	9	10	7	5	5	18	10	7	5	5	9	12	11	5	6	8	10	13	7	5	

		Gesture Stages																																					
SN	Navigation Stage	A1b	A2b	A3b	A4b	A5b	A6b	A7b	A8b	A9b	A10b	A11b	A12b	A13b	A14b	A15b	A16b	A17b	A18b	A19b	A20b	A21b	A22b	A23b	A24b	A25b	A26b	A27b	A28b	A29b	A30b	A31b	A32b	A33b	A34b	A35b	A36b	A37b	Average
1	Start - hovering	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
2	Start-A (1/4)	1	0	1	1	0	1	0	0	1	0	1	1	0	0	1	1	1	0	1	1	1	1	0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	0.6216
3	Start-A (2/4)	1	0	1	0	1	1	0	0	1	1	1	1	0	0	1	1	1	0	1	1	1	1	0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	0.6486
4	Start-A (3/4)	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0.8919
5	Start-A (4/4)	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0.9189
6	A - Orient/step	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0.9189
7	A-B (1/4)	1	0	0	1	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	0	1	1	0.7568
8	A-B (2/4)	1	0	0	1	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	0	1	1	0.7568
9	A-B (3/4)	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	0.8378
10	A-B (4/4)	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	0.8649
11	B - Orient/step/none	1	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	1	0	1	1	0	0	1	1	1	1	0	1	1	0	0	1	0	1	1	0.6757
12	B-C (1/6)	1	1	0	0	0	0	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0.7568
13	B-C (2/6)	1	1	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0.8108
14	B-C (3/6) Window ascent/descent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
15	B-C (4/6) Window go through	1	1	0	0	0	1	1	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0.7838
16	B-C (5/6)	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0.8919
17	B-C (6/6)	1	1	1	0	1	1	0	1	1	1	1	0	0	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	0.7297
18	C - Orient	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
19	C-Goal (1/6)	1	0	0	0	1	1	1	1	0	0	1	0	0	1	0	1	1	1	0	0	1	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1	1	0.6216
20	C-Goal (2/6)	1	0	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.8378
21	C-Goal (3/6) Window ascent/descent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
22	C-Goal (4/6) Window go through	1	0	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0.8378
23	C-Goal (5/6)	1	0	1	0	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0.8108
24	C-Goal (6/6) - landing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
		19	10	10	6	13	16	15	16	12	16	18	10	14	13	15	19	19	15	14	17	19	19	6	14	17	19	19	15	12	13	19	18	16	14	11	17	19	

Figure B.10: Speech gesture ratio result table.

Appendix C

Program Codes

C.1 Teensy 3.2 Flight Controller Codes

The flight test video can be found in the accompanying thesis CD-R "*./multimedia/*" folder.

C.1.1 iQuad_Jade_v08.ino

Available on thesis CD-R "*./Appendix C - Program Codes/iQuad_Jade_v08/*" folder.

C.1.2 calibration.h

Available on thesis CD-R "*./Appendix C - Program Codes/iQuad_Jade_v08/*" folder.

C.1.3 iQuad Jade speech and keyboard control modified code

Available on thesis CD-R "*./Appendix C - Program Codes/speech_to_quad/*" folder.

C.2 MATLAB MCPU Operation Simulation Codes

Available on thesis CD-R "*./Appendix C - Program Codes/matlab/*" folder.

C.3 Python Gazebo simulation - graphical MCPU Operation Simulation Codes

Available on thesis CD-R "*./Appendix C - Program Codes/msvg_nav/*" folder.

C.4 Speech capture and processing into control symbol program code

Available on thesis CD-R "*./Appendix C - Program Codes/msvg_speech_cmds/*" folder.

C.5 Gesture capture and processing into control symbol program code

Available on thesis CD-R "*./Appendix C - Program Codes/msvg_finger_gestures/*" folder.

C.5.1 Gesture capture and processing - Haar cascade fist gesture program code

Available on thesis CD-R "*./Appendix C - Program Codes/haar_cascade_fist/*" folder.

C.5.2 Gesture capture and processing - uEye camera frame sequence program code

Available on thesis CD-R "*./Appendix C - Program Codes/frame_sequence/*" folder.

C.6 keyboard control symbol processing program code

Available on thesis CD-R "*./Appendix C - Program Codes/msvg_v3_odroid_sbc/*" folder.

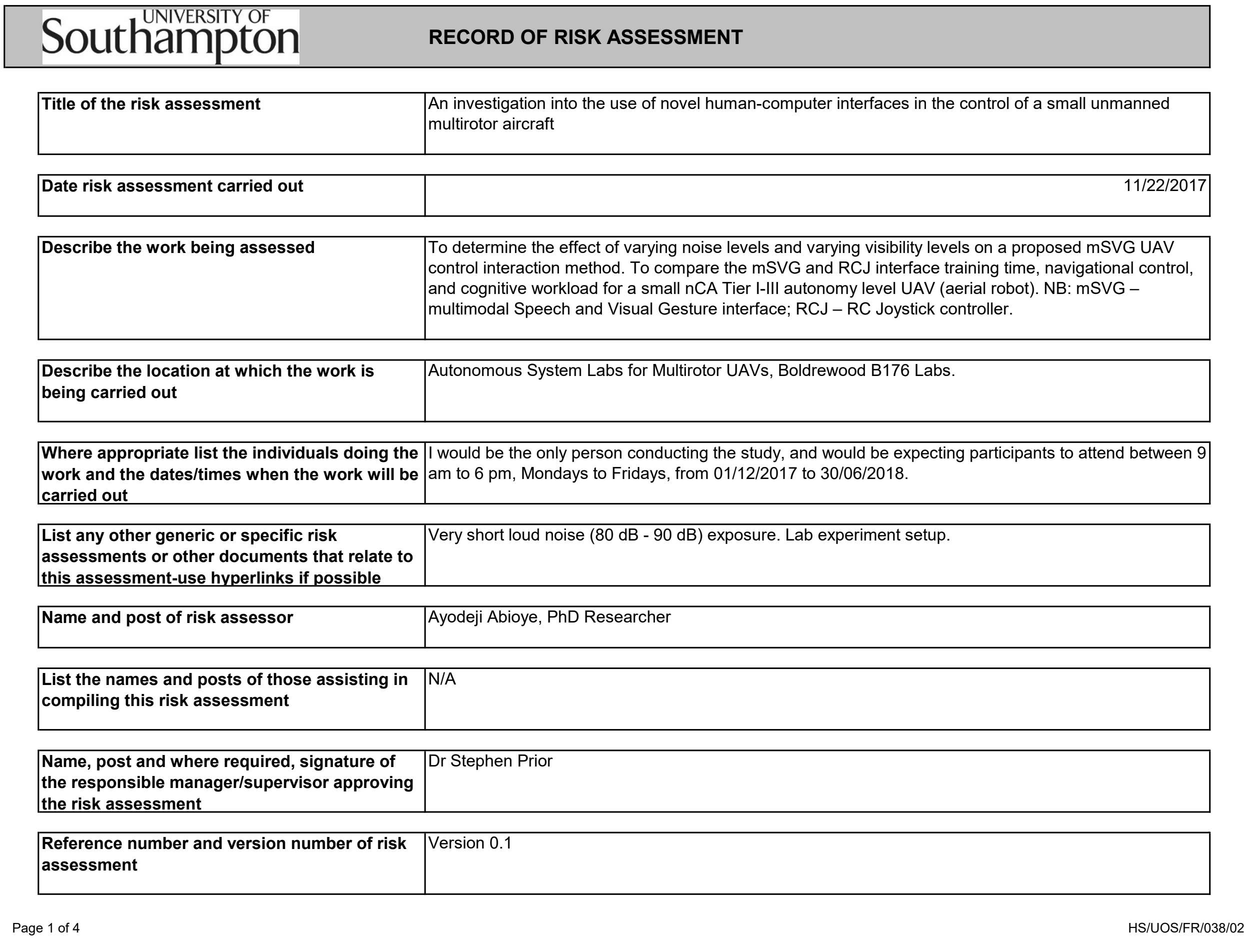
C.7 ROS Gazebo input control symbol processing and execution program code

Available on thesis CD-R "*./Appendix C - Program Codes/msvg_v4_linux_desktop/*" folder.

Appendix D

Ethics Application

D.1 Risk Assessment



UNIVERSITY OF Southampton

RECORD OF RISK ASSESSMENT

Title of the risk assessment	An investigation into the use of novel human-computer interfaces in the control of a small unmanned multirotor aircraft
Date risk assessment carried out	11/22/2017
Describe the work being assessed	To determine the effect of varying noise levels and varying visibility levels on a proposed mSVG UAV control interaction method. To compare the mSVG and RCJ interface training time, navigational control, and cognitive workload for a small nCA Tier I-III autonomy level UAV (aerial robot). NB: mSVG – multimodal Speech and Visual Gesture interface; RCJ – RC Joystick controller.
Describe the location at which the work is being carried out	Autonomous System Labs for Multirotor UAVs, Boldrewood B176 Labs.
Where appropriate list the individuals doing the work and the dates/times when the work will be carried out	I would be the only person conducting the study, and would be expecting participants to attend between 9 am to 6 pm, Mondays to Fridays, from 01/12/2017 to 30/06/2018.
List any other generic or specific risk assessments or other documents that relate to this assessment-use hyperlinks if possible	Very short loud noise (80 dB - 90 dB) exposure. Lab experiment setup.
Name and post of risk assessor	Ayodeji Abioye, PhD Researcher
List the names and posts of those assisting in compiling this risk assessment	N/A
Name, post and where required, signature of the responsible manager/supervisor approving the risk assessment	Dr Stephen Prior
Reference number and version number of risk assessment	Version 0.1

Page 1 of 4 HS/UOS/FR/038/02

Assessment

Title of risk assessment	An investigation into the use of novel human-computer interfaces in the control of a small unmanned multirotor aircraft

Risk Acceptability	
1-3	Risk acceptable
4-6	Risk to be reduced if readily possible
7-14	Risk to be reduced if reasonably practicable
15-25	Risk unacceptable

Risk Matrix			Severity				
			very low	low	medium	high	very high
			1	2	3	4	5
Likelihood	certainty	5	5	10	15	20	25
	likely	4	4	8	12	16	20
	possible	3	3	6	9	12	15
	less likely	2	2	4	6	8	10
	improbable	1	1	2	3	4	5

						Risk Factors			
ref	Task/Aspect of work	Hazard	Harm and how it could arise	Who could be affected?	Existing measures to control risk	Overall Likelihood	Overall Severity	Residual Risk score	Any changes or extra controls?
	Short loud noise levels (80 dB - 90 dB)	hearing damaged	hearing damage over very long exposure	Both the experiment participant and experiment conductor	Such loud noises are limited to no more than 30 seconds, with a recovery time of at least 2 min. Total exposure under 5 min for the whole 3 hours experiment.	1	3	3	No
	LED lighting systems	Weariness	Weariness my arise if participant is sensitive to LED lighting systems	Subject	Default to the use of normal lighting	1	1	1	No
	Multi-screen computer setup	Eye damage	Long exposure to simulator display screen with varying screen brightness and texture	Subject	Sit a good distance from screen, adjust screen level and angles to suitable levels for participant, and recommend screen breaks every 40 min.	1	1	1	No
	Use of electrical wires	Possibility of electrocution	Electrocution could occur if participant comes in contact with exposed electrical wires or faulty connectors.	Both Subject and Conductor	Ensure no exposed wiring, and tidy Lab workspace, and walk path wiring.	1	1	1	No

	Wires attached to equipment	Tripping over wires	Equipment wires in walk paths could cause a fall	Both Subject and Conductor	Remove all wires from walk path or neatly tape to ground and make participants aware of such potential trip wires	1	1	1	No
	Other Lab components	Tripping over other Lab equipments	Participant may trip over other lab equipments if they choose to walk around beyond the experiment designated area	Subject	Advise participants to remain within experiment designated area. Also, clearly mark off "NO GO" areas.	1	1	1	No

Post Risk Assessment Actions

Title of risk assessment An investigation into the use of novel human-computer interfaces in the control of a small unmanned multirotor aircraft

Have any of the specialist control measures listed below been identified as required during this risk assessment? - indicate yes or no - if yes then include details on the post assessment action list below.	**yes/no**
is any exposure monitoring required?	no
Is any occupational health monitoring required?	no
Are there any hazards or other factors that could affect pregnant or nursing mothers?	no

Is any specfic training required before people can carry out this work?	no
Site local safety induction	

Are any additional procedures or risk assessments required as a result of this asssessment?	no

	Post Assessment actions		
ref	**action**	**by whom**	**by when**

D.2 Approved ERGO Ethics Application

UNIVERSITY OF Southampton

ERGO application form – Ethics form

All mandatory fields are marked (M*). Applications without mandatory fields completed are likely to be rejected by reviewers. Other fields are marked "if applicable". Help text is provided, where appropriate, in italics after each question.

1. APPLICANT DETAILS

1.1 (M*) Applicant name:	Ayodeji Opeyemi ABIOYE
1.2 Supervisor (if applicable):	Dr Stephen D. PRIOR Dr Glyn T. THOMAS Dr Sarvapali D. RAMCHURN Dr Peter SADDINGTON
1.3 Other researchers/collaborators (if applicable): *Name, address, email, telephone*	-

2. STUDY DETAILS

Abbreviations

mSVG	*Multimodal Speech and Visual Gesture*
NASA-TLX	*NASA Task Load Index*
nCA	*Navigation Control Autonomy*
RC	*Radio Controlled*
RCJ	*RC Joystick*
ROS	*Robotic Operating System*
UAV	*Unmanned Aerial Vehicle*

2.1 (M*) Title of study:	An investigation into the use of novel human-computer interfaces in the control of a small unmanned multirotor aircraft
2.2 (M*) Type of study (*e.g. Undergraduate, Doctorate, Masters, Staff*):	Doctorate
2.3 i) (M*) Proposed start date:	18/12/2017
2.3 ii) (M*) Proposed end date:	31/07/2018

2.4 (M*) What are the aims and objectives of this study?
To determine the effect of varying noise levels and varying visibility levels on a proposed mSVG UAV control interaction method. To compare the mSVG and RCJ interface training time, navigational control, and cognitive workload for a small nCA Tier I-III autonomy level UAV (aerial robot).

UNIVERSITY OF Southampton

2.5 (M*) Background to study (*a **brief** rationale for conducting the study*):
The most popular HCI control interface for small multirotor UAVs, is the RC joystick controller (RCJ), which may be difficult to learn, require long training hours, and may demand too much of the operators cognitive effort to operate in task. After investigating many existing HCI alternatives, a multimodal speech and visual gesture (mSVG) control interface was proposed as a more intuitive and natural high-level control alternative for small multi-rotor UAVs. According to Green et al. (2007), "It is clear that people use speech, gesture, gaze and non-verbal cues to communicate in the clearest possible fashion." Therefore, this research attempts to adopt the human-human interaction method for human-aerobotic interaction.

2.6 (M*) Key research question (*Specify hypothesis if applicable*):
Q1: Identify the visibility and noise level range of mSVG effectiveness and the peak mSVG performance conditions. H1: The mSVG training time should be significantly shorter than the RCJ training time (estimated to be less than one-tenth). H2: mSVG and RCJ may have a similar performance level for an nCA Tier I-III autonomy level task. However at higher autonomy levels mSVG may outperform RCJ, whereas at lower autonomy levels RCJ may outperform mSVG. That is nCA Tier I-III autonomy level may be the equilibrium point of these two interfaces.

2.7 (M*) Study design (*Give a **brief** outline of basic study design*) *Outline what approach is being used, why certain methods have been chosen.*
This study would be conducted on a computer-based UAV simulator, augmented with external hardware-in-the-loop components (single-board computers, cameras, microphones, speakers, and lighting systems), in order to interact with the physical world, and to provide the natural alternative method of mSVG interaction with the UAV operator participant. The study participants would be mostly sited in front of a three-screen UAV simulation computer workstation for no more than three hours (total experiment duration), during which the participant would be asked to perform two major experiments – Task A and Task B, each of which are about 80 minutes long. Task A measures the effect of varying noise levels and varying visibility levels on mSVG interaction method by varying ambient noise level across five intervals between 50 dB and 90 dB while fixing lighting levels through five intervals between 5500 Lux to 50 Lux. Task B compares the mSVG and RCJ interface training time, navigational control, and cognitive workload for a small nCA Tier I-III autonomy level UAV. In order to do this, Task B is divided into two sub categories B1 and B2, with B1 focused on collecting RCJ comparison data, and B2 on capturing mSVG data to be compared with RCJ. This is to be performed in the order B1B2 by one-half of the participants and B2B1 by the other half of the participants in order to counter-balance potential first exposure bias or fatigue effect when performing the second part of Task B. Experiment is designed to last no more than 3 hours with 5 min breaks every 40 min. Participants would not be exposed to any dangerous sound level (sounds above 90 dB), and loud noises (80 dB – 90 dB) may only be used in short burst of 30 seconds or less, just enough to corrupt speech capture input, and exposure would not exceed a total of 5 min in the 3 hours experiment. Should a participant find the noise level inconvenient, noise cancelling headphones would be provided. Subject must have good hand dexterity and be between the ages of 18 to 69 years inclusive. Full consent must be given by each participant, and each participant has the right to withdraw at any time. Also, the experiment would be video and audio recorded, capturing visual gesture inputs, speech inputs, RC joystick inputs, instruments (sound meter and light meter) readings, and the corresponding tasks output as displayed on the workstation. These recordings would be used to document and confirm

UNIVERSITY OF Southampton

observations, and also aid in the analysis and discussions of the results such as the sources of errors which could be human, environmental, or system.

3. SAMPLE AND SETTING

3.1 (M*) How are participants to be *approached*? *Give details of what you will do if recruitment is insufficient. If participants will be accessed through a third party (e.g. children accessed via a school) state if you have permission to contact them and* ***upload any letters of agreement to your submission in ERGO****.*

Subjects must be volunteers and must willingly want to be part of the project. The participants would be recruited via word of mouth and via an announcement at a session of the MSc Unmanned Aircraft System Design "SESA6078 – UASD Group Design Project" course, with the permission of the module lead, Dr Stephen Prior (my supervisor) and in a second year undergraduate module session "FEEG2001 – System Design and Computing", where I am a demonstrator. Ideally, half the participants should have UAV RCJ flying experience, and the other half, no prior UAV RCJ flying experience. But a ratio of 1:5 to 5:1 would also be acceptable. If the number of participants is insufficient, or an acceptable ratio of experienced to non-experienced UAV RCJ flying participants is not reached, a pilot study may then be conducted instead.

3.2 (M*) Who are the proposed sample and where are they from (e.g. fellow students, club members)? *List inclusion/exclusion criteria if applicable. NB The University does not condone the use of 'blanket emails' for contacting potential participants (i.e. fellow staff and/or students).*

It is usually advised to ensure groups of students/staff have given prior permission to be contacted in this way, or to use of a third party to pass on these requests. This is because there is a potential to take advantage of the access to 'group emails' and the relationship with colleagues and subordinates; we therefore generally do not support this method of approach.

If this is the only way to access a chosen cohort, a reasonable compromise is to obtain explicit approval from the Faculty Ethics Committee (FEC) and also from a senior member of the Faculty in case of complaint.

Participants may or may not be student and staff members of the University, but must be between the ages of 18 to 69 years inclusive. Participants may or may not have prior UAV RCJ flying experience, but a good distribution of both may be required.
Exclusion criteria:
1) Inability to fly a UAV via a handheld RC joystick controller, probably due to hand dexterity problems, fidgety hands, hand disabilities, etc.
2) Sensitivity to short exposure (< 30 s) to loud noise levels (80 dB – 90 dB)
3) Sensitivity to LED based lighting systems

3.3 (M*) Describe the relationship between researcher and sample (*Describe any relationship e.g. teacher, friend, boss, clinician, etc.*)

Participants must be volunteers, however participants may be friends, colleagues, or students of the researcher. UAV researchers/developers can also participant in the study as their prior knowledge, experience, or bias would not affect the results

3.4 (M*) Describe how you will ensure that fully informed consent is being given: (*include how long participants have to decide whether to take part*)

UNIVERSITY OF Southampton

Signed written consent will be obtained from any participant who wishes to part take in the study. Upon showing interest, a participant information sheet would be sent to the participant and the participant would have at least 48 hours to decide whether they would like to part take or not, and have the right to withdraw from the project at any time without the need for any justification, even during the experiments.

4. RESEARCH PROCEDURES, INTERVENTIONS AND MEASUREMENTS

4.1 (M*) Give a brief account of the procedure as experienced by the participant
(Make clear who does what, how many times and in what order. Make clear the role of all assistants and collaborators. Make clear total demands made on participants, including time and travel). ***Upload any copies of questionnaires and interview schedules to your submission in ERGO.***

First the participants will be provided with a printed copy of the participant information sheet. If the participants is satisfied and wishes to continue with the tests, then a consent form would be provided the participant must give written consent before the test can begin.

The participant starts with a control experiment (A.1) as setup by the experiment conductor, to test that the participant can interact with the UAV simulator via speech, gestures and a combination of both under optimum conditions (quiet environment and good lighting). Sitting in front of the UAV simulator:
A.1) The participant requests nearest shelter/crevice location using speech only, then using visual gesture only, and finally using a combination of both speech and visual gesture. The participant writes down the UAV response in each case in a participant logbook, which is then checked by the conductor. This should last no longer than 20 min
A.2) Then the first test experiment to observe the effect of ambient noise levels on speech interaction is conducted. The participant requests nearest shelter/crevice location using speech only at 50 dB ambient noise level, This is then repeated for 60 dB, 70 dB, 80 dB, and 90 dB ambient noise levels. In each case, the participant writes down the UAV response in the participant logbook, which is then checked by the conductor. This should last no longer than 15 min.
A.3) Then the second test experiment to observe the effect of ambient lighting levels on visual gesture interaction is conducted. The participant requests nearest shelter/crevice location using visual gestures only at 5500 Lux ambient visibility level, This is then repeated for 3500 Lux, 1500 Lux, 500 Lux, and 50 Lux ambient lighting levels. In each case, the participant writes down the UAV response in the participant logbook, which is then checked by the conductor. This should also last no longer than 15 min.
A.4) Then the third test experiment to observe the effect of simultaneously varying ambient noise levels and ambient lighting levels on multimodal speech and visual gesture interaction is conducted. Again, the participant requests the nearest shelter/crevice location using any or both of speech and visual gestures at 5500 Lux ambient lighting level and at 50 dB ambient noise level. This is then repeated for 60 dB, 70 dB, 80 dB, and 90 dB ambient noise levels at 5500 Lux ambient lighting level. The lighting level is then changed to 3500 Lux, and then to 1500 Lux, and then to 500 Lux, and finally to 50 Lux, in each case varying the ambient noise levels from 50dB to 90 dB as previously done. For each iteration, the participant writes down the UAV response in the participant logbook, which is then checked by the conductor. This should last no longer than 30 min.

The participant then continues onto the second part of the study which is aimed at comparing the mSVG and RCJ interfaces in terms of training time, nCA Tier I-III navigation control, and cognitive workload metrics.
B.1.1) Here, the participant confirms their previous flying experience and then perform the RealFlight Drone Simulator level challenges using an RCJ, with the aim of measuring participant's skill levels, if they have previous flying experience, or quick training the participant if not. The participant starts from Level 1, noting down their score, time, and number of trials performed in order to pass this level on the participant logbook. This is repeated for Level 2, Level 3, Level 4, etc. until either the participants gets stuck at a Level or finishes Level 10. In each case, recording the score, time, and number of trials, which is then checked by the conductor. This should last no longer than 20 min.
B.1.2) The participant then performs a ROS Gazebo based UAV Simulator flight test using the RCJ control interface for navigation control. The flight test path begins at the start position, with the UAV navigating along straight lines, making right angle or acute angles turns, and going through hanging rings, to finish at the goal position. The flight test is to be repeated two more times to determine average/best performance, in each case recording the finishing time, the path accuracy score, the turning score, and the ring hoops score in the participant logbook, which is then checked by the conductor. The participant then completes the NASA-TLX survey to estimate cognitive workload for the RCJ-based navigation. This should last no longer than 20 min.

B.2.1) The participant is then introduced to the mSVG Tier I-III UAV navigation control syntax, which the participant then learns by using them to control a UAV in a ROS Gazebo Simulator open world environment. This should last no longer than 20 min.
B.1.2) The participant then performs the ROS Gazebo based UAV Simulator flight test, same as the one in B.1.2, using the mSVG control interface for navigation control. The flight test path begins at the start position, with the UAV navigating along straight lines, making right angle or acute angles turns, and going through hanging rings, to finish at the goal position. The flight test is to be repeated two more times to determine average/best performance, in each case recording the finishing time, the path accuracy score, the turning score, and the ring hoops score in the participant logbook, which is then checked by the conductor. The participant then completes the NASA-TLX survey to estimate cognitive workload for the mSVG-based navigation. This should last no longer than 20 min.

In total, the study should last no longer than 3 hours, i.e. ~2 hrs 40 mins experiment with 20 mins break distributed across the experiment.

5. STUDY MANAGEMENT

5.1 (M*) State any potential for psychological or physical discomfort and/or distress?

Some participants may potentially find the noise levels between 80 dB – 90 dB uncomfortable, even though such exposure does not exceed normal everyday experience. If so, the participants may be issued with noise cancelling headphones to dampen the noise or the noise level generator may be limited to 80 dB for such participants.

5.2 (M*) Explain how you intend to alleviate any psychological or physical discomfort and/or distress that may arise? (if applicable)

Participants sensitive to loud noise levels between 80 dB – 90 dB would be issued noise cancelling headphones to dampen the noise, also the noise level generator may be limited to 80 dB for such participants. In addition to these, testing can be stopped for such participants at any time upon alerting the researcher, and if preferred by the participant, experiment Task A, involving such noise levels, can be skipped completely.

5.3 Explain how you will care for any participants in 'special groups' *(i.e. those in a dependent relationship, vulnerable or lacking in mental capacity)* **(if applicable)?**

N/A

5.4 Please give details of any payments or incentives being used to recruit participants (if applicable)?

£20 Amazon vouchers, sourced from the research fund.

5.5 i) How will participant anonymity and/or data anonymity be maintained (if applicable)?
Two definitions of anonymity exist:
i) Unlinked anonymity - *Complete anonymity can only be promised if questionnaires or other requests for information are not targeted to, or received from, individuals using their name or address or any other identifiable characteristics. For example if questionnaires are sent out with no possible identifiers when returned, or if they are picked up by respondents in a public place, then anonymity can be claimed. Research methods using interviews cannot usually claim anonymity – unless using telephone interviews when participants dial in.*
ii) Linked anonymity - *Using this method, complete anonymity cannot be promised because participants can be identified; their data may be coded so that participants are not identified by researchers, but the information provided to participants should indicate that they could be linked to their data.*

Linked anonymity. Participants data would be coded by assigning a unique identification number, archived as a zip file, which is then password protected, and then stored on a University password protected computer, backed up on a BitLocker encrypted external hard drive. These would be the only place whereby the participant's personal data and identification number can be linked and identified. Only the researcher can access the password protected file on the password protected computer and BitLocker encrypted hard drive, and therefore only the researcher can identify participants by their personal data and identification numbers.

5.5 ii) How will participant confidentiality be maintained (if applicable)?
Confidentiality is defined as the non-disclosure of research information except to another authorised person. Confidential information can be shared with those who are already party to it, and may also be disclosed where the person providing the information provides explicit consent.

Participant's confidentiality will be maintained through the use of participant identification number and all confidential information will be stored on a University password protected computer, backed up on a BitLocker encrypted hard drive. Only the researcher named above have access to the password protected computer and encrypted backup hard drive.

5.6 (M*) How will personal data and study results be stored securely during and after the study? *Researchers should be aware of, and compliant with, the*

UNIVERSITY OF Southampton

Data Protection policy of the University. You must be able to demonstrate this in respect of handling, storage and retention of data.
All data collected from every participants during and after the study will be stored and secured on a password protected University computer and backed-up on an external hard drive encrypted with BitLocker using a strong password mixing uppercase, lowercase, numbers, and symbols. This will be completed and kept in compliance with the Data Protection Policy of the University. In accordance with the University of Southampton Research Data management Policy, all significant research data should be held for a minimum of ten years.

5.7 (M*) Who will have access to these data?
The researcher and primary supervisors only.

N.B. – Before you upload this document to your ERGO submission remember to:

1. Complete ALL mandatory sections in this form

2. Upload any letters of agreement referred to in question 3.1 to your ERGO submission

3. Upload any interview schedules and copies of questionnaires referred to in question 4.1

D.3 Participant Information

UNIVERSITY OF Southampton

Participant Information Sheet

Study Title: An investigation into the use of novel human-computer interfaces in the control of a small unmanned multirotor aircraft

Researcher: Ayodeji Abioye
ERGO number: 30377

Please read this information carefully before deciding to take part in this research. It is up to you to decide whether or not to take part. If you are happy to participate you will be asked to sign a consent form.

<u>Abbreviations</u>

mSVG	*Multimodal Speech and Visual Gesture*
NASA-TLX	*NASA Task Load Index*
nCA	*Navigation Control Autonomy*
RC	*Radio Controlled*
RCJ	*RC Joystick*
ROS	*Robotic Operating System*
UAV	*Unmanned Aerial Vehicle*

What is the research about?
Could a human interact with an aerial robot via speech and gestures, as one would with another human? This research aims to investigate such alternative human-computer control interfaces that could be used to control small multirotor UAVs (aerial robots).This study is part of a PhD research, which is investigating novel control interfaces for small unmanned multirotor aircraft, conducted at the University of Southampton, and partly supported by the PTDF Fund (Nigeria).

This particular study is aimed at determining the effect of varying noise levels and varying visibility levels on a proposed mSVG UAV control interaction method. And to compare the mSVG and RCJ interface training time, navigational control, and cognitive workload for a small nCA Tier I-III autonomy level UAV (aerial robot). NB: mSVG – multimodal Speech and Visual Gesture interface; RCJ – RC Joystick controller.

Why have I been asked to participate?
You have been asked to participate in order to observe your experience using a proposed multimodal speech and visual gesture control method and to compare this with the popular RC joystick controller used to fly and control small multirotor UAVs (unmanned aerial vehicles).

What will happen to me if I take part?
This study would be conducted on a computer-based UAV simulator, augmented with external hardware-in-the-loop components (single-board computers, cameras, microphones, speakers, and lighting systems), in order to interact with the physical world, and to provide the natural alternative method of mSVG interaction with the UAV operator participant. You would be mostly sited in front of a three-screen UAV simulation computer workstation for no more than three hours (total experiment duration), during which the you would be asked to perform two major experiments – Task A and Task B, each of which are about 80 minutes long. Task A measures the effect of varying noise levels and varying visibility levels on mSVG interaction method by varying ambient noise level across five intervals between 50 dB and 90 dB while fixing lighting levels through five intervals between 5500 Lux to 50 Lux. Task B compares the mSVG and RCJ interface training time, navigational control, and cognitive workload for a small nCA Tier I-III autonomy level UAV. In order to do this, Task B is divided into two sub categories B1 and B2, with B1 focussed on collecting RCJ

UNIVERSITY OF Southampton

comparison data, and B2 on capturing mSVG data to be compared with RCJ. Experiment is designed to last no more than 3 hours with 5 min breaks every 40 min.

Please note that audio and video information are captured and recorded by the microphone and cameras, which are being used by the researcher to analyse experiment results.

Also, your total time commitment should be no longer than 3 hours, limited to just a single visit, NO follow up visit, and you should NOT be contacted again.

Are there any benefits in my taking part?
Yes, you would have an opportunity to try out an alternative HHI-like UAV control interface, and get an opportunity to test your RCJ flying skill level. Data obtained from your participation could inform the design of future UAV control interfaces. In addition to these, a £20 Amazon voucher is given at the end of the experiment to compensate for time and travel.

Are there any risks involved?
There is a risk of slight discomfort that may be caused by a short exposure to loud noise levels (80 dB – 90 dB). Such loud noises are limited to no more than 30 seconds, with a recovery time of at least 2 min. Total exposure is under 5 min for the whole 3 hours experiment.

Two LED lighting systems are used to vary ambient visibility level, for visual gesture capture, some persons may be sensitive to this and may get weary, if so, alert researcher and normal lighting would be restored.

The use of multi-screen computer may be uncomfortable for some persons. Some measures put in place to reduce any discomfort includes good sitting arrangement, good distance from screen, adjustable screen level and angles, and recommend screen breaks every 40 min.

Use of electrical wires could result in the possibility of electrocution. To avoid this, the researcher would make sure there are no exposed wires around experiment areas.

There is a risk of tripping over equipment wiring. To avoid this, the researcher would remove all unnecessary wiring, and tape wires neatly to the ground, walls, or appropriate supporting frames.

There is a risk of tripping over other lab equipment. You are therefore advised to remain within designated experiment area and avoid clearly marked “no go” areas of the lab.

Will my participation be confidential?
Yes, all data collected will be kept confidential in compliance with the Data Protection Act and University of Southampton data protection policy. To keep data confidential, all written data would be stored in a locked cabinet, and all electronic data will be stored on password protected computers and backed up on a BitLocker encrypted hard drive, that only the researcher have access to. Video and audio data will be stored on a password protected University computer, and backed up on a BitLocker encrypted external hard drive. Your data would be anonymised by assigning a unique ID number to your data. No one, other than the researcher and primary supervisors, will be able to link your name with the unique ID number. Data that would be published in journal articles, conferences, or meetings will not report any names or unique ID numbers to maintain the anonymity of the data. All data will be kept for 10 years.

What should I do if I want to take part?
You can complete the participant signup form on https://www.hai-research.com/participate or contact the researcher directly via email at aoa2g15@soton.ac.uk

What happens if I change my mind?
Participation in the study is entirely voluntary and you may withdraw at any point during the study without giving a reason for doing so and without affecting your rights. Upon withdrawal, you also reserve the right to request that data collected up to the point of withdrawal may or may not be used or should be destroyed immediately.

UNIVERSITY OF Southampton

What will happen to the results of the research?
The results of the research would be written up as part of the researchers PhD thesis, published in relevant academic journals, and presented in relevant conferences. The anonymised data may be made available for future research projects. All publications and anonymised data relating to the research would be made available through the university research repository. Research results and collected anonymised data would be stored for 10 years.

What happens if something goes wrong?
In the unlikely event that you wish to make a complaint or express a concern, you should contact the University's Research Integrity and Governance Management on 023 8059 5058 or via email at rgoinfo@soton.ac.uk

Where can I get more information?
If you require any further information or have any question regarding taking part in this study, please contact the researcher,

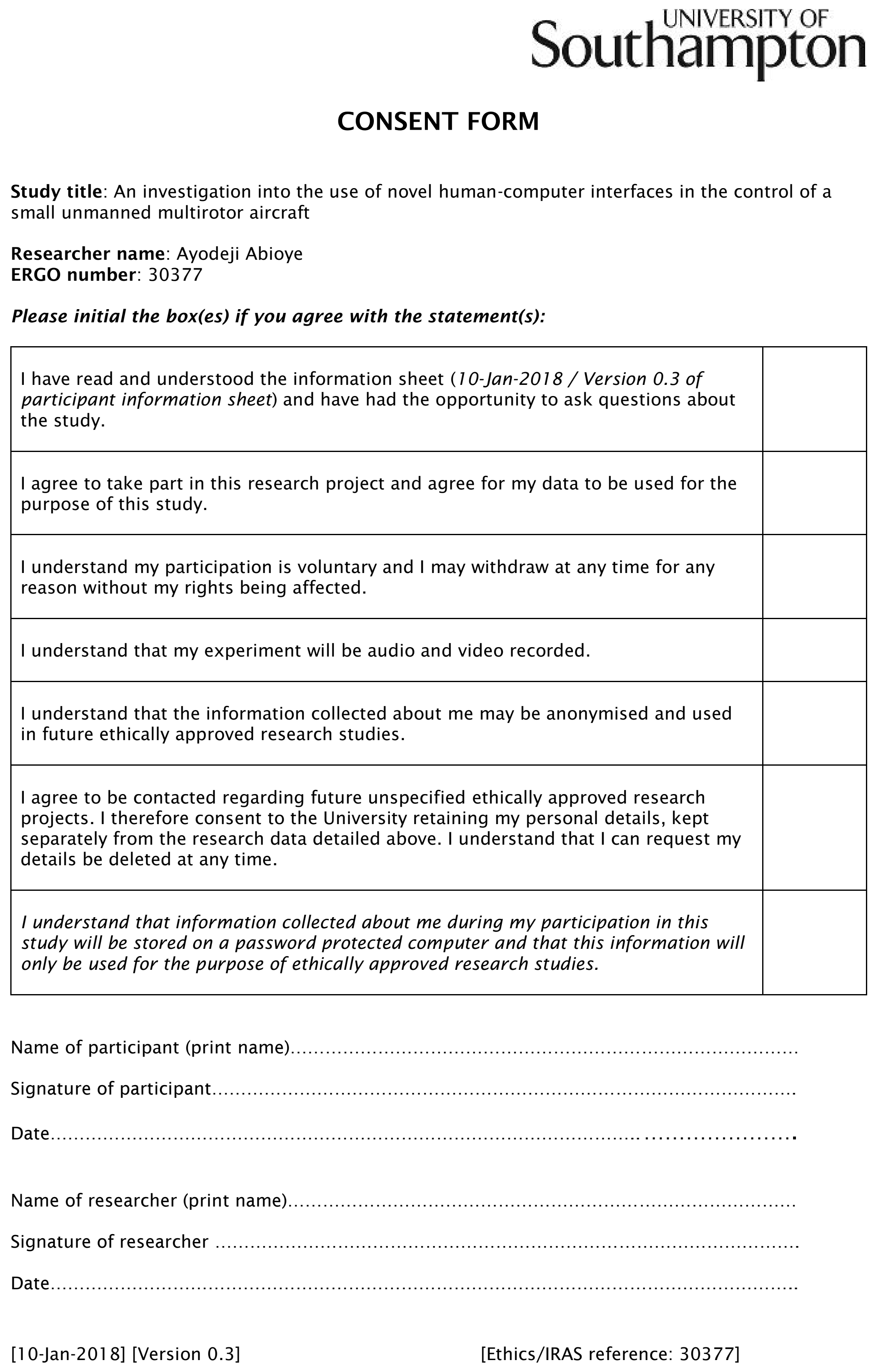
Ayodeji Abioye
PhD Student | Computational Engineering and Design Group
Aeronautics, Astronautics and Computational Engineering
Faculty of Engineering and the Environment
Building 176/5001 (104), Boldrewood Campus
University of Southampton | Southampton SO16 7QF
United Kingdom.

Email: aoa2g15@soton.ac.uk

Website: https://www.hai-research.com/

Thank you for considering participating in this study and for taking the time to read through this participant information sheet.

D.4 Participant Consent Form



UNIVERSITY OF Southampton

CONSENT FORM

Study title: An investigation into the use of novel human-computer interfaces in the control of a small unmanned multirotor aircraft

Researcher name: Ayodeji Abioye
ERGO number: 30377

Please initial the box(es) if you agree with the statement(s):

I have read and understood the information sheet (*10-Jan-2018 / Version 0.3 of participant information sheet*) and have had the opportunity to ask questions about the study.	
I agree to take part in this research project and agree for my data to be used for the purpose of this study.	
I understand my participation is voluntary and I may withdraw at any time for any reason without my rights being affected.	
I understand that my experiment will be audio and video recorded.	
I understand that the information collected about me may be anonymised and used in future ethically approved research studies.	
I agree to be contacted regarding future unspecified ethically approved research projects. I therefore consent to the University retaining my personal details, kept separately from the research data detailed above. I understand that I can request my details be deleted at any time.	
I understand that information collected about me during my participation in this study will be stored on a password protected computer and that this information will only be used for the purpose of ethically approved research studies.	

Name of participant (print name)……………………………………………………………………………

Signature of participant…………………………………………………………………………………………

Date……

Name of researcher (print name)……………………………………………………………………………

Signature of researcher …………………………………………………………………………………………

Date……

[10-Jan-2018] [Version 0.3] [Ethics/IRAS reference: 30377]

D.5 Participant Logbook

Participant Name:___________________________

Date: ______________________________________

Participant No.:_____________________________

7.8.1 hai-research experiment handbook

7.8.1.1 Preliminary setup (few hours before)

a. Print and post "Experiment in Progress" sign onto lab entrance glass door
b. Clean lab if needed – sweep, clear tables, remove food, clear bags
c. Setup camera – tested and ready to record
d. Start Linux desktop computer, queuing the RotorS ROS Gazebo Simulator
e. Start Speech and Gesture capture programs on Odroid XU4 SBC
f. Setup flight test on Laptop queuing Real Flight Drone Simulator

7.8.1.2 Experiment (to be completed within 3 hours)

a. Participant briefing:
    i. What to expect during the experiments – quickly talk through the activities to be conducted in Task A and Task B sections of the experiments (reference – participant info sheet)
    ii. Go through the consent form together
    iii. Signing of consent if happy
    iv. Start camera recording
b. Begin TASK A.1 – checking that participant's speech and gesture can be appropriately captured using the following commands

Preliminary measurements:

- Participant to microphone: ______________________ cm
- Microphone to speaker: _________________________ cm
- Participant to camera: ___________________________ cm
- Ambient noise level: _____________________________ dB
- Participant voice level: __________________________ dB
- Ambient Lighting level: __________________________ Lux

Control Mode	S/No.	Commands	Success/Failure	Comment (no. of trials...)
Speech	1	Go Forward		
	2	Go Backward		
	3	Step Left		
	4	Step Right		
	5	Hover		
	6	Land		
	7	Go Forward Half Metre		
	8	Go Backward One Metre		
	9	Hover One Metre		
	10	Step Left Half Metre		
	11	Step Right One Metre		
	12	Stop		
Gesture	1	Forward		
	2	Backward		
	3	Right		
	4	Left		
	5	Stop		

c. Begin TASK A.2 – Vary noise level
   i. Start noise program and vary from 50 dB to 90 dB and record in table below. Circle successes and strike out failures at louder noise levels

Control Mode	S/No.	Commands	Success/Failure	Comment (no. of trials...)
55 dB	1	Go Forward		
	2	Go Backward		
	3	Step Left		
	4	Step Right		
	5	Hover		
	6	Land		
	7	Go Forward Half Metre		
	8	Go Backward One Metre		
	9	Hover One Metre		
	10	Step Left Half Metre		
	11	Step Right One Metre		
	12	Stop		
60 dB	1	Go Forward		
	2	Go Backward		
	3	Step Left		
	4	Step Right		
	5	Hover		
	6	Land		
	7	Go Forward Half Metre		
	8	Go Backward One Metre		
	9	Hover One Metre		
	10	Step Left Half Metre		
	11	Step Right One Metre		
	12	Stop		

65 dB	1	Go Forward		
	2	Go Backward		
	3	Step Left		
	4	Step Right		
	5	Hover		
	6	Land		
	7	Go Forward Half Metre		
	8	Go Backward One Metre		
	9	Hover One Metre		
	10	Step Left Half Metre		
	11	Step Right One Metre		
	12	Stop		
70 dB	1	Go Forward		
	2	Go Backward		
	3	Step Left		
	4	Step Right		
	5	Hover		
	6	Land		
	7	Go Forward Half Metre		
	8	Go Backward One Metre		
	9	Hover One Metre		
	10	Step Left Half Metre		
	11	Step Right One Metre		
	12	Stop		
75 dB	1	Go Forward		
	2	Go Backward		
	3	Step Left		
	4	Step Right		
	5	Hover		
	6	Land		
	7	Go Forward Half Metre		
	8	Go Backward One Metre		
	9	Hover One Metre		
	10	Step Left Half Metre		
	11	Step Right One Metre		
	12	Stop		
80 dB	1	Go Forward		
	2	Go Backward		
	3	Step Left		
	4	Step Right		
	5	Hover		
	6	Land		
	7	Go Forward Half Metre		
	8	Go Backward One Metre		
	9	Hover One Metre		
	10	Step Left Half Metre		
	11	Step Right One Metre		
	12	Stop		
85 dB	1	Go Forward		
	2	Go Backward		
	3	Step Left		
	4	Step Right		

	5	Hover		
	6	Land		
	7	Go Forward Half Metre		
	8	Go Backward One Metre		
	9	Hover One Metre		
	10	Step Left Half Metre		
	11	Step Right One Metre		
	12	Stop		

d. Begin Task A.3 – vary visibility
   i. Vary room visibility with Start noise program and vary from 50 dB to 90 dB and record in table below. Circle successes at louder noise levels

Light Temperature	Light Intensity	Fingers	Commands	Success/Failure	Comments
White	Minimum Room Lux ________	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		
	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		
	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		
	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		

Light Temperature	Light Intensity	Fingers	Commands	Success/Failure	Comments
Yellow	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		
	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		
	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		
	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		

Light Temperature	Light Intensity	Fingers	Commands	Success/Failure	Comments
Both (White + Yellow)	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		
	______ Lux	1	Forward		
		2	Backward		
		3	Right		
		4	Left		
		5	Stop		

e. Begin Task B1.1 – flight test

Table 7-3: Participant ________________________ with __________ hrs /__________ years' flight experience RealFlight drone simulator flight test results. Flying since (month/year)____________

Date	Level	Score	Time (s)	No. of trials	Outcome	Time to beat (s)
	1					
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					

f. Begin Task B1.2 – RC Joystick flight path_v02 control test task
   i. Queue UAV in starting area and start vokoscreen capture
   ii. Queue participant to start navigation and Start Stopwatch
   iii. Record Task B1.2 completion time

Completion Time: ____________ Minutes ______________ Seconds

Completion Time (in seconds only): __________________ Seconds

   iv. Stop vokoscreen capture
   v. Participant completes the NASA TLX cognitive workload survey questionnaire for Task B1.2

g. Begin Task B1.3 – RC Joystick flight path_v02 control test task with hover thrust control (similar to mSVG)
   i. Queue UAV in starting area and start vokoscreen capture
   ii. Queue participant to start navigation and Start Stopwatch
   iii. Record Task B1.3 completion time

Completion Time: ____________ Minutes ______________ Seconds

Completion Time (in seconds only): __________________ Seconds

iv. Stop vokoscreen capture
v. Participant completes the NASA TLX cognitive workload survey questionnaire for Task B1.3

h. Begin Task B2.1 – familiarization with speech and gesture commands

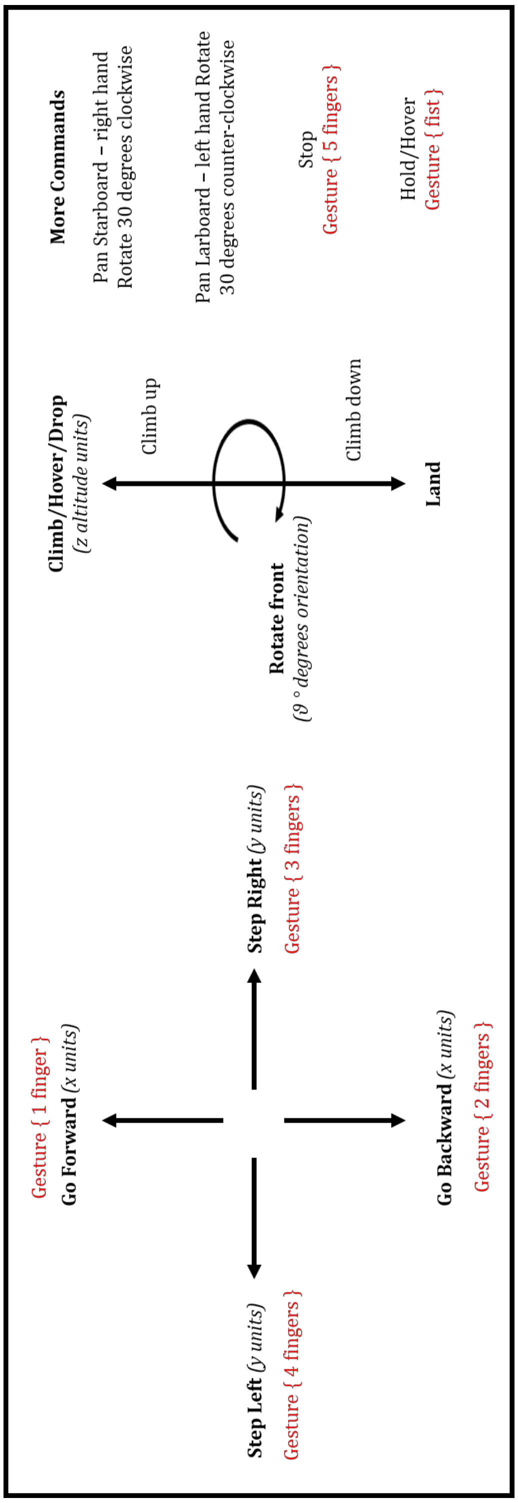


Figure 7-26 : mSVG navigation control command syntax

i. Begin Task B2.2 – mSVG flight path_v02 control test task
   i. Queue UAV in starting area and start vokoscreen capture
   ii. Queue participant to start navigation and Start Stopwatch
   iii. Note when and where speech only, gesture only, or both commands were issued simultaneously, and perhaps why.

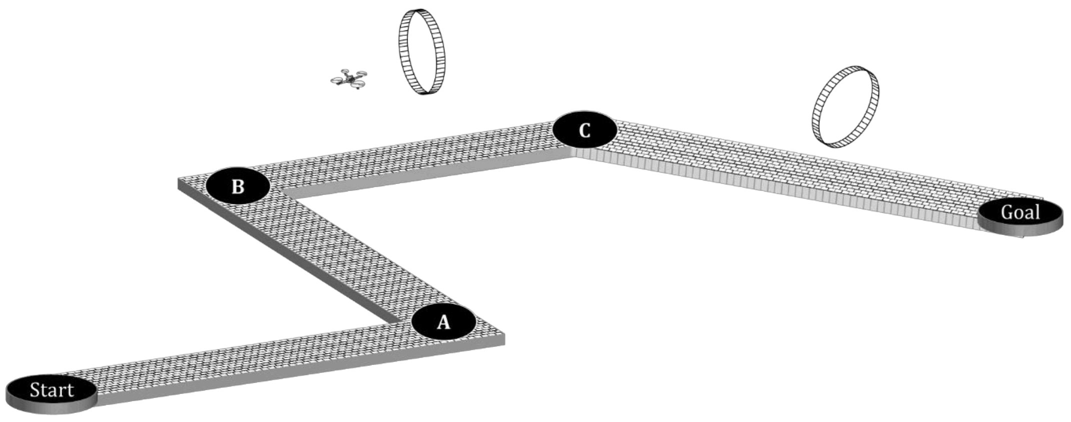


Figure 7-27: Flight Path_v02 design

S/No.	**Navigation Stage**	**Speech only**	**Gesture only**	**Speech + Gesture**	**Time to marker pts (s)**	**Comments**
1	Start - hovering					
2	Start-A (1/4)					
3	Start-A (2/4)					
4	Start-A (3/4)					
5	Start-A (4/4)					
6	A - Orient/step					
7	A-B (1/4)					
8	A-B (2/4)					
9	A-B (3/4)					
10	A-B (4/4)					
11	B - Orient/step/none					
12	B-C (1/6)					
13	B-C (2/6)					
14	B-C (3/6) Window ascent/descent					
15	B-C (4/6) Window go through					
16	B-C (5/6)					
17	B-C (6/6)					
18	C - Orient					
19	C-Goal (1/6)					
20	C-Goal (2/6)					
21	C-Goal (3/6) Window ascent/descent					
22	C-Goal (4/6) Window go through					
23	C-Goal (5/6)					
24	C-Goal (6/6) - landing					

iv. Record Task B2.2 completion time

Completion Time: ____________ Minutes ______________ Seconds

Completion Time (in seconds only): __________________ Seconds

Room visibility (lux): _________ Room Noise Level (dB): ______ Room Temperature (^{o}C): _____

v. Stop vokoscreen capture
vi. Participant completes the NASA TLX cognitive workload survey questionnaire for Task B2.2

7.8.1.3 Post-experiment (data verification, storage, set down, clear up, analysis)

a. Stop camera recording
b. Amazon voucher

D.6 NASA TLX Survey Questionnaire

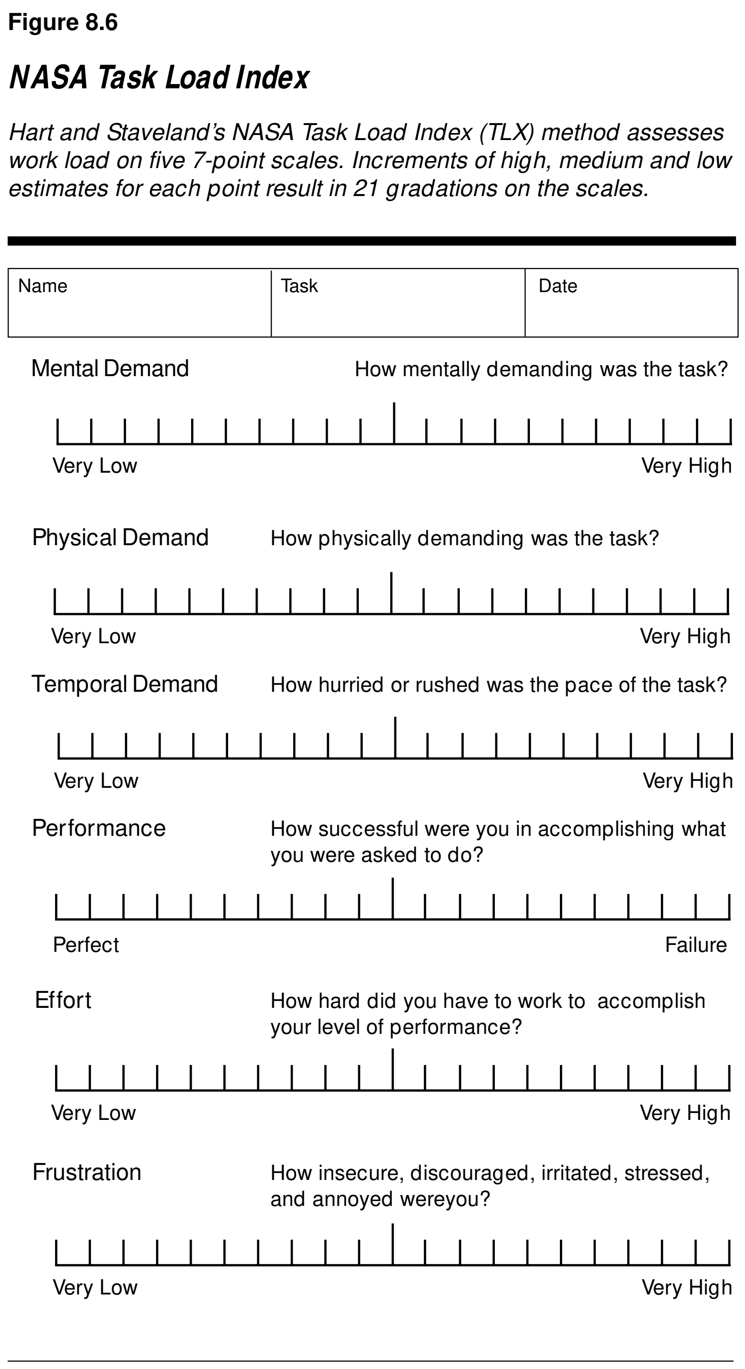


Figure 8.6

NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?

Very Low — Very High

Appendix E

Publications

E.1 Journal I (2019) - Applied Sciences - Published

Abioye, A. O., Prior, S. D., Saddington, P. and Ramchurn, S. D. (2019) Effects of Varying Noise Levels and Lighting Levels on Multimodal Speech and Visual Gesture Interaction with Aerobots, Applied Sciences. Basel, Switzerland: MDPI, 9(10), pp. 1-29. https://doi.org/10.3390/app9102066

Available on thesis CD-R "*./Appendix E - Publications/*" folder.

E.2 Book Chapter I (2017) - IGI Global - Published

Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P. and Ramchurn, S. D. (2017) Multimodal Human Aerobotic Interaction, in Issa, T., Kommers, P., Issa, T., Isaas, P., and Issa, T. B. (eds) Smart Technology Applications in Business Environments. Pennsylvania, USA: IGI Global, pp. 39-62. Chapter 3.

Available on thesis CD-R "*./Appendix E - Publications/*" folder.

E.3 Conference Paper I (2016) - IADIS IHCI - Published

Abioye, A. O., Prior, S. D., Thomas, G. T. and Saddington, P. (2016) The Multimodal Edge of Human Aerobotic Interaction, in Blashki, K. and Xiao, Y. (eds) International Conferences Interfaces and Human Computer Interaction. Madeira, Portugal: IADIS Press, pp. 243-248.

Available on thesis CD-R "*./Appendix E - Publications/*" folder.

E.4 Conference Paper II (2018) - IEEE ICASI - Published

Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P. and Ramchurn, S. D. (2018) Quantifying the effects of varying light-visibility and noise-sound levels in practical multimodal speech and visual gesture (mSVG) interaction with aerobots, in Meen, Prior, and Lam (eds) 2018 IEEE International Conference on Applied System Invention (ICASI). Chiba, Tokyo, Japan: IEEE, pp. 842-845. https://doi.org/10.1109/ICASI.2018.8394395.

Available on thesis CD-R "*./Appendix E - Publications/*" folder.

E.5 Conference Paper III (2018) - Springer TAROS - Published

Abioye, A. O., Prior, S. D., Thomas, G. T., Saddington, P. and Ramchurn, S. D. (2018) The Multimodal Speech and Visual Gesture (mSVG) Control Model for a Practical Patrol, Search, and Rescue Aerobot, in Giuliani, M., Assaf, T., and Giannaccini, M. E. (eds) Towards Autonomous Robotic Systems - Part of the Lecture Notes in Computer Science book series (LNCS, volume 10965). Bristol, UK: Springer International Publishing, pp. 423-437. https://doi.org/10.1007/978-3-319-96728-8_36.

Available on thesis CD-R "*./Appendix E - Publications/*" folder.

E.6 Conference Posters I (2015) - NEC Birmingham - Unpublished

Abioye, A. O., Prior, S. & Thomas, T. "Aerial robot control interfaces - Poster" NEC Birmingham, November 2015.

Available on thesis CD-R "*./Appendix E - Publications/*" folder.

E.7 Conference Posters & Abstracts II (2017) - UK-RAS Network Conference - Unpublished

Abioye, A. O., Prior, S., Thomas, T., Saddington, P. & Ramchurn, S. "A practical mSVG interaction method for patrol, search, and rescue aerobots - Extended Abstract and Poster" In UK-RAS Network Conference (December) 2017, Bristol, UK.

Available on thesis CD-R "*./Appendix E - Publications/*" folder.