UNIVERSITY OF SOUTHAMPTON

Crowd Robotics: Real-time crowdsourcing for crowd controlled robotic agents

by

Elliot Salisbury

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
Electronics and Computer Science

April 2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Elliot Salisbury

Major man-made and natural disasters have a significant and long-lasting economic and social impact on countries around the world. The response effort in the first few hours of the aftermath of the disaster is crucial to saving lives and minimising damage to infrastructure. In these conditions, emergency response organisations on the ground face a major challenge in trying to understand what is happening, and where the casualties are. Crowdsourcing is often used in disasters to analyse the masses of data generated, and report areas of importance to the first responders, but the results are to slow to inform immediate decision making. This thesis describes techniques for utilising real-time crowdsourcing to analyse the disaster data in real-time. We utilise this real-time analysis to influence or control robotic search agents, unmanned aerial vehicles, that are increasingly being used in disaster scenarios. We investigate methods for reliably and promptly aggregating real-time crowd input, for two different crowd robotic applications. First, direct control, used for directing a robotic search and rescue agent around a complicated and dynamic environment. Second, real-time locational sensing, used for rapidly mapping disasters and to augment a pilot's video feed, such that they can make more informed decisions on the fly, but could be used to inform a higher artificial intelligence process to direct a robotic agent. We describe two systems, CrowdDrone and CrowdAR, that use state-of-the-art methods for human-intelligent control and sensing for crowd robotics.

Contents

List of Figures

Acknowledgements

I would like to express my gratitude towards my supervisors, Gopal and Seb, for their support throughout my PhD and for their continued guidance as I begin my journey into the working world.

Thank you to all my lab mates who helped make the lab an enjoyable place to work. Except for Fred and James, whose constant arguments and squabbling is something I could frankly have done without.

Finally, thanks to all the members, new and old, of Southampton University Mountaineering Club, whose friendship and adventure loving spirit, has taken us to some of the best places in the world. Thank you for making my time here at Southampton so great. I will miss you all.

Chapter 1

Introduction

In recent years, the advances in network speeds and accessibility have allowed for an unprecedented amount of social connectivity and user-generated content. Online communities have formed that involve millions of people from around the world, encouraging diversity and independent creativity. These collectives are often referred to as online crowds (von Ahn and Dabbish, 2004). Notably, it has been shown that aggregating human intelligence, from diverse backgrounds and experience, can be more accurate than any individual judgement made by a single member of the crowd (Galton, 1907) (i.e., the wisdom of the crowd). This effect can be harnessed as part of a process called crowdsourcing.

Crowdsourcing is the mechanism by which the combined intelligence contained within a crowd can be harvested in order to take advantage of its invaluable qualities (Howe, 2006). Additionally, humans are better at some tasks than a computer, for example, vision-based problems or contextual understanding. Thus a large and computationally challenging problem can be crowdsourced by reducing it into many microtasks (i.e., small and easily accomplished tasks) that are then given to a large online crowd to solve. Once the crowd has finished every microtask, the results are aggregated to solve the larger problem. The calculated solution tends to be more accurate, takes less time, and is less costly than it otherwise would have been to process the problem with an expert, or solely by computational means.

Thus, a key step in the crowdsourcing process is the aggregation of human input, that is, their opinions, judgements or ideas. Simple methods can sometimes be very effective, such as taking the mean judgement (e.g., product reviews on Amazon), or by taking the majority's opinion (e.g., democratic elections). However, novel applications of crowdsourcing that have recently emerged involve complex scenarios that require new techniques to aggregate opinions. For example, crowdsourcing has been applied to the problem of detecting events through aggregated tweets or text messages (Morrow et al., 2011; Sakaki et al., 2010) or for enabling blind users to ask visual questions (Bigham et al., 2010).

Due to the growth of crowdsourcing platforms, large numbers of workers are now available online simultaneously. This has led to the development of real-time crowdsourcing (Bernstein et al., 2011), in which crowd input is elicited, aggregated, and a solution calculated within a matter of seconds. This responsiveness enables applications that appear automated, as they interact with end users in real-time, yet their responses still retain human intelligence. There currently exists a small number of applications that apply real-time crowdsourcing principles, for example, conversational assistants for blind users (Lasecki et al., 2013c), or tools for sociologists to rapidly detect and analyse subjective events in large video datasets (Lasecki et al., 2014).

One such field of real-time crowdsourcing is crowd robotics, the influence or control of robotic agents through an online real-time crowd (Lasecki et al., 2011). Crowd robotics adds human intelligence to the sensing tasks that robots perform (e.g., sensing where to go next, or the location of search targets), bridging the gap between the limitations of Artificial Intelligence (AI) and the requirements of robotic end users.

A key application area where crowd robotics could have a major impact is that of disaster response. Major man-made and natural disasters have a significant and long-lasting economic and social impact on countries around the world. The response effort in the first few hours of the aftermath of the disaster is crucial to saving lives and minimising damage to infrastructure. In these conditions, emergency response organisations on the ground face a major challenge in trying to understand what is happening, where the casualties are, and how best to rescue them. In order to help support this response effort, satellite imagery and aerial footage are increasingly being used by emergency responders and charitable organisations (e.g., Rescue Global[1], Red Cross[2]), to help map disasters. However, the vast amount of imagery coming from such sources takes significant time (i.e., hours or even days) to analyse in order to determine points of interest or clear rescue targets. This is not fast enough in the highly dynamic and time-critical settings we consider. Here, targets need to be identified immediately to enable a quick response or to direct the collection of imagery in real-time (e.g., towards areas of high interest, or more closely examine potential rescue targets).

Search and rescue organisations use manned and unmanned aircraft (i.e., UAVs) equipped with cameras to assess a situation and find areas of interest. The aerial imagery from these flights is typically reviewed retrospectively (i.e., after the flight), but this approach has a number of drawbacks. First, if the aerial vehicle is flown manually, in order to record footage for later crowdsourced analysis, the pilot must maintain *continuous focused attention* so that they can spot search targets (e.g., damaged buildings, floating

[1]`www.rescueglobal.org`
[2]`www.redcross.org.uk`

wreckage in the ocean) and then fly the vehicle to get a closer look. Inevitably, the pilot will eventually become fatigued, as these tasks are typically tedious (e.g., flying over an ocean) or visually taxing (e.g., flying over a city), and thus crucial information may be missed (Cummings et al., 2010). Furthermore, it is hard to envisage that such an approach would scale to hundreds or thousands of UAVs in the future. Second, if the aircraft is flown on a fixed flight path, as is typical with UAVs, then the benefit of diverting off course when a target is identified (post-flight) is lost, which can enable responders to gather imagery of potentially important regions that lie at the boundary of the vehicle's field of view. Third, in the past, standard crowdsourcing tools (e.g., MicroMappers[3], TomNod[4]) have been used to review aerial footage post-flight (Meier, 2014d,b). Due to the delayed nature of this crowdsourcing effort, responders who launched the UAV cannot get quantifiable results (e.g., the location of search targets, or the extent of damage and where to prioritise rescue efforts) immediately. Thus, using aerial footage for situational awareness can still be a time consuming and costly process.

Automating the use of UAVs by search and rescue organisations would require AI and Computer Vision algorithms. These algorithms need to be capable of reliably detecting any number of search targets, despite ever changing conditions (e.g., lighting conditions, orientation, partial occlusion). Work has been done to train computer vision classifiers for this purpose. For example, algorithms have been developed to detect damaged buildings (Ouzounis et al., 2011), or injured people (Teacy et al., 2015). However, as disasters are dynamic and often have different search requirements, it is infeasible to have an algorithm for each eventuality. Instead, a more general purpose approach would describe the target in natural language and use human perception to understand and detect them.

Therefore, real-time crowdsourcing enables the analysis of aerial footage to take place live, while the aircraft (and potentially any mobile camera) is flying. In our case, when applied to disaster response, we can use a real-time crowd to find and tag any number of search targets in a live video feed. Their annotations can then be aggregated and used to rapidly map disasters, and augment the live video feed seen by the aircraft's pilot, thus reducing their cognitive load. Furthermore, the use of real-time crowdsourcing has a number of benefits over a single expert pilot. First, during a disaster the widespread media attention draws in a lot of online volunteer workers who are willing to perform many types of crowdsourcing tasks. This provides a powerful and free workforce, but with the drawback that many of these workers may be inexperienced. Second, many more people may observe the live camera feed, and in so doing, they are more likely to observe and accurately identify important details than a single pilot could. Third, while a single observer may become fatigued, a real-time crowdsourced team can be a tireless workforce. Provided that mechanisms are in place such that individuals of

[3]`www.micromappers.wordpress.com`
[4]`www.tomnod.com`

the crowd that become too tired and stop contributing to the task, can be replaced by newly recruited crowd workers, then the workforce is always being replenished with fresh individuals. This mechanism is one of the challenges involved in the recruitment of real-time crowds. Finally, through the use of in-flight analysis, we reduce the delay between data gathering and the reviewing of footage. More importantly, in-flight analysis allows us to react to events or features the crowd has spotted (e.g., injured people in the distance, or footprints leading off-trail) and move the robot off its original search path to investigate.

Against this background, in this thesis, we develop novel techniques for real-time crowd-sourcing that enable human-intelligent sensing for the use of manipulating robotic agents during situational awareness missions. In what follows we describe the challenges and potential solutions to real-time aggregation of crowd input for human-intelligent sensing and influencing the control of robotic agents in real-time environments.

1.1 Problem Statement

In disaster response operations that gather situational awareness with UAVs, current practice is to use a team of UAV pilots. These teams are typically limited in size due to the cost of employment and transportation to the disaster site. In contrast, crowd robotics could provide a much larger team with which to operate a robot without the constraints of a physically present team of expert pilots. This thesis addresses the challenges with deploying crowd robotics within the context of disaster response operations.

Crowd robotics requires that we combine the input of many individuals in the crowd, in such a way to create a timely and more effective output than any individual in the crowd could provide. To provide effective influence on the robot's operations, we need to address the challenge of selecting input that is beneficial to the disaster responders' search efforts. Ideally, input would provide sensible information to the robot or human operators that enable them to take actions that support their goal (e.g., in the domain of search and rescue operations, these may include following an optimal search path, or investigating the location of detected search targets). In a more constrained setting, we recognise that machine learning techniques could be used for similar purposes. For example, supervised machine learning techniques could be capable of learning how to perform effective search and rescue operations, or identify search targets in a video feed. However, in the domain of humanitarian aid, objective ground truth data is often hard to come by or unavailable. Without ground truth data, machine learning techniques that require we have labelled examples of when an outcome is favourable or a target present, is infeasible.

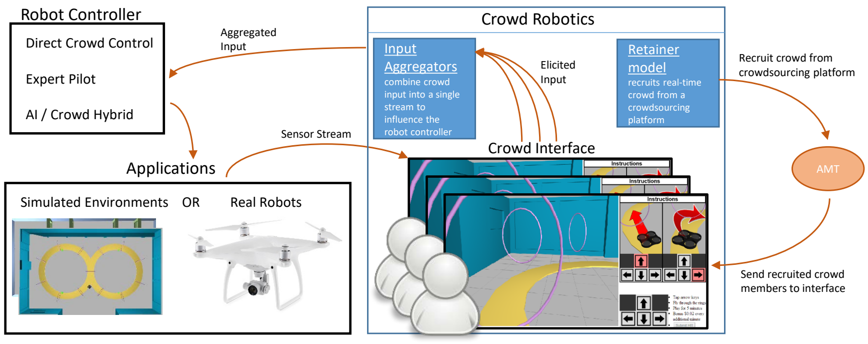

Robot Controller
Direct Crowd Control
Expert Pilot
AI / Crowd Hybrid
Aggregated Input
Crowd Robotics
Input Aggregators
combine crowd input into a single stream to influence the robot controller
Elicited Input
Retainer model
recruits real-time crowd from a crowdsourcing platform
Recruit crowd from crowdsourcing platform
AMT
Sensor Stream
Crowd Interface
Applications
Simulated Environments OR Real Robots
Send recruited crowd members to interface

Figure 1.1: Diagram showing the general architecture of a crowd robotics system.

Furthermore, as disasters are dynamic and often have different search requirements, it is infeasible to have an algorithm for each eventuality. In situational awareness operations, we do not know the optimal search path, and training computer vision classifiers despite ever changing conditions (e.g., lighting conditions, orientation, partial occlusion), wildly different search targets (e.g., damaged buildings, people), or visually different targets in each deployment (e.g., cultural and architectural differences may effect the appearance of buildings) means that training an algorithm for each eventuality is infeasible. Crowd robotics aims to be a general purpose solution, and by using human intelligence we do not need masses of unavailable data to train an AI system.

Furthermore, for such a crowd robotics system to work, it must solve the challenges raised in Section 1.3. In brief, those challenges are as follows. First, it must be capable of recruiting a number of simultaneously connected workers, which for practical reasons would be gathered online with a crowdsourcing platform, such as Amazon Mechanical Turk[5] (AMT). Second, it needs to maintain the crowd size by recruiting new workers when others disconnect, and to encourage participation from the already connected crowd, ensuring their task is both engaging enough that the workers wish to perform the task for longer, and that the workers provide frequent and meaningful input to the crowd robotic system. Third, because crowd robotics systems may often require rapidly recruited crowd workers (e.g., when a robot is deployed with short notice giving little time to recruit the crowd) there needs to be a low barrier to entry. A barrier to entry, such as requiring workers to have been preselected, or to have proved their value through qualifications (i.e., a common way to filter for high quality workers on AMT) would reduce the number of workers that could be rapidly recruited. Thus, the recruited workers are potentially unskilled or untrustworthy, and by providing poor input, could then negatively affect the search, wasting valuable time and resources. Instead, a crowd robotics system requires a method of calculating operator reliability on the fly, and limiting the influence of unreliable or malicious participants.

Subsequently, this requires that the robot maintains a constant connection to the internet, such that it can stream its sensors to the crowd and receive their feedback. A

[5] `www.mturk.com`

real-time crowd robotics system can use the agreement between crowd workers to calculate operator reliability and help filter undesirable actions introduced by unreliable users. For example, Figure 1.1 shows the architecture of a crowd robotics system, in which a real-time crowd provides directional commands to an input aggregation algorithm. The result of this aggregation is passed to the controller of a robot, that acts on the combined input and streams the robot's camera footage back to the crowd. The crowd having seen this movement, can provide new input to direct the robot further. It is currently unknown how best to build a real-time crowd robotic system, how the aggregation of real-time input affects the robotic agent, or how the crowd workers would appropriate such a system. Below is a set of requirements that such a system must meet:

I **Open System:** The system should be open to use from any workers of the online crowd. Reducing barriers to entry helps to recruit and utilise many simultaneous workers. However, this means that workers may vary greatly in their experience, skill, and reliability. Although expert or vetted crowd workers may also be used, and is discussed in Chapter 6.

II **Agent:** There needs to exist some automated search agent (e.g., a robot, a human with a live camera feed) whose actions can be influenced by the collective input of the crowd.

III **Real-Time:** The feedback from the crowd should be timely enough to provide sensor input to the search agent, thereby influencing their path.

IV **Unsupervised:** The system should work without the need for direct human supervision ensuring that aggregated crowd output remains beneficial to the search efforts. Hence, because of Requirement I, the system needs to be able to evaluate the reliability of its input, and invalidate unreliable input.

V **Robust to Noise:** While we expect that the crowd will be for the most part benevolent, the system must be capable of filtering out noise generated from unintentional commands, users experiencing lag, and malicious input.

VI **Scalability:** As more robots are added to the system, the performance should not be detrimentally affected. The size of the crowd should be scalable too, crowd workers can dynamically join and leave, and the system should scale from very few crowd workers to very large numbers of workers without sacrificing responsiveness or reliability.

Next, we discuss the current practice in situational awareness operations, and show how these can be improved with crowd robotics.

1.2 Existing Solutions

So far this work has discussed crowd robotics, but there are a number of other approaches to gathering situational awareness that either do not use a crowd (Requirement I), do not use a robot (Requirement II), or do not perform in real-time and thus not impacting the search efforts effectively (Requirement III).

Firstly, search and rescue teams already use UAVs for their planning and search phases, and they have already been shown to help save lives in the field (Ouzounis et al., 2011; Drone Adventures, 2014). These operations are typically performed either using fixed flight paths, which do not satisfy Requirement III, or through a human operator, which does not satisfy Requirement VI. In addition to this, these methodologies do not utilise crowdsourcing to alleviate the stress and monotonous nature of these tasks, and so Requirement I is not met.

Secondly, crowdsourcing platforms, such as Tomnod[6] (Lin et al., 2013) and CrisisMappers[7], search satellite imagery, classifying where there is damage or danger. These solutions satisfy Requirements I and V but they do not meet Requirement IV, as human supervision is still needed to verify the targets found by the crowd. Furthermore, they do not satisfy Requirement III (i.e., influencing the path of an active search agent). As such, some of the data gathered may be wasteful, and fail to fully search areas that the crowd would deem important.

Finally, the MicroMappers crowdsourcing platform can be used to handle the footage from a UAV (Meier, 2014d). This meets Requirements I, II, and V. However, post flight imagery is uploaded to the platform and it is only then that the online crowd can assess the amount of damage captured in those images, failing to satisfy Requirement III.

Notably, each of the above approaches suffer from an inefficient delay between gathering the data, and assessing it, and thus each fail to meet Requirement III. In turn, in this report we develop algorithms that achieve a more effective and accurate assessment of the footage by focusing instead on in-flight review. Thereby creating a new type of sensor feed, a human-intelligent sensor, that uses human perception to interpret the environment and can be used to influence the flight path of the UAV to cover a more relevant search area than a fixed flight path. This is further discussed in Chapter 2. Next we discuss a number of challenges involved in a crowd robotics system.

[6]www.tomnod.com
[7]www.crisismappers.net

1.3 Research Challenges

There are a number of potential challenges that this research attempts to address. First, any crowd robotics system requires a means to recruit, redirect, maintain, and pay a real-time crowd. Second, we require a method to reliably and quickly elicit meaningful data from a crowd of potentially inexperienced workers. Third, crowd robotics uses inputs that are inherently diverse in their accuracy and timeliness, given that crowd workers vary in performance and latency. Finally, crowdsourced workers are self-interested, they have their own goals and may not act in the crowd robotic system's best interests. We will address each in turn.

1.3.1 Real-time Crowd Recruitment and Management

Real-time crowdsourcing requires that we have a collection of users who are all simultaneously connected. In order to harness the power of real-time crowds, we need a method of recruiting crowd workers, redirecting them to the task once enough have been recruited, maintaining the recruited crowd size, and finally paying the crowd when the task is done. Maintaining a crowd size is important as many of the tasks that use real-time crowdsourcing require a multitude of workers simultaneously connected in order for the aggregation algorithms to effectively filter out the noise created from latency and from unreliable crowd workers. However, as members of the crowd are free to leave at any time, it is also important to replace them with new crowd workers so that the quality of the aggregated output does not drop, thus ensuring that the system scales (Requirement VI). Therefore, it is important to develop tools to enable the recruitment and management of real-time crowds, specifically involving rapid recruitment strategies that can provide a consistent crowd size, despite worker drop out.

1.3.2 Eliciting Input from the Crowd

Crowd workers sourced from open platforms (i.e., crowdsourcing marketplaces, such as AMT) come from very different backgrounds and have very different skills. Hence it has been shown that the methods used to elicit input from the crowd should be simple and intuitive, ensuring that crowd workers promptly understand their task (Law and Ahn, 2011). The method for eliciting input from a crowd worker must be quick such that the input remains timely and relevant to robotic control, and the aggregation algorithms that use this input must run in real-time so that the aggregated output can be used to influence the robot on the fly. Thus, we require interfaces to be designed with this real-time interaction in mind. Human performance at these tasks must be considered into the design of the elicitation method (i.e., how fast vs. how accurate can humans provide simple input). Furthermore, the type of input (e.g., mouse clicks, button presses) elicited from the crowd affects which methods of aggregation are applicable. Thus the elicitation method needs careful consideration.

1.3.3 Latency

The network performance of crowd workers often varies greatly due to regional differences and their internet service providers' performance. This results in disparities in the rate at which input is received from a large population of diversely located crowd workers, and thus their late input may be different and may not agree with the majority. This difference is not because they intentionally disagree with the majority, but because their input is about a different context. Hence, there is a need to apply corrective measures to the input of crowd workers to reduce the effects of unavoidable latency.

1.3.4 Reliability of Crowd Workers

Because some crowd robotic systems may enlist the help of anyone (Requirement I), and because some crowd robotics systems involve participants that may be paid for their time, there may be crowd workers who will game the system to perform the task while putting in the least amount of effort. Such crowd workers are referred to as being malicious, deliberately providing incorrect or non-sensible input to the system. Therefore, whatever the worker's intentions, it is important to model those workers' reliability and give less credence to the opinions of those who are untrustworthy. Furthermore, calculating a worker's reliability is not a simple task. We cannot simply reward workers who give favourable input (i.e., accurately identifying search targets), because in real settings it is neither simple or practical to detect whether a worker's input led to a favourable outcome. Computer vision is not currently capable of verifying search targets accurately (Felzenszwalb et al., 2010; Bak et al., 2010), and using an expert to inspect the outcome would not satisfy Requirement IV.

From this discussion we can conclude that the problem of crowd robotics is difficult and challenging. No off-the-shelf solution exists for our problem at present. Thus, we aim to conduct our research to solve these challenges.

1.4 Research Objectives

Set against the challenges listed in the previous section, the objectives of this research are as follows:

To develop algorithms and mechanisms to recruit and utilise online crowds for real-time crowd robotics. In particular, we aim to design and develop scalable solutions that enable *the crowd to provide accurate and effective input, yet timely enough that it can be used in a real-time crowd robotic system. The algorithms suitable for crowd robotic systems will be capable of coping with the issues of latency and unreliable workers. Specifically, these solutions ultimately aim to control robotic search agents to carry out situational awareness tasks.*

As discussed in Section 1.2, existing approaches to situational awareness tasks do not meet the requirements (see Section 1.1) for real-time crowd robotics. We expand on this in Chapter 2. Existing solutions typically suffer from delayed feedback, from the time the data gathering takes place, to the time the data gets analysed and the information can be used to form an effective rescue operation.

Hence, in this thesis, we propose a number of novel approaches to utilising a real-time crowd, through the use of various elicitation and aggregation methods designed for real-time control. Moreover, we propose methods for benchmarking these approaches so that we may compare and contrast the methods of aggregating crowd input in the context of crowd robotics. By so doing we expect to create a more accurate and timely situational awareness tool than simply using an individual person or small team to pilot a UAV.

1.5 Research Contributions

In disaster response settings, we envision two scenarios in which a crowd is used to provide intelligent sensing. First, in situations where an AI fails to identify a sensible path to take, the crowd can provide directional sensing, to help guide the robotic agent. Second, in situations where rapid assessment of information is required and AI is not capable of analysing the imagery, the crowd can search for the location of described targets in the live imagery, enabling rapid mapping and the augmentation of an expert pilot's video feed, allowing the expert to make more informed decisions.

The first contribution of this work is a real-time recruitment tool, described in Chapter 3, that enables a requester (i.e., a user of the tool who wishes to request workers to take part in a real-time crowdsourced task) to rapidly recruit a simultaneously-connected crowd of online workers. This tool allows a separate agent to control the recruitment strategies, automating the process of posting jobs to online marketplaces, and intelligently stopping and starting recruitment to achieve a desired number of workers. The tool provides an intuitive interface for requesters to create tasks, begin recruitment sessions, and to monitor and eventually pay the workers that are recruited in the session.

The second contribution of this work is a range of novel aggregation methods for directional sensing, see Chapter 4. In brief, a number of voting mechanisms are developed that calculate crowd workers' reliability based on their agreement with the rest of crowd, and then either select the most reliable user as sole controller of the robot, or apply a weight based on their reliability to their vote in a plurality election for the direction the robot should move towards. Some schemes infer a natural ordering to the voted on actions, and attempt to rank the available action space. These methods are then evaluated in our novel benchmarking environment, CrowdDrone. CrowdDrone uses a simulation environment that accurately simulates the flight characteristics of a UAV and provides a challenging environment for the crowd to fly through, while collecting metrics about their performance.

The third contribution is a novel technique for human intelligent locational sensing, detailed in Chapter 5. This can be used, for example, to augment a pilot's video feed with aggregated information from a real-time crowd. We elicit spatial and temporal information about targets in a video stream from a real-time crowd. Through the use of computer vision techniques and clustering algorithms for calculating consensus, we develop a tool that can be used in a number of novel applications, such as an overlay that reacts in real-time to a video stream that highlights search targets to the expert pilots, or rapidly mapping disaster zones as the UAV is flying over head. We evaluate the performance of this system using both paid and volunteer crowds in a humanitarian setting, discussed in Chapter 6. We expect that such systems would likely be used by both types of crowd, and by getting insight from both target audiences we can identify a set of guidelines for improving real-time crowdsourcing, and for improving the functionality of platforms for humanitarian volunteers to use.

1.5.1 Publications

Conference proceedings:

- E. Salisbury, S. Stein, and S. Ramchurn. Real-time Opinion Aggregation Methods for Crowd Robotics. In *Autonomous Agents and Multiagent Systems (AAMAS 2015)*, May 2015b
- E. Salisbury, S. Stein, and S. Ramchurn. CrowdAR: Augmenting Live Video with a Real-Time Crowd. In *Conference on Human Computation & Crowdsourcing (HCOMP 2015)*, November 2015a
- E. Salisbury, S. Stein, and S. Ramchurn. Crowdar: a live video annotation tool for rapid mapping. *Procedia Engineering*, 159:89–93, 2016

In the media:

- **iRevolutions:** Using Computer Vision to Analyze Aerial Big Data from UAVs During Disasters[8]

[8]https://irevolutions.org/2015/10/12/computer-vision-big-data-uavs/

- **MicroMappers:** Results: Crowdsourcing the Analysis of UAV Video Feeds for Disaster Response[9]

1.6 Report Outline

The remaining chapters of this work are structured as follows:

Chapter 2 provides a background of crowdsourcing in disaster response and the application of unmanned vehicles. Then we go on to detail the literature on real-time crowdsourcing. Finally we discuss typical opinion aggregation techniques, and key voting protocols that we exploit.

Chapter 3 describes a tool for real-time recruitment and management of workers from crowdsourcing marketplaces. We begin the chapter explaining the requirements of the recruitment manager, and then discuss how it is implemented and the design decision that were made. We show how a user interacts with the recruitment manager, and we go on to describe the concept of a recruitment agent. The recruitment agent is responsible for dynamically changing recruitment strategies to ensure a constant supply of real-time crowd workers. Finally, we describe the recruitment agent used by this work, and go on to suggest improvements that could be made to future recruitment agents.

Chapter 4 proposes novel approaches for aggregating real-time input in order to direct a robotic agent, enabling low-level control decisions to be aggregated from real-time crowd input. The chapter begins with a formal definition of the problem. Next, we define a number of approaches used for input aggregation in the domain of low-level real-time control. This is followed by a description of the experimental setup, and the benchmarking process. Finally the results of the experiment are presented and discussed.

Chapter 5 proposes a novel approach to eliciting locational information from crowd workers in real-time. This creates a real-time human-intelligent sensor for detecting any targets that can be described in natural language. We use this for a number of humanitarian focused applications, such as augmenting a pilot's sensor feed with information gathered from a real-time crowd. The chapter describes the approach taken and details a formal definition of the problem. This is followed by a description of the experimental setup, and the evaluation of the approach.

Chapter 6 studies the application of human-intelligent locational sensing to the domain of humanitarian aid. We recruit experienced volunteers from humanitarian organisations to participate in real-time human-intelligent sensing tasks. We acquire a large amount of feedback from the participants, and use these insights to produce a set of guidelines

[9]`https://micromappers.wordpress.com/2015/09/09/results-crowdsourcing-the-analysis-of%2Dvideo-feeds-from-uavs-for-disaster-response/`

to support future work in real-time crowdsourcing, in human-intelligent sensing tasks, and in crowdsourcing for humanitarian aid.

Chapter 7 summarises the outcome of the research described in this work. It also provides a detailed description of future directions that research in this area could follow.

Chapter 2

Background

The field of crowd robotics sits at the intersection between two areas of research, namely the application of robotic assistance, and the field of real-time crowdsourcing. Because our work focuses on the domain of humanitarian aid, where robots are typically used for gathering situational awareness, we review the application of robotic agents used in disaster response scenarios, in Section 2.1 and existing applications of crowdsourcing in disaster scenarios in Section 2.2.

Then, in Section 2.3, we go on to examine the emerging field of real-time crowdsourcing. Similar to traditional crowdsourcing that uses remote humans as part of a computational process, real-time crowdsourcing uses a collection of simultaneously connected users as part of a continuous computational process. Finally, we discuss related methodologies for aggregating crowd input in Section 2.4.

2.1 Unmanned Vehicles in Disaster Response

To act effectively, disaster response teams require some essential information, such as viable transit routes, hazardous zones, and signs of human presence (Rego, 2001). This information can be assessed from aerial imagery. This imagery can be gathered through numerous means, such as satellites, planes, or more recently unmanned aerial vehicles. It is well known that the faster response times result in much lower fatality rates (Quon and Laube, 1991; Marconi et al., 2013). Therefore, to be able to act effectively as early as possible during the response effort, the timely gathering of this information and then assessing the situation is of the utmost importance.

In many situations satellite imagery of a disaster site may not be available, may be prohibitively expensive to procure, or may be occluded with cloud cover. UAVs, however, can provide higher quality imaging by flying below the cloud cover, for a lower cost, and with a faster response time. In addition to this, even though commercial satellites can cover a larger area, their resolution per m^2 is relatively poor, whereas a UAV gathering aerial photos can achieve a far greater resolution. This enables those viewing the camera feed to more accurately identify targets in the footage (Meier, 2014a).

UAVs are already proving to be an effective method for government organisations and first responders to quickly gather necessary information. Emergency services in the UK are beginning to use UAVs, for example the police force has already used them to search for missing people and gather evidence for use in court (BBC News, 2017). Likewise, Rescue Global, an organisation that provides immediate crisis and disaster reconnaissance, use UAVs in their disaster response operations. They are equipped with cameras and connected via 3/4G links, ensuring constant contact with headquarters (Rescue Global, 2013). UAVs have also been widely used in recent disasters such as the Haitian earthquake (Ouzounis et al., 2011), the Filipino typhoon and the Fukushima nuclear disaster (Drone Adventures, 2014). UAVs can even perform well in extreme conditions such as high altitude, mountainous environments like those encountered in alpine rescue (Marconi et al., 2013).

Even though there are a number of positive examples of the use of UAVs, they do come with some downsides. First, due to their limited battery life and compact size, they have a very limited range compared to manned vehicles, and thus cannot search as much ground. Second, they require a skilled team to fly and monitor the sensor data returned, and this removes valuable manpower from the first responders team. Third, no matter how skilled or experienced the team is, they are only human and are therefore prone to error and fatigue, which may lead them to miss targets at the time of flight. Fourth, a single UAV search mission can return hours of footage that requires reviewing. This is a problem because the dataset is so large that the analysis can be very time consuming and costly to perform by hand. Finally, teams often observe important targets in post flight review of the footage, which is then too late to get a closer view or to react when the environment is dynamic (Meier, 2014b).

Crowdsourcing is expected to solve a number of these drawbacks. Crowdsourcing reduces these large search and analysis problems into small parallelizable tasks, while real-time crowdsourcing (Section 2.3), also enables us to reduce the issues of fatigue and potentially react to observed targets during the flight, unlike the delayed approach of current crowdsourcing methodology.

2.2 Current Crowdsourcing Solutions in Disaster Response

Crowdsourcing is the process of outsourcing work to a group of people. Websites, or web-based tools, that enable workers to perform this work, are known as crowdsourcing platforms. Many of these platforms have emerged for online volunteers to provide support to both the disaster affected population and the emergency responders. However, volunteer crowds are not always present (i.e., the number of volunteers decreases as the aftermath of the disaster continues) or are tasked elsewhere. As such, some crisis crowdsourcing efforts have instead chosen used paid workers (Morrow et al., 2011). For this purpose, crowdsourcing marketplaces exist, such as Amazon Mechanical Turk, that enable those requesting work to pay their workers for performing the task at hand.

When hiring workers from these marketplaces, or when using crowdsourcing platforms with a low barrier to entry (as is more often the case with crisis crowdsourcing), the quality and expertise of a worker cannot often be guaranteed, without greater recruitment costs in both money and time. When dealing with unreliable, or even malicious, workers, crowdsourcing platforms typically aggregate the input of many crowd workers to create a reliable output. Previous crowdsourcing work has suggested, as a rule of thumb, 30% of responses are poor or noisy, and suggests using the aggregate of three or more workers to achieve acceptable results (Bernstein et al., 2015). Although, similar (or better) accuracy can be achieved with fewer workers, and thus less cost, through the use of machine learning techniques (Tran-Thanh et al., 2014).

The application of crowdsourcing in disaster scenarios is becoming more prevalent in recent years, due in part to its notable success stories of furthering the efforts of rescue teams around the globe (Harvard Humanitarian Initiative, 2010; Meier, 2013). In what follows, we detail existing examples of crowdsourcing techniques that have been used in humanitarian settings, how they relate to our research efforts, as well as the fundamental human challenges involved.

2.2.1 Online Humanitarian Volunteers

A disaster scene can be a chaotic environment and both the local affected population and the first responders often need situational awareness to act effectively. Given that access to technology has grown, the affected communities are able to seek information online, provide first-hand knowledge to responders and give support to their community. For example, it has been shown that affected populations seek local knowledge or help from their community (Shklovski et al., 2008), they can text (Morrow et al., 2011) or tweet (Starbird and Palen, 2011) their needs to first responders, or they can reach out for emotional support on social networks (Palen and Vieweg, 2008).

More people having access to communication technology means there are more messages to interpret, and this can quickly overwhelm humanitarian groups. Crowdsourcing has been used to reduce humanitarian organisation's workload (Meier, 2014c). Crowdsourcing breaks the workload into microtasks that are then outsourced to a crowd of online workers, who quickly and easily work on a single microtask, vastly parallelising the effort (Law and Ahn, 2011).

For example, on the 12th of January, 2010, a catastrophic magnitude 7 earthquake struck Haiti, with death tolls estimated at over 200,000 deaths, and an even greater number of residential buildings destroyed. Several days after the earthquake, the Ushahidi Project[1] set up an SMS number that enabled anyone in Haiti to communicate with first responders by texting their location and urgent needs, allowing responders to coordinate, and suggest available resources. However, most of these texts were in the local dialect of Haitian Creole, which the majority of first responders could not interpret. In response, the Ushahidi project utilised the crowdsourcing platform, CrowdFlower[2], to perform the translation of these texts, and by the end of the search and rescue phase, some 10,000 text messages had been translated (Morrow et al., 2011). The volunteer efforts demonstrated by Ushahidi were a critical moment for international humanitarian aid, in which the world was introduced to the idea of crowdsourced humanitarian aid (Meier, 2013). The Ushahidi project is a good example of how unskilled workers (Requirement I) can still solve very difficult and large problems.

SMS messages are not the only form of communication with first responders, and efforts have been made to process online data feeds, such as Twitter[3], where users exchange messages called tweets, to understand the events in real-time (Sakaki et al., 2010). Victims of a disaster may attempt to tweet their needs, but often their tweets may be lost amongst the volume of other tweets. Online humanitarian volunteers on Twitter, or 'Voluntweeters', have taken to interpreting actionable messages sent from the affected community and amplifying (i.e., retweeting) them (Starbird and Palen, 2011), or even fowarding the message directly to those who can help (Starbird, 2013). Furthermore, given the sheer number of messages sent during large scale disasters, previous crowdsourcing efforts have struggled to keep up. Systems, such as AIDR (Imran et al., 2014), have been developed that apply machine learning algorithms to automatically classify tweets and thus reduce the crowdsourced workload.

(Palen et al., 2010) envision a future of disaster management where members of the public use technology to receive highly localised and timely information regarding the emergency. Supporting the public and enhancing their ability to make better decisions can improve local resilience during a disaster (Shklovski et al., 2008). However, as crowdsourcing during a crisis becomes more prevalent, practitioners are stressing the importance of standards, frameworks, and protocols for future crowdsourcing efforts during disaster management (Laituri and Kodrich, 2008; Liu, 2014; Palen et al., 2010). This work has identified a number of research questions that need addressing:

- A need to examine who participates, what motivates them, and the ethical issues involved in participating in online humanitarianism.

[1] `www.ushahidi.com`
[2] `www.crowdflower.com`
[3] `www.twitter.com`

- A need to consider privacy for both the online humanitarian volunteers, and the victims. Reporting may put workers at risk, particularly during political crises. Likewise, there may be restrictions about collecting and storing personally identifiable information of the victims (Ausbrook, 2015).
- Consideration for how malicious users could negatively affect the crowdsourced results, particularly during terrorism.
- A need to investigate certain crowdsourcing configurations. For example, although most crisis crowdsourcing tends to be volunteer based or unpaid, it is worth considering the effect of monetary or other reward based systems.

In response to some of these questions, past work has investigated the activities of volunteer users of Twitter (Starbird and Palen, 2011), forums (Goggins et al., 2012) or social media sites (Palen and Vieweg, 2008), how the communities on these platforms form, and what motivates them. In previous disasters, volunteers have been observed as self-organising (Starbird and Palen, 2011; Palen and Vieweg, 2008), often banding together temporarily to help out, and then later disband. Some instead join organisations such as the SBTF or Humanity Road, while others prefer to remain unaffiliated. When asked why they volunteer, some mention personal connections to the affected. However, most are intrinsically motivated, simply believing it is the 'right thing to do'. For example, the voluntweeters stated that it *"doesn't matter if I actually helped directly"* or *"I'll never know if my tweets actually helped but that's okay"* (Starbird and Palen, 2011), expressing similar motivations to the volunteers who take part in our study described in Chapter 6.

However, when dealing with traumatic events, voluntweeters comment on the emotional difficulties involved in rescue attempts that arrived too late, or calls for help that went unanswered. *"it was horrid to hear all these cries for help; it haunts me to this day and I still have nightmares about it"*. Some even go as far as quitting because it is *"emotionally draining"* (Starbird and Palen, 2011). As a potential solution, during traumatic events, volunteers and the affected have been known to provide emotional support for each other on social networks (Palen and Vieweg, 2008). Furthermore, the social media volunteers share a large concern for the privacy, and respect for the family or friends, of the affected (Palen and Vieweg, 2008), mirroring the questions raised above. The concerns raised by these volunteers express similar concerns to the volunteers who took part in our study (Section 6.3.3), emphasising the need for greater support for the online volunteers.

Despite this previous work, little is known about the volunteers who tag and classify imagery. For example, how these volunteers would appropriate a real-time system of tagging live imagery, how they would wish to interact with live systems and their concerns about doing so, and what are the ethical implications of using such systems?

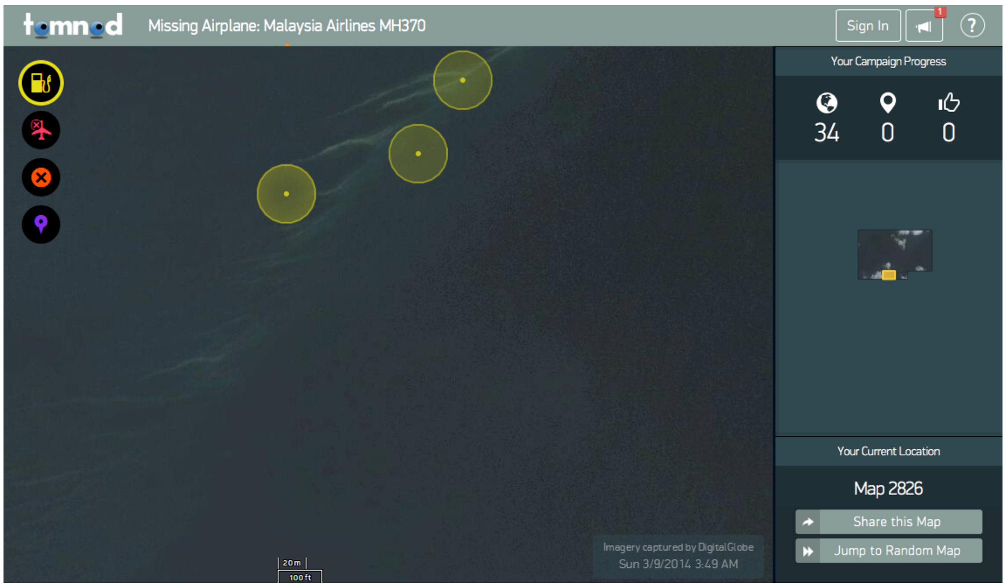


Figure 2.1: An example of Tomnod's interface, showing a user who believes they have found an oil spill.

Next we describe crowdsourcing efforts targeted at information gathering through the analysis of imagery, specifically we look at crowdsourcing platforms that show imagery to their crowd during disasters, often for the purpose of annotating maps that can be used to better inform first responders.

2.2.2 Imagery Assessment

Crowdsourcing in disaster scenarios is often used to analyse the masses of imagery (e.g., aerial imagery) generated by a disaster. Crowdsourcing is used for this task because human intelligence outperforms computers at image recognition and understanding. The crowd's analysis often focuses on the task of updating and annotating maps for improved situational awareness (Ziemke, 2012; Laituri and Kodrich, 2008). Tomnod (Lin et al., 2013) and MicroMappers (Meier, 2014c) achieve this by analysing and tagging imagery. For example, Tomnod acquires satellite imagery of the target area. Users then visually inspect the top-down images, searching for any signs of life, damaged buildings, or debris. Once they believe they had spotted a noteworthy feature, they would tag that feature and classify it as being one of a number of categories relevant to the search task at hand (see Figure 2.1). MicroMappers takes a similar approach, refining digital maps from satellite imagery, and identifying damage from aerial imagery (Meier, 2014c). They recently extended the platform to include UAV imagery (Meier, 2014d), which provides oblique (i.e., angled at 45 degrees) imagery (as opposed to top-down) which can be used for a more accurate classification of damage (Kakaes et al., 2015).

As an example of these systems working, on July 25, 2012, two climbers were reported lost in the Peruvian Andes. Subsequently a mountain rescue team launched a search and rescue mission on the mountainside. In addition to this, friends, family and acquaintances, were recruited to help out, despite living thousands of miles away, through the use of Tomnods' platform. Within a few hours, all of the satellite imagery had been viewed by multiple members of the online crowd, and a large number of possible clues or signs of life had been tagged, such as footprints, or campsites. In more detail, if a number of crowd workers all tag the same location, that tag is said to have reached a consensus and would later be validated by expert mountaineers, satisfying Requirement V. Subsequently, the locations of the most promising tags were then sent to the search and rescue team on the ground. The search ended the following morning when the rescue team found the bodies of the two climbers, who had fallen to their death, just below a track of footprints identified by the crowd (Lin et al., 2013), showing that Requirements I and V are viable, but it lacked the real-time nature of Requirement III. Had a search and rescue team been able to perform the crowd analysis during the search, perhaps the rescue team could have found the fallen climbers sooner, and thus increased their chance of survival (Marconi et al., 2013).

In a similar vein, following the disappearance of the Malaysia Airlines flight MH370, Tomnod was again utilised to help find the wreckage. On March 8, 2014 the airplane ceased all communication with the outside world and the transponder signal was lost. Subsequently the plane was presumed crashed in the ocean, and so began the largest search and rescue operation in history. A huge international effort was carried out both on the ground and online to search for the plane. Disasters such as this gain a lot of media attention all across the world. As a consequence of this media attention, a large user base of well meaning individuals are interested in providing help. Given the rise of internet connectivity, it is now possible for these people to provide valuable information at little cost to themselves, and from the comfort of their own home. Tomnod harnessed the power of online crowds, releasing satellite photos of large areas of the southern Indian Ocean, in which the crowd tagged interesting features, of what they thought was floating debris, or wreckage. However, Tomnod received so much attention, that the website was unable to handle the load and became unresponsive, thus highlighting a need for scalability (Requirement VI).

In addition to the online search for the Malaysia Airlines flight, manned flights took place over the ocean. During the flight a small crew would manually look out the windows, or look at the display of various sensors, in an attempt to spot floating debris in the ocean scattered from the impact. It is an incredibly difficult task to identify debris from a fast moving plane for a number of reasons. Firstly, waves often obscure the line of sight to the debris. Secondly, with the limited number of people aboard the flight, they could not be observing all angles at once, the plane could easily fly past debris without anyone onboard noticing. And thirdly, observing the ocean is a very monotonous task, thus human performance quickly drops. Consequently, this approach did not find anything of note. However, we believe this task would benefit from using a crowd-robotics system. Costs could be reduced by streaming on-board camera footage to the online crowd, satisfying both Requirements I and III.

Outside of the disaster response setting, many crowdsourcing platforms for annotating large datasets of images have been developed by the research community. These platforms are often used to label large computer vision datasets (Deng et al., 2009). For example, LabelMe (Russell et al., 2008) is an online tool to annotate and label objects within images for use in computer vision research. Similarly, the annotation of video requires comparable techniques to that of still images. However, performing the same procedures as used in image analysis on a per frame basis can be time-consuming and wasteful. Videos have some consistency between frames and this can be intelligently exploited to interpolate annotations between frames (Vondrick et al., 2013). For example, LabelMe (Yuen et al., 2009), and VATIC (Vondrick et al., 2013) are both systems that crowdsource video annotation. These approaches have workers localising objects in a video by drawing a rectangle around the target. This bounding box annotates the location of the search target in the current frame, and workers then scroll forward through the video, and update the rectangles location in future frames. The workers are essentially key framing the rectangles position and size throughout the video, and thereby annotating the entire video with the search targets location for each frame. These approaches work when the video is pre-recorded, and their interfaces are designed as such. However, these platforms are not suitable for real-time live annotation, where we require rapid input from workers. As we show in Chapter 5 we achieve this live video annotation using real-time crowdsourcing. Next we describe existing real-time crowdsourcing frameworks.

2.3 Real-Time Crowdsourcing Frameworks

A typical crowdsourcing workflow divides a computationally challenging problem into many microtasks (i.e., a small and easily accomplished task for a human), often referred to as Human Intelligence Tasks, or HITs. This leverages parallelization and human intelligence to solve a whole host of challenging problems (e.g., image labelling, audio transcription, handwriting and text recognition). However, these microtasks have no strict time frame for completion. Thus, this approach is not feasible for short-lived or real-time processes (e.g., a crowd of workers controlling a robot) in which we have strict time constraints on the input from workers.

To address these shortcomings, the retainer model (Bernstein et al., 2011, 2012) was introduced that enables a workflow for continuous real-time crowdsourcing. Real-time crowdsourcing is used for tasks that require input in real-time, and as such requires multiple (due to the need to aggregate crowd input) simultaneously connected crowd workers. This approach enables a crowd to provide input within a narrow timeframe (of a few seconds) needed for interactive tasks. Furthermore, in these workflows, workers are typically engaged for longer periods of time, allowing them to receive feedback from the system as the task evolves due to the workers' collective input. This can result in workers learning the nature of the system they are influencing, and even improving over time. Real-time crowdsourcing has been used for many broad applications, from quickly polling human opinion (Bernstein et al., 2011), to helping blind users navigate through a real-world environment (Lasecki et al., 2013b).

The retainer model is a method of pre-hiring crowd workers before they are needed and asking them to return to the task when notified. Depending on the retainer duration and the small bonus incentive for returning quickly, 50% of the hired workers would return within two seconds, and 75% within three seconds. Typically the fastest crowd-powered interfaces used to be limited by a median response time of nearly a minute (Bigham et al., 2010). However, with the use of a retainer agreement, it is possible to have access to a large real-time crowd within two seconds (Bernstein et al., 2011, 2012). Unfortunately, the downside to such a system is the recruitment cost. A real-time task, due to requiring more redundant workers, and the cost of the retainer, can often be more expensive than typical crowdsourcing tasks. In typical crowdsourcing tasks, past work has shown reliable and accurate recruitment can be achieved in budget limited settings (Tran-Thanh et al., 2014, 2015), but this has not yet been demonstrated in a real-time setting where the recruitment constraints are different.

At the time of this research, there are two existing platforms, LegionTools (Gordon et al., 2015) and TurkServer (Mao et al., 2012a), that enable the recruitment of simultaneously connected, real-time crowds. Both these platforms use a retainer-like system, monitoring the connected workers and ensuring that tasks are routed to the crowd workers when available. However, they take different approaches when it comes to recruiting the crowd workers. A typical rapid recruitment strategy involves creating many short lived HITs, ensuring the task is more prominent on the marketplace's job listings, and allowing the HITs to reach more workers and help recruit crowds more quickly. It is often difficult to predict in advance how many workers will accept the HITs in the desired recruitment timeframe, under recruitment involves waiting for longer and crowd workers may drop out or become annoyed, while over recruitment requires paying for workers unnecessarily. LegionTools' recruitment strategy is to automatically submit HITS on AMT until a desired number has been created. Whereas, TurkServer requires an end user to manually create the HITs on the crowdsourcing marketplaces, enabling finer control of the number of HITs created and therefore the number of people recruited and over how much may be paid to workers, but requires far more end user engagement. Both of these platforms lack the ability to monitor the crowd in such a way to enable more reactive recruitment strategies, to gather statistics about crowd engagement to improve upon the recruitment strategies, or for automated payment based on user input received during the task. To address these shortcomings, a new recruitment tool was created in Chapter 3.

Now, there are a number of applications that already utilize a crowd in real-time, and these can be categorised into two groups. First, there are **crowd control** applications, in which the crowd-powered application has been given full autonomy to interact with and affect the end-user and the environment (e.g., taking a photo, controlling a user-interface, moving a robot). Second, there are **crowd sensing** applications, in which the application provides information sourced from a real-time crowd, that can inform and influence an end-user's decisions.

2.3.1 Crowd Control

Bernstein et al. were the first to propose a real-time crowdsourcing framework. They developed a system called Adrenaline, a crowd powered camera (Bernstein et al., 2011), that attempts to solve the problem of capturing a photo at the right time. It does this by recording a video and asking a real-time crowd to select the best frame of the video. The users would scan through a short video, and when at least a third of the crowd workers are focusing on the same quarter of video footage, the whole crowd's search space narrows to that same quarter of footage. This rapid refinement process repeats itself a number of times, until a single frame has been chosen. Thus, Adrenaline decides on which frame is best, and sends this to the end-user.

Closer to our work, some applications have been used to share the control of robotic hardware (Schulz et al., 2000; Goldberg et al., 2000) or existing software interfaces (Lasecki et al., 2011) to online real-time crowds. Minerva, a telepresence robot, was allowed to roam the atrium of a museum, and through the use of various online interfaces, users at home could view the camera feed and instruct the robot to visit various exhibits (Schulz et al., 2000). This method of control required the environment and the location of the exhibits to be known in advance. In a similar vein, a system called Ouija was developed to allow online users to guide a Ouija planchette around a board, naively averaging all users' input to direct the planchette (Goldberg et al., 2000). However, under certain conditions, this simplistic method of aggregation can result in output that directly contradicts the desires of its users. For example if the crowd's opinion is divided and half wish to go left, and half wish to go right, then the linear average is a vector in-between these two, forward, resulting in an outcome that no member of the crowd desired (Lyon and Pacuit, 2013). Hence, neither Minerva or Ouija, are suitable for effective real-time control in unknown environments. Likewise, there has been work on using live streaming platforms, such as Periscope or Facebook Live, to stream events or museum tours to an online crowd (Hamilton et al., 2016). However, such an approach requires a human actor in charge of the stream to interact with the crowd, and support the communication and participation of the crowd to influence the stream.

Moreover, Lasecki et al. developed the Legion system in which any user interface can be shared by a real-time synchronous crowd (Lasecki et al., 2011). The crowd may be asked to perform a wide range of tasks, such as word processing, data entry, or the remote control driving of a small toy car. Legion demonstrates a number of methods for aggregating, in real-time, noisy or erroneous input from the crowd and the effectiveness of these strategies is shown to be dependent on the type of interface shared. However, Legion aims to be general purpose for all types of interface, and does not leverage specialised aggregation methods that can be more domain focused (i.e., for robotic control). Furthermore, the paper does not investigate the characteristics and performance of these aggregation methods in varying conditions (e.g., the robot performing a variety of different tasks)

The application described above combine the timely input from crowd workers to make a decisive action. In contrast, some crowd-powered applications instead aggregate crowd input to provide a real-time stream of information, which an end-user may choose to act upon.

2.3.2 Crowd Sensing

There are many examples of crowdsourcing applications that provide additional information in order to support the user. For example, Waze[4] provides real-time traffic and road information, sourced from Waze's community of drivers, such that an end-user can decide on the best route to take. Likewise, Twitter's members provide real-time information as events unfold (See Section 2.2.1) and can be used to help decision makers (Starbird and Palen, 2011). Furthermore, these applications are examples of real-time crowdsourcing that do not need to post HITs on crowdsourcing marketplaces via a recruitment tool in order to achieve frequent real-time crowd worker input. Instead, they rely on having a large enough pool of active crowd members.

Other applications broadcast live surveillance footage to online crowds that detect illegal immigration (Tewksbury, 2012), or shoplifting (Trottier, 2014). In these applications, conscientious citizens log onto a website to continuously watch a video feed, and when they observe illegal activity they inform an expert (e.g., border patrol, or shop security) who may then decide to take action in order to prevent the proposed illegal activity. Notably, this surveillance approach encouraged a high number of false positives, low participation, and a large number of workers abandoning the task after only a short while due to boredom (Dunphy et al., 2015).

Likewise, live imagery has previously been broadcast from a UAV in a mountain rescue attempt (Egglestone et al., 2013). This work used an alert model similar to those above, which asks the crowd to continuously watch the feed and simply press a button to

[4]`http://www.waze.com`

alert an expert when they observe the search target. This resulted in a large crowd response when the search target was on screen. However, while this approach may work in a simple search and rescue scenario, with only a single search target, this approach would struggle to provide useful and discernible information in more complex search scenarios. For example, in a reconnaissance scenario where the objective is to find multiple targets. It is possible, in this scenario, that multiple search targets are present on screen simultaneously, and therefore it is not enough to have a single alert button. We must also know the location of the targets on screen to differentiate between the multiple targets.

We require more detailed input from crowd workers than a simple alert. To this end, a number of applications have used real-time crowds as a human-intelligent sensor, providing a stream of data about a video feed. These are typically applications for which computer vision algorithms are not yet sophisticated enough to classify automatically, or where a training data set is too small to accurately train the algorithms, whereas a crowd has the human intelligence to understand the task intuitively. Similarly, despite the recent advances in computer vision and deep learning, these approaches still require large relevant training datasets, and because disasters are dynamic and each deployment is likely very different, with different search requirements, and in different environments, the relevant training data does not yet exist.

Chorus (Lasecki et al., 2013b) is one such example of real-time crowdsourced human-intelligent sensor. Chorus recruits workers to observe live video captured from a cell-phone and to converse with the person behind it, answering any general question about wildly different subjects and settings. Likewise, the Legion (Lasecki et al., 2011) tool observes a live feed from a robot and utilise crowd input to a shared control interface, to influence the actions of the robot.

Closer to our work, a number of applications have used real-time crowds as a human-intelligent sensor, providing a stream of data about a video feed. LegionAR (Lasecki et al., 2013a) uses the crowd to classify activities performed by actors in a live video stream, and Glance (Lasecki et al., 2014) codes the timings of behavioural cues. Likewise, Zensors (Laput et al., 2015) uses a real-time crowd as a human sensor. Users of Zensors can set up a camera feed, and ask a question about a property within that feed (e.g., how many cars are in the parking lot?, is food available?). Images from the camera are streamed to crowd workers who observe the image and provide a classification in real-time. Furthermore, Zensors also uses machine learning to detect when a video stream has changed significantly and needs new crowd judgement, and attempts to learn features in the image that correlate with the crowd judgements to eventually remove the need for hiring a crowd altogether. However, the Zensors interface only provides a number of datatypes (e.g., Yes/No, Number, Scale) with which workers can classify images. Thus, their approach may work on simple discrete datatypes. However, given that the location of a search target is not a single discrete position, but rather a continuous area, changing in size, shape and location over time, their approach does not fit our needs.

All of the applications discussed in this section aggregate a crowd's input to reduce noise, and improve accuracy. In the next section we review a number methods that can be used to do this in real-time settings.

2.4 Aggregating Opinions from the Crowd

Deciding on the worker's interface used to elicit judgements from a crowd will later affect which methods of input aggregation is applicable, and how accurate or useful it will be. Thus it is important to consider the following:

- **Elicitation Method:** how the crowd is asked to perform tasks, for example by showing a video feed and asking users to use their keyboard, or by showing still images and asking users a series of yes/no questions.
- **Input Type:** the data type of the input elicited from the crowd, i.e., whether it is discrete button presses, xy-positions, or a textual response.

The real-time nature of control limits the types of input we can elicit. It is impractical to rapidly elicit textual input, such as those used by product reviews, or SMS translations. Therefore, we limit the scope of our discussion to deal only with simple and quick inputs, such as key presses and mouse clicks. Simple inputs are used by other real-time crowdsourcing applications, such as Zensors (Laput et al., 2015), and Legion (Lasecki et al., 2011) where rapid input is a priority.

Similarly, the real-time constraints limit the methods of aggregating crowd judgements that are feasible. Methods that require deliberation or communication between crowd workers may be impractical to perform with the strict time constraints implied by crowd robotics. For example, deliberation groups or the Delphi method, in which crowd workers independently suggest actions and later revise their plans after hearing the other's judgements, are impractical for real-time input. Similar types of aggregation techniques have been used by Chorus (Lasecki et al., 2013c), or TurkServer (Mao et al., 2016), but are not suitable for crowd robotics. Thus, we limit our scope to mathematical aggregation and voting techniques.

Within these constraints, we consider a number of well-known aggregation methods for both **discrete input** and **continuous input**.

2.4.1 Discrete Input

Eliciting discrete valued input in real-time is more practical with keyboard button presses. For example, to give votes on low-level directional commands. For this case, plurality (also commonly known as majority) voting is the most common form of aggregating discrete input (List, 2012). The plurality method works by choosing the input that the majority agrees with, for example in a binary decision, the choice that has $> 50\%$ of agreement.

The Condorcet jury theorem proved that if the people voting have more than a 50% chance of choosing the best or optimal input, and make uninfluenced, independent, judgements, then the plurality method will converge to selecting the optimal input, as the number of voters increase (Condorcet, 1785). However, this only holds true when there is a binary decision. For robotic control problems, the input set may be greater than a binary decision. Furthermore, the voters will not be uninfluenced, as they are made aware of the aggregated outcomes, and thus this feedback will influence their next vote.

Moreover, social choice theory shows that we can calculate fairer results if we elicit crowd workers' rankings of each candidate in order of preference, rather than eliciting a vote of only their first preference. This is called a preferential ballot (Saari, 1995). There are various voting methodologies that use these ranked votes to calculate a winner that minimises the distance from all voters' rankings. Thus choosing a winner that might not have been any voters' first choice, but a winner that is broadly acceptable to most voters. These methods typically attempt to calculate the Condorcet winner. The Condorcet winner is the candidate who would win a pairwise majority election against every other candidate. One such method of approximating the Condorcet winner is the Kemeny-Young voting method, a paradigm from social choice theory, and it has been shown to have good performance when applied to the domain of aggregating the opinions of the crowd in human computation problems (Mao et al., 2012b). Pairwise comparisons of each voter's rankings is used to calculate the Condorcet winner, and a metric is used to calculate the distance between pairwise rankings, this is known as the Kendall-Tau metric. To find the winner of the election, all possible rankings are considered. The ranking with the minimum summed distance from each submitted rank wins the election. While this method more accurately represents the crowd's opinions, it is computationally expensive to perform pairwise comparisons. Instead, the Borda count method is well used approximation, and importantly it is fast to compute. The Borda count method gives each candidate, for each ballot, a number of points corresponding to its rank. Once these points have been summed, the candidate with the most points is elected the winner. In the context of crowd robotics, these voting methodologies could be used for low-level control, voting on the discrete actions (e.g., move forward, left, right) that the robotic agent could take.

2.4.2 Continuous Input

Eliciting continuous valued input in real-time is more practical with mouse clicks. For example, clicking on the location of targets in an image or video stream.

The simplest aggregation method would be to take the average of all inputs, this is known as the unweighted linear averaging, Equation 2.1. As discussed previously, this was the method applied by Ouija (Goldberg et al., 2000). It is the method famously used by statistician Francis Galton, in the classic example of the wisdom of the crowd effect. Galton averaged all the guesses of an ox's weight at a county fair competition, and the result was closer than the estimates of most crowd members (Galton, 1907).

$$\bar{J} = \frac{1}{N}\sum_{i=1}^{N} j_i \tag{2.1}$$

Where N is the number of members in the crowd and j_i is the judgement given by crowd member, i.

Additionally, this method is considered a good default option, when little is known about the members' reliability or the distribution of the members' inputs (Armstrong, 2001; Graefe et al., 2014). This method performs best if the judgements are distributed around a central value, e.g., the true weight of an ox (as in (Galton, 1907)).

The weighted linear average, Equation 2.2, builds upon the previous averaging method, and instead weighs certain member's suggestions higher than others. This is only feasible when it is known that some of the crowd members are more reliable (or more skilled) than others. Moreover, this method has already been shown to work well for combining expert opinion (Cooke, 1991; Clemen, 2008). A member's weight, w_i, can be calculated based on past information about how well they are performing, or how much they agree with the rest of the crowd.

$$\bar{J}_w = \frac{1}{N}\sum_{i=1}^{N} w_i j_i \tag{2.2}$$

Furthermore, this method could be improved by assigning a crowd members' weighting based on their own self reported confidence. There are studies in the field of psychology that suggest confidence correlates with accuracy (Koriat, 2012). Likewise, a member's weight could be assigned based on peer respect, i.e., how much other crowd members respect their opinion (Lehrer and Wagner, 1981). All members of the crowd would get to assign a weight to who they believe are the experts or novices, but each member would have a different opinion of who the experts are. However, eliciting the members' confidence in themselves, or in others, while also eliciting their usual input in real-time is infeasible, as requesting that much input from a member in such a short time is too taxing to maintain real-time responses.

These methods discussed so far are only viable if the crowd judgements centre around a single true value, and this is not always the case for some real-time crowdsourcing tasks. For example, a crowd robotics task of clicking on the location of search targets in an image. A search target may be observed over a range of pixels in an image, and so a single location value would not accurately represent the target, or multiple targets may exist and so the crowd judgements do not centre around a single true value.

Instead, clustering algorithms can be used to group crowd judgements. A cluster may represent an area classified by crowd judgements, and multiple clusters may exist in the same image. There are a number of algorithms that can be used for clustering. For example, K-Means (MacQueen et al., 1967) is a general purpose clustering algorithm, that works well when we have prior knowledge of how many clusters exist, and when we expect the clusters to be even sizes, and with convex geometry. However, we may not have prior knowledge of how many clusters (i.e., search targets) we expect to find in an image, and the shape of the search targets are not guaranteed to be convex. Instead, DBScan (Ester et al., 1996) is a clustering algorithm that requires less prior information, by determining which points (i.e., crowd judgements) are within a radius of each other, and that there exists a minimum number of points connected, the algorithm can automatically determining the number of clusters and allow for arbitrarily shaped clusters. The radius is a parameter that can be tuned depending on the expected size of search targets in the imagery, while the minimum number of points can be tuned to how many unique crowd judgements we need before we trust the aggregated judgement, Section 2.2.

2.5 Summary

Existing approaches to search and rescue have an inefficient delay between the search phase, and the assessment of the very large data sets gathered. We discussed how the assessment of UAV imagery could be improved by breaking the task into many HITs (microtasks) and crowdsourcing their evaluation. We also discussed how this could be further improved this with the addition of real-time crowdsourcing, such that data analysis can be done in-flight, allowing for real-time changes to the search path.

We evaluated existing real-time crowd applications (Gordon et al., 2015; Mao et al., 2012a), and how these solutions use a retainer model and methods of recruitment to gather a large crowd quickly. However, existing approaches related to crowd robotics do not meet our requirements. For example, systems that use the crowd for low-level control of a robot, Minerva (Schulz et al., 2000) and Ouija (Goldberg et al., 2000) were too simple and could not work in unknown environments. The Legion (Lasecki et al.,

2011) system improved upon this, enabling crowd control of a toy car. However, Legion's aggregation schemes were designed for the control of any general interface. Likewise, systems that use the crowd as a human-intelligent sensor, such as Zensors (Laput et al., 2015), currently provide simplistic datatypes, and cannot provide information that would be beneficial to the humanitarian efforts (e.g., real-time location of search targets in a video feed). In contrast, our work focuses on the humanitarian setting, and can leverage this for more powerful strategies that better cope with the real-time nature of human-intelligent sensing and crowd robotics.

We then surveyed various input aggregation techniques in the context of crowd robotics. We discussed methods for both discrete and continuous input. Discrete input, which is useful for low-level robotic control, would best suit voting methodologies, such as a Majority or Borda voting, to decide on the most preferred discrete action. While for continuous input, used when eliciting positional data about multiple search targets, we showed that a clustering algorithm, such as DBScan, would work best.

The next chapter describes a tool used for recruiting real-time crowd workers from crowdsourcing marketplaces, that extends upon the work of LegionTools and Turkserver discussed in Section 2.3. This tool is used for all real-time crowdsourcing experiments undertaken in this work.

Chapter 3

Real-time Recruitment

In order to harness the power of real-time crowds, we need a method of recruiting them, redirecting crowd workers to the task once enough have been recruited, maintaining the recruited crowd size, and finally paying the crowd when the task is done. Maintaining a crowd of a suitable size is important to ensure that enough input is received from the crowd to overcome challenges such as slow rate of input from individual workers, and filtering of unreliable input through calculating a consensus with other simultaneously connected workers. This chapter describes the process by which we recruit a real-time crowd, enabling Requirements I, III, and VI, as detailed in Section 1.1, and therefore underlies all the following work in this thesis.

Current crowdsourcing marketplaces[1] and platforms[2] are not designed with real-time crowdsourcing in mind, as discussed in Section 2.2. Crowdsourcing marketplace enable online workers to browse through large numbers of HITs (i.e., small microtasks that workers can complete for payment). Workers can preview the task, see how much they would be paid for the job, and choose whether to accept the work. Typically the HITs do not have strict time constraints, and they may be posted on the marketplace for days. Workers may come and go as they please, accepting the HIT at any time, and take as long as they need to complete the task, and with no constraint on multiple workers performing the task simultaneously.

However, given that real-time crowdsourcing requires a number simultaneously connected workers, and that the tasks they perform are usually short lived, real-time crowdsourcing recruitment platforms need to work around the constraints of these crowdsourcing marketplaces, Section 2.3. Real-time crowdsourcing platforms achieve this using a number of recruitment strategies (see Section 2.3 and 3.2). Typically, real-time crowdsourcing platforms such as Legiontools (Gordon et al., 2015) or Turkserver (Mao et al.,

[1]Such as Amazon's Mechanical Turk, or Crowdflower.
[2]Such as Tomnod, or Micromappers.

2012a), create many short lived HITs, ensuring the task is more prominent on the marketplace's job listings, and allowing the HITs to reach more workers and help recruit crowds more quickly. Then, by sending the recruited workers to an external website where their presence can be counted, they wait for the task to begin. This is known as the retainer model (Bernstein et al., 2011). Once a desired crowd size has been recruited, all workers will be notified that the task is ready, and will be redirected to it, ensuring simultaneous workers from the very start of the task.

At the time of this research, existing systems (Gordon et al., 2015; Mao et al., 2012a), are able to post HITs, redirect crowd workers, and manually manage their payment afterwards. However, these systems require significant manual observation and control during the entire recruitment process. Thus, towards making real-time crowdsourcing more automated, a goal that benefits crowd robotics, we identify the following requirements:

- A Recruitment Agent, to monitoring the crowd for more reactive recruitment strategies. This enables quicker recruitment and more consistent crowd sizes to participate in tasks, with less manual oversight.
- Gather statistics about crowd engagement to improve upon the recruitment strategies.
- Task agnostic automated payment, such that the task communicates to the recruitment tool how much to pay workers, as payment may be based on the specific rules of the task, or the crowd workers' input received during the task.

In what follows, we describe the development of a recruitment manager tool that improves upon existing systems and addresses the requirements identified above. This tool is used throughout this research in order to provide a reliable and consistent source of real-time crowd workers. We describe the tool in detail, the challenges it solves, and how it is used. We then go on to describe the concept of a recruitment agent, a software agent responsible for strategically communicating with the crowdsourcing marketplace in order to rapidly and reliably recruit crowd workers.

3.1 Recruitment Manager

The recruitment manager is a tool designed to help task requesters (i.e., the user who requests crowd work, they authorise the posting of HITs, and the payment of the crowd workers) recruit a real-time crowd. The tool was created using a Django webserver, allowing it to be accessible online, to be controlled and monitored from any device, but requires a user login in order to control the management interface. The tool can be hosted on either the same server, or a separate host, to the machine hosting the real-time crowdsourcing task. Enabling the tool to be used by external parties without the need to install and run their own version of the real-time crowdsourcing platform. The manager should assist the users, the requesters, in performing the following steps:

- **Step 1: Create Task Templates:** A template of the task that the requester wishes the workers to perform. It stores the parameters required to create a task on the crowdsourcing marketplace, the parameters used by the recruitment agent, and the URLs defined by the task used for communication between the recruitment tool and the task.
- **Step 2: Create Sessions:** Start an instance of a task, in which workers will be recruited, and perform the task at hand.
- **Step 3: Recruit workers for the session using an automated recruitment agent:** Ensure we have a pool of simultaneously connected crowd workers.
- **Step 4: Monitor the session:** Allow requesters to monitor how many workers are presently tasking, among other metrics, in order to make strategic recruitment decisions.
- **Step 5: Pay the workers:** Manage the payment of workers, accept or rejecting their work as appropriate.
- **Step 6: Help resolve any issues that might arise:** Occasionally, problems with tasks may arise (e.g., bugs, worker complaints), and the manager should make resolving these issues easier (e.g., providing access to work logs, and enabling easy payment of workers).

Next, we describe how the tool assists the requesters in performing these steps.

3.1.1 Step 1: Create Task Templates

The requesters often desire to run a number of different tasks, each of which may differ greatly in how it is performed, and so the tool should be agnostic to the type of task. For example, the work that we discuss later in this thesis (Chapter 4, and Chapter 5) detail two distinct tasks, both are functionally different, with different requirements, and with different methods of payment for the crowd workers. Therefore, the tool provides an interface for requesters to manage the workers and their recruitment, and simply routes workers to an external task, a 3rd party website developed by the requester.

Real-time crowdsourcing tasks often need to be run multiple times, especially in a research environment to gather statistically significant results. We enable requesters to be able to create a 'task template' that describes the task. The manager uses this template

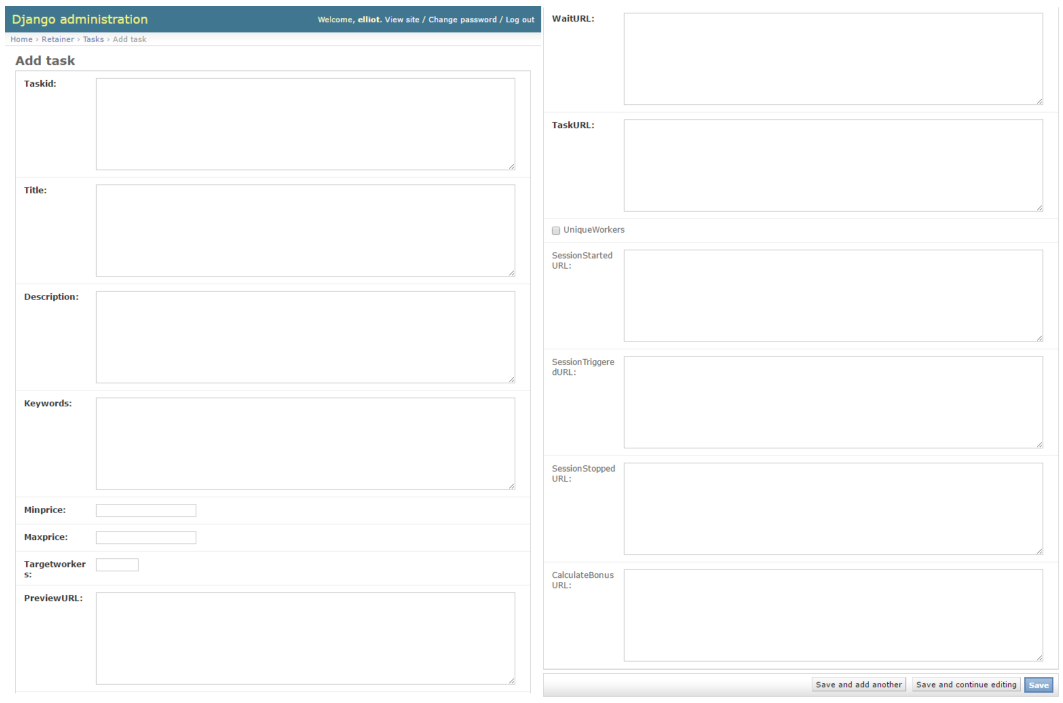


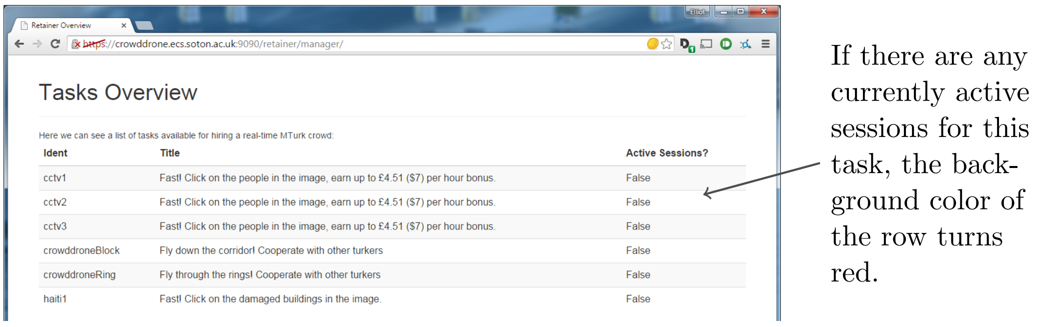
Figure 3.1: The admin interface for creating tasks

when creating HITs to be posted on the marketplace, and define the bounds in which a recruitment agent can act in order to recruit the workers (e.g., how much they can pay, or how many workers to recruit). Using the admin interface, requesters can create these task templates, see Figure 3.1. The template defines the following properties:

- The title used to advertise the task to workers on the marketplace.
- A short description displayed to workers about the task they will be performing.
- Keywords so that workers can more easily search the marketplace for the tasks.
- The minimum and maximum price that the task requester is willing to pay for workers to accept the HIT
- The target number of workers desired to be performing the real-time task.
- Whether the task requires unique workers, especially important for research where we wish to remove the training effect repeat tasks may have.
- A number of URLs that specify where the workers will be redirected to when they are previewing, waiting, or performing the task.

3.1.2 Step 2: Create Sessions

A Session, as defined here, is an instance of a task. During the session, multiple HITs are created in order to gather simultaneously connected workers, who then perform the

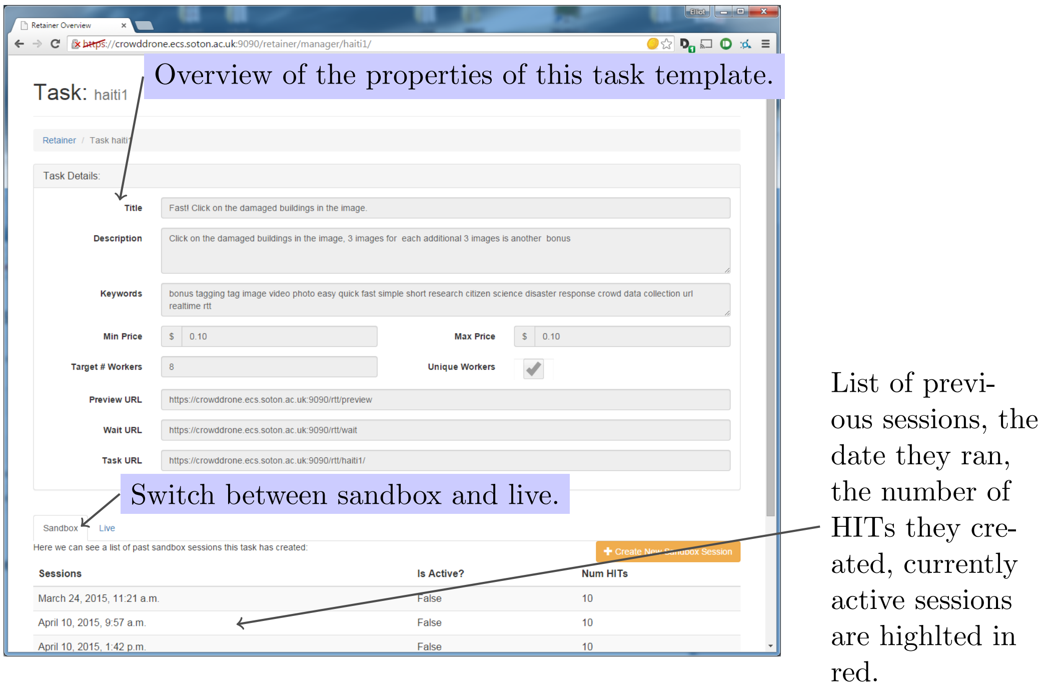

Retainer Overview
Elliot
~~https~~://crowddrone.ecs.soton.ac.uk:9090/retainer/manager/
If there are any currently active sessions for this task, the background color of the row turns red.

Tasks Overview

Here we can see a list of tasks available for hiring a real-time MTurk crowd:

Ident	Title	Active Sessions?
cctv1	Fast! Click on the people in the image, earn up to £4.51 ($7) per hour bonus.	False
cctv2	Fast! Click on the people in the image, earn up to £4.51 ($7) per hour bonus.	False
cctv3	Fast! Click on the people in the image, earn up to £4.51 ($7) per hour bonus.	False
crowddroneBlock	Fly down the corridor! Cooperate with other turkers	False
crowddroneRing	Fly through the rings! Cooperate with other turkers	False
haiti1	Fast! Click on the damaged buildings in the image.	False

Figure 3.2: The list of task templates.


Retainer Overview
Elliot
~~https~~://crowddrone.ecs.soton.ac.uk:9090/retainer/manager/haiti1/
Overview of the properties of this task template.
Switch between sandbox and live.
List of previous sessions, the date they ran, the number of HITs they created, currently active sessions are highlted in red.
Create New Sandbox Session

Task: haiti1

Retainer / Task haiti1

Task Details:

Title	Fast! Click on the damaged buildings in the image.		
Description	Click on the damaged buildings in the image, 3 images for each additional 3 images is another bonus		
Keywords	bonus tagging tag image video photo easy quick fast simple short research citizen science disaster response crowd data collection url realtime rtt		
Min Price	$ 0.10	Max Price	$ 0.10
Target # Workers	8	Unique Workers	✓
Preview URL	https://crowddrone.ecs.soton.ac.uk:9090/rtt/preview		
Wait URL	https://crowddrone.ecs.soton.ac.uk:9090/rtt/wait		
Task URL	https://crowddrone.ecs.soton.ac.uk:9090/rtt/haiti1/		

Sandbox Live

Here we can see a list of past sandbox sessions this task has created:

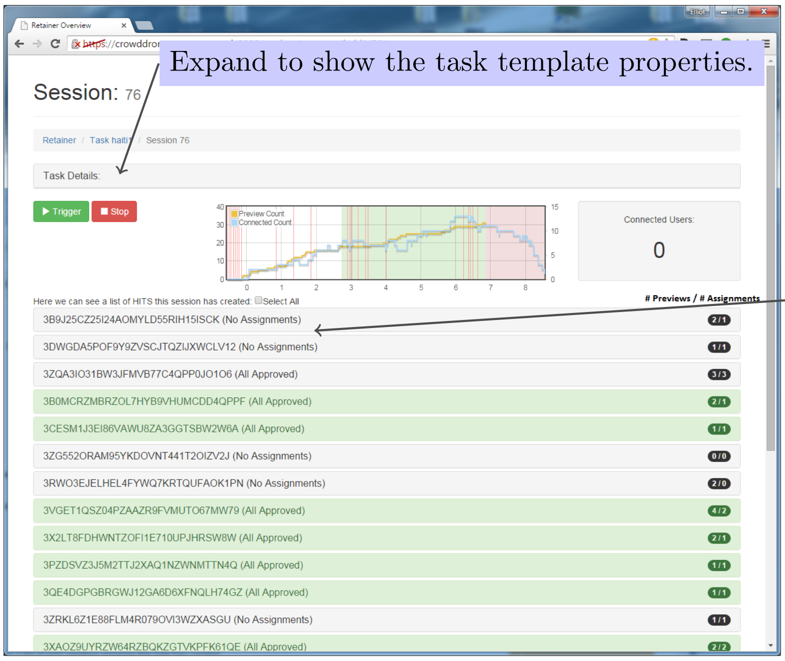
Sessions	Is Active?	Num HITs
March 24, 2015, 11:21 a.m.	False	10
April 10, 2015, 9:57 a.m.	False	10
April 10, 2015, 1:42 p.m.	False	10

Figure 3.3: The details for a given task template.

real-time task at hand. For example, in the domain of crowd robotics (Chapter 4 and Chapter 5), the Task template describes the crowd robotic problem we're solving (e.g., control, or sensing), and a Session would be a single deployment of the drone, streaming the live footage to the crowd. Multiple deployments would involve creating multiple Sessions.

The main landing page for the recruitment manager displays a list of the available task templates (see Figure 3.2). This page provides an overview of the templates registered in the recruitment manager, and colours the tasks in red if they currently have an active recruitment session, to alert the requester to investigate and monitor the ongoing session. From here requesters can select the task type they had previously created in Figure 3.1, and proceed to the task details page.

The task template details page (see Figure 3.3), displays the template properties created



Here we can see a list of HITS this session has created: Select All

	# Previews / # Assignments
3B9J25CZ25I24AOMYLD55RIH15ISCK (No Assignments)	2 / 1
3DWGDA5POF9Y9ZVSCJTQZIJXWCLV12 (No Assignments)	1 / 1
3ZQA3IO31BW3JFMVB77C4QPP0JO1O6 (All Approved)	3 / 3
3B0MCRZMBRZOL7HYB9VHUMCDD4QPPF (All Approved)	2 / 1
3CESM1J3EI86VAWU8ZA3GGTSBW2W6A (All Approved)	1 / 1
3ZG552ORAM95YKDOVNT441T2OIZV2J (No Assignments)	0 / 0
3RWO3EJELHEL4FYWQ7KRTQUFAOK1PN (No Assignments)	2 / 0
3VGET1QSZ04PZAAZR9FVMUTO67MW79 (All Approved)	4 / 2
3X2LT8FDHWNTZOFI1E710UPJHRSW8W (All Approved)	2 / 1
3PZDSVZ3J5M2TTJ2XAQ1NZWNMTTN4Q (All Approved)	1 / 1
3QE4DGPGBRGWJ12GA6D6XFNQLH74GZ (All Approved)	1 / 1
3ZRKL6Z1E88FLM4R079OVI3WZXASGU (No Assignments)	1 / 1
3XAOZ9UYRZW64RZBQKZGTVKPFK61QE (All Approved)	2 / 2

List of created HITs. They are colored green, when all assignments have been approved or rejected. Yellow, when workers are performing the HIT, or awaiting payment. Gray, if no assignments.

Figure 3.4: The details for a given session.

in the admin interface (see Figure 3.1), and provides access to previous and ongoing sessions, and enables requesters to create a new recruitment session. Many crowdsourcing marketplaces provide a sandbox marketplace enabling developers to test their HITs before posting them to the live marketplace. The recruitment manager provides access to both the sandbox and live marketplaces. After choosing the marketplace (by default the sandbox is chosen), the requester is shown a list of previous and ongoing recruitment sessions, and, as before, colours an ongoing session in red. The requester can choose to either view a past or ongoing session, or to create a new session, and is sent to the session details page.

3.1.3 Step 3: Recruit Workers

Once a new session is created, the manager begins recruiting crowd workers. This is achieved through the use of a recruitment agent, discussed in more detail in Section 3.2. Briefly, the agent posts HITs, job adverts, to a crowdsourcing marketplace and waits for workers to accept the HIT.

3.1.4 Step 4: Monitor a Session

The requester must monitor an ongoing recruitment session because they are in charge of deciding when to send workers to the task, and when to stop the recruitment of more workers. Knowing when best to perform these actions is difficult and task dependent, and could be, for example, influenced by how many workers the requester believes the task needs, how long left they believe the task will be running for, and how many workers are currently performing the task and how likely those workers are to disconnect in the remaining time. We describe a possible extension to this tool in Section 3.2 that details how this could be automated.

The session details page (see Figure 3.4), allows requesters to observe and control the recruitment process. In a single session, a recruitment agent may create many HITs on a crowdsourcing marketplace, recruiting crowd workers to perform in this session of our task. This page displays a reactive list[3] of the HITs that a recruitment agent has posted.

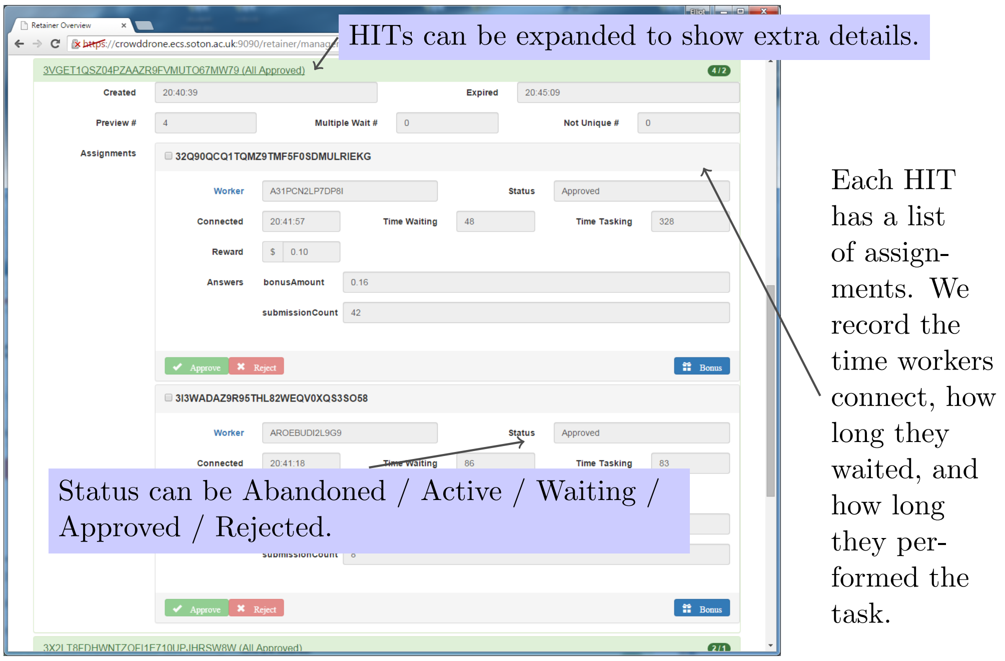
The requester can observe the number of recruited workers in the waiting room, and those performing the task. Then, once the required number of workers has been recruited, the requester can trigger the platform to redirect the workers, sending them to the task. As new workers are recruited, they too are held in the waiting room, and can be triggered to join the tasking workers if the requester desires. Monitoring the crowd size and ensuring a constant number of workers is a cumbersome task and would normally be left to the recruitment agent (we elaborate on this in Section 3.2). When the task is nearing the end (e.g., the UAV flight is almost over), the requester would need to manually stop the recruitment process. This ensures that workers do not join just before the end of the task. This is beneficial because workers may get frustrated by joining a task when there is no work left to be done, and the requester does not have to pay workers for the unnecessary waste of their time. In order to automatically stop the recruitment, the task needs the capability to communicate with the recruitment agent (as discussed in Section 3.2).

Furthermore, on this page a graph is shown that displays when these jobs were posted, the number of connected workers, and how many times our jobs have been previewed by other workers. Here we can see how the number of worker's vary over time, and how it varies in relation to HIT postings.

In addition, further down the page, the requester can observe a dynamic list of the HITs posted by the recruitment agent. Each of these HITs can be expanded to show more details, Figure 3.5. Initially, a newly created HIT will only display the time at which it was created, the time in which it will expire, and three metrics:

- The number of times workers have previewed the HIT.
- The number of times a worker has attempted to accept the HIT but been denied as they were already performing the HIT in a separate browser tab. Workers cannot work on two real-time tasks at the same time, as their focus must be on a single task.
- The number of times a worker has attempted to accept the HIT but been denied as they had already performed in a similar session in the past

[3]the list automatically updates as the agent posts more HITs



HITs can be expanded to show extra details.

Retainer Overview

https://crowddrone.ecs.soton.ac.uk:9090/retainer/manag

3VGET1QSZ04PZAAZR9FVMUTO67MW79 (All Approved) 4/2

Created	20:40:39			Expired	20:45:09
Preview #	4	Multiple Wait #	0	Not Unique #	0

Assignments

32Q90QCQ1TQMZ9TMF5F0SDMULRIEKG

Worker	A31PCN2LP7DP8I			Status	Approved
Connected	20:41:57	Time Waiting	48	Time Tasking	328
Reward	$ 0.10				
Answers	bonusAmount	0.16			
	submissionCount	42			

3I3WADAZ9R95THL82WEQV0XQS3SO58

Worker	AROEBUDI2L9G9			Status	Approved
Connected	20:41:18	Time Waiting	86	Time Tasking	83

3X2LT8FDHWNTZOEI1F710UPJHRSW8W (All Approved) 2/1

Status can be Abandoned / Active / Waiting / Approved / Rejected.

Each HIT has a list of assignments. We record the time workers connect, how long they waited, and how long they performed the task.

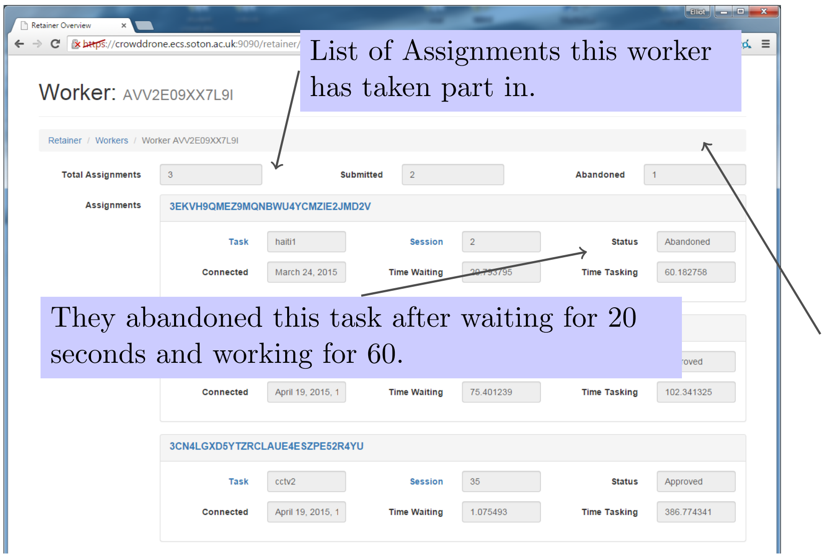
Figure 3.5: The details for a posted HIT in a session.

These three counts begin at zero, but will dynamically update as the session goes on. These metrics give the requester an understanding about how prominent their HITs are on the marketplace (i.e., preview count), and how crowd workers are interacting with them (i.e., accepting the HIT or being denied to perform the HIT). This understanding helps requesters fine tune the task template parameters, such as task description or pay.

As workers viewing the preview of the posted HITs choose to accept the HIT, they are assigned to the HIT, and an "Assignment" is created and dynamically added to the HIT details. The assignment details the worker who performed the task, their current status in performing the task (Abandoned, Active, Waiting Approval, Approved, Rejected), when they connected to the session, how long they were in the waiting room before the task began, and how long they spent performing the task. We show the base payment they will be paid for accepting the HIT. Upon a worker submitting the task, their status will change to Waiting Approval, and the "Answers" box can be filled with custom data from the task the requester has created. The example shown in Figure 3.5, the "Answers" box contains the custom data that details the amount the worker should be bonused, and how many images (submissionCount) the worker annotated, see Chapter 5.

3.1.5 Step 5: Pay Workers

In this step, the requester can observe the data they have received and choose to Accept or Reject the workers' submissions. The requester can then choose to give a bonus to the worker, as is typical with real-time crowdsourced tasks, as the base payment is typically


Retainer Overview

https://crowddrone.ecs.soton.ac.uk:9090/retainer/

List of Assignments this worker has taken part in.

Worker: AVV2E09XX7L9I

Retainer / Workers / Worker AVV2E09XX7L9I

Total Assignments	3	Submitted	2	Abandoned	1

Assignments

3EKVH9QMEZ9MQNBWU4YCMZIE2JMD2V

Task	haiti1	Session	2	Status	Abandoned
Connected	March 24, 2015	Time Waiting	20.753795	Time Tasking	60.182758

They abandoned this task after waiting for 20 seconds and working for 60.

					roved
Connected	April 19, 2015, 1	Time Waiting	75.401239	Time Tasking	102.341325

3CN4LGXD5YTZRCLAUE4ESZPE52R4YU

Task	cctv2	Session	35	Status	Approved
Connected	April 19, 2015, 1	Time Waiting	1.075493	Time Tasking	386.774341

CCTV task they accepted 2 tasks in same session. the session says unique workers only, but would not restrict the same worker from performing in the same session twice, as the footage is still different when they join later

Figure 3.6: The details page of a crowd worker, we can see which tasks they have performed and when.

only an incentive to join the waiting room. The full payment is acquired through performing the real-time task, and can only only be calculated later and paid through the bonus. The bonus is automatically calculated, by using a URL specified when creating the task type, so that different task types can implement their own bonusing strategy.

3.1.6 Step 6: Resolve Worker Issues

It is important to remember that this tool is recruiting real people, who are working to get paid. By working on the requester's real-time tasks, they can not be earning money for working on other requesters' tasks at the same time. Thus, if something goes wrong during the requester's Session, the crowd workers may have lost out on the ability to work on other tasks during this time and thus have lost potential earnings. It is important to maintain good worker relations, for example, on Amazon Mechanical Turk workers can use 3rd party websites to rate and review the requesters, which may affect the requester's ability to recruit more workers in the future. With that in mind, the recruitment manager is designed to keep worker frustration to a minimum, to be able to quickly and easily investigate and resolve any worker relation issues that may arise (e.g., due to bugs in the task, disagreement over payment, or disagreement about rejected work).

Thus, by clicking on the worker ID in the HIT details, requesters can observe the track record of individual workers (see Figure 3.6). Requesters can observe what tasks they have performed in the past, and this can help search for the relevant data to resolve the issues.

3.2 The Recruitment Agent

A recruitment agent is the software responsible for interacting with the crowdsourcing marketplace. An agent must be capable of communicating with a crowdsourcing marketplace, and implement an effective recruitment strategy to rapidly hire workers.

The agent in this work is responsible for recruiting from Amazon Mechanical Turk, and this section will discuss the challenges involved in rapidly recruiting workers from this marketplace. We foresee different agents could be developed for any of the various crowdsourcing marketplaces. Many of the marketplaces differ in their target audience of crowd workers (i.e., expert crowdsourcing marketplaces are vastly different to traditional crowdsourcing), and thus may require different strategies of recruitment. For example, in Section 6.2, we will require volunteer workers, and the recruitment strategy we will use will be different to the strategies used for recruiting on AMT.

The agent we developed for recruiting from Amazon Mechanical Turk uses a number of heuristics in order to ensure rapid recruitment. When the recruitment process is started, the agent immediately performs its control loop until manually told to stop recruitment. The control loop first begins by monitoring the number of presently connected workers (At the start of a session, this will be zero). If the number of workers is less than the target crowd size, the agent will begin posting HITs to AMT every 10 seconds. Newly created HITs appear first on AMT's list of available HITs, and so repeatedly creating HITs maximises the visibility of our HITs to the workers.

The agent then monitors these HITs and expires any that are older than 3 minutes. These HITs are not likely to be near the top of the newly created HITs list, and thus less likely to be seen by workers. This clean up step is vital to ensure that the agent is not monitoring hundreds of HITs (as the AMT API is rate limited and slow to respond, reducing the number of HITs to monitor reduces the communication required between AMT). Once the target number of connected workers has been reached, the agent stops posting HITs to AMT, although more workers could still continue to be recruited through the already posted HITs. These remaining HITs then expire after 3 minutes, and no more recruitment will occur until the number of connected workers drop below the target. This heuristic works well at initially recruiting slightly more than the target number of workers, ensuring rapid replacement of workers should one leave early. The target number of workers can then be redirected to the task when it begins, and any remaining workers can be kept in the retainer pool and redirected if and when a tasking worker drops out.

A simple recruitment agent such as this can suffer from a number of drawbacks. This approach is slow to react when the number of workers falls below the target threshold, as there is some significant delay between posting HITs to the marketplace, and crowd workers accepting the HIT and connecting to the task. Recruiting enough workers to fulfil the target crowd size can be challenging. The number of available workers on Amazon Mechanical Turk can vary throughout the day, the requester may limit the task to unique workers (i.e., those who haven't performed the task before), or if workers deem the pay as too low, then finding available and willing workers to fulfil the target crowd size can cause a significant slowdown in the rate of recruitment.

To compound further on this challenge, the slower the rate of recruitment, the longer currently recruited workers have to wait before the task can begin. The longer these workers wait, the more likely they are to drop out, causing the remaining workers to have to wait even longer. This churn behaviour can quickly reinforce itself, and no automated methods currently exist to address this issue. Moreover, keeping workers waiting for a long time can make them unhappy and more likely to review the requester poorly, making it harder to recruit in future. Manual observation and intervention can help alleviate the issue before it arises, requiring the requester to notice the recruitment slowdown. The requester could then terminate the trials and try again at a later time, or increase the base pay to encourage more workers to accept the HITs.

Although the recruitment agent used in this work is functional, and performs well for the tasks we ask of the crowd in this thesis, we can envision a more powerful agent that addresses some of this simplistic agent's drawbacks. Future work should investigate how to improve the rate of recruitment, and how to reduce the amount of manual overhead required. For example, a sophisticated agent would be able to:

- Vary HIT price, and the rate of job posting to appropriately respond to the increase in demand for workers, or the shortage of supply. The supply of workers could be influenced by external factors, such as the time of day. While the demand for workers could be influenced by the number of presently connected workers, or the dropout rate of the connected workers.
- Learn the relationship between the number of workers previewing the HIT and the number of workers that accept the HIT. By observing the number of workers currently previewing HITs, it could give an idea of how many more HITs it should post to the marketplace in order to recruit a desired number of workers.
- Learn how long workers typically stay on a task before leaving, or the behaviour patterns of workers before they leave (e.g., workers may begin to perform slower

before leaving), an agent could pre-emptively recruit before the workers leave, thereby ensuring a more consistent crowd size.

- Provide a mechanism for tasks to communicate with the recruitment agent. The manager is task agnostic, and as such the task is an external process. The task needs the capability to communicate with the agent in order to let the agent know when to begin or terminate the recruitment process, and to vary the target crowd size dynamically.

As it currently stands, in all realtime crowdsourcing platforms, human involvement is required to monitor the task and manually start and stop recruitment. An agent that addresses the above challenges would remove the need for human involvement in the recruiting process. Real-time crowdsourcing could become an automated process, and enable more sophisticated workflows, or enable a greater number of repeat experiments. For example, in the context of real-time crowdsourcing tasks during live UAV flights, the recruitment agent would automatically be alerted to begin recruiting while the UAV is being prepared for take-off, and the moment it takes flight the crowd workers are redirected to the task. As the task continues, some workers may disconnect, the recruitment agent would learn when best to post jobs, and how much to pay the workers, in order to replace the leaving workers and always ensure that enough workers are active so that the task of influencing the UAVs flight can be performed reliably. When the task is nearing completion, or the UAV lands, the recruitment agent would be alerted that it no longer needs to perform it's recruitment duties.

3.3 Summary

We have described the implementation of a tool that enables the recruitment and management of real-time crowds. We introduced the concept of a recruitment agent, and detailed our initial implementation of this agent. We then outline the requirements of a more powerful recruitment agent that could be used to automate the task of real-time recruitment. In the following chapters we describe the use of this tool when applied to the context of crowd robotics.

Chapter 4

Real-time Crowdsourcing for Control

Humanitarian relief organisations are using UAVs to transport small care packages (D'Andrea, 2014), and perform aerial surveillance and assess hazardous situations (Murphy and Cycon, 1999). Most of these applications currently require a skilled pilot, and cannot be automated. For example, drone couriers delivering aid would require advanced natural language processing to navigate to a given description of the drop target, or aerial surveillance would require accurate computer vision to be able to identify the areas of interest and vision comprehension to be able to understand hazards.

Real-time crowdsourcing for the control of robotic agents can be used to help automate these tasks by outsourcing the human perception and understanding required by these tasks. Accordingly, we need to understand how best to aggregate real-time crowd input, (see Section 2.4), to effectively control a robot. Furthermore, the choice of aggregation method may differ depending on the task at hand. Thus, in this chapter we first describe CrowdDrone, Section 4.1, a platform that enables us to evaluate these real-time opinion aggregation methods in the space of controlling a robotic agent, a UAV. Then we go onto design methods for the aggregation of real-time crowd input in a robotic control setting, Section 4.2. Finally, we evaluate these aggregation methods in different scenarios Section 4.3, in order to understand which aggregators perform best and when to use them.

4.1 CrowdDrone

A crowd robotics system for real-time control must be capable of gathering and maintaining a real-time crowd, recruited online using a crowdsourcing platform, Chapter 3. Furthermore, a crowd robotics system requires a robot that can maintain a constant

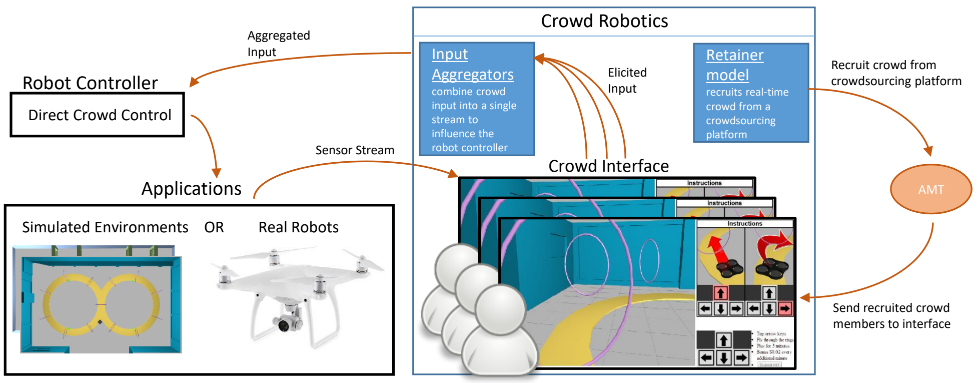

Crowd Robotics
Aggregated Input
Robot Controller
Direct Crowd Control
Input Aggregators
combine crowd input into a single stream to influence the robot controller
Elicited Input
Retainer model
recruits real-time crowd from a crowdsourcing platform
Recruit crowd from crowdsourcing platform
AMT
Sensor Stream
Applications
Crowd Interface
Simulated Environments OR Real Robots
Send recruited crowd members to interface
Instructions

Figure 4.1: The CrowdDrone System Diagram

connection to the internet, such that it can stream sensor output to the crowd, then act upon their feedback. Given that a crowd robotics system is open to anyone online (Requirement I), the experience, skill, and reliability of crowd members may vary greatly. For this reason crowd robotics requires opinion aggregation methods that can filter out this noise (Requirement V). Furthermore a crowd robotics system must also be scalable, such that as crowd members dynamically join and leave, the system should be capable of aggregating their input in a timely manner such that it can be used to influence the path of the robot (Requirement III).

To address these requirements, we developed the open-source platform, CrowdDrone, available for download online.[1] CrowdDrone is powered by standard web technologies (i.e., HTML, JavaScript, WebSockets) and a well-known robotics library, ROS[2], that enables seamless portability from real robots to simulated robots. CrowdDrone can use simulated environments to enable researchers to quickly develop dynamic environments, that would otherwise be cost prohibitive to build in reality. We use an established simulation environment, Gazebo.[3] It is developed and maintained by the Open Source Robotics Foundation, and it realistically simulates robots and environments to help avoid the common problems of robotics, such as short battery life and dangerous behaviours towards the robot or human operators.

CrowdDrone applies the crowd robotic principles mentioned above to the control of a UAV, specifically a quadcopter. A multi-rotor UAV was chosen, as it is more commonly used by emergency responders and drone delivery systems. In this work, the robot is simulated in a physically realistic environment, as it enables fast creation of interesting test scenarios (see Section 4.3.1), but the system could just as easily be applied to a real robot.

CrowdDrone uses the recruitment management tool, developed in Chapter 3, to hire and maintain a real-time crowd from AMT. Once a crowd is recruited, an alert can be sent

[1]`https://github.com/ElliotSalisbury/CrowdDrone`
[2]`http://www.ros.org`
[3]`http://www.gazebosim.org`

to all participants simultaneously when the task is ready. The crowd then observes the imagery from a camera on board the robot, and votes on the direction that they believe would be the best to move the robot towards (see Figure 4.1) in order to reach their goal. These votes are combined through various real-time input aggregation methods (discussed in Section 4.2.1).

Within crowd robotics systems, most members of the crowd are non-experts, and so we must ensure that the control scheme is simple and accessible (i.e., it is safe to assume all AMT users have keyboards, whereas a control pad is not a sensible control scheme). For this reason, we use a reduced set of actions for a UAV. Even though a UAV is capable of four or more degrees of freedom, we constrain the input set to influence only two degrees, moving forward or backward, and yawing left or right. As such, the control scheme can be reduced to simply the four arrow keys on a standard keyboard. The arrows on the keys intuitively represent the directions that the crowd workers intend the robot to move towards. However, with an intuitive control scheme or crowd worker training, the elicitation and aggregation of real-time input for more than two degrees of freedom would be possible.

In what follows we first model the decision problem to be solved in the CrowdDrone platform. We then go on to describe the novel voting methods for real-time aggregation of crowd member inputs.

4.2 Input Aggregation Model

As in Section 4.1, CrowdDrone consists of three main components. A robotic agent, a real-time crowd, and an input aggregation algorithm, this section details the later. In more detail, the agent, a drone, has a set of actions that it may perform, Θ. As previously mentioned, the control method is simplified, such that the agent is capable of taking only four directional actions with which to move, forward ($\uparrow$), backward ($\downarrow$), yawing left ($\leftarrow$), and yawing right ($\rightarrow$).

$$\Theta = \{\uparrow, \downarrow, \leftarrow, \rightarrow\} \tag{4.1}$$

The actions of the drone may be dictated by votes generated by a crowd. In more detail, we assume the crowd is comprised of N members, $W = \{1, 2, \cdots, N\}$. Each crowd member, u, is capable of voting for actions they wish the agent to perform, $v_u^t \in \Theta$, where, $t \in \mathbb{R}_{\geq 0}$, is the timestamp of when this vote is received, and $t = 0$ is the start of the experiment. This voting process allows crowd members to express their opinion about which direction they wish the agent to go, by interacting with the given interface, such as pressing the $\uparrow$ key on their keyboard to suggest that they intend the agent to move forward.

Let $V_u = \left\{v_u^{t_1}, v_u^{t_2}, \cdots, v_u^{t_k}\right\}$ denote the set of all actions performed by the member u, and V is the union of these over all members of the crowd, that is:

$$V = \bigcup_{u \in U} V_u \tag{4.2}$$

We then define an input aggregation function that combines the votes of the members, into a single stream of output actions that the agent will perform.

$$f : U \times V \times \mathbb{R}_{\geq 0} \longrightarrow \Theta \tag{4.3}$$

The function f takes as parameters, U the set of all members; V the set of all votes; t the current time; and outputs an action $\theta \in \Theta$.

If all the inputs of the crowd were known in advance, function f could be computed offline with traditional crowdsourcing methodologies. However, due to the real-time nature of our applications, f has to be computed online, with real-time crowdsourcing. This means that the output θ at any time t can only be computed based on all inputs received prior to time t (rather than all those beyond t in the offline case). Hence, in the next section we explore the algorithms that implement f in real-time.

4.2.1 Input Aggregators

Aggregating votes from a real-time source presents a number of key challenges:

- **Computational Complexity:** as the crowd size grows large, so too will the rate of input received per second. The input aggregation algorithms must remain responsive even at large scale.
- **Accuracy:** due to unreliable or malicious crowd members, we need to be able to filter out their input, and aggregate input beneficial to the task at hand. To accurately represent the crowd, we want to perform the actions that move the robot closer to where the majority of the crowd wishes to go.

The following methods have been chosen such that they can be applied in real time, in unknown environments without a ground truth. Thus, in what follows, we describe the voting strategies used for the input aggregation evaluated as part of the CrowdDrone system. We then go on to compare the various aggregation methods (Section 4.2.2), discussing their relative strengths and weaknesses.

4.2.1.1 Mob

The *Mob* aggregator as implemented in Legion (Lasecki et al., 2011), is the simplest and most naive approach to combining the input of the crowd. Whenever a member in the crowd submits a vote, the agent will immediately perform the desired action, i.e.

$$f_{mob}(U, V, t) = \arg\max_{v_u^x \in V} x \tag{4.4}$$

The idea behind this method is that the majority of votes will be those most agreed upon by the crowd and the agent will make progress towards the crowd's desired direction. Given that some votes may not agree with the majority this method would result in 'thrashing', where the agent performs an action and shortly after, performs the opposite action. Furthermore, this method does not consider the members' reliability or whether the rest of the crowd agrees with the vote. Hence, this method is susceptible to noise introduced by members who repeatedly vote for undesirable actions.

4.2.1.2 Leader

This input aggregator was originally described in Legion (Lasecki et al., 2011). The aggregator hosts elections, selecting the crowd member who has agreed most often with the other crowd members' votes and then allows them full control of the search agent for the duration of the next election. It was designed to address a number of issues. First, the need for quick responses to external events happening in real-time systems. Secondly, in situations of uncertainty, for example a situation where two equally viable paths are presented. In this situation the *Mob* and *Majority Vote* (see Section 4.2.1.3) methods would struggle when the crowd's intentions are divided, but the *Leader* aggregator has only one person in control and thus no divided consensus.

The *Leader* aggregator hosts elections, $e \in \mathbb{Z}_{\geq 0}$ with a duration $d \in \mathbb{R}_{\geq 0}$, in which a crowd agreement metric is calculated for each crowd member. Agreement is a measure of inter-voter homogeneity, put simply, how often their votes are similar to other crowd members. The member with the highest agreement wins the election to assume direct control of the agent. The crowd agreement metric is calculated for the current election by creating a vector, $\mathbf{c}^{(e)}$, where each element of the vector relates to an element in the set of actions Θ, and the value of that element is the count of the members who voted for that action, as seen below.

$$\mathbf{c}_\theta^{(e)} = |votedFor(e \cdot d, \theta)| \tag{4.5}$$

Then we generate a vector for each member, $\mathbf{v}_u^{(e)}$ in Equation 4.6, again where each element of the vector relates to an action in θ, and the value of that element is the number of votes made by the member, u, for that action over the duration, d, of the election as shown below.

$$\mathbf{1}_\theta(x) := \begin{cases} 1 & \text{if } x = \theta, \\ 0 & \text{if } x \neq \theta. \end{cases}$$

$$\mathbf{v}_{u,\theta}^{(e)} = \sum_{v_u^t \in V_u : (e-1)\cdot d < t \leq e \cdot d} \mathbf{1}_\theta(v_u^t) \tag{4.6}$$

With the two vectors, $\mathbf{c}^{(e)}$ and $\mathbf{v}_u^{(e)}$, now defined, we can calculate an individual member's agreement with the crowd by performing a dot product of the two vectors.

$$VC(\mathbf{v}_u, \mathbf{c}) = \frac{\mathbf{v}_u \cdot \mathbf{c}}{||\mathbf{v}_u|| * ||\mathbf{c}||} \tag{4.7}$$

This metric is then used in a similar manner as before in Equation 4.12. The initial weight of the member is $w_u^{(0)} = 0.5$, the updated weight is then calculated using the VC metric and combined with its historical agreement.

$$w_u^{(e+1)} = \alpha w_u^{(e)} + (1 - \alpha) VC(\mathbf{v}_u^{(e)}, \mathbf{c}^{(e)}) \tag{4.8}$$

This calculation results in the member who has consistently agreed most with the crowd, having the highest weight. Thus, this member is selected to become the leader. Given this, the search agent will only perform the leader's votes during the next election, Equation 4.9.

$$f_{leader}(U, V, t) = latestVote(\arg\max_{u \in U} w_u^{(\lfloor t/d \rfloor)}) \tag{4.9}$$

There are drawbacks to this approach though, as it only uses the wisdom of the crowd effect to select the best crowd member. As a consequence, the control scheme can never achieve greater performance than that of an individual.

4.2.1.3 Real-Time Majority

The *Real-Time Majority* aggregator calculates a winner as follows. In brief, whenever a member in the crowd submits a vote, the *Real-Time Majority* aggregator gathers all the votes over the past d seconds and calculates the action θ that most crowd members voted for, while weighting the influence of those users based on their past agreement with the crowd.

In more detail, the *Real-Time Majority* aggregator first requires a weight for each crowd member u to reduce the influence of unreliable crowd members. This weight is iteratively recalculated every d seconds and is a value proportional to that member's agreement with the crowd. To calculate this weighting, it is first beneficial to define a function (Equation 4.10) that retrieves the latest vote from a member, and a function (Equation 4.11) that returns the set of members whose latest vote was for a given action θ in the past d seconds.

$$latestVote(u) = v_u^{t'} \mid \forall v_u^{t''} \in V : t' \geq t'' \tag{4.10}$$

$$\begin{aligned} votedFor(t, \theta) = \{u \mid \exists v_u^{t'} &= latestVote(u) \; \wedge \\ v_u^{t'} &= \theta \qquad\quad \wedge \\ t - d &< t' \leq t \quad \} \end{aligned} \tag{4.11}$$

With those two functions defined, we can now calculate the weight $w_u^{(i)}$ of a given member u during the recalculation iteration i. The weight is calculated as the percentage of members who all voted for the same action.

$$w_u^{(i+1)} = \alpha w_u^{(i)} + (1 - \alpha) \frac{|votedFor(i \cdot d, latestVote(u))|}{\sum\limits_{\theta \in \Theta} |votedFor(i \cdot d, \theta)|} \tag{4.12}$$

Where $w_u^{(0)} = 0.5$, an initial value chosen to give fair weighting when little is known about the member's reliability. Here, α is a historical decay rate to reduce the influence of previous agreement, it is tuned to the same value as described in (Lasecki et al., 2011), and ensures the weighting represents a recent agreement history. Then, whenever a vote is received, the action chosen by the agent is calculated using the following function:

$$f_{vote}(U, V, t) = \underset{\theta \in \Theta}{\arg\max} \sum_{u \in votedFor(t, \theta)} w_u^{(\lfloor t/d \rfloor)} \tag{4.13}$$

Where $\lfloor t/d \rfloor$ calculates the latest iteration index at current time t. For each action, the weights for the members that voted for it are summed and the action with the highest sum will be the action executed by the agent.

4.2.1.4 Real-Time Borda

The Borda count election method is often used in social choice theory (Mao et al., 2012b). Borda count requires voters to rank candidates in order of preference, giving 1 point to the least preferred candidate, 2 to the second least, and so on. The candidate who receives the most points wins the election. It is considered a fairer method than majority voting because it can sometimes elect broadly acceptable candidates, instead of those preferred by the majority. Hence, in the context of CrowdDrone, the *Real-Time Borda* aggregator can be used to reduce the ability of malicious members, who may have the majority, to influence the election. Instead, because the most-voted for action is not always the winner, more broadly acceptable actions voted for by the well-meaning crowd may win the election. It is worth noting that although no malicious workers were detected during our experimentation, it is still worth considering mitigating the effect they may cause given the potential for disastrous consequences, discussed in Section 6.5.4.

Given that crowd robotics requires real-time input from crowd members, it is not feasible to elicit rankings of actions in order of the crowd member's preference. Instead, from a single vote we can assume a natural ranking. For example, if the member intends the agent to move forward ($\uparrow$), then we can assume that they rank forward ($\uparrow$) as their top choice, and since backward ($\downarrow$) is the opposite action, it would rank as their least desired action. However, little can be inferred, from just a single vote, about the actions in-between these, the actions of yawing left ($\leftarrow$), right ($\rightarrow$), or performing a new element of the action set, no action at all ($\emptyset$), and would therefore all be given the same ranking. This assumed ranking may make sense in the general case, but may not hold true in all scenarios. A more complex approach would be to infer user preference from further sensing data (e.g., if a wall is detected to the left, the right action could have higher preference). Furthermore, we introduce $\emptyset$ to denote an instance of 'no action', so that if there is divided consensus amongst the crowd, the best action may be to remain stationary. This ranking is denoted as $\phi_u \in \Phi^{(t)}$, where $\Phi^{(t)}$ is the set of all members' votes in the past d seconds from the current time t converted into rankings.

Additionally, due to the nature of robotic control, we can refine this method further. As in Section 4.1, CrowdDrone uses a simplified set of actions for input, but this does not mean that the output set must also be reduced. The robotic agent can perform both linear and angular components in a single action at the same, such as moving forward while rotating left. With this in mind, the *Real-Time Borda* aggregator allows us to calculate a member's ranking for both the linear action set and the angular action set. We can then run two elections for both linear and angular motion and combine the two winning components into a single expanded action. The robotic agent will then perform a combined linear and angular action. Until now, the use of this expanded output set has been impractical for all other aggregation methods. The *Leader* aggregator can only forward the reduced inputs of the leader, and a *Real-Time Majority* vote on both linear and angular component actions would ensure that both elections always select an angular and a linear action combined. *Borda* ranked voting enables us to sometimes choose inaction, $\emptyset$, for a single component (e.g., to apply only a forward linear action, but have no angular component).

To denote this increased set of output actions, we introduce a set $\vec{\Theta}$ of vectors comprising of a linear component and an angular component.

$$\vec{\Theta} = \{\uparrow, \emptyset, \downarrow\} \times \{\leftarrow, \emptyset, \rightarrow\}$$

Thus, we need to convert a crowd member's vote θ into two rankings, a linear and angular ranking. We represent this as a vector ϕ_u, which is comprised of a linear ranking and an angular ranking (see Equation 4.14). For example, if a crowd member votes $\uparrow$, the vote is translated to the two rankings $\big\langle\langle 3,2,1\rangle, \langle 1,2,1\rangle\big\rangle$. The first linear ranking shows the member prefers $\uparrow$ to $\downarrow$, and the second angular ranking shows preference for no angular action $\emptyset$, rather than turning.

$$\phi(\theta) = \begin{cases} \big\langle\langle 3,2,1\rangle, \langle 1,2,1\rangle\big\rangle & \text{if } \theta = \uparrow, \\ \big\langle\langle 1,2,3\rangle, \langle 1,2,1\rangle\big\rangle & \text{if } \theta = \downarrow, \\ \big\langle\langle 1,2,1\rangle, \langle 3,2,1\rangle\big\rangle & \text{if } \theta = \leftarrow, \\ \big\langle\langle 1,2,1\rangle, \langle 1,2,3\rangle\big\rangle & \text{if } \theta = \rightarrow \end{cases} \tag{4.14}$$

Real-Time Borda then calculates the sum of the rankings, $\bar{\phi}^{(t)}$, for each possible candidate vector $\vec{\theta} \in \vec{\Theta}$, from every ranked vote as follows:

$$\bar{\phi}^{(t)}_{\vec{\theta}} = \sum_{\phi_u \in \Phi^{(t)}} \phi_{u\vec{\theta}} \tag{4.15}$$

The chosen action performed by the robotic agent is then the candidate with the greatest sum of points, as computed by f_{borda} (see Equation 4.16).

$$f_{borda}(U, V, t) = \arg\max_{\vec{\theta} \in \vec{\Theta}} \bar{\phi}^{(t)}_{\vec{\theta}} \tag{4.16}$$

4.2.2 Comparison

Now that the four methods have been introduced, here we discuss the relative strengths and weaknesses of each aggregation method. In more detail, in our applications, operating in real time and responding to real-time changes is critical. The *Mob* aggregator is likely to remain the most responsive to real-time changes, because it instantly performs the latest vote. Then the *Real-Time Majority* and *Real-Time Borda* both recalculate the best action to perform after every new vote is received, ensuring that their output is always up to date with the crowd's opinion. Finally, the *Leader* aggregator aims to be responsive to real-time events because, with a single user in charge, there is no deliberation required. Provided the leader reacts quickly, so too will the agent. Furthermore in situations of uncertainty, for example a situation where two equally viable paths are presented, the *Mob*, *Real-Time Majority* and *Real-Time Borda* aggregators could struggle when the crowd's intentions are divided, but the *Leader* aggregator has only one person in control and thus no divided consensus.

These aggregators, however, come with a number of shortcomings. For example, the *Mob* aggregator does not consider the crowd members' reliability or whether the rest of the crowd agrees with the vote. Hence, some instantly applied votes may contradict the previous vote, resulting in 'thrashing', where the agent performs an action and shortly after, performs the opposite action. Furthermore, this method is susceptible to attacks where even a single malicious member can undermine the entire system (Stefanovitch et al., 2014). Thus, while *Mob* may perform well in ideal conditions, in applications where the reliability of the crowd is not controlled, *Mob* is unlikely to perform well. Conversely, the *Leader* aggregator chooses only the most agreed-with crowd members, and thus is less susceptible to malicious crowd members. However, the aggregator can only ever select the most average crowd member for direct control. As a consequence, the control scheme can never achieve a greater performance than that of the average individual, who may be too slow to react in real time. On the other hand, the *Real-Time Majority* and *Real-Time Borda* are required to balance real-time reactions with reliability. Enough people must first vote for an action before it happens, whereby there is a delay between the first vote for an action, and it eventually being acted upon.

Next, we describe the method and results of the empirical evaluation of the aggregation methods detailed in this section.

4.3 Empirical Evaluation

The evaluation of the real-time consensus methods, described above, allows us for the first time to determine which methods of aggregation are suitable for crowd robotics in a real-world experiment. Furthermore we aim to determine if some methods are more suited to specific scenarios, such as a dynamically changing environments.

The following experiments uses a real-time simulation environment (Section 4.3.1) capable of realistically simulating the dynamics of the agent in a given scenario, and a simultaneous crowd of voters with an interface for shared control (bottom right of Figure 4.1), in which the crowd can observe the simulated agent and then vote on actions they wish the agent to take. It was deemed necessary for CrowdDrone to use a simulated environment for the practical reason that performing experiments with real-time crowds from AMT is time-consuming, and that it enables us to quickly create dynamic environments. To ensure fair benchmarking, we keep the user interface and the dynamics of the simulated agent the same, only varying the method of input aggregation and scenario.

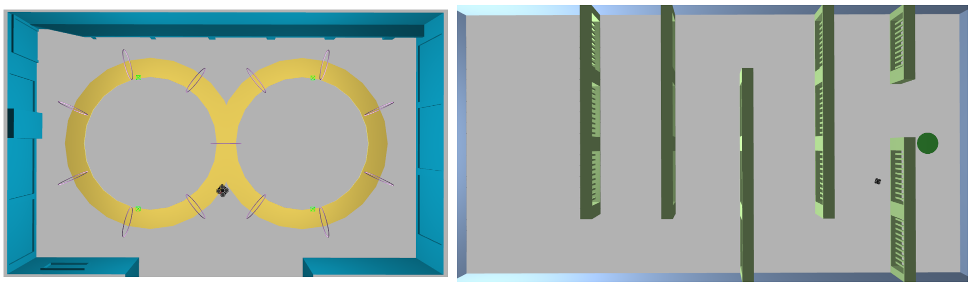

(a) The Figure Of Eight Scenario (b) The Dynamic Blocked Path Scenario

Figure 4.2: The CrowdDrone testbed environments

When hiring the real-time crowd from AMT, the crowd was instructed in how to control the UAV with a pictorial diagram (seen in Figure 4.1). Short bullet points informed them that others would also be controlling the drone and that they should ensure the flight remains sensible. However, the crowd was not directly incentivised to perform the tasks correctly. Instead, they were informed only that they would receive payment provided that they vote, regardless of whether those votes were beneficial.

For these experiments, the constant parameter, d, used by the *Leader*, *Real-Time Majority*, and *Real-Time Borda* aggregators, denotes the election duration. This parameter determines how long the aggregators wait before considering members' votes no longer relevant. If the duration is too short, we would disregard many members' votes and then our calculation of the crowd's intentions would be biased towards the faster crowd members. However, if d is too long, then the aggregation algorithm may consider outdated votes. Outdated input may no longer be relevant if the robotic agent has since moved far from the initial location of the vote. We empirically chose the duration to be 2 seconds long, due to pilot experiments showing that the mean crowd input rate is 40 votes per minute. An election with a duration of 2 seconds ensures that we are capturing most of the members' latest votes when calculating a consensus.

4.3.1 Real-time Environment

As discussed in Section 4.1, CrowdDrone uses a simulated environment as it enables the fast development of dynamic scenarios. The environment used for the evaluation in this paper is bounded by visually distinctive walls that serve two purposes. Firstly, this prevents the drone from leaving the experimentation area. Secondly, it provides the users with an easy means to orientate themselves in the world. Inside this environment we can envision a number of scenarios that provide a challenge for calculating group consensus. For this study, we have considered two scenarios. The first is a simple scenario designed to evaluate the performance of the aggregation algorithms (Section 4.3.1.1). The second is a scenario in which the crowd must respond to changes in the environment (Section 4.3.1.2).

4.3.1.1 The Figure of Eight Scenario

The *Figure of Eight* (*FOE*) scenario has a number of equidistantly spaced rings arranged in a figure of eight pattern. Thirteen rings are placed in total, six around each circular section, and one shared ring in the middle (see Figure 4.2a). The crowd then attempts to fly the drone through the rings. This provides a simple environment in which to test and measure the basic functions of crowd robotics. Given the uniformity of the layout of the rings, the time taken to fly between each ring is a simple metric to compare the aggregator methods against. Additionally, given that in this scenario we know the optimal path beforehand, we can also calculate a positional error from the optimal path to measure how much the aggregators deviate from it.

4.3.1.2 The Dynamic Blocked Path Scenario

The *Dynamic Blocked Path* (*DBP*) scenario presents the crowd with the end goal far away down a long empty corridor. They must then navigate down this corridor. However, at five equally spaced points down the corridor, a wall will appear directly in the path of the drone. This impedes their progress and they must navigate around this new wall, reacting dynamically to the new environment, and attempt to reach the end goal (see Figure 4.2b). The wall consists of four randomly creatable sections. The two sections closest to the drone will always appear, ensuring the crowd must react, and one of the remaining two sections is created randomly, to ensure the crowd cannot learn where the open section may appear. This scenario is designed to test the crowd's ability to react to changes.

4.3.2 Evaluation Metrics

In order to compare the various aggregation algorithms, we define the following metrics relevant to the field of crowd robotics.

1. **Time between rings:** this is a general measure of how well the input aggregation algorithm performs. A longer time taken to travel a fixed distance can mean that there are more contradictory movements, more hesitation, or that a less than optimal path was taken.
2. **Positional Error:** the average distance from the optimal path can show how well the input aggregation method is filtering out noise, and responding to real-time updates.
3. **Stationary time:** the time spent stationary per minute is a measure of when the drone is either rotating or stopped (i.e., action $\emptyset$ is chosen) because of divided consensus as in the case of *Real-Time Borda*.

4. **Inaction time:** the time spent, per minute, not performing any action can be an indicator of low crowd participation.

5. **Input rate:** the rate of users' inputs per minute. Higher input rates correlate with greater crowd engagement, and can also be indicative of disagreement with the agent's actions.

6. **Reaction time:** the time taken between the wall appearing, in the *DBP* scenario, and the drone reacting to the new environment. The drone is considered to have reacted when its heading has changed by at least 20 degrees in an attempt to change course and navigate around the newly spawned obstacle.

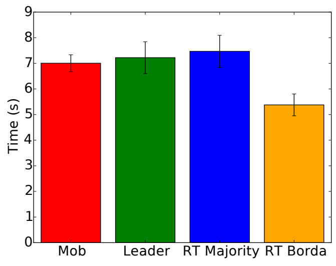
4.3.3 Results

In this section we evaluate the various aggregation methods (Section 4.2.1) as used in the two scenarios (Section 4.3.1). We ran 10 experiments for each aggregation method on each scenario. The real-time nature of these experiments is prohibitive to running large numbers of repeats. The total number of active crowd members (those who voted at least once), varied from 4 to 13 per experiment, and on average, 4.4 votes from unique crowd members were received every d seconds (i.e., 2 seconds). For the *FOE* scenario, we acquired on average 29 minutes of flight time, passing through an average of 254 rings, or over 1km of flight for each aggregation method.

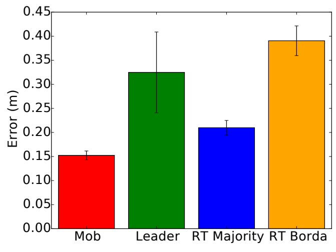
4.3.3.1 Performance

The performance measured in the *FOE* scenario shows that the *Real-Time Borda* aggregator is significantly[4] faster at travelling between the rings, than other aggregators (Figure 4.3a), and 1.8 seconds faster than the *Leader* aggregator, the current state-of-the-art aggregation method for real-time control. However, despite this faster result, the *Real-Time Borda* aggregator shows a similar positional error (39cm) to that of the *Leader* method (Figure 4.3b). Instead, for positional accuracy, the *Real-Time Majority* aggregator is significantly closer to the optimal path (10 centimeters) than the *Leader* aggregator. The *Mob* aggregator has even greater positional accuracy (15cm). This is because it instantly applies the crowd input, thus if it navigates off-course, a crowd member will quickly correct it. However, this is only true if there are no malicious crowd members. *Mob* has no method for reducing the influence of unreliable crowd members, and thus is susceptible to a single crowd member quickly and repeatedly voting for their own intentions, and thus navigating the drone off-course (Stefanovitch et al., 2014).

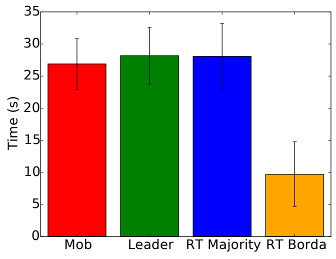
[4]All results reported as significant were confirmed with t-tests at a Bonferroni corrected significance level $p < \frac{0.05}{4}$ and all figures show 95% confidence intervals.



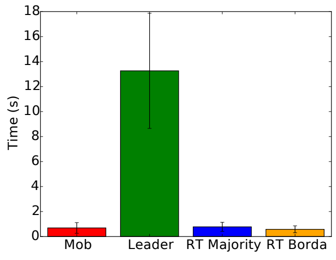
(a) Time Between Rings
(*Figure of Eight*)



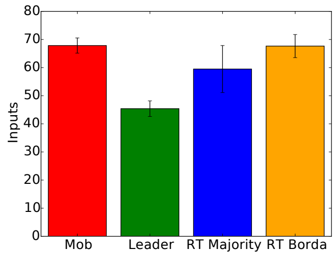
(b) Positional Error
(*Figure of Eight*)



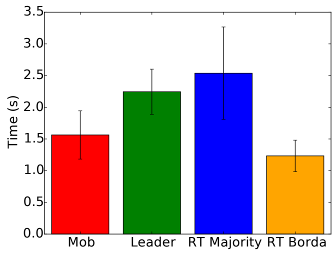
(c) Stationary Time
(*Figure of Eight*)



(d) Inaction Time
(*Figure of Eight*)



(e) Input Rate
(*Figure of Eight*)



(f) Reaction Time
(*Dynamic Blocked Path*)

Figure 4.3: Performance Graphs

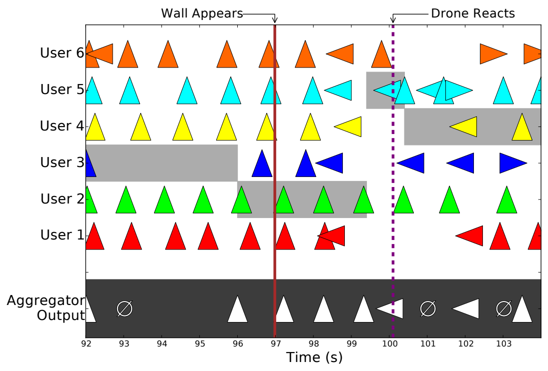
The *Real-Time Borda* aggregator also shows significantly less time spent stationary (9 seconds) than the others (Figure 4.3c), as the aggregator enables the drone to move and turn at the same time. Notably, the other aggregators spend nearly half the time rotating, which is expected on a circular path such as the Figure of Eight. The *Leader* aggregator remains inactive 12.5 seconds per minute, significantly longer than any other aggregator (Figure 4.3d), which is because control is given to a single operator, and the drone can only ever act as quickly as that operator gives commands.

These results show that if the application is likely to contain a flight path that requires a lot of turning (i.e., patrolling or surveying), then the *Real-Time Borda* saves time. Whereas if the task requires precision (i.e., package delivery), then the *Real-time Majority* aggregator would be a better choice.

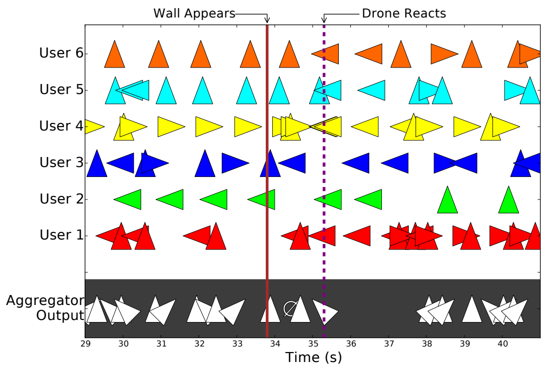
4.3.3.2 User Input

The *Leader* aggregator has significantly fewer inputs per minute (Figure 4.3e), which is indicative of a lower crowd engagement in these tasks. This effect may be explained by a lack of feedback. For many users their input 'appears' to have no effect, given that only a single user is controlling the robot, whereas, for other aggregation methods, the crowd's input has an appreciable effect on the drone's actions.

Figure 4.4 shows an example of typical crowd input for the *DBP* scenario centered around the moment a wall appears. We can see the clear differences in user input rate between the *Leader* aggregator and the *Real-Time Borda* aggregator.



(a) *Leader* Aggregator



(b) *Real-Time Borda* Aggregator

Figure 4.4: A short extract of real crowd input and the aggregated output. This illustrates the difference in Reaction Time between the aggregation methods, and the difference in Input Rate received from individual crowd workers.

4.3.3.3 Response to dynamic events

The reaction time is the time taken for the drone to react to an external change (see Figure 4.3f). The *Real-Time Borda* aggregator is significantly faster to react (1.2 seconds), the separated linear and angular preferential elections require fewer crowd members to vote for an action before the drone reacts. The *Mob* aggregator is also able to react quickly (1.6 seconds), as it instantly applies the input of the fastest crowd member, but slow crowd members can negatively affect its reaction time. Similarly, the *Real-Time Majority* can only react to the event when a majority of users have changed their vote, thus slow users can affect this aggregator too. The *Leader* aggregator, instead, only reacts as quickly as the crowd member currently in direct control of the UAV. If the leader reacts too slowly, the crowd begins to disagree with them, and another leader will be elected (see Figure 4.4a).

Figure 4.4 shows a comparison between the *Leader* and the *Real-Time Borda* aggregators, for the same *DBP* scenario with similar crowd sizes. At the far left of Figure 4.4a we can see the current leader highlighted with a grey background, User 3, but due to inaction control is then passed to User 2. When the wall appears, User 2 does not react, while the rest of the crowd shows disagreement with this, and thus control is again passed, now to User 5. It is the combination of low input rates and these slow to respond leaders, that make the *Leader* aggregator a poor choice for scenarios that require quick reaction times. In contrast, Figure 4.4b shows that the *Real-Time Borda* aggregator outputs many more actions per minute, often making many micro adjustments along the flight path. After the wall appears, we can see the drone begins turning even before the majority of users have reacted. This is due, in part, to User 1 and 3's fast reactions. Furthermore, we can see that User 2 has been consistently disagreeing with the crowd, therefore User 2's votes would have a lower weighting.

In general our results show that the *Leader* aggregator is often unsuitable for real-time control of a UAV. Instead, for applications that require speed, and quick reactions the *Real-Time Borda* aggregator performs best. On the other hand if the application requires accuracy, at the cost of speed, *Real-Time Majority* is a better choice.

4.4 Summary

In this chapter, we introduced CrowdDrone as a testbed platform for evaluating various real-time aggregation methods of crowdsourced input for directional sensing. We then described two novel methods of real-time aggregation, *Real-Time Majority* and *Real-Time Borda*, and used CrowdDrone to evaluate their suitability for crowd robotic applications. Our empirical evaluation on a real-time crowd hired from AMT showed that the *Leader* aggregator, the state-of-the-art method, is not ideal for real-time control of a UAV. The input rate and reaction times of individual users are too slow for practical use. The *Mob* aggregator is also not usable for our applications, given that it does not filter unreliable users. Thus, malicious crowd workers are capable of manipulating the UAV to their own ends. Instead, *Real-Time Majority* and *Real-Time Borda* provide faster or more accurate control while still reducing the impact of unreliable crowd workers. The results show that *Real-Time Majority* is useful in applications where precision is required, and that *Real-Time Borda* is a faster alternative for patrolling and surveying applications.

The next chapter looks at how we can use real-time crowdsourcing to identify the location of search targets in live video feeds, CrowdAR. Unlike real-time crowdsourcing for control, which can be used to directly influence a robotic agents path, real-time crowd sourcing for sensing can be used to provide additional information live, while the robot is active. However, CrowdAR differs from CrowdDrone in the techniques used for elicitation of input from the crowd and how to aggregate it. This live locational information can be used by a number of different processes, such as reducing the cognitive burden of a pilot, or rapidly mapping detected targets onto a map. Furthermore, an AI hybrid could be envisioned that combines both real-time crowdsourcing for control and sensing, such that, instead of a pilot, the live information feed is used to inform an AI control algorithm, thereby indirectly controlling the robotic agent, as opposed to direct control from the crowd. We explore a use case that augments an expert's (i.e., a pilot) video feed and improve their decision making in real-time.

Chapter 5

Real-time Crowdsourcing for Sensing

In order for Humanitarian workers to make time critical decisions, such as triage, or rapidly responding to those most in need, the first responders require up to date situational awareness. Increasingly, UAVs equipped with cameras are being used to gather situational awareness and find areas of interest. However, these feeds are typically reviewed retrospectively (i.e., after the UAV flight) and can take days to analyse. While potentially helpful information can be extracted in this way, it does not enable useful actions to be taken live (e.g., flying the UAV to get a better vantage point, or rapidly responding to a developing crisis). In the previous chapter, we introduced real-time crowdsourcing for robotic control, enabling intelligent flight decisions to be made live by the crowd. These flight decisions are not designed to provide situational awareness, and instead focused on intelligent navigation. In this chapter we address this challenge, and develop a real-time crowdsourcing system for analysing aerial imagery live (i.e., during the aircraft's flight), in order to enable a more timely humanitarian response.

Analysing live aerial imagery requires human perception and understanding. For example, search and rescue teams use UAVs to search for areas of interest that may differ from one situation to another (e.g., finding missing persons, locating damaged buildings, or identifying floating plane debris), and frequently change appearance (e.g., due to lighting conditions, cloud cover, or different architecture). These search tasks are highly variable and the targets are different each time with few or no prior examples, natural language is the easiest way for end users(i.e., the humanitarian workers) to convey what they are searching for. Given that search and rescue or situational awareness operations vary wildly between deployments, developing a single algorithm for each search target or for each possible deployment is infeasible. Likewise, a deep learning approach requires masses of data for training, and should a new search target be required it would need retraining. Often times in a disaster, labelled data has yet to be collected, and previously collected data may not apply during this deployment due to differences in search requirements, differences in architecture, and differences in environment, affecting the ability of the classifier to work effectively. A lot of the work in this area is still in its infancy, and effective automation is still a long way off (Law and Ahn, 2011).

Real-time crowdsourcing for human-intelligent sensing can be used to locate search targets in a live video feed. These annotations can then be aggregated and used in any number of real-time processes, Section 5.2.5. One such example explored in this work is to augment the live feed with the crowdsourced data, thereby reducing the cognitive load required by the observer. These annotations could be used to inform the user viewing the video feed (e.g., a UAV pilot of damage, or a CCTV operator of suspected targets) enabling the expert to make more effective timely decisions, such as surveying the areas of most damage, or responding to immediate events. Furthermore, real-time crowdsourcing enables many more people to observe the live camera feed, and in so doing, they are more likely to observe and accurately identify important details (i.e., finding search targets) than a single expert could.

In order to enable this human-intelligent locational sensing, a number of important challenges need to be addressed.

- We need to understand human performance at providing locational information in a real-time video annotation task (Requirement III).
- We need to develop the platform and design an effective interface in order to elicit such real-time input from unskilled workers (Requirement I and III).
- Given that using crowd workers provides no guarantee on worker reliability, we need multiple workers to provide simultaneous input such that their tags can be aggregated and filtered in real-time. Therefore, we need a method to aggregate multiple worker's input in real-time, requiring a process that has a low computational cost (Requirement I, IV, and V).
- As there is an inevitable latency between crowd worker input and the live video feed, mechanisms must be developed to reduce the visible or negative effects of this lag (Requirement III).
- Develop a number of key applications that can use real-time human-intelligent sensing.
- Evaluate how users interact with such platforms.

Thus, in this chapter we describe Crowd Augmented Reality, or CrowdAR, a real-time crowd-powered system for providing . It is motivated by augmenting live video for UAV pilots during a disaster scenario to reduce cognitive load, but could be used in many other scenarios, as shown in Section 5.2.5.

We first evaluate human performance at providing locational information in real-time tasks, Section 5.1. Second, we describe CrowdAR, Section 5.2, and it's key contributions, how it addresses eliciting real-time input Section 5.2.2, how it reduces the effects of latency Section 5.2.3, and how it aggregates input to reduce unreliable input Section 5.2.4. Finally, we then evaluate the performance of this system in Section 5.3.

5.1 Human Performance on Real-time Input Interfaces

We need to understand human performance at locational sensing, such sensing requires crowds to provide the location of moving search targets in a live video feed in real-time. However, live video annotation presents a number of challenges that need to be investigated. First, we investigate the appropriate time constraints that can be met by workers providing the location of targets, such that the information can be utilized while the targets are still visible. Second, we investigate the accuracy of workers when providing real-time locational input, including how their performance is affected by external factors of the video feed (e.g., how fast the search targets are moving, or how many search targets are present on screen).

Real-time crowdsourcing projects elicit input from crowd workers within a matter of seconds, enabling real-time interaction with the application. The strategies used by these applications for rapid input elicitation are typically simplistic (Section 2.4), enabling workers to perform them quickly and easily. In this instance, providing rapid locational input that could be used identify search targets in imagery, video, or point cloud data, we choose to limit the input to solely using mouse clicks on the search targets, thereby reducing the complexity and learning curve of the task. This raises the question of how fast and how accurately can crowd workers provide this rapid click input.

There is a long history of work modelling human performance on motor tasks. Human factors, psychology, neuroscience, and human-computer interaction (HCI) have all explored people's ability to acquire, track, and mark a target. The field of HCI has studied how models of human performance can be applied to motor input tasks, for example, Fitt's law (Fitts, 1954), and was later adapted to model cursor-pointing tasks (MacKenzie, 1992). However, this previous work does not address human performance at clicking on multiple moving targets, as may be the case in tagging live video. Furthermore, by distributing tasks over multiple contributors, real-time performance on demanding psycho-motor tasks has been improved beyond the ability of any single member of the crowd (Lasecki et al., 2012, 2015b, 2013a). In order to understand the limits of a human-intelligent sensing system, we need to investigate where the bounds of human performance lie in regards to successfully identifying search targets given the speed and

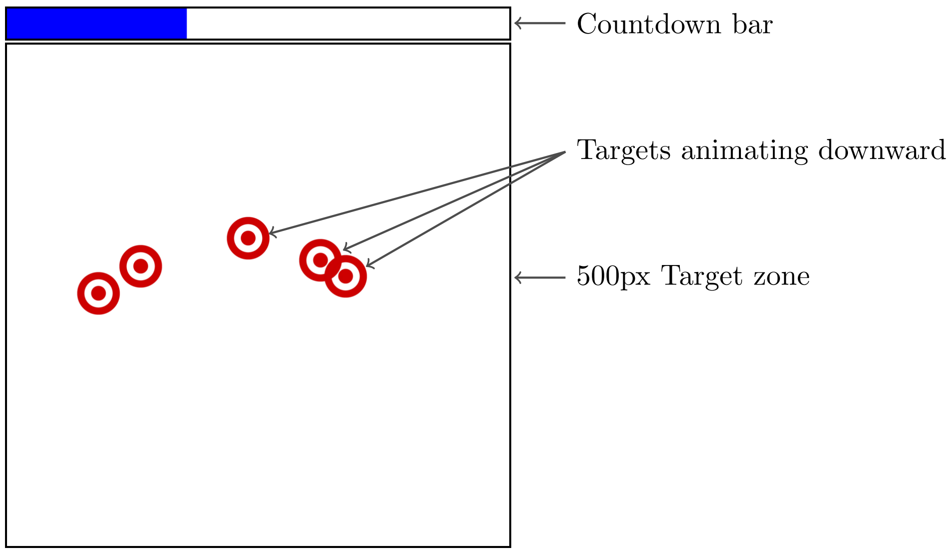


Figure 5.1: Screen shot of our moving-target task UI.

number of targets present in the video stream. In the case of robotic control, this performance would limit how fast the robot could move, or how visually complicated the scene could be. Therefore, in what follows, we set out to explore the parameters that influence human performance in real-time target identification tasks.

5.1.1 Task Design

In this section, we design an experiment to investigate the crowd worker performance on a target-identification task. We measure the individual and collective response speed and accuracy and find that human-performance degrades when there are multiple fast moving targets on screen. we investigate the use of 2 different interfaces crowd workers may be presented with to interact with this system. First, we show the targets animating smoothly from start to end, a proxy for displaying live video, the Live Video Interface (LVI). Second, we show a still frame of the animation for a short duration, similar to previous Fitt's law experiments where the targets are stationary, the Still Frame Interface (SFI). Our still frame presentation methods helps to improve worker performance, and through aggregation techniques, the performance can be improved further still. While prior work has explored individual performance on controlled motion-based target acquisition tasks (Jagacinski et al., 1980), this is the first work to explore collective performance on a task where moving targets are displayed with varying tradeoffs in movement speed and continuity of motion.

In order to evaluate the crowd workers' accuracy and speed, we designed a simple task in which they must click on multiple moving targets within a square zone 500 pixels in width. The targets that the crowd workers aim for are a circle with a radius of 25 pixels (see Figure 5.1). Once a crowd worker clicks on a target it disappears, but if the crowd worker clicks and misses the target there is a visual cue notifying the user to the misclick.

The crowd worker is presented with a short description of the task, and a start button. Once the task begins, targets are then spawned at the top of the zone and move linearly in a random downward angle towards the bottom of the zone. The start location of the target is constrained to the top edge of the zone, but may randomly appear anywhere along the length of that edge. Likewise, to ensure that the random downward angle keeps the target within the bounds of the square zone, we randomly select a location on the the bottom edge of the zone, and move the target towards this location. This is considered a round of targets, and within a round all targets move at the same speed. When all targets are destroyed, either by crowd workers clicking on the target or when all the targets reach the end of their movement animation, the round is over. Once the round has ended, there is a 500ms delay and a new round begins. Each new round will spawn a different number of targets, with a different animation speed.

In the LVI, targets animate from the top of the zone towards the bottom, but given the random start and end locations, may vary greatly in path length, and thus the time they are present onscreen. This would give crowd workers more time to click some targets than others. To rectify this, we calculate the minimum time it would take for a target to reach the bottom edge of the zone (i.e., an exactly vertical path) and the round will end after this time has elapsed.

The SFI shows a still frame from this animation. The targets remain stationary for a given duration, but after this duration the targets will instantly 'jump' to where they would have been if they had been animating smoothly. This is analogous to showing still frames of a live video, and selecting the most recent frame once the duration is up. Depending on the speed of the targets, the number of still frames shown per round may vary, the faster the targets move, the fewer the number of frames that may be shown before the round is over. The SFI also activates a countdown bar at the top of the zone. At the start of a new round the bar is at 100% and animates towards 0% over the duration of the still frame. This helps workers understand how long they have left on this frame, such that the targets do not unexpectedly jump.

5.1.2 Experiment

We experiment with four different treatments, showing crowd workers either the live video interface (LVI), or the still frame interfaces (SFI), with one of 3 different still frame durations ranging from 1, 2, or 3 seconds. We recruited 160 crowd workers from Amazon Mechanical Turk to perform the target clicking tasks, and each worker is presented with one of the 4 interface treatments, an equal number of workers (40) per treatment. The workers are paid $0.30 for the trial, and to avoid a training effect, could

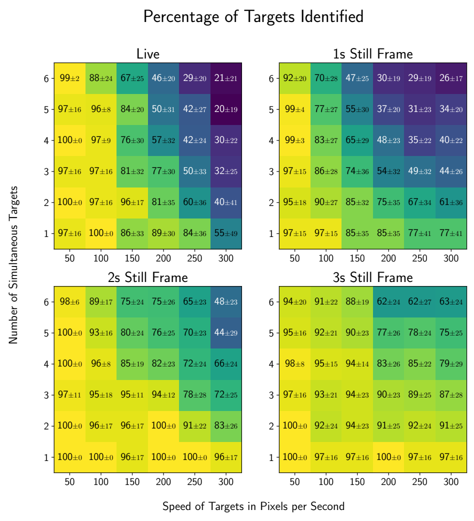


Figure 5.2: Average percentage of targets identified.

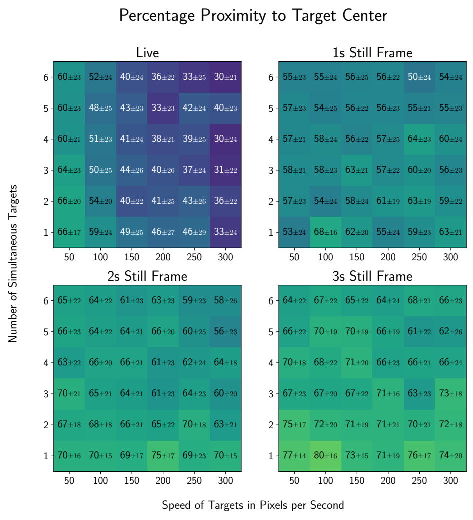


Figure 5.3: Average proximity to the center of the target. 100% being the exact center.

only participate once in the trials. Enforcing unique workers was achieved by giving workers a custom "qualification" that was used as hiring constraint. Only workers without such a qualification could participate. Furthermore, only workers from the US were allowed to participate, ensuring that the demographics of our study match those that typically take part in most of the real-time crowdsourcing tasks described in past literature (Section 2.3).

We investigate the effects of the different interfaces in a between subjects study. Each crowd worker will be shown either the LVI, or SFI with a duration of 1, 2, or 3 seconds. We measure each crowd worker's performance on a total of 36 rounds, each round has a different target speed, and number of targets present. The targets speed range from 50 pixels per second to 300 pixels per second, with increments of 50. The number of targets present in a round, range from 1 to 6. The order in which these rounds are presented is randomized to avoid a bias in the ordering of the rounds.

5.1.3 Results

As the data is multifaceted, we present this data in multiple heatmaps, one for each interface, live, and the 3 still frame durations. We explore how these different interfaces affect human performance at clicking on multiple moving targets.

In Figure 5.2 we can see that as we progress from live, to still frames with longer durations, crowd workers are able to identify (i.e., click on) a greater percentage of the faster moving targets. In Figure 5.3 workers are observed as having greater accuracy (i.e., their clicks are closer to the target centre) on the still frame tasks. In Figure 5.4 we

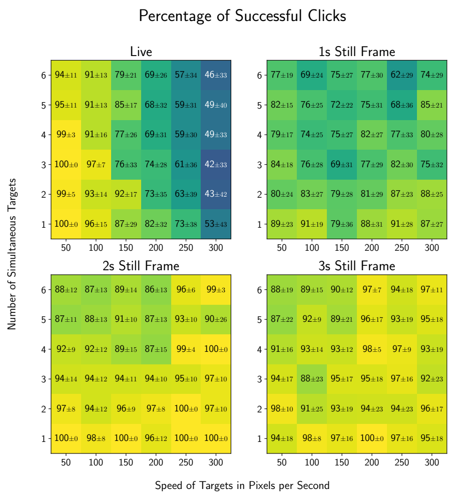


Figure 5.4: Percentage of workers clicks that land on the target.

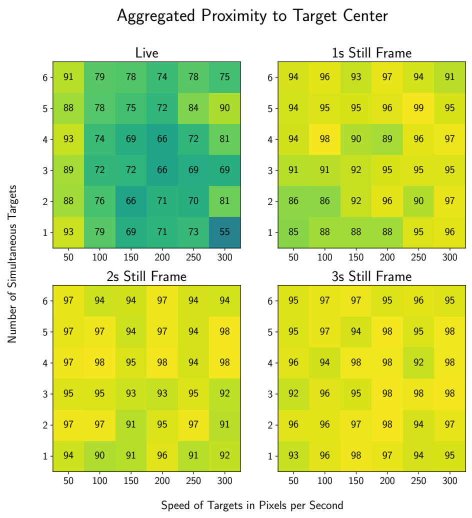


Figure 5.5: Aggregated proximity to the center of the target.

can see that the speed at which the targets move affects the ability of the LVI workers to click on the targets. At the highest speeds of our tests, over half of the workers' clicks missed the target and instead hit the background. Naturally, the speed of the targets does not affect the workers' ability to successfully hit the target in the SFI, although a shorter duration of the SFI does negatively impact the workers' ability.

Furthermore, by aggregating crowd workers' clicks, we can provide a better performance than any or most individual members of the crowd. We calculate the average position of all workers' clicks on the target. However, since the path of the targets was randomized, to remove the effect this might have had, we rotate each target, and the workers mouse position, such that each target followed same vertical path. Thus allowing us to aggregate a between subjects randomised path. In Figure 5.5 we observe how the accuracy of the aggregated results is improved upon compared to the individual (Figure 5.3).

5.1.4 Discussion

In the context of using worker clicks to identify the location of targets in a live video stream, we have observed that showing the live motion of targets to workers makes the task harder. As the rounds become more challenging (i.e., the number of targets and the speed of the targets increase), workers become less accurate and less able to provide the location of all the targets. In contrast, the still frame interface is able to retain the worker's accuracy and ability to click on multiple targets even at more challenging rounds. The longer the duration of the still frame, the greater the improvement, but the greater the delay in input being received by the real-time process. We need to choose a value for the delay that is as short as possible (in order to enable real-time processing of crowd worker input) but retain good human performance, especially during the harder

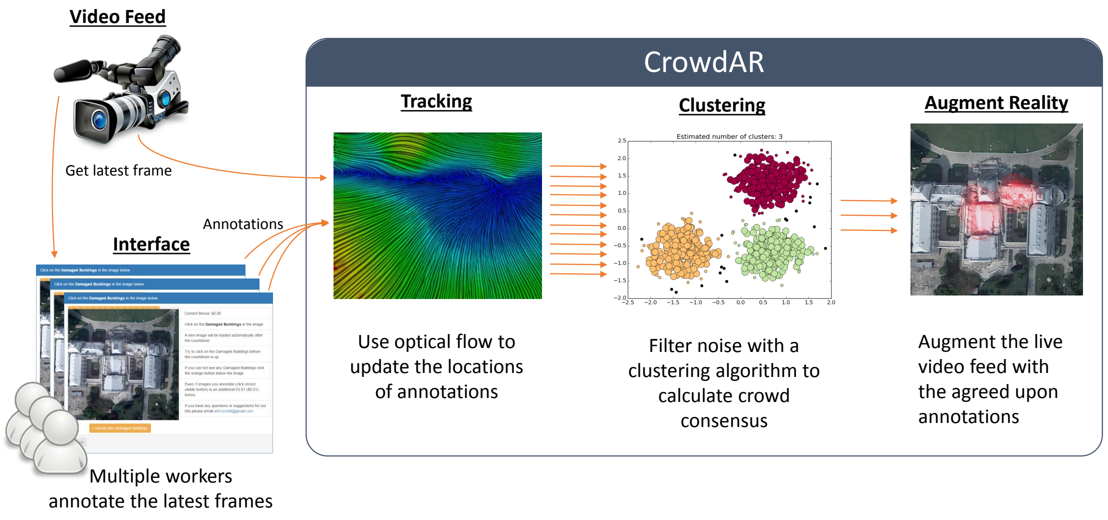

Video Feed
Get latest frame
Annotations
Interface
Multiple workers annotate the latest frames
CrowdAR
Tracking
Clustering
Augment Reality
Use optical flow to update the locations of annotations
Filter noise with a clustering algorithm to calculate crowd consensus
Augment the live video feed with the agreed upon annotations

Figure 5.6: CrowdAR process diagram

settings (i.e., when there are many fast moving targets, the top right of the heatmaps). We propose that a 2 second duration, given our results, would be best suited for real-time locational sensing, it performs consistently better than LFI and the 1s SFI, while we observe diminishing returns between the 2 and 3s SFI. In particular, the percentage of targets identified Figure 5.2, and the percentage of successful clicks Figure 5.4, was used to make this decision, as accuracy (i.e., proximity to target centre, see Figure 5.3) can be improved through aggregation regardless.

The goal of this study is to inform the design of a live video tagging interface, specifically for tagging live aerial imagery during a disaster. Next, using the knowledge gained in this preliminary study, we describe the implementation of CrowdAR, a more sophisticated interface for real-time crowd workers to identify search targets in live video feeds.

5.2 CrowdAR

In this section we outline the architecture of CrowdAR, a system for human-intelligent sensing from a live video given the input of a real-time crowd, and discuss the trade-offs considered during its design. We show the general purpose nature of this tool using two domains, video surveillance (i.e., CCTV) and crowd robotics (i.e., a UAV flight). CrowdAR consists of five key components.

1. Recruiting a crowd of simultaneously online workers.
2. A web interface that real-time crowds interact with.
3. Reducing perceived latency to update a worker's previously provided tags to their most likely location in the current frame of the live feed.

4. Calculating consensus amongst workers to reduce noise.
5. Processing crowd tags to provide a meaningful output, for example, by augmenting live video feed seen by an expert to reduce cognitive load, or rapidly annotating a map with the locations of detected targets.

These five components will be described in more detail through the remainder of this section.

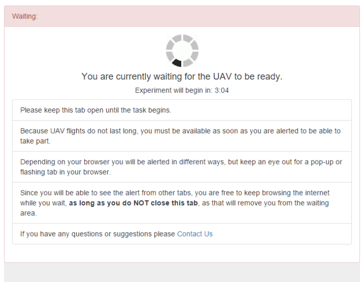
5.2.1 Recruiting a Real-time Crowd

Due to the nature of live video, and especially live aerial footage, the video feed will not always be broadcasting. Furthermore, in a crowd robotics domain, the broadcast will likely only be active for a short time, given that battery life of a robot, especially with UAVs, is limited. To ensure that we have a crowd of workers ready to begin tagging footage the moment the feed is active, we need to pre-recruit a crowd. We use the recruitment management tool described in Chapter 3.

The recruitment manager pre-recruits a crowd and holds them in a waiting room, until work is available (see Figure 5.7a). Given that we intend to use CrowdAR in disaster response scenarios, we expect our crowds to consist of both paid workers and volunteers, with volunteer crowds making up the bulk of the work initially while media coverage of the disaster is high. The method of pre-recruiting a crowd differs for paid and volunteer crowds. Paid crowds from crowdsourcing marketplaces such as AMT, can be called up at any time to do a test of the system. A paid crowd is pre-recruited using the strategy described by the recruitment management tool in Chapter 3, and given the large user base of these platforms we can ensure fairly rapid recruitment. In contrast, recruitment of volunteers may not be as fast, given the smaller pool of available volunteers, whom may require greater notice to become available.

The volunteers we recruited in Chapter 6 have intrinsic motivations to help during actual deployments, and also often partake in trials of new humanitarian systems to help improve them. This motivation helped in our recruitment process but also meant that we had to book a specific time to perform the trial (see Section 6.2). Volunteers could be alerted in a number of ways, for example via emails, push notifications, or Skype calls. The volunteers are informed about the upcoming task and asked to follow a link to the waiting room. Once a real-time crowd has been recruited, they can be sent to the Live Tagging Interface, see Section 5.2.2.3.

Because real-time crowdsourcing requires a constant number of simultaneously active users, if this number begins to drop, we need to replace these workers within a matter of seconds. This has been shown to be possible (Bernstein et al., 2011) when using the recruitment manager on crowdsourcing marketplaces, whereas it is currently unknown


Waiting:

You are currently waiting for the UAV to be ready.

Experiment will begin in: 3:04

Please keep this tab open until the task begins.

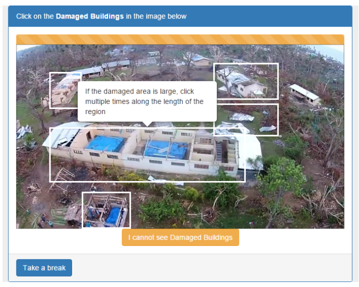
Because UAV flights do not last long, you must be available as soon as you are alerted to be able to take part.

Depending on your browser you will be alerted in different ways, but keep an eye out for a pop-up or flashing tab in your browser.

Since you will be able to see the alert from other tabs, you are free to keep browsing the internet while you wait, **as long as you do NOT close this tab**, as that will remove you from the waiting area.

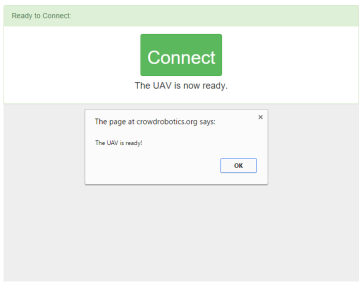
If you have any questions or suggestions please Contact Us

(a) The Waiting Room


Click on the **Damaged Buildings** in the image below

If the damaged area is large, click multiple times along the length of the region

(b) The Tutorial Page


Ready to Connect:

The UAV is now ready.

The page at crowdrobotics.org says:

The UAV is ready!

(c) Alerted to Connect to the Live Feed

Figure 5.7: The Introductory Interfaces Before Connecting to the Live Feed

how quickly volunteer workers can be recruited or replaced for real-time crowdsourcing tasks. However, when asked, almost all online humanitarian volunteers said they would immediately drop what they were doing to help during a real disaster.

5.2.2 Interface

CrowdAR could be used to identify and track any target asked of the crowd through a natural language description. In the video surveillance domain, the description could be a witness description of a search target. Whereas, in our crowd robotics domain we expect the description would be given by the UAV operator and describe the search targets that they intend to find during their deployment. For our investigation we choose to identify damaged buildings, as this is a typical task required by first responders, and thus rapidly identifying these will support their humanitarian efforts. The following section describes the system in the context of identifying and tracking damaged buildings in live aerial footage.

From the perspective of the crowd workers, CrowdAR's interface consists of three main stages. The first stage that participants will encounter is the **Waiting Room**. Here, users wait for the live video feed to begin broadcasting. The second stage is the **Tutorial** page where participants can acquaint themselves with the upcoming task. Finally in the third stage, once the live feed is ready, users will connect to the **Live Tagging Interface** where they annotate frames from the live video feed. We cover each of these in turn.

5.2.2.1 Waiting Room

The waiting room informs participants that they will be alerted when the feed is ready (see Figure 5.7a). Provided the workers keep the waiting room tab open in their browser, they may continue to use their computer as usual, and browse other pages. Given this freedom to browse other pages, workers may then review the tutorial page. Once alerted, the majority of workers typically return to the task within 2 seconds (Bernstein et al., 2012).

5.2.2.2 Tutorial

The tutorial was designed in partnership with the Standby Task Force (SBTF), see Section 6.1. Their years of experience in dealing with similar tasks and in training their volunteer workforce, was essential for creating a tutorial that was succinct and informative. Together, we iterated on a number of versions, identifying areas where the participants would want more information, and how to get that across to them in a familiar and easy to use manner. The final version of the tutorial is seen in Figure 5.7b. As can be seen, it displays an interactive example of the live interface.

In more detail, an example still frame from a UAV flight is shown to the crowd. Within the frame, white rectangles highlight the types of damage they might observe. Hovering over these rectangles with the cursor provides a popover giving more detail about the damage, and how they should interact with it. They are told to click within the confines of the building, and that if the damaged region is large, they should try to click multiple times within its area, ensuring CrowdAR receives good coverage of the entire building. Our deployments aimed to identify damaged buildings after a disaster, thus we adapted the tutorial to this scenario. Depending on the task at hand, different tutorials could be developed.

Once the live aerial feed is ready to broadcast, participants are alerted, Figure 5.7c, and they begin tagging the live feed.

5.2.2.3 The Live Tagging Interface

The live tagging interface[1] enables users to identify search targets in the live video feed. A naive implementation of the web interface (used by previous examples (Tewksbury, 2012; Trottier, 2014)) is to simply stream the live video to the workers. Eliciting real-time input requires workers to click or drag their mouse over moving search targets. However, given the preliminary investigation (Section 5.1) into real-time input interfaces, we conclude that a still frame interface with a short duration of 2 seconds provides the best tradeoff between human performance and real-time data. In addition, this method requires less bandwidth (which may be limited in disaster scenarios).

Users are shown a single still frame from the live feed for a short duration, a countdown timer bar that displays the duration left of the still frame, and two buttons, indicating that there is no damage visible in the frame, or that the user wishes to take a break (see

[1] A video of how to interact with the interface is shown here: `https://vimeo.com/140445323`

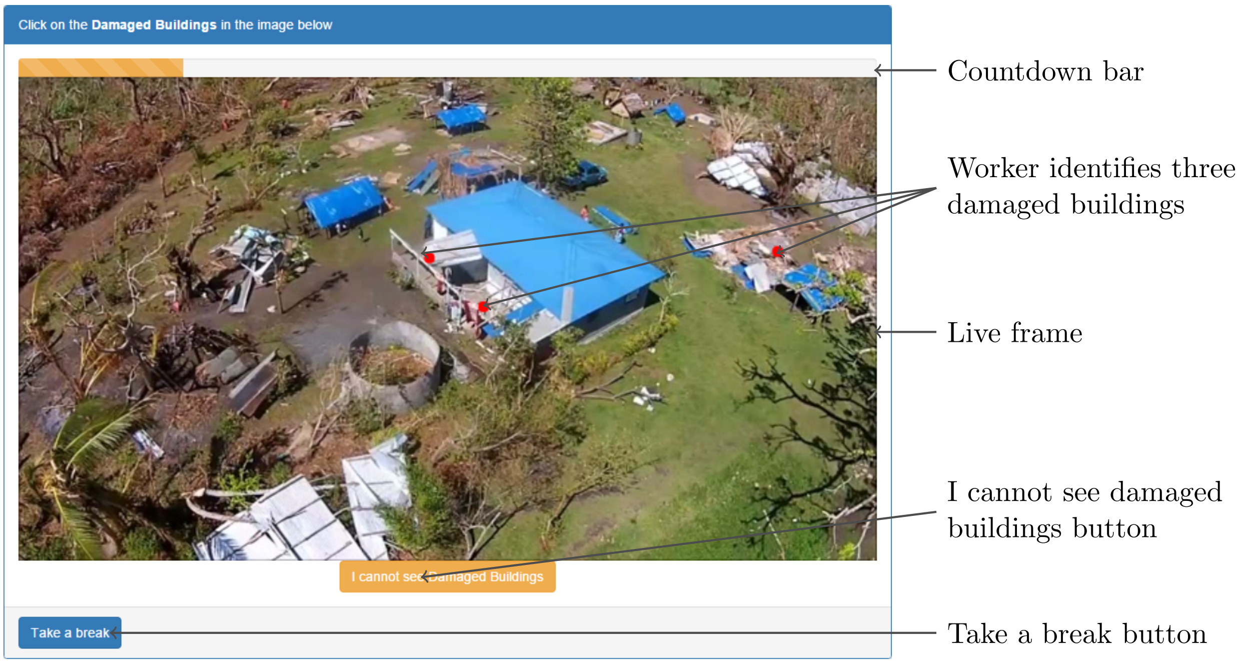

Click on the **Damaged Buildings** in the image below
Countdown bar
Worker identifies three damaged buildings
Live frame
I cannot see damaged buildings button
I cannot see Damaged Buildings
Take a break
Take a break button

Figure 5.8: The Live Tagging Interface

Figure 5.8). This enables us to identify whether a lack of annotation was due to this, or due to worker inactivity.

This still frame interface works as follows. The crowd worker, u, is presented with a still frame, f_t, which is the most recent frame of the video at time t. Above the frame, the worker observes a bar counting down d seconds until the next frame is presented. Section 5.1 revealed that showing still frames with a short a duration results in greater accuracy when more search targets are present on screen, and when the targets are moving fast. As previously stated, we have chosen d to be 2 second which gives an acceptable tradeoff between worker performance, and the need for real-time input from workers. Furthermore, 2 seconds is enough time for workers to click on multiple targets, and for CrowdAR to receive the data soon enough to process the tags live.

The worker may then click on the frame any number of times, indicating where they believe the search target to be. These annotations are denoted by $\alpha_{u,t}$.

$$\alpha_{u,t} = \{a_1, a_2, \ldots, a_n\} \tag{5.1}$$

Where a_i is an x, y coordinate in the frame. After the countdown, all of the worker's annotations, $\alpha_{u,t}$, are transmitted to CrowdAR for further processing. If the worker does not tag anything in this frame then $\alpha_{u,t} = \emptyset$.

The still frame interface introduces some delay between the received annotations and the current frame of the live video. Hence the need to balance a shorter d against human performance. In what follows, we describe the methods used to help rectify this, but if the value of d is too long, or the search targets move too fast, then the targets will be out of frame of the video before we even receive the worker's tags.

5.2.3 Reducing Perceived Latency

There is always some unavoidable delay between the received annotations, $\alpha_{u,t}$, for frame f_t, and the current frame of the live video feed, $f_{t'}$.

$$t' - t = d + l \tag{5.2}$$

Where d is the duration of the still frame task, and l is the delay caused by inherent network latency and by the still frame interface described above. This means that the annotations used to augment the video feed may no longer be relevant or in the correct location. We therefore require a mechanism to reduce this latency such that the human-input remains relevant to the live feed. Thus, CrowdAR uses two techniques, preloading and tracking, to help rectify this inherent lag.

5.2.3.1 Preloading

Just in time preloading is used so that the worker does not need to wait l seconds for the next still frame to download. We measure the network latency, l, between the CrowdAR system and each worker. Thus, the worker's interface should start preloading the next frame $d - l$ seconds into the countdown, such that the frame is downloaded just before the still frame countdown expires. This ensures that the worker is always viewing the most recent frame, without needing to wait for it to download, thus increasing their throughput.

5.2.3.2 Tracking

Computer vision techniques are used to further reduce the effects of network, l, and interface latency, d. Specifically, CrowdAR uses dense optical flow (Farnebäck, 2003) to track workers' annotations through the video feed, ensuring that they are tracking the movement of the search target. Optical flow is a technique to determine the movement between two frames at the pixel level. Calculating the movement of pixels between every frame allows us to iterate through from the frame that was annotated to the live frames being shown to the pilot, updating the tag's location. Optical flow performs best on feature rich video feeds as in our example scenarios, but may fail to track if the target becomes occluded. However, given the high vantage point, this is rarely an issue in the case of aerial footage, but some surveillance applications, such as CCTV footage, may be affected depending on the placement of the camera. Nevertheless, to rectify this, and to reduce the effects of noise introduced by tracking, we time-out tracked annotations after 10 seconds. Workers are then required to occasionally click on the same target, ensuring that their annotations do not time-out.

Let O be a function that transforms an annotation, a_i, from its location in frame f_t to its tracked location in the current frame, $f_{t'}$.

$$O(a_i, f_t, f_{t'}) \to a'_i \tag{5.3}$$

Then we propagate all of the workers' annotations for frame f_t to the live frame, $f_{t'}$, giving the following.

$$\alpha'_{u,t} = \{O(a_i, f_t, f_{t'}) | a_i \in \alpha_{u,t}\} \tag{5.4}$$

This method is then applied to all of the worker's annotations.

$$A'_u = \bigcup_{t'-10<t<t'} \alpha'_{u,t} \tag{5.5}$$

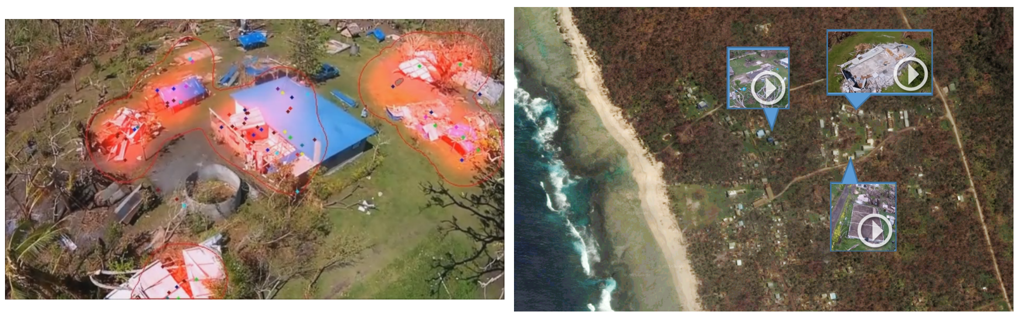
We now have all workers annotations tracked to the current frame, but because little is know about the quality of workers and their input, we need to first verify the annotations. We achieve this by calculating which annotations have been agreed upon by multiple workers, as we discuss next.

5.2.4 Calculating Consensus

Because CrowdAR is an open platform, workers of varying skill may participate. Thus it is possible that some workers will prove unreliable, providing incorrect input either accidentally or maliciously. To filter this noise we require multiple workers' to aggregate their input and calculate a consensus.

To that end, once the annotations are tracked to the current frame, we need to identify if there is indeed something relevant worth augmenting the video with, or if the annotation can be considered just noise. An annotation is considered relevant if multiple crowd workers have tagged roughly the same location in the frame. A clustering algorithm, DBSCAN (Ester et al., 1996), is used to calculate a consensus of the crowd's input. A cluster with at least three unique workers' annotations would be considered a verified search target. This value can be set by the system designer and will likely depend on the domain and crowd size[2] The chosen clustering algorithm does not require a predetermined number of clusters (i.e., the number of targets present is unknown), and can detect clusters of arbitrary shapes (i.e., the shape of targets is unknown). In particular, clustering allows us to determine a rough size and shape of the target, which,

[2]More sophisticated techniques for choosing this value have been left for future work.



(a) Augmented Live Video Shown to Pilot (b) Mapped Snippets

Figure 5.9: The Live Feed Interface and Output to First Responders

in turn, enables us to better process the crowd tags to, for example, augment the live feed. However, should multiple targets be close to, or even overlap each other, the clustering algorithm may consider them as only a single target. In our case of improving surveillance feeds, this is not an issue as an expert is still in the loop.

Let C be a function that calculates the set of clusters from a set of annotations.

$$C(\alpha') \rightarrow \{c_1, c_2, \ldots, c_n\} \tag{5.6}$$

Where c_i is a set of x, y coordinates in the frame. Thus, $\mathcal{C}$ is the set of clusters from each workers', $u \in U$, complete tracked annotations, A'_u.

$$\mathcal{C} = C(\bigcup_{u \in U} A'_u) \tag{5.7}$$

These verified annotations can now be further processed to provide experts with easy and understandable visualisations of the crowdsourced data, as we show next.

5.2.5 Processing Crowd Tags

CrowdAR is capable of analysing the live crowd data in a number of ways. Given the nature of crowdsourcing systems, and the importance of humanitarian aid, we suggest applications that retain humanitarian expert involvement, but simply reduce the cognitive load required of them to analyse the incoming data streams.

$$\Lambda(f_{t'}, \mathcal{C}) \rightarrow \mathbf{f_{t'}} \tag{5.8}$$

Where Λ is the live-processing function, that utilises the crowd consensus, the clusters $\mathcal{C}$, to display up to date, human-intelligent sensor readings to the expert operator. The implementation of Λ could be done in any number of ways. Some examples are detailed below. In the case of CrowdAR, we overlay a heatmap generated from each cluster (see Figure 5.9a).

For example, the following two methods are of benefit to humanitarian aid. First, CrowdAR can augment live video to create a human-intelligent sensor feed, showing the pilot the crowdsourced annotations during the flight, see Figure 5.9a. Thus, allowing the pilot to divert the path of the aircraft to investigate areas of interest identified by the crowd. An expert who views the feed could see the clusters, as processed above, overlaid onto the original frame (see Figure 5.9a). This could be useful for search and rescue flights. For example, flights over the ocean looking for missing boats and floating wreckage. CrowdAR could instead notify the pilot the moment the crowd spots something of interest. This enables the pilot to make a better judgement and decide to investigate immediately. In contrast, if the footage was analysed post flight, returning to the same area again would not be possible as ocean currents take the debris away.

Second, CrowdAR can rapidly map the search targets (e.g., damage) onto a GIS application (e.g., Google Maps). Once an area of damage has been identified by the crowd and tracked through the video by CrowdAR, a short snippet can be generated by cropping the video and highlighting just the search target. These short snippets of video may then be sent off for further verification and classification, by either an expert or a trained crowd. This significantly reduces the workload of the experts, who need only accept or reject the snippets, rather than watch a whole video. Unfortunately, if the novice crowd fails to observe a search target, then the expert will never observe the snippet to verify it. In practice, this can be solved by informing the crowd to be less discriminate in their tags (see Section 6.5.3). These snippets can then be plotted onto a map, see Figure 5.9b. All of this can be performed simultaneously with the live video feed tagging, and snippets can be verified and plotted within a matter of minutes of the UAV first observing the target.

Next we evaluate the performance of CrowdAR, investigating the accuracy of CrowdAR's live video annotations, and show how crowd worker inputs vary over time.

5.3 Empirical Evaluation

Here we evaluate the performance of CrowdAR, described in the previous section. This system works with any video feed, be that live or the playback of pre-recorded footage. The following experiments use pre-recorded video for the purpose of repeatability. CrowdAR is capable of analysing surveillance footage in real-time, highlighting the results to an observer intending to reduce their cognitive load. In this work, we focus on the domain of disaster response, and helping gather situational awareness, but it is possible to use CrowdAR in many scenarios. With that in mind, we evaluate CrowdAR on two types of surveillance footage, CCTV and aerial surveillance.

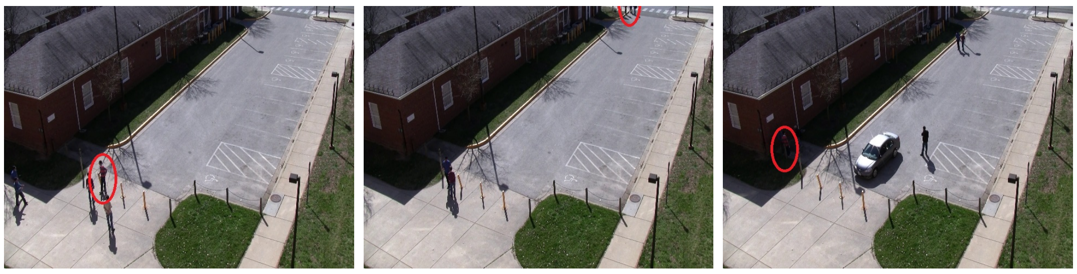

(a) Example Frame (b) Partial Occlusion (c) Poor Illumination

Figure 5.10: Example frames from the CCTV task, the search target is highlighted with a red circle.

5.3.1 Surveillance Footage

We use two different datasets to test the capabilities of this system. First, we use real CCTV footage[3] to investigate CrowdAR's ability to detect a search target given a natural language description (e.g., from a witness report). This is challenging for a single observer as the task is tedious, the description could be vague, or the target hard to see. Second, we use aerial footage[4] to investigate CrowdAR's ability to track multiple search targets (e.g., damaged buildings) in a much more challenging setting. This task is visually taxing and a single worker is unlikely to observe every detail.

5.3.1.1 CCTV

This is a highly tedious task, where, in current practice, experts have to watch multiple monotonous video feeds for prolonged periods of time. CrowdAR's approach to CCTV scales better than current practice. CrowdAR highlights the location of targets in the video feed, thus reducing the need for constant expert observation. Instead, experts could multitask more effectively, only needing to briefly observe the CrowdAR output and note the current location of the tracked suspects.

To be able to measure the accuracy of CrowdAR, in a real-world setting, we require video footage annotated with ground truth to compare the crowd performance against. Hence, we used the VIRAT Video dataset (Oh et al., 2011), which provides labelled surveillance footage of real world scenes.

In this experiment we test the crowd's ability to annotate a video feed given a natural language description. The video contains a number of people waiting in a parking lot, and the crowd workers are asked to identify the search target from a colloquial description, "guy in the striped sweater". This individual enters the video after 10 seconds, and stays visible for the remaining 14 minutes. Watching a fairly uneventful

[3]Watch an example of CrowdAR's augmented CCTV footage here: `https://vimeo.com/132313166`
[4]Watch an example of CrowdAR's augmented aerial footage here: `https://vimeo.com/130873053`

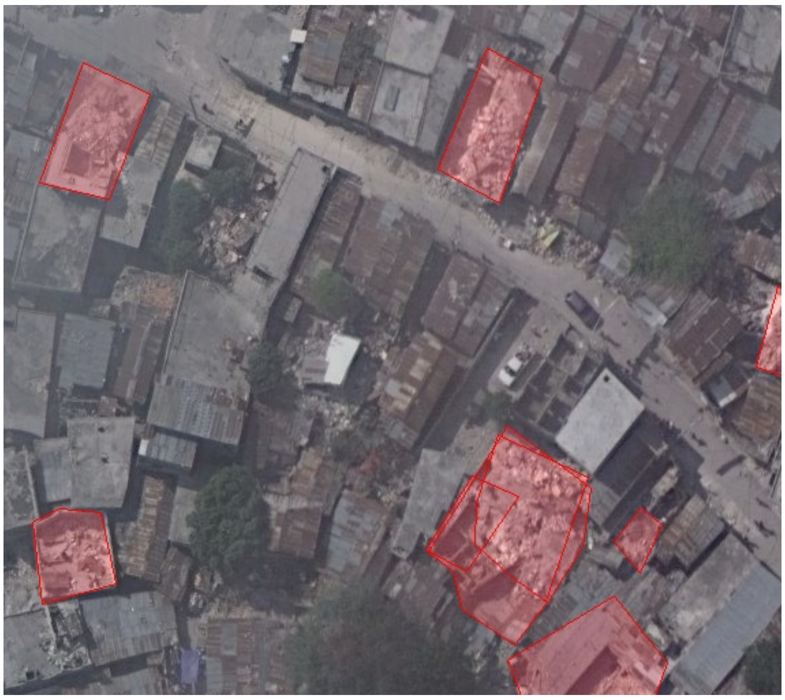


Figure 5.11: Example frame and ground truth (in red) from the Aerial Surveillance task

CCTV feed is a tedious task, and a single observer is likely to lose focus. This still remains a challenging problem for computer vision, as the target moves around the scene into areas of shade where the stripes on his top are no longer visible (large changes in illumination), or to partially occlude himself from the shot, such that only his legs are visible (see Figure 5.10).

Note in practice, we would envisage tedious tasks like this CCTV setting to be working in tandem with an automated process (similar to Zensors (Laput et al., 2015)), in which a computer vision algorithm is used to detect a scene change and request a crowd to check if a target is present, and to then alert an expert.

5.3.1.2 Aerial Surveillance

This relates more closely to the disaster response domain we are addressing, and can be a very visually taxing task, where, in current practice, expert pilots make flight decisions based on what they can observe. However, due to the nature of aerial surveillance, the video feed may move quickly, leaving little time to observe crucial information and make informed decisions. CrowdAR could enable the pilots to make more informed decisions, by highlighting search targets that expert pilots may miss.

Due to the difficulty in obtaining live UAV footage of a disaster scene, we instead generate a realistic video from high resolution aerial imagery. The imagery was sourced from Google Crisis Response[5], and contained imagery gathered of Port Au Prince shortly after the 2010 Haiti earthquake. We choose an arbitrary flight path over the Haitian capital and record a video scanning over the imagery, replicating the movement of an aircraft. The ground truth data is sourced from a UNITAR survey that identified the location and the destruction of every building in Haiti (UNOSAT, 2010). From this, an outline of all 266 damaged building on the flight path was manually created (see Figure 5.11).

The crowd was then asked to click on any "damaged building" they see. There are often many damaged buildings in the frame, and a single observer would struggle to identify them all. This is a challenging problem for computer vision to solve, as the damage appears in many forms and cloud cover sometimes obscures the buildings below.

5.3.2 Evaluation Metrics

The overall aim of CrowdAR is to assist an expert in doing their task more effectively. However, measuring expert performance is highly domain-specific and can often be difficult, especially in a repeatable, controlled experiment (e.g., preventing a crime, or achieving good situational awareness in disaster response). Instead, in order to evaluate the performance of CrowdAR, we take a domain-agnostic approach and focus on the quality of the produced annotations, using objective metrics. We compare the aggregated output of CrowdAR to that of a single worker. Both involve using non-expert crowd workers, but investigate whether aggregating many novices' input results in better performance in CrowdAR's real-time locational sensing setting. Thus, by comparing to a single worker, we show that aggregation is necessary to achieving a high performance. In so doing, we define the following metrics:

- **Precision:** The fraction of true positives, over all classifications, both true and false positives (i.e., $\frac{TP}{TP+FP}$). Put simply, in the case of worker clicks, it is the percentage of their clicks that they correctly identified. For CrowdAR, it is the percentage of clusters that correctly highlight a search target, for every frame.

- **Recall:** The fraction of true positives, over all the possible search targets, both true positives and false negatives (i.e., $\frac{TP}{TP+FN}$). In more detail, in the case of worker clicks, it is the percentage of search targets they correctly identified per frame viewed. For CrowdAR it is the percentage of search targets correctly highlighted by the clusters per frame.

  The actual numerical value of recall is not too important, and relates more to the difficulty of the dataset. Instead we are comparing the difference between individual workers' recall, and CrowdAR's recall.

[5]`www.google.org/crisisresponse`

- **Worker Responses Over Time:** The proportions of different response types given by crowd workers over the duration of the video. We use this to investigate how worker performance reacts to events in a video feed. We measure four types of responses, correct clicks (i.e., true positive), incorrect clicks (i.e., false positive), when a worker indicates they cannot see the target, and when a worker views the frame but does not provide feedback.

5.3.3 Experimental Settings

Throughout the experiments, we set the duration of the still frame interface, d, to 2 seconds. This was chosen, because preliminary experimentation showed this to be a reasonable value to balance human performance against throughput of annotations, see Section 5.1. Furthermore, this duration is slightly greater than the average network latency we measured in preliminary experiments (1.5 seconds on average in our experiments), enabling the preloading strategy, see Section 5.2.3.1. In practice, this duration can be adjusted by the system designer to take into account the complexity of the task, desired quality and network latency.

The total number of active crowd members (those who clicked at least once) varied from 8 to 28 per experiment, there were 349 unique workers in total, and on average, the system received 3 annotations from unique crowd members every second. Workers were paid \$0.01 for every 3 frames they annotated, a milestone bonus of \$0.01 was offered for every 20 frames, encouraging workers to keep going. This works out at a maximum hourly wage of \$6.90 per worker, should the worker annotate every frame they were presented.

In order to determine the effect that crowd size has on precision and recall, we computed these metrics for 100 samples of crowd input for each crowd size (from 3 to 20). Our results found no statistical significance in these metrics, but this was partly due to high levels of variance in the low crowd sizes. This variance decreases as more crowd members are present. In practice, we found that the appropriate crowd size is dependent on the latency of the workers and the video being annotated.

5.3.4 Results

We ran 10 experiments for each video feed. The real-time nature of these experiments is prohibitive to running large numbers of repeats. This section details the results of our experiments on both aerial and CCTV surveillance footage.

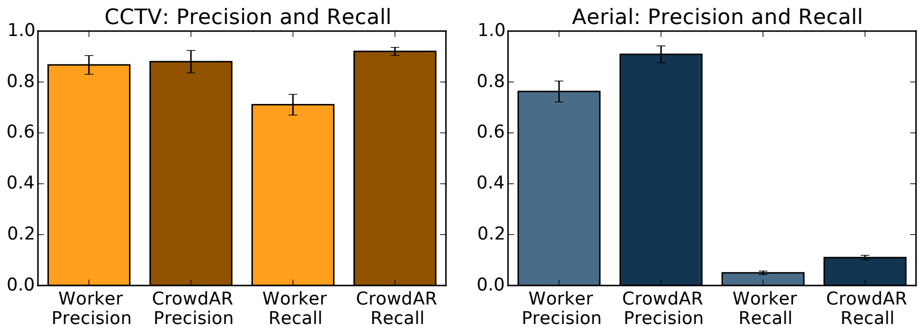


Figure 5.12: The precision and recall of workers clicks and the tracked clusters of CrowdAR.

5.3.4.1 Precision and Recall

The performance of both the Aerial surveillance and CCTV footage can be seen in Figure 5.12. For the CCTV footage, a single worker's clicks are on average 87% accurate, while CrowdAR's clusters are 88% accurate, which is not a significant improvement. However, a single worker on average only recalls (i.e., identifies) the target in 71% of the frames they observe. Whereas CrowdAR, because it has many people observing the same feed, performs significantly[6] better at keeping track, recalling the target in 92% of frames.

CrowdAR's main performance improvements can be seen on more visually challenging video feeds, such as the Aerial footage. In the Aerial footage, we can see that despite the lower precision of individual crowd workers' annotations, 76%, CrowdAR's clustering algorithm is able to filter the noise, and so 91% of the reported damaged buildings are correct, significantly better than any single worker. Furthermore, while a single worker on average only identifies 5% of the damaged buildings observed, CrowdAR is able to correctly identify 11% of the buildings, a significant improvement and more than doubling the performance of a single observer. The recall values for the aerial footage are quite low compared to that of the CCTV footage. This is due to the difficulty of the dataset, while the CCTV footage has only a single target that remains mostly visible for the duration of the video. In contrast, the Aerial dataset has a large number of damaged buildings (i.e., 266), and some show little sign of damage from the air.

5.3.4.2 Responses

Here we measured the responses of workers during a video stream, and their change over time, seen in Figure 5.13. Notably, these graphs show that the number of workers

[6]All results reported as significant were confirmed with t-tests at $p < 0.05$ and all figures show 95% confidence intervals.

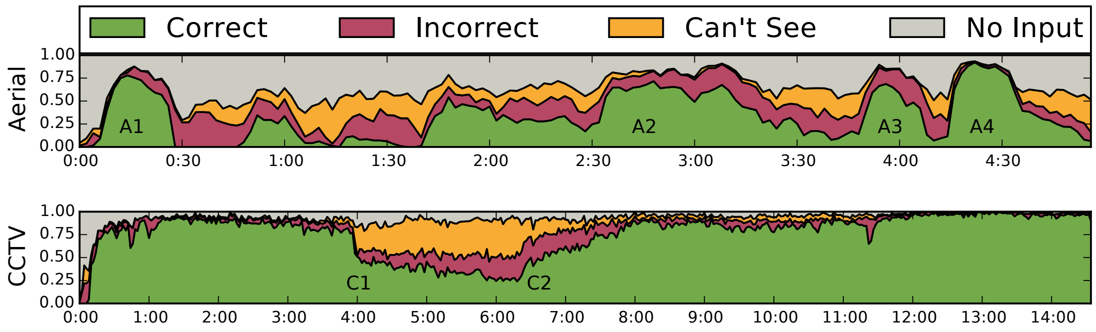


Figure 5.13: Workers' Inputs over the duration of the video feed.

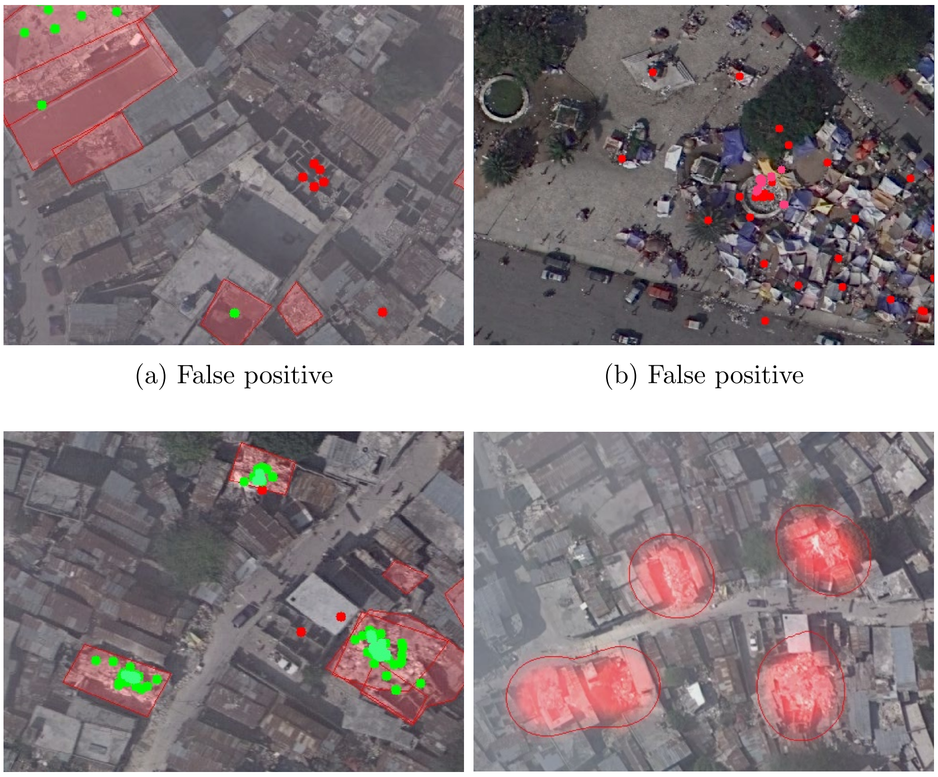
providing input corresponds to the difficulty of the search task. For example, the CCTV footage has a near constant input rate, whereas the Aerial footage is a visually difficult challenge, and we notice a significant difference in response rate. Workers are busy searching the footage, and it is more likely the countdown will timeout before the worker can provide an input.

Furthermore, for the Aerial footage we observe spikes of correct tags (A1,A3,A4) when large easy to identify buildings enter the frame, and a prolonged period of multiple damaged buildings at A2. Similarly, in the CCTV footage we observe a sudden drop in correct clicks at C1, which corresponds to when the search target is partially occluded (Figure 5.10b). Notably, the input rate does not change, and instead many workers indicate that they cannot see the target, while others begin tagging the wrong person. However, the longer the target remains partially off-screen, the greater the confusion between targets, and the incorrect count rises. As the target slowly walks back into frame, C2, the correct annotations slowly begin to rise as well.

In the aerial footage, and part way through the CCTV footage, there is a sizable percentage of incorrect clicks. From visual inspection, these tend to arise not because of worker maliciousness, but because of honest mistakes. For example, workers incorrectly identify a different individual also wearing a sweater (seen in lower left of Figure 5.10b), or tag garden walls that may look like a missing roof, or gravel that may be similar to rubble (see Figure 5.14a and 5.14b respectively).

5.4 Summary

CrowdAR annotates more search targets, and with more precision than a single worker could achieve. This performance can help augment live video and reduce the cognitive load of the observer. This could be used to inform an expert where search targets are located, enabling more effective and timely decision making. We show from our results that tracking and clustering real-time input data from the crowd is suitable for augmenting live video. Furthermore, there are a number of future applications for these


(a) False positive (b) False positive

(c) True Positive (d) Augmented Frame

Figure 5.14: Examples of true and false positives, worker clicks are colored in green (correct) or red (incorrect). In (d) we can see an example of the augmented output, with identified clusters colored randomly.

annotations. For example, providing input to an autonomous path planner onboard a UAV, or in filtering large segments of video with little interest.

CrowdAR could be improved in a number of ways. For example, there are other techniques for tracking workers' tags in a video, such as TLD, which could be used to detect and learn features on, or around the tag, and look for similar features in the next frames (Kalal et al., 2012; Dalal and Triggs, 2005). This may improve tracking in scenarios with occlusion, and improve the precision of CrowdAR. However, reliable tracking of arbitrary targets is an area of active research, and would add more complexity to CrowdAR in order to detect and remove false positives when the tracking fails.

Additionally, the interface speed could be varied per worker, depending on a specific workers' latency and reaction times. This would result in more effective utilisation of workers, providing CrowdAR with data in a more timely manner.

Furthermore, we can see from the difference in worker response rates for easy (CCTV) and difficult (Aerial) footage, that this could be used to represent a measure of confidence from the worker, which could be used to filter noise, further improving precision. Likewise, distributing smaller subsections of the footage amongst the crowd would reduce the visual difficulty of the search task, and thus increase the recall.

In the next chapter, we recruit experienced volunteers for similar situational awareness tasks, as we look in more detail at applying human-intelligent sensing and crowd robotics to the domain of humanitarian aid.

Chapter 6

Application of Crowd Robotics to Humanitarian Aid

In order to evaluate the real-time crowdsourcing system, specifically for the use of human-intelligent sensing applied to live aerial video for humanitarian aid, we conducted a study using CrowdAR with volunteers recruited from MicroMappers. We aimed to discover the key requirements and difficulties to implementing such a system when used by volunteers in a humanitarian response scenario.

In Section 2.2.1 we raised the need to understand the volunteers, what motivates them, and how these volunteers would employ real-time systems for tagging live imagery. We wish to address the concerns of the workers, understand how they interact with live systems, and raise the ethical implications of these interactions.

In the previous chapter we outlined the architecture of CrowdAR, and evaluated it on workers from Amazon Mechanical Turk. As mentioned in 5.2.1, we expect that volunteer usage of this or similar systems would be high. Through close collaboration with core team members of volunteer organisations, the Standby Task Force and MicroMappers, we gathered requirements and iterated on designs for the tutorial and live tagging interface, integrating CrowdAR into the MicroMappers system. The following chapter revolves around this collaboration, and details the insights discovered though their interaction, providing valuable knowledge to improve future real-time crowdsourcing systems or real-time humanitarian volunteering efforts.

6.1 The Standby Task Force and MicroMappers

The Standby Task Force is a global network of volunteers trained and ready to collaborate online in the immediate aftermath of a natural disaster. They refer to themselves as ‘digital humanitarians‘ and provide online support to humanitarian efforts across the world. All members must agree to follow a code of conduct (available on their website), that essentially requires volunteers to treat all human life as equal, to remain neutral and never discriminate, to strive to verify information before sharing or acting on it. They coordinate via a web platform, self-organising into groups, and communicating via Skype.[1] The humanitarian tasks are broken down into a number of team responsibilities, (task team, geo-location team, media monitoring, SMS, translation, etc). Volunteers self-select which team is of most interest to them, with each team having 1-3 coordinators and its own Skype chat to coordinate. Notably, SBTF actively seeks to improve and extend volunteer skill sets through training, including working and experimenting with emerging technologies. The SBTF delivers results by using a variety of online tools, including tools developed by MicroMappers.

MicroMappers is a website developed by QCRI, in partnership with the United Nations, and the Standby Task Force. MicroMappers contains a collection of crowdsourcing apps, known as ‘Clickers’, specifically designed for digital humanitarian response. The purpose of these apps is to quickly make sense of the masses of user generated multimedia content posted on social media sites during disasters. MicroMappers does this through the use of microtasking, making it as simple and as easy as a single click of the mouse for anyone at home to become a digital humanitarian too.

6.2 Participants and Conditions

All participants were volunteers from The Standby Task Force and MicroMappers, and were not otherwise affiliated with this project. Participants were recruited via a blog post on the MicroMappers website. The post explained to the participants that we intended to run a trial of an experimental system, that they would be annotating simulated live footage, that the footage would be a prerecorded feed and not during an actual disaster.

Through discussions with the volunteer organisations we decided upon the time and date of a live deployment of the CrowdAR system. We choose a weekend, a date when the two main demographics of the MicroMappers crowd, Europe and North America, were most likely available, and multiple timeslots in the afternoon were chosen, in two timezones containing the two largest demographics. Once the date was agreed upon, a blog post was written to advertise the trial to the members of the Standby Task Force and MicroMappers, and volunteers registered their interest in participating. We received 145 participants registering for one of four timeslots, 3pm and 4pm British time, and 3pm and 4pm Eastern time.

Emails reminding participants of the task were then sent out 30 minutes before their registered timeslot. Subsequently, 84 participants (58%) returned to perform the task

[1] `www.skype.com`

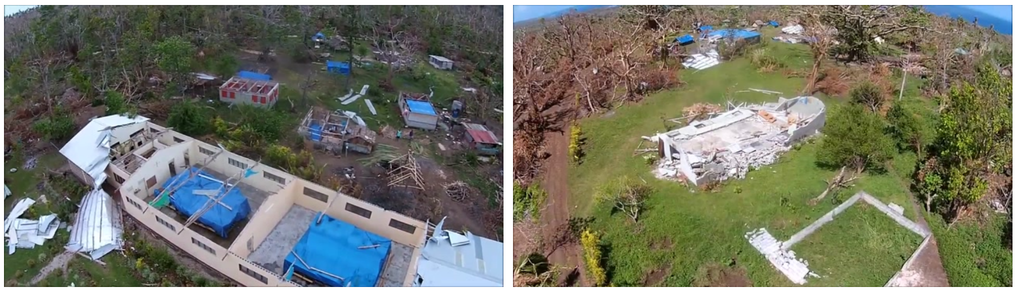

(a) Video A (b) Video B

Figure 6.1: Example footage from the two videos shown to the MicroMapper volunteers

at their registered timeslot. The live broadcast actually started 5 minutes after the advertised timeslot, allowing for some leeway for late arrivals and for participants to familiarise themselves with the tutorial.

Each timeslot showed two videos, one directly after the other. These videos were sourced from UAV flights in Vanuatu, during the aftermath of Cyclone Pam[2]. Both videos show comparable amounts of damage, in similar scenarios (see Figure 6.1). We chose these videos, instead of the generated high altitude footage (Section 5.3.1.2), as it became clear that low altitude, oblique imagery provides clearer easier to identify footage, and is representative of the type of imagery the volunteering crowd typically deals with. The downside to using this footage is the lack of ground truth available, so although we cannot measure their performance, we are instead observing how the volunteers make use of a real-time crowdsourcing system.

6.3 Worker Feedback

In this section, we evaluate both paid (Chapter 5) and volunteer (Chapter 6) workers' perceptions of the real-time crowdsourcing system. We presented the paid workers with an optional feedback form. The form asked a series of questions to gauge how the volunteers felt about the system, whether they believed the payment was fair, how long they would wish to wait in a retainer waiting room, and whether they would participate again in the future. Over 60% of workers replied to the feedback form, many of whom express similar feelings. For the volunteer workers, we conducted both an optional feedback form and semi-structured interviews. First, a survey was emailed to all volunteers after the trial, and over half the volunteers filled out the survey (45). The survey asked a series of questions to gauge how the volunteers felt about the system, what they believed it was trying to achieve, to suggest improvements, and to request permission to contact later for phone interviews. Following on from this, we conducted semi-structured interviews

[2]Anonymised videos of the annotated live footage can be seen here:
VideoA: `https://vimeo.com/140445338`
VideoB: `https://vimeo.com/140445339`

with five participants. The survey and interviews were transcribed and coded by two researchers independently.

Our results generally show that the system worked well. The workers from both paid and volunteer crowds had a high work rate, providing input on average once every 2.2 seconds (depending on the number of workers connected, this amounted to nearly 12 tags/second, and 49,000 tags over 4 runs). Many of the worker's report the ease of use of the system (e.g., *"It was simple to use"*, *"It's easy and kind of fun!"*), suggesting that they enjoyed using the system. In what follows, we describe the feedback from the survey and interviews.

6.3.1 Interface Concerns

In this section, we investigate the usability of the current state-of-the-art real-time crowdsourcing tools designed for annotating live video, and to define the challenges that need to be addressed when building future real-time crowdsourcing tools. In order to achieve this, we need to understand how workers, especially online humanitarian volunteers, would use such a system, and what concerns they have with current interfaces. We asked participants to provide general feedback about the system and interface they used.

Participants often stated that the speed at which the image refreshed was too fast (2s) – about 22 out of 45 respondents. Despite this, experimentation has shown this speed gives a good trade off between high worker accuracy and the low latency crucial to augmenting live video (see Section 5.1). However, those results do not consider the stress of the humans involved. Although, a number of users also reported that within a few minutes, they got used to the speed:

> *"I improved my work after some clicks. in the beginning, it was little bit difficult, but after that, I could do it perfectly."*
>
> *"It was also very intense, on the first video I had to take a quick break, I felt rushed and nervous, but when I came back I felt more prepared and got into the swing of it."*
>
> *"The simplistic user-interface is a delight to use. At first, I thought the reaction time was too short, but when you settle in, it becomes more manageable."*

Additionally, the speed of the interface comes with some further downsides. Some complained that when clicking at the end of the frame's countdown, the click would land on the next frame, resulting in an incorrect click somewhere irrelevant on the next image. This seems to frustrate them:

> *"It was a little nerve wracking. I felt that the video moved slowly, but sometimes I would click too late and I would be clicking on a new screen."*

> *"May have overclicked, as I wasn't sure how often we were supposed to mark things. Rapid shifts in view from the UAV sometimes meant that my click fell outside of a building I was trying to flag."*

Another commented on the "I cannot see Damaged Buildings" button. They show clear understanding of its purpose, but suggest that it is not a feasible input method given the fast nature of real-time annotation tasks:

> *"It worked quite well. It runs very fast though and it is easy to miss clicking the "no damage" button if there is a lot too look at. It becomes a "I didn't see anything in 2 seconds button" which isn't the same thing at all. I understand the necessity – so you know if someone is actually doing the job or just letting the task run unattended."*

Participants were asked to click on any damaged building that they saw in a frame. Due to the annotations timing out (Section 5.2.3.2), and large buildings requiring multiple annotations, workers were asked to click multiple times along the length of the building, and on subsequent frames:

> *"It was hard to know how many times to click on the damaged buildings."*

> *"Sometimes I just felt that I was screwing everything by clicking too often at the same building and having no sufficient time to click everything or just to evaluate properly on what to click."*

Further along these lines, users were not clear about whether they should click on the outline or at the centre of a building. Some clearly misread the tutorial and did not realise the boxes drawn on the demo were not going to appear in the real shots:

> *"The tutorial made it seem like boxes could be created around the buildings. I could only click a single point. Wasn't sure if I should click around the edges of a structure and if I should click on the same structure in each new frame."*

> *"I wasn't exactly sure, you know, if I should click the boundaries, if I just needed to click around them."*

In some cases, users reported using specific strategies to click on buildings, as per their own understanding of the task:

> *"In most cases, yes, I just tried to get, you know, if there was four or five corners that would make, say, a polygon I would surround it."*

In addition, some participants pointed to the fact that the system did not give sufficient feedback whether they were correctly tagging buildings:

> *"It was good, but difficult to know if it matters where you clicked on it."*

> *"To many people, many developing nation buildings look damaged, so at some points I wanted to click on everything. The key distinctions weren't there, nor was I getting responses from the system helping me gauge that difference."*

Finally, a number of volunteers felt that the interface should give greater control to the user, explicitly stating that they wanted the ability to focus or zoom the image. These controls are commonly available on the mapping tools (Lin et al., 2013) used by online humanitarian volunteers:

> *"I would rather be in control of positioning the pictures. That constant movement threw me off. I felt like I was being rushed."*

> *"It will really help if there are controls to zoom in on a target, because sometimes, it is hard to differentiate debris from buildings from an altitude."*

Likewise, users felt that they could provide more fine-grained input to the system, they believed that simply clicking was not good enough for the work of emergency responders. For example, one user mentioned the difference in top-down and oblique imagery:

> *"You can detect damage/no damage from satellite photos. Maybe the UAV project could focus on details that the others can't? Instead of asking for damage/no damage - maybe one could ask for what kind of damage? Phone/lines, trees, roads, roof, wall, broken windows? Just a thought..."*

In summary, volunteers felt pressured by the speed of the interface. They felt a need for more feedback, the system should inform workers when and where their annotations need updating, if, for example, they time-out or lose tracking. Likewise, workers want positive reinforcement, feedback that informs them when they have correctly identified a building (i.e., verified by multiple workers). Finally, volunteers wanted an interface that gives them more refined controls to help identify damage (e.g., zooming and panning) and to provide more fine grained input (e.g., classifying the type of damage).

These concerns would suggest that current live video tagging interfaces are not yet intuitive enough for untrained crowds. The concerns raised by both paid and volunteer workers, and the desires for changes to be made to the user interface present challenges to be addressed by future real-time video annotation software, and real time crowdsourcing tools. We suggest some potential solutions to these challenges in Section 6.5.1.

6.3.2 Desire for more Context

Many volunteers expressed a broad range of elements that they would have liked to know about before starting the task. For example, they wanted to know the purpose of the task and the geographical and social settings within which the task was being executed, and how their input is processed afterwards. For example, one user reported that they would have clicked on targets they were less confident about, had they known it would be reanalysed by an expert afterwards.

> *"If you're asking people to choose places and there's going to be immediate action upon those choices, to take a sort of extreme example of people under rubble, then one might be rather cautious of the features that one picks out. Whereas, if you know it's going to be sort of combined and reanalysed, one might be generous."*

Moreover, some users reported that they were unclear about how accurate they should be when clicking on buildings and this led them to performing worse in their view. For example, one user reported that:

> *"It was unclear at what level of certainty we should click. I tended to assume reasonable likelihood. If I had known that someone else was going to be re-examining it, I might have clicked on cases in which I had greater doubt."*

A number of users reported that they found it hard to identify damaged buildings because they were not sure what damaged buildings would look like in different countries. In the Vanuatu videos, a blue tarpaulin would be mistakenly tagged as a damaged building as a result:

> *"Also, what did the blue tarps mean? Was that just standard construction or an indication of a building that had been damaged and was being covered up"*

This suggests that at times the instructions given by those requesting live video annotation (i.e., click on damaged buildings) may be vague or under specified to the workers, especially given the dynamic nature and unknown environments that disaster response works in. Workers may require a two way communication with the task requester to clarify exactly what needs tagging (e.g., "Should I tag the tarp?").

Some participants found it difficult to identify damaged buildings because they were not sure what they would look like in different countries. However, experienced volunteers, participating in earlier mapping activities for the same disasters, found it easy to identify damaged structures:

> *"I found it somewhat difficult to assess whether building were actually damaged or whether they were shacks or temporary structures that looked like damaged buildings."*

> *"I volunteer for SBTF and tagged a lot of pictures during [Cyclone] Pam, so the images were familiar to me. It was easy to spot damaged buildings."*

> *"You have straw houses, the kind of structures you're not used to if you live in the western world, for instance. So you don't really know, is this a shed, a house, what is it? What is it that I'm seeing, because you are not familiar with that. But, I was like: 'Okay, I've seen this before. This is a straw house that's collapsed."'*

Finally, a number of users had different understandings and questions about 'who' would be using the data that was being generated. Some volunteers believed the annotations were going to be used to train an algorithm. While another mistook the behaviour of the UAV as being controlled by the clicks. While these are not the case at present, they are intended directions of future work. With further testing and safeguards in place, hybrid AI/crowdsourced intelligence systems could help automate search and rescue flights. The volunteers in this study were recruited from a blog post, and similar posts on the blog talk about this kind of technological advancement, so some bias here is possible:

> *"It was trying to collect training data for potentially an algorithm that would look for damaged buildings."*

> *"As you clicked on a damaged building it seemed like the drone started focusing and circling the object. This way you started getting different views of same object and were able to confirm your assessment of the damage."*

Others believed (as intended) that the output was going to be used to provide a rapid overview of damage to NGOs, governments and media:

> *"The tool could also help governments understand the extent of the damage and media report on accurate and specific information."*

In summary, volunteers desire more information about how their annotations will be processed and used afterwards, and how uncertain they can be before providing a tag. They desire the ability to ask clarifying questions to the humanitarian organisation they're tagging for. These issues can be solved with more communication between volunteers themselves, and between volunteers and the humanitarian organisations requesting the work. Finally, they have shown how training is of importance to the volunteers, enabling them to more easily understand the context they're working in (e.g., identifying damaged buildings). Next, we consider the stressful and emotional impact on volunteers using such tools, and what practices can be put in place to alleviate these concerns.

6.3.3 Ethical Concerns of Real-time Crowdsourcing

Real-time crowdsourcing in disaster response raises some significant ethical challenges. These challenges were raised by some of the participants' feedback. For example, one volunteer expressed the need to ensure that they are working for a good cause, as it is easy to envisage a scenario where providing real-time information to disreputable organisations could cause more harm than good:

> *"Because data is power, I guess, and you want to make sure that what you're doing, and what you're contributing to is good, is doing good, and that you've been working for the right causes."*

Furthermore, another of the SBTF participants who had previously experienced working on classifying satellite and aerial imagery was particularly sensitive to real-time UAV imagery, given that it is of a higher resolution and live:

> *"We're dealing with people in vulnerable situations in catastrophe, and I don't know how I feel about it, like if it was a live, unfolding event of something terrible, that I had to watch and tag"*

In this case, the participant clearly highlights the exercise as being stressful, particularly when the victims' suffering will be clearly visible. Their concerns echo recent issues raised by the Red Cross about the rights of disaster victims and those of the online volunteers (Ausbrook, 2015), and is similar to the responses seen in previous studies (Starbird and Palen, 2011; Palen and Vieweg, 2008) (See Section 2.2.1).

There are two distinct issues that need addressing here. First, there seems to be a need to balance the value of collecting live imagery during disasters against the privacy and dignity of disaster victims. Second, online workers, by signing up for humanitarian work, expose themselves to seeing images of victims, an experience which may be both stressful and emotionally difficult to handle.

6.4 Discussion

In what follows, we focus on the feedback received from both our paid and volunteer workers. We highlight the general themes that emerge from our results around the usability of the system and the challenges of running live crowdsourcing exercises with online humanitarian volunteers in general. In Section 6.5, we then use these insights to identify key design guidelines for real-time crowdsourcing platforms deployed with online humanitarian volunteers.

6.4.1 The Wisdom of the Real-Time Crowd

Despite our best efforts, collaborating with core members of the Standby Task Force, running pilot studies, and designing the tutorial with their input, there were clear issues that resulted from the tutorial and task design. For example, a number of usability issues emerged, such as their understandings of where to click, what level of accuracy was needed, and how many times they should click. However, from the levels of accuracy obtained when aggregating worker's clicks (on average, 86%, improving upon the 70% individual accuracy of the Workers), and the redundancy due to multiple inputs from different volunteers, seems to compensate for their individual errors arising due to these usability issues which is the point of crowdsourcing the exercise (i.e., the crowd being better than any individual in the system).

6.4.2 Context and Control for Digital Humanitarians

Our interviews revealed that participants felt the need for more control over, and context around, the task to perform well. Digital humanitarians have an intrinsic motivation to help save lives and they have a clear mandate to perform their tasks as accurately as possible. Controlling the image to zoom in and to pan are seen as standard tools in a mapping activity as is typically available on satellite imagery mapping tools that MicroMappers used. In addition to this, to correctly identify damaged buildings specific to an area, it is very important to train them to recognise objects they might not have seen before.

These needs raise two important issues. First, while zooming in and out of pictures may be easily implementable, controlling the flight path of a UAV from multiple crowd inputs in real-time remains a research challenge. Some solutions have been proposed Chapter 4, but these have only been validated with workers on AMT for very different problems where the goal of the worker is unique and very clear. Second, the digital humanitarian community is composed of both experienced and inexperienced volunteers. This means that some may have had experience tagging images in similar disaster zones and may not need to be given much training or context to the exercise but for those who have not seen imagery from UAVs or from similar disaster zones, they clearly need to be more informed about what the task involves, who the task is being done for and how the data is going to be used.

6.4.3 Ethics of Real-time Crowdsourcing

As mentioned by some of the participants, real-time crowdsourcing in disaster response raises some significant ethical challenges. Their concerns echo recent issues raised by the Red Cross about the rights of disaster victims and those of digital humanitarians (Ausbrook, 2015). On the one hand, there seems to be a need to balance the value of collecting live imagery during disasters against the privacy and dignity of disaster victims. On the other hand, digital humanitarians, by signing up for volunteer work, expose themselves to seeing images of victims, an experience which may be both stressful and emotionally charged given that they cannot help except by continuing to tag.

6.4.4 Working with Humanitarian Volunteers

While a number of studies exist on the methods to undertake crowdsourcing experiments, we believe ours was one of the first to try and validate a real-time crowdsourcing prototype with practitioners rather than paid workers on AMT. Digital humanitarians have intrinsic motivations to undertake trials of new systems and help improve them and this greatly helped in the recruitment process (with 84 participants turning up to run the exercise) but this also meant that they had to book a specific time to execute the mission. This contrasts with a crowd on AMT which can be called up at any time to do a test of the system. The feedback from digital humanitarians also tended to be much richer (as shown above) than what we obtain from online forums or end of HIT surveys for our AMT deployments.

6.5 Implications for Design

Based on our quantitative and qualitative analysis, we next identify key design guidelines for real-time crowdsourcing platforms that need to be deployed with digital humanitarians.

6.5.1 Crowdsourcing Real-Time Video Annotation

Our analysis shows that numerous usability challenges emerge when crowdsourcing real-time video annotation. To address these challenges, we have distilled our feedback into the following four recommendations:

1. The speed of the interface, while required for live annotation, stressed the workers. Given that different users have different reaction times and each feels the need to control what they see and what to focus on (See Recommendation 4), enforcing strict time constraints may be unfeasible. Methods to adapt the interface to the workers' capabilities, while still retaining the high throughput need to be developed. For example, intelligently selecting the frames to be annotated, thereby reducing the workload, and over time, as users become better at the task they may increase the frequency of images. This would allow for annotations with strict time constraints (from experienced workers) and more accurate, non-real-time annotations (from novice workers).

2. The input method (i.e., clicking) raised a number of problems. Even though clicking is the simplest input method for current tracking systems, it requires multiple inputs along the length of long buildings, or clicking again when the annotations timeout (Section 5.2.3.2). Intuitive methods to annotate imagery, and methods to reliably track those annotations, requiring less manual input along the length of large targets, and resistant to changes in occlusion need to be developed.

3. Workers expressed a desire for feedback informing them when annotations are correct help them gauge how well they did individually so that they may improve their performance. However, in situations where the task is ill-defined or vague (i.e., search and rescue tasks without known ground truth), the correctness of the workers' clicks is hard to measure. Giving positive reinforcement when workers' agree with others is often used in these situations (Lin et al., 2013), but in real-time settings may lead to a groupthink mentality (i.e., learning to incorrectly tag undesired features because others are).

4. Workers expressed a need for more powerful controls (e.g., zoom, and pan), these are seen as standard tools for mapping work. Workers felt they could provide more accurate results if they had greater control over the imagery they were tagging. Finally, with high resolution close up imagery, more detailed classification could be achieved. This would require new methods of input to be able to both identify the presence of damaged buildings, and to then classify the type of damage.

6.5.2 Improvements for Real-Time Crowdsourcing

The insight gained from our paid workers, specifically on the retainer and improvements that could be made to make the experience better, apply to all real-time crowdsourcing practices, and hint towards future work that could be undertaken to improve real-time crowdsourcing platforms.

1. Some workers mentioned reluctance to wait due to external commitments. By showing expected wait times, or the number of workers connected, and how many more are required by the task, may encourage workers to stay around.
2. Many workers expressed they would not wait for more than five minutes. Depending on the system, and the minimum crowd size required, this may not be possible. Future work intends to explore the possibilities of faster recruitment methods Section 7.2.
3. Workers suggested that being paid for waiting would encourage them to wait for longer, as opposed to being paid a similar lump sum regardless of waiting time. There is no evidence as of yet how successful various methods are at retaining workers. Further studies into real-time worker incentives are required.

6.5.3 Supporting Online Humanitarian Volunteers

Feedback from our volunteers revealed a number of issues that could be improved upon to provide greater support to the online humanitarian volunteers:

1. Provide clear details how the input from volunteers is used. Our work intends to keep experts within the loop, observing the annotations given by volunteers but still making the final judgement call. Understanding this process would help volunteers gauge how uncertain they can be before providing annotations. Additionally, providing workers with information about the region and the social context of the deployment they are volunteering on allows them to better judge what to tag and what to ignore (e.g., the blue tarpaulins during our deployment).
2. Provide a means of communication with the humanitarian organisations, to ask clarifying questions. In live deployments, the search requirements may change frequently, or the tagging instructions may be vague or under specified to the workers, especially given the dynamic nature and unknown environments of search and rescue operations. Enabling two way communication between workers and experts reviewing footage would help address these issues.
3. Volunteers are very keen to be trained to do their tasks. Members of the SBTF already perceive the value of training, and those volunteers with experience stated it was easier to identify damage. For interactive and cooperative tasks such as live video augmentation, training exercises with the humanitarian organisation's pilots could be organised. These deployments would train both volunteers and pilots on their understanding of the task, their ability to cooperate and communicate, and providing them with feedback on their performance.

6.5.4 Ethical Real-Time Crowdsourcing

Real-time crowdsourcing raises important privacy and ethical issues. Streaming live footage from a disaster zone without the victims' permission to hundreds of users may violate their human rights (Article 8). This could raise moral dilemmas for digital humanitarians who may be torn between trying to help affected communities but, by virtue of their overarching mission, goes against their key principles. Past work has looked at privacy screening through a slow two-stage process or applying filters, such as blurring, to the imagery shown to the crowd (Laput et al., 2015; Lasecki et al., 2015a). These techniques obscure the original image, which may affect the crowds ability to search for targets. Hence, further work is required before techniques for maintaining privacy are applicable to real-time crowdsourcing of live video, particularly in sensitive deployments, such as disasters.

Another key aspect of live video annotation is that there is very little guarantee on what users may witness. Feeds from disaster zones are likely to contain distressing images and we believe it is important to prepare users for what is to come. We propose that this can be addressed in two main ways. One way is to control who has access to the live video feed, and have advisories that explain what may be visible in the feeds when the volunteers sign-up for access. Another way would be to provide community and psychological support to the volunteers, which they could freely participate in to discuss their concerns if they are affected by any of the imagery.

Furthermore, real-time crowdsourcing deployments may allow anyone with an internet connection to access live footage from disaster zones at zero cost. This opens the initiative to exploitation by less well-meaning individuals who may want to sell this information to the media, post it to social media to raise their own profile, to gather foreign intelligence, or even to provide false information in a way that suits them before the full scale of the event is known (e.g. during terrorism). Such pernicious uses of real-time data may be preventable to some degree (e.g., through copyright law and the data protection act, accountability, or through code of conducts (Kakaes et al., 2015)) but may be inevitable in a global context. These are problems that all crisis crowdsourcing organisations may have to face soon, given that increasing access to technology (e.g., Periscope[3], Facebook Live[4]) means that many members of the public are increasingly streaming live footage as events unfold (e.g., victims used Facebook Live during the Grenfell Tower fire (The Telegraph, 2017)).

[3] www.periscope.tv
[4] live.fb.com

Chapter 7

Conclusion

This thesis advanced the state of the art in the field of real-time crowdsourcing for crowd robotics, and specifically the techniques for utilising a real-time crowd for either direct control of a robotic agent or for locational sensing useful for situational awareness applications, such as analysing a live video feed from a robotic agent. We presented the motivation behind this research, that it is beneficial to improve the situational awareness of search and rescue operations taking place at the scenes of disasters. The challenges of real-time crowdsourced robotics were then described, and the difficulties involved in using real-time crowds for robotic control. The research objective was identified as the development of techniques for utilising a real-time crowd to influence a robotic search and rescue operation.

7.1 Summary and Key Results

Chapter 2 reviewed the literature in various fields and evaluated it in the context of crowd robotics. We first detailed the current state of crowdsourcing in humanitarian settings, and the use of autonomous vehicles in disaster response. Then, we discussed the new field of real-time crowdsourcing and how it applies to crowd robotics. Finally, we discussed aggregation methods for combining the crowdsourced inputs, disregarding methods that will not work within our real-time constraints.

In Chapter 3 we developed a tool for real-time recruitment that was used throughout the work in this thesis. The tool helps to manage and maintain a real-time crowd of simultaneously connected workers. The tool helps to rapidly recruit workers, redirecting them to the task once enough have been recruited, maintaining the recruited crowd size, and helps pay the crowd appropriately when the task is done. We then introduced the concept of a recruitment agent, that is responsible for intelligently recruiting workers, dynamically posting HITs to the marketplace depending a number of external factors (e.g., time of day, decreased supply of workers) or internal factors (e.g., task nearing completion, task requiring more workers).

Chapter 4 introduced a platform for evaluating input aggregation algorithms for control of robotic agents. We detailed the various input aggregation methods used in this benchmarking platform. We described how the *Mob* and *Leader* algorithms are implemented, previously used in Legion (Lasecki et al., 2011). Then, we introduced two novel approaches to real-time aggregation, *Real-time Majority* and *Real-time Borda*. Finally, the four approaches were empirically evaluated using a novel benchmarking system for crowd-robotics and with a hired crowd from Amazon Mechanical Turk. We showed that our first novel aggregator, *Real-Time Majority*, is useful in applications where precision is required, and that *Real-Time Borda* is a faster alternative for patrolling or surveying applications.

Chapter 5 introduced a system that acts as a real-time human-intelligent sensor for providing locational data. It can be used to rapidly map disasters or augment live video feeds with crowdsourced information. We first investigated how to build an interface that optimises human performance at the task of identifying targets in real-time, discovering the limits of human performance at varying interface speeds, target speeds, and target counts. We then detailed how the whole system was implemented and discussed the trade-offs in it's design. We evaluated the system's performance on two types of surveillance video, and showed that our aggregation techniques for real-time crowd input that is temporally and spatially distinct, out perform that of individuals in the crowd at accurately locating multiple targets.

Chapter 6 investigated volunteer users, a large demographic of our intended end users. We discuss how to support the volunteers, provifing them with more context, with a means of communication with the task requesters, and training opportunities. We also developed design guidelines for real-time locational sensing interfaces, improving upon the speed and input methods, and providing feedback to the workers. Furthermore, we provide insights on how to improve real-time crowdsourcing platforms in general, and reveal workers preferences towards waiting in a retainer, and suggest ways to manage the workers while they reside in the retainer.

7.2 Future Work

We envision three scenarios in which a crowd may influence a robot's actions. In this thesis we have investigated two of these scenarios, the ability for the crowd to perform low-level control, and for the crowd to augment a pilot's video feed. Future work intends to investigate a third scenario, a hybrid approach, in which crowd-driven human-intelligent sensors inform the decisions of path planning algorithms.

In Section 3.2, we detailed the possibilities of faster real-time recruitment methods. Real-time recruitment, without the need for human supervision, could be achieved by creating an API with which the real-time task could communicate with the recruitment agent to start and stop recruitment, and to inform the agent how many workers the task requires. Furthermore, real-time recruitment agents could reduce the costs to the requester by optimising the timing of posting HITs, the payment given to workers, and the descriptions (i.e., how the task is advertised) of the job listing on crowdsourcing platforms.

Bibliography

J. Armstrong. Combining forecasts. In J. Armstrong, editor, *Principles of Forecasting*, volume 30 of *International Series in Operations Research & Management Science*, pages 417–439. Springer US, 2001.

R. Ausbrook. Drones Can Be Used To Enforce Property Rights, 2015.

S. Bak, E. Corvee, F. Bremond, and M. Thonnat. Person re-identification using spatial covariance regions of human body parts. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 435–440, Aug 2010.

BBC News. First uk police drone unit launched in devon, cornwall and dorset, 2017.

M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger. Crowds in Two Seconds: Enabling Realtime Crowd-powered Interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 33–42, New York, NY, USA, 2011. ACM.

M. S. Bernstein, D. R. Karger, R. C. Miller, and J. Brandt. Analytic Methods for Optimizing Realtime Crowdsourcing. *CoRR*, abs/1204.2995, 2012.

M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. *Communications of the ACM*, 58(8):85–94, 2015.

J. J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh. VizWiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 333–342, New York, NY, USA, 2010. ACM.

R. T. Clemen. Comment on cooke's classical method. *Reliability Engineering & System Safety* , 93(5):760 – 765, 2008. Expert Judgement.

M. Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix.* Paris: l'Imprimerie Royale. (Reprint, 1972, Chelsea, New York), 1785.

R. M. Cooke. Experts in uncertainty: opinion and subjective probability in science, 1991.

M. L. Cummings, S. Bruni, and P. J. Mitchell. Human supervisory control challenges in network-centric operations. *Reviews of Human Factors and Ergonomics*, 6(1):34–78, 2010.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

R. D'Andrea. Can Drones Deliver? *Automation Science and Engineering, IEEE Transactions on*, 11(3):647–648, July 2014.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255, June 2009.

Drone Adventures. Fukushima - a Drones Eye View. *Drone Adventures*, April 2014. `http://droneadventures.org/fukushima`.

P. Dunphy, J. Nicholson, V. Vlachokyriakos, and P. Briggs. Crowdsourcing and CCTV : the Effect of Interface , Financial Bonus and Video Type, 2015.

P. Egglestone, D. Ansell, and C. Cook. DEMO: UAVs in Crowd Tagged Mountain Rescue. In *Proceedings of International Conference on Making Sense of Converging Media*, AcademicMindTrek '13, pages 286:286—-286:287, New York, NY, USA, 2013. ACM.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

G. Farnebäck. Two-Frame Motion Estimation Based on Polynomial Expansion. In J. Bigun and T. Gustavsson, editors, *Image Analysis*, volume 2749 of *Lecture Notes in Computer Science*, pages 363–370. Springer Berlin Heidelberg, 2003.

P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, Sept 2010.

P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.

F. Galton. Vox Populi. *Nature*, 75:450–451, 1907.

S. Goggins, C. Mascaro, and S. Mascaro. Relief Work After the 2010 Haiti Earthquake: Leadership in an Online Resource Coordination Network. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 57–66, New York, NY, USA, 2012. ACM.

K. Goldberg, B. Chen, R. Solomon, S. Bui, B. Farzin, J. Heitler, D. Poon, and G. Smith. Collaborative teleoperation via the Internet. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 2019–2024 vol.2, 2000.

M. Gordon, J. P. Bigham, and W. S. Lasecki. Legiontools: A toolkit + ui for recruiting and routing crowds to synchronous real-time tasks. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15 Adjunct, pages 81–82, 2015.

A. Graefe, J. S. Armstrong, R. J. Jones Jr, and A. G. Cuzán. Combining forecasts: An application to elections. *International Journal of Forecasting*, 30(1):43–54, 2014.

W. A. Hamilton, J. Tang, G. Venolia, K. Inkpen, J. Zillner, and D. Huang. Rivulet: Exploring participation in live events through multi-stream experiences. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video*, TVX '16, pages 31–42, New York, NY, USA, 2016. ACM.

Harvard Humanitarian Initiative. Disaster relief 2.0: The future of information sharing in humanitarian emergencies. In *Disaster Relief 2.0: The future of information sharing in humanitarian emergencies.* HHI; United Nations Foundation; OCHA; The Vodafone Foundation, 2010.

J. Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg. Aidr: Artificial intelligence for disaster response. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 159–162. International World Wide Web Conferences Steering Committee, 2014.

R. J. Jagacinski, D. W. Repperger, S. L. Ward, and M. S. Moran. A test of fitts' law with moving targets. *Human Factors*, 22(2):225–233, 1980.

K. Kakaes, F. Greenwood, M. Lippincott, S. Dosemagen, P. Meier, and S. Wich. *DRONES AND AERIAL OBSERVATION: New Technologies for Property Rights, Human Rights, and Global Development. A Primer.* New America, 2015.

Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.

A. Koriat. When are two heads better than one and why? *Science*, 336(6079):360–362, 2012.

M. Laituri and K. Kodrich. On line disaster response community: People as sensors of high magnitude disasters using internet GIS. *Sensors*, 8(5):3037–3055, 2008.

G. Laput, W. S. Lasecki, J. Wiese, R. Xiao, J. P. Bigham, and C. Harrison. Zensors: Adaptive, Rapidly Deployable, Human-Intelligent Sensor Feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1935–1944, New York, NY, USA, 2015. ACM.

W. S. Lasecki, C. D. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. Kushalnagar, and J. P. Bigham. Real-time captioning by groups of non-experts. In *In Proceedings of the Symposium on User Interface Software and Technology (UIST 2012)*, 2012.

W. S. Lasecki, M. Gordon, S. P. Dow, J. P. Bigham, D. Koutra, M. F. Jung, S. P. Dow, and J. P. Bigham. Glance: Rapidly Coding Behavioral Video with the Crowd. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 551–562, New York, NY, USA, 2014. ACM.

W. S. Lasecki, M. Gordon, J. Teevan, E. Kamar, and J. P. Bigham. Preserving privacy in crowd-powered systems. *HAIDM, AAMAS 2015*, 2015a.

W. S. Lasecki, J. Kim, N. Rafter, O. Sen, J. P. Bigham, and M. S. Bernstein. Apparition: Crowdsourced user interfaces that come to life as you sketch them. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1925–1934, 2015b.

W. S. Lasecki, K. I. Murray, S. White, R. C. Miller, and J. P. Bigham. Real-time Crowd Control of Existing Interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 23–32, New York, NY, USA, 2011. ACM.

W. S. Lasecki, Y. C. Song, H. Kautz, and J. P. Bigham. Real-time crowd labeling for deployable activity recognition. *Proceedings of the 2013 conference on Computer supported cooperative work - CSCW '13*, page 1203, 2013a.

W. S. Lasecki, P. Thiha, Y. Zhong, E. Brady, and J. P. Bigham. Answering Visual Questions with Conversational Crowd Assistants. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, pages 18:1–18:8, New York, NY, USA, 2013b. ACM.

W. S. Lasecki, R. Wesley, J. Nichols, A. Kulkarni, J. F. Allen, and J. P. Bigham. Chorus: A Crowd-powered Conversational Assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 151–162, New York, NY, USA, 2013c. ACM.

E. Law and L. v. Ahn. Human computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(3):1–121, 2011.

K. Lehrer and C. Wagner. *Rational Consensus in Science and Society: A Philosophical and Mathematical Study*, volume 24. D.Reidel Publishing Company, Dordrecht, 1981.

A.-M. Lin, A. Huynh, L. Barrington, and G. Lanckriet. Search and discovery through human computation. In P. Michelucci, editor, *Handbook of Human Computation*, pages 171–186. Springer New York, 2013.

C. List. The theory of judgment aggregation: an introductory review. *Synthese*, 187(1): 179–207, 2012.

S. B. Liu. Crisis crowdsourcing framework: designing strategic configurations of crowdsourcing for the emergency management domain. *Computer Supported Cooperative Work (CSCW)*, 23(4-6):389–443, 2014.

A. Lyon and E. Pacuit. The wisdom of crowds: methods of human judgement aggregation. In *Handbook of Human Computation*, pages 599–614. Springer, 2013.

I. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7(1):91–139, 1992.

J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

A. Mao, Y. Chen, K. Z. Gajos, D. C. Parkes, A. D. Procaccia, and H. Zhang. Turkserver: Enabling synchronous and longitudinal online experiments. In *Proceedings of the Workshop on Human Computation (HCOMP 2012)*, pages 33–39, 2012a.

A. Mao, W. Mason, S. Suri, and D. J. Watts. An experimental study of team size and performance on a complex task. *PloS one*, 11(4):e0153048, 2016.

A. Mao, A. D. Procaccia, and Y. Chen. Social Choice for Human Computation. In *Proceedings of the Fourth Workshop on Human Computation (HCOMP-12)*, pages 136–142, 2012b.

L. Marconi, S. Leutenegger, S. Lynen, M. Burri, R. Naldi, and C. Melchiorri. Ground and aerial robots as an aid to alpine search and rescue: Initial sherpa outcomes. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–2, Oct 2013.

P. Meier. Human computation for disaster response. In P. Michelucci, editor, *Handbook of Human Computation*, pages 95–104. Springer New York, 2013.

P. Meier. An Introduction to Humanitarian UAVs and their Many Uses. *iRevolution*, 2014a. `http://irevolution.net/2014/05/01/intro-to-humanitarian-uavs/`.

P. Meier. Debrief: UAV/Drone Search and Rescue Challenge. *iRevolution*, 2014b. `http://irevolution.net/2014/05/20/debrief-uav-search-rescue-challenge/`.

P. Meier. *Digital Humanitarians: How Big Data Is Changing the Face of Humanitarian Response*. CRC Press, 2014c.

P. Meier. Using MicroMappers to Make Sense of UAV Imagery During Disasters. *iRevolution*, 2014d. `http://irevolution.net/2014/05/29/micromappers-aerial-imagery-disasters/`.

N. Morrow, N. Mock, A. Papendieck, and N. Kocmich. Independent Evaluation of the Ushahidi Haiti Project. *Development Information Systems International*, 8, 2011.

D. W. Murphy and J. Cycon. Applications for mini VTOL UAV for law enforcement, 1999.

S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C. C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai. A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video. In *IEEE Computer Vision and Pattern Recognition*, number 2, pages 527–528, 2011.

G. Ouzounis, P. Soille, and M. Pesaresi. Rubble detection from vhr aerial imagery data using differential morphological profiles. In *Proceedings of the 34th International Symposium for Remote Sensing of the Environment*. International Center for Remote Sensing of Environment (ICRSE), 2011.

L. Palen, K. M. Anderson, G. Mark, J. Martin, D. Sicker, M. Palmer, and D. Grunwald. A Vision for Technology-mediated Support for Public Participation & Assistance in Mass Emergencies & Disasters. In *Proceedings of the 2010 ACM-BCS Visions of Computer Science Conference*, ACM-BCS '10, pages 8:1—-8:12, Swinton, UK, UK, 2010. British Computer Society.

L. Palen and S. Vieweg. The Emergence of Online Widescale Interaction in Unexpected Events: Assistance, Alliance & Retreat. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, CSCW '08, pages 117–126, New York, NY, USA, 2008. ACM.

T. K. Quon and J. A. Laube. Do faster rescues save more lives. *Risk Analysis*, 11(2): 291–301, 1991.

A. J. Rego. National Disaster Management Information Systems & Networks : An Asian Overview. *GDIN 2001*, 2001.

Rescue Global. Technical and communication capabilities. Technical report, Rescue Global, 2013.

B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

D. Saari. *Basic geometry of voting*, volume 12. Springer, 1995.

T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 851–860, New York, NY, USA, 2010. ACM.

E. Salisbury, S. Stein, and S. Ramchurn. CrowdAR: Augmenting Live Video with a Real-Time Crowd. In *Conference on Human Computation & Crowdsourcing (HCOMP 2015)*, November 2015a.

E. Salisbury, S. Stein, and S. Ramchurn. Real-time Opinion Aggregation Methods for Crowd Robotics. In *Autonomous Agents and Multiagent Systems (AAMAS 2015)*, May 2015b.

E. Salisbury, S. Stein, and S. Ramchurn. Crowdar: a live video annotation tool for rapid mapping. *Procedia Engineering*, 159:89–93, 2016.

D. Schulz, W. Burgard, D. Fox, S. Thrun, and A. B. Cremers. Web interfaces for mobile robots in public places. *Robotics Automation Magazine, IEEE*, 7(1):48–56, March 2000.

I. Shklovski, L. Palen, and J. Sutton. Finding Community Through Information and Communication Technology in Disaster Response. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, CSCW '08, pages 127–136, New York, NY, USA, 2008. ACM.

K. Starbird. Delivering Patients to Sacré Coeur: Collective Intelligence in Digital Volunteer Communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 801–810, New York, NY, USA, 2013. ACM.

K. Starbird and L. Palen. "Voluntweeters": Self-organizing by Digital Volunteers in Times of Crisis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1071–1080, New York, NY, USA, 2011. ACM.

N. Stefanovitch, A. Alshamsi, M. Cebrian, and I. Rahwan. Error and attack tolerance of collective problem solving: The DARPA Shredder Challenge. *EPJ Data Science*, 3 (1):13, 2014.

W. T. L. Teacy, S. Julier, R. D. Nardi, A. Rogers, and N. R. Jennings. Observation modelling for vision-based target search by unmanned aerial vehicles. In *14th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1607–1614, 2015.

D. Tewksbury. Crowdsourcing homeland security: The Texas virtual borderwatch and participatory citizenship. *Surveillance and Society*, 10:249–262, 2012.

The Telegraph. Watch: Facebook live video from inside grenfell tower, 2017.

L. Tran-Thanh, T. D. Huynh, A. Rosenfeld, S. D. Ramchurn, and N. R. Jennings. Budgetfix: Budget limited crowdsourcing for interdependent task allocation with quality guarantees. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 477–484, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

L. Tran-Thanh, T. D. Huynh, A. Rosenfeld, S. D. Ramchurn, and N. R. Jennings. Crowdsourcing complex workflows under budget constraints. In *AAAI*, pages 1298–1304, 2015.

D. Trottier. Crowdsourcing CCTV surveillance on the Internet. *Information, Communication & Society*, 17(5):609–626, 2014.

UNOSAT. Haiti earthquake 2010: Remote sensing based building damage assessment data, March 2010.

L. von Ahn and L. Dabbish. Labeling Images with a Computer Game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, volume 6 of *CHI '04*, pages 319–326, New York, NY, USA, 2004. ACM.

C. Vondrick, D. Patterson, and D. Ramanan. Efficiently Scaling up Crowdsourced Video Annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013.

J. Yuen, B. Russell, C. Liu, and A. Torralba. LabelMe video: Building a video database with human annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1451–1458, September 2009.

J. Ziemke. Crisis mapping: The construction of a new interdisciplinary field? *Journal of Map & Geography Libraries*, 8(2):101–117, 2012.