UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
Department of Aeronautics and Astronautics

Enhancing Autonomous Vision-Based Navigation in Space with Deep Learning Technologies

by

Benjamin Felix Guthrie
MEng
ORCiD: 0000-0001-7513-1521

A thesis for the degree of
Doctor of Philosophy

August 2022

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
Department of Aeronautics and Astronautics

Doctor of Philosophy

Enhancing Autonomous Vision-Based Navigation in Space with Deep Learning Technologies

by Benjamin Felix Guthrie

Due to the steadily increasing quantity of debris in orbit, active debris removal is widely considered to be a necessary technology; however, no such missions have yet been flown. This is largely down to the challenges faced by the guidance, navigation and control system for close-range rendezvous, and the risks involved. In this thesis, deep learning technologies are applied to the problem of autonomous visual guidance in space, with the aim of increasing the technology readiness level of active debris removal.

The key difficulty involved with applying deep learning technologies to this problem is the lack of freely available training data. Therefore, a simulation framework is constructed to generate labelled image datasets for training deep learning models. Both the datasets and software are publicly released in order to facilitate further research in this domain.

An approach for determining the attitude evolution of an unknown and uncooperative debris object is proposed, based upon a siamese convolutional neural network. The network detects and tracks inherently useful features from visual sensor data, from which the rotational state is inferred. The approach is fully end-to-end trainable, including an adapted outlier rejection algorithm, allowing for simple finetuning and application to different situations.

The method is analysed numerically using simulated data, displaying improved performance compared with state-of-the-art algorithmic approaches. It is also shown to be capable of real-time performance while generalising well to unseen targets. This approach can therefore be considered a feasible solution for safely performing guidance and navigation in active debris removal, satellite servicing and other close proximity operations around a previously unknown object in space.

Finally, this research is placed within the context of developing a fully autonomous, artificial intelligence-based on-board guidance and navigation system. Several potential further applications of deep learning to the problem are discussed, along with the remaining challenges faced by these technologies.

Contents

List of Figures

List of Tables

Declaration of Authorship

7. Parts of this work have been published as:
   Ben Guthrie, Minkwan Kim, Hodei Urrutxua, and Jonathon Hare. Image-based attitude determination of co-orbiting satellites using deep learning technologies. *Aerospace Science and Technology*, 2021c. ISSN 1270-9638. . URL https://www.sciencedirect.com/science/article/pii/S1270963821007422,
   Ben Guthrie, Minkwan Kim, Hodei Urrutxua, and Jonathon Hare. Improving autonomous guidance using machine learning technologies. In *Stardust-R – Second Global Virtual Workshop*, September 2021d,
   Ben Guthrie, Minkwan Kim, Hodei Urrutxua, and Jonathon Hare. Attitude reconstruction of an unknown co-orbiting satellite target using machine learning technologies. In *2021 AAS/AIAA Astrodynamics Specialist Conference*, July 2021b. URL https://eprints.soton.ac.uk/451305/,
   Ben Guthrie, Minkwan Kim, Hodei Urrutxua, and Jonathon Hare. Image-based attitude determination of co-orbiting satellites enhanced with deep learning technologies. In *Advances in the Astronautical Sciences*, volume 175, pages 3164–3182. American Astronautical Society, May 2021a. URL https://eprints.soton.ac.uk/451304/

Signed:.. Date:..................

Acknowledgements

This work was supported by the EPSRC Centre for Doctoral Training in Next Generation Computational Modelling Grant No. EP/L015382/1. The use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, were also critical for the completion of this work.

Many people have contributed to this PhD, either directly with suggestions and advice or indirectly by their support over the last four years. As I look back now, being so near the end, it is clearer than ever that this would not have been possible without all of you.

Firstly, I would like to thank my PhD supervisors, Minkwan Kim, Hodei Urrutxua and Jon Hare. Your advice and willingness to help at every part of the project is the main reason it has got to this stage, and your passion and interest has helped carry me through even when I was doubting what I had achieved. I am grateful to have had the opportunity to work with such a fantastic team.

Thanks also to my family and friends, particularly to the lab group – Rike, Tom, James, Johannes and Nathan – and to Tom and Ed for making the PhD more enjoyable and for showing that, no matter how bad I felt my work was going, there's always someone having an equally bad time of it. And finally, to Ania, for your endless love and support which has helped get me through these last few years.

Definitions and Abbreviations

Abbreviations

ADR	Active Debris Removal
CNN	Convolutional Neural Network
DL	Deep Learning
GNC	Guidance, Navigation and Control
LVLH	Local-Vertical-Local-Horizontal
NN	Neural Network
RANSAC	Random Sample Consensus
TRL	Technology Readiness Level
VBN	Vision-Based Navigation

Feature extraction and estimation

$\mathbf{M}_k$	Feature map
$\mathbf{p}_k$	Feature point coordinates
$\{\mathbf{p}_k\}$	Feature point set
(x, y, z)	Coordinates in the world frame
(x_v, y_v, d)	Coordinates in the camera view frame
σ	Loss weight
d	Pixel depth
f	Focal length
L	Loss term

Orbital dynamics

$\mathbf{r}$	Satellite position vector
μ	Gravitational constant
Ω	Longitude of the ascending node
ω	Argument of periapsis
a	Semimajor axis
e	Orbit eccentricity
f	True anomaly
h	Angular momentum
i	Orbit inclination
r	Orbit radius

Relative orbital motion

$\boldsymbol{\theta}$	Rotation vector
$\boldsymbol{\omega}$	Angular velocity vector
$\mathbf{e}$	Rotation axis
$\mathcal{F}_n$	Coordinate frame n
ω	Angular velocity
$\mathbf{q}$	Rotation quaternion
n	Mean motion
Q	Rotation matrix

Chapter 1

Introduction

In the past 60 years, the amount of debris in the Low Earth Orbit (LEO) has been increasing steadily (Kessler and Cour-Palais, 1978; Liou et al., 2010), thus posing a threat to current space infrastructures and future missions. Figure 1.1 illustrates this increase in debris objects, taken from ESA's recent annual space environment report (European Space Agency, 2021). In fact, the amount of debris would continue to increase even if all future space launches were to be stopped; this is known as the Kessler syndrome.

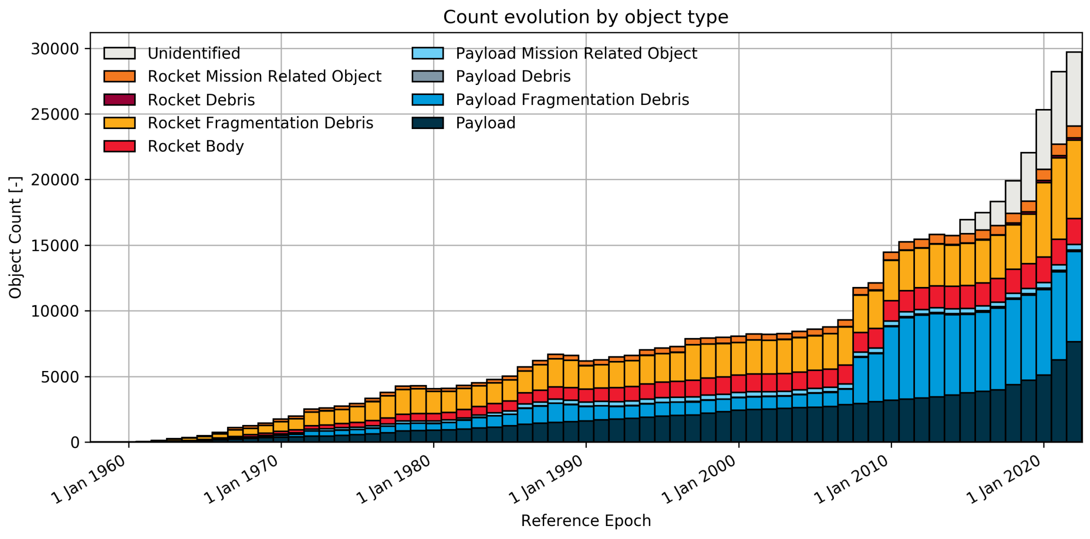


FIGURE 1.1: A count of the objects in orbit by type (European Space Agency, 2021).

Consequently, recent years have seen a concentration of efforts aimed at reducing the risks of space debris. A key focus of these has been on methods of actively removing debris objects from orbit, in an Active Debris Removal (ADR) mission (Bonnal et al., 2013; Biesbroek et al., 2017; Forshaw et al., 2016). However, most of the technologies required for these missions remain mostly unproven and at a relatively low technology readiness level (TRL), as many key challenges remain to be met before a mission can be flown. In particular, an ADR mission would pose very demanding Guidance, Navigation and Control (GNC) requirements to guarantee the safety of close proximity operations near the debris, which may include docking or berthing with it.

GNC for in-orbit proximity operations and rendezvous is intrinsically challenging from a technological viewpoint, due to the danger of collisions, high reactivity required and the difficult lighting conditions for the visual navigation system. In addition, autonomous approaches need to be developed to enable robotic missions, due to the high latencies involved with communication with the ground. It is critical that proximity operations do not result in a collision, thereby adding to the space debris problem. Furthermore, many in-orbit debris removal solutions require a limited angular rate or the identification of a suitable grappling location on the target. Therefore, an ADR spacecraft must be capable of accurately and robustly determining the relative position and rotational state of the target object, which may be particularly difficult in situations where the target geometry, physical integrity, angular velocity and rotation axis may be a priori unknown. Indeed, even if the target object is tumbling and partially fragmented, thus rendering any pre-loaded geometrical characterisation of the object useless, GNC autonomous algorithms must still be capable of reacting to such situations and enable the ADR mission to continue.

1.1 Background to Active Debris Removal

The key goal of an ADR mission is to remove a debris object from orbit, either by raising it to a graveyard orbit or leading it to a controlled re-entry trajectory. In the majority of cases, the approach consists of first capturing the debris object (henceforth referred to as the target) by a spacecraft (the chaser), prior to removal. This is the most commonly proposed approach to debris removal, although there has been research into contactless removal methods, which will also be discussed.

A key limitation of many ADR methods is the need to correctly determine the characteristics of the target prior to removal. Due to the wide range of types of space debris and the limited knowledge of their characteristics, it is not possible to use a single debris removal method for all targets. Shan et al. (2016) have categorised space debris objects into four groups by their non-cooperativeness, and suggest possible capturing and removal methods for each category. The conclusions are shown in Table 1.1.

Capture of a debris can be achieved for example by stiff-connection capturing methods, such as using tentacles (Chiesa et al., 2015; McMahan et al., 2006) or robotic arms (Reintsema et al., 2010). Stiff-connection capturing methods have several advantages. They are easy to test on the ground, as the same dynamics can be imitated

Categories	**Physical properties**	**Docking interface**	**Characteristics**	**Examples**	**Capturing methods**	**Removal methods without capturing**
A	Known	Extant	Fully known	Dysfunctional satellite	All	DAS/ contactless/ contact
B	Known	Non-existent	Compatibility-restricted	Rocket stages	Tentacles/ net capturing/ Harpoon	DAS/ contactless/ contact
C	Unknown	Extant	Dockable and unpredictable	Foreign satellite	All	DAS/ contactless/ contact
D	Unknown	Non-existent	Opaque	Fragmentation debris	Net capturing	DAS/ contactless/ slingshot

TABLE 1.1: Categories of non-cooperativeness and their associated capturing and removal methods (Shan et al., 2016).

in a non-space environment; indeed, the German Aerospace Centre (DLR) has developed a ground-based simulator to test the dynamic behaviour of a robotic arm during docking, called the European Proximity Operations Simulator (EPOS) (Zebenay et al., 2012). These methods have a higher Technology Readiness Level (TRL) than other methods, as they have already been applied in on-orbit servicing missions (Flores-Abad et al., 2014). However, these methods have high GNC requirements due to the necessity for rendezvous and docking with a target which could be tumbling.

An alternative option is flexible-connection capturing, e.g. using a net (Starke et al., 2010; Biesbroek et al., 2017) or harpoon (Dudziak et al., 2015). Net capturing has reduced requirements on precision and allows for a large capturing distance. As such, there is no need for rendezvous and docking. However, the tumbling capability is not yet known – the net could be twined by a high tumbling angular velocity – and the technologies are more difficult to test on the ground, since the mechanics of the system is heavily dependent on the physical conditions in orbit. Besides this, the collision between the net and target may generate more debris if inappropriate contact is made. There are therefore several research areas which still need to be developed for this technology: net modelling and material choice (Benvenuto et al., 2015; Benvenuto and Carta, 2013); simulation of the impact influence (Benvenuto and Lavagna, 2013); the deployment process, e.g. using a net gun (Bischof, 2003); and investigation of the tumbling capability. ROGER is a project sponsored by the European Space Agency (ESA) which aims to use a net capturing system for debris removal (Starke et al., 2010).

Removal of the target from orbit can be performed after capture or, in some cases, without capturing the target at all. Examples include space environment-based methods, such as drag augmentation (Ruggiero et al., 2011) or the Electro-Dynamic Tether (EDT) (Nishida et al., 2009). Alternative methods have been proposed based on contactless removal, e.g. ground-based laser systems (Phipps, 2014) or the Ion Beam Shepherd (IBS) (Bombardelli et al., 2013), and contact removal, e.g. slingshot (Missel and Mortari, 2013) or adhesive (Trentlage and Stoll, 2015) methods. Since there is no need to capture the target in the majority of these methods, the GNC requirements are lower on the whole, except for IBS because it requires maintaining a distance of 10-20m throughout the removal process. However, contactless and space environment-based methods tend to take a very long time to remove objects compared to capture and removal approaches, and are limited to a small range of masses of targets.

1.1.1 An Active Debris Removal Mission

Active debris removal missions consist of several stages (Kervendal et al., 2013). Figure 1.2 demonstrates a typical mission structure for a capture-based removal approach. The on-board guidance, navigation and control system must be capable of performing navigation during all stages of the mission. However, the mission stages pose different challenges for the GNC system; thus, separate navigation methods will be required at different stages throughout the mission.

Following launch and target selection, the mission begins with a phasing stage, in which the orbital parameters of the target's orbit are determined and the plane of the chaser orbit is aligned with the target. The orbital characteristics can be determined off-board, for example using a ground-based radar system (Mehrholz et al., 2002). The phasing ends with the acquisition of an aim point which is sufficiently close to the target to enable the final part of the approach. There is no particular need for autonomy in the target selection and phasing stages.

The approach stage typically begins at a distance of around 2km from the target and involves far-range rendezvous (Fehse, 2003). At this stage, the navigation system switches from absolute to relative navigation, using relative measurements of range and direction between the target and chaser satellites. These measurements could be acquired from radar, LiDAR or relative GPS (RGPS) sensors. The main objective of the far-range rendezvous stage is to achieve the required relative position, velocity and angular rate conditions to enable close-range rendezvous operations.

Differing from typical rendezvous missions, the characteristics of a debris target tend to not be fully known a priori. Therefore, the approach is followed by a fly around/inspection stage. During this stage, the rotation rate and rotation axis of the

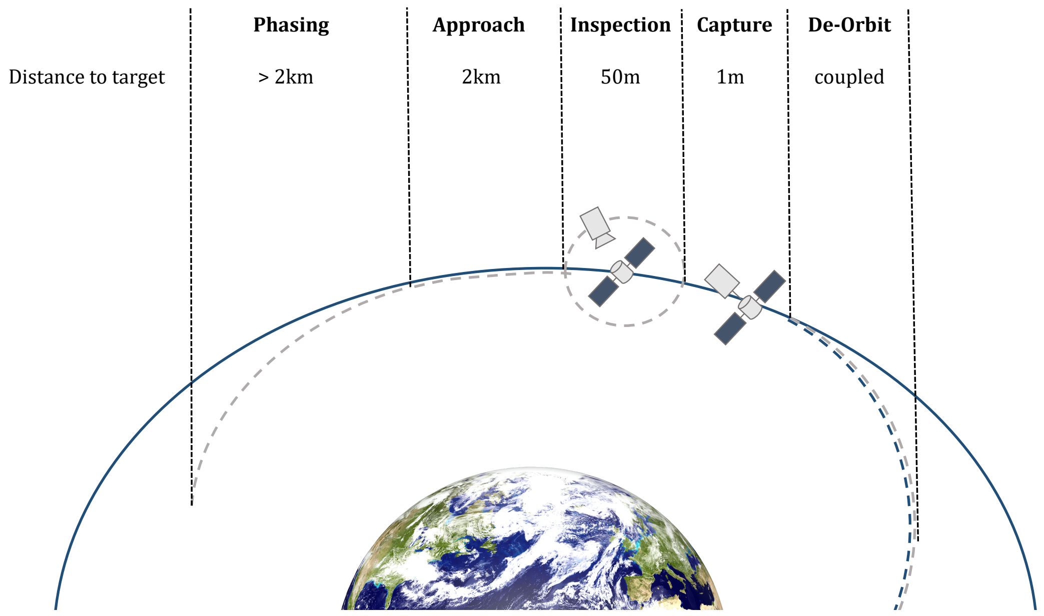


FIGURE 1.2: Stages of an active debris removal mission.

target are estimated, along with, if required, the mass, inertia or even the reconstruction of a 3D model of its geometry. Depending on the capture method, it may also be necessary to identify a location for grasping the debris. During the inspection, the chaser spacecraft will be placed in a stable orbit around the target, such as the "football" orbit detailed in Section 2.1.2.3.

Once the necessary information has been gathered by the chaser, the target is captured by the chaser. The specific operation of this phase depends on the capturing method used; several of the proposed options have been detailed previously. In general, this stage involves close-range rendezvous, aligning the target and chaser within the required range, range rate and angular rate, and possibly docking with it. In this stage the navigation system should be robust, autonomous and avoid any risk of collisions.

The final stage is de-orbitation of the debris. This consists firstly of stabilisation of the new chaser-target system (McKnight et al., 2013), including detumbling and attitude control. Once stabilised, the coupled system is navigated to either a graveyard or re-entry orbit.

The above mission plan considers a single shot chaser removing a single debris object. This would be useful as an initial demonstration mission but would most likely lead to an excessive mission cost for a single debris. Thus, Bonnal et al. (2013) list several potential schemes for removing multiple debris during one mission. These include: using a single chaser to deorbit several debris, which will have a very large δV budget; a large chaser delivering deorbiting kits to the debris; or multiple chasers each carrying a number of deorbiting kits. In order to determine the feasibility of each solution it is necessary to consider the trade-off between the size of the launcher required and the cost of the chaser functions. However, the cost is as yet very difficult to analyse and indeed it may reduce significantly in the future, since a high production rate of small chasers and autonomous deorbiting kits could lead to much lower costs for multi-target missions.

1.1.1.1 Proposed Missions

While no space debris has yet been removed, there are several missions which aim to demonstrate the technologies in use. ESA's CleanSpace project has proposed the e.Deorbit mission which, with a projected launch in 2023, is expected to be the first in-orbit demonstration of active debris removal on a "real" target. The project aims to study the applicability of robotic arm and net technologies for debris capture, including testing of a net deployment in space (Biesbroek et al., 2017). The chosen target for the mission is ENVISAT, a dysfunctional satellite belonging to the ESA whose mission ended in 2012. Not only is ENVISAT at the top of the priority list for removal, according to a study performed by the university of Braunschweig (Braun et al., 2013), but it will also serve as an effective benchmark target for future ADR missions. However, one of the challenges faced by the e.Deorbit team is that there will be no in-orbit demonstration on a smaller debris prior to the planned mission launch. Instead, it will rely on computer simulations as well as several ground-based or cubesat missions to test individual technologies, such as image processing algorithms and the robotic arm technology.

The RemoveDEBRIS mission from the University of Surrey (Forshaw et al., 2016) has been proposed to demonstrate some of the required technologies for ADR. A platform was launched to the International Space Station (ISS) on 2 April 2018 which consists of the main satellite as well as two cubesats as artificial debris targets. The aim of the mission is to demonstrate net capture, harpoon capture, vision-based navigation and dragsail de-orbitation, in a controlled space environment. RemoveDEBRIS has successfully demonstrated both net and harpoon capture mechanisms on an artificial target (Aglietti et al., 2020). Investigation of a vision-based navigation system has also been performed; this consisted of taking images of a cubesat and its surroundings using flash LiDAR and a colour camera. The data and imagery are then post-processed on ground and compared with GPS readings to verify the accuracy of the algorithms. These demonstration missions are invaluable for validation of the TRL of all elements of an ADR mission.

The closest example to date of a real situation comes from the Mission Extension Vehicle (MEV) by Northrop Grumman. MEV-1, launched on 9 October 2019, demonstrated the first docking with a satellite which was not built with docking in mind. Using a combination of visual, LiDAR and infrared sensors, the vehicle acquired and docked with Intelsat 901. This is a significant step towards autonomous docking; however, the existence of a known docking point (in the form of the liquid propellant engine), as well as the full knowledge of and cooperative nature of the target, mean that this technology cannot immediately be applied to the more challenging situation of capturing an uncooperative, unknown and possible tumbling debris object.

1.1.2 Guidance, Navigation and Control Requirements

For the successful capture and removal of debris, a chaser satellite must be capable of performing a number of different functions: both close- and far-range rendezvous; mechanical interfacing; control, detumbling and orientation of the debris; and de-orbitation (Bonnal et al., 2013). Therefore, these missions have very strict GNC requirements (Vasconcelos et al., 2016; Colmenarejo et al., 2013). Fehse (2003) states that the required navigation accuracy for rendezvous and mating operations is usually 1% of the range or better. According to a preliminary mission analysis conducted by Castronuovo (2011), numerical values are provided for the relative motion between chaser and target as below 1.5 m in range, 1cm/s in range rate and $4^\circ/\text{s}$ in relative angular rate. The GNC system must therefore be capable of autonomous guidance with collision avoidance within this range of the target, as well as robust and accurate determination of the target's orbit parameters. In addition, Yamamoto et al. (2019) outline the many potential technical challenges to be faced by the guidance system, including tumbling targets, lack of docking mechanisms and fuel management.

The low TRL of ADR methods is strongly linked to the need for a GNC system which is capable of conducting all of the previously defined mission stages with a high degree of autonomy. Several of the required technologies must still be validated, such as:

- Robust guidance during approach and de-orbit with an uncooperative target.
- Robust control during approach, capture, stabilisation and de-orbit.
- Identification of characteristics of an unknown, uncooperative target.
- Relative position, velocity and attitude estimation.

In this work, we shall focus on the latter two tasks, which can be categorised under Vision-Based Navigation (VBN). This is in contrast to the other tasks which fall under guidance and control. These are performed during inspection and close-range rendezvous. These tasks require the greatest degree of autonomy and robustness due to the time-sensitive nature of these mission phases. The accurate identification of the state and characteristics of a debris target is critical to enable many of the debris removal methods detailed previously.

While radar and GPS sensors are good options during the phasing and approach stages, the accuracy of these are insufficient to meet the requirements for close-range rendezvous. Instead, visual navigation sensors are generally considered to be the best choice in this case, either using passive (e.g. optical camera) or active (e.g. LiDAR) sensors. Visual sensors are capable of high accuracy but are sensitive to varying lighting conditions. Active sensors are more robust to lighting variations but tend to have higher power consumption.

It is crucial that an ADR mission does not cause further damage to the target, thereby creating more debris in the process. Therefore, for the chaser must be equipped with a collision avoidance system. In addition, in contact-based removal missions, a suitable surface or limb must be identified on the debris target for impaling, grasping or grappling. For example, solar arrays are very fragile and so contact with these surfaces must be avoided. Thus, the margin for error is very low in these missions, further increasing the requirements on the robustness of the GNC system.

1.2 Visual Navigation for Rendezvous with Tumbling Debris

A short-range rendezvous with an uncooperative and unknown target such as this has not yet been performed and induces challenges in developing a robust VBN system. As discussed, these challenges include the lack of knowledge of the target parameters, the need for matching with the target's orbit, and robust collision avoidance. Besides this, no single technology can cover the entire required functionality; the phasing, long-range and close-range rendezvous stages will all require different navigation techniques.

We therefore focus our attention on determining the 6-D pose (position and attitude) of the target to aid with rendezvous. Due to the short distances involved, we are interested in the *relative* motion between the chaser and target in orbit (Vallado and McClain, 2007). The dynamics of this relative motion is detailed in the following chapter. While the relative position and velocity can be computed using simple image processing algorithms, attitude determination is one aspect of ADR navigation which requires further improvements in order to increase the overall TRL, particularly in cases where the target is unknown and uncooperative (Kervendal et al., 2013; Pirat et al., 2017; Colmenarejo et al., 2017).

1.2.1 Previous work

Vision-based navigation appears to be the most popular proposed tool for GNC of ADR missions, though there are different opinions on which sensors are most suitable.

Kanani et al. (2012) present a method using a monocular camera which is capable of assessing the attitude and position of the target. Other methods employ stereo cameras (Rosso et al., 2013) or active 3D sensors such as LiDAR (Kervendal et al., 2013) to also visualise the depth of the object. LiDAR sensors are relatively insensitive to illumination conditions and have a wide range of working distances. However, they have a high power consumption – though this can be offset by instead using a photonic mixer device to provide depth information (Klionovska et al., 2018) – and a required minimum distance between the chaser and the target. Despite these drawbacks, the majority of research into motion estimation of an uncooperative satellite suggest using active LiDAR sensors (Opromolla et al., 2015; Rems et al., 2015; Aghili and Parsa, 2009).

A downside of many of the proposed methods is that they require a 3D model of the target, with relative navigation being performed by matching observed features with the known 3D model. In ADR missions, the target's shape will not always be fully known a priori and often could have changed or been damaged in orbit, and so this model would may to be constructed during the identification phase.

Since ADR is still an unproven technology, these methods are purely theoretical. For technologies which have been successfully demonstrated, one may look towards similar applications of VBN. Examples include asteroid navigation or on-orbit rendezvous and docking in space, or autonomous guidance of unmanned aerial vehicles on the ground.

VBN has been used in navigation around comets and asteroids, notably around the comet Churyumov-Gerasimenko in ESA's Rosetta mission (Muñoz et al., 2017). Due to the stringent navigation requirements, small size and large distances from Earth, an autonomous guidance system is required. Optical navigation algorithms are the crucial enabling technology for successful operation in this domain. Since there does not exist prior knowledge of the comet parameters and shape, this is a similar case to performing proximity operations about an unknown debris target.

The Rosetta spacecraft uses a single navigation camera with a 5° field of view and 1024×1024 resolution for visual tracking. Initially, the images from the camera are processed on the ground to identify landmarks such as craters on the surface of the comet, generating a point cloud. The landmark selection process uses L-maps, which are 3D digital models of small patches on the target body. Matching landmarks in the navigation images with those in the database allows for tracking of the spacecraft's position with respect to the comet. This information is combined with the outputs of star trackers for navigation of the Rosetta spacecraft around the comet.

Alternative methods for asteroid navigation have also been proposed. The solution of Rowell et al. (2015) consists of several stages. First, a 3D model of the asteroid shape is constructed, which is used to construct a database of landmarks. In contrast to the

L-map technique described above, landmarks are selected using a more conventional corner detector to extract features; these features are correlated with their immediate surroundings to identify strong object landmarks. The optical navigation itself is in two parts. Transient image features are tracked over successive frames to determine the trajectory of the camera from the starting point. However, this relative navigation leads to an accumulation of error; to overcome this, the filter is periodically reset by solving for the true instantaneous camera state, through observation of target landmarks of known position. This reduces the computational cost by not solving for the exact position at each timestep, while also allowing guidance to continue even when few known landmarks are observed in the images. In both cases, the initial landmark identification is performed off-line. Since ADR missions will likely have a much lower mission duration, this would be impractical. Instead, landmark identification would need to be fully automated.

Optical navigation has been used in on-orbit rendezvous missions, with potential applications in satellite servicing and refuelling. Optical navigation is being used over the traditional radar rendezvous sensors due to its higher accuracy, despite the lower operational range (Mokuno and Kawano, 2011). Several rendezvous and docking demonstrations have been conducted using these technologies, such by Orbital Express (Pinson et al., 2008) and the Demonstration of Autonomous Rendezvous Technologies (DART) (Rumford, 2003). However, in these demonstrations the target was well known and fitted with a reflector to aid the optical navigation algorithms significantly. Methods for on-orbit servicing of uncooperative satellites have been investigated (Benninghoff et al., 2014), though this is effectively identical to the ADR problem in terms of the GNC requirements, with a similar lack of empirical evidence of their applicability.

Possibly the most common example of optical navigation is in guidance of ground-based vehicles, such as self-driving cars or drones (Fuentes-Pacheco et al., 2015). However, the limited processing power available onboard spacecraft, combined with the need for highly robust yet autonomous algorithms, and the very different environment geometry and dynamics, render many terrestrial approaches invalid in space. Analysing the usage of VBN in these applications therefore has limited usefulness for our purposes.

In order to obtain information about the relative position and attitude of the target, we must be able to track features of interest on the object. In the methods of Pinson et al. (2008) and Rumford (2003), this is relatively simple due to the existence of reflectors in a known location on the target. However, for the case of an unknown, uncooperative target, the chaser must instead determine and extract features from camera images of the target; for ADR, it is desired that this process is fully automated. These features provide a means of identifying and characterising small patches in an image. The simplest method of feature extraction uses edge detectors, which detect points at which the image brightness changes sharply, often at the edges of objects (Canny, 1987). More modern feature extraction techniques include the scale-invariant feature transform (SIFT) (Lowe, 2004) and speeded up robust features (SURF) (Bay et al., 2006), which use more advanced image processing algorithms to both detect and describe local features. However, in most recent applications, neural networks have begun to be used to learn optimal features for the specific problem, as will be discussed in Chapter 4.

Tracking features between pairs of images requires finding matches between features in both images. The Kanade-Lucas-Tomasi (KLT) feature tracker is an approach which uses spatial intensity information to search for the position that yields the best match (Tomasi and Kanade, 1991). However, this approach assumes brightness consistency between images. This is not a reasonable assumption when dealing with a space environment, in which the lighting conditions change constantly due to the relative positions of the satellites, the Sun and the Earth. In contrast, in more modern approaches, the features each have an individual descriptor. In SIFT, these take the form of a vector containing information about the 16×16 neighbourhood of pixels around the feature. The features are then matched by identifying their nearest neighbours. Steps are also taken to achieve robustness against changes in rotation and lighting conditions.

Image features can provide a means of estimating the 6-D pose of an object. Assuming the coordinates of an observed feature are known on the target, then the object's state vector can be determined from a minimum of four feature locations (Faugeras, 1993). Kalman filters have been extensively used for pose estimation, due to their ability to reduce the effects of measurement noise and provide estimations in the event of a measurement failure (Markley and Crassidis, 2014). The extended Kalman filter (EKF) has been used for estimation of the parameters of unknown space objects (Dong and Zhu, 2015).

An alternative to tracking individual local features is to analyse the motion of all pixels in the image, a method referred to as optical flow (Fortun et al., 2015). This involves matching of all pixels, often by interpolating between matched features. By analysing the pixel velocities across the whole image, it would be possible to estimate the change in position and attitude of the target. However, since these methods operate on all pixels in each frame, they tend to have a large computational cost. Besides this, the motion of a light source will give rise to optical flow even if the object is stationary.

Visual information can also be used to construct a 3D model of a target object; this is useful for later tracking since it is possible to locate object landmarks in the 3D model. Generation of the object map could be achieved by the technique of structure from motion (SfM) (Hartley and Zisserman, 2004). A similar approach is simultaneous localisation and mapping (SLAM), which consists of both generating a map of the environment and locating the camera within it, and has applications in autonomous guidance of ground vehicles (Durrant-Whyte and Bailey, 2006). However, these processes are expensive in terms of both computational cost and memory (Agunbiade and Zuva, 2018). This reduces their applicability to space missions due to the available hardware, which is limited by size, cost and power consumption.

1.2.2 Our Contributions

This work focuses on the problem of estimating the rotational state of an unknown and uncooperative tumbling debris. This is an important element of the GNC system for ADR missions, since it is necessary to align the chaser with the rotation of the target prior to capture. This information is also crucial for a number of other tasks, such as matching with or constructing a 3D model, or identifying suitable grasping locations. We present a method of predicting the rotational state of a debris target from observation by a chaser using visual sensors.

Whereas the majority of approaches in literature assume a fully known target (either by matching with a 3D model of the target or by training deep learning models on single-target datasets), this work instead proposes a solution to the completely general case. No assumptions are made about the existence of prior knowledge of the target geometries; instead, the presented method is constructed to be generalisable to all types of debris object.

The prediction of the rotational state takes place during the inspection stage of the mission. The chaser observes the debris object over a short inspection period, during which the chaser is in a relative circular orbit about the target. The rotational state is thus estimated using the stream of data from onboard visual sensors over this period. Chapter 2 details the dynamics of this situation and presents the conversion from the observed rotation to the true rotation, taking into account the motion of both target and chaser, within the target-centred rotating reference frame.

The task is now formulated as the prediction of the observed rotation of an object between successive frames of image data. As mentioned in the previous section, feature extraction and matching is a popular conventional approach to this task. In this approach, the location of image features are found which may correspond to, for example, a sharp corner or edge on the object. The features are matched between images to predict the motion of the object over the small timestep. We adopt this technique, building upon it by integrating novel deep learning technologies. Chapter 4 details the construction of the feature extraction and matching network. By making use of the core properties of neural networks, this method is able to match features without the need for the complex feature descriptors used in conventional methods. In addition, we show that the neural network approach is capable of extracting features which are intrinsically more useful for the specific task on which it has been trained.

Once two sets of matched feature locations have been acquired, the target's rotation over the timestep is estimated from the motion of the features. This is solved as a root-mean-square deviation (RMSD) minimisation problem, as detailed in Chapter 5. Additionally, an outlier rejection scheme is formulated, alongside a recursive filtering method which is presented as an alternative to Kalman filters for deep learning applications. In order to enable end-to-end training of the entire network, these algorithms are all implemented within the deep learning framework.

However, deep learning approaches have a requirement for large, labelled datasets. Due to the lack of existing real-world datasets, and since there are no synthetic datasets which contain the required information, a simulation model is first constructed to generate the required data. Visualisations of the viewpoint of a chaser satellite are generated, obeying the dynamics of on-orbit relative motion described in Chapter 2, including visual and depth data. The data is labelled with not only the information required for training the rotation estimation network, but also a number of other labels with the intent of enabling further research into deep learning applications in this domain. These datasets are publicly released alongside the simulation software, in order to facilitate future research.

In addition, a short study is conducted into potential applications of deep learning and whether the technologies can be applied to solve some of the remaining problems faced by the navigation systems for active debris removal. Two potential applications have been identified as promising avenues of research. Examples of these have been implemented and analysed in brief, with the results shown in Chapter 3. The results demonstrate that object detection and image segmentation approaches can be easily applied to this problem, to aid in relative navigation or identification of grasping locations on the target. This study also illustrates the importance of the rotation estimation approach, as this is one key challenge which cannot simply be solved by applying known deep learning techniques.

To summarise, our contributions to the field are as follows. It is important to note that Chapters 2 and 3 consist largely of applying existing techniques in new areas, whereas Chapters 4 and 5 are entirely original research.

- The construction of a simulation model for generating labelled image datasets for use in training deep learning networks. The datasets and simulation model are made available publicly to facilitate further research (**Chapter 2**).
- A study into potential applications of deep learning to visual navigation in ADR missions, which have been enabled by this data (**Chapter 3**).

- The development of a novel neural network-based method for feature extraction and matching (**Chapter 4**).
- Estimation of the rotation of a debris target, with outlier rejection and filtering, embedded within a deep learning framework (**Chapter 5**).

Chapter 2

Visualisation of Relative Motion in Earth Orbit

A key difficulty involved with applying deep learning approaches to visual navigation in space is the lack of suitable large datasets of labelled image data. There is very little real on-orbit image data freely available, so in most cases synthetic datasets must be used instead. These are constructed by accurate simulation of the output of visual sensors under the dynamics of on-orbit proximity operations described previously.

Some few datasets exist which aim to tackle this problem. Sharma and D'Amico (2020) contribute the Spacecraft Pose Estimation Network (SPEED), which has since been made publicly available through a competition on pose estimation run by the European Space Agency (Kisantal et al., 2020). This dataset contains images of two known satellites, with labels of the satellite pose, complimented with a smaller set of real images generated from a ground-based testbed. Similarly, the current SPAcecraft Recognition leveraging Knowledge of Space Environment (SPARK) challenge run by the University of Luxembourg provides image and depth data of 10 spacecrafts and five debris objects, to enable both classification and detection tasks. However, these datasets have only been made available recently so were not applicable to our applications. In addition, they contain only the required information for the specific task in mind, so are not applicable to many different problems.

The absence of available datasets is a key reason for the lack of progress in this field to date. The aim of this chapter is to meet this demand by constructing and releasing a simulation framework capable of generating datasets of labelled visual data of on-orbit proximity operations, thereby facilitating future research in this area.

The goal of the simulation is to generate labelled synthetic training data which accurately simulates the relative motion between an uncontrolled spinning debris target and a chaser satellite. First, we detail the dynamics of this situation, deriving equations to describe the relative motion between the two satellites as well as the attitude dynamics of a tumbling target. Following this, sections 2.2 and 2.3 discuss the construction of a simulation framework for generating the datasets. The simulation obeys the described dynamics of close-range rendezvous, generating image data which is labelled with all relevant information about the states of the two satellites. Datasets are constructed which will be used to train and validate the attitude determination algorithm detailed in the following chapters. Both the datasets and the simulation framework are provided alongside this work to enable further research.

2.1 Dynamics of Co-Orbiting Satellites

The dynamics of satellites in orbit has been well studied in literature. The case of two co-orbiting satellites is more complex, since solving and subtracting the absolute motion of each satellite can result in poor accuracy due to floating point errors. However, as mentioned previously, the inspection and capturing stages of the mission is the focus of this work. In this case, the two satellites are nearby, within approximately 50 metres of each other. This simplifies the problem, enabling the use of equations which describe the motion relative to a target satellite. These equations are provided in Section 2.1.2. The dynamics of the rotational motion of a satellite are then covered. However, first it is necessary to introduce the concept of a Keplerian orbit and the orbital elements which describe it.

2.1.1 Keplerian Elements

An orbit can be fully defined by six parameters, known as the Keplerian elements. From these, the position and velocity of the satellite can be determined at any point in the orbit. These parameters are defined here before being used to derive the equations in the following sections. The orbit is illustrated in Figure 2.1, relative to a reference plane – for Earth orbits, this is usually taken to be the equatorial plane.

The first two parameters define the shape and size of the orbit. The orbital eccentricity, e, describes the deviation from a circular orbit. An eccentricity of 0 means that the orbit is circular, while orbits with $0 < e < 1$ are elliptical. The semimajor axis, a, is defined as the average of the distance to the apoapsis and the distance to the periapsis; in a circular orbit, this is simply equivalent to the orbit radius. The semimajor axis defines the orbit size.

The orbital plane is defined, relative to the reference plane, by three angles. These are: the inclination, i; the longitude of the ascending node, Ω; and the argument of periapsis, ω. The motion of the satellite is constrained to within this orbit plane. The

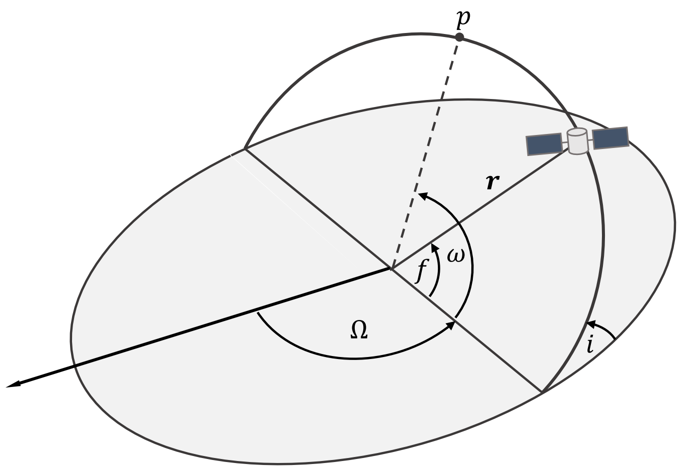


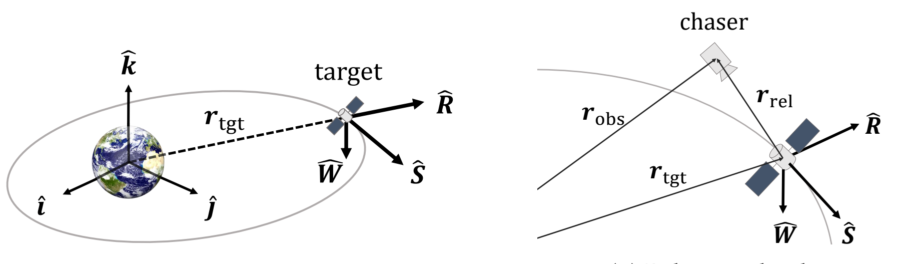
FIGURE 2.1: The Keplerian orbital elements. The elements are defined relative to the reference plane (in grey) and the reference direction (the black arrow).

position of the satellite in the orbit is then defined by another angle, which makes up the final orbital element. This is referred to as the true anomaly, f.

2.1.2 Relative Orbital Motion

The existence of equations to describe the relative motion between two close-orbiting satellites is crucial for spacecraft formation flying and rendezvous operations, among other applications. These equations describe the state of a *chaser* satellite relative to a nearby *target* satellite, represented in the coordinate frame centered on the target. The knowledge of the relative motion between satellites is of significant importance for the proximity operations in an ADR mission, where the target is taken to be the uncooperative debris object which is to be removed by the chaser. In the following section the derivation of the Clohessy-Wiltshire equations for relative orbital motion with a circular reference orbit will be derived; for a more in-depth discussion see Vallado and McClain (2007, Ch. 6). A few solutions for non-circular reference orbits will also be covered in less detail.

In order to determine the equations of motion, we must first construct the reference coordinate frame. It is important to note that, since the reference frame is centered on the non-stationary target satellite, the frame is itself rotating. Figure 2.2 illustrates the target-fixed rotating reference frame. Following Kaplan's notation (Kaplan, 1976), the $\hat{\boldsymbol{R}}$ axis is collinear with the position vector, $\hat{\boldsymbol{S}}$ is in the direction of the velocity vector aligned with the local horizontal (this will be aligned with the velocity vector only in circular orbits), and $\hat{W}$ is normal to the orbit plane. This reference frame is referred to as the local-vertical-local-horizontal (LVLH) frame.



(A) LVLH frame (B) Relative orbital motion

FIGURE 2.2: The target-centred LVLH reference frame.

The relative distance vector of the chaser is therefore taken to be

$$\mathbf{r}_{rel} = \mathbf{r}_{ch} - \mathbf{r}_{tgt}, \tag{2.1}$$

where $\mathbf{r}_{rel}$ is the relative position vector, and $\mathbf{r}_{ch}$ and $\mathbf{r}_{tgt}$ are the chaser and target position vectors respectively. The general equations of motion of a satellite in orbit, assuming no external forces, can be described as:

$$\ddot{\mathbf{r}} = \frac{\mu \mathbf{r}}{r^3}, \tag{2.2}$$

where μ is the gravitational constant of the central body.

Twice differentiating Equation 2.1 and substituting the equations of motion for the chaser and target satellites yields

$$\ddot{\mathbf{r}}_{rel} = -\frac{\mu \mathbf{r}_{ch}}{r_{ch}^3} + \frac{\mu \mathbf{r}_{tgt}}{r_{tgt}^3}. \tag{2.3}$$

In order for this equation to be useful, we first want to remove the dependence on $\mathbf{r}_{ch}$. Therefore we find an expression for the chaser's position vector divided by the cube of its magnitude:

$$\frac{\mathbf{r}_{ch}}{r_{ch}^3} = \frac{\mathbf{r}_{tgt} + \mathbf{r}_{rel}}{(r_{tgt}^2 + 2\mathbf{r}_{tgt} \cdot \mathbf{r}_{rel} + r_{rel}^2)^{3/2}}. \tag{2.4}$$

At this point, we make two key assumptions to simplify the equations. Firstly, r_{rel}^2 is assumed to be small compared with r_{tgt}^2, and can therefore be removed from the expression. The expression is then simplified by a using a binomial series, and all terms higher than first order are rejected. The result of these assumptions is that the relative motion equations will only hold for near-orbiting satellites, and will begin to break down at larger distances. However, this is a reasonable assumption to make as it only makes sense to use the relative motion in cases where the satellites are in close proximity.

The procedure described above leads to

$$\frac{\mathbf{r}_{ch}}{r_{ch}^3} = \frac{\mathbf{r}_{tgt} + \mathbf{r}_{rel}}{r_{tgt}^3} \left\{ 1 - \frac{3}{2} \left(\frac{2\mathbf{r}_{tgt} \cdot \mathbf{r}_{rel}}{r_{tgt}^2} \right) + \dots \right\}, \tag{2.5}$$

which, when substituted into Equation 2.3 – again making use of the assumption that r_{rel} is small – yields

$$\ddot{\mathbf{r}}_{rel} = -\frac{\mu}{r_{tgt}^3} \left\{ -\frac{3\mathbf{r}_{tgt}}{2r_{tgt}} \left(\frac{2\mathbf{r}_{tgt} \cdot \mathbf{r}_{rel}}{r_{tgt}} \right) + \mathbf{r}_{rel} \right\}. \tag{2.6}$$

We now want to represent the relative position of the chaser in the target's reference frame. Firstly, we see that the above equation can be rearranged in terms of the relative unit vectors. $\mathbf{r}_{tgt}/r_{tgt}^3$ is simply the unit vector $\hat{\boldsymbol{R}}$, while the dot product $(\mathbf{r}_{tgt} \cdot \mathbf{r}_{rel})/r_{tgt}$ gives the component of the position in this direction, i.e. the x component. However, the rotating reference frame also introduces a number of additional terms into the equation, which are dependent on the characteristics of the target's orbit.

2.1.2.1 Clohessy-Wiltshire Equations

We will first look at the case in which the orbit of the target satellite is near-circular. These equations were first developed by Clohessy and Wiltshire (1960) to analyse spacecraft rendezvous. The advantage of assuming a circular orbit is that it significantly simplifies the additional terms from the rotating reference frame, as will be shown, enabling the equations to be solved analytically.

The following equation relates the inertial and relative vectors in a rotating frame:

$$\ddot{\mathbf{r}}_R = \ddot{\mathbf{r}}_I - \dot{\boldsymbol{\omega}}_R \times \mathbf{r}_R - 2\boldsymbol{\omega}_R \times \dot{\mathbf{r}}_R - \boldsymbol{\omega}_R \times (\boldsymbol{\omega}_R \times \mathbf{r}_R), \tag{2.7}$$

where $\mathbf{r}_R$ and $\mathbf{r}_I$ are the reference and inertial vectors respectively, and $\boldsymbol{\omega}_R$ is the angular rate. Each of the added terms on the right hand side are caused by, respectively: the tangential acceleration due to changing angular velocity; the Coriolis acceleration; and the centripetal acceleration. Section 2.1.3.2 discusses the derivation of this equation in more detail.

Solving these terms for the reference orbit and substituting into Equation 2.6, we get

$$\ddot{\mathbf{r}}_{rel_R} = -\frac{\mu}{r_{tgt}^3} \left\{ \mathbf{r}_{rel} - 3x\hat{R} \right\} + \dot{\omega}y\hat{R} - \dot{\omega}x\hat{S} - 2\left(-\omega\dot{y}\hat{R} + \omega\dot{x}\hat{S} \right) + \omega^2 x\hat{R} + \omega^2 y\hat{S}. \tag{2.8}$$

As a result of the target's orbit being circular,

$$\begin{aligned} \omega &= \sqrt{\frac{\mu}{r_{tgt}^3}} \\ \dot{\omega} &= 0. \end{aligned} \tag{2.9}$$

Therefore, simplifying and rewriting each vector individually results in the following Clohessy-Wiltshire (CW) equations describing the relative motion along each axis:

$$\ddot{x} - 2\omega\dot{y} - 3\omega x = 0, \tag{2.10a}$$

$$\ddot{y} + 2\omega\dot{x} = 0, \tag{2.10b}$$

$$\ddot{z} + \omega^2 z = 0. \tag{2.10c}$$

The general case of relative orbital motion cannot be solved analytically, and instead must be integrated numerically. However, as a result of the simplifications granted by the assumption that the orbit is circular, it is possible to find an exact solution to the Clohessy-Wiltshire equations which represents the position and velocity of the chaser at time t, in terms of its initial state:

$$x(t) = \frac{\dot{x}_0}{\omega}\sin(\omega t) - \left(3x_0 + \frac{2\dot{y}_0}{\omega}\right)\cos(\omega t) + \left(4x_0 + \frac{2\dot{y}_0}{\omega}\right) \tag{2.11a}$$

$$\begin{aligned} y(t) = &\left(6x_0 + \frac{4\dot{y}_0}{\omega}\right)\sin(\omega t) + \frac{2x_0}{\omega}\cos(\omega t) - \\ &- (6\omega x_0 + 3\dot{y}_0)\,t + \left(y_0 - \frac{2\dot{x}_0}{\omega}\right) \end{aligned} \tag{2.11b}$$

$$z(t) = z_0\cos(\omega t) + \frac{\dot{z}_0}{\omega}\sin(\omega t) \tag{2.11c}$$

$$\dot{x}(t) = \dot{x}_0\cos(\omega t) + (3\omega x_0 + 2\dot{y}_0)\sin(\omega t) \tag{2.11d}$$

$$\dot{y}(t) = (6\omega x_0 + 4\dot{y}_0)\cos(\omega t) - 2\dot{x}_0\sin(\omega t) - (6\omega x_0 + 3\dot{y}_0) \tag{2.11e}$$

$$\dot{z}(t) = -z_0\omega\sin(\omega t) + \dot{z}_0\cos(\omega t). \tag{2.11f}$$

The CW equations have been used to solve various orbital mechanics problems. For example, by inverting the equations it is possible to determine the required initial velocity for the chaser to reach the target after a certain time; this is a common problem in spacecraft rendezvous. They are also well suited for computational applications which require the determination of relative orbital motion between two objects, due to the existence of a general analytical solution. This leads to lower errors and a reduced computational cost compared with a numerical integration method. However, the use of the CW equations is limited to orbits with eccentricities close to 0, otherwise the assumptions fall apart. The solution for the case of elliptic orbits will be investigated in the following section.

The equations of relative motion are also influenced by other factors which have not been discussed. For example, we have not considered the oblateness of Earth and assume zero thrust. For more information on the impacts of these effects, see Vallado and McClain (2007).

2.1.2.2 Equations of Motion for Elliptic Orbits

The equations of motion may take either time or true anomaly as the independent variable. For circular reference orbits the choice is largely irrelevant since either leads to the same form of equations. However, for elliptic orbits there exist separate solutions in true anomaly, for example the Tschauner-Hempel equations (Tschauner and Hempel, 1965), or in time, such as Melton (2000)'s state transition matrix (STM) solution. Solutions in time have the advantage that there is no need to solve Kepler's equation in order to produce results at specific times; on the other hand, anomaly-based methods tend to perform better at long-term predictions.

When moving from circular to elliptic orbits we need to take into account the changing angular velocity of the LVLH frame, ω, due to the eccentricity, e, of the target's orbit. The Tschauner-Hempel (TH) equations achieve this by extending the CW model and converting from a time-based to anomaly-based representation.

$$\ddot{x} - \dot{\omega}y - 2\omega\dot{y} - \omega^2 x - \frac{2\mu}{r^3}x = 0, \tag{2.12a}$$

$$\ddot{y} + \dot{\omega}y + 2\omega\dot{x} - \omega^2 y + \frac{\mu}{r^3}y = 0, \tag{2.12b}$$

$$\ddot{z} + \frac{\mu}{r^3}z = 0, \tag{2.12c}$$

where $r = \frac{a(1-e^2)}{1+e\cos f}$. Here, the angular rate can defined in terms of the mean motion $n = \sqrt{\mu/a^3}$ and the true anomaly f, i.e. the position in orbit. Thus, $\omega = \dot{f} = \frac{n(1+e\cos f)}{(1-e^2)^{3/2}}$ and $\dot{\omega} = \ddot{f} = \frac{-2n^2 e\sin f(1+e\cos f)^3}{(1-e^2)^3}$.

In order to convert the equations from the time domain into the anomaly domain, we use the following relationships between the derivation over time (using the notation $(\dot{\cdot})$) and over anomaly (represented by $(\cdot)'$).

$$(\dot{\cdot}) = (\cdot)'\dot{f}, \qquad (\ddot{\cdot}) = (\cdot)''\dot{f}^2 + \dot{f}\dot{f}'(\cdot)' \tag{2.13}$$

This leads to the Tschauner-Hempel equations below, using the same coordinate system as we used to derive the CW equations.

$$x'' = \frac{3 + e\cos f}{1 + e\cos f}x - \frac{2e\sin f}{1 + e\cos f}y + \frac{2e\sin f}{1 + e\cos f}x' + 2y', \tag{2.14a}$$

$$y'' = \frac{2e\sin f}{1 + e\cos f}x + \frac{e\cos f}{1 + e\cos f}y - 2x' + \frac{2e\sin f}{1 + e\cos f}y', \tag{2.14b}$$

$$z'' = -\frac{1}{1 + e\cos f}z + \frac{2e\sin f}{1 + e\cos f}z'. \tag{2.14c}$$

The TH equations can be used over the CW equations for cases when the orbit of the target satellite has a non-zero eccentricity. However, there is no general solution so the equations must be integrated numerically using the equations in Section 2.2.2. Besides this, in order to determine the time response it becomes necessary to solve Kepler's equations for the target's orbit.

Melton (2000) provides an alternative solution with the Melton state transition matrix. This method has two key advantages over the TH equations: firstly, the model uses time as the independent variable, so there is no need to solve Kepler's equations to produce time-dependent results; and additionally, the state transition matrix is a direct, non-iterative algorithm. These attributes make the Melton STM well-suited to instances when fast, on-board calculation of relative motion is required. However, the solution is approximate and thus is only accurate to order e^2, which makes it unsuitable for practical purposes with eccentricities greater than 0.3.

Similarly to the TH method above, Melton extends upon the CW equations, beginning with Equations 2.12. The time-dependent variables r and ω can be expanded in powers of eccentricity, as below:

$$r/a = 1 - e\cos nt - e^2/2 - e\cos nt - (e^2/2)\cos 2nt + \mathcal{O}(e^3), \tag{2.15a}$$

$$\omega = h/r^2 = (h/a^2)1 + 2e\cos nt + (e^2/2)[5\cos 2nt + 1] + \mathcal{O}(e^3), \tag{2.15b}$$

where h and a represent the specific angular momentum and semimajor axis of the *reference* orbit, respectively. The angular momentum can be defined in terms of Keplerian elements as $h = \sqrt{a(1-e)/\mu}$. Writing the equations of motion in matrix form,

$$\dot{\boldsymbol{x}}(t) = A(t)\boldsymbol{x}(t), \qquad \boldsymbol{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T, \tag{2.16}$$

it can be found that, since $A(t)$ consists of periodic terms, it can also be expanded in powers of e. The same is true for the STM, $\phi(t)$:

$$A(t) = A_0 + eA_1(t) + e^2A_2(t) + \dots, \tag{2.17a}$$

$$\phi(t) = \phi_0(t) + e\phi_1(t) + e^2\phi_2(t) + \dots, \tag{2.17b}$$

where

$$\phi_0(t) = \exp(A_0 t). \tag{2.18}$$

By Equation 2.16, the STM must also satisfy the differential equation:

$$\begin{aligned}\dot{\phi}(t) &= A(t)\phi(t), \\ &= [A_0 + eA_1(t) + \ldots] \times [\phi_0(t) + e\phi_1(t) + \ldots].\end{aligned} \tag{2.19}$$

Collecting the coefficients of like powers of e and integrating, we now obtain the following expressions for the elements of ϕ:

$$\begin{aligned}\phi_1(t) &= \int_0^t e^{A_0(t-s)} A_1(s) e^{A_0 s} ds, \\ \phi_2(t) &= \int_0^t e^{A_0(t-s)} \left[A_1(s)\phi_1(s) + A_2(s)e^{A_0 s}\right] ds, \\ &\vdots, \\ \phi_n(t) &= \int_0^t e^{A_0(t-s)} \left[A_1(s)\phi_{n-1}(s) + \cdots + A_n(s)e^{A_0 s}\right] ds.\end{aligned} \tag{2.20}$$

Analysis of Melton's STM shows that including the first order terms increases the accuracy significantly even for low values of e, while the second order terms become more important at higher values of eccentricity. However, the model is ineffective for highly eccentric orbits, such as those with $e > 0.3$.

There exist several further solutions to the relative motion problem, in terms of both time and true anomaly, which will not be discussed in detail here. For example, Carter (1998) and Yamanaka and Ankersen (2002) present STMs in terms of f, the latter of which is very compact and simple to implement. Lee et al. (2007) present a time-dependent STM based upon the analytical solution to the two-body problem; however, though the nature of the STM appears to be time-explicit, Kepler's equation must be solved at each time step.

2.1.2.3 The Inspection Orbit

An active debris removal mission, as detailed in Section 1.1.1, will often include an inspection stage in which various characteristics of the target are measured. In this stage, the chaser will aim to enter a relative circular orbit (also known as a "football" orbit) about the target, enabling the inspection of the entire object. The chaser's control system will ensure that this football orbit is kept, and that the target remains in view of the chaser's optical sensors.

However, the case where the target is in a circular orbit, inspection of the response of the CH equations reveals an important result. It can be shown that an initial state of

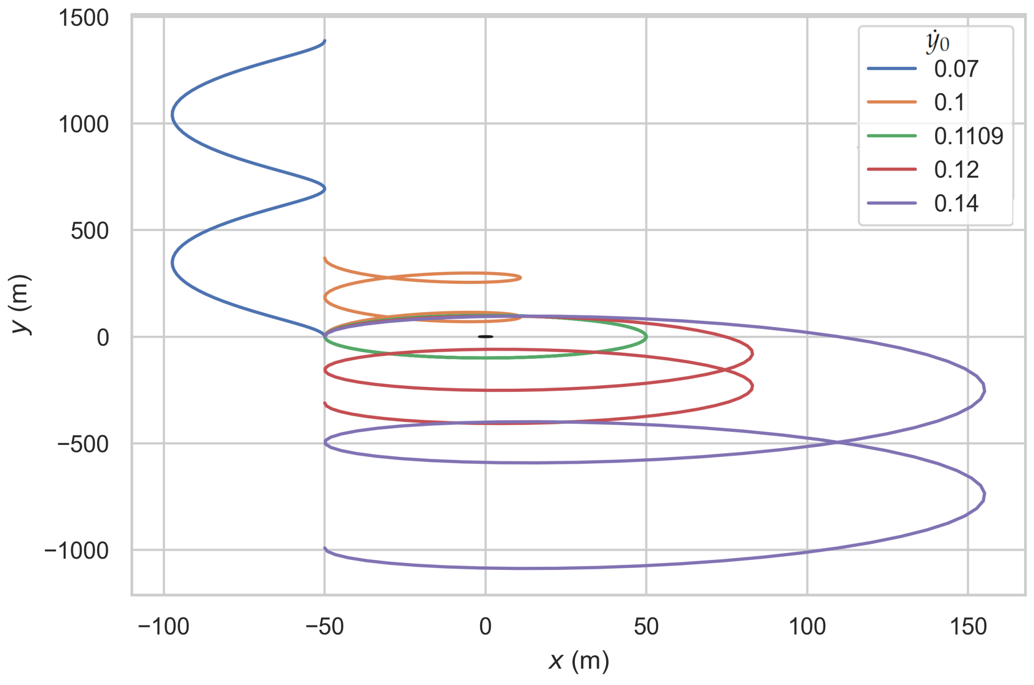


FIGURE 2.3: Motion of the chaser relative to the target over two periods, under different initial $\dot{y}_0$, for an initial x_0 displacement of 50m.

$\dot{y}_0 = -2\omega x_0$ with $x_0 \neq 0$ and $y_0, z_0, \dot{x}_0, \dot{z}_0 = 0$, leads to an elliptical relative orbit about the target. This result is illustrated in Figure 2.3, in which case the value $-2\omega x_0 = 0.1109$; this initial velocity results in the path shown in green. This figure shows the relative orbits for several different values of $\dot{y}_0$, selected to cover a range of situations.

Thus, for a target with orbit eccentricity of 0, the football orbit can be acquired and maintained using only a single manoeuvre. This is useful as it provides a simple test case which we use in our simulations in Section 2.3 to generate on-orbit image data during the inspection stage.

2.1.3 Relative Attitude

The previous section introduces the concept of relative orbital motion between two close-orbiting satellites. However, besides the relative motion it is also necessary to consider the attitudes of the two satellites. This section discusses a number of different attitude parameterisations, and introduces a method for the determination of the attitude of a target by observation from a chaser satellite.

The ability to accurately determine a target's attitude is particularly important for active debris removal problems, where the target may be tumbling at a high angular rate. The dynamics of the rotation are typically not known to the chaser satellite beforehand. For many ADR techniques, as discussed in Chapter 1, it is necessary to either detumble the target or align the chaser with the target, prior to capture or removal. Knowledge of the target's attitude is therefore required.

In the following, we describe several parameterisations useful for describing the attitude of a satellite, before introducing the attitude kinematics which describe the relationship between the attitude and the angular velocity in the target's LVLH frame. The information provided here can be found in more detail in Markley and Crassidis (2014).

2.1.3.1 Attitude Parameterisations

The aim of an attitude parameterisation system is to enable the conversion from a *fixed* frame, such as the Earth Centred Inertial (ECI) frame, to a *moving* frame, for example the body frame centred on a rotating target satellite. When deriving the equations of relative orbital motion, we work with the moving target-centric LVLH reference frame which rotates about the Earth as the satellite moves through its orbit. In the case of attitude dynamics, we are interested in the orientation of this reference frame after rotation about an axis **e** which passes through its origin, as shown in Figure 2.4. The rotation can be uniquely defined by this rotation axis along with the angle of rotation θ; however, it is often more convenient to use one of the following parameterisations.

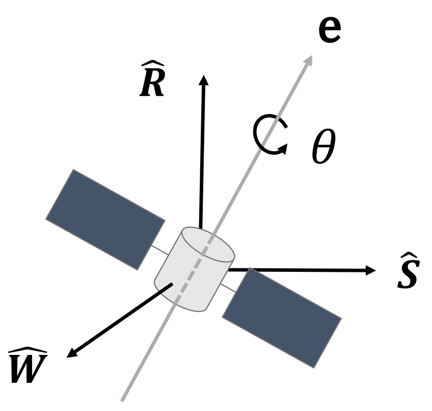


FIGURE 2.4: A rotation of the target-centred body frame.

Rotation Matrices If we denote the fixed frame by the unit vectors $[\hat{i}_1, \hat{j}_1, \hat{k}_1]$ and the moving frame by $[\hat{i}_0, \hat{j}_0, \hat{k}_0]$, then the same vector can be represented in the different frames as:

$$\mathbf{v} = x_1\hat{i}_1 + y_1\hat{j}_1 + z_1\hat{j}_1, \qquad \mathbf{v} = x_0\hat{i}_0 + y_0\hat{j}_0 + z_0\hat{j}_0. \tag{2.21}$$

The components of the vector in frame 0 ($\mathcal{F}_0$), the moving LVLH frame, can be found by taking the dot product of the vector in frame 1 ($\mathcal{F}_1$) with the unit vectors $\hat{\imath}_0$, $\hat{\jmath}_0$ and $\hat{k}_0$, respectively. Or, in matrix form

$$\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{bmatrix} (\hat{\imath}_1 \cdot \hat{\imath}_0) & (\hat{\jmath}_1 \cdot \hat{\imath}_0) & (\hat{k}_1 \cdot \hat{\imath}_0) \\ (\hat{\imath}_1 \cdot \hat{\jmath}_0) & (\hat{\jmath}_1 \cdot \hat{\jmath}_0) & (\hat{k}_1 \cdot \hat{\jmath}_0) \\ (\hat{\imath}_1 \cdot \hat{k}_0) & (\hat{\jmath}_1 \cdot \hat{k}_0) & (\hat{k}_1 \cdot \hat{k}_0) \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}. \tag{2.22}$$

This coordinate transformation can be written more compactly as

$$\mathbf{v}_0 = Q_{01}\mathbf{v}_1, \tag{2.23}$$

where Q_{01} is the *rotation matrix* from $\mathcal{F}_1$ to $\mathcal{F}_0$. The matrix Q_{10}, from $\mathcal{F}_0$ to $\mathcal{F}_1$ is simply the transpose of Q_{01}. Multiple rotations may be simply represented by a composition of rotation matrices:

$$Q_{01} = Q_{02}Q_{21}. \tag{2.24}$$

The rotation matrix representation is the simplest form of attitude parameterisation, and benefits from robustness and the existence of the matrix composition relation. However, the interpretation of the matrices is not intuitive; there is no easy way to determine the direction and angle of rotation from observing the matrix Q. Rotation matrices also contain more parameters than are necessary to define the rotation.

Euler Angles Euler angles present a parameterisation system which is more intuitive than rotation matrices, while also reducing the number of parameters required. This theorem states that any rotation in three dimensions may be represented by three rotations about the axes. These rotations are often performed in a Z-X-Z sequence: a rotation around the $\hat{z}$ axis followed by a rotation about the *new*, rotated $\hat{x}$ axis, then a final rotation about the *new* $\hat{z}$ axis.

This sequence of rotations can be represented in rotation matrix form as a composition of three rotations:

$$Q_{01} = Q_{03}Q_{32}Q_{21} = R_z(\phi_3)R_x(\phi_2)R_z(\phi_1), \tag{2.25}$$

which is equivalent to

$$Q_{01} = \begin{bmatrix} \cos(\phi_3) & \sin(\phi_3) & 0 \\ -\sin(\phi_3) & \cos(\phi_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi_2) & \sin(\phi_2) \\ 0 & -\sin(\phi_2) & \cos(\phi_2) \end{bmatrix} \cdot \begin{bmatrix} \cos(\phi_1) & \sin(\phi_1) & 0 \\ -\sin(\phi_1) & \cos(\phi_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.26}$$

While more intuitive than the rotation matrix representation, Euler angles suffer from a number of other issues. There exists a singularity when the rotation about the second axis is 0 or 180: at this point, there are infinitely many solutions to the Euler sequence. In addition, the computational cost of successive trigonometric evaluations mean that this parameterisation is poorly suited for high-speed computation.

Quaternions Instead of representing the attitude by a series of the rotations around the axes, it can be simplified to a single rotation by an angle θ about the rotation axis specified by a unit vector $\mathbf{e}$, as illustrated in Figure 2.4 above. This leads to the definition of the rotation vector $\boldsymbol{\theta} = \theta\mathbf{e}$. The vector $\mathbf{e}$ remains invariant in the transform; its coordinates in the fixed frame are the same as in the rotating frame.

$$\begin{aligned}\mathbf{e} = [e_x e_y e_z]\begin{bmatrix}\hat{i}_0\\ \hat{j}_0\\ \hat{k}_0\end{bmatrix} = [e_x e_y e_z]\begin{bmatrix}\hat{i}_1\\ \hat{j}_1\\ \hat{k}_1\end{bmatrix}\\ \mathbf{e} = Q_{01}\mathbf{e}\\ (Q_{01} - I)\mathbf{e} = 0.\end{aligned} \tag{2.27}$$

Therefore, $\mathbf{e}$ is the eigenvector associated to the eigenvalue $\lambda = +1$. We can use $\mathbf{e}$ and θ to define the *quaternion* representation, also known as the Euler-Rodrigues parameters.

$$\mathbf{q}(\mathbf{e}, \theta) = \begin{bmatrix}\mathbf{e}\sin(\theta/2)\\ \cos(\theta/2)\end{bmatrix} = \begin{bmatrix}e_x\sin(\theta/2)\\ e_y\sin(\theta/2)\\ e_z\sin(\theta/2)\\ \cos(\theta/2)\end{bmatrix}. \tag{2.28}$$

A significant advantage of the quaternion parameterisation is that it expresses the attitude as a homogenous quadratic function of the elements of the quaternion, as shown below, and therefore requires no trigonometric function evaluations.

$$Q_{01} = \begin{bmatrix}q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4)\\ 2(q_2q_1 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4)\\ 2(q_3q_1 + q_2q_4) & 2(q_3q_2 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2\end{bmatrix}, \tag{2.29}$$

where

$$\mathbf{q} = [q_1\ q_2\ q_3\ q_4]^T = (q_1, q_2, q_3) + q_4. \tag{2.30}$$

Quaternions are also more efficient than the rotation matrix, since they include four components instead of nine. A rotation quaternion is a unit quaternion, which means it has unit norm. This is the only constraint placed upon the quaternion, as opposed to the six constraints on the rotation matrix by orthogonality. Additionally, for rotation quaternions the rotation $\mathbf{q}$ is equivalent to $-\mathbf{q}$.

Similarly to the rotation matrix, the quaternion representation has a composition relation

$$\mathbf{q}_{01} = \mathbf{q}_{21} \odot \mathbf{q}_{02}. \tag{2.31}$$

Given the two rotations $\mathbf{q}_\mathrm{c} = (c_1, c_2, c_3) + c_4$ and $\mathbf{q}_\mathrm{d} = (d_1, d_2, d_3) + d_4$, the quaternion composition is given by the Hamilton product:

$$\mathbf{q}_\mathrm{c} \odot \mathbf{q}_\mathrm{d} = \begin{bmatrix} c_4 & -c_3 & c_2 & c_1 \\ c_3 & c_4 & -c_1 & c_2 \\ -c_2 & c_1 & c_4 & c_3 \\ -c_1 & -c_2 & -c_3 & c_4 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}. \tag{2.32}$$

We will also introduce the alternative quaternion operation $\mathbf{q}_\mathrm{c} \otimes \mathbf{q}_\mathrm{d}$, since this will be more useful in attitude analysis. The two operations are related by $\mathbf{q}_\mathrm{c} \otimes \mathbf{q}_\mathrm{d} = \mathbf{q}_\mathrm{d} \odot \mathbf{q}_\mathrm{c}$ and the former is given by

$$\mathbf{q}_\mathrm{c} \otimes \mathbf{q}_\mathrm{d} = \begin{bmatrix} c_4 & c_3 & -c_2 & c_1 \\ -c_3 & c_4 & c_1 & c_2 \\ c_2 & -c_1 & c_4 & c_3 \\ -c_1 & -c_2 & -c_3 & c_4 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}. \tag{2.33}$$

In order to rotate a point vector $\mathbf{u}$ by a quaternion $\mathbf{q}$, the vector must be converted to the quaternion representation $\mathfrak{u} = (u_x, u_y, u_z) + 0$. The rotated vector is then calculated as

$$\mathfrak{v} = \mathbf{q} \otimes \mathfrak{u} \otimes \mathbf{q}^* = \mathbf{q}^* \odot \mathfrak{u} \odot \mathbf{q}, \tag{2.34}$$

where $\mathbf{q}^*$ is the conjugate of $\mathbf{q}$, given by $\mathbf{q}^* = (-q_1, -q_2, -q_3) + q_4$. This allows us to work entirely in the quaternion representation without the need to convert to and from the attitude matrix.

The quaternion can be extracted from an attitude matrix by normalising one of the following four vectors. To avoid confusion, we will represent the matrix Q_{01} as A, with components $A_{11}, A_{12}, \ldots$

$$\begin{gathered} \begin{bmatrix} 1 + 2A_{11} - \mathrm{tr}A \\ A_{12} + A_{21} \\ A_{12} + A_{31} \\ A_{23} - A_{32} \end{bmatrix} = 4q_1\mathbf{q}, \qquad \begin{bmatrix} A_{21} + A_{12} \\ 1 + 2A_{22} - \mathrm{tr}A \\ A_{23} + A_{32} \\ A_{31} - A_{13} \end{bmatrix} = 4q_2\mathbf{q} \\ \begin{bmatrix} A_{31} + A_{13} \\ A_{32} + A_{23} \\ 1 + 2A_{33} - \mathrm{tr}A \\ A_{12} - A_{21} \end{bmatrix} = 4q_3\mathbf{q}, \qquad \begin{bmatrix} A_{23} - A_{32} \\ A_{31} - A_{13} \\ A_{12} - A_{21} \\ 1 + \mathrm{tr}A \end{bmatrix} = 4q_4\mathbf{q}, \end{gathered} \tag{2.35}$$

where tr$A$ denotes the trace of A. The numerical error is minimised by choosing the vector with the greatest norm; this is the vector with the largest value of $|q_i|$ on the right hand side.

Due to the discussed advantages of the quaternion representation, it is well suited for computation applications. As such, we use quaternions to represent the attitudes of the chaser and target satellites in our simulation. The main disadvantage of quaternions is their unintuitive interpretation, but this is an issue only when looking at the problem from an analytical perspective.

2.1.3.2 Attitude Kinematics

Attitude dynamics covers the aspects of rotational motion which can be analysed without consideration of forces or torques; the introduction of these leads us to the domain of attitude kinematics. We will represent the attitude matrix of the satellite as $Q_{01}(t)$. This is the rotation matrix to convert from the fixed frame to the moving frame at the current time t. We also introduce the concept of an angular velocity vector $\boldsymbol{\omega}_0^{01}$. This notation expresses a rotation from $\mathcal{F}_1$ to $\mathcal{F}_0$, represented in $\mathcal{F}_0$. The frame in which the rotation is represented is important, as we shall see later.

Attitude Matrix In order to determine the time dependence of the attitude matrix, we begin with the fundamental definition of the derivative of Q_{01}:

$$\dot{Q}_{01}(t) \equiv \lim_{\Delta t \to 0} \frac{Q_{01}(t+\Delta t) - Q_{0_1}(t)}{\Delta t} \quad = \lim_{\Delta t \to 0} \frac{Q_{01}(t+\Delta t)Q_{10}(t) - I_3}{\Delta t} Q_{01}(t). \tag{2.36}$$

As Δt goes to zero, we can represent the difference between the identity matrix and the product $Q_{01}(t+\Delta t)Q_{10}(t)$ by a rotation vector:

$$Q_{01}(t+\Delta t)Q_{10}(t) = \exp(-[\Delta\boldsymbol{\theta}_0^{01}\times]) \approx I_3 - [\Delta`_0^{01}\times], \tag{2.37}$$

where $\boldsymbol{\theta}_0^{01}$ corresponds to the small rotation from $\mathcal{F}_1$ to $\mathcal{F}_0$ represented in $\mathcal{F}_0$. This is equivalent to the rotation vector defined previously as $\boldsymbol{\theta} = \theta\mathbf{e}$. The notation $[\mathbf{v}\times]$ corresponds to the cross product matrix of the vector $\mathbf{v}$:

$$\mathbf{v} = [x, y, z]^T, \quad [\mathbf{v}\times] = \begin{bmatrix} 0 & -z & x \\ z & 0 & -x \\ -y & y & 0 \end{bmatrix}. \tag{2.38}$$

Substituting Equation 2.37 into Equation 2.36 results in the fundamental equation of attitude kinematics:

$$\dot{Q}_{01}(t) = \lim_{\Delta t \to 0} \frac{-[\Delta \boldsymbol{\theta}_0^{01} \times]}{\Delta t} Q_{01} = -[\boldsymbol{\omega}_0^{01}(t)\times] Q_{01}(t), \tag{2.39}$$

where the angular velocity vector is defined by

$$\boldsymbol{\omega}_0^{01}(t) \equiv \lim_{\Delta t \to 0} \frac{\Delta \boldsymbol{\theta}_0^{01}}{\Delta t}. \tag{2.40}$$

The angular velocity vector for a composition of rotations $Q_{01} = Q_{02}Q_{21}$ is equivalent to the sum of the individual angular velocities, provided that they are expressed in the same reference frame:

$$\boldsymbol{\omega}_0^{01} = \boldsymbol{\omega}_0^{02} + \boldsymbol{\omega}_0^{21} = \boldsymbol{\omega}_0^{02} + Q_{02}\boldsymbol{\omega}_2^{21}. \tag{2.41}$$

Vector Kinematics Vector kinematics can be used to describe the relationship between the representations of a vector in two different frames, dependent on the relative rotation of the frames. This is critical in deriving the relative orbital motion equations described in Section 2.1.2. Consider the vector $\mathbf{x}$ in the frames $\mathcal{F}_0$ and $\mathcal{F}_0$:

$$\mathbf{x}_0 = Q_{01}\mathbf{x}_1. \tag{2.42}$$

The time derivative of $\mathbf{x}_0$ is given by

$$\begin{aligned} \dot{\mathbf{x}}_0 &= Q_{01}\dot{\mathbf{x}}_1 + \dot{Q}_{01}\mathbf{x}_1 = Q_{01}\dot{\mathbf{x}}_1 - [\boldsymbol{\omega}_0^{01}\times]Q_{01}\mathbf{x}_1 \\ &= Q_{01}\dot{\mathbf{x}}_1 - \boldsymbol{\omega}_0^{01} \times \mathbf{x}_0. \end{aligned} \tag{2.43}$$

This expresses the derivative of a vector in one frame as the sum of two terms: the transformation of the derivative in another frame and a term reflecting the rotational motion between the two frames. Differentiating this equation again gives

$$\begin{aligned} \ddot{\mathbf{x}}_0 &= Q_{01}\ddot{\mathbf{x}}_1 - [\boldsymbol{\omega}_0^{01}\times]Q_{01}\dot{\mathbf{x}}_1 - \boldsymbol{\omega}_0^{01} \times \dot{\mathbf{x}}_0 - \dot{\boldsymbol{\omega}}_0^{01} \times \mathbf{x}_0 \\ &= Q_{01}\ddot{\mathbf{x}}_1 - [\boldsymbol{\omega}_0^{01}\times](\dot{\mathbf{x}}_0 + \boldsymbol{\omega}_0^{01} \times \mathbf{x}_0) - \boldsymbol{\omega}_0^{01} \times \dot{\mathbf{x}}_0 - \dot{\boldsymbol{\omega}}_0^{01} \times \mathbf{x}_0 \\ &= Q_{01}\ddot{\mathbf{x}}_1 - \boldsymbol{\omega}_0^{01} \times (\boldsymbol{\omega}_0^{01} \times \mathbf{x}_0) - 2\boldsymbol{\omega}_0^{01} \times \dot{\mathbf{x}}_0 - \dot{\boldsymbol{\omega}}_0^{01} \times \mathbf{x}_0. \end{aligned} \tag{2.44}$$

The above equation describes the added centripetal, Coriolis and Euler acceleration when converting from an acceleration in the inertial reference frame to a rotating frame. The derivation of the motion of two satellites relative to the target's rotating reference frame therefore makes use of this relationship; note the similarity to Equation 2.7.

Quaternions As explained previously, the quaternion representation of the attitude is often more useful than the rotation matrix since it has fewer parameters and fewer constraints. As such, we will look at how the kinematic equation differs when using quaternions.

Again, we begin with the derivative of the quaternion in time

$$\dot{\mathbf{q}}(t) \equiv \lim_{\Delta t \to 0} \frac{\mathbf{q}(t+\Delta t) - \mathbf{q}(t)}{\Delta t}. \tag{2.45}$$

It can be shown that the quaternion can be represented in terms of the rotation vector (in quaternion form $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3) + 0$) as follows:

$$[\mathbf{q}\otimes] = \exp[(\boldsymbol{\theta}/2)\otimes]. \tag{2.46}$$

We can therefore represent the rotation from $\mathbf{q}(t)$ to $\mathbf{q}(t+\Delta t)$ as the exponential of a rotation vector:

$$\mathbf{q}(t+\Delta t) = \exp[(\Delta\boldsymbol{\theta}/2)\otimes]\mathbf{q}(t) \approx \mathbf{q}(t) + [(\Delta\boldsymbol{\theta}/2)\otimes]\mathbf{q}(t). \tag{2.47}$$

Inserting into Equation 2.45 and taking the limit as $\Delta t \to 0$ results in

$$\dot{\mathbf{q}}(t) = \frac{1}{2}\boldsymbol{\omega}(t) \otimes \mathbf{q}(t) = \frac{1}{2}\mathbf{q}(t) \odot \boldsymbol{\omega}(t), \tag{2.48}$$

where the angular velocity vector $\boldsymbol{\omega}(t)$ is defined by Equation 2.40. Again, the vector here is given in quaternion form $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3) + 0$. We can also write the angular velocity in terms of the quaternion rate:

$$\boldsymbol{\omega} = 2\mathbf{q} \odot \dot{\mathbf{q}}. \tag{2.49}$$

The kinematic equation for the quaternion is very similar to that of the attitude matrix. This relationship allows us to determine the satellite's attitude given its angular velocity, or vice versa, using solely the quaternion representation.

2.1.4 Relationship Between the Observed and True Rotational Motion

The visual sensors on-board the chaser spacecraft observe the rotation of the target over time. However, the *observed* change in attitude between two successive instants of time is a combination of the rotations of both the target and the chaser. Since the chaser is an operational spacecraft, and thus presumably equipped with sufficiently accurate sensors and a fully functional attitude and orbit control system, one can assume that the motion of the chaser satellite is known. Therefore, it is possible to determine the actual rotational state of the target by means of in-orbit observations of its relative motion as observed from the chaser.

Instantaneous rotations[1] between two frames (or 'rotations', for short) are most effectively described in terms of quaternions, for the reasons given in Section 2.1.3. Thus, the instantaneous rotation from frame $\mathcal{F}_A$ to frame $\mathcal{F}_B$ is given by the quaternion $\mathbf{q}_{BA}$. Quaternions have a convenient composition relation that allows the concatenation of successive rotations when multiple frames are considered:

$$\mathbf{q}_{\mathrm{CA}} = \mathbf{q}_{\mathrm{CB}} \otimes \mathbf{q}_{\mathrm{BA}}. \tag{2.50}$$

It is important to note that instantaneous rotations and the attitude of an object are deeply related concepts, as the attitude of an object can be described through an instantaneous rotation of its body frame from a given *departure* reference frame. Consequently, in order to describe an object's change in attitude between two successive images (in practice, each taken at successive instants of time), it suffices to find the quaternion that describes the instantaneous rotation exhibited by its body frame from one image to the next. To this ends, body frames must be defined for both, the target and the chaser, and they must be observed at successive instants of time, each of which will provide an image to feed into the landmark detection algorithm.

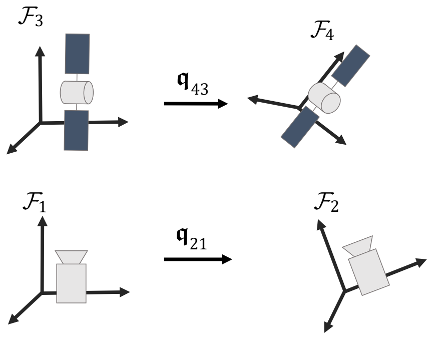


FIGURE 2.5: Depiction of the body frames of the target and chaser satellites, before and after a time step δt.

This is illustrated in Figure 2.5, which depicts two different instants of time at which the attitude of the aforementioned body frames is considered. The frame $\mathcal{F}_1$ is defined as a reference frame that is aligned with the chaser's body frame at the initial instant of time, where the first image is obtained; thus, note that in the subsequent rotational motion the chaser's body frame will evolve, whereas $\mathcal{F}_1$ will remain unchanged in the LVLH space. After a time step δt, when the second image is to be obtained, the chaser's body frame would have evolved (following its attitude dynamics) to its

[1]The term *instantaneous* is here used to devoid the concept of a rotation from any notion of time dependence, and thus highlight that the considered rotation is simply defined as the difference in attitude between two frames, regardless of kinematic considerations.

current pose; thus, another reference frame can be defined, namely $\mathcal{F}_2$, that is aligned with the chaser's body frame at this instant of time. Therefore, the change of attitude exhibited by the chaser's reference frame in the considered timeframe, is equivalent to the instantaneous rotation from frame $\mathcal{F}_1$ to frame $\mathcal{F}_2$, which can thus be represented by the quaternion $\mathbf{q}_{21}$. Similarly, reference frames $\mathcal{F}_3$ and $\mathcal{F}_4$ can be defined so they match the attitude of the target's body frame at the two considered instants of time, where quaternion $\mathbf{q}_{43}$ establishes the rotation from $\mathcal{F}_3$ to $\mathcal{F}_4$. Finally, in order to describe the attitude of each of these four reference frames, it is useful to define a separate inertial reference frame, $\mathcal{F}_0$, which can be arbitrarily defined and used as a common departure reference frame.

Determining the rotational state of the target requires that the quaternion $\mathbf{q}_{43}$ be known from images at any two successive instants of time. However, since $\mathcal{F}_3$ and $\mathcal{F}_4$ are unknown, the determination of $\mathbf{q}_{43}$ can only be done indirectly. To this end, the only information available in practice is the target's *observed* change in attitude as seen by the chaser, i.e. the relative attitude of the target body frame referred to the chaser body frame. The relative attitude of the target with respect to the chaser is described by quaternions $\mathbf{q}_{31}$ and $\mathbf{q}_{42}$, respectively, for each of the two instants of time considered. The change of the target's attitude relative to the chaser is an important value: this is the output of the rotation estimation algorithm presented in Chapter 5, as well as the same quaternion used in the definition of the loss term L_{rot} provided later in Equation (4.2). Thus, for the sake of notation consistency, we shall denote this change of the relative attitude by $\hat{\mathbf{q}}$, where no subscript is indicated; from the quaternion composition relation it is straightforward to see that this quaternion is related to $\mathbf{q}_{31}$ and $\mathbf{q}_{42}$ by means of

$$\mathbf{q}_{42} = \hat{\mathbf{q}} \otimes \mathbf{q}_{31}. \tag{2.51}$$

The rotational state of the chaser satellite with respect to an inertial frame $\mathcal{F}_0$ is assumed to be known with accuracy at all times, thus quaternions $\mathbf{q}_{10}$ and $\mathbf{q}_{2\text{-}}$ are available, and therefore so is $\mathbf{q}_{21}$. The desired rotation, $\mathbf{q}_{43}$, can thus be written in terms of the known initial state of the chaser body frame, i.e.

$$\mathbf{q}_{43} = \mathbf{q}_{41} \otimes \mathbf{q}_{13} = \mathbf{q}_{41} \otimes \mathbf{q}_{31}^{*}. \tag{2.52}$$

where $\mathbf{q}_{31}^{*}$ refers to the the conjugate of $\mathbf{q}_{31}$, as seen in Equation 2.34.

In order to solve for $\mathbf{q}_{43}$, Equation (2.52) shows that $\mathbf{q}_{41}$ needs to be expressed in terms of known quantities. Using rotation composition and combining with Equation (2.51) yields:

$$\mathbf{q}_{41} = \mathbf{q}_{42} \otimes \mathbf{q}_{21} = \hat{\mathbf{q}} \otimes \mathbf{q}_{31} \otimes \mathbf{q}_{21}, \tag{2.53}$$

thus Equation (2.52) can be rewritten as

$$\mathbf{q}_{43} = \hat{\mathbf{q}} \otimes \mathbf{q}_{31} \otimes \mathbf{q}_{21} \otimes \mathbf{q}_{31}^{*}. \tag{2.54}$$

Clearly, this expression still contains an unknown quantity, $\mathbf{q}_{31}$, since the true initial state of the target is unknown; finding a work-around requires some additional considerations. At this stage, it must be noted that we have not at any point introduced any assumption nor constraint in the definition of the target body frame, beyond the fact that it needs to be a body frame, i.e. a frame rigidly attached to the target object. Usual choices for a body frame are typically based either on geometric considerations, or on the mass geometry of the object and its principal axes of inertia; however, the orientation of the body frame can actually be arbitrary, as long as it rotates with the body it is attached to. One must also bear in mind that the target object can in principle be of unknown geometry and properties, so for a rotation estimation algorithm to be general, it is actually desirable that it does not depend on a-priori information nor predefined body frames, and thus that it be an automatic, self-starting algorithm that will work on any target object, regardless of its shape or inertia properties.

With these considerations in mind, and since one is free to choose any body frame for the target, a convenient choice is to define a target body frame which is intentionally aligned with the chaser body frame at the very first instant of time. It is easy to see that this choice yields to an initial quaternion $\mathbf{q}_{31}$ associated to an identity rotation matrix, since the body frame of the chaser and the target would both be coincident, i.e. there would be no rotation between them. Hence, Equation (2.54) would simply to

$$\mathbf{q}_{43} = \hat{\mathbf{q}} \otimes \mathbf{q}_{21}. \tag{2.55}$$

Obviously, in subsequent instants of time this would no longer be the case if the target, the chaser or both are rotating, so Equation (2.54) would need to be used in all remaining time steps; indeed, as time evolves and the rotation estimation algorithm is thus successively applied at subsequent instants of time, the quaternion $\mathbf{q}_{31}$ for images belonging to successive instants of time would change, but it would now be a known quantity, because it can be computed from the angular velocity by integrating the equations of motion for the relative attitude; indeed, the proposed choice for the target body frame allows to set the initial conditions at the very first instant of time, which allows to start the integration of the equations of motion, which in turn can provide $\mathbf{q}_{31}(t)$ at any subsequent instant of time, and therefore the rotation estimation algorithm would be complete.

Alternatively, note that if one is solely interested in determining the instantaneous rotation between any two subsequent frames (i.e., a quaternion), but not the angular velocity vector, then the arbitrary target body frame could actually be reset or redefined at each time step so it recurrently coincides with the chaser body frame; therefore, in practice Equation (2.55) could be used at each time step instead of Equation (2.54). Note, however, that with this procedure, computing the angular velocity would require extra caution due to the target body frame being re-defined in each time step.

The proposed choice of the target body frame provides a universal algorithm in the sense that it does not require any a-priori or predefined information about the target object; however, there are situations where one needs to specifically select a user-specified target body frame, e.g. based on recognisable features of the target object. For example, this would be the case for a rendezvous and docking manoeuvre, where the location of docking stations, solar arrays and other peripherals of the target are provided in a specific target body frame, and thus it would be required that the target's attitude relative to the chaser be computed using a predefined target body frame. In this case, the presented procedure would still remain valid, with the notable advantage that the attitude of $\mathcal{F}_3$ with respect to $\mathcal{F}_1$ at the initial instant would be provided, and therefore the quaternion $\mathbf{q}_{31}$ is initially known, so Equation (2.54) can be used all along without the need to define an ad-hoc arbitrary reference frame.

2.2 Numerical Simulation of Relative Motion

A simulation framework is constructed to compute the states of both target and chaser satellites, as well as all other elements of the environment, obeying the dynamical equations described previously. In this section, the construction of the simulation framework is described, with the aim of matching the real-world situation as closely as possible. This numerical simulation is used to construct a scene at each point in time, from which realistic image data is visualised following the approach in Section 2.3.

2.2.1 Simulation Framework

The simulation consists of permuting the states of the chaser and target satellites according to the translational and rotational equations of motion. These states are then passed to the next stage to perform the visualisation. Several parameters control the simulation case; varying these parameters between simulations enables the construction of large datasets with differing conditions, all of which are possible cases of on-orbit relative motion.

In order to provide further control over the data, the software has two separate modes of operation. The full simulation mode obeys all the dynamics of relative motion described up to now, accurately visualising the inspection stage during on-orbit rendezvous. However, there also exists a partial simulation mode which provides the ability to restrict certain parameters of the simulation. Under this operation mode, the software does not simulate the real-world situation, instead allowing the user to generate output data under certain specific conditions. For example, different test cases can be constructed with only the lighting direction or rotation axis varying between them, enabling the analysis of a method's response to different, known, situations.

2.2.1.1 Simulation Case

It is desired that the generated image data be used in approaches for classifying parameters of the debris target. As such, the simulation case consists of the inspection stage of an active debris removal mission, in which the chaser orbits about the debris target enabling it to view the object from several directions. For simplicity, the target is considered to be in a circular orbit such that the Clohessy-Wiltshire equations may be applied, with the chaser in the football orbit given in Section 2.1.2.3.

The other objects in the environment that we consider are the Sun and the Earth. The simulation uses the target-centred LVLH frame as its reference frame, so the states of both of these objects will be moving during the simulation. Because we are assuming a circular orbit, the motion of the Earth can be simply represented as a constant rotation, with period equal to the orbit period of the target satellite.

On the other hand, the motion of the Sun is more complex, as this is affected by the position of the target in Orbit as well as the motion of the Earth. For this case, the sun-pointing vector is calculated using the PSA algorithm provided by Blanco-Muriel et al. (2001). This algorithm is able to determine the direction of the Sun given the geographical position and time. We make a reasonable assumption that the vector relative to a satellite in Earth orbit is equivalent to the vector relative to an viewpoint on the surface of the Earth, due to the large difference in magnitude between the orbit height and the distance to the Sun.

2.2.1.2 Control over Simulation Parameters

The parameters of the simulation can be defined in a configuration file, at runtime, or randomised during data generation. This allows for significant control over the generated data. Each of the two operation modes will have different parameters to adjust. As discussed, the full simulation captures the true states of the satellites under relative motion, whereas the partial simulation is useful for generating data under certain conditions but does not necessarily undergo realistic on-orbit dynamics.

In the full simulation mode, the orbit parameters of the target must be supplied along with the initial relative states of the the target-chaser system. As the orbit is assumed to be circular and the target is in a known monitoring trajectory, it is sufficient to define only the orbit height and initial displacement between the satellites. The rotational state of the target satellite can then be provided in the form of an initial attitude and an angular velocity. A start time can also be provided, to prevent only the same section of the orbit from being simulated in each case.

During a partial simulation, the states of both satellites are defined at initialisation, including relative positions and angular velocities. Instead of computing the equations of motion and the states of satellites, the Earth and the Sun, these values are provided explicitly, enabling full control over the generated imagery.

In both simulation modes, there are also a number of further parameters which affect the visualised image data. These include the field of view of the camera, capturing rate and duration. In addition, the 3D model of the target satellite can be varied between simulations. These are discussed in more detail in Section 2.3.

2.2.2 Implementation of the Equations of Relative Motion

As discussed in Section 2.1.2, there are a number of different models for solving the relative orbital motion problem, with advantages and drawbacks to each. Models with a direct analytical solution, such as the Clohessy-Wiltshire equations and state transition methods, are better suited to individual long-term predictions than iterative algorithms such as the Tschauner-Hempel equations. Additionally, time-dependent models have the advantage that there is no need to solve Kepler's equation at each time step; though equations in true anomaly tend to perform better over longer timescales. However, since it is necessary anyway in our case to determine the state at several time steps, the drawbacks of iterative methods become less pronounced.

Despite the simulation test case assuming a circular orbit, thus enabling the use of the CH equations, we implement each of the three models discussed in Section 2.1.2, using both STM and numerical integration methods. Thus the accuracy of the different methods can be compared, and the simulation can be extended beyond the initial test case described above.

The numerical integration method attempts to solve the differential equation

$$x'(f) = g(f, x(f)), \tag{2.56}$$

assuming true anomaly is the independent variable. In the case of relative orbital motion, this differential equation is expressed as

$$\boldsymbol{x}'(f) = A(f)\boldsymbol{x}(f), \tag{2.57}$$

where $\boldsymbol{x}$ is the state vector of the chaser satellite,

$$\boldsymbol{x} = [x, y, z, x', y', z']^T. \tag{2.58}$$

For the Tschauner-Hempel equations, the matrix A is found from Equation 2.14:

$$A(f) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{3+e\cos f}{1+e\cos f} & \frac{-2e\sin f}{1+e\cos f} & 0 & \frac{2e\sin f}{1+e\cos f} & 2 & 0 \\ \frac{2e\sin f}{1+e\cos f} & \frac{e\cos f}{1+e\cos f} & 0 & -2 & \frac{2e\sin f}{1+e\cos f} & 0 \\ 0 & 0 & \frac{-1}{1+e\cos f} & 0 & 0 & \frac{2e\sin f}{1+e\cos f} \end{bmatrix}. \tag{2.59}$$

In order to solve the differential equation, Euler's method is employed. This replaces the derivative of x with the finite difference approximation

$$x'(f) \approx \frac{x(f+\delta f) - x(f)}{\delta f}, \tag{2.60}$$

where δf is the step size. For iterative techniques, the size of the step impacts the accuracy of the simulation, where smaller step sizes lead to better approximations. This equation can be rearranged to represent the value of x at the next step in terms of x and x' at the current step:

$$x_{n+1} = x_n + \delta f x_n' = x_n + g(f_n, x_n). \tag{2.61}$$

Substituting in Equation 2.57 results in the following expression for numerical integration of the relative orbital motion equations:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_{\boldsymbol{n}}[I + \delta f_n A(f_n)]. \tag{2.62}$$

The notation δf_n is used because each step may not have the same change in true anomaly. For example, this is the case when converting from an even division in time to a step in anomaly, for a non-circular reference orbit.

The Euler method is a first order method, which means that the global truncation error is roughly proportional to the step size. The error in the numerical integration can be reduced by employing more complex algorithms, for example higher order

Runge-Kutte methods or multistep methods such as Adams-Bashforth. However, the implementations of these methods are more costly.

The STM is much more simple to implement and, for time-dependent methods such as the Melton's STM or the CW equations, is simply taken as

$$\boldsymbol{x}(t) = \phi(t, t_0)\boldsymbol{x}(t_0). \tag{2.63}$$

2.3 Visualisation of On-Orbit Proximity Operations

The numerical simulation described in the previous section provides the states of all relevant objects in the environment, at each point in time. In this section, the state information is used to construct the scene as viewed by the chaser satellite, and thereby generate realistic on-orbit image data. This visualisation framework is capable of constructing large datasets for use in training deep learning models, or for validating visual navigation approaches in this environment.

The generated datasets contain visual and depth images of the view of a tumbling target satellite as seen by a chaser spacecraft. The data is labelled with the position and pose of both satellites, along with class labels, bounding boxes and surface segmentation labels, in order to facilitate a number of different deep learning applications.

The visualisation must accurately capture the space environment, including lighting conditions and background. In addition, in order to ensure that the deep learning network does not learn to classify only a specific satellite geometry, several different debris models must be included in the data. An example of the image data generated by the framework is shown in Figure 2.6.

2.3.1 Visualisation Framework

The visualisation uses Blender[2] as the backend for modelling the image data. Blender is a widely used, open source 3D creation suite, supporting the entire pipeline from both modelling to rendering image scenes. Its ability to integrate with other modelling software also enables the simple use of externally sourced 3D models.

One of the most important features of Blender is the existence of a Python Application Programming Interface (API). This enables the use of Python scripting to easily generate simulation environments and to construct large, varied datasets with different targets and environment parameters quickly. Our simulation framework

[2]https://www.blender.org

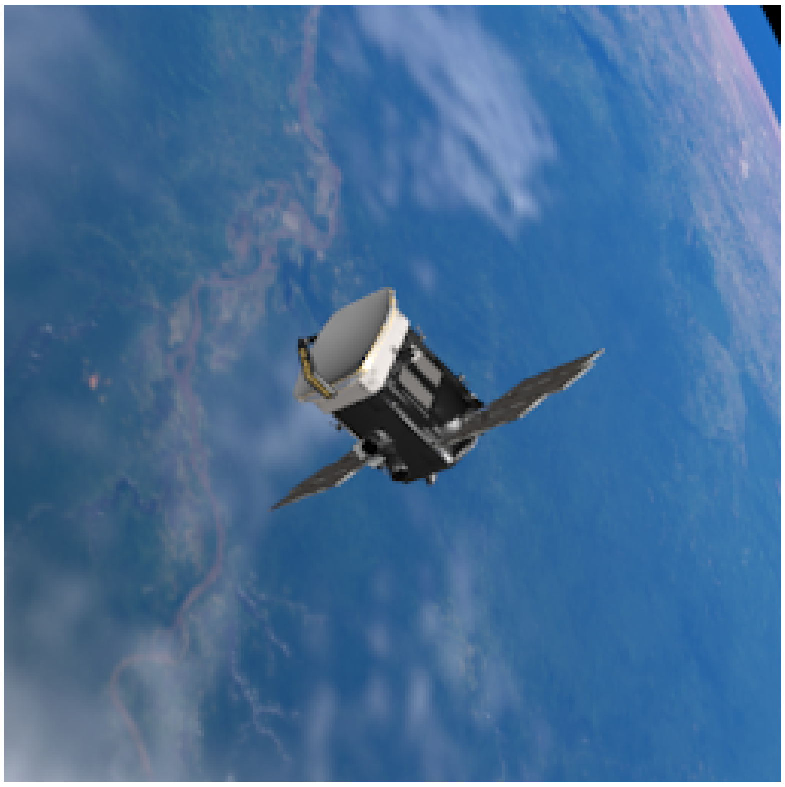


FIGURE 2.6: An example of the image data generated by the on-orbit rendezvous visualisation framework.

uses python scripting to construct and render scenes of on-orbit proximity operations, with configurability provided by the simulation parameters.

2.3.2 Lighting Conditions and Space Environment

The unique lighting conditions in space present a key difficulty for optical navigation algorithms, due to the high contrast and reflections, so these must be emulated in the simulation framework. We consider the Sun to be the sole light source, which can be simulated as a distant point source. The direction of the light source changes over the course of the simulation, due to the position of the target in orbit, the direction in which the chaser is facing, and the motion of the Earth. This is calculated using as detailed in Section 2.2.1. For simplicity, we do not consider the reflected light from the Earth or the Moon as these effects are outweighed by the incident light from the Sun. In addition, the chaser is considered to be fitted with a small light source, as is often the case in reality in missions where optical cameras are used. This can aid in situations where the lighting on the target is poor.

The other element of the space environment to be implemented is the Earth. The Earth will be in the background of a large proportion of the images and this can have a significant impact on the accuracy of vision-based navigation algorithms, as

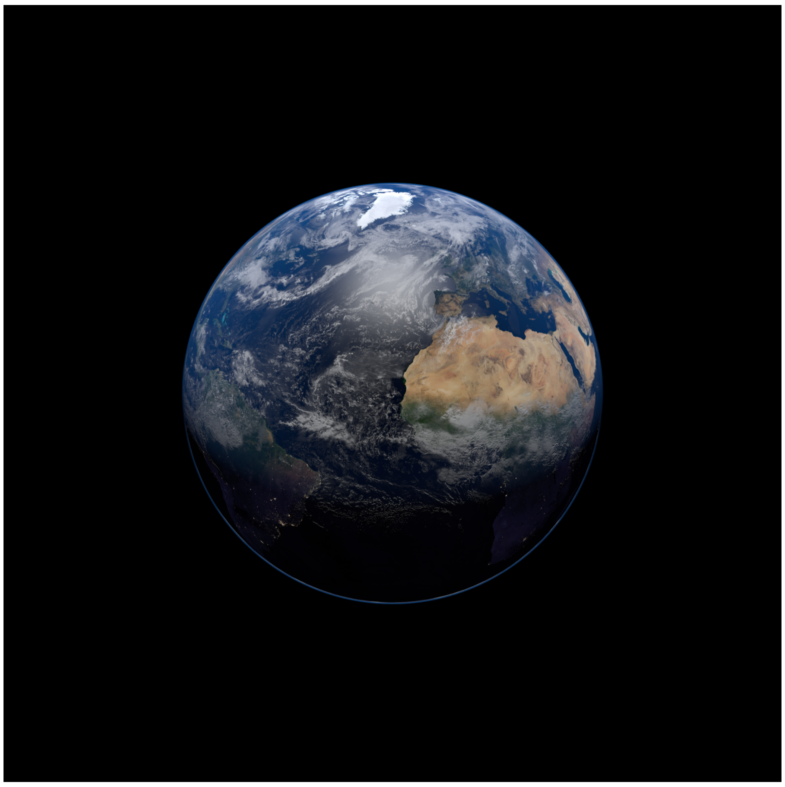


FIGURE 2.7: The Earth, visualised within Blender.

demonstrated in the RemoveDebris mission (Aglietti et al., 2020). Thus, it is crucial that this is captured within the data to ensure the performance of the algorithm in both cases, with and without the Earth visible in the background of the images.

The image textures used in constructing the model of the Earth are taken from NASA Visible Earth's[3] Blue Marble collection. This collection provides high-resolution satellite images of the Earth collected during different seasons, along with topography and cloud data. This data has been used to construct a realistic visualisation of the Earth, including landscape textures, cloud layer and atmosphere. The visualisation is shown in Figure 2.7, though it should be noted that, in most cases, only a small portion of the Earth will be visible in the images. In addition, both day and night time visualisations are included, depending on the incident light from the Sun. This further increases the variation of the data, thereby ensuring the generalisation of the navigation methods to various potential situations.

2.3.3 Target Satellites

Again, NASA provides valuable resources to aid in 3D modelling, in the form of a database of 3D models[4]. This contains a large number of satellite and rocket 3D

[3]https://visibleearth.nasa.gov/

[4]https://nasa3d.arc.nasa.gov/models

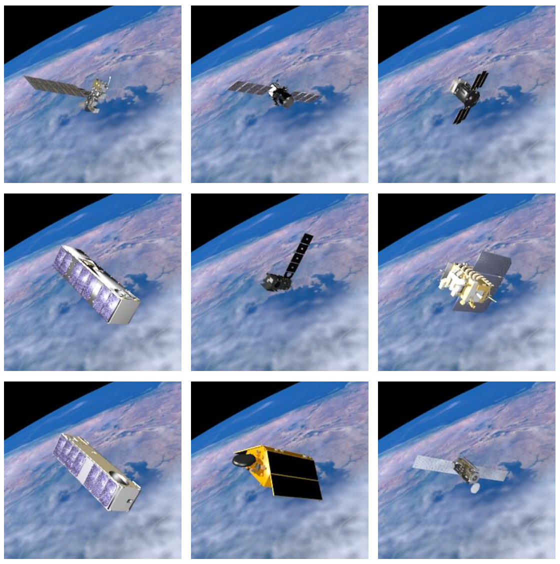


FIGURE 2.8: The satellite 3D models available for simulation.

models for visualisation or 3D printing. Being an open source software, Blender integrates well with other modelling environments, allowing these satellite models to be imported easily into the simulation environment. Therefore, we have made good use of this resource by incorporating a number of the satellite models into the visualisation. Figure 2.8 demonstrates a number of these models. Targets have been selected such that there is significant variation between the geometries, which encourages our algorithms to have better generalisability.

This data is designed for use in applications to spacecraft navigation, particularly aimed at ADR. For these tasks, it is desired that the data consists of targets which represent common targets in ADR missions, such as rocket upper stages. However, due to the limitations of the available 3D models, the satellite targets are instead more general and varied. As discussed, this should enable the training of models which generalise well to any target geometries; however, for analysing the applicability of the model specifically to ADR problems, it may be useful to extend the simulation software by including specific debris models.

To this end, the simulation framework is developed such that further models can be easily added at a later date, for example to enable the generation of data prior to a mission targeting a known satellite. Adding the 3D model file to the relevant directory is sufficient to automatically integrate the model as a debris target within the visualisation, including generating labels datasets using the filename of the 3D model.

One of the labels used for the output data, as discussed in the following section, is a classification of the different surfaces on the target. In order to generate these labels, new images must be rendered under identical conditions, with the pixel intensity corresponding to the surface type: a white pixel where the visible surface matches the classification, or black elsewhere. These are constructed by replacing the actual satellite model with a separate model with the same geometry, with black or white surfaces accordingly. Thus, new models must be constructed for each satellite and each surface label, in order for these satellites to be included within the surface classification datasets.

2.3.4 Generating Image Data

The output of the simulation not only provides the dynamical states of both spacecraft, but also simulates the outputs of an optical camera, a RGB-D sensor and a LiDAR, all onboard the chaser. Optical and RGB-D sensors both provide a colour image, with the latter also providing the depth at each pixel. A LiDAR sensor also measures the depth of an image, with a lower resolution which depends on the sampling rate of the sensor; this data is generated from the depth image by sampling pixels at a given sampling rate, and returning a point cloud containing sparse 3D information. These forms of sensor data are illustrated in Figure 2.9. The different data formats open up a greater number of potential applications for the generated datasets.

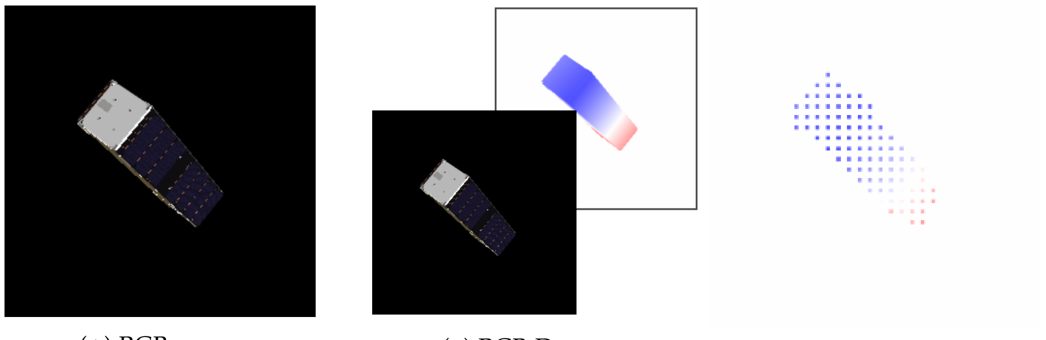

(A) RGB sensor (B) RGB-D sensor (C) LiDAR sensor

FIGURE 2.9: Visualisation of different simulated visual sensors.

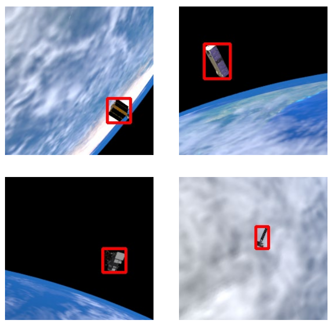
The visual sensors are accurately simulated using a perspective projection. The format of the generated image data can therefore be selected using the sensor parameters, such as the image height and field of view; these parameters can be matched to a real camera to be used in a mission, in order to generate more realistic images for a specific use case. The capture rate and simulation duration can also be defined to select the number of images and the size of the time step between them.

2.3.4.1 Data Labels

In order to use the datasets to train deep learning models, the images must be labelled with the relevant information that the model is attempting to acquire. We wish to enable a large number of potential applications with these datasets, so the data is annotated with several different labels.

Firstly, the dynamical states of both satellites, including attitudes and relative positions, are provided at each instance. This information is important since it will often be the end goal of many algorithms for relative navigation. Thus, the data can be used to train a neural network which aims to track either rotational or translational motion, or equally, to validate an algorithm for the same task.

In addition, some more complex labels are included. Two tasks are identified as promising further applications of deep learning in VBN for rendezvous; these are satellite detection to track or crop the image around the target, and surface segmentation for identifying the best surface for grasping or impaling. These are discussed in more detail in Chapter 3. Therefore, the data is also annotated with the required labels for these cases. These are: the coordinates of a bounding box around the image, the satellite class name, and the surface labels which are simply binary images. These labels are illustrated in Figure 2.10.


(A) Bounding box labels

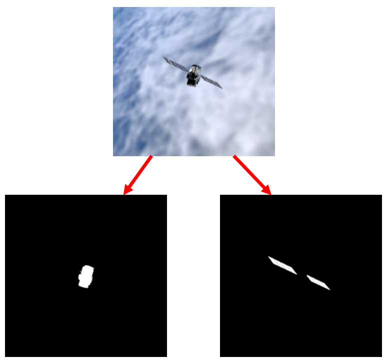

(B) Surface labels of the satellite body and solar panels

FIGURE 2.10: Examples of some of the data labels.

2.3.4.2 Constructing the Datasets

The visualisation software makes it easy to construct large datasets, consisting of many different simulation cases and thousands of image files. These can be constructed by providing the parameters for each simulation and appending the results to the dataset. Alternatively, the software can construct any number of simulations, randomising the parameters in each case, thereby enabling the rapid generation of large, varied datasets.

The generated datasets consist of a directory of image files, each of which has a corresponding data label containing all the information described above. The data is then divided into three subsets, referred to as the train, validation and test subsets.

This allows the data to be used in different situations; for example, the model can be validated using data which is different to that which it has seen during training. The fraction of the data contained in each subset is another controllable parameter.

For the specific task of estimating the change in attitude between image frames, which is the goal of the following chapters of this work, the datasets must be further processed. The output images must be collected into a format which can be used in training the neural networks. These datasets consist of a collection of pairs of successive images, with a small time step having passed between the two images. Each image pair must then be labelled with the rotational state of the target satellite across this time step, which is the desired output of the algorithm. This label is the observed relative rotation $\hat{\mathbf{q}}$, as in Equation (2.51).

For these datasets, we select the position and attitude parameters of the target and chaser such that there is a large variation in relative angular rate between simulations, from 0.5° per step up to 7°. In order to increase the size and variation of the datasets, we also collect image pairs with a gap of 2, 3 or 4 times the time step, enabling us to investigate the response for up to 28° rotations in a single time-step, although the higher end is likely to be difficult to solve.

A dataset of approximately 60,000 image pairs is used as training data for the following deep learning models. In this dataset, several simulations are run to generate the data, between which the debris targets, rotations speeds and illumination conditions are varied. Separate, smaller datasets are created for testing and validation, with different simulation parameters; this data is not seen by the model during training. A separate test dataset is generated for each satellite target, with all other parameters remaining constant between the datasets for more fair comparison.

2.3.5 Effectiveness in Representing Real-Life Conditions

The key aim for the generated datasets is that they may be used to train deep learning-based models, to analyse their performance and suitability to the task and environment, and thereby enable their application in real missions. Therefore, it is important that the data effectively represents the challenges of the space environment, including lighting conditions, sensor limitations and target geometries. We discuss here the extent to which the datasets capture these conditions, and ways in which they could be further improved to better represent these challenges.

Due to the control that the simulation framework has over the realistic lighting sources, surface parameters and visual sensors, it is able to capture some of the key challenges due to the lighting conditions, such as reflections from different surface materials and the lack of ambient lighting. In addition, we also capture the different possible backgrounds depending on the orientation of the chaser, target and Sun. This is crucial for ADR missions, in which the co-orbiting satellites will be undergoing relative motion throughout. Different potential on-board visual sensors are simulated, including both RGB and LiDAR sensors, with adaptable parameters for the ability to imitate specific real-world cameras. These also obey by provided limitations in capture rate given by the power availability on-board. Finally, we represent different geometries of target satellites, allowing for training models which generalise to different potential debris targets without the need for a-priori knowledge of them.

However, there nonetheless remain a number of improvements which could be made in order to more accurately represent the real-world environment. Firstly, the most difficult lighting conditions have not yet been considered, for example effects such as lens flare or the Sun being within the camera view. These are tasks which have been noted as high priority for further work. In addition, as noted earlier, the addition of different satellite models - particularly upper stages - would make the data more realistic for ADR missions. For this reason the framework has been constructed to allow easy additions of new target objects. Even despite these additions however, it is likely that some real-world data may be required for fine-tuning models prior to their use in real missions.

2.3.5.1 Generalising to Real-World Data

Synthetic data presents a number of advantages: 1) it enables the use of deep learning in environments where real data would be very time consuming or difficult to process; 2) the size of datasets made up of synthetic data can be much larger than would be achievable with real data, due to the speed of generation; and 3) by simply changing certain parameters of the simulation, it is possible to generate new datasets under different conditions to further test the model. In our simulation framework, all parameters are contained in an editable configuration file, which can be either specified precisely or randomised before generating a dataset of tens of thousands of image pairs, a task which would be unrealistically time-consuming for real data.

However, the use synthetic data may lead to difficulties at a later stage, when aiming to use the model to generalise to real images. The synthetic data will not match exactly to real data – in particular, the camera and sensors will not be as accurate as in the simulation, where the depth of each pixel is known precisely. In addition, it is difficult to assess to which extent the synthetic data matches the real data closely enough so the network trained with synthetic data would perform equally well for in-flight operation use.

In this regard, the capability of the model to generalise to real data could be improved by distorting the data (Sundermeyer et al., 2018), which encourages the model to learn

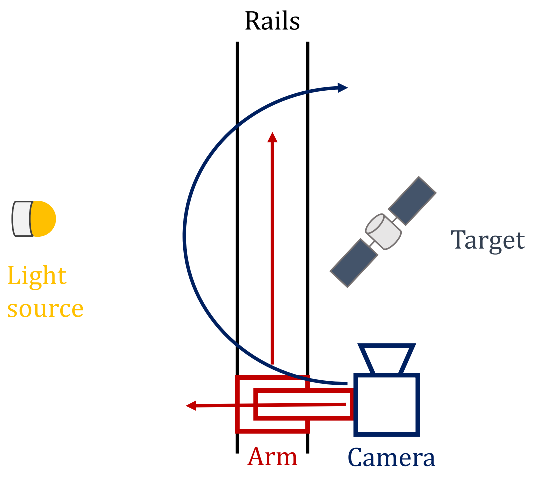


FIGURE 2.11: The setup of the lab environment for capturing realistic image data.

the object features and disregard the inaccuracies in the sensor. However, in order to verify the generalisability of a method, the best approach is to use real image data.

To this end, a plan has been detailed to enable the acquisition of realistic data in a laboratory environment. The aim of this lab environment is to generate images of a rotating debris target from a moving viewpoint. The layout of the realistic simulation environment is detailed in Figure 2.11.

The lab environment uses a single light source to act as the Sun, which should be the only source of light. The target is a 3D printed model of a satellite, identical to one of those in the simulation detailed previously. A camera is mounted on a six degree-of-freedom robotic arm, which in turn can move on a set of rails. This enables the motion of the camera in a small arc around the target, emulating a section of the football orbit. The target itself is rotating by the use of a second robotic arm. Since all distances and motion is known, it is simple to label the resultant image data with the required information. The same experiment may be repeated with a LiDAR sensor in place of the visual camera to acquire depth information under the same conditions.

2.3.6 Access to Simulation Software and Datasets

The simulation software and a number of training and test datasets are provided for use in future research. The software is available at `https://Ben-Guthrie.github.io/satvis`, along with instructions for setting up the environment and generating and using the datasets. In addition, we host several pre-made datasets, including those used in the analysis in this thesis, at `https://www.kaggle.com/benguthrie/inorbit-satellite-image-datasets`.

Chapter 3

Applications of Deep Learning to Vision-Based Navigation

This chapter briefly analyses a few of the most significant challenges which we believe can be mitigated by the application of machine learning technologies. For two of these potential applications, example implementations of these technologies are provided along with a discussion of the performance achieved. As this is simply a quick demonstration of potential applications, the implementation and analysis is less in-depth than in the following chapters; however, it is nonetheless sufficient to obtain promising results, suggesting that further research in these areas will be valuable.

A number of the remaining technical challenges stem from the GNC system, due to the difficulty of achieving fully autonomous visual-based navigation. Yamamoto et al. (2019) define the three biggest challenges for the visual navigation system as the following:

- The relative navigation around the target is difficult due to the lack of dedicated markers or retroreflectors.
- The target may be tumbling, requiring the synchronisation of rotational motion prior to capture.
- There is no dedicated docking mechanism on the target.

A key aim of this work has been the investigation of the potential improvements which can be achieved by adapting machine learning technologies to this problem. Therefore, for each of these three challenges, we provide a machine learning approach which can aid the GNC system in overcoming it. These can be used alongside or as an alternative to other navigation technologies, potentially resulting in a modular system in which the output of one element can aid in the computation of another.

The first challenge listed above sets a requirement for an accurate relative navigation system for all stages of rendezvous. For close-range rendezvous, a LiDAR system is likely to provide the best accuracy for relative position estimates. However, by employing techniques from object detection - which is a central topic of machine learning research - the target can be detected and tracked from visual images at a further distance, during the approach phase. This technique can locate a satellite target and, if desired, classify it into different categories of debris. The same approach could also be used in close-range rendezvous to detect certain objects on or elements of a spacecraft, such as an engine, for more precise relative navigation.

The second challenge is the focus of the following chapters of this work. The ability to accurately track the attitude motion enables the required mitigation strategies, such as detumbling technologies or the synchronisation of angular rates. As this is not a task to which existing solutions can be effectively applied, it was necessary to develop an original method for estimating the rotational motion of an unknown target. All discussion of this problem is thus relegated to Chapters 4 and 5, which detail the development of this method and analyse its suitability for this task.

Finally, we propose a strategy for mitigating the absence of a docking mechanism. Clearly, this is less of a challenge in some debris removal approaches, such as net or harpoon capturing, when there is no need for docking with the target using a dedicated docking mechanism. However, it is nonetheless important to define a target location on the satellite for making contact which minimises the risk of breakups. In most cases, if the target is unknown or uncooperative, it is useful to have a method of analysing the surfaces of the target to avoid fragile surfaces, extract optimal docking locations, or or determine the makeup of the satellite. We propose a method based on the concept of image segmentation, which uses convolutional neural networks to segment and label the different surfaces on the object, using visual images.

The simulated datasets in Chapter 2 are constructed with the goal of enabling a large number of different machine learning applications. Therefore, these same datasets can be used to train both of these new approaches. This work could potentially result, in the future, in a modular GNC system consisting of separate elements, all of which can be trained on the same data, running on an on-board GPU system, and passing the outputs between themselves. For example, the prediction of the relative motion can be combined with the surface segmentation to identify a location for making contact, predict the evolution of its location in time, and navigate towards it.

3.1 Satellite Detection and Tracking

Object detection is one of the central problems in computer vision, and one which has seen a large amount of research in the past two decades (Zou et al., 2019). Since the

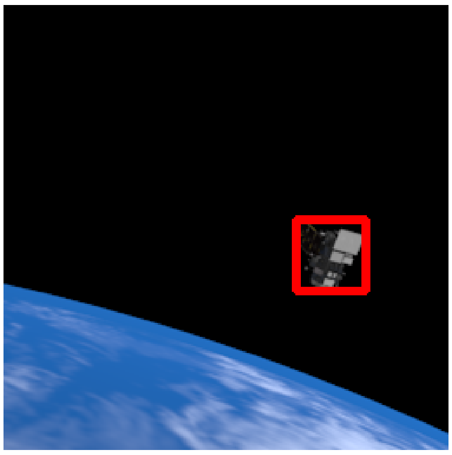


FIGURE 3.1: A bounding box around a satellite object in a scene.

breakthrough achieved by Krizhevsky et al. (2012) applying deep CNNs to the problem, the research has focused almost entirely on CNN-based models. These approaches have yielded unprecedented advancements in the field, providing far better results than conventional methods on all validation datasets, such as ImageNet (Deng et al., 2009) and MS-COCO (Lin et al., 2014).

Object detection is an extension of the image classification task. Whereas image classification involves simply classifying an entire scene into one of several classes, object detection requires both classification and detection. This means that the scene could contain multiple objects, each of which should be detected and labelled by the model. Typically, the detection involves finding the four corners of a bounding box around the object, such as that in Figure 3.1 for example. This corresponds to the possible location of one of the classes of objects which the model has been trained to look for.

The ability to find the satellite target in an image is of particular use in the approach stage of an ADR mission. In particular, the location of the bounding box in the view frame can enable an estimation of the relative position of the target. Furthermore, tracking the object over successive frames provides an estimate of the relative translational motion. This information is useful for relative navigation during the approach, up to the close range rendezvous stage. As mentioned previously, at close ranges the bounding box around the whole object will be less useful for relative navigation purposes, particularly if only a small segment of the satellite is visible. Instead, a LiDAR based approach is likely to be best. However, object detection may still be useful in this stage, for example to detect common elements of a satellite, such as an engine, to aid with the docking manoeuvre.

In addition, it is useful to classify debris objects prior to capture; for example, the optimal capturing approach may depend on the type of object. CNNs have already been applied to classify objects in space, for example using brightness measurements (Linares and Furfaro, 2016). However, the chaser spacecraft has an advantage in being a short distance from the target and having access to full optical data, allowing for more precise classification beyond what is possible from light curves or brightness data.

3.1.1 Implementation

In order to demonstrate the potential of these technologies, we implement an object detection CNN and use it to detect and track satellites in image data. Due to this being a demonstration, we simply use a pre-trained model which is finetuned on the satellite image data. The datasets generated previously are used to train and analyse the model, since these already contain all the necessary data and labels.

3.1.1.1 Pre-Trained Models

Due to the large amount of research in this domain in recent years, several models have been developed which show impressive results on object detection tasks. It makes sense to simply adapt one of these pre-trained models to this problem, since the task is very similar to what they have been trained on. This significantly simplifies the implementation and reduces the training time, since the model only requires finetuning on the new datasets.

There exist several freely available pre-trained object detection CNN models. These typically fall into one of two categories. The first is based on two-stage detection, in which a set of region proposals are extracted, each of which possibly contain an object. These are then passed to a classifier CNN to predict the presence of an object within each region. This approach was first introduced by the Regions with CNN features (R-CNN) model (Girshick et al., 2014) and has been improved upon since then, more recently in Faster R-CNN (Ren et al., 2016) which also uses a CNN for the region proposal and is capable of near-real time operation at 17 fps (Zou et al., 2019).

The second category of object detectors are one-stage detectors. First proposed by Redmon et al. (2016) in You Only Look Once (YOLO), this approach simply applies a single network to the whole image once. The network simultaneously divides the image into regions and makes predictions on each region. Unsurprisingly, this model is much faster than the two-stage approach, although it was found to lead to a slight drop in localisation accuracy. The YOLO model has since seen several improvements, with Redmon later releasing YOLOv2 and YOLOv3 implementations. Other researchers have further developed the approach, most recently with YOLOv5 (Jocher et al., 2021).

Since real time performance is critical for space applications, YOLOv5 is selected as the best model for our case. It also has the advantage of being quick to implement, train and use, because it has been designed with this in mind. YOLOv5 has several models with different network architectures; we use YOLOv5s which is a small and fast version of the network.

3.1.1.2 Training and Datasets

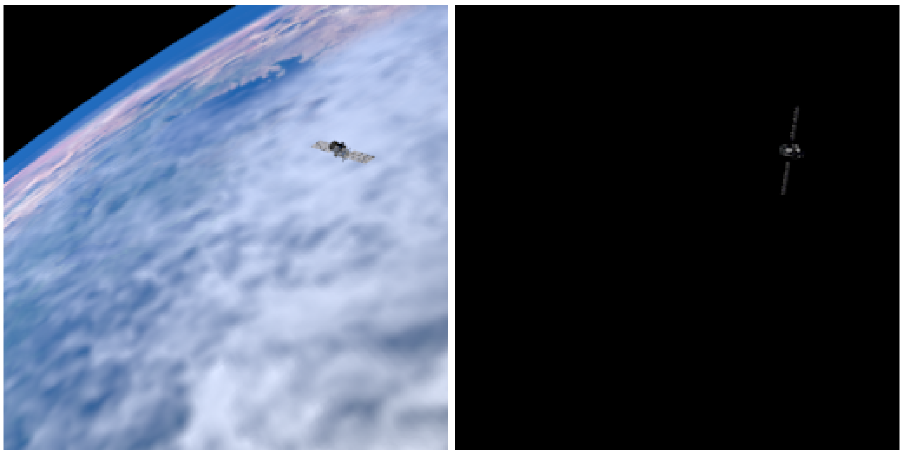
The model is finetuned on the training datasets to learn to detect satellite objects. This uses the bounding box and class labels generated in Chapter 2 as the target. However, there are two options for training the model, depending on the desired output. If the goal is simply to detect and track any (possibly unknown) satellite object, then the model can be trained with a single class label, "satellite" which corresponds to any satellite target. On the other hand, the model could also be used to both detect and classify different types of satellite, by using multiple labels in training. This latter method would be less likely to detect a previously unseen satellite though, as it will not fall into any of the learned classes. Therefore, the best usage may involve a combination of both single-class and multi-class approaches.

The ground truth class labels and bounding boxes, which provide the targets for the model, are generated by our simulation software. Due to the fact that synthetic data is used, these labels can be generated automatically, thus reducing the time consumption of constructing the datasets. The corners of the bounding box are determined by converting the target to a mesh, projecting it into the camera space, and taking the maximum and minimum value in each axis. These operations can all be performed within the Blender pipeline.

Beginning with the pre-trained weight values, the full training dataset is then used for finetuning the weights. For this, we use a dataset of 12,400 images. Due to the relatively small size of the network, this finetuning is very fast. The finetuned model is then able to be implemented for satellite detection and tracking with no further modification necessary. In the following section, the performance of this model is analysed on the test datasets.

3.1.2 Results and Discussion

Following the finetuning of the YOLOv5 model on our satellite datasets, we analyse the performance of the model by applying it to a number of different test cases. For each case, a test dataset of 30 images is constructed. These are taken from different points in an orbit observing a satellite target. We also investigate the two different



(A) Earth in the background (B) Poorly illuminated target

FIGURE 3.2: The two test cases investigated in the analysis.

outputs of the model: the bounding box and the satellite label, as these each have different use cases.

In order to verify the applicability of these techniques to relative navigation, they should be robust to different conditions which could occur during a debris removal mission. Therefore, the test cases are formulated as follows. For each of the satellite targets available in the simulation software, two datasets are constructed. First, we investigate the effects of background noise, by including the Earth in the background of the images. The second dataset does not include the Earth, but instead looks at the case where the lighting is poor, with the target barely illuminated. All other parameters of the simulation, including the relative position and orientation of the observing chaser satellite, are kept constant between the datasets. Figure 3.2 shows example images from each of the two test cases, for the Calipso satellite target.

We begin our analysis with a qualitative investigation of the output of the network. Looking at a single test case, Figure 3.3 shows the predicted bounding box about the satellite target at different points in orbit. As stated earlier, for consistency across the results the viewing angle and relative position are kept constant. In most cases, the bounding box appears to accurately detect the location of the satellite in the image. However, as can be seen in Figure 3.3, there are occasions where the solar panel blends in with the background and so is not picked up by the model; this could also be caused by the fact that all targets in the data consist of a somewhat similar body, while the shape and details of the solar panels differ considerably.

It is possible to numerically quantify the accuracy of a bounding box prediction. For this, the Intersection over Union (IoU) evaluation metric is most common. This is, quite simply, calculated as simply the region of overlap between the predicted and ground truth boxes, divided by the total region covered by both. So, if the two boxes are identical then the IoU of the prediction will be 1, whereas if only half of each of the

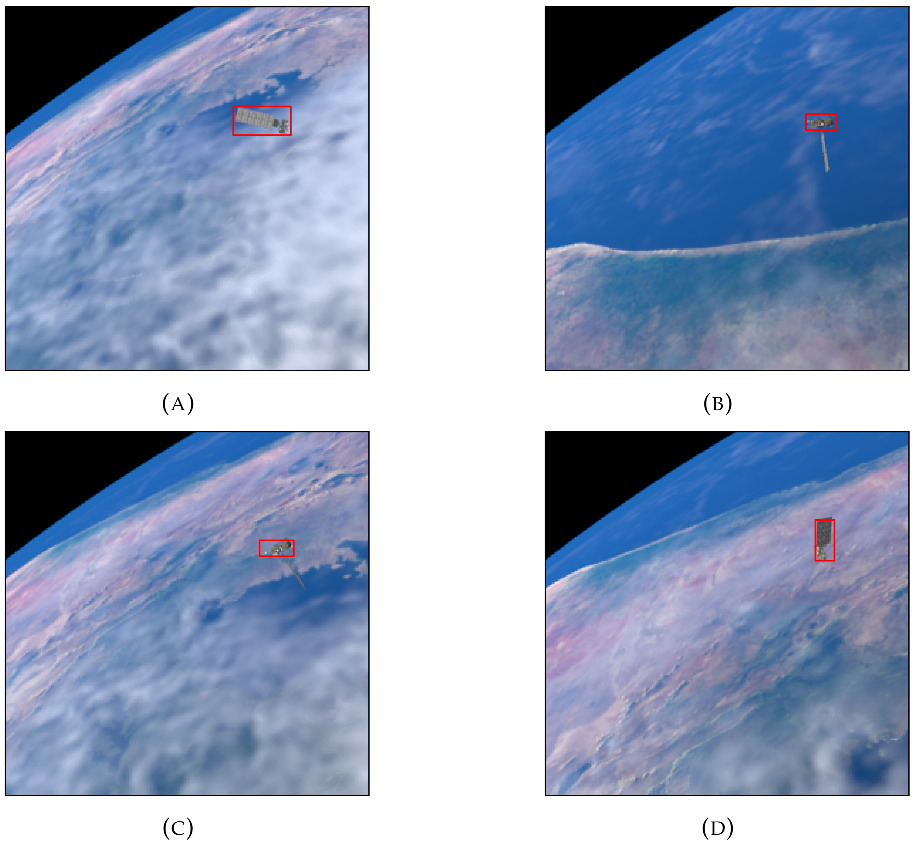


FIGURE 3.3: Predicted bounding boxes for various images from a test dataset. The IoU scores for each are: (A) 0.832, (B) 0.220, (C) 0.206, (D) 0.731.

boxes intersect, the value is 1/3. The IoU values are shown beneath each of the predictions in the figure.

However, it is important to note that it is possible that the prediction is in fact more realistically accurate than the label. This is because the labels are given in terms of an integer number of pixels, whereas the optimal bounding box may be better defined on a scale smaller than a pixel, which the model is capable of doing. In fact, by investigating the output of the model, we find that the coordinates of the bounding poxes for predictions with an IoU greater than 0.9 tend to within a rounding error of the true label coordinates, but may actually be more accurate.

The performance of the model is analysed by looking at the mean IoU over each test dataset. This allows us to compare the performance with different satellite types and under different conditions. Figure 3.4 shows the results of such an investigation, giving the mean prediction accuracy for each of the discussed test cases. This mean value is computed over each of the test datasets.

A small drop in accuracy is noted on the datasets containing the Earth in the background. As we showed earlier, this is caused by parts of the satellite becoming difficult to distinguish from the background. This is most common for satellites with

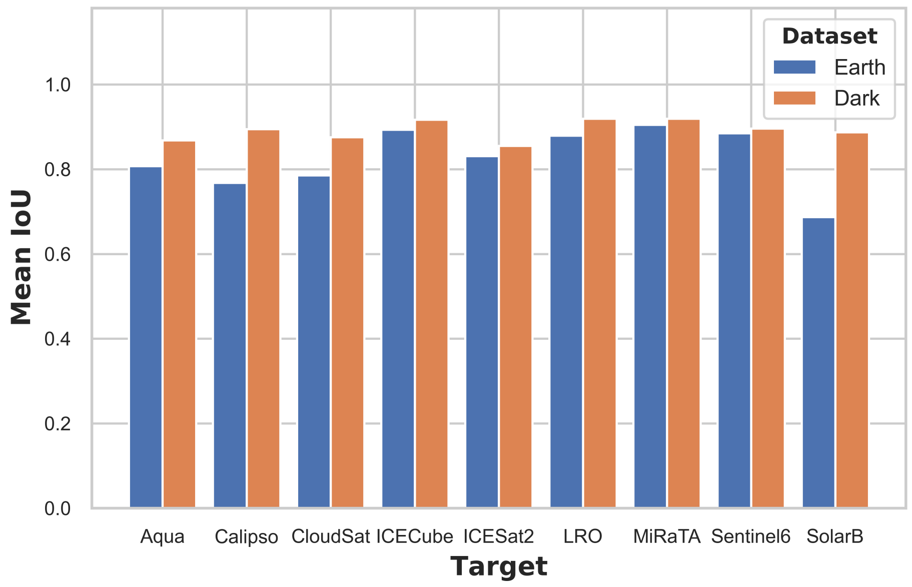


FIGURE 3.4: Mean IoU value for the different satellite targets under the two different conditions.

long, thin solar panels, such as Calipso and SolarB, which can become very difficult see at certain orientations. However, the loss in accuracy is lower than in our previous investigations into rotation estimation. This is because the network is much deeper in this case and so deals better with more complex images. On the other hand, it performs well under poor illumination conditions, which is also important. However, it would still be useful to conduct further analysis under more challenging conditions, taking into account reflections or camera glare, for example. Finally, we note that the average IoU in all cases remains below approximately 0.9; again, this could simply be down to the labels not being entirely accurate.

Another important quality to investigate is the model's ability to generalise to unseen targets. This will enable the approach to be used for relative navigation and tracking with any target, without requiring retraining of the network. Figure 3.5 illustrates the results of this investigation. In this experiment, the accuracy is measured for each of the satellite targets using two different models, one of which was trained on a full dataset containing all targets, the other of which was trained on a reduced set containing only those targets with names in orange.

We do note a slight drop in accuracy for those targets which are not present in the training data, but this is not a significant difference. The largest drop occurs with the CloudSat target, possibly because this is most different from the other satellites in the dataset. Even so, this result suggests that, even with only six different satellites in the data, the machine learning-based model is capable of learning what a satellite looks like and accurately detecting and tracking unseen satellite targets in images. On the other hand, this investigation has been relatively simple and has not considered the

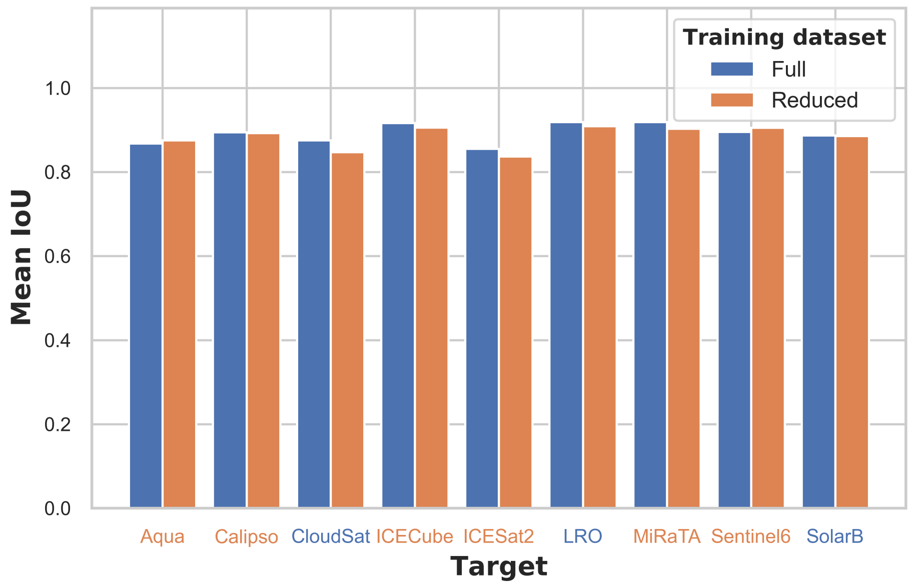
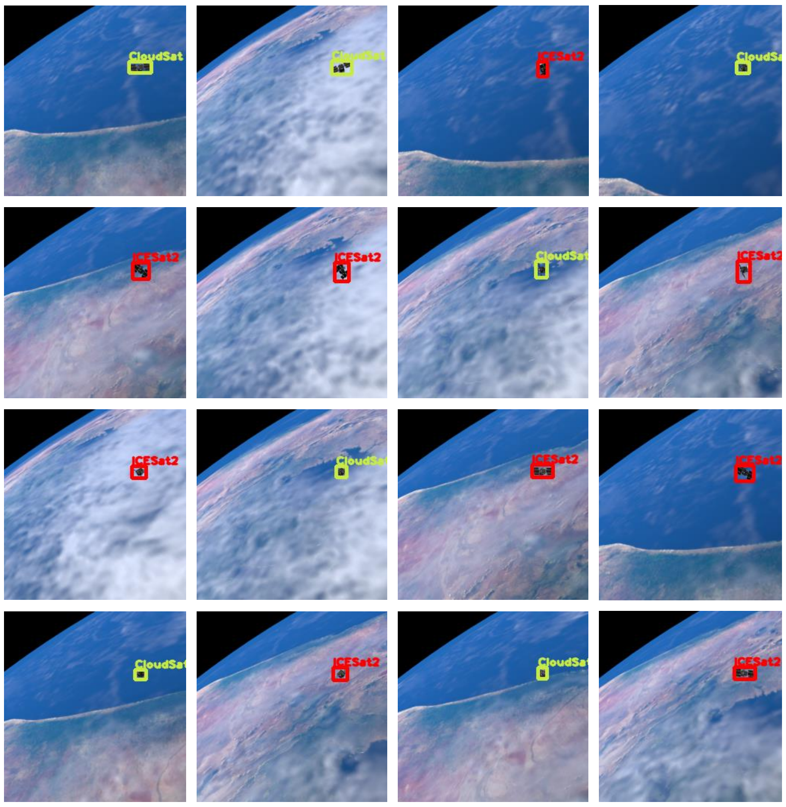


FIGURE 3.5: Difference in mean IoU values between targets which have been seen during training and those which have not. Targets with names in blue are in the full dataset only; those with names in orange are in both.

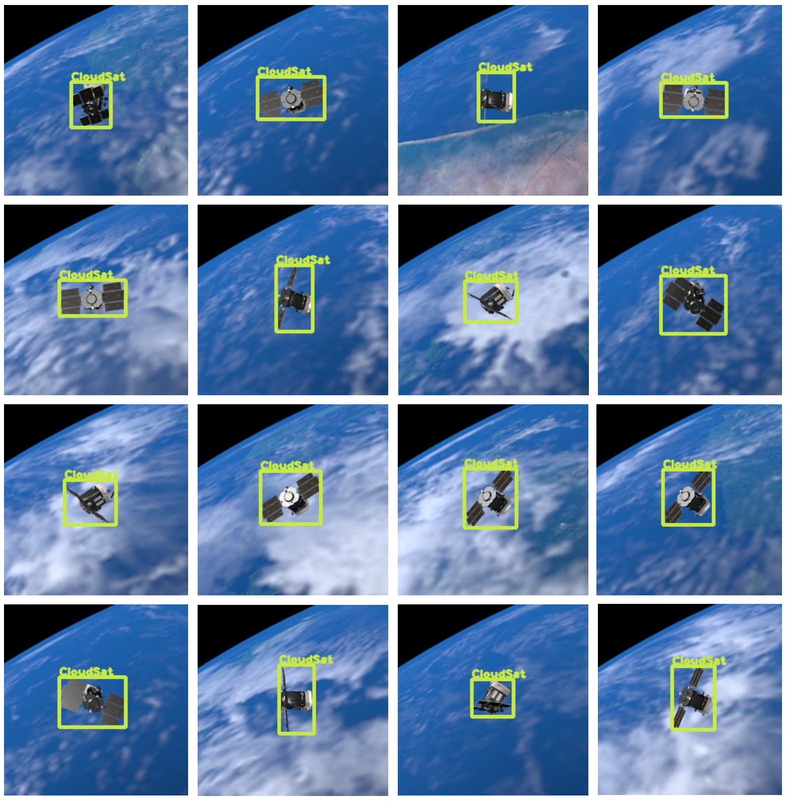
presence of other objects in the images besides the Earth, nor has it analysed the range capabilities of the approach, which are useful investigations to be made before adopting this approach.

Thus far, the analysis has concentrated on the accuracy of the predicted bounding box, which can be useful for relative navigation and tracking. However, the model also has the ability to predict a label alongside the object location. This may be useful to classify a debris target into different classes during the observation phase; these could be CubeSats, communications satellites or satellite fragments. This information may help the chaser spacecraft to autonomously decide on the best method of de-orbit on the fly, for example, depending on the characteristics of the target. For the analysis, we simply use the satellite name as the classifier.

However, due to the similarity between many of the satellite targets, the model finds it difficult to distinguish between them. For example, there are multiple targets in the dataset which consist of a cube-like body with one or two solar panels at the sides. In addition, a low resolution of image (256×256 pixels) is used; higher resolutions contain more information and thus would lead to an increase in accuracy, at the cost of slightly slower computation times. Also, clearly the accuracy depends on the distance to the target. These results are illustrated qualitatively in Figure 3.6, which shows the predicted classes for a batch of test images containing the CloudSat target, during the approach and inspection stages of a mission. A green box corresponds to the CloudSat label which is a correct prediction; red means that the satellite has been incorrectly labelled.



(A) 200m from the target



(B) 70m from the target

FIGURE 3.6: The predicted satellite labels over a batch of 16 images, when observing a CloudSat target at two different distances.

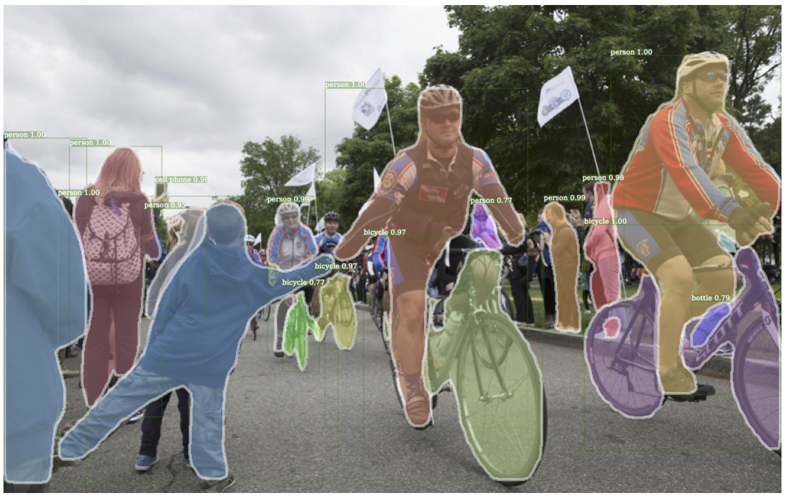


FIGURE 3.7: An example scene segmented using Mask R-CNN (He et al., 2016).

This method is accurate at close distances from the object, such as during the inspection phase, but the accuracy can be seen to fall off as this distance increases. However, there are a number of ways to improve this, as it is useful to be able to detect and classify a target during the approach phase. As mentioned, we could simply increase the resolution of the images, but this does have downsides and is somewhat limited. Another approach would be to classify types of satellites instead of labelling each satellite target individually. We observe that the Sentinel6 model, which has a more unique shape, is correctly labelled in 97% of cases at 200 metres. This suggests that collecting satellites into groups based on their overall structure will be more effective at correctly classifying the target earlier in the approach phase.

3.2 Surface segmentation

Another task with very interesting potential applications is surface segmentation. We have discussed previously the importance of identifying suitable surfaces on a target for contact-based removal. This is because it is crucial that the capture and removal process does not generate more debris, which is a risk when making contact with fragile surfaces such as solar panels. Again, we will show that machine learning technologies can provide a solution to this problem.

Similarly to object detection, image segmentation is a key research area in computer vision and machine learning. There are many similarities between the two tasks; however, instead of detecting discrete objects in a scene and returning a bounding box describing their positions, in image segmentation the scene is divided into zones of pixels, where each pixel is given a label corresponding to a class of objects (or no object). An example of a segmented scene is shown in Figure 3.7.

Clearly, a similar technique can be applied to segment different surfaces on a satellite target. This could be used to determine the makeup of surfaces, such as the type of material used in construction, to determine the most robust surface. As well as this, though, the same technique can detect and track different common elements of a spacecraft, such as solar panels, engines or antennas. This is useful not only in determining fragile objects to avoid, but additionally the Mission Extension Vehicle by Northrop Grumman has already demonstrated the possibility of docking with the engine of a satellite which has not been constructed with docking in mind.

3.2.1 Implementation

Image segmentation techniques are implemented and applied to the problem of detecting surfaces on a satellite target. Again, pre-trained models are used which are then finetuned and tested using our generated datasets. The goal of the implemented model is to determine surfaces on the target which would be the best candidates for making contact with, using a grapple or harpoon for example. Currently, the capabilities are demonstrated by using only a few different surface labels, but this could be easily extended in the future to include more labels and satellite models, simply by adding these to the dataset generation.

3.2.1.1 Pre-Trained Models

Like object detection, image segmentation is a domain of computer vision which has seen a large amount of research in recent years. There have been several different traditional methods for surface segmentation, for example edge segmentation based on edge detection or region-based segmentation which looks for similarities between adjacent pixels. However, with the advancement in neural network-based methods in recent years, these now significantly outperform the conventional approaches.

Image segmentation tasks usually fall into one of three categories. Semantic segmentation classifies all pixels of an image into a number of semantic classes. This provides information about the different objects in the scene, but if several objects of the same class are close to each other or overlapping, the segmentation will be less useful. On the other hand, instance segmentation provides a separate instance for each object, without predicting the class that the object falls into. The problem above will be solved in this case, but the output contains less information about the scene. Therefore, the final task, known as panoptic segmentation, combines the two methods, providing separate instances of objects as well as classifying each instance. For our use case, semantic segmentation is sufficient since we are most interested in classifying the types of each observable surface.

Unlike a simple classification task, segmentation models often consist of both an encoder and decoder. The encoder decomposes the input into a latent space representation of the scene, which is then used by the decoder to construct the output. The output of a semantic segmentation model is typically an n-channel binary image, where n is the total number of classes. Each channel therefore contains a binary map corresponding to whether that class exists at that pixel location.

Several deep learning segmentation models have been proposed, each of which differ in the structure of a convolutional encoder-decoder network. This network was first made popular in two different works at a similar time, SegNet (Badrinarayanan et al., 2017) and U-Net (Ronneberger et al., 2015). SegNet is a fully convolutional network which uses a combination of convolutional and pooling layers to generate a sparse feature map, followed by the same number of upsampling and convolutional layers to densify the feature map. Finally, a softmax activation at the end generates the pixel-wise classifications. U-Net uses a similar approach but includes skip connections between each level of the encoder and decoder, which helps to reduce the data loss caused by passing through the bottleneck of the sparse feature maps. These works paved the way for future research including DeepLab (Chen et al., 2018) and Mask R-CNN (He et al., 2016).

Mask R-CNN is selected as the best model to implement for the surface segmentation task; this model is provided freely with pre-trained weights from the MS-COCO dataset. Mask R-CNN is an extension of the Faster R-CNN model for object detection described in the previous section. First, a CNN is used to extract features from the image, using the ResNet 101 architecture. A region proposal network is applied to the output feature maps; all the regions are then converted to the same shape and passed through a fully-connected classification network. This is similar to the original object detection network, resulting in a class label and bounding box for each object. However, Mask R-CNN also generates a segmentation mask for each region that contains an object, by passing the regions through a fully-convolutional network similar to SegNet. This approach performs panoptic segmentation as each instance of a class is detected separately and assigned a bounding box and class label.

3.2.1.2 Training and Datasets

In order to finetune the Mask R-CNN model, labelled training data is required. The labels must provide ground truth segmentation maps of each different object class. As discussed earlier, our simulation software is capable of generating these labels automatically from the satellite models. These labels are provided alongside the images in the train and test datasets.

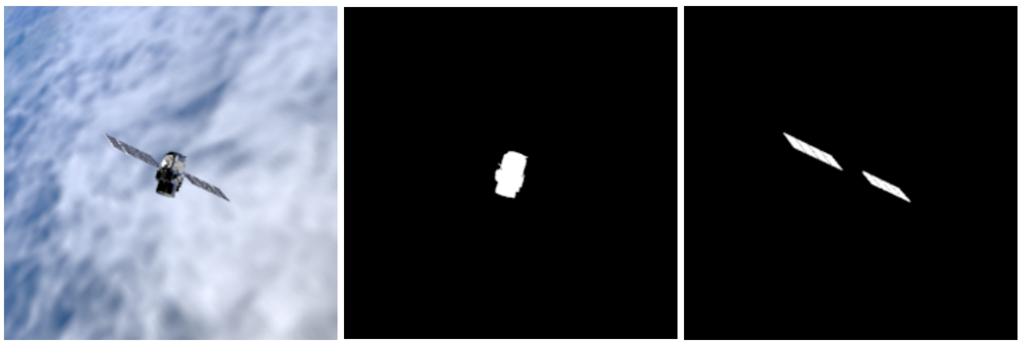


FIGURE 3.8: An example image in the training dataset along with the generated surface masks.

As a demonstration, only two different labels are provided for the data. These correspond to either fragile solar panels – which should be avoided during rendezvous and docking – or the solid satellite body. This alone provides useful information about the satellite target for the debris removal spacecraft, but could easily be extended in the future with more labels.

The process of generating the dataset labels involves the construction of n binary maps for each of the n surface labels. In this simple case, $n = 2$. These segmentation maps are created by performing a visualisation of the target disregarding any light sources, where the textures of the 3D model of the target are modified to be a black and white mask. This results in images such as those in Figure 3.8.

This data is sufficient for finetuning and testing the model. In order to investigate the generalisation performance, several training and test datasets are constructed with different satellite targets or backgrounds. The model is finetuned and then analysed using these datasets.

3.2.2 Results and Discussion

As before, the model's performance is analysed using a number of test datasets, under different situations and with different satellite targets. Ideally, the approach should be capable of accurately distinguishing between the two surface classes – namely, the satellite body and solar panels – regardless of background noise and generalising to unseen satellites.

First, we provide a qualitative representation of the prediction accuracy. In Figure 3.9, the predicted surface labels are plotted alongside the ground truth data, for three different satellite targets at different orientations. The model predicts which pixels correspond to one of the two labels, along with a confidence score for each segment. We keep only those with a confidence score greater than 0.5. The class segments are

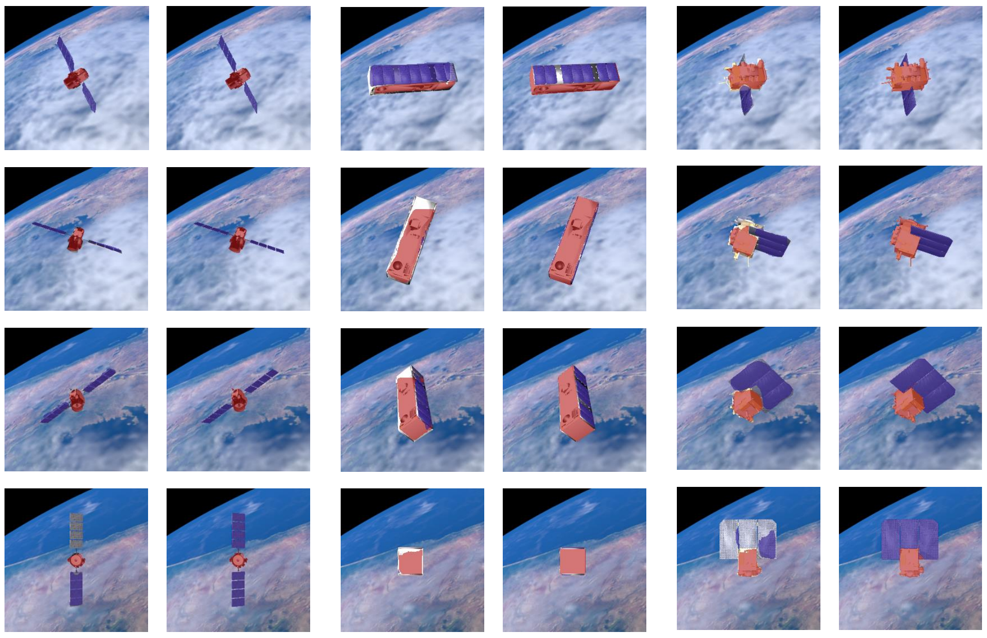

(A) Calipso (B) ICECube (C) LRO

FIGURE 3.9: The predicted surface segmentations (left) alongside the ground truth (right) at four points in orbit, for three different satellite targets: Calipso, ICECube and LRO.

overlaid on top of the original image, shown in red for the satellite body and purple for the solar panels.

This figure highlights the effectiveness of the approach alongside some of the issues. In most cases, the two surfaces are labelled accurately, even when observing three notably different satellite shapes. This is particularly true for the satellite body; however, the solar panels appear to cause problems in some cases. For example, the back side of the solar panels often have a significantly different texture to the front side. This can result in an incorrect label, as shown in the bottom-left images of the Calipso target. This could perhaps be solved by adding a separate class corresponding to the back of a solar panel. Furthermore, the bottom-right images illustrate show that significant reflections off the solar panels can also be difficult for the model.

While the initial analysis above is useful in validating the method, it is also important to be able to quantify the prediction accuracy, in order to compare results. Again, we can use the IoU metric for this case. However, as we are now dealing with pixel-wise labels, the computation is slightly different. The IoU value is calculated for each class as the number of overlapping pixels divided by the total number of pixels with that label in the prediction and ground truth. Simply, this is equivalent to

$$\mathrm{IoU} = \frac{\text{pred} \bigcap \text{truth}}{\text{pred} \bigcup \text{truth}}. \tag{3.1}$$

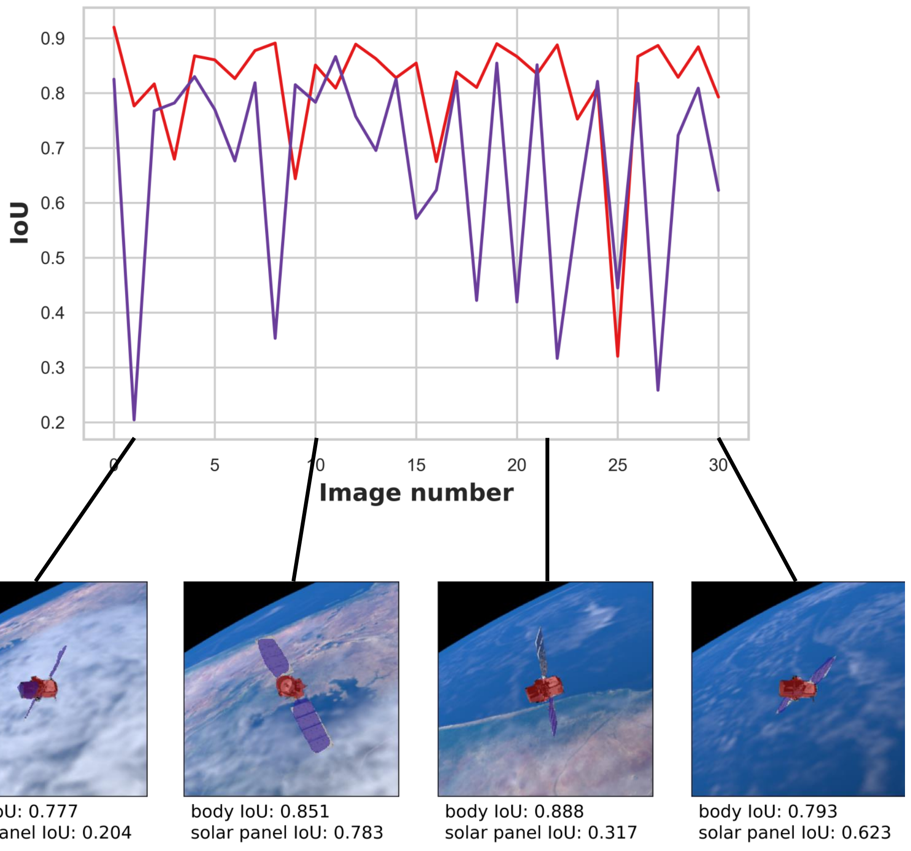


FIGURE 3.10: A plot of the IoU values for each surface label over 30 timesteps, observing the Calipso target. The red and purple plots correspond to the body and solar panel labels, respectively. The plot is labelled with the image and prediction at several steps.

Although the same metric is used for this as the previous task, the results cannot be directly compared, both because of the increased difficulty of this task and the fact that the outputs of the two models are not equivalent.

In the following investigations, the accuracy is measured as the mean IoU over the full test dataset. However, it should be noted that this may not always be the best metric. A reason for this can be seen in the bottom images of the middle column of Figure 3.9. In this image, the solar panels have almost disappeared, so they have not been labelled by the model, though the labels do exist in the ground truth as a few pixels of the surface are still visible. This means that the IoU of the solar panels in this case is 0, despite the error appearing to be very small; this will have a significant impact on the mean IoU over the dataset. Figure 3.10 illustrates this by plotting the IoU for both labels over the first 30 images from the Calipso test dataset. At several points, we label the plot with the image and prediction at that point, to give a better indication of the reasons for a drop in accuracy.

Next, the generalisation characteristics of the model are investigated. For this, the model is tested on several 200-image datasets under different conditions, taking the

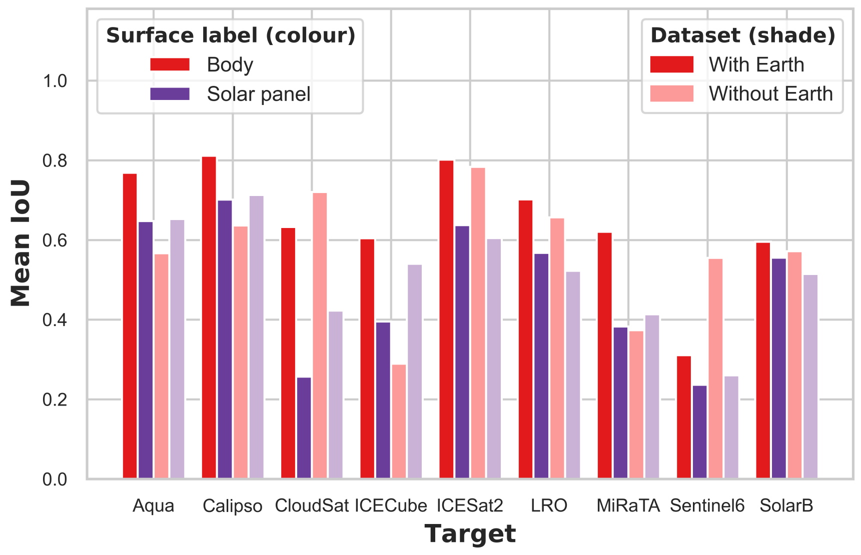


FIGURE 3.11: The mean IoU over each test dataset, when the Earth is present in the images or not.

mean IoU for each as described previously. In Figure 3.11, the impact of background noise is investigated by testing the performance on two different datasets, with or without the Earth visible. The graph illustrates the IoU of each of the two surface labels, under the two conditions.

In several cases a small drop in accuracy can be observed with the more complex images. However, there are also a number of results which show the opposite response. This suggests that the variation is most likely caused by the randomness of the neural network approach, as well as the discussed issues with computing the mean IoU, rather than the difficult image data. For some targets, especially Sentinel6 and CloudSat, the variations are particularly pronounced. This can be explained by the fact that Sentinel6 has the most unique body shape, while the CubeSat models are more likely to have disappearing solar panels than those with the panels separate from the satellite body. It is possible that increasing the variation of the training data or using a different evaluation metric will provide better results here.

It is again useful to investigate the generalisability to unseen satellite targets. This is performed by finetuning the model two separate times, using a full or reduced training dataset. The analysis is shown in Figure 3.12. The full training data contains all the satellite targets plotted, whereas the reduced data contains only those labelled in the lighter colour.

The model appears to be capable of accurately labelling the satellite body even for those targets which were not seen during the training. However, the accuracy of the solar panel labels is noticeably worse for both the LRO and SolarB models. This is most likely caused by the different textures and design of the solar panels; again,

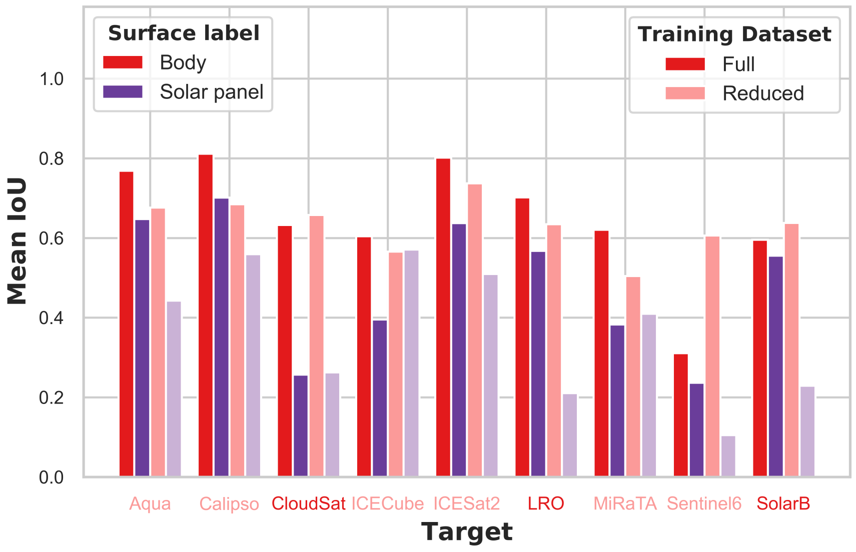


FIGURE 3.12: The mean IoU over each test dataset, using a model which was trained on the full set of targets or a smaller reduced set. The targets with names in the darker colour are only present in the full dataset, while the others are in both.

increasing the variation of satellite models in the dataset is likely to improve this performance. The fact that the ICECube model is not affected backs up this assumption, since the solar panel textures are similar to the other CubeSat model in the dataset, MiRaTa. Nonetheless, these results show promise for applying the techniques to real-world situations.

3.3 Machine Learning Applications in GNC for Space Missions

So far, we have demonstrated how machine learning technologies can be used to help solve three of the key challenges of visual navigation for active debris removal. In order to facilitate this research, we have developed a simulation framework for generating training datasets. However, there remain a number of other problems to be faced by the GNC system. We discuss these problems here along with a summary of the machine learning which can be applied to the situations, and detail some advancements made in these areas to date.

The most popular topic of research in this area involves the application of machine learning techniques for spacecraft control. Adaptive and autonomous controllers for spacecraft motion control are of significant interest, so many researchers have turned to machine learning techniques to achieve this. Izzo et al. (2019) and Shirobokov et al. (2021) each perform valuable in-depth reviews of the recent innovations in this area. The latter takes a more tentative viewpoint on the necessity and effectiveness of the approaches, but both conclude that machine learning methods are a useful and promising topic of research for spacecraft control. However, there remains further work to be done, particularly into the validity, stability and reliability of machine learning approaches and their application to more realistic problems, for example considering the hardware and software available on-board.

There are a number of common spacecraft control problems to which machine learning methods have been applied. For example, one such problem is trajectory design. This problem consists of planning a trajectory with minimal cost, which is usually the fuel cost. This can be solved using either supervised learning (Cheng et al., 2019; Izzo and Öztürk, 2021) or reinforcement learning (Carnelli et al., 2009; Federici et al., 2021) techniques. Indeed, these are the two bases from which most research in this domain is developed. These methods can be applied to other control problems including angular motion stabilisation (MacKunis et al., 2016; Biggs and Fournier, 2020), spacecraft docking (Wei et al., 2018) and control for landing on the surface of celestial bodies (Gaudet et al., 2020; Jiang et al., 2019). In addition, although the applications to vision-based navigation are limited, these approaches have been applied to the collection of visual data, for optimal shape reconstruction (Brandonisio et al., 2021) or maximising scientific image acquisition (Piccinin et al., 2020).

Much of the research into machine learning applications to different problems are incited by the construction of challenges or competitions, which allow researchers to compete against each other to solve a specific problem. Some examples include the satellite pose estimation challenge (Kisantal et al., 2020; Park et al., 2021), the collision avoidance challenge (Uriot et al., 2021) and the Global Trajectory Design Competition (GTOC) (Izzo, 2007), all of which are run by ESA. These challenges lead to interesting developments in their respective fields and are particularly useful for developing machine learning models, as large datasets tend to be provided to all contributors. This is often the limiting factor in machine learning research, and is the reason why we have released our datasets alongside this work.

The uses of AI in spacecraft GNC become particularly interesting when you consider the possibility of combining different machine learning approaches into a fully autonomous AI-based GNC system. In their survey, Izzo et al. (2019) discuss several potential synergies which can be obtained by combining different machine learning techniques; these are often more effective approaches than the individual methods for solving the complex challenges in space. Indeed, it can be seen that the methods we investigate can also benefit from synergies. For example, the surface segmentation combined with relative motion estimation can be used to determine and track an optimal contact point to enable autonomous rendezvous and docking. This idea of a combined modular GNC system is one of the most exciting areas in which we expect to see more research in the near future.

3.3.1 Disadvantages of Machine Learning Applications

Despite the exciting prospects which we foresee in this domain, it is import to understand the reality of the current state of the research. Machine learning approaches have received some criticism due to a number of issues associated with the technologies. These must be solved before the technologies can be adopted in real missions.

Arguably, the most important difficulty associated with the use of AI for guidance and control is the lack of explainability. When developing a guidance system, it is useful to know the reason behind the choices made by the system. This allows us to advance our understanding of the problem at the same time as improving the performance of the system. Instead, DL models are generally a black box, which means that there is no explainability behind any of the decisions made. In a similar vein, it is difficult to quantify the errors in a DL approach, which can cause problems when integrating the model within a pipeline. As an example, it is difficult to implement a Kalman filter in our rotation estimation approach, as the filter must be initialised with an estimate of the process noise.

Another key difficulty with these applications is one which we have covered already. This is the requirement for large, labelled datasets in order to train the models. One approach to reduce this problem is to use pre-trained networks, although as we have already shown, this is not always applicable to all problems. We have attempted to aid in this area by publicly releasing synthetic datasets. However, if only synthetic data is used, there still remains the challenge of adapting a method which has been trained using synthetic data, to a real-world environment. Real-world data is much more challenging to acquire, although it is possible to increase the size of such datasets by applying random transformations to the data or using Generative Adverserial Networks (GANs) (Park and D'Amico, 2020).

Finally, the validation of DL approaches is another challenge which requires further work. Often, such methods obtain very impressive results on a specific task, but do not perform as well on different tasks. This is due to a lack of understanding of the strengths and weaknesses of the technologies. It is important to demonstrate that the system has good generalisability characteristics in order to adapt to a wide range of situations. In addition, it is necessary to thoroughly validate the performance of an AI based GNC system in order to increase the trust in the approach, due to the fact that they are seen as a black box. One way in which this is beginning to be achieved is through the challenges detailed earlier, which provide benchmarks with which to compare different approaches against one another and against the conventional algorithms.

Chapter 4

Neural Networks as Feature Extractors

The previous chapter made use of the generated datasets to apply deep learning approaches to visual navigation in space. The technologies were shown to be able to aid in solving two of the three key remaining challenges faced by the visual navigation system in ADR missions. However, there remains the challenge of capturing a tumbling satellite, which requires the accurate estimation of the rotational state of the target. This is not a task to which previous technologies can be applied directly; instead, in the upcoming two chapters, we demonstrate a novel approach by which deep learning techniques can accurately estimate the rotation of an uncooperative, unknown and tumbling satellite.

Since the breakthroughs led by Krizhevsky et al. (2012), convolutional neural networks (CNNs) have been widely applied in various image processing problems. Examples of applications include image classification (Krizhevsky et al., 2012), facial recognition (Farfade et al., 2015) and object detection (Dhillon and Verma, 2020). In most cases, the CNN approach significantly outperforms the conventional methods, leading the way for new advances in many research areas including computer vision, autonomous vehicles and healthcare.

In all these examples, the CNN is in effect working as a feature extractor. The convolutional layers extract complex features from the image data; these features are then passed through classification layers to provide the result. For example, in an object detection model, the convolutional layers may look for features such as a tail, four wheels or two eyes. These are high-level features which do not have an obvious relationship with the inputs.

However, for the task of predicting the rotation of an object, we are more interested in the change in location of low-level features such as corners and edges. In Section 4.2, we show that CNNs can also be used to extract these low-level features which are able to capture contrast and spatial information. We detail the design of a neural network architecture which aims to extract and match low-level image features between successive image frames. A number of loss terms are incorporated to encourage the network to learn more useful image features for the specific problem. In Section 4.3, the performance of our neural network-based feature extractor is analysed and compared with conventional computer vision algorithms. However, we must first cover the state of the art in deep learning and neural networks, in order to place our work within it.

4.1 Background and Previous Work

The field of computer vision has seen significant advancements in recent years contributed by machine learning technologies, many of which have revolutionised the field. For example, image classification is perhaps the most well-known problem in computer vision, with the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) being a popular system for showcasing the best-performing algorithms (Russakovsky et al., 2015). Following the recent advancements achieved by applying machine learning to the challenge, image classification is now largely considered a solved problem, with recent results showing that most competing teams achieved classification accuracies above 95%. Indeed, 2017 marked the final year of the ILSVRC for this reason.

The central concept of machine learning is that of using a large set of training data to determine patterns, which can then be used to categorise unknown data in a test dataset (Bishop, 2006). The ability to correctly categorise new data which differs from that used in the training set is known as generalisation. Machine learning has been used, for example, in speech recognition (Deng and Li, 2013), image classification (Deng et al., 2009) or medical diagnosis (Kononenko, 2001).

Machine learning algorithms often make use of artificial neural networks (ANNs), which attempt to mimic the biological network of neurons in the human brain. ANNs consist of a number of layers of processing units which simulate neurons. These are interconnected in such a way that signals can travel through the network in both serial and parallel (Anderson, 1995). Each path contains an adaptive weight that can be tuned by the learning algorithm in order to improve the model.

Deep Learning is a key subset of machine learning, which has seen a significant amount of attention in recent years. There are several contrasting opinions on the definition of deep learning, though a general view is that it describes learning with "deep", i.e. many-layered, ANNs (Goodfellow et al., 2016). Deep networks are able to capture more complex relations between inputs and outputs, though have significantly more parameters to learn.

Convolutional neural networks (CNNs) are a further subset of deep neural networks, which are particularly suited to computer vision applications. These networks accept an image as input and pass it through a series of convolutional and pooling layers to learn image features. In contrast to the traditional approaches to feature extraction discussed previously, CNNs learn through training the best features to solve the specific problem.

CNNs are not a recent discovery; research was kicked off by LeCun et al. (1998) with LeNet5, which was the inspiration for many of the future architectures. However, due to the low computing power available, CNNs saw little use for the following years. Then, in 2012, Krizhevsky et al. (2012) demonstrated the power of CNNs for image classification tasks, winning the ILSVRC by a significant margin. This represented a breakthrough for deep learning and led to a revolution in the field, and thereby also in the larger field of machine learning.

Shortly after this, CNNs were found to lead to a similar increase in performance in object detection problems (Girshick et al., 2014). Since then, they have been used in a wide variety of applications, including speech recognition (Sainath et al., 2013), pose estimation (Tompson et al., 2014), and text classification (Kim, 2014). These methods are capable of producing better results than could be achieved with traditional techniques (Islam et al., 2016). However, image classification and object detection remain two of the most popular applications of CNNs, with much of the innovation coming through these channels.

A CNN must be trained on a large set of training data before it can be used to generate predictions. The most common method of training is supervised learning, whereby the network is provided with a dataset of training images, each with a labelled output. The training data is used by the model to learn the filter parameters such that the distance between the expected and labelled output is minimised, before attempting to classify some test data. A very large amount of training data is required in order to build an accurate model. This requirement prompted the construction of the ImageNet image database (Deng et al., 2009), a freely available collection of millions of labelled images. As an alternative to supervised methods, unsupervised learning algorithms are given an unlabelled dataset containing many features, and they learn the useful properties of this dataset. Semi-supervised learning is a combination of both; they typically use a small amount of labelled data alongside a large amount of unlabelled data (Goodfellow et al., 2016).

However, in many applications, training a neural network model from scratch is not desired or, indeed, necessary. Instead, a model that has been pre-trained on a different dataset, such as ImageNet, can be finetuned on a separate problem. This is significantly less time-consuming and resource-intensive, allowing for deep learning models to be quickly and easily adapted to different tasks. In addition, the required size of the training datasets is greatly reduced in this case. A few of these commonly used network architectures are detailed in the following section.

4.1.1 Structure of Neural Networks

The structure of a neural network contains an input and an output layer, with several “hidden” layers in between. The layers each consist of a number of nodes, each of which is connected to every node in the next layer. A small example of such a network is illustrated in Figure 4.1. The output is constructed by passing the inputs through all hidden layers as shown.

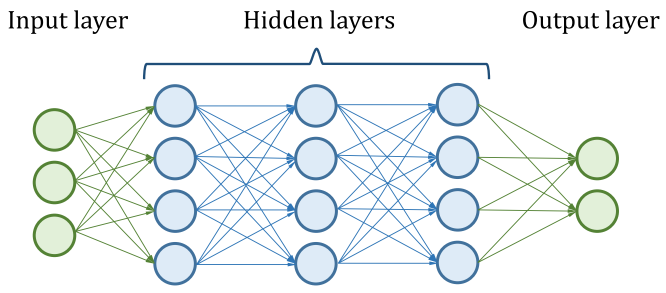


FIGURE 4.1: The structure of a small neural network.

Each connection between nodes has an associated weight w_i which is multiplied by the value of the previous node, x_i, and each node has an additional bias term. The sum of weights and biases is passed through an activation function, which activates or deactivates the node based on the value of $(\sum_i^m x_i w_i + \text{bias})$. The values of the weights are learned during the training loop, by attempting to minimise a loss function applied to the output.

A convolutional neural network is built upon similar principles, though there are a few significant differences. CNNs take in an image as input instead of a vector, and the node connections are replaced with convolutional filters over the input. Again, the layers are all interconnected within the network. As shown in Figure 4.2, these networks often consist of several convolution and pooling layers to extract features from the image, followed by a classification stage of fully-connected neural layers as in Figure 4.1.

The role of a convolutional layer is to perform a convolution operation over the input, resulting in a 2D feature map. This operation is equivalent to a multiplication of a set of weights with the input, similarly to a traditional neural network. A single convolutional filter will extract low-level features such as edges and blobs from an

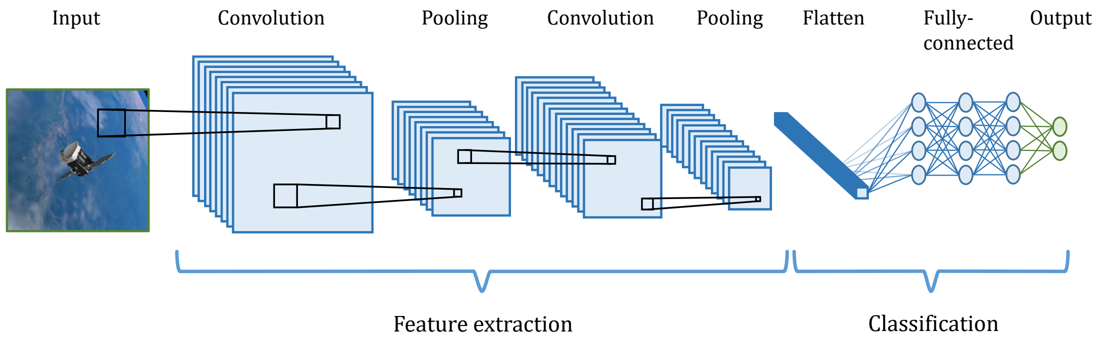


FIGURE 4.2: An example of a convolutional neural network, combining feature extraction and classification layers.

image, while a deeper network with many layers will result in more complex, high-level features. The convolutional layer can be adapted by changing the filter size, number of filters or stride length, all of which have an impact on the output of the layer. As before, an activation function is applied to the output of the convolutional layers. The most common choice is the rectified linear unit (ReLU), which simply returns the input value if it is positive, or 0 otherwise. Finally, pooling layers are often used to reduce the spatial size of the feature maps, reducing the number of parameters and therefore the computational power required. These layers are also useful for extracting features which are rotational and positional invariant.

There are several choices to be made when constructing a CNN model. The network depth, number of filters per layer, and the use of pooling layers are some parameters which will all have an impact on the performance of the model, but this performance is difficult to predict beforehand. This fact, combined with the difficulty of training a network from scratch, mean that pre-constructed and pre-trained models are often used as a starting block in many applications. VGG16 (Simonyan and Zisserman, 2014) and ResNet (He et al., 2015) are two such models which have made significant progress in the field in recent years.

The VGG network architecture consists of a 16-layer and a 19-layer network (referred to as VGG16 and VGG19). One key characteristic of this architecture is its simplicity; they were the first to use 3×3 convolutions, finding that stacking 3×3 layers can emulate the effect of larger filters. This simplicity allowed for a greater network depth; at the time, 16 and 19 layer networks were considered very deep. This enabled the extraction of more complex features from images. However, Simonyan and Zisserman found the VGG networks to be very difficult to train. Their solution was to train smaller networks first, and use these as initialisations for the deeper networks - called pre-training. This is a very time consuming process, significantly increasing the training time of the model. Due to the large feature size and depth, once trained VGG is also slow to run (Canziani et al., 2016).

The ResNet model, proposed by He et al. (2015), introduced an innovative new form of network architecture, consisting of a sequence of residual blocks. Previously to this, it had been found that, as a convolutional network gets deeper, it becomes significantly harder to train, and its performance begins to saturate or even degrade; this is known as the *vanishing gradient* problem. The residual block, shown in Figure 4.3, attempts to solve this problem by using a shortcut connection. Put simply, this allows the network to bypass weight layers if they do not improve the model. This idea revolutionised the research field, enabling extremely deep networks to be trained – up to 200 layers for ImageNet and 1000 layers on CIFAR-100 (He et al., 2016).

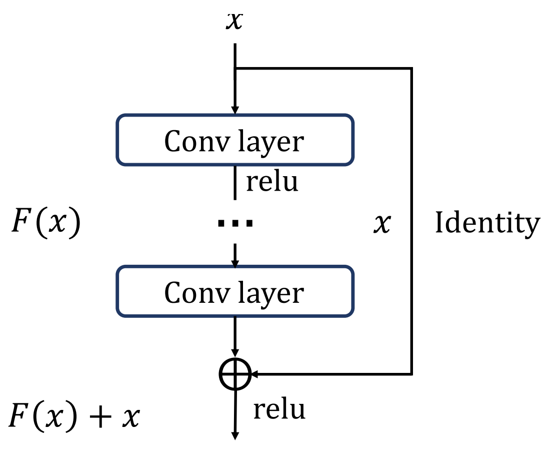


FIGURE 4.3: A residual block.

The deep ResNet implementations, such as ResNet50, were found to outperform the previous models, including VGG19. In addition, due to the use of global average pooling rather than fully connected layers, the model size is in fact substantially smaller than VGG19, despite being much deeper. Besides this, the 3×3 convolutions actually consist of a 1×1, 3×3 and 1×1 convolution. The 1×1 convolutions reduce the dimensionality of the feature map, thus speeding up the convolution operation. As illustrated in Figure 4.4, this leads to higher accuracy and faster run times than the VGG architectures.

4.1.1.1 Siamese Neural Networks

A siamese neural network (SNN) is a subcategory of neural networks which contains two or more branches, each applied to different inputs. The branches each contain identical layers, with weights shared between the branches. Thus, each branch attempts to extract the same features from each of the input images. The outputs of the branches are then often combined before being passed to a further classification or comparison stage. This enables the different inputs to be compared against each other rather than using the network's learned knowledge of the training data.

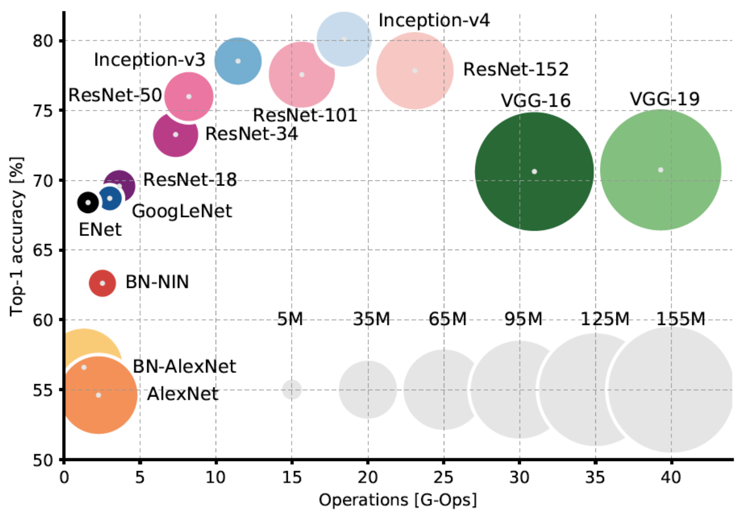


FIGURE 4.4: Analysis of the accuracy and required number of operations of different CNN architectures (Canziani et al., 2016).

An example use case of SNNs is in visual tracking (Zhang and Peng, 2019). In this case, an object from the first frame of a video is selected to be tracked. For each successive video frame, the current frame is processed alongside the view of the object from the first frame. By comparing the feature map of the video frame with that of the object of interest, it is possible to place the object within the scene, even when it has not been seen by the network during training.

4.1.2 Applications of Deep Learning in Space

Deep learning techniques have already been proposed for a small number of applications in space, for example crater detection (Silburt et al., 2019), asteroid classification (Carruba et al., 2020) and satellite state determination and tracking from noisy on-board measurements (Mortlock and Kassas, 2021). To date, though, the real-world applications have been very limited, mainly consisting of the analysis of mission data (Vida and Roettenbacher, 2018; Shallue and Vanderburg, 2018).

However, it could be expected that the inclusion of deep learning technologies can solve many of the remaining challenges of autonomous VBN of spacecraft, particularly in the context of proximity operations about an uncooperative target. These cases have seen little attention to date, with much of the research concentrating on the guidance and control system (Izzo et al., 2019; Uriot et al., 2021), rather than the visual navigation problem. Only a small amount of recent research has focused on a similar problem to our own: that of estimating the state of an uncooperative satellite from visual data. This research is addressed at the end of the following section.

There are two key difficulties with employing deep learning technologies in space missions: specifically, the availability of training sets and the high computational requirements when using deep networks. However, a key advantage of deep learning algorithms is their capability for computation on a GPU. There have been many advancements in GPU technology in recent years, making them an attractive candidate for on-board computation in space (Kosmidis et al., 2019). GPUs offer improved performance and energy efficiency for a lower cost when compared with CPUs.

4.1.3 Deep Learning for Pose Estimation

Deep learning algorithms have been previously investigated with respect to learning the orientation of objects for 6D object detection applications, which could be applied to the attitude determination problem. In general, this is a well-studied problem, with several potential solutions. For example, one of the common solutions is to use pose regression, where an observed point cloud is compared with the known point cloud of the object (Zeng et al., 2017), although this requires depth information and a-priori knowledge of the target object. Alternatively, monocular pose estimation has been demonstrated using CNNs (Wang et al., 2020), though again the training data contains the target object and therefore it must be known beforehand to generate the training data. Despite the recent research, the task of pose estimation is not an easy one; there are a number of challenges to be faced by the algorithms.

Generating training data of images labelled with full 6D object poses is considerably more effort than for 2D object detection. Therefore, many learning-based estimation methods are restricted to the few pose-annotated datasets which currently exist (Saxena et al., 2009; Wohlhart and Lepetit, 2015). Other approaches train on synthetic images from a rendered 3D model, though this introduces challenges in generalising to real test images (Sundermeyer et al., 2018).

These algorithms face further difficulties during training. The naive approach of simply regressing a fixed parameter describing the object pose, such as the quaternion, is generally unsuccessful due to representational constraints and pose ambiguities (Mahendran et al., 2017). However, the alternative approach of classification also faces challenges, since even coarse intervals of 5° lead to over 50,000 possible classes, where each class will appear only very sparsely in the training data. The complexity can be reduced somewhat by separately classifying a change in viewpoint and an in-plane rotation (Kehl et al., 2017).

Perhaps the most critical issue to tackle for object pose estimation is the existence of symmetries, which can cause pose ambiguities. In these cases, where an object is semi-symmetrical, a 2D image of the object could be correctly described by several different potential 3D orientations, particularly if parts of the object are obstructed. This is often the case in space images where the targets are often of simple geometries – cylindrical or cuboid for example – with few distinctive features. There are several strategies to overcome this problem, such as ignoring one axis of rotation (Wohlhart and Lepetit, 2015), though most are manual and tedious. One promising solution is to learn descriptors rather than solving for the pose directly. For example, Sundermeyer et al. (2018) use autoencoders to learn descriptors solely based on the object views.

Autoencoders were introduced by Rumelhart et al. (1986) as a means of reducing the dimensionality of data such as images or audio. These are trained to reconstruct the input after passing through a learnable encoder and decoder. The resulting latent space provides a descriptor of the input. Sundermeyer et al. (2018) found that by employing autoencoders, symmetrical ambiguities may be disregarded. In this approach, assigning 3D orientations to the descriptors only occurs after the training. An estimation of the pose of a real test object can then be found from a nearest-neighbour search of the object descriptor against a set of representations at several orientations. However, in this approach a 3D model of the object is required a priori.

A similar approach to the above can be used to learn image features. CNNs inherently detect features in the input images, though this information is hidden in the internal layers of the network. However, approaches such as that of Jakab et al. (2018) can determine features directly from unlabelled image data. Two images are used as input – a source and a target – where the object of interest is distorted in some way between the images. These could be two frames from a video or a single image which has undergone a transformation. The target image is encoded with the coordinates of keypoints, and the source image and encoded target are passed to a generator network which aims to reconstruct the target image from this information. The model thereby learns semantically meaningful keypoints from unsupervised training. These features are likely to be more useful than those generated from traditional feature extractor algorithms, which are interested only in local pixel intensities.

Recent research has investigated the applications of deep learning to attitude determination of satellites. In general, this is a more difficult problem than the general case, due to the featureless targets, lighting conditions, background and limited power available for on-board computation. Sharma and D'Amico (2019) present a method for pose estimation with a monocular camera using a convolutional neural network. This work also provides the Spacecraft Pose Estimation Network (SPEED), which has since been made publicly available through a competition on pose estimation run by the European Space Agency. Similar research has also looked at pose estimation from monocular images (Alimo et al., 2020; Oestreich et al., 2020), often using finetuned neural network architectures such as ResNet and VGG19.

However, in each of these cases, the target spacecraft is known a-priori; either the 3D model is provided, or the network is trained on a dataset containing a single satellite model. This work presented here aims to instead solve the more general case, in which the structure of the debris target is entirely unknown. Due to the significantly increased complexity of this task, the problem must be decomposed from one of pose estimation to an estimation of the rotational state of the target. The same approaches therefore cannot be applied. Furthermore, it is proposed that finetuning a network which has been pre-trained on an image processing task such as ImageNet will result in a network that looks for abstract features, which are not useful in determining the pose of an object.

4.1.4 Conventional Feature Extractors

A conventional computer vision-based approach to feature extraction will generally consist of a three-part process. First, the locations of feature points are detected in the image. These will be simple features such as edges, corners or blobs (regions of interest). Then, each feature is assigned a descriptor which describes the appearance around the feature point. Finally, these descriptors are compared across images to match similar features.

A simple example of a feature detection algorithm is Harris and Stephens (1988)'s corner detector, which simply finds points in an image which appear to be a corner of an object. Feature descriptors can then be extracted from the surrounding pixel intensities. However, there have been many improvements upon this simplistic approach in the past decades, to both the feature detection and description stages.

For example, the scale-invariant feature transform (SIFT) (Lowe, 2004) detects a large number of feature vectors by applying a difference of Gaussians to a series of smoothed and resampled images. The features are assigned descriptors which are invariant to translation, scaling and rotation, making them well suited to matching similar objects between images.

Speeded-up robust features (SURF) Bay et al. (2006) was a further innovation which presented a fast algorithm for feature extraction and description. This enabled its use in real-time applications such as object tracking. Finally, ORB was developed as an alternative to SIFT and SURF (Rublee et al., 2011). This algorithm is able to achieve similar performance to SIFT and better than SURF, while retaining real-time capability.

The above algorithms provide feature points with associated descriptors. The final task is to match the features between images based on their descriptors. One option is to use a brute-force matching approach. For each feature in the first set, the descriptor is compared with all features in the second set, with the closest one being returned as a match. Unsurprisingly, this algorithm is not efficient. A faster alternative is to use the fast library of approximate nearest neighbours (FLANN) approach (Muja and Lowe, 2009).

4.2 Extracting and Matching Features from Space Imagery

In this section, we detail a novel approach for estimating the change in pose of a spacecraft over time. Unlike many of the examples of pose estimation in Section 4.1.3, we do not use a typical CNN approach. Instead, we take inspiration from the simple concept of feature extraction and matching and combine it with deep learning technologies. This merged approach is inherently more generalisable than a conventional deep learning approach, since the intermediate stage of the network consists solely of the locations of low-level image features. These should not have a significant dependence on the geometry of the object and thus the approach works whether the network has seen the object during training or not. On the other hand, by incorporating neural network approaches, the approach is able to learn inherently more useful image features for the specific problem at hand, compared with conventional feature extraction methods based solely on computer vision principles. In addition, while conventional algorithms require a greyscale image as input, our approach is applicable to various different types of visual sensor, including colour and depth (RGB-D) sensors, monocular cameras and LiDAR sensors, provided that the model has been trained using a similar type of data. This enables a larger number of use cases, while also making use of more of the available information in the data.

The key concept of our approach consists of using a siamese convolutional neural network, where the outputs of the visual sensor on-board the chaser satellite at two different instances in time are passed to each of the two network branches. A similar technique is employed to the example in Section 4.1.1.1; by passing in two images of the scene separated by a time step δt, the state of the object in the scene can be compared between the images. From this, the rotation of the target over the timestep is estimated.

In this chapter, we are interested in only the feature extraction stage of the network. As described in Section 4.1.1, this often consists of a number of convolutional and pooling layers. In most examples in literature, the full feature maps output from this stage would be passed on for further processing, such as in a classification stage made up of fully-connected neural layers. Instead, we decompose the feature maps into an (x, y) position of each image feature, before and after the rotation; only these feature locations are then passed to a rotation estimation stage. The details of the rotation estimation approach are provided in Chapter 5; for now, it is sufficient to state that the motion of the features is used to estimate the rotation of the target over the timestep.

In Section 4.1, several applications of CNNs have been discussed. In these applications, the convolutional layers are in effect working as feature extractors; however, the extracted features are high-level and complex. For the task of predicting the rotation of an object, we are more interested in the change in location of low-level features, such as corners and edges. In this section, we will address the development of a CNN architecture which aims to extract and match low-level image features between successive image frames.

We find that the model struggles to begin learning a useful response when trained from scratch against the loss in the final estimate of the rotation. We overcome this difficulty through the use of several loss terms which aim to aid the model in finding optimal image features; these loss terms are discussed in the Section 4.2.2. We also first pre-train the feature extraction stage of the network separately on a simpler task, in which it attempts to extract features which are correctly matched and which follow the motion of the object, before finetuning the network by applying it to the rotation estimation task.

4.2.1 The Feature Matching Network

The aim of the feature extraction network is to capture low-level image features and match them between image frames. In other words, it should be able to extract a feature corresponding to the same landmark in two different images. This problem is best solved through the use of a siamese network. As both branches of a siamese network have the same weights, we would expect the same image features to be extracted from each of the two inputs; these features will simply have moved in space due to the rotation of the object over the time step.

In a conventional feature extraction algorithm, the two sets of features would then need to be matched by using a feature descriptor approach such as SIFT (Lowe, 2004) or ORB (Rublee et al., 2011). However, we can avoid this by making use of a key property of neural networks: the ordering inside the network is important. This means that the first result in the output vector should always correspond to the same image feature, regardless of the input, and so on. Since both inputs are passed through identical networks, we therefore propose that it is possible to avoid costly feature descriptor and matching algorithms completely, instead matching the features simply by their index in the output vector. This results in two sets of implicitly matched feature locations at times t and $t + \delta t$. These sets of feature locations are referred to as $\{\mathbf{p}_k\}$ and $\{\mathbf{p}'_k\}$, respectively.

We wish for the network to capture low-level image features which best describe the object's rotation. This can be achieved through the structure of the network.

4.2.1.1 Network Architecture

As addressed in Section 4.1.1, constructing a network from the ground up can be a difficult process, both due to the number of parameters and the unintuitive nature of the networks. In addition, training a full network requires larger datasets and more computational resources than finetuning an existing network. Thus, in many cases, the best approach is to use a pre-constructed network, replacing the final classification layers and finetuning on the new dataset; several of the works detailed in Section 4.1.3 use this approach.

Therefore, the first network architecture investigated consisted of using the pre-trained VGG16 network as each of the two branches of the model. However, this layout was found to be incapable of learning useful feature locations. This is likely due to the fact that networks such as this, which are built for image classification tasks, will extract very complex, high-level features. In addition, the resultant feature maps tend to lose positional information; this is caused by the use of many pooling layers.

Clearly, for a unique approach such as this, it is necessary to construct a new network architecture. We investigate several different architectures in order to determine the optimal structure for this approach. There are a few ... from the theory which can aid with this task, although much of the process consists of trial and error. Very deep networks have been used in classification tasks due to their ability to obtain abstract, high-level features to describe a scene; thus, it can be expected that the model could be encouraged to extract low-level features by instead using a shallow CNN with few convolutional layers. This has a further advantage in that shallower networks tend to suffer less from generalisation problems. In addition, pooling layers are known to provide position and rotation-invariant features; however, this positional and rotational information is critical for our approach, and so these layers should be avoided.

Therefore, a convolutional neural network is proposed to detect and match landmarks in the two input images, with the layout shown in Figure 4.5. This is a siamese network, where each input image passes through the same network to extract image landmarks. Following the conclusions made above, the network contains a small number of convolutional layers and no pooling layers. The size and structure of the network is varied, with Figure 4.5 illustrating the shallowest network investigated. The results of this investigation are given in Section 4.3.2.

It should be noted that, while the approach is based on the concept of deep learning, the network itself is actually very shallow. Instead, the effective size of the network is increased by the width of the convolutional filters and the number of filters in each layer. For this reason we refer to the task still as one of deep learning.

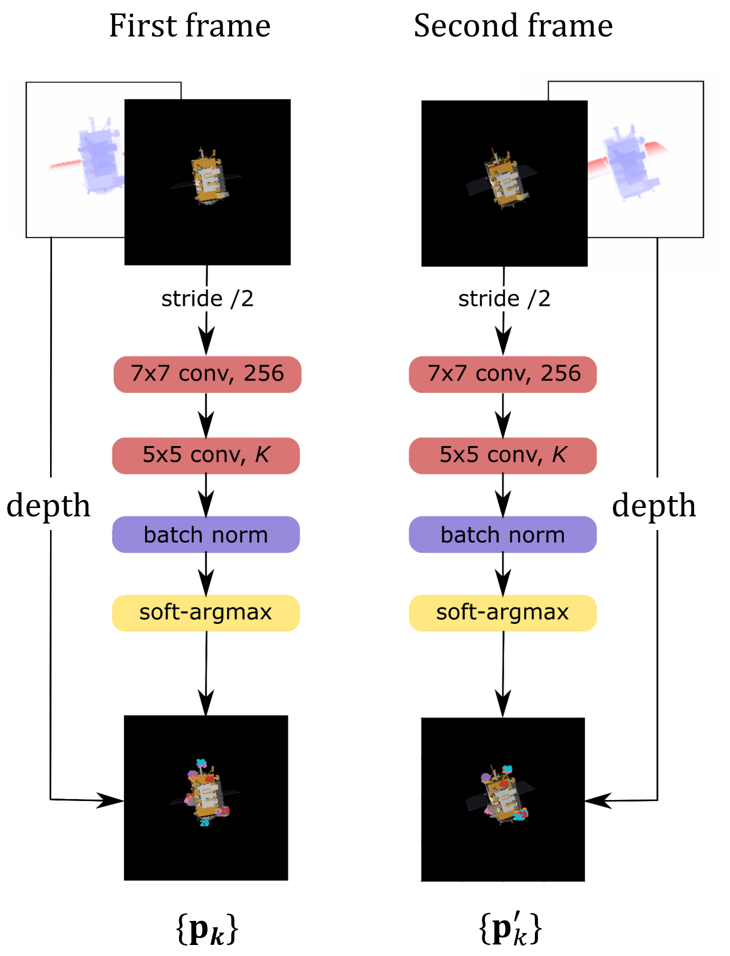


FIGURE 4.5: The feature extraction network architecture.

In the network shown in Figure 4.5, we have a convolutional layer with 256 filters of size 7×7 followed by a 5×5 convolutional layer. The final convolutional layer must have K filters, where K is the desired number of features to extract from the image.

In each of the two branches, the input images are passed through the convolutional layers to obtain a set of K feature maps, $\{\mathbf{M}_k\}$. These feature maps each correspond to the likelihood of a specific feature existing at each pixel in the image. A batch normalisation layer is included at this point to standardise the outputs. The landmark locations are then extracted from the feature maps by a spatial soft-argmax operation (Honari et al., 2017).

The soft-argmax layer aims to find the coordinates of the highest peak in the feature map. The function consists of a softmax operation over the image dimension - this determines the areas of maximum intensity in the feature maps, which should correspond to the location of the image feature - followed by a spatial argmax, which returns the coordinates of the location of maximum intensity in the feature map. Unlike a simple argmax, the soft-argmax is fully differentiable which ensures that the network is end-to-end trainable.

First, the softmax function is applied to the two-dimensional feature maps to obtain a probability distribution over the height and width:

$$S_k(u,v) = \frac{\exp(\mathbf{M}_k(u,v)/T)}{\sum_{u=1}^{H}\sum_{v=1}^{W}\exp(\mathbf{M}_k(u,v)/T)} \tag{4.1}$$

where $\mathbf{M}_k$ is the k-th channel of the feature map, with H and W being the height and width of the map, and $S_k(u,v)$ the value of the probability distribution of feature k at the pixel (u,v). The temperature, T, is a parameter which can be set beforehand or learned by the network. Summing this probability distribution over the height and width provides the (x,y) location of the maximum of each feature map in the camera view frame. This is achieved by passing it through a spatial argmax layer. The output of this layer is considered to be the position of the landmark at index k, denoted as $\mathbf{p}_k = (x_k, y_k)$.

The result, after passing the two images through the full network, is two ordered sets of K feature coordinates. This is passed to the second stage of the network for rotation estimation, as will be discussed in the next chapter.

Additionally, by observing the magnitude of the peak in the output of the softmax layer, it is possible to obtain a numerical value for the confidence in that feature. A feature with high confidence is more likely to correspond with a strong image feature such as a corner or edge. This value is used in the loss function (in Section 4.2.2) to encourage stronger features and in the rotation estimation stage (in Section 5.2.2) to reject weaker features.

4.2.1.2 Network Design Advantages

The feature extraction network is very shallow, with the effective size instead depending on the layer widths. There are a number of reasons for this design choice. The network is designed for use in the general problem, where there exists no a-priori knowledge of the target geometry. In addition, the task is one of rotation estimation rather than pose estimation. For these reasons, it is very important that the network extracts coordinates of simple image features, such as corners and edges. These are more useful features to describe the rotation compared with the more arbitrary features which tend to be provided by deeper networks. Similarly, pooling layers are avoided as these have been shown to lose spatial information.

In addition, rather than training a neural network to perform the full task of rotation estimation from sensor data directly, the network is instead only responsible for extracting the coordinates of the features. The network learns to extract optimal image features to describe the rotation, without performing the task of rotation estimation within the network. This is instead delegated to geometric algorithms, which are described in the following chapter.

The result of this design is that the network learns to extract geometric image features regardless of the specific target in the image, as the network is trained only to find the coordinates of key feature points rather than learning the properties of the target in the data. This will prevent overfitting to the training data and increase the robustness of the approach in different, previously unseen, situations.

The network design also provides further advantages. The model can take input from various different sensors, including RGB-D, LiDAR and monocular cameras, using the identical approach and simply finetuning on the different data. Due to its simplicity, the model is also fast to train. Finally, the model performance can be easily adjusted; for example, number of features extracted may be selected by adjusting the number of filters in the final convolutional layer.

4.2.2 Loss Terms

During training, a neural network attempts to learn weights which minimise a loss function. The selection of the loss function is therefore very important to the performance of the model. In this section, we detail the process of constructing this loss function. Several terms are selected to encourage the network to extract features which are inherently useful for the task of rotation estimation.

Initially, the feature extraction network alone is pre-trained to begin learning to extract strong, well-matched features. To this end, the main loss term is specifically designed to produce features which follow the rotation of the target over the time step.

Once a set of learnt feature locations $\{\mathbf{p}_k\}$ are available for a given image, and if the quaternion $\mathfrak{q}$ describing the *true* rotation between the two considered images is also known, from the data labels, then one would expect that the set of features in this first image, $\{\mathbf{p}_k\}$, each rotated by $\mathfrak{q}$, would result in the features from the second image, $\{\mathbf{p}'_k\}$. Therefore, the loss term L_{rot} is calculated as the difference between this expected result and the features observed in the second image:

$$L_{\text{rot}} = \sum_{k=1}^{K} \left| \mathfrak{p}'_k - (\mathfrak{q}^* \otimes \mathfrak{p}_k \otimes \mathfrak{q}) \right|^2 \tag{4.2}$$

where $\mathfrak{p}_k$ and $\mathfrak{p}^{'}_k$ are the feature vectors from the first and second image inputs, respectively, expressed in quaternion form; $\mathfrak{q}^*$ is the conjugate of $\mathfrak{q}$; $(\otimes)$ denotes the quaternion product; and the operation $\mathfrak{q}^* \otimes \mathfrak{p}_k \otimes \mathfrak{q}$ refers to a rotation of the vector $\mathbf{p}_k$ by the quaternion $\mathfrak{q}$.

With this loss term alone, there would be a risk that the network could break down by finding a local minimum where all features are at the centre of rotation, in which case L_{rot} would tend to 0. This can be prevented by incorporating an additional separation loss term, L_{sep} (Zhang et al., 2018). This term encourages the features to spread out over the image by preventing them from clumping together. The separation loss term is given by

$$L_{\text{sep}} = \sum_{k,k'=1;\, k \neq k'}^{K} \exp\left(- \frac{|\mathbf{p}_{k'} - \mathbf{p}_k|^2}{2\,\sigma_{\text{sep}}^2} \right) \tag{4.3}$$

where σ_{sep} is a weighting parameter.

As discussed in the previous section, we would expect that a higher peak in the feature map will likely correspond to a strong feature. Therefore, we include a further loss term to look at the concentration of the feature map around the feature location, termed as L_{conc}. The concentration of each feature map is computed by applying a Gaussian mask centred at the feature location across the output of the softmax layer. The loss term is then calculated as

$$L_{\text{conc}} = \sum_{k=1}^{K} \exp\left(- \frac{1}{2\,\sigma_{\text{conc}}^2} \sum_{u=1}^{H} \sum_{v=1}^{W} \mathbf{M}_k(u,v)\; G_{(x_k,y_k)}(u,v) \right) \tag{4.4}$$

where $G_{(x_k,y_k)}$ is a Gaussian mask centered at the feature location (x_k, y_k) and σ_{conc} is a weighting parameter. The value $\sum_{u=1}^{H} \sum_{v=1}^{W} \mathbf{M}_k(u,v)\; G_{(x_k,y_k)}(u,v)$ is the same as the confidence value which is assigned to each feature pair. The proposed loss function for the feature extraction task is the sum of the three terms described above, and therefore presents adjustable weighting factors that allow for fine control over the output of the feature detector.

After pre-training the network to extract useful image features, the entire model - including feature extraction and rotation estimation stages - is trained on the full task of estimating the rotation of the object. In this case, the loss function will instead be a measure of the difference between the true and estimated rotations. This is described in the following chapter, in which we address the rotation estimation approach.

4.2.3 Training and Validation

The model is implemented using PyTorch's machine learning framework (Paszke et al., 2019) and run on the University of Southampton's high-performance compute cluster, Iridis. Training and testing scripts are constructed which enable investigations of its performance under different parameters. The parameters associated with the model can be gathered into the following categories:

- **Network architecture.** The network can be adjusted by varying the number of convolutional layers in the network, as well as the number and size of filters in each layer. An important parameter is the number of filters in the final convolutional layer, K, as this decides the number of feature locations to extract. These parameters are crucial for determining the optimal network structure.

- **Training hyperparameters.** These can include the batch size, number of epochs, learning rate and the dataset used. These parameters affect how the network learns; for example, a larger batch size will speed up training at the cost of higher memory consumption, while the learning rate controls how quickly the model adapts during training.
- **Loss function.** Each of the loss terms outlined in Section 4.2.2 has an adjustable weighting parameter. Changing these affects the learned weights and therefore the performance of the network. For example, putting more weight on the L_{sep} term will encourage the network to extract features which are more spread out, possibly at a detriment to the quality of the features.

For many of these parameters, it is the case that the optimal value will not be known, so a number of experiments are conducted to determine these. The investigations are detailed in Section 4.3.

The datasets used in training and investigations are constructed using the simulation framework described in Chapter 2. The training data consists of approximately 60,000 pairs of images, taken from around 100 different simulation cases, with differing target types, positional and rotational states, and lighting conditions. One tenth of the data is used for validation; this data derives from an entirely separate simulation case to any of those used in training. Testing datasets have also been constructed with specific parameters, in order to validate the model's performance under different conditions.

As stated previously, the model is trained in two parts. This training approach is taken since it was found that the task of estimating the rotation from images is a very challenging one to learn with no initialisation. Instead, the feature extraction is first trained alone against the loss terms detailed in Section 4.2.2. The network is able to learn a response to this task, extracting low-level features which are correctly matched between the two images. The entire model, including the rotation estimation stage detailed in the following chapter, is then trained to find features which are most useful for describing the rotation. These are referred to as the pre-training and finetuning operations. By initialising the finetuning with the weights learned during pre-training, the model is able to better learn to extract the optimal features.

After training a model, the learned weights can be saved to a file and reloaded at a later point to either finetune the model further, or to test its performance on a test dataset. The following section details several of the tests which were performed to determine the best parameters for the model, along with a comparison with conventional feature extractors.

4.3 Investigations and Results

A number of investigations have been carried out on the developed feature extraction model. The details of its operation are illustrated by looking at the outputs of the inner layers, which can give insight into the optimal network structure. In addition, the network contains several adjustable parameters which have an impact on its performance, such as the loss weights and network size; the best values of these can only be found by experimentation. Finally, the performance of the model is analysed, by investigating its generalisability to different data and comparing the accuracy with conventional approaches.

4.3.1 Testing Methodology

In all investigations, the network is passed a dataset of test images which have been generated separately from the training data, so the same situation will not have been seen by the network during its training. The test datasets each contain 300 images and a separate dataset is used for each case, for example for different targets or angular rates.

For ease of use and to speed up the investigations, a test script has been constructed which is capable of several tests with different parameters at once, including training the models, running all tests and collating the results and model parameters for plotting. However, since this is a deep learning model, requiring reconstruction and training for each change in the parameters, the investigations are nonetheless slow to run and so the number of test cases must be limited. Similarly, analysis code has been developed which integrates with the output of the tests, allowing for rapid investigation of the results.

In the results in this section, the accuracy is measured by the loss in the prediction of the rotation. For now, it is not necessary to give details on the rotation estimation process, since this will be detailed in the following chapter; instead, the loss can be considered to be a measure of the usefulness of the features in describing the rotation of the target. The loss of each prediction is calculated as the distance between the true and predicted rotation quaternions at that step,

$$\theta_{\text{loss}} = 2 \arccos \left(\left| \mathbf{q}_{\text{true}} \cdot \mathbf{q}_{\text{pred}} \right| \right) . \tag{4.5}$$

This loss is averaged over the entire 300 image dataset for each test case. In some cases, to provide a more meaningful comparison of the performance at different angular rates, the loss angle is normalised by dividing by the true rotation angle of the target over the timestep.

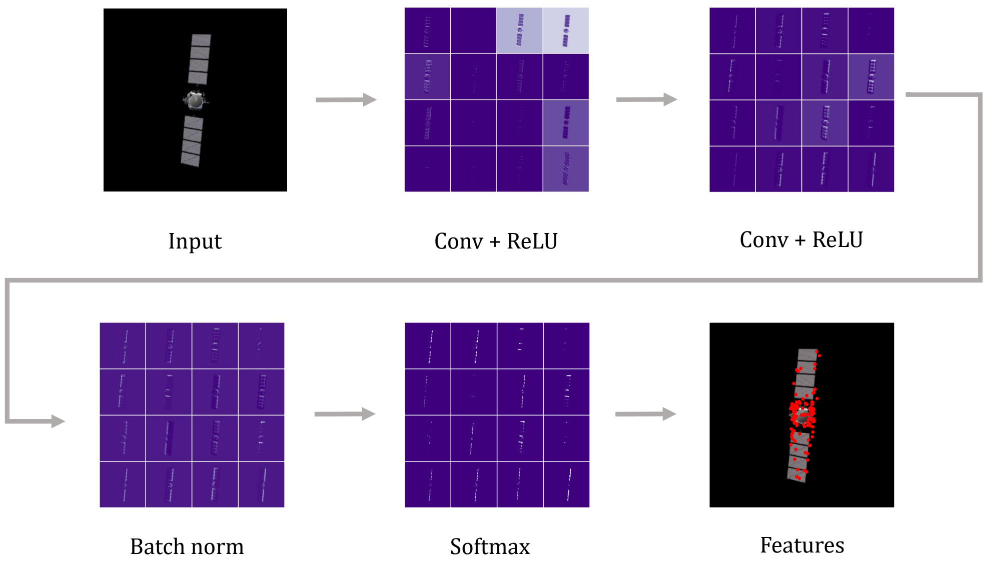


FIGURE 4.6: A small section of the feature maps at each layer of the neural network feature extractor.

4.3.2 Neural Network Structure

We begin the analysis of the approach by investigating the structure of the network, to ensure that the model performs as expected. In addition, some assumptions were made in the previous section about the optimal network architecture, based on the theory; these assumptions are verified by comparing the results between several models with different architectures.

4.3.2.1 Visualisation of the Feature Extraction Method

The neural network-based feature extraction approach can be illustrated by observing the outputs of the inner layers. This allows us to visualise what is happening inside the network and explains how the approach is able to find the location of important image features, using a few convolutional layers. Figure 4.6 illustrates the outputs of all layers in the shallow network shown previously in Figure 4.5. Only a small patch of the inner layers is pictured, showing the outputs of 16 of the filters in each layer, and we look at only one of the two network branches; this is simply to allow for a more clear illustration. The inner layers are visualised as a colour map, where a higher value in the feature map corresponds to a lighter pixel.

As shown, the input image is passed through two convolutional layers to extract the low-level image features. Any number of convolutions could be used, of course, though more layers would be more difficult to visualise and could lead to more abstract features. Moreover, after only two convolutional layers, the feature maps can already be seen to concentrate around the corners and edges of the object, as desired. A batch normalisation layer normalises the intensities of each filter across the whole batch of images, before passing the result to the `softmax` layer. As explained previously, the `softmax` aims to find the points of highest intensity in the feature maps; these can be seen as the brightest spots in the visualisation. Finally, the location of these maximum points in image coordinates is determined by an `argmax` function. These are considered to be the position of image features. Overlaying these feature locations onto the input results in the final image in the figure.

From the selection of loss terms used in training the model, we would expect that the feature points will be spread out over the object, corresponding to strong features such as corners and edges, and be in useful locations for describing the motion of the object. This appears to be the case in the example. The features are mostly clustered about the centre of the satellite; this is clearly providing stronger features than the long solar panels. However, there are some features spread out along the entire object, thanks to the use of the separation loss term.

The next task is to verify that the features extracted from each of the two network branches can be matched correctly by their index. In order to demonstrate this, in Figure 4.7 we plot the feature points using a sequential colour scheme. Due to the large number of features, the colours must be repeated; however, this should give an indication that features are correctly matched if groups of features in similar locations on the object also have the same colours, in each of the two input images.

We also use this figure to illustrate the confidence of the network in each extracted image feature. The "confidence" of a feature point is a measure of the strength of the feature and is given by the magnitude of the maximum point in the softmax layer. This is used in calculating the loss term L_{conc} and can give an indication of the strongest features extracted from the image, which we use in the following chapter to reject the weakest few. In the figure, the confidence value is visualised by the opacity of the feature point, where the weaker features are more transparent.

From observation, it appears that the network is functioning as theorised. The features at the same index in each output correspond to the same location on the target, and these features follow the motion of the target over the timestep. This is exactly the desired response for a feature matching approach, as the motion of the features will provide an accurate indication of the object's motion.

In addition, Figure 4.7 shows that the features at the corners of the object are much stronger than those along long edges. This is likely because there is a less well defined maximum point on the edges, so the exact location of the image feature is somewhat ambiguous. This result also suggests that the loss term L_{conc} could be tuned to encourage features to focus more on these areas, if desired.

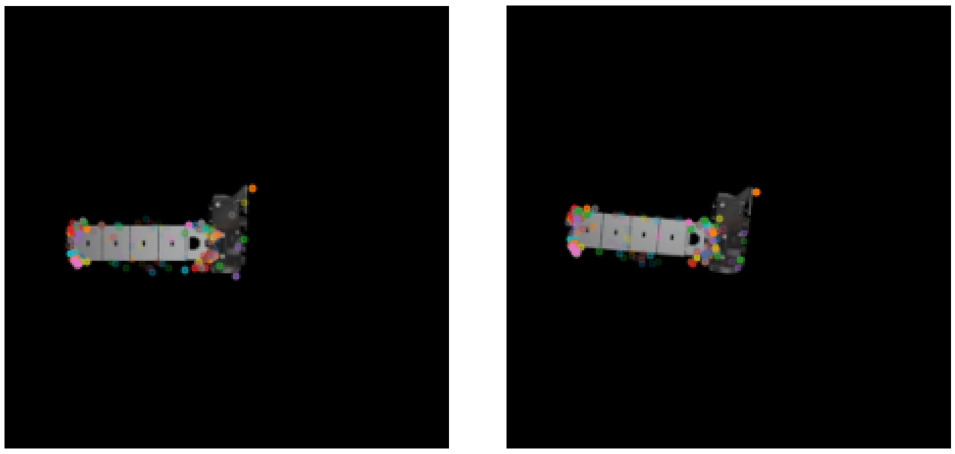


FIGURE 4.7: A representation of the network's ability to implicitly match features in two images, between which the target has rotated by 7°, using only the index in the output. The features are plotted using a qualitative colour map, so that a similar feature with the same colour in both images should indicate a good match. The opacity of each feature point is equal to its confidence value.

4.3.2.2 Network Architecture

In the investigations above, the network used is the two-convolution network shown in Figure 4.5. However, other network architectures have also been implemented and their performance investigated. Figure 4.8 illustrates the accuracy of several architectures, with different numbers of convolutional layers and with or without pooling layers. Previously, we proposed that shallow networks without pooling would perform best at extracting low-level image features; the goal of this investigation is to verify that assumption and provide a numerical evaluation of different architectures.

The accuracy of each network is evaluated by the normalised loss, which is the angular distance between the true and predicted rotation quaternions divided by the rotation angle over the timestep in the data. This is averaged over the full 300-image dataset for each rotation angle plotted on the x-axis. In all datasets, the same target is used, in this case the "Calipso" satellite, with identical initial states. The response of the model to the rotation angle and other parameters will be discussed later; in this investigation, we are more interested in the difference in performance between the different networks.

By comparing the different network architectures, it can be observed that the shallower networks, with fewer than six convolutions, appear to perform slightly better overall. However, the test with four convolutions outperformed that with only two, suggesting that there is a trade-off occurring as the number of layers is increased. This makes sense, as a deeper network will have more parameters and therefore be

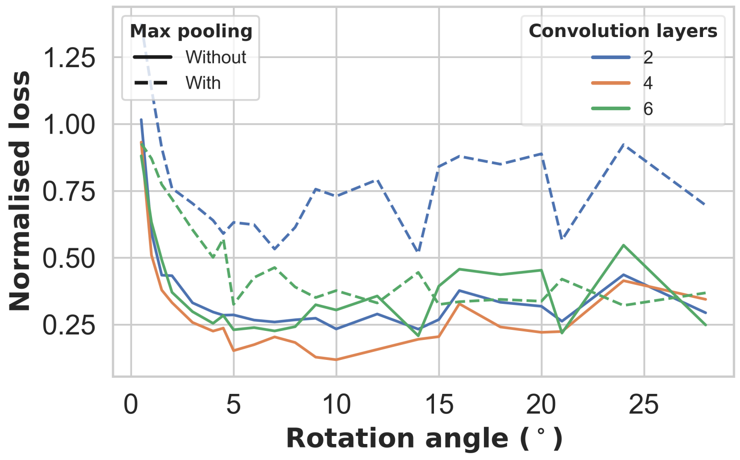


FIGURE 4.8: Loss in the prediction of rotation using different network architectures, averaged over the test dataset and normalised by the true rotation angle.

more able to learn, while on the other hand, a deeper network could lose positional information and result in more abstract, high-level features. In addition, adding pooling layers appears to reduce the accuracy, as predicted, although interestingly this has a smaller effect in the deeper network. This is perhaps due to the fact that this network does not use the exact feature location as much, instead finding a different, more abstract way to learn information about the object's state.

In order to gain more insight into the effects of increasing the network depth, Figure 4.9 plots the extracted features onto the input image, using several networks with different architectures. In the first row, all feature points are plotted with equal intensity, while in the second, we again scale the intensity of the plotted point by the strength of the feature, given by its confidence value. This provides an indication of both the ability of the network to find useful image features.

The most obvious observation from this figure is that the networks with pooling layers result in a large spread of feature locations which do not match up well with the object, with low confidence values overall. From this, it is clear to see the reason for their poor performance in the previous investigation. However, we also see that the feature locations tend to spread out from the object as the number of layers increases. This is a dangerous situation because, even if the features rotate with the object correctly, the depth readings at each location will not be accurate.

The reason for this outcome is illustrated in Figure 4.10. Here, we show a small section of the outputs of the first and last convolutional layer in the six-convolution network. This is repeated for the same network but this time including pooling layers. After several convolutions, a noticeable amount of distortion can be observed at the edges of the object. This causes the edges and corners to appear to be further apart than in

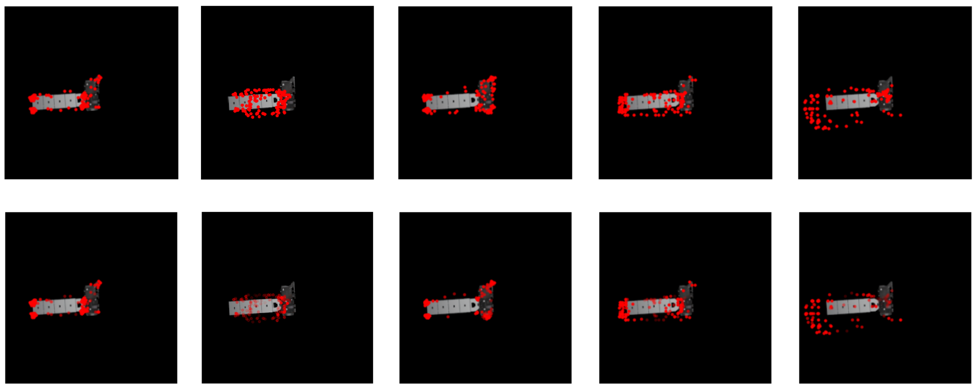


FIGURE 4.9: Comparison of the extracted image features using different network architecture. Images from top to bottom: the input image annotated with the feature locations; the same as above, where the opacity of each feature point is given by its confidence value. Networks investigated, from left to right: two convolutional layers; two convolutions with pooling; four convolutions; six convolutions; six convolutions with pooling.

reality, resulting in the feature points falling outside the boundaries of the target in the previous figure.

On observing this result, we add a further parameter to the model which scales the extracted feature locations inwards towards the centre of the image, by a scalar amount which is learned during training. While this does slightly improve the prediction accuracy, it is still not on par with the shallower networks. This could be because the effect is different for each filter in the convolutional layers, so a single scalar value is not sufficient to capture and correct for it.

On the other hand, from observing Figure 4.9, the deeper networks appear to produce feature points which spread out to cover the entire target, moreso than the shallow two-convolution network. This could be due to the fact that these networks have more trainable parameters, thus can better learn a response to the given loss terms, such the separation term L_{sep}. We attempt to obtain a similar result by increasing the width of the network, by increasing the number of filters in each layer, thereby increasing the network's effective size. Again, this does not have the intended effect, with no noticeable improvements. Therefore, if a better solution can be found for the spreading of feature points, we believe that the extracted feature locations might be improved by increasing the network depth.

4.3.3 Model Parameters

The neural network has a few parameters which can't be learned and instead must be provided prior to training. Due to the difficulty of estimating the impact of these

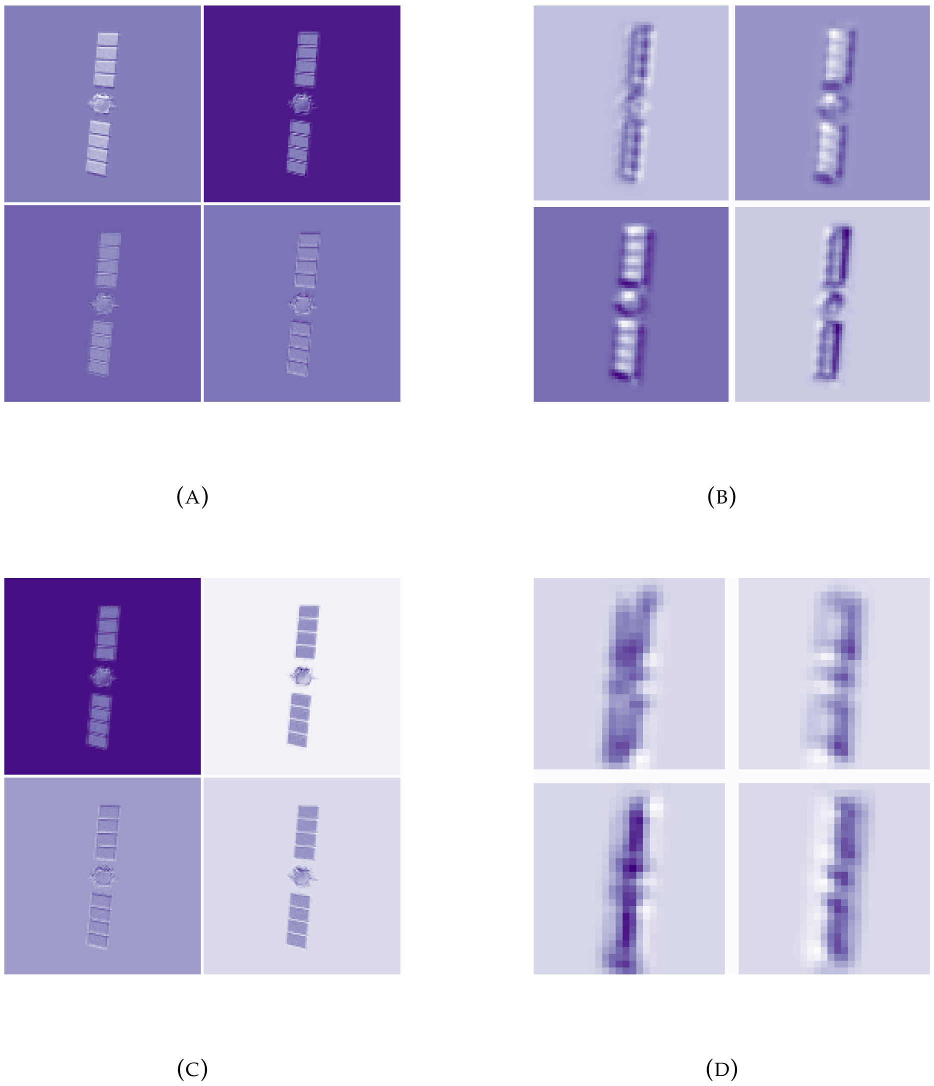


FIGURE 4.10: Visualisation of the outputs of hidden layers in deeper networks. (A) and (B) show the output of four filters in the first and last convolutional layers respectively, in a network containing six convolutions. The second row is the same, except that the network also contains pooling layers.

parameters, the best way to determine the optimal values is through experimentation. Unfortunately, since these parameters must be defined at initialisation, the network must be completely re-trained in each test case. This is a very time-consuming process and so the number of test cases must be limited. This section discusses the experiments carried out on two network parameters: the weights of the loss terms and the number of filters in the final layer (which is equivalent to the number of features extracted).

The loss weights are an incredibly important parameter since these determine the quality of the extracted feature points. The two additional loss terms described in Section 4.2.2, L_{sep} and L_{conc}, each have a weighting term as shown in Equations 4.2.2

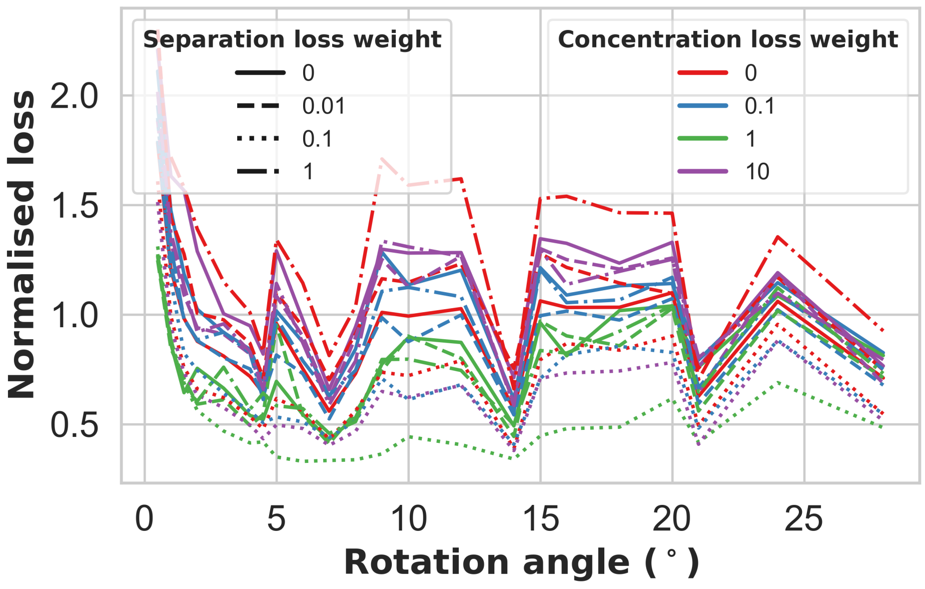


FIGURE 4.11: The effect of the loss weights on network performance.

and 4.2.2. Adjusting these two parameters adjusts the relative weighting of the three loss terms in the loss function, causing the network to evaluate certain types of features more highly that others.

Figure 4.11 demonstrates the effect of changing the loss weights on the accuracy of the network. Four different values of both weighting terms are investigated, in all possible combinations. When the loss weight is 0, this means that that loss term is not used; if both are 0, then the rotation loss, L_{rot}, is the only term in the loss function.

Changing the weighting values has a significant impact on the model's performance. In fact, in many cases, the normalised loss does not drop much below 1; this means that the loss is equal to the rotation angle. This is due to the fact that there is a dangerous local minimum to the rotation loss in Equation 4.2, in which the network can predict a rotation of $0°$ and thereby have a loss of 0 in this term. The weighting of the other terms is therefore very important to ensure that the network can escape this local minimum.

The significant variation in the results when changing the weightings suggests that further analysis would be worthwhile in this case, since it is likely that the optimal weightings will be somewhere between the values chosen in this investigation. However, due to the time-consuming nature of the experiment, in our case we select the values of σ_{sep} and σ_{conc} as 0.1 and 1 respectively for all further analysis.

The next network parameter we are interested in is the number of features extracted by the network. As mentioned, this parameter is equivalent to the number of filters in the final convolutional layer, since one feature point is extracted for each filter. Since

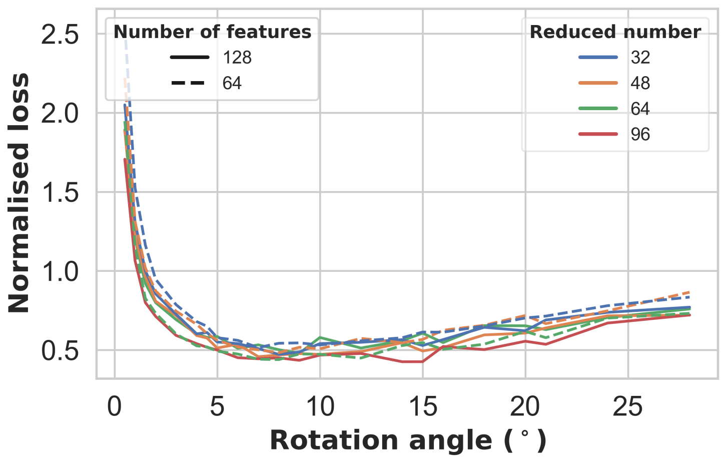


FIGURE 4.12: Loss in the prediction for different numbers of extracted features. The total number of features extracted from the network is reduced by rejecting those with the lowest confidence values, down to the reduced number.

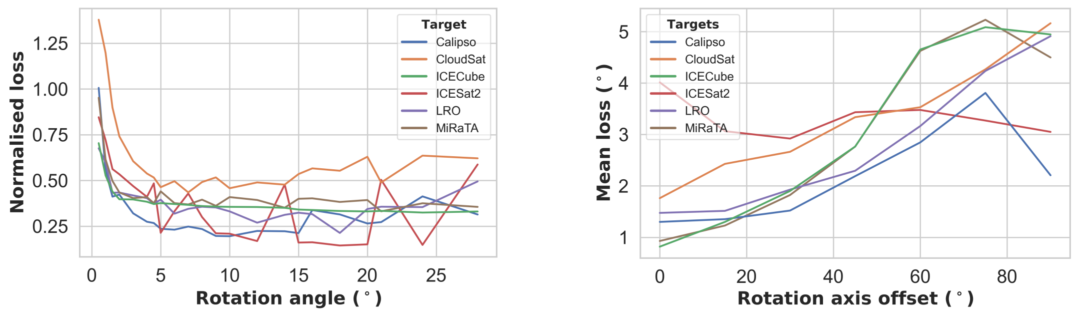
deep learning processes are designed to be fully parallelised and run on a GPU, increasing the number of filters should not increase the computation time, unlike a conventional algorithm. However, more filters results in a larger number of trainable parameters which requires more memory, so the maximum number of features will be limited by the capacity of the on-board GPU.

It would be expected that a larger number of extracted feature points would provide more information about the target, and therefore lead to a higher accuracy of the prediction. However, this effect is likely to be less impactful as the number of features increases. In fact, Figure 4.12 demonstrates that the consequence of changing the number of features is very small. This graph plots networks with different numbers of filters and also different numbers of low-confidence features rejected by the model. However, any observable difference in performance is not large enough to be attributed to anything more than randomness in the model training.

4.3.4 Generalisability

We now investigate the network's response to different conditions. These experiments are conducted using the partial simulation mode of the data generation software, allowing the construction of test datasets in which specific conditions are captured, such as lighting variations and different angular rates. Due to the approach used, which decomposes the information at the intermediate stage to simply the locations of low-level image features, we would expect that our model generalises well to unseen data, since it should not overfit to specific target geometries or features. However, the use of low-level features may cause it to not be robust to certain challenging situations, such as occlusion of surfaces or cluttered image data. The results shown in this section attempt to determine the advantages of our approach as well as addressing the areas which may need improvement.

Figure 4.13 illustrates the results of an investigation into the effects of different rotational states on the accuracy of the model. First, the normalised loss is plotted against the rotation angle; the normalised loss is used as it enables the comparison of relative accuracy across all datasets. Next the loss is investigated when changing only the axis of rotation. In this investigation, the rotation axis begins along the z-axis of the view direction. This is the simplest case, since all motion is captured within the view plane and there should be a lower risk of occlusions. While keeping the rotation angle over each step constant, the rotation axis is offset from this point gradually, up to 90°, at which point it is along the y direction. This is a more challenging situation because the motion of the object is mostly towards or away from the the camera, so occlusions and disappearing edges are more likely, and most of the information must be taken from the depth, z, at each point, rather than the (x, y) location.



(A) Normalised loss with rotation angle

(B) Loss with rotation axis, for a 4° rotation

FIGURE 4.13: Investigation of the model's generalisability to different rotational states.

From this experiment, we first notice that the normalised loss is significantly higher at very low rotation angles. This can be explained by the fact that, in this case, very small errors in the feature locations will have a larger impact, since the motion of the features over the timestep is very small. On the other hand, as the rotation angle increases, matching the features becomes more challenging. However, there is not a clear increase in loss with the higher rotation angles with any of the satellite targets, suggesting that the model is able to match features even between images with significant variation in orientation. Since the normalised loss remains somewhat consistent over a wide range of rotation angles, and there should be advantages at both small and large angles, the overall accuracy of a prediction may be improved by combining information from several different step sizes.

In the second plot, an obvious trend can be seen, with the loss increasing as the rotation axis is offset from the view z direction. This was somewhat expected due to the discussed difficulties with rotations which are not constrained to within the view plane. However, the magnitude of this response is higher than expected and implies a definite issue with the approach. It is determined that much of this is due to inaccuracies in the depth values. When observing the visualisation of the network in the earlier section, we noted that the feature locations would occasionally not match up exactly with the original image. While the motion of the feature points would still follow the target correctly, this offset can lead to significant errors in the depth readings, since these are taken by sampling the depth map at the specific location. It is therefore necessary to find a solution to this problem.

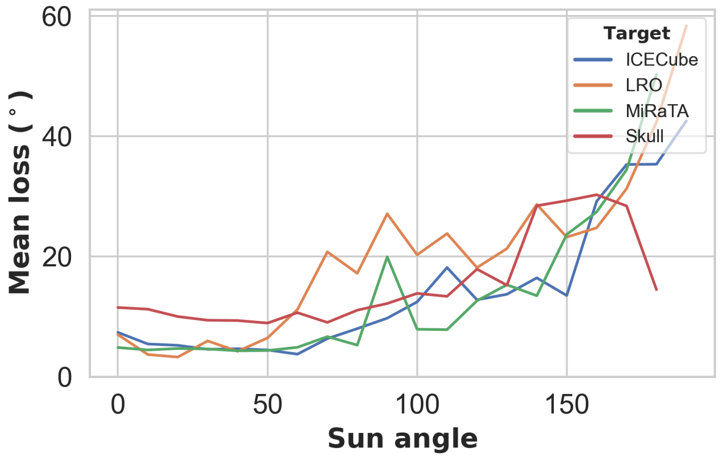
The above observations lead to two avenues of research to improve the model's performance. Both are investigated in detail in the following chapter. As the model's performance has been shown to vary significantly with the rotation axis, we clarify that, in all other investigations, the rotation axis is selected to be somewhere in the middle of this range. In this way, we attempt to capture the model's response while avoiding bias either way.

Next, we investigate the response of our model under different lighting conditions, since this is a critical challenge in space missions. Figure 4.14 contains a comparison of the mean model loss, averaged over the test datasets. In this test case, the illumination direction is rotated between 0° (behind the observer) to 180° (behind the target). The figure also shows the camera view under the change in illumination angle, to illustrate the amount of information present in the image. In the case where the Sun is behind the target, we do not consider the light incident on the camera from the Sun; we are only interested in the amount of illumination on the object in this study.

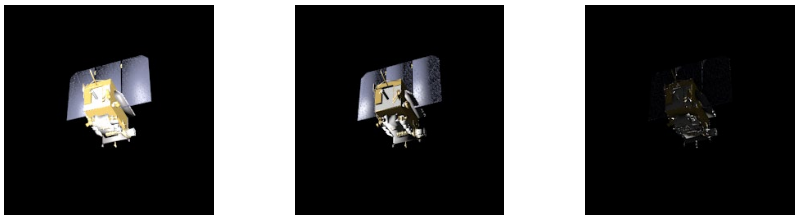
Unsurprisingly, the accuracy is highly dependent on the illumination of the object, since there is clearly much less information present in the latter cases. However, these situations are not realistic. In a real mission, the chaser will most likely be fitted with its own light source for illumination in situations where the lighting from the sun is insufficient. If we include a low-powered light source in our simulation when generating the data, we find that the accuracy becomes completely independent of the direction of the Sun: as long as the front surface of the object is lit up, changing the direction of the much stronger source of illumination has no effect.

One of the most important qualities of the model is its generalisability to unseen satellites. This approach aims to be applicable to any satellite target without reconfiguration or retraining, so it is crucial that it does not overfit to the training data. The approach has been designed with this in mind; by explicitly extracting low-level features, the model should be inherently more generalisable than a fully neural network-based approach.

Figure 4.15 demonstrates the outcome of an investigation into the generalisability of the model. Two training datasets were constructed, one containing all satellite targets and another which contains only a few targets. The same network is trained on each



(A) Mean loss when changing the illumination angle through 0° to 180°



(B) The view from the chaser satellite where the sun direction is 0°, 90° and 180°, respectively

FIGURE 4.14: Model loss response to changing the illumination angle, tested on datasets with rotation angles of 20° per step.

of the two datasets, resulting in two different models. The two models are then both used to predict the rotation of a number of different satellite targets, some of which were in both datasets and others in only the full dataset. This allows for the comparison of performance between models which have been trained on the same target as they are observing and those which have not.

For the targets which were not in the smaller training data (in the plot, these are the final three, with names in red), there is a slight improvement when using the model which saw those satellites during the training. However, a similar result can be seen for the majority of targets, even those in both datasets. This could simply be down to the fact that a more diverse training dataset results in a better trained model in any application using deep learning. Nonetheless, the lack of a significant variation between the two models is a promising result, suggesting that our method generalises well to unseen geometries.

The final experiment in this section investigates the ability of the model to extract useful features in the presence of unwanted background information. Similarly to above, the network is trained on two different datasets. In the first, the simulation is constrained to views in which the Earth is not visible in the background. In the second, all views are allowed, so some of the images will contain the Earth as well as

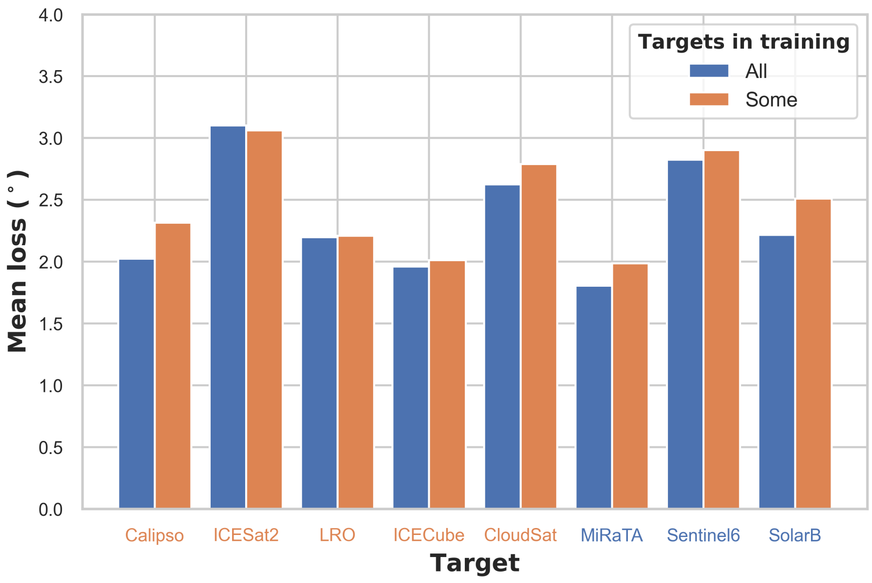


FIGURE 4.15: A plot of the loss when observing different satellite targets, using networks trained on either all or some of the targets, observing a rotation of 5° per step. The latter dataset contains only the first five targets, with names in blue.

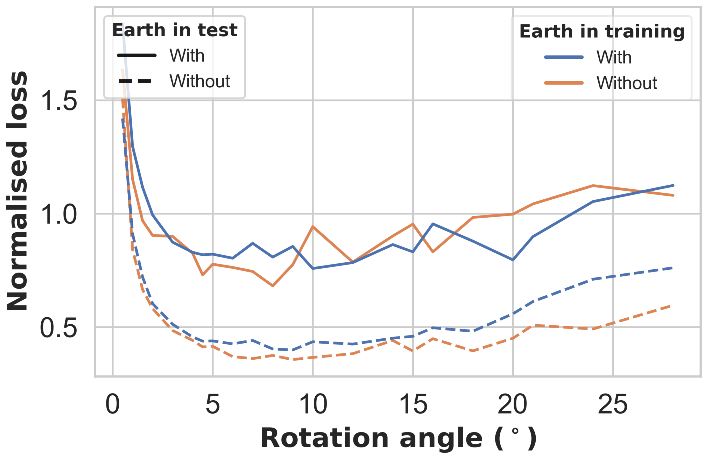


FIGURE 4.16: Model loss when the Earth is visible in the background, with and without training on datasets containing the Earth.

the satellite. We then test each model on test datasets with or without the Earth in the background. The result is plotted in Figure 4.16.

Again, this result demonstrates a lack of robustness of our model. The accuracy is significantly reduced if the images contain anything beyond just the satellite itself. This is because the convolutional filters are acting simply as low-level feature detectors, so they find features in the background as well as on the object. Unfortunately, including this situation in the training is not sufficient for the network to learn to ignore the background. This could be because the combination of loss terms used in training does not prevent the model from extracting features in the background; the separation term would decrease and concentration term would not increase in these cases, so the increase in the rotation loss term only appears to be insufficient to train the network against this.

A further investigation with different weightings of the loss terms may improve the accuracy in this situation. As well as this, we suggest a number of other solutions to the problem. The depth map could be used instead of the RGB camera image to ensure that only the nearby target is considered. Alternatively, feature points on the background would not move in the same way as those on the target; these could therefore be rejected using an outlier rejection scheme. Other solutions could involve cropping the image around the target, for example using an object detection network, or simply increasing the depth of the network in the hope that this improves the learnability of the model and enables it to more intelligently extract features.

4.3.5 Comparison with Conventional Feature Extractors

Finally, we compare the siamese neural network approach to feature extraction and matching with conventional algorithms for the same task. The two algorithms we implement are: using an ORB feature detector with brute force feature matching, and SIFT feature descriptors with FLANN-based matching. Both of these are commonly used algorithms for this task, and are described in Section 4.1.4.

Besides the feature extraction method, all other elements of the approach are identical in each test case: the locations of the matched feature points are passed to the same rotation estimation algorithm. Therefore, the comparison simply determines which approach extracts the most useful, best matched feature points for describing the object's rotation. This comparison is shown in Figure 4.17.

This model appears to have an improved performance when compared with the conventional approaches. The significant improvement over the same approach using conventional feature matching algorithms shows that our neural network-based approach is capable of learning intrinsically more useful image features for this specific problem. The computer vision approaches may have difficulty matching features in this challenging environment, since the matching is based on descriptions of the surrounding pixels and many of the features look very similar. By training on images in this environment, we can learn to better match these features, and to extract features which best describe the rotation.

This model also has further advantages over the conventional approaches. While the latter require a greyscale image only, our model can accept any form sensor data, including combinations of depth and visual sensors. The requirement is simply that the model be trained or finetuned using the specific input data. In addition, the

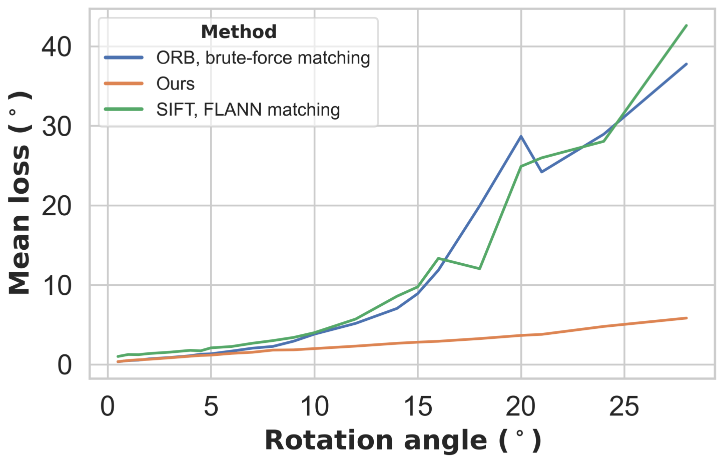


FIGURE 4.17: Comparison between our approach and two conventional feature extraction and matching algorithms: ORB feature detection with brute force matching; and SIFT features with FLANN-based matching.

number of features extracted can be altered without significantly impacting the computation time.

Another comparison of interest may have been between this model and deep learning pose estimation models, for example those used in the ESA pose estimation challenge, since this is a similar task. However, a direct comparison cannot be made between this model and those used in the challenge, since the problems which the networks are being used to solve are quite different. These other models could not be directly applied to this data and task since they require a-priori knowledge of the target spacecraft with which to compare. Similarly, this model could not be used with the challenge data since it is only capable of predicting rotations, whereas the data provides only a single image from which to predict an orientation. Besides this, even if the model or data were adapted, this model would likely be outperformed by the alternative approaches which are designed for the specific task, due to the design of the network itself (the simplicity and shallowness of the network ensure better generalisation to unseen targets, but would be a detriment in the case that the target is known). For these reasons, there is no easy way to effectively compare this approach with the state of the art in pose estimation, since that is not the problem which it is attempting to solve.

4.3.6 Timing Analysis

While the results thus far are promising, of almost equal importance to the model accuracy is that the computational cost of the algorithm is not too high. In order to be applied to ADR missions, the algorithm must be capable of real time operation on on-board hardware with limited power availability. Therefore, we analyse the computational cost of our algorithm and compare it with the conventional approaches.

In a timing comparison, we find that our model falls very slightly behind the conventional algorithms in speed of computation, although it is still capable of real time operation. This is in part due to the fact that the rotation estimation algorithm is not well optimised for the GPU. We noted a sampling rate of 23Hz for our model, increasing to 30Hz if only performing the feature extraction. In contrast, when using the ORB feature extractor with brute force matching, we can achieve up to 50Hz for the feature detection and matching, though this is also reduced to 25Hz if we include the same rotation estimation algorithm. Analysis of light curves (Šilha et al., 2018) shows that the minimum observed period of rotation for an object in LEO is approximately 1 second, with most being significantly higher. Even in the most extreme case, a sampling rate of 20Hz would result in a rotation angle of 18° per step, which is within the range of reasonable accuracy for our model.

The proposed model also uses a GPU for the majority of the calculations while the conventional approaches use CPU processing. GPUs offer improved performance and energy efficiency for a lower cost when compared with CPUs, making them an attractive candidate for on-board computation in space (Kosmidis et al., 2019). For demonstration, the speed of operation of the model is compared on both a GPU and CPU. The results of this comparison are shown in Table 4.1.

Network size	128, 96	64, 32
CPU	793s	528.9s
GPU	26.80s	21.13s

TABLE 4.1: Comparison of the computation time for a dataset of 1194 image pairs, using a CPU or GPU computation. The network size is given by the number of filters in the final convolution layer and the number remaining after rejecting low confidence features.

This analysis demonstrates the effectiveness of the GPU in reducing the computational cost of the approach through parallelisation of the neural network. This furthers the case for on-board GPUs on satellites. However, the flip side is that this also shows that a GPU is necessary for this algorithm to achieve real-time performance. An interesting further analysis would involve running the algorithm on a small, low-power GPU as would be used on-board in real space missions.

Despite the benefits of our model when compared with the conventional approaches, there are nonetheless many improvements still to be made. Most importantly, the individual losses are too high to be used to reconstruct the attitude by integration, since the losses will balloon out of control. In the next chapter, we discuss the approach used for estimating the rotation, provide further in-depth analysis of the causes of the loss, and detail several of our attempts to improve the estimate using the features extracted by our network. Additionally, we investigate using a filtering scheme to improve upon the measurements over time, since it does not make sense to only use a single image pair to predict the rotation at each step given the stability of the situation.

Chapter 5

Rotation Estimation from Matched Landmarks

The model detailed in the previous chapter is capable of extracting matched feature points from two time-separated images. In this chapter, we look into methods to estimate the rotation using the observed motion of these feature points over the timestep. The aim of this investigation is to develop an algorithm which can be appended to the output of the CNN-based feature extractor, resulting in a model which is capable of predicting the instantaneous rotational rate of an object in real time. The instantaneous rotation can be integrated over time to reconstruct the attitude of the debris target.

The full model, including feature extraction and estimation stages, should be fully end-to-end trainable, so that the output of the latter stage can be used to train the feature extractor. This will enable the model to learn to extract features which are more useful for the specific task of estimating the rotation. However, this requirement poses certain conditions on the estimation algorithm. Since it must be integrated within the machine learning framework, the algorithm must be fully differentiable and parallelisable, with a fixed size of input and output vectors.

Different approaches of estimating the rotation are investigated which are based on either conventional algorithms or fully-connected neural networks. The implementation of the chosen approach is detailed, along with a method of detecting and rejecting poorly matched features from the previous stage. Finally, the performance of the full model is investigated and the results analysed.

5.1 Methods of Estimating the Rotation

The problem of estimating a rigid-body transformation which aligns two sets matched 3D points has a number of well-defined algorithmic solutions (Eggert et al., 1997). However, we also investigate several neural network approaches to the problem, since one of the aims of this work is to determine the applicability of machine learning in all stages of an autonomous navigation system.

It should be noted that the provided information is sufficient to also compute the relative translational motion of the target satellite. However, this has been well covered in past research and we consider it a less challenging problem, which will not provide as valuable an indication of the usefulness of our extracted feature points. In our work, we therefore focus our discussion and analysis on the sole problem of rotation estimation. Nonetheless, the methods detailed in this section determine the full transformation, including both rotation and translation, between the two sets of points.

5.1.1 Conventional Algorithmic Approaches

The goal of the estimation algorithm is to determine the optimal rotation and translation which matches the first set of feature points onto the second. This can be considered as a residual minimisation problem, which can be solved by a root-mean-square-deviation (RMSD) approach. Alternatively, iterative approaches have been used to gradually improve upon the estimate, with the trade-off of longer computation times.

5.1.1.1 Root-Mean-Squared Deviation Approach

Given two sets of matched points corresponding to the locations of extracted image features, we aim to estimate the rotation of the target over the time step during which the two images were acquired, using the RMSD algorithm. This approach provides the rotation which minimises the residual between the observed and predicted features, i.e. the difference between feature positions observed in an image and those predicted by rotating the features from their observed positions in the preceding image. This is relatively robust to noise in the position and matching of feature points, since the goal is simply to minimise a residual rather than solving for the exact transformation between points.

Conventionally, the residual minimisation problem can be solved using a singular value decomposition (SVD). However, Coutsias et al. (2004) have proposed an alternative approach to solve the RMSD problem using quaternion operations.

Singular value decomposition The aim of the SVD algorithm is to find the rotation matrix R and translation $\boldsymbol{t}$ which minimises the residual

$$\Sigma^2 = \sum_{k=1}^{K} \left\| \boldsymbol{p}'_k - (\mathrm{R}\boldsymbol{p}_k + \boldsymbol{t}) \right\|^2 \tag{5.1}$$

We must first find the centroid of the two sets of points, denoted as $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}'$, then calculate the 3×3 matrix

$$\mathrm{X} = \sum_{k=1}^{K} (\boldsymbol{p}_k - \bar{\boldsymbol{p}})(\boldsymbol{p}'_k - \bar{\boldsymbol{p}}')^T \tag{5.2}$$

where the superscript T refers to the matrix transposition. The rotation matrix $\hat{\mathrm{R}}$ can be calculated directly from the SVD of X.

$$\begin{aligned} \mathrm{USV}^T &= \mathrm{X} \\ \hat{\mathrm{R}} &= \mathrm{VU}^T \end{aligned}$$

This fails when the determinant of $\hat{\mathrm{R}}$ is less than zero, which means we have a reflection. In this case, the third column of V must first be multiplied by -1 (Arun et al., 1987). The translation is then found by

$$\hat{\boldsymbol{t}} = \bar{\boldsymbol{p}}' - \hat{\mathrm{R}}\bar{\boldsymbol{p}} \tag{5.3}$$

Quaternion approach In the quaternion approach, the residual computation is fully expressed in quaternion notation. In this case, the residual is instead given by

$$\mathrm{E} = \frac{1}{K} \sum_{k=1}^{K} \left(\hat{\mathfrak{q}}^* \otimes \mathfrak{p}_k \otimes \hat{\mathfrak{q}} - \mathfrak{p}'_k \right) \otimes \left(\hat{\mathfrak{q}}^* \otimes \mathfrak{p}_k \otimes \hat{\mathfrak{q}} - \mathfrak{p}'_k \right)^* . \tag{5.4}$$

The value of the rotation quaternion which minimises this residual can be found from the correlation matrix, C, between the two sets of 3D points. This optimal value is equal to the eigenvector corresponding to the maximum eigenvalue of the matrix F:

$$\mathrm{F} = \begin{bmatrix} c_{11} + c_{22} + c_{33} & c_{23} - c_{32} & c_{31} - c_{13} & c_{12} - c_{21} \\ c_{23} - c_{32} & c_{11} - c_{22} - c_{33} & c_{12} + c_{21} & c_{13} + c_{31} \\ c_{31} - c_{13} & c_{12} + c_{21} & -c_{11} + c_{22} - c_{33} & c_{23} + c_{32} \\ c_{12} - c_{21} & c_{13} + c_{31} & c_{23} + c_{32} & -c_{11} - c_{22} + c_{33} \end{bmatrix} \tag{5.5}$$

where c_{ij} are the components of the correlation matrix C.

The quaternion-based approach has a number of advantages. For the use case that we are interested in in particular, the main advantage is that there is no need to convert between different parameterisations inside the network, since the operations and outputs are all defined in terms of quaternions. This increases the speed of the computation and avoids potential ambiguities which can arise during the conversion.

5.1.1.2 Iterative Closest Point Algorithm

An alternative method for predicting the motion of an object is the iterative closest point (ICP) algorithm. Instead of using the extracted feature locations, ICP instead attempts to minimise the difference between two full point clouds. Unlike other solutions to this problem, there is no need to match the points beforehand; instead, the algorithm estimates the correspondence between points simultaneously to the transformation.

The ICP process starts with two point clouds, which might be generated by a LiDAR sensor at two different instances in time. In general, the process is as follows. First, points are selected from the full clouds, for example by using all points or randomly sampling a subset at each iteration (Rusinkiewicz and Levoy, 2001). For each point in the first set, a match is found by the closest point in the second set.

Given these two matched sets of points, an approach such as the RMSD above can be used to estimate the optimal combination of rotation and translation to minimise the distance between the matched points. The initial point cloud is then transformed by this rotation and translation, resulting in two new sets of points.

This results in the same situation as to begin with, except that the point clouds are closer together. The above steps are iterated until a solution is found within a threshold, or after a certain number of iterations.

The ICP algorithm can estimate the motion of a target with reasonable accuracy. There is also no requirement for feature extraction and matching, instead computing the result directly from the input point clouds. However, being an iterative process, it tends to be slower than a non-iterative approach. In addition, feature extraction can be useful in determining the features which are most useful for describing the motion, rather than using all points in the point cloud equally.

5.1.2 Neural Network Approaches

A neural network approach for estimating the transformation is also proposed. One possible solution involves constructing a fully-connected neural network to predict the transformation given the output of the feature extractor. However, the mathematics of the problem are well-defined once the sets of matching feature points have been extracted, as shown above, so the neural network approach is unlikely to provide an advantage in this case.

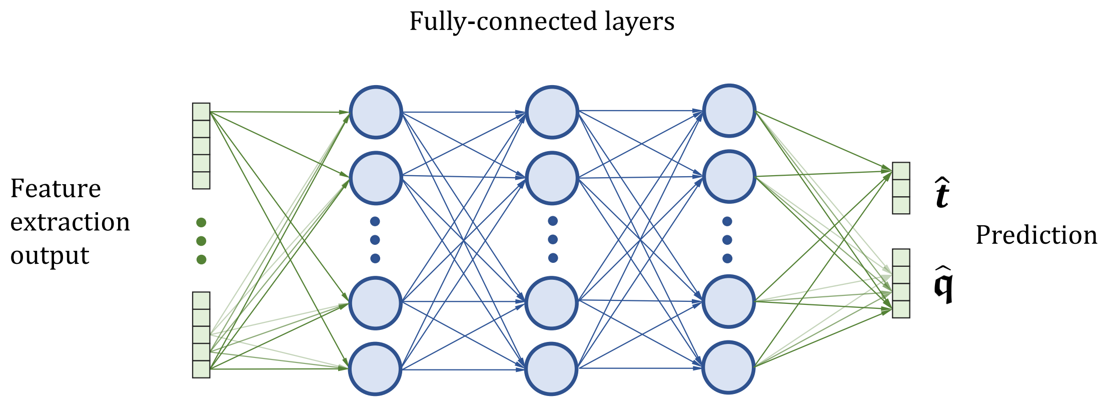


FIGURE 5.1: An example of a fully-connected prediction network.

Therefore, alternative methods are investigated which employ a fully neural network approach, including feature extraction and estimation within a single end-to-end network. A disadvantage of this approach is that it is more likely to overfit to the training data, and the problem is a challenging one so the network might struggle to initialise and begin learning a solution; on the other hand, it is able to use more information than only the feature locations to predict the output.

5.1.2.1 Fully-Connected Neural Network

In a classification problem, a few fully-connected layers are often appended to the end of the network. These layers are used to predict, for example, an object class, using the output of the feature extraction layers. A similar approach can be used in this case, where the final layer consists of a dense layer with length 7, consisting of the 4-element rotation quaternion, $\hat{\mathbf{q}}$, and a translation vector, $\hat{\mathbf{t}}$. An example of such a network is shown in Figure 5.1.

The fully-connected layers can be placed after the feature matching stage of the network described in the previous chapter. In this case, the network learns to correspond the motion of feature points with a transformation. However, as we discussed previously, there is likely to be little advantage in using a neural network for such a task since algorithms exist to solve this exact task optimally.

As an alternative, the input to the network could be taken from the full flattened feature map of the final layer, before the argmax function which extracts the coordinate points. This is more similar to other methods in literature for estimating the attitude of a target; several of the solutions to ESA's pose estimation challenge, such as that of Proença and Gao (2020), use a CNN followed by fully-connected classification layers to regress the satellite pose. However, these approaches have only been shown to work when the model has been trained with datasets containing the same specific target at different orientations, because the network has to learn to classify the object at a high level. Therefore, this network is unlikely to generalise well to different target geometries. In addition, the size of the final feature map must be sufficiently small before flattening it and passing through the fully-connected layers, otherwise the number of parameters in the network will become very large, increasing the computation time and memory requirements. Therefore, the size of the feature maps must be reduced, for example by using pooling layers - which, as we showed in the previous chapter, can result in a loss of important positional information.

The full network can be trained together, using as the loss function a measure of the distance between the prediction and the true values. Alternatively, the network can be initialised by pre-training only the feature extraction layers, then finetuning all together. Often, this is treated as a classification problem instead of a regression to overcome ambiguities in the attitude (Proença and Gao, 2020), by constructing classes corresponding to different orientations.

5.1.2.2 Global-Local Network

An alternative network structure is proposed for predicting a transformation directly from input images. This architecture aims to retain more information about the image than just the feature locations, without passing the entire feature map to the fully-connected layers.

The feature extraction and matching network from Chapter 4 is used to generate the matched feature points from the two input images. This network generates K 6-element vectors containing the positions and depths of each feature in the first and second image frame. Alongside this, a separate CNN performs a classification of the input images. The pre-trained VGG-19 network is used for this task (Simonyan and Zisserman, 2014), with the final fully-connected layer having a size of 64. This results in a vector of length 64 that contains global information about the images, such as the satellite geometry, lighting conditions or occlusions, which could be useful for generating the prediction.

The outputs of the two networks are concatenated together before being passed to the prediction network. Again, we use a simple fully-connected network for this case, as illustrated in Figure 5.1. This approach is able to use more information about the images than the conventional approaches, while retaining its generalisation capability by still using the feature matching approach and minimising the network size.

5.1.2.3 Pre-trained Networks

The final neural network method we investigate involves using pre-trained networks. These have the advantage of being much faster to train and are well suited to the problem that they have been trained to solve. However, no pre-trained models exist for our specific problem, so instead object detection networks are adapted to the task.

For this approach, we modify the ResNet model (He et al., 2015) to take two inputs. While this could be achieved by passing in a single 6-channel image, we instead modify the architecture so that it becomes a siamese network, similar to our own feature extraction network. However, in comparison with our approach, the two branches are concatenated together before the final convolutional layer. This means that the final convolutional and fully-connected layers together are used to predict the transformation.

5.1.3 Kalman Filtering

Kalman filters (KF) have been employed in attitude estimation problems for many years (Lefferts et al., 1982; Shuster, 1989; Choukroun et al., 2006), and remains a commonly used approach to this day. Given a sequence of noisy measurements, taken for example from an on-board attitude sensor, the role of the KF is to minimise the variance in the prediction by estimating the probability distribution of the variables at each timestep.

However, the ordinary KF assumes a linear process, which is not applicable to the attitude estimation. In particular, it is not capable of preserving the unit norm constraint on the rotation quaternion. This led to the development of the extended Kalman filter (EKF), which is a non-linear version of the KF. In this approach, the non-linear system is continuously linearised about an estimate of the current mean and covariance.

The EKF algorithm must be initialised with an estimate of the initial state and process noise. Then, at each step, the predicted state is updated by projecting forward the previous state and error. The prediction is then corrected by using the measurement from the current step. This requires the calculation of the *Kalman gain*, which is in effect a measure of the noise covariance in both the measurement and the projected state, computed from the estimated noise in the current measurement and the process. The estimated state is thus updated using the measurement and prediction, weighted by the Kalman gain. As long as sufficiently acuurate estimates of the measurement and process noise are provided, the result is an estimated state at each timestep which is more accurate than the original noisy measurements.

There have been further extensions to non-linear Kalman filtering. For example, Garcia et al. (2019) compare the performance of three different non-linear filters for spacecraft attitude determination, using real data. They find that the three filters - the EKF, Unscented KF and Cubature KF - have very competitive results, with advantages and disadvantages to each. In addition, although much of the work has focused on predicting the attitude of a single spacecraft using on-board sensors, the same approach can easily be applied to determining the relative attitude of an unknown target satellite.

5.2 Implementation of the Rotation Estimation Algorithm

Since the mathematics of the problem are well known, it makes sense to pass the output of the feature extraction stage through an algorithm such as those discussed in the previous section, rather than attempting to use a neural network approach for the entire task. Therefore, the quaternion-based RMSD approach is selected as the method of predicting the rotation. However, the fully neural network approaches are also investigated, with the results discussed in the following section. The RMSD algorithm is implemented within the machine learning framework, so that the entire model can still be trained end-to-end using the output of the motion estimation stage.

Given two successive input images separated by a small time-step, the siamese convolutional neural network extracts and matches image features. A depth reading must also be provided by the on-board sensors in order to assign a z value to each of the x, y feature locations, resulting in two sets of matched 3D points. In order to improve the accuracy of the estimate, extracted features with low confidence and those which appear to be outliers are rejected. The position of the remaining features are then passed to the attitude estimation stage, where the observed rotation of the target relative to the chaser is determined using the RMSD approach detailed in Section 5.1.1.1. In fact, this algorithm is sufficient to calculate the combined rotational and translational motion of the target. However, we consider the translational motion to be a less interesting problem which has been well covered previously and which will not provide as valuable an indication of the usefulness of our extracted feature points; therefore, we concentrate solely on the rotational motion and do not include any relative translational motion in our datasets.

We have shown in Chapter 2 that the observed instantaneous rotation, relative to both the chaser and target's rotating body frames, can be defined in terms of an arbitrarily selected reference frame at each step. Assuming that the motion of the chaser is known, the true instantaneous rotation of the target can be determined. By integrating these estimates over time, the attitude of the target at all times can be reconstructed.

5.2.1 Acquiring Depth Information

The feature extraction network provides only the location of the feature points in image coordinates. However, this is insufficient information to describe a 3D transformation. A third value, the depth, is therefore required for each point. This can be acquired from a depth sensor such as LiDAR.

The depth sensor will generate either a depth map or a sparse point cloud; in the latter case, a depth map can be generated from the point cloud using the method of Ku et al. (2018). By taking samples of the depth at the same frequency as the colour images, a depth map is associated with each input image. The depth map contains the distance to each pixel in the image.

Each feature point contains the location of that feature in image coordinates. These coordinates can be used to sample the input depth map, providing the 3D position of each feature. However, it is then necessary to convert the feature coordinates from the view frame to the world frame, since the feature coordinates are in the range $[-1, 1]$ while the depth is given in metres. For this, knowledge of the camera parameters is required, in particular the angular field of view.

5.2.1.1 Sampling the Depth Map

The simplest method to acquire the depth values is to convert the feature coordinates from the range $[-1, 1]$ in the view frame, (x_v, y_v), to an integer index corresponding to the closest pixel in image coordinates, by the relation

$$(x_{\text{img}}, y_{\text{img}}) = \lfloor ((x_v, y_v)/2 + 0.5) \times (\textit{image\_size} - 1) \rceil. \tag{5.6}$$

However, we discover that the process of converting to integer coordinates is not differentiable. This causes the gradient to be lost at this point, preventing the network from using the outputs for training.

A solution to this comes from the computer vision technique of bilinear interpolation. This is commonly used for upsampling a 2D image, by obtaining values at locations anywhere within the bounds, not just the original pixel locations. The closest four surrounding data points are used to interpolate a data point between these. This is equivalent to linear interpolation in the x-direction followed by the y-direction. Unlike linear interpolation, this technique is not linear but quadratic at the sample location; nonetheless, the result is a fully differentiable process for sampling a 2D grid.

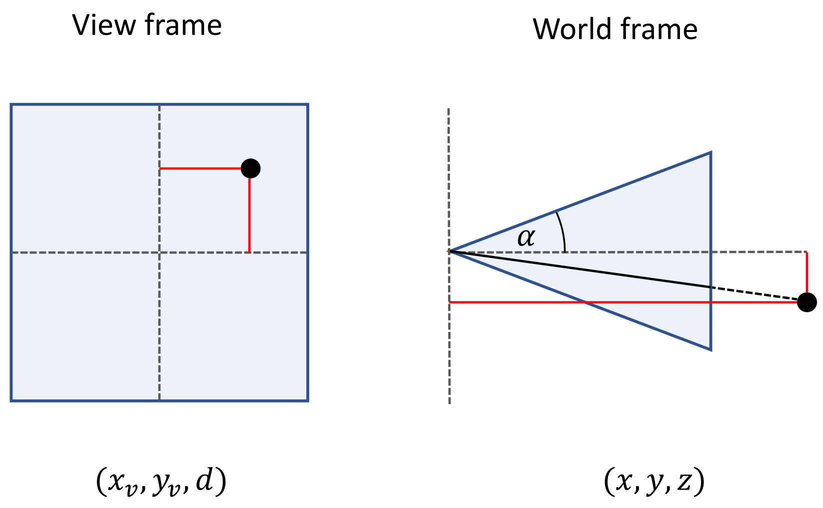


FIGURE 5.2: An illustration of the conversion of feature point coordinates from the camera view frame to the world frame.

5.2.1.2 Correcting Image Coordinates

Due to the camera perspective, the location of a feature point in the view frame does not directly correlate with its location in world coordinates. Figure 5.2 demonstrates the difference between the two frames. After the feature extraction stage, each feature point is given by its coordinates in the view frame, (x_v, y_v), in the range $[-1, 1]$. These are then given an associated depth value, d. Before these points can be used to predict the translation, the global (x, y, z) position must be found.

The camera frame can be imagined as the projection onto a view plane, where the distance to the view plane is governed by the angular field of view of the camera. This distance is referred to as the focal length, f, and can be calculated as

$$f = \frac{1}{\tan \alpha} \tag{5.7}$$

where α is equal to half the angular field of view. Note that the focal length is normally multiplied by the image size in pixels; however, since the pixel coordinates are already normalised, we also normalise the focal length by the same factor.

The coordinates (x_v, y_v) are points on the view plane which are in line with the object in the world coordinates. Given the focal length and the z-value of the point, the world coordinates can be calculated by the law of similar triangles:

$$\frac{x}{z} = \frac{x_v}{f}, \quad \frac{y}{z} = \frac{y_v}{f} \tag{5.8}$$

However, the depth values computed by a depth sensor are often given as the distance travelled by the ray, which is different to the z coordinate. This difference is computed using the cosine of the angle between the vectors $[0, 0, f]$ and $[x_v, y_v, f]$; this is equivalent to the angle of the right-angled triangle with length z and hypotenuse d.

Therefore, the conversion from view coordinates to world coordinates, correcting for the perspective of the camera, can be determined simply using trigonometric relations. Assuming an aspect ratio of 1 : 1, and observing towards the negative z direction, the world coordinates can thus be calculated as:

$$x = -x_v z \tan\alpha, \tag{5.9a}$$

$$y = -y_v z \tan\alpha, \tag{5.9b}$$

$$z = -\frac{d}{\tan\alpha\sqrt{x_v^2 + y_v^2 + (1/\tan\alpha)^2}}. \tag{5.9c}$$

In order to speed up the loading of the depth images, a grid is constructed at initialisation which contains the value of $\left(\tan\alpha\sqrt{x_v^2 + y_v^2 + (1/\tan\alpha)^2}\right)$ at each pixel location. The depth map is simply divided by this grid at each iteration. In addition, the value $\tan\alpha$ is calculated once to avoid time-consuming trigonometric computations.

5.2.1.3 Problems with Depth Readings

The analysis from the previous chapter highlighted a potential issue with the depth measurements using this approach. The position of a feature point such as a corner would often be slightly further out than the observed feature in the original image, due to distortions and blurring caused by the convolutional filters. While this did not appear to a significant problem at that stage, because the motion of the features still followed the motion of the target, any small inconsistency in the feature point coordinates has a large impact on the accuracy of the depth reading, since this is taken from the coordinates of the depth map. In particular, if the feature point falls outside the boundaries of the object, there will be no associated depth value at this point.

Two approaches are investigated to counteract the effects of this problem. First, a learnable scaling parameter is included in the model to scale the feature locations towards the centre. However, this will only improve the cases where the target is centred at the origin of the image, or else the scaling should be performed relative to the observed centre of the target. In addition, the magnitude of the distortion effect is different for each filter, so a single scaling parameter is insufficient to capture this for each case.

In cases where it is not possible to associate a depth value to a feature point, due to it falling outside the target, we instead look at nearby pixels to obtain the measurement. The model observes the closest few pixels in each direction and takes the average of all of those for which a depth value exists. If this is not the case for any of the surrounding pixels, then the depth is taken to be 0. This approach ensures that each feature point is more likely to be able to be matched to the world coordinates, though the associated depth values will not be fully accurate. Additionally, this makes no difference to the feature points which are in the wrong location but do not fall outside the boundaries of the target.

The optimal solution to this problem would likely involve using the depth as well as the RGB image when extracting the features. Then, the extracted feature points can also contain the depth associated with the actual location of the features in the original image, rather than the coordinates after the extraction. However, further research is required to determine the best method for achieving this.

5.2.1.4 Depth Prediction

We wish to for the model to integrate with any type of visual sensor, including monocular cameras. However, monocular cameras do not provide a measure of the depth, yet this is required in order to predict the transformation using the approaches detailed previously. The depth must therefore be predicted by the network at each feature location.

The depth is predicted from the RGB image using a fully-convolutional network, which is an approach commonly used in semantic segmentation to associate each pixel with a label (Long et al., 2015). However, instead of a class label, we wish the network to learn a depth value. This network is trained separately using the actual depth maps as the true output, before integrating it into the full model and finetuning. The main risk with this method, though, is that it will reduce the generalisation capabilities of the network, since the network will have to learn the approximate dimensions of the specific objects in the training data in order to estimate the depth.

5.2.2 Outlier Rejection

Since the landmark matching method used is simplistic in nature, the matches may not always be accurate. For example, one landmark may show the top-right corner of the target; after a rotation, the top-right corner may be a different part of the target which was previously unseen. Therefore, we would like to ignore landmarks which are not useful or are matched poorly.

We achieve this, in part, by assigning a value for the confidence of each pair of detected landmarks. The confidence value is simply a measure of the likelihood that a certain feature corresponds with a strong object landmark. This confidence measure is calculated in a similar way to the loss term L_{conc}, i.e. by applying a Gaussian mask, centred at the landmark location, across the output feature map. This value is used to reject those features with the lowest confidence.

We also implement an outlier rejection algorithm within the end-to-end learnt network. A common choice for outlier rejection is the random sample consensus method, RANSAC (Fischler and Bolles, 1981). However, this is an iterative algorithm, which makes it unsuitable for use within deep learning networks in its current state. The RANSAC algorithm is therefore adapted to make it parallel and fully differentiable. Figure 5.3 illustrates the adapted RANSAC algorithm for outlier rejection within a neural network. The steps of the algorithm are as follows:

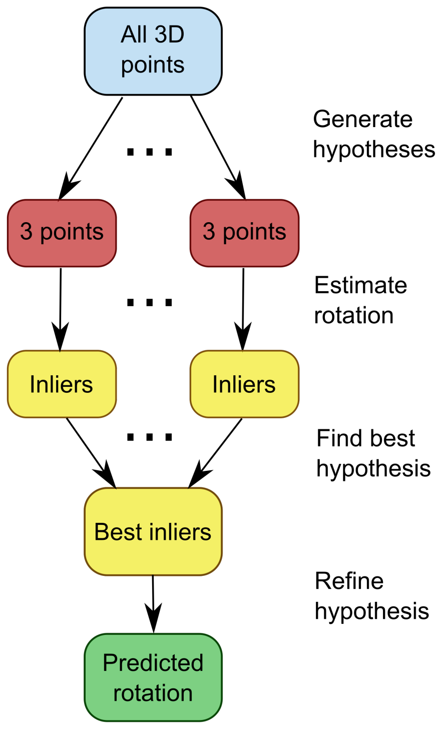


FIGURE 5.3: The adapted RANSAC algorithm.

1. **Generate hypotheses.** Randomly sample N sets of three feature point pairs from the sets of all features. These form the set of hypotheses H. Each hypothesis H_n contains a set of three points $(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3)$ randomly sampled from $\{\mathbf{p}_k\}$. Three points are the minimum required to predict a rotation.
2. **Score hypotheses.** For each hypothesis H_n, estimate the rotation using only the sample of three features. Apply this rotation to all $\{\mathbf{p}_k\}$ to obtain $\{\mathbf{p}_k^r\}$. Each feature pair $(\mathbf{p}_k, \mathbf{p}_k')$ for which $|\mathbf{p}_k' - \mathbf{p}_k^r| / |\mathbf{p}_k| < \delta$ is an inlier, while all other pairs are rejected as outliers.
3. **Rank hypotheses.** Order hypotheses based on the number of inliers, and select the best one.

4. **Refine hypothesis.** Using all inliers from the best hypothesis, but ignoring outliers, predict the rotation.

Being differentiable and parallelised, this implementation of the RANSAC algorithm may be contained within the end-to-end trainable neural network. This enables the parameters of the algorithm itself, such as the cut-off value δ, to be trained alongside the rest of the network.

5.2.3 Loss Terms

The entire network is trained together to increase the accuracy of the predictions. Since only the feature extraction stage has trainable parameters, the training will focus on encouraging the extraction of more "useful" image features. These are simply the feature points which result in the best performance when used to predict the rotation. As such, the loss function used in the training should be a measure of the accuracy of the estimated rotation quaternion, compared with the true value from the data labels.

In this case, to calculate the loss function we use the geodesic distance between the true and predicted rotation quaternions, which is commonly used in orientation estimation tasks (Mahendran et al., 2017; Salehi et al., 2018). The geodesic distance refers to the angle of shortest distance between the two quaternions in 4D space. The quaternion loss can be simply defined as

$$L_{\text{quat}} = 1 - \langle \mathbf{q}, \hat{\mathbf{q}} \rangle^2 \tag{5.10}$$

where $\hat{\mathbf{q}}$ is the predicted rotation quaternion and the operation $\langle \mathbf{q}, \hat{\mathbf{q}} \rangle$ denotes the quaternion inner product. This loss term L_{quat} is equivalent to $(1 - \cos\theta)/2$, where θ is the geodesic distance between quaternions, but is preferable over using θ itself since it avoids trigonometric operations and results in a loss in the range $[0, 1]$ (Huynh, 2009). In addition, using the geodesic distance avoids the problems with ambiguity in quaternions, i.e. that the quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation.

We also investigate other alternatives for the rotation loss term. Using the above definition, there is a risk that the network will put more emphasis on minimising the loss in cases where the rotation angle is larger. In these situations, the geodesic loss will almost always be higher than cases with small rotations, since it is simply a measure of the distance between prediction and truth, so these will have a bigger impact on the loss. As an attempt to counteract this, we can use as the loss term the geodesic distance scaled by the true rotation angle:

$$L_{\text{scaled}} = \frac{\arccos\left(2\langle \mathbf{q}, \hat{\mathbf{q}} \rangle^2 - 1\right)}{2 \arccos q_4}. \tag{5.11}$$

This has the downside of requiring trigonometric operations which are slow to compute, but will give a more fair calculation of the loss in different situations. This problem does not exist in most orientation estimation approaches, in which the actual orientation is regressed rather than a rotation over a timestep.

If a neural network approach is used to predict the quaternion, it is necessary to restrict the output to be a rotation quaternion. This means that the quaternion must have a norm of 1. This can be achieved by using a separate loss term L_{norm}, which penalises predictions without unit norm. For example,

$$L_{\text{norm}} = \exp\left(||\hat{\mathbf{q}}| - 1|\right). \tag{5.12}$$

However, there is another solution to this problem. The quaternion notation is in fact over-defined: the first three elements alone are sufficient to describe the rotation. Therefore, we can use only a three outputs to predict the rotation, (q_1, q_2, q_3). The final element can be obtained from these by $q_4 = \sqrt{1 - q_1^2 - q_2^2 - q_3^2}$, before passing it to the loss function. This will always be a rotation quaternion with unit norm, so there is no need for the L_{norm} loss term.

Finally, we construct a separate loss term using the output of the outlier rejection stage. We find that the accuracy of a prediction is heavily dependent on the number of feature points which are rejected as outliers. Therefore, we propose that encouraging the network to reduce the number of rejected points using the loss function will result in better extracted features and improved estimates overall. The inlier loss term is thus given as

$$L_{\text{inl}} = \exp\left(-\frac{K_{\text{inl}}}{\sigma_{\text{inl}}}\right), \tag{5.13}$$

where K_{inl} refers to the number of features which are retained as inliers after the outlier rejection and σ_{inl} is a weighting term.

In orientation estimation problems, the task is often considered as a classification instead of a regression. A number of different orientations are collected into "bins", and the network attempts to associate the image with one of these. In this case the mean square error (MSE) can be used as the loss function. However, we do not consider this approach to be as applicable to the task that this work focuses on, since we wish to accurately compute a rotation rather than an orientation, so this approach is not investigated here.

5.2.4 The Rotation Estimation Algorithm

The estimate of the rotation is computed using the quaternion-based RMSD approach outlined in Section 5.1.1.1. Since the approach must be contained within the end-to-end trainable model, it must be implemented within the machine learning framework; in our case this is PyTorch. Therefore, the method must be differentiable, with a fully defined gradient at all points.

The key difficulty with this implementation lies in the eigenvalue decomposition. In the case that eigenvalues are repeated, or very close to each other, the gradient is not defined. This causes an error in the training process because an undefined gradient means that the output is impossible to calculate. Repeated eigenvalues are most likely to occur in the specific situation where, in stage 2 of the RANSAC process, three similar feature points - by which we mean features which are very close to each other - are selected and used to estimate the rotation. In this case we would not expect to obtain a useful estimate even in the absence of errors, so it is not an issue if the solution to this problem introduces discrepancies in these situations.

The most obvious solution would be to check for similar eigenvalues beforehand and correct the matrix accordingly. This still requires eigenvalue decomposition of the original matrix though, therefore requiring the gradient to be defined, so it does not solve the problem. However, we find that the computation of the singular values of a matrix has a well-defined gradient at all times. Since the matrix F is a real symmetric matrix, the singular values are equivalent to the eigenvalues. Thus, in order to prevent errors in the gradient, the singular values of the matrix F are investigated at each iteration. If any fall within a small distance from each other, then a random deviation is added to the diagonal of the matrix so that the eigenvalues become distinct while retaining the symmetricity of the matrix.

5.2.5 Using Previous Measurements

The rotational dynamics of a tumbling satellite in orbit are unlikely to change significantly over the relatively short timescales that we are interested in. Therefore, using only the information from a single step to calculate the instantaneous rotation is not optimal. Instead, past measurements should be used to refine the current estimate.

Conventionally, a Kalman filter would often be used in this case; such filters have been used in attitude estimation systems for many years (Lefferts et al., 1982). However, a difficulty with Kalman filtering is the requirement for an estimate of the measurement and process noise. In machine learning models, these noise values are very difficult to accurately estimate, so it is difficult to correctly initialise the Kalman filter.

Instead, we pass the estimate from the previous step directly to the machine learning network. Internally, the network uses this estimate, as shown in Figure 5.4, following the feature extraction and outlier rejection stages. For each individual pair of features $(\mathbf{p}_k, \mathbf{p}_k^{'})$, a four-element strength tensor is constructed, combining the predicted accuracy of the previous measurement with the confidence in the current feature. This tensor is passed to a single fully-connected layer of 16 nodes, which uses the provided information to estimate a single weighting value. This weight w_k is used to scale the position of the feature from $\mathbf{p}_k{}'$ towards $\hat{\mathbf{p}}_k'$. This latter value is calculated as, in quaternion form, $\mathbf{q}_{t-1}^* \otimes \mathbf{q}_k \otimes \mathbf{q}_{t-1}$. Again, the weights of the hidden layer are identical across each feature pair.

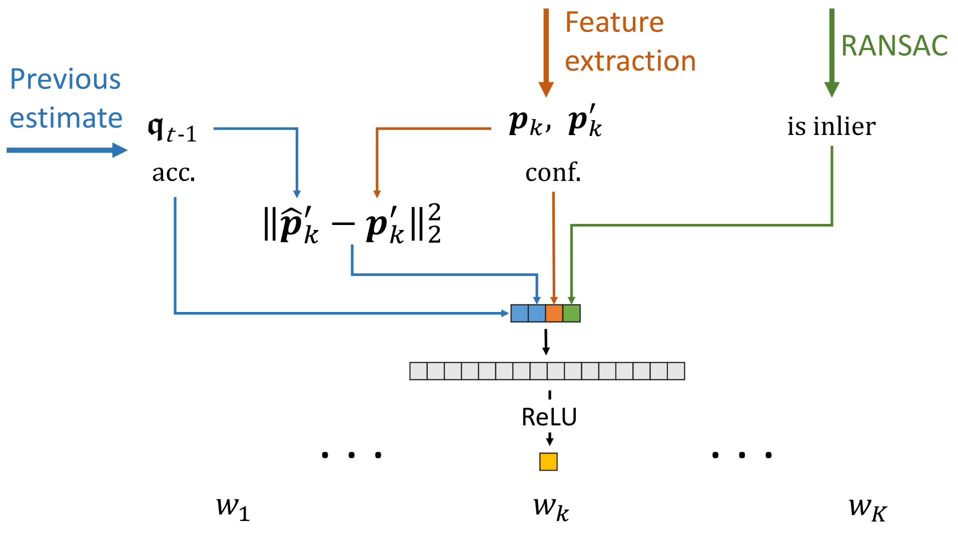

Feature extraction
RANSAC
Previous estimate
$\mathbf{q}_{t\text{-}1}$
$\boldsymbol{p}_k,\ \boldsymbol{p}_k'$
is inlier
acc.
conf.
$\|\hat{\boldsymbol{p}}_k' - \boldsymbol{p}_k'\|_2^2$
ReLU
w_1
w_k
w_K

FIGURE 5.4: Using the previous estimate to improve upon the extracted feature locations.

The strength tensor is constructed from from the following four scalar values:

- The estimated accuracy of the previous measurement. The accuracy of the output of step $t-1$ is predicted by to $1 - L_{\text{rot}}(\mathbf{q}_{t-1}, \mathbf{q}_{t-2})$. In other words, this is a measurement of the similarity between the current and previous rotation measurements. If no previous measurement exists, for example on the first iteration or when tracking has been lost, the accuracy is zero; the network can learn to ignore the previous measurement entirely in this case.
- The distance between the observed features in the second image, $\mathbf{p}_k'$, and the observed features from the first image rotated by the past measurement of the quaternion, $\hat{\mathbf{p}}_k'$. The distance is measured as the square of the L2 norm between the two point vectors.
- The confidence of the extracted features, taken from the intensity of the softmax layer. This is the same value as used to calculate the loss term L_{conc}.
- A value of 1 or 0 corresponding to whether the specific feature pair was kept as an inlier or rejected as an outlier during the outlier rejection stage.

During training, a value for the previous measurement must be provided; however, since the inputs are randomly selected from the dataset at each iteration, this value would not be known. We investigate two different methods to overcome this problem.

First, we generate an estimate of the output of the previous step by adding noise to the true (expected) output of the current step. The downside of this approach is that the computed noise must closely match the actual measurement noise, in order for the network to learn a useful response. The second approach deals with the problem by using a series of images from successive steps at each training iteration. This allows for the network to pass the estimate from one iteration to the next and gradually improve upon it during training, which is more similar to the desired operation in practice. However, this approach significantly increases the amount of memory required for training, and so the previous estimate must be released every few iterations to clear the memory. Both approaches are investigated in the performance analysis.

5.2.6 Reconstruction of Attitude from Rotations

The observed rotation of the target, as seen by the chaser spacecraft, is a combination of the motions of both target and chaser. The observed rotation $\hat{\mathbf{q}}$, by observation and using the quaternion composition property, can be defined as

$$\mathbf{q}_{42} = \hat{\mathbf{q}} \otimes \mathbf{q}_{31}. \tag{5.14}$$

where the quaternion $\mathbf{q}_{42}$ refers to the rotation from frame $\mathcal{F}_2$ to $\mathcal{F}_4$ and similarly, $\mathbf{q}_{31}$ is the rotation from $\mathcal{F}_1$ to $\mathcal{F}_3$. In other words, $\hat{\mathbf{q}}$ defines the rotation from $\mathcal{F}_3$ as observed by a chaser in $\mathcal{F}_1$, to $\mathcal{F}_4$ as observed from $\mathcal{F}_2$.

Simply using quaternion composition relations, we find that the actual rotation of the target, $\mathbf{q}_{43}$, can be written in terms of the observed rotation as

$$\mathbf{q}_{43} = \hat{\mathbf{q}} \otimes \mathbf{q}_{31} \otimes \mathbf{q}_{21} \otimes \mathbf{q}_{31}^{*}. \tag{5.15}$$

We assume that the state of the chaser is known at all times due to on-board sensors. In this case, the only unknown quantity is $\mathbf{q}_{31}$. However, there is no constraint on the definition of the initial target body frame $\mathcal{F}_3$; as such, we are free to select any arbitrary frame for this, and the rotation will be defined in relation to this. At the first iteration, it is convenient to select a target body frame which is aligned with the chaser. At all subsequent iterations, the initial body frame $\mathcal{F}_3$ can be found by $\mathbf{q}_{43}$ at the previous step; this way, the attitude of the target at each step is reconstructed.

5.2.6.1 Combining Observations over Different Timesteps

In the previous chapter, we noted that the prediction could be improved by using information taken from several different timesteps. For example, a small timestep will show only a small rotation, in which case the feature matching is more reliable but the

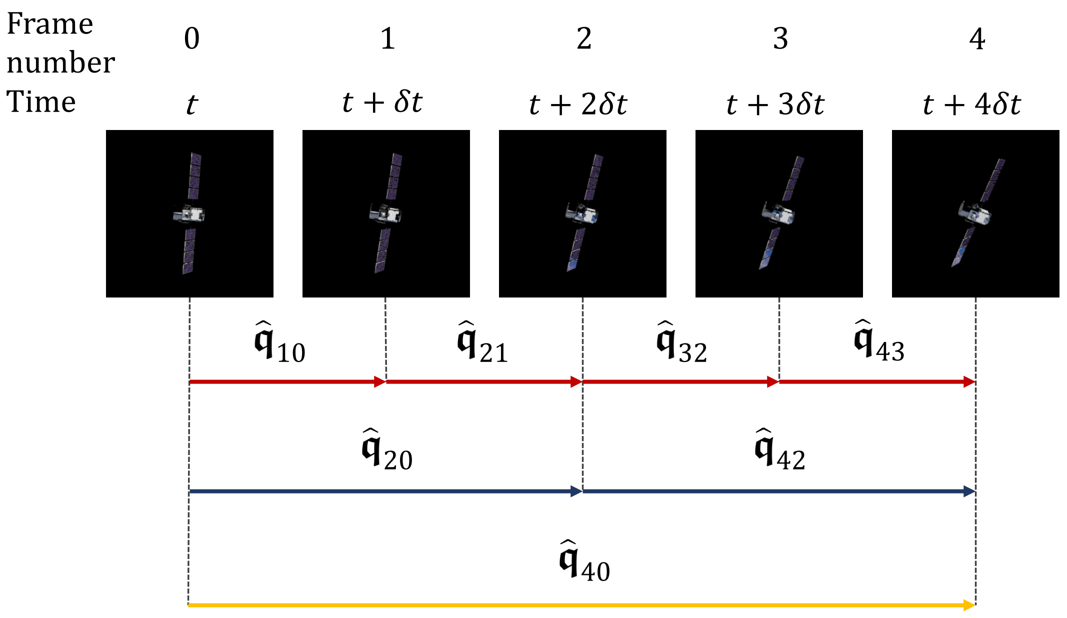

Frame number
Time
0
1
2
3
4
t
$t+\delta t$
$t+2\delta t$
$t+3\delta t$
$t+4\delta t$
$\hat{\mathfrak{q}}_{10}$
$\hat{\mathfrak{q}}_{21}$
$\hat{\mathfrak{q}}_{32}$
$\hat{\mathfrak{q}}_{43}$
$\hat{\mathfrak{q}}_{20}$
$\hat{\mathfrak{q}}_{42}$
$\hat{\mathfrak{q}}_{40}$

FIGURE 5.5: Combining the predicted rotation quaternions from different step sizes.

rotation is more ambiguous since the motion contained in the images is small. On the other hand, at larger timesteps there is more information in the data but the change in feature position is more pronounced and there are more likely to be occlusions and disappearing edges, so the feature matching is more challenging.

Therefore, we combine predictions taken over several different timesteps in the final estimation of the rotation. The method is illustrated in Figure 5.5. Every 4 steps, an estimate of the quaternion from $\mathcal{F}_0$ to $\mathcal{F}_4$ is computed over 3 different step sizes.

First, four predictions are made for every image pair with steps separated by δt, $\hat{\mathfrak{q}}_{10}$, $\hat{\mathfrak{q}}_{21}$, $\hat{\mathfrak{q}}_{32}$ and $\hat{\mathfrak{q}}_{43}$. The full quaternion over the four steps can be computed from the quaternion composition: ${}_1\hat{\mathfrak{q}}_{40} = \hat{\mathfrak{q}}_{43} \otimes \hat{\mathfrak{q}}_{32} \otimes \hat{\mathfrak{q}}_{21} \otimes \hat{\mathfrak{q}}_{10}$. The same can be done for steps separated by $2\delta t$, resulting in the second prediction, ${}_2\hat{\mathfrak{q}}_{40} = \hat{\mathfrak{q}}_{42} \otimes \hat{\mathfrak{q}}_{20}$. Finally, ${}_3\hat{\mathfrak{q}}_{40}$ is taken simply from the estimation of $\hat{\mathfrak{q}}_{40}$ over the full timestep, $4\delta t$.

The three estimates of the quaternion must then be averaged together to give the final prediction. Markley et al. (2007) present a method for computing the weighted average of quaternions in a computationally efficient manner. This involves constructing the 4×4 matrix

$$\mathrm{M} = \sum_{i=1}^{n} {}_i\hat{\mathfrak{q}}\, {}_i\hat{\mathfrak{q}}^T w_i \tag{5.16}$$

where w_i is the weight of the quaternion at index i, ${}_i\hat{\mathfrak{q}}$. The average quaternion is taken as the eigenvector of M corresponding to the maximum eigenvalue. The weight terms allow for different step sizes to be given a different importance in the prediction.

5.3 Results and Discussion

Following the pretraining of the feature extractor, detailed in the previous chapter, the full model is trained against the loss in the final prediction of the rotation. Again, the performance of the approach is analysed under several different conditions, in this case focusing on the rotation estimation section of the network rather than the feature extraction. The effectiveness of the different attempts at improving the accuracy are also investigated.

We note that training the full network against the final prediction is a difficult task for the model. In fact, the loss during the finetuning does not change significantly over successive training iterations. This means that the model is not improved from the state after pre-training. There are a number of possible reasons for this. The relationship between the weights of the convolution layers and the output is very unintuitive, and so may be too complicated for the model to learn. In addition, the two-convolution mode which we use in the analysis (due to the reasons addressed in the previous chapter) may not have sufficient parameters or network depth to learn the complex, non-linear relationships. Alternatively, it may be the case that simply further investigation of loss weights, a larger dataset or longer training times are required.

5.3.1 Estimation Accuracy

We begin the analysis with a comparison of our full model with three different conventional approaches. Again we use the two conventional feature extraction methods as comparisons, but this time we also include the iterative closest point (ICP) algorithm. As described in Section 5.1.1.2, this method uses the full point cloud and iteratively improves upon an estimate of the rotation. The comparison is shown in Figure 5.6.

Along with the improvements from the conventional feature extractors noted in the previous chapter, we also note that our method achieves similar or better performance than the ICP algorithm for all sizes of rotation angle. This is a promising result, as the ICP algorithm has already been used in real space missions. In addition, due to the iterative nature of the algorithm, our approach is significantly faster; in our implementations, we found a difference in computation time of approximately 10 times, although it should be noted that there exist methods of speeding up the ICP algorithm which are not used in this implementation.

We have also discussed the possibility of our network to integrate with several different visual sensors, since this is another potential advantage of the machine learning approach. In theory, the method can learn to extract features from any type of

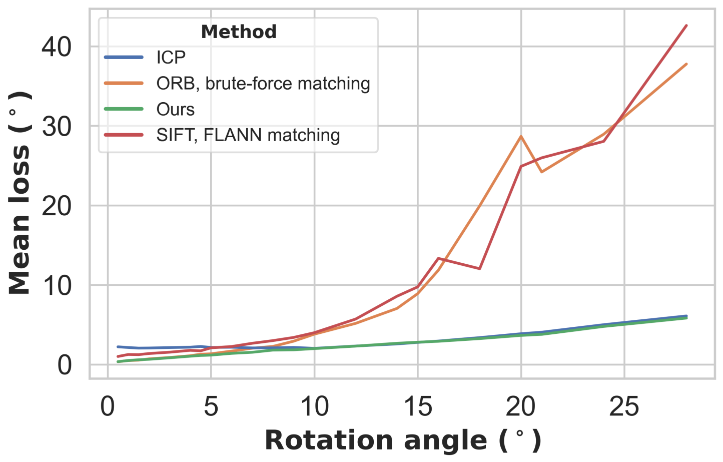


FIGURE 5.6: A comparison in performance between our feature extraction approach and three conventional computer vision approaches for estimating the rotation.

input data. However, it is required that the extracted feature points have an associated depth value in order to acquire the 3D locations. If this is not provided then it can be learned by the network, but this is a difficult task with only the information in a colour image, especially if the target satellites vary significantly in size.

Figure 5.7 illustrates a quick investigation into using several different visual sensors, each of which has been simulated in the simulation framework. These are: the RGB-D camera which has been used in the analysis to date; a LiDAR sensor with a lower angular resolution than the RGB-D sensor, taken by sampling every 4 pixels in each direction; an RGB camera and LiDAR sensor, where the first is used to extract features and the second only for depth values; a merged RGB and LiDAR sensor, where the features are extracted from a full 6-channel (RGBXYZ) image; and finally, a monocular RGB camera where the depth is predicted by the network from the RGB image.

It is clear from this comparison that the RGB image alone is not sufficient to predict an accurate estimate of the depth. This is not surprising since there is not enough information in the image if no assumptions can be made about the size of or distance to the target. We also see that the LiDAR sensor is less useful than the RGB-D camera, though again this is unsurprising due to the lower angular resolution used. However, we do see that combining the LiDAR and RGB sensor outputs in the feature extraction stage improves the performance somewhat. This is an interesting outcome and could be a useful point of further research.

Earlier in this chapter, several different approaches were proposed for estimating the rotation, besides the RMSD method has been implemented in the model. These included several fully machine learning models, such as the siamese ResNet implementation and the global-local network, as well as simply using neural networks to predict the rotation given the locations of the extracted feature points. However, in

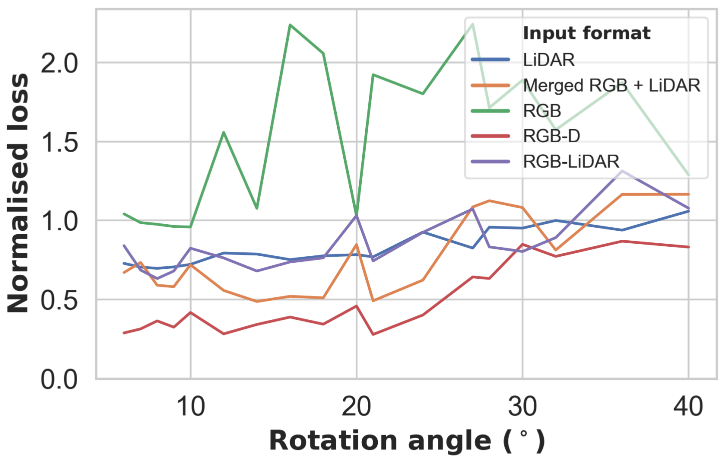


FIGURE 5.7: Comparison between different input data types, from different visual sensors.

all cases we find that these models are incapable of learning the correct response, regardless of the parameters of the training. This could be due to the very complex relationship between the inputs and the output quaternion. However, it means that there is no useful comparison to be made between the approaches.

5.3.2 Outlier Rejection

A key element of the estimation process is the outlier rejection stage, using our adapted RANSAC algorithm. The goal of the algorithm is to improve the accuracy by rejecting feature points which appear to be outliers, either due to poor feature extraction or incorrect matching. Figure 5.8 shows that this algorithm significantly improves the average prediction accuracy for every satellite target in the datasets, when compared with an approach which uses all feature points to make the prediction. This shows that the adapted RANSAC algorithm is a useful outlier rejection scheme for use within machine learning models.

However, it can be seen that, as the rotation angle over the step increases, the accuracy of the approach with RANSAC decreases. As this does not occur in the basic approach, it can be determined that the drop in accuracy is related to the outlier rejection algorithm. This observed effect is investigated further in Figure 5.9. In this figure, the loss angle is plotted against the number of inliers remaining after the RANSAC algorithm at each step. We show the results for four different rotation angles.

Figure 5.9 illustrates the two main sources of error in the model. It is clear from this figure that at very low rotation angles, any small error in the position of features has a large impact on the result so the predictions have a higher loss overall. As we increase the angle, the relative accuracy in the majority of cases increases, but this is

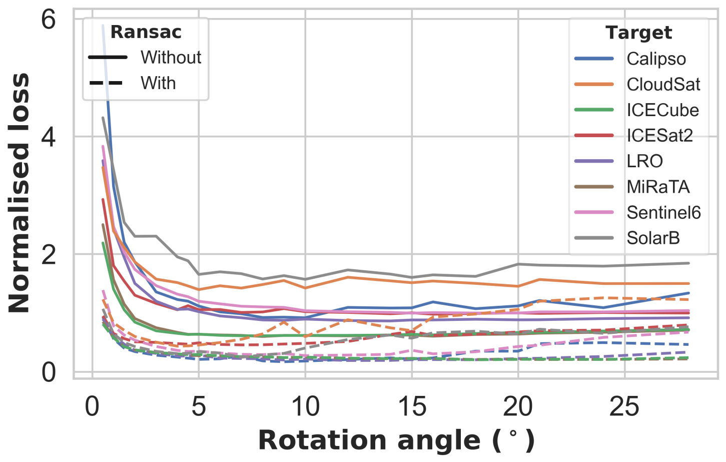
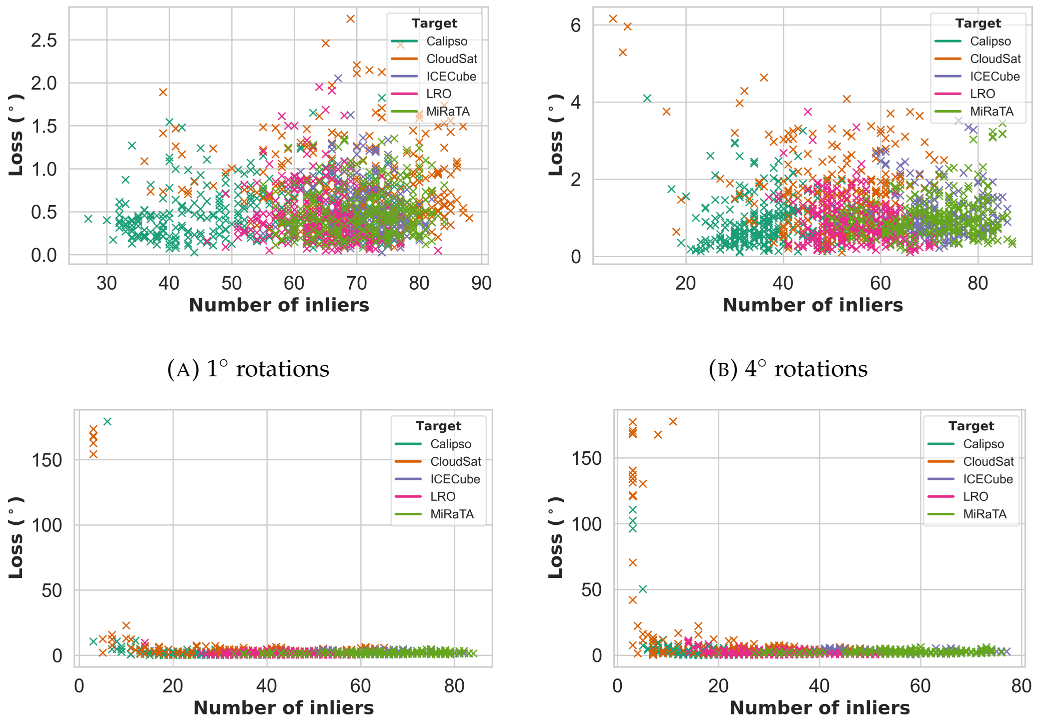


FIGURE 5.8: The loss in the final prediction normalised by the rotation angle, for several different satellite targets, with and without outlier rejection.



(A) 1° rotations

(B) 4° rotations

(C) 8° rotations

(D) 12° rotations

FIGURE 5.9: Step-wise loss against number of inliers. A larger rotation angle results in improved accuracy in the general case but leads to higher likelihoods of erroneous feature matching.

counteracted by an increased chance of a failed prediction, which can introduce errors even up to 180° in the worst case. These failures clearly have a very large impact on the mean loss over a full dataset, potentially resulting in higher losses than is realistic in the previous analysis.

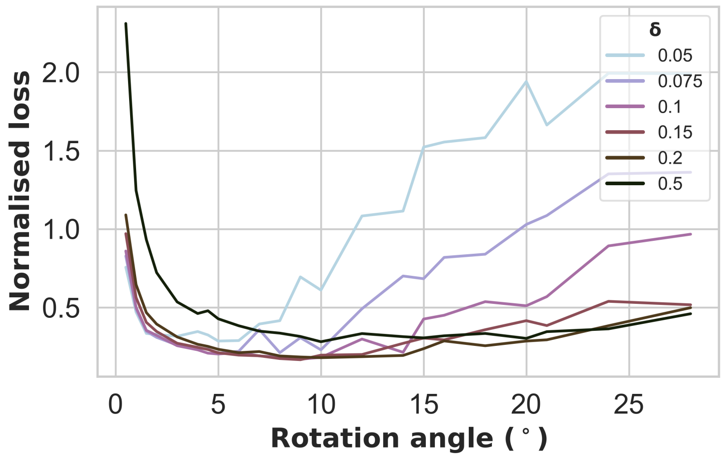


FIGURE 5.10: The impact of increasing the cutoff parameter in the RANSAC algorithm on the mean loss over the test datasets, for different rotation angles.

The combination of these two effects depends on the satellite target. For the more detailed targets, such as Calipso, CloudSat and LRO, there is more information in the images to work with so the accuracy in the best case can be very high. However, this also means there is a higher likelihood of self-occlusions, due to large solar panels for example, which can introduce errors. In contrast, the simpler, CubeSat-like satellites (ICECube and MiRaTa) contain fewer details but are less prone to these issues, so are less likely to result in failed predictions. In the figure, these targets can be seen to have a slightly higher loss in general but fewer steps with very high losses.

The other important result to note here is that the failed predictions occur when the model fails to find a sufficient number of inliers to correctly define the rotation. This suggests that it is possible to predict a prediction failure by looking at the number of inliers, and if this falls below a certain threshold, rejecting that step preemptively.

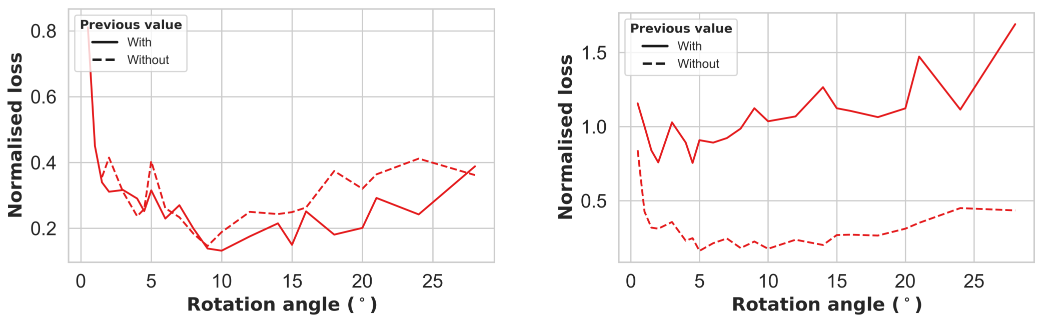
This result also suggests that the parameters of the RANSAC algorithm could perhaps be adjusted to reduce the likelihood of these failed predictions. The cutoff parameter, δ, is user-defined and has a large impact on the performance of the algorithm. A low value of δ will cause too many feature pairs to be rejected as outliers, whereas if it is too high then outliers will not be rejected when they should be. Figure 5.10 details an experiment to determine the optimal value of this cutoff parameter, using Calipso as the target satellite in each case.

The optimal value of the cutoff is clearly dependent on the rotation of the target. This makes sense logically, as the value which determines whether a point is an inlier is a measure of the distance between the observed and expected positions after a rotation. If the rotation is larger, then these distances will also be larger. Clearly, a better measure could be obtained if this value could be scaled by the rotation angle. However, it is not possible to predict this angle before the outlier rejection stage.

This figure illustrates the trade-off to take into account when selecting the value of the cutoff. If it is too small, we increase the likelihood of the failed predictions shown in the previous analysis. On the other hand, we see that increasing δ higher begins to lead to an increased loss overall, as too many feature pairs are retained when they should be rejected. We select a value of 0.2 as it provides the best balance between the two effects.

5.3.3 Improving the Predictions

We have discussed a number of other methods for improving the accuracy of the predictions over time. First, we aim to improve the per-step prediction accuracy by incorporating information from more than the single image pair acquired at each iteration. As previously mentioned, two different training methods are used to incorporate the output of the previous iteration to improve the current estimate. In the first, a previous measurement is generated by adding noise to the true output. In the other, the actual previous measurement is used but must be reset every few iterations due to the high memory consumption. The results of both methods are provided in Figure 5.11.



(A) Generated previous measurement (B) Actual previous measurement

FIGURE 5.11: Loss of the model with and without using the output of the previous iteration, using the two different training methods. The Calipso satellite is used as the target in this experiment.

In Figure 5.11a, we note a slight improvement in performance when incorporating the output from previous iterations. This is a promising result, although further improvements will be necessary before this approach is applicable to real situations; it is likely that adjustment of the noise parameters used in generating the previous measurement would lead to improvements in this case. On the other hand, the method demonstrated in Figure 5.11b does not appear to successfully train the model, resulting in a significant loss of accuracy. Again, this approach will require further investigation.

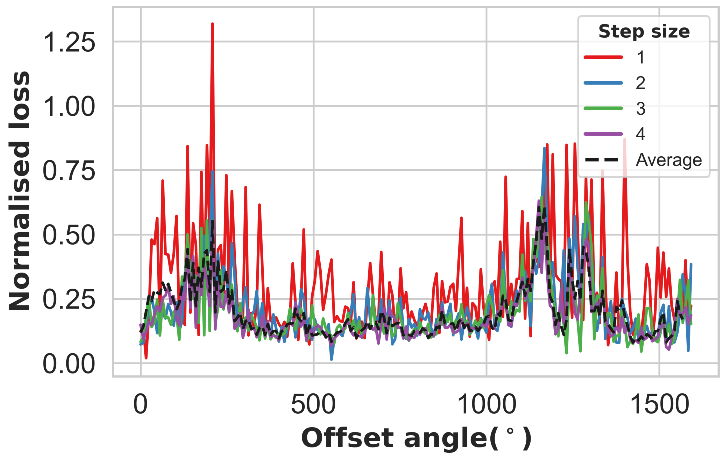
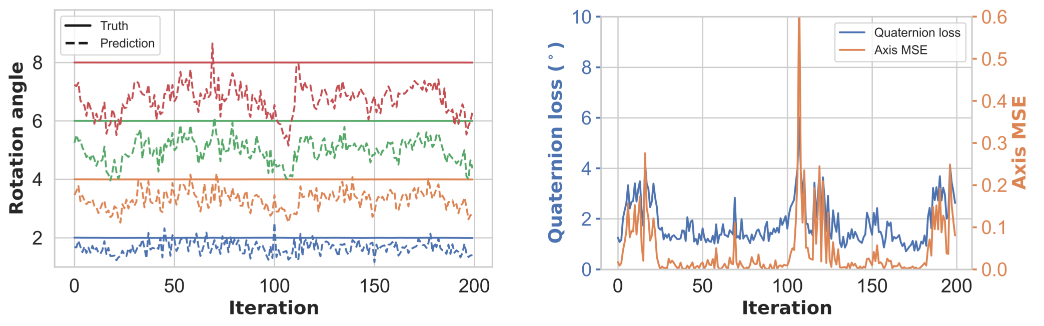


FIGURE 5.12: An illustration of the normalised loss in the prediction at each step, looking at different step sizes between images, where the rotation of the target over a single step is 2°.

Another approach which we investigate to improve the accuracy of the output involves making use of predictions over different step sizes. This means that the model can balance the two main sources of error in the approach, which are caused by too-small or too-large rotations respectively. By taking an average of three predicted quaternions at different step sizes, the result should not be impacted as heavily by either of the sources of error.

Figure 5.12 demonstrates the difference in loss in the predictions for different step sizes. The rotation of the target at each step in the dataset is 2°, so a step size of 4 would correspond to an 8° rotation, and so on. In this graph, we are looking at the error at each iteration in the data, where the x axis is simply the offset of the target from its initial state. It is important to note that the loss on the y axis is simply the loss in the prediction at that step, rather than the cumulative loss over the dataset. The loss angle is normalised by the rotation angle multiplied by the step size, allowing for a valid comparison between them.

The averaged quaternion can definitely be seen to avoid the large spikes in the loss in the individual predictions. It may be possible to improve this even further, however. As mentioned previously, the accuracy of a prediction can be predicted internally by looking at the number of inliers kept at each step. This value can be used to provide a weighting for each quaternion in the averaging algorithm, as described in Section 5.2.6. Therefore, predictions which are less likely to be accurate will make a smaller contribution to the final quaternion.



(A) Discrepancies in the predicted rotation angle for different step sizes.

(B) Loss in the quaternion compared to the error in the predicted rotation axis.

FIGURE 5.13: An investigation into the error within the predicted quaternion.

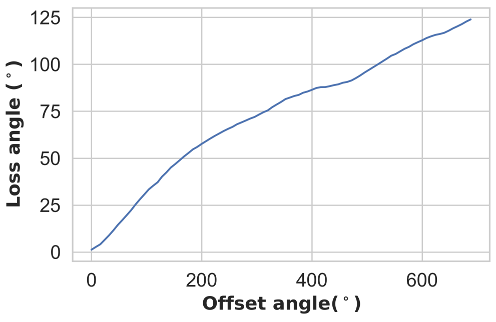
5.3.3.1 Remaining Error in the Predictions

After making all the additions detailed in this chapter, we have significantly improved the output of the model. However, we see that even in the best cases, the error in the prediction does not drop below approximately 20% of the target's rotation angle. This is a somewhat puzzling outcome and encourages further analysis of the errors themselves. We therefore investigate the error in the prediction in Figure 5.13.

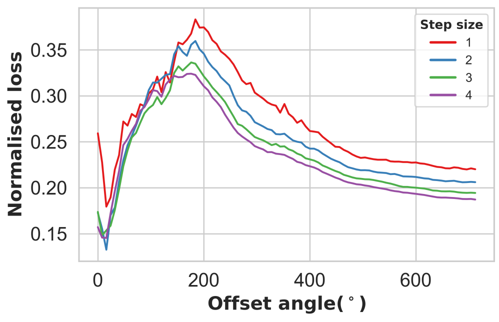
The first of these figures looks at the difference between the angular part of the predicted quaternion and the true value. In each case, the predicted rotation angle is consistently $15 - 20\%$ *lower* than the truth. This coincides closely with the amount of error in the best-case predictions shown previously. It can thus be implied that there is some systematic error in the prediction which causes the output to be biased towards smaller rotations.

This is backed up by the comparison in Figure 5.13b, which shows both the full loss in the quaternion as well as the mean-square error (MSE) of only the predicted rotation axis. Clearly, different approaches are used to calculate the errors of the two values and the scale is not the same. Nonetheless, the comparison suggests that the loss in the majority of cases is overwhelmingly impacted by this error in the predicted rotation angle.

There are a number of potential causes of the systematic error. It could be due to inaccuracies in the data itself – for example, the depth maps returned by the Blender simulation may not be sufficiently accurate, or the labelled rotations may not match the reality exactly. This is difficult to rule out because, as we discussed in Chapter 2, no other datasets exist to compare with which contain the required labels.
Alternatively, the error could be caused by poorly matched features which should have been rejected by the RANSAC algorithm but have not. Finally, the cause could indeed be the same issue as was discussed in the previous chapter: the blurring and



(A) Accumulated loss in the attitude.



(B) Loss in the averaged angular velocity.

FIGURE 5.14: The loss in the predictions over three full rotations of the target.

distortions introduced by the convolutional filters can cause the feature points to appear to be further from the centre of rotation than in reality, which could result in a smaller predicted rotation. Unfortunately, it has not yet been possible to track down and eliminate the source of this error.

5.3.3.2 Using the Predicted Rotations

Finally, we discuss how the predicted instantaneous rotation at each iteration can be used by the chaser spacecraft. Unfortunately, due to the small errors present at each step, it is unrealistic to be able to use this prediction to reconstruct the attitude of the target at each instant of time, because the errors will quickly accumulate. This is illustrated in the left part of Figure 5.14, where the accumulated loss very quickly explodes. It can be useful for tracking motion over a small timescale, for example tracking a docking point during rendezvous. However, if instead it is desired to determine the attitude of the satellite over a long time, it would be better to use one of the orientation detection approaches detailed earlier, in which the orientation is predicted at a single instant in time by comparing the observation with a 3D model or ground truth; in this case, there is no accumulation of errors caused by integrating a predicted angular velocity over time.

On the other hand, the second figure shows that this method is useful for determining the rotational state of the target, which is critical to enable several debris removal approaches. This information can allow the chaser to synchronise its angular motion with a tumbling target before capture and removal. If we assume the angular momentum of the target to be constant, which is realistic due to the small external forces acting on the satellite, we see that the estimate of the angular velocity can be improved over time. The speed of this convergence could clearly be improved by using a more complex approach such as a Kalman filter. However, we do see that the final value still contains a loss of approximately 20%, due to the systematic error discussed previously, so there are still improvements to be found.

Chapter 6

Conclusions

This thesis has aimed to illustrate the potential for an improved autonomous vision-based navigation system in space, by employing novel machine learning technologies. Specifically, methods have been proposed for improving attitude determination, relative navigation and autonomous docking, for unknown and uncooperative debris targets. These advancements will help to increase the technology readiness level of active debris removal and be essential for enabling autonomous debris removal and rendezvous missions in the future.

The key limiting factor in ML research in this domain is the lack of freely available training data. Therefore, a simulation framework has been constructed which is capable of generating large datasets of image data. In order to maximise the potential applications of the data, the images are labelled with a number of different parameters, including the relative states, debris class and surface labels. This data, along with the simulation software itself, is made freely available to encourage further research in this domain[1]. In addition, the ability to generate data under specific conditions is also provided, because one often-stated downside of ML approaches is their lack of generalisability to different situations.

Making use of this data, a neural machine learning model is trained to predict the change in attitude of an unknown target over time. A shallow convolutional neural network tracks the motion of low-level image features between successive timesteps, by implicitly matching features using only their index in the output. By training on the specific problem, it is shown that the convolutional feature detector outperforms conventional feature detection and matching approaches on the same test data. However, there can be problems caused by occlusions and disappearing edges, or by mismatched features due to the simplicity of the approach.

[1] `https://ben-guthrie.github.io/satvis/`

Using these matched features, the change in attitude is tracked over each timestep using a quaternion-based root-mean-square-deviation algorithm. Taking into account the problems stated above, an outlier rejection scheme is implemented to only keep the best feature matches. For this, the random sample consensus algorithm is adapted to be fully differentiable and parallelised, such that it can be implemented within an end-to-end trainable ML network. This approach performs equivalently to or better than the iterative closest point algorithm, which has previously been used in space missions, with a lower computational cost.

Finally, potential applications of ML are detailed for all sections of the guidance, navigation and control system. Two further technologies are proposed which can be applied to the visual navigation, in the form of object detection and image segmentation networks. These methods both are enabled by the developed simulation software. The state of the art in ML for spacecraft control is also briefly discussed, which is another exciting research area in which it is expected that ML will lead to many advancements in the future.

6.1 Further Work

This is evidently a very important topic for future research and innovation. To this end, a comprehensive list of potential future work is provided, to improve upon the approach in this work or investigate further applications of ML technologies. These are divided into separate topics.

6.1.1 Simulation Framework

Firstly, it would be useful to make further improvements to the simulation software. One such improvement would be expanding the dataset to increase the variation of the datasets. This could involve increasing the number of satellite targets, or perhaps more interestingly, including other types of debris as targets. In particular, the current satellite targets do not accurately represent the most common targets for ADR missions; including objects such as rocket upper stages may make the data more suitable specifically for ADR applications. Similarly, it may be interesting to ensure that the same methods perform well when the target is a broken-down part of a satellite.

The simulation software has been constructed in such a way that it is very easy to add different debris targets to the data, simply by adding the model file to the correct directory; instructions for this are provided on the web page. Therefore, the above task will be a relatively simple improvement to the data. In addition, this allows specific debris targets to be added to the software to generate data for a specific mission.

Another simple but valuable task would involve enhancing the data labels. The satellite targets are currently labelled with only two different surface types. In the future, it will be interesting to incorporate more varied surface labels. This will enable more in-depth analysis of the surface segmentation method while also increasing its possible applications, for example by detecting engines or surface materials. Again, increasing the number of surface labels in the simulation is relatively simple, requiring only that a black-and-white surface model be added to the relevant directory.

On a similar note, the debris classifier labels could be made more useful for the specific tasks which might be necessary in a debris removal mission. For example, instead of classifying debris objects by name, it may be better to use the debris type as the classifier. This would enable the chaser to determine the optimal removal method for the specific object observed.

A more complex suggestion for further work could be improving the realism of the simulated image data. While the data is simulated in a realistic environment, accurately emulating the situation in orbit, there are a number of camera and lighting effects that occur in the real world which have not been considered. The simulation could thus be further developed by including these effects, such as lens flare or improved reflections. In addition, it would be interesting to use known on-board camera parameters to better simulate the real-world situation.

Finally, expanding on the previous suggestion, data could be generated using a real-world simulation environment, as described in Section 2.3.5.1. However, it is more difficult and time-consuming to generate the labels for this realistic data, so these datasets will be useful for validating the developed models but would not be large enough to use for training. Park et al. (2021) have already constructed a simulation environment for this purpose and demonstrated its usefulness for validating machine learning models.

6.1.2 Rotation Estimation Model

A number of potential research topics related to our rotation estimation model are also determined. Several of these suggestions have been brought up during the previous analysis; they are compiled together in this section for reference.

One key problem with the rotation estimation approach is shown to be linked to the depth data. It is implied that the model does not make the best use of this data. Thus, the model could be improved by using the depth in different ways. One such suggestion is to use the depth to extract the features, either instead of the colour image or by combining the colour and depth data into a single input image. The network would then have access to a larger amount of information from which to determine the best feature locations. This can be achieved by simply providing a different input to the model and re-training it; there should be no need to change the structure of the network itself.

However, there remains the problem which has been discussed during the analysis, where the feature points after passing through the network are not exactly lined up with the true location of the observed feature. This is caused by the distortions introduced by the convolutional filters in the network. Therefore, extracting the depth value using the feature coordinates results in incorrect depth readings, which are sometimes not on the object.

This problem could potentially be solved by acquiring a depth value alongside the feature locations, instead of afterwards. For example, the image and depth data could both be provided to the feature extraction network, with a separate branch of the network being used to predict the depth for each extracted feature. This will result in more accurate 3D points which will likely reduce the remaining systematic errors observed in the outputs of the model.

It could also be possible to make better use of the outputs of the model themselves. It has been shown that an improved estimate can be achieved by taking the average predictions over different sizes of timestep. However, in this implementation each prediction was given equal weight. The model provides further information which can give an indication of the confidence of a prediction; these can be used to give more weight to those predictions with a higher confidence. For example, the number of inlier features, confidence of the extracted features and previous trends could all be used to estimate the strength of a prediction.

Furthermore, the predictions could be improved over time by making use of a filtering scheme. In this work, both Kalman filtering and neural network approaches are investigated to achieve this. However, in both cases, problems are found which have not yet been solved. In the case of Kalman filtering, the difficulty comes from accurately estimating the noise in the system, as this is required for initialising the filter. This approach would require more investigation into modelling the noise. On the other hand, passing the previous estimates through a neural network is shown to have promising results, but requires further work into the method of training the model, as discussed in Chapter 5.

Finally, a valuable experiment to validate the real-world performance of the approaches would involve testing the algorithms on a small-scale GPU. A chaser spacecraft will have significant power and memory limitations, so this will be critical to ensure on-board computation is possible. An analysis into the characteristics of embedded GPUs for space applications is performed in Kosmidis et al. (2019).

6.1.3 Potential Applications

Finally, there are many potential further applications of ML to this problem. A number of such applications are suggested in Chapter 3, which are believed to be interesting topics for future research. In addition, combining several different applications together could lead to further improvements.

For example, the three models demonstrated in this thesis could be adopted into an autonomous visual navigation system. The different elements can be combined together to further improve the performance. As an example, the satellite detection method tracks and classifies a debris object during the approach phase. The detector and rotation estimator together provide an estimate of the motion of the target during the approach. The rotation estimation and surface segmentation methods work together to describe the rotation of object and to select and track an optimal contact point. The debris removal approach, which is selected given the class of debris object, is then performed targeting this point.

Alternatively, the described approach could be adapted to different tasks. One such example is asteroid navigation. This is a problem to which the approach appears to be well suited, as there is often very little information about the target available prior to the mission. In addition, the need for autonomy is even more pronounced in this case, due to the larger distance from the ground. Similarly to the debris removal problem, there is a requirement for classification, relative motion estimation and landing site selections, which can be provided by the described approaches. The feature extraction approach is also likely to perform well in this case due to the heavily textured surfaces.

References

Farhad Aghili and Kourosh Parsa. Motion and parameter estimation of space objects using laser-vision data. *Journal of Guidance, Control, and Dynamics*, 32(2):538–550, 2009. .

Guglielmo S. Aglietti, Ben Taylor, Simon Fellowes, Thierry Salmon, Ingo Retat, Alexander Hall, Thomas Chabot, Aurélien Pisseloup, C. Cox, Zarkesh A., A. Mafficini, N. Vinkoff, K. Bashford, Cesar Bernal, François Chaumette, Alexandre Pollini, and Willem H. Steyn. The active space debris removal mission removedebris. part 2: In orbit operations. *Acta Astronautica*, 168:310–322, 2020. ISSN 0094-5765. .

Olusanya Agunbiade and Tranos Zuva. Simultaneous localization and mapping in application to autonomous robot. In *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, pages 1–5, 2018. .

Ryan Alimo, Daniel Jeong, and Kingson Man. Explainable non-cooperative spacecraft pose estimation using convolutional neural networks. *AIAA Scitech 2020 Forum*, Jan 2020. .

James A Anderson. *An introduction to neural networks*. MIT press, 1995.

K. Somani Arun, Thomas S. Huang, and Steven D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8.

Heike Benninghoff, Toralf Boge, and Florian Rems. Autonomous navigation for on-orbit servicing. *KI - Künstliche Intelligenz*, 28(2):77–83, Jun 2014. ISSN 1610-1987. .

Riccardo Benvenuto and Riccardo Carta. Active debris removal system based on tethered-nets: experimental results,. In *Proc. 9th Pegasus-AIAA Student Conference*, 2013.

Riccardo Benvenuto and Michèle Lavagna. Flexible capture device for medium to large debris active removal: simulation results to drive the experiments. In *Proc. 12th Symposium on Advanced Space Technologies in Robotics and Automation*, 2013.

Riccardo Benvenuto, Samuele Salvi, and Michèle Lavagna. Dynamics analysis and GNC design of flexible systems for space debris active removal. *Acta Astronautica*, 110(Supplement C):247 – 265, 2015. Dynamics and Control of Space Systems.

Robin Biesbroek, Luisa Innocenti, Andrew Wolahan, and Sara Morales Serrano. e. deorbit—esa's active debris removal mission. In *Proceedings of the 7th European Conference on Space Debris*, page 10. ESA Space Debris Office, 2017.

James D. Biggs and Hugo Fournier. Neural-network-based optimal attitude control using four impulsive thrusters. *Journal of Guidance, Control, and Dynamics*, 43(2): 299–309, 2020. .

Bernd Bischof. *Roger - Robotic Geostationary Orbit Restorer*. American Institute of Aeronautics and Astronautics, 2003.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Manuel Blanco-Muriel, Diego C. Alarcón-Padilla, Teodoro López-Moratalla, and MartÍn Lara-Coira. Computing the solar vector. *Solar Energy*, 70(5):431 – 441, 2001. ISSN 0038-092X. .

Claudio Bombardelli, Hodei Urrutxua, Mario Merino, Jesús Peláez, and Eduardo Ahedo. The ion beam shepherd: A new concept for asteroid deflection. *Acta Astronautica*, 90(1):98 – 102, 2013. NEO Planetary Defense: From Threat to Action - Selected Papers from the 2011 IAA Planetary Defense Conference.

Christophe Bonnal, Jean-Marc Ruault, and Marie-Christine Desjean. Active debris removal: Recent progress and current trends. *Acta Astronautica*, 85:51 – 60, 2013. ISSN 0094-5765. .

Andrea Brandonisio, Michele Lavagna, and Davide Guzzetti. Reinforcement learning for uncooperative space objects smart imaging path-planning. *The Journal of the Astronautical Sciences*, pages 1–25, 2021.

Vitali Braun, A. Lüpken, Sven Flegel, Johannes Gelhaus, Marek Moeckel, Christopher Kebschull, Carsten Wiedemann, and Peter Vörsmann. Active debris removal of multiple priority targets. *Advances in Space Research*, 51:1638–1648, 05 2013. .

John Canny. A computational approach to edge detection. In Martin A. Fischler and Oscar Firschein, editors, *Readings in Computer Vision*, pages 184 – 203. Morgan Kaufmann, San Francisco (CA), 1987. ISBN 978-0-08-051581-6. .

Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *CoRR*, abs/1605.07678, 2016.

Ian Carnelli, Bernd Dachwald, and Massimiliano Vasile. Evolutionary neurocontrol: A novel method for low-thrust gravity-assist trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 32(2):616–625, 2009. .

Valerio Carruba, S Aljbaae, RC Domingos, A Lucchini, and P Furlaneto. Machine learning classification of new asteroid families members. *Monthly Notices of the Royal Astronomical Society*, 496(1):540–549, 2020.

Thomas E. Carter. State transition matrices for terminal rendezvous studies: Brief survey and new example. *Journal of Guidance, Control, and Dynamics*, 21(1):148–155, 1998. .

Marco M. Castronuovo. Active space debris removal—a preliminary mission analysis and design. *Acta Astronautica*, 69(9):848 – 859, 2011. ISSN 0094-5765. .

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin P. Murphy, and Alan Loddon Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:834–848, 2018.

Lin Cheng, Zhenbo Wang, Fanghua Jiang, and Chengyang Zhou. Real-time optimal control for spacecraft orbit transfer via multiscale deep neural networks. *IEEE Transactions on Aerospace and Electronic Systems*, 55(5):2436–2450, 2019. .

Alessandro Chiesa, Franco Fossati, Giovanni Gambacciani, and Emanuele Pensavalle. Enabling technologies for active space debris removal: The cadet project. 2015.

D. Choukroun, I.Y. Bar-Itzhack, and Y. Oshman. Novel quaternion kalman filter. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):174–190, 2006. .

W. H. Clohessy and R. S. Wiltshire. Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, Sep 1960. .

Pablo Colmenarejo, Giovanni Binet, Luigi Strippoli, Thomas Vincent Peters, and Mariella Graziano. GNC aspects for active debris removal. In *Proc. 2nd CEAS Specialist Conference on Guidance, Navigation and Control*, April 2013.

Pablo Colmenarejo, Mariella Graziano, Gabrielli Novelli, Dario Mora, Pedro Serra, Angelo Tomassini, Karol Seweryn, G. Prisco, and Jesús Gil-Fernández. On ground validation of debris removal technologies. 07 2017.

Evangelos Coutsias, Chaok Seok, and Ken Dill. Using quaternions to calculate rmsd. *Journal of computational chemistry*, 25:1849–57, 11 2004. .

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

Li Deng and Xiao Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5): 1060–1089, May 2013.

Anamika Dhillon and Gyanendra K Verma. Convolutional neural network: a review of models, methodologies and applications to object detection. *Progress in Artificial Intelligence*, 9(2):85–112, 2020.

Gangqi Dong and Z.H. Zhu. Position-based visual servo control of autonomous robotic manipulators. *Acta Astronautica*, 115:291 – 302, 2015. ISSN 0094-5765. .

Roger Dudziak, Sean Tuttle, and Simon Barraclough. Harpoon technology development for the active removal of space debris. *Advances in Space Research*, 56 (3):509 – 527, 2015. Advances in Asteroid and Space Debris Science and Technology - Part 1.

Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006. ISSN 1070-9932. .

David W Eggert, Adele Lorusso, and Robert B Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine vision and applications*, 9(5):272–290, 1997.

European Space Agency. ESA's annual space environment report, 2021. URL `https://www.sdo.esoc.esa.int/environment_report/Space_Environment_Report_latest.pdf`. Reference: GEN-DB-LOG-00288-OPS-SD.

Sachin Sudhakar Farfade, Mohammad Saberian, and Li-Jia Li. Multi-view face detection using deep convolutional neural networks, 2015.

Olivier Faugeras. Three-dimensional computer vision: a geometric viewpoint. *MIT press*, 01 1993.

Lorenzo Federici, Boris Benedikter, and Alessandro Zavoli. *Machine Learning Techniques for Autonomous Spacecraft Guidance during Proximity Operations*. 2021. .

Wigbert Fehse. *Automated Rendezvous and Docking of Spacecraft*. Cambridge Aerospace Series. Cambridge University Press, 2003.

Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. .

Angel Flores-Abad, Ou Ma, Khanh Pham, and Steve Ulrich. A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, 68:1 – 26, 2014.

Jason L. Forshaw, Guglielmo S. Aglietti, Nimal Navarathinam, Haval Kadhem, Thierry Salmon, Aurélien Pisseloup, Eric Joffre, Thomas Chabot, Ingo Retat, Robert Axthelm, Simon Barraclough, Andrew Ratcliffe, Cesar Bernal, François Chaumette, Alexandre Pollini, and Willem H. Steyn. Removedebris: An in-orbit active debris removal demonstration mission. *Acta Astronautica*, 127:448 – 463, 2016. ISSN 0094-5765. .

Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 134:1–21, 2015.

Jorge Fuentes-Pacheco, Jose Ascencio, and J Rendon-Mancha. Visual simultaneous localization and mapping: A survey. *Artificial Intelligence Review*, 43, 11 2015. .

Roberta Garcia, Paula Pardal, Helio Kuga, and Maria Zanardi. Nonlinear filtering for sequential spacecraft attitude estimation with real data: Cubature kalman filter, unscented kalman filter and extended kalman filter. *Advances in Space Research*, V63: 1038–1050, 01 2019. .

Brian Gaudet, Richard Linares, and Roberto Furfaro. Deep reinforcement learning for six degree-of-freedom planetary landing. *Advances in Space Research*, 65(7): 1723–1741, 2020. ISSN 0273-1177. .

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618, 9780262035613.

Ben Guthrie, Minkwan Kim, Hodei Urrutxua, and Jonathon Hare. Image-based attitude determination of co-orbiting satellites enhanced with deep learning technologies. In *Advances in the Astronautical Sciences*, volume 175, pages 3164–3182. American Astronautical Society, May 2021a. URL `https://eprints.soton.ac.uk/451304/`.

Ben Guthrie, Minkwan Kim, Hodei Urrutxua, and Jonathon Hare. Attitude reconstruction of an unknown co-orbiting satellite target using machine learning technologies. In *2021 AAS/AIAA Astrodynamics Specialist Conference*, July 2021b. URL `https://eprints.soton.ac.uk/451305/`.

Ben Guthrie, Minkwan Kim, Hodei Urrutxua, and Jonathon Hare. Image-based attitude determination of co-orbiting satellites using deep learning technologies. *Aerospace Science and Technology*, 2021c. ISSN 1270-9638. . URL `https://www.sciencedirect.com/science/article/pii/S1270963821007422`.

Ben Guthrie, Minkwan Kim, Hodei Urrutxua, and Jonathon Hare. Improving autonomous guidance using machine learning technologies. In *Stardust-R – Second Global Virtual Workshop*, September 2021d.

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. ISBN ISBN: 0521540518.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

Sina Honari, Pavlo Molchanov, Stephen Tyree, Pascal Vincent, Christopher Pal, and Jan Kautz. Improving landmark localization with semi-supervised learning, 2017.

Du Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35:155–164, 10 2009. .

SM Sofiqul Islam, Shanto Rahman, Md Mostafijur Rahman, Emon Kumar Dey, and Mohammad Shoyaib. Application of deep learning to computer vision: A comprehensive study. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 592–597, May 2016.

Dario Izzo. 1st act global trajectory optimisation competition: Problem description and summary of the results. *Acta Astronautica*, 61(9):731–734, 2007. ISSN 0094-5765. . Global Trajectory Optimization. Results of the First Competition Organised by the Advanced Concept Team (ACT) of the European Space Agency (ESA).

Dario Izzo, Marcus Märtens, and Binfeng Pan. A survey on artificial intelligence trends in spacecraft guidance dynamics and control. *Astrodynamics*, 3(4):287–299, 2019.

Dario Izzo and Ekin Öztürk. Real-time guidance for low-thrust transfers using deep neural networks. *Journal of Guidance, Control, and Dynamics*, 44(2):315–327, 2021. .

Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NIPS 2018)*, pages 1–12, 2018.

Xiuqiang Jiang, Shuang Li, and Roberto Furfaro. Integrated guidance for mars entry and powered descent using reinforcement learning and pseudospectral method. *Acta Astronautica*, 163:114–129, 2019. ISSN 0094-5765. . Fourth IAA Conference on Dynamics and Control of Space Systems (DYCOSS2018).

Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Ayush Chaurasia, TaoXie, Liu Changyu, Abhiram V, Laughing, tkianai, yxNONG, Adam Hogan, lorenzomammana, AlexWang1900, Jan Hajek, Laurentiu Diaconu, Marc, Yonghye Kwon, oleg, wanghaoyang0106, Yann Defretin, Aditya Lohia, ml5ah, Ben Milanko, Benjamin Fineran, Daniel Khromov, Ding Yiwei, Doug, Durgesh, and Francisco Ingham. ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations, April 2021. URL https://doi.org/10.5281/zenodo.4679653.

Keyvan Kanani, Antoine Petit, Eric Marchand, Thomas Chabot, and Bernard Gerber. Vision based navigation for debris removal missions. *Proceedings of the International Astronautical Congress, IAC*, 4, 01 2012.

Marshall H. Kaplan. *Modern spacecraft dynamics and control*. John Wiley and Sons, Inc., 1976.

Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: making rgb-based 3d detection and 6d pose estimation great again. *CoRR*, abs/1711.10006, 2017.

Erwan Kervendal, Thomas Chabot, and Keyvan Kanani. *GNC Challenges and Navigation Solutions for Active Debris Removal Mission*, pages 761–779. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

Donald J. Kessler and Burton G. Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, 83(A6): 2637–2646, 1978.

Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

Mate Kisantal, Sumant Sharma, Tae Ha Park, Dario Izzo, Marcus Martens, and Simone D'Amico. Satellite pose estimation challenge: Dataset, competition design, and results. *IEEE Transactions on Aerospace and Electronic Systems*, 56(5):4083–4098, 2020. .

Ksenia Klionovska, Jacopo Ventura, Heike Benninghoff, and Felix Huber. Close range tracking of an uncooperative target in a sequence of photonic mixer device (pmd) images. *Robotics*, 7:5, 01 2018. .

Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine*, 23(1):89 – 109, 2001.

Leonidas Kosmidis, Iván Rodriguez, Álvaro Jover, Sergi Alcaide, Jérôme Lachaize, Jaume Abella, Olivier Notebaert, Francisco J. Cazorla, and David Steenari. Gpu4s: Embedded gpus in space. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pages 399–405, 2019. .

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

Jason Ku, Ali Harakeh, and Steven L Waslander. In defense of classical image processing: Fast depth completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 16–22. IEEE, 2018.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. .

Daero Lee, John Cochran, and Jung Jo. Solutions to the variational equations for relative motion of satellites. *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, 30:669–678, 05 2007. .

Ern J Lefferts, F Landis Markley, and Malcolm D Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5): 417–429, 1982. .

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Richard Linares and Roberto Furfaro. Space object classification using deep convolutional neural networks. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1140–1146, 2016.

J.-C. Liou, N.L. Johnson, and N.M. Hill. Controlling the growth of future leo debris populations with active debris removal. *Acta Astronautica*, 66(5):648 – 653, 2010.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004. ISSN 1573-1405. .

W. MacKunis, F. Leve, P.M. Patre, N. Fitz-Coy, and W.E. Dixon. Adaptive neural network-based satellite attitude control in the presence of cmg uncertainty. *Aerospace Science and Technology*, 54:218–228, 2016. ISSN 1270-9638. .

Siddharth Mahendran, Haider Ali, and René Vidal. 3d pose regression using convolutional neural networks. *CoRR*, abs/1708.05628, 2017.

Landis Markley and John Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. 01 2014. ISBN ISBN: 978-1-4939-0802-8. .

Landis Markley, Yang Cheng, John Crassidis, and Yaakov Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30:1193–1196, 07 2007. .

Darren McKnight, Frank Di Pentino, Adam Kaczmarek, and Steve Knowles. Detumbling rocket bodies in preparation for active debris removal. In *6th European Conference on Space Debris*, 2013.

William McMahan, Vilas Chitrakaran, Matthew Csencsits, Drew Dawson, I.D. Walker, Bryan Jones, M. Pritts, D. Dienno, M. Grissom, and Chris Rahn. Field trials and testing of the octarm continuum manipulator. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 2336–2341, 06 2006. .

Dieter Mehrholz, L Leushacke, W Flury, R Jehn, H Klinkrad, and M Landgraf. Detecting, tracking and imaging space debris. *ESA Bulletin*, 109:128–134, February 2002.

Robert G. Melton. Time-explicit representation of relative motion between elliptical orbits. *Journal of Guidance, Control, and Dynamics*, 23(4):604–610, 2000. .

Jonathan Missel and Daniele Mortari. Removing space debris through sequential captures and ejections. *Journal of Guidance, Control, and Dynamics*, 36(3):743–752, 2013.

Masaaki Mokuno and Isao Kawano. In-orbit demonstration of an optical navigation system for autonomous rendezvous docking. *Journal of Spacecraft and Rockets*, 48(6): 1046–1054, 2011. .

Trier Mortlock and Zaher M Kassas. Assessing machine learning for leo satellite orbit determination in simultaneous tracking and navigation. In *2021 IEEE Aerospace Conference (50100)*, pages 1–8. IEEE, 2021.

Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. 2009.

Pablo Muñoz, Vicente Companys, Frank Budnik, Bernard Godard, Dario Pellegrinetti, Gabriele Bellei, Rainer Bauske, and Waldemar Martens. Rosetta navigation during the end of mission phase. In *2017 International Symposium on Space Flight Dynamics, Matsuyama, Japan, ISSFD-2017-015*, 2017.

Shin-Ichiro Nishida, Satomi Kawamoto, Yasushi Okawa, Fuyuto Terui, and Shoji Kitamura. Space debris removal system using a small satellite. *Acta Astronautica*, 65 (1):95 – 102, 2009.

Charles Oestreich, Tae W. Lim, and Randy Broussard. *On-Orbit Relative Pose Initialization via Convolutional Neural Networks*, page 0457. 2020. .

Roberto Opromolla, Giancarmine Fasano, Giancarlo Rufino, and Michele Grassi. A model-based 3d template matching technique for pose acquisition of an uncooperative space object. *Sensors*, 15(3):6360–6382, 2015. ISSN 1424-8220. .

Tae Ha Park and Simone D'Amico. Generative model for spacecraft image synthesis using limited dataset. In *2020 AAS/AIAA Astrodynamics Specialist Conference*, 2020.

Tae Ha Park, Marcus Märtens, Gurvan Lecuyer, Dario Izzo, and Simone D'Amico. SPEED+: next generation dataset for spacecraft pose estimation across domain gap. *CoRR*, abs/2110.03101, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Claude R. Phipps. A laser-optical system to re-enter or lower low earth orbit space debris. *Acta Astronautica*, 93(Supplement C):418 – 429, 2014.

M Piccinin, G Zanotti, S Silvestrini, A Capannolo, A Pasquale, and M Lavagna. Cubesat exploration missions to binary asteroids: on board autonomy and intelligent imaging towards science return enhancement. In *2020 AAS/AIAA Astrodynamics Specialist Conference*, pages 1–14, 2020.

Robin Pinson, Richard Howard, and Andrew Heaton. Orbital express advanced video guidance sensor: Ground testing, flight results and comparisons. August 2008. ISBN 978-1-60086-999-0. .

Camille Pirat, Muriel Richard-Noca, Christophe Paccolat, Federico Belloni, Reto Wiesendanger, Daniel Courtney, Roger Walker, and Volker Gass. Mission design and gnc for in-orbit demonstration of active debris removal technologies with cubesats. *Acta Astronautica*, 130:114–127, 2017. ISSN 0094-5765. .

Pedro F. Proença and Yang Gao. Deep learning for spacecraft pose estimation from photorealistic rendering. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6007–6013, 2020.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

D Reintsema, J Thaeter, A Rathke, W Naumann, P Rank, and J Sommer. Deos–the german robotics approach to secure and de-orbit malfunctioned satellites from low earth orbits. In *Proceedings of the i-SAIRAS*, pages 244–251. Japan Aerospace Exploration Agency (JAXA) Japan, 2010.

Florian Rems, Juan Antonio Moreno Gonzalez, Toralf Boge, Sebastian Tuttas, and Uwe Stilla. Fast initial pose estimation of spacecraft from lidar point cloud data. In *Proc. 13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA). Noordwijk, The Netherlands*, 2015.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

F Rosso, Francesco Gallo, Walter Allasia, E Licata, Paolo Prinetto, D Rolfo, Pascal Trotta, Alain Favetto, Marco Paleari, and Paolo Ariano. Stereo vision system for capture and removal of space debris. pages 201–207, 01 2013.

Nick Rowell, Matin N. Dunstan, Steve M. Parkes, Jesus Gil-Fernández, Irene Huertas, and Sohrab Salehi. Autonomous visual recognition of known surface landmarks for optical navigation around asteroids. *The Aeronautical Journal (1968)*, 119(1220): 1193–1222, 2015. .

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

Antonio Ruggiero, Mariano Andrenucci, and Leopold Summerer. Low-thrust missions for expanding foam space debris removal. In *Proc. 32nd Internationa Electric Propulsion Conference*, 2011.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X.

Timothy E. Rumford. Demonstration of autonomous rendezvous technology (DART) project summary. volume 5088, 2003. .

Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001. .

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. .

Tara N. Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for LVCSR. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618, May 2013.

Seyed Sadegh Mohseni Salehi, Shadab Khan, Deniz Erdogmus, and Ali Gholipour. Real-time deep pose estimation with geodesic loss for image-to-template rigid registration. *IEEE transactions on medical imaging*, 38(2):470–481, 2018.

Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Learning 3-d object orientation from images. pages 794 – 800, 06 2009. .

Christopher J. Shallue and Andrew Vanderburg. Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. *The Astronomical Journal*, 155(2):94, Jan 2018. ISSN 1538-3881. .

Minghe Shan, Jian Guo, and Eberhard Gill. Review and comparison of active space debris capturing and removal methods. *Progress in Aerospace Sciences*, 80 (Supplement C):18 – 32, 2016.

Sumant Sharma and Simone D'Amico. Pose estimation for non-cooperative rendezvous using neural networks, 2019.

Sumant Sharma and S. D'Amico. Neural network-based pose estimation for noncooperative spacecraft rendezvous. *IEEE Transactions on Aerospace and Electronic Systems*, 56:4638–4658, 2020.

Maksim Shirobokov, Sergey Trofimov, and Mikhail Ovchinnikov. Survey of machine learning techniques in spacecraft control design. *Acta Astronautica*, 186:87–97, 2021. ISSN 0094-5765. .

Malcolm Shuster. A simple kalman filter and smoother for spacecraft attitude. *Journal of the Astronautical Sciences*, 37(1):89–106, 1989.

Ari Silburt, Mohamad Ali-Dib, Chenchong Zhu, Alan Jackson, Diana Valencia, Yevgeni Kissin, Daniel Tamayo, and Kristen Menou. Lunar crater identification via deep learning. *Icarus*, 317:27–38, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Juergen Starke, Bernd Bischof, W-O Foth, et al. Roger a potential orbital space debris removal system. *38th COSPAR Scientific Assembly*, 38:2, 2010.

Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. 09 2018.

Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

Jonathan Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *CoRR*, abs/1406.2984, 2014.

Christopher Trentlage and Enrico Stoll. The applicability of gecko adhesives in a docking mechanism for active debris removal missions. In *Proc. 13th Symposium on Advanced Space Technologies in Robotics and Automation*, 2015.

J Tschauner and P Hempel. Rendezvous zu einem in elliptischer bahn umlaufenden ziel. *Astronautica Acta*, 11(2):104–+, 1965.

Thomas Uriot, Dario Izzo, Luís F Simões, Rasit Abay, Nils Einecke, Sven Rebhan, Jose Martinez-Heras, Francesca Letizia, Jan Siminski, and Klaus Merz. Spacecraft collision avoidance challenge: design and results of a machine learning competition. *Astrodynamics*, pages 1–20, 2021.

David A. Vallado and Wayne D. McClain. *Fundamentals of Astrodynamics and Applications*, volume 4, chapter 6, pages 389–416. Springer, Berlin, Germany, 3 edition, 2007. ISBN 978-0-387-71831-6.

José Vasconcelos, Murray Kerr, Paulo Rosa, Miguel Hagenfeldt, Andrea Fabrizi, Baltazar Parreira, Jorge Ambrósio, Massimo Casasco, and Guillermo Ortega. GNC design and validation for rendezvous, detumbling, and de-orbiting of envisat using a clamping mechanism. March 2016.

Krisztián Vida and Rachael M Roettenbacher. Finding flares in kepler data using machine-learning tools. *Astronomy & Astrophysics*, 616:A163, 2018.

Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6d: Self-supervised monocular 6d object pose estimation. In *European Conference on Computer Vision*, pages 108–125. Springer, 2020.

Caisheng Wei, Jianjun Luo, Honghua Dai, Zilin Bian, and Jianping Yuan. Learning-based adaptive prescribed performance control of postcapture space robot-target combination without inertia identifications. *Acta Astronautica*, 146: 228–242, 2018. ISSN 0094-5765. .

Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. *CoRR*, abs/1502.05908, 2015.

Toru Yamamoto, Yu Nakajima, Takahiro Sasaki, Naoki Okada, Misuzu Haruki, and Koji Yamanaka. GNC strategy to capture, stabilize and remove large space debris. In *1st International Orbital Debris Conference, Texas, USA*, number 6109, 2019.

Koji Yamanaka and Finn Ankersen. New state transition matrix for relative motion on an arbitrary elliptical orbit. *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, 25:60–66, 01 2002. .

Melak Zebenay, Roberto Lampariello, Toralf Boge, and Daniel Choukroun. A new contact dynamics model tool for hardware-in-the-loop docking simulation. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2012.

Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker Jr, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017.

Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations, 2018.

Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4591–4600, 2019.

Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey, 2019.

Jiří Šilha, Jean-Noël Pittet, Michal Hamara, and Thomas Schildknecht. Apparent rotation properties of space debris extracted from photometric measurements. *Advances in Space Research*, 61(3):844–861, 2018. ISSN 0273-1177. .