Robotic 3D Reconstruction Utilising Structure from Motion

Louis G. Clift

A thesis submitted for the degree of Doctor of Philosophy

School of Computer Science and Electronic Engineering
University of Essex

January 2017

Abstract

Sensing the real-world is a well-established and continual problem in the field of robotics. Investigations into autonomous aerial and underwater vehicles have extended this challenge into sensing, mapping and localising in three dimensions. This thesis seeks to understand and tackle the challenges of recovering 3D information from an environment using vision alone. There is a well-established literature on the principles of doing this, and some impressive demonstrations; but this thesis explores the practicality of doing vision-based 3D reconstruction using multiple, mobile robotic platforms, the emphasis being on producing accurate 3D models. Typically, robotic platforms such as UAVs have a single on-board camera, restricting which method of visual 3D recovery can be employed. This thesis specifically explores Structure from Motion, a monocular 3D reconstruction technique which produces detailed and accurate, although slow to calculate, 3D reconstructions. It examines how well proof-of-concept demonstrations translate onto the kinds of robotic systems that are commonly deployed in the real world, where local processing is limited and network links have restricted capacity. In order to produce accurate 3D models, it is necessary to use high-resolution imagery, and the difficulties of working with this on remote robotic platforms is explored in some detail.

Acknowledgements

I would like to thank my supervisor, Dr Adrian Clark, who has not only supported me throughout this PhD but was the one who started the whole journey off and made it possible with funding from the EPSRC. There have been many twists and turns during the years and it has been a real pleasure to work alongside Dr Clark. I would also like to add a special thanks to Dr Clark for enabling me to explore other research areas and supporting research which at the time was not directly related to primary research topic. Thank you to Dr Adeyemi-Ejeye and the Network Access Laboratory for inviting me to collaborate and produce novel work in the field of Ultra-HD video streaming.

The School of Computer Science & Engineering has felt like home with countless hours spent exploring new ideas and hammering away at the keyboard to make them work. Access to lab space and kit has never been an issue and I would like to thank Robin Dowling of the robotics arena for the support and effort in keep the robots ticking during the research. I would also like to thank the University's Media Services team for supporting my research in large-format image processing by providing cameras, broadcast cranes and other equipment.

I would like to thank all of my friends and colleagues who have helped keep me sane and made the experience truly worthwhile. Finally I would like to add a special thanks to my family who have always been there to support me and specifically to my Mum and Dad who have been my inspiration and motivation to work hard and push forward.

Contents

List of Figures

CHAPTER 1

Introduction

Digitising the real world in a form that computers can access for visualisation and decision making is a complex challenge which has been at the forefront of many fields within computer science. An accurate 3D model provides invaluable information that extends beyond visualisation. 3D modelling is often used in the design stages of a project with the aid of Computer-Aided Design (CAD) or Computer-Aided Engineering (CAE) software packages which are used to provide rich, detailed models of the final product. These systems are in use in projects ranging from small jewellery through to stadiums and skyscrapers. Recent developments in software and hardware have introduced Computer-Aided Manufacturing (CAM) systems into industry which enable 3D models to be printed straight from an engineer's workstation. Computer-aided systems are beginning to offer designers tools which enable them to capture objects and allow a user to reverse-engineer the real world.

This work explores the specifics of capturing the real-world through the use of computer vision algorithms, namely Structure from Motion, for use in computer systems whether that be for art, design, documentation or robotics.

1.1 Motivation and Objectives

This research is inspired by the accuracy and impact of 3D reconstructions when used in visualisation in either virtual or augmented reality systems. The work is ultimately driven by the desire to bring 3D reconstruction methods into the field of robotics for use in live mapping and navigation tasks. Both aerial and underwater systems face similar challenges in localisation and path planning in 3D space. Not only does a vehicle need to know its (often rough) location within the environment but it needs to plan a trajectory through the unknown environment too, taking into account any obstacles and more recently, other remote systems. This is a well established problem in the field which is discussed further in Chapter 2.

The objective of the research within this thesis is to break down Structure from Motion, a 3D reconstruction technique which uses 2D images to create 3D models. The aim is to evaluate the use of 3D point cloud data generated from vision as a tool for not only visualisation but for use in 3D navigation and path planning algorithms. Generating a 3D point cloud using Structure from Motion is a computationally expensive task requiring significant resources and time. To reach the goal of achieving real-time 3D point cloud generation using vision alone is a difficult task which may not be possible given the selected technique. Ultimately the research question that this thesis seeks to answer is: can 3D reconstruction be used with robotic systems to perform remote robotic 3D reconstruction?

There exists a trade-off between payload, flight time and core operational requirements. Path planning from a vision-only point cloud changes the operational requirements of a robotic platform, reducing the need to rely on more traditional approaches such as lasers.

1.2 Contributions

The first publication from this research was in the form of a poster presentation to the UAV Special Interest Group of the Remote Sensing & Photogrammetry Society, RSPSoc. The work, *Using UAVs and UGVs to Build 3D Models of Ground Features*, was presented at the 'Remote Sensing from small unmanned aerial systems' workshop held on 4th July 2013 [1]. The publication was aimed at providing an overview of the objectives of this thesis, comprised of early experimental work and feasibility studies. The workshop provided the opportunity to understand the field and confirm that the research aims of this thesis were in line with current UAV/UAS research.

Live Ultra-HD video transmission was investigated as part of a collaborative project with the University of Essex Access Laboratory. The work involved an investigation into on-line streaming requirements of live 4K content. The work resulted in being accepted for a dual-poster presentation after peer review at one of the leading conferences in the broadcast industry, the International Broadcast Convention Conference, IBC Conference in 2014. The publication, *Delivering Live 4K Broadcasting Using Today's Technology* [2], received much attention and discussion due to the intentional focus on using existing encoders to produce live 4K streaming rather than using the newer incoming H.265 (HEVC) standards. This work forms the core of Chapter 7.

Determining Positions and Distances Using Collaborative Robots [3] details the process of using a UAV's on-board downward-facing camera to infer position with respect to ground robots without the aid of GPS information. The process of calculating altitude based on the observed fiducials is detailed in Chapter 6.

The most recent publication, *Video Frame Extraction for 3D Reconstruction* [4], presents a method for selecting optimal frames from a live (or pre-recorded) video sequence for use in 3D reconstruction. The approach is specifically designed to produce images for Structure from Motion based 3D reconstruction. This work is discussed at length in Chapter 5.

1.3 Thesis Structure

The research area spans two key fields within Computer Science, namely Robotics and Computer Vision. The literature review in Chapter 2 builds up on the brief introduction in this chapter by exploring the literature in the areas of 3D modelling and reconstruction, as well as relevant robotic sensing approaches and collaborative robotics. This chapter presents Structure from Motion as the primary 3D reconstruction approach which will be at the core of the investigations throughout the thesis. Chapter 3 presents the fundamental mathematics, which start with a simple single camera model and progresses through to multi-view reconstruction.

Once the core theories have been presented and verified, the thesis changes focus towards a more practical approach. Chapter 4 explores the limitations and performance of 3D reconstruction utilising Structure from Motion following on from the theoretical discussions in the previous chapters. The experiments build up a set of capture requirements which provide a robotic platform with an optimised capture strategy. Following these experiments, Chapter 5 then proposes solutions to the problem of working with video sources with a 3D reconstruction pipeline.

Chapter 6 begins with a method for estimating altitude using an on-board camera from a low-cost consumer UAV. The chapter then discusses the importance of the choice in robotic communication systems and follows the implementation of such a system without the use of a generic framework.

Chapter 7 investigates a related issue of working with large image resolutions in vision systems. The chapter addresses the issue of transmitting live high-resolution imagery, a common stumbling block for vision-based systems, through experimental work and a series of scale tests before deploying the solution to a real-world test with a worldwide audience.

Chapter 8 returns back to the core theme of the thesis by combining the work of the previous chapters together to understand whether, given the results seen so far,

that the components can indeed form a complete robotic 3D capture system. This chapter builds a prototype which demonstrates the feasibility of such a system using the research undertaken in this thesis.

Finally, the thesis draws to a close in Chapter 9 where the research is critically assessed against its aims and objectives. The limitations of the implementations, experiments and hardware are all discussed. A discussion on the future direction of both the area and specifically the research presented in this thesis are covered before a final summary of the core work.

CHAPTER 2

Literature Review

This chapter explores the algorithms and techniques which are used to digitise the 3D world. There is a vast array of approaches available that enable engineers to develop systems that capture accurate models in a range of fields. We start by exploring traditional hardware solutions before entering the fields of Computer Vision and Robotics. Computer Vision offers a new set of solutions to the problem which rely on imagery rather than dedicated depth sensors such as sonar or laser range finders. Standard imaging sensors are less expensive than these specialist sensors and are able to avoid some of their restrictions and limitations.

Following on from the discussions of image-based 3D recovery this chapter then considers automated and robotic data capture systems and their intended applications. This review should provide the reader with a good base of where this work sets out to contribute both technically and the area of application for such a system.

2.1 3D Modelling

Before reviewing the research literature, it is worth pausing to understand the aim of a 3D capture system and answer a couple of key questions such as what do we

mean by a 3D reconstruction / model and where are these systems currently used?

Definition 2.1.1. Model — A three-dimensional representation of a person or thing or of a proposed structure, typically on a smaller scale than the original: *'a model of St Paul's Cathedral'. – Oxford English Dictionary*

Documentation is a key component to almost every project which is often produced to ensure the quality and accuracy of the project's outcomes. For many products this documentation will include a series of models to show what the end result will look like, or in the case of archæology what the item originally looked like. Typically we think of a model as being a miniature replica of an object. Modelling is used in a range of design phases where designers, engineers and architects will build various prototypes before committing to a final version for production. These models often start life on a computer in the form of 2D and 3D Computer Aided Drawings (CAD). CAD and the newer field of Computer Aided Engineering (CAE) systems offer designers, engineers and customers tools which produce a visualisation of the final product and, in the case of CAE systems, simulations of the performance of the design. CAD systems first emerged pre-1960's with a numerical control programming tool named PRONTO (Program for Numerical Tooling Operations) [5], published by Hanratty in 1957, followed by Sutherland's SKETCHPAD [6] in 1964. Since the introduction of these tools, significant developments in computing technology have pushed the field forward, leading to a vast number of proprietary packages being developed by industrial manufacturers, even before the rise of UNIX systems. Many of these early systems were designed to be replacements for manual drafting processes. It was not until the release of UNIX workstations and then the IBM PC in 1981 that the now widely known CAD/CAE/CAM systems emerged. Autodesk's AutoCAD was first released in 1983, the first significant CAD package for the PC that would be familiar to today's CAD users. These CAD systems have evolved and constantly push the limitations of computer-based modelling, simulation and design in both 2D and 3D content creation.

Once a computer-generated model is produced, a real-world prototype or product

mock-up then created to check that the computer-based designs would look and behave correctly. Accurate scale models are used in a range of fields such as aerospace, automotive design and architecture, where the models are placed in test facilities to evaluate response to real-world physics such as wind, water and natural events. Simulating real-world physics is the subject of a vast array of research disciplines including mathematics, computer graphics and physics amongst others.

The discussion to date covers the process of design through to build where the model is created, but not captured. If we refer to the dictionary definition of a model being a 3D representation of a person or object, we can also use the term 3D model in the converse case. How can we capture a real object which already exists? The primary topic of this thesis focuses specifically on this challenge of recovering accurate 3D models of real-world objects and environments from robotic systems in the real world.

A wide range of sectors is adopting 3D reconstructions to advance their respective fields both academically and commercially, such as archæology [7], engineering [8], design [9], medical diagnosis and imaging [10], global mapping [11, 12], as well as entertainment industries such as film, television and gaming. Through the use of digital models, medical professionals for example are able to offer more accurate and detailed examinations through the use of 3D models captured using Magnetic Resonance Imaging (MRI), X-ray Computed Tomography (CT) or Computerized Axial Tomography (CAT) scanners. The data from these scanners are processed by the systems to produce interactive and detailed models of the patient which can be analysed by experts anywhere in the world [10, 13, 14].

2.2 3D Sensors

3D reconstruction systems can be divided into two core themes, contact and non-contact systems. A contact-based system uses robotics to 'touch' a probe against a surface and measure the 3D position of the probe on-contact. These systems are

highly accurate and are used in precision manufacturing plants to confirm the accuracy of the parts produced. Due to their complexity and the need for accuracy, these systems are extremely expensive and specialised to a particular target. This section focuses on non-contact methods, as these systems are considerably more flexible in application. Of the non-contact approaches, techniques are categorised based on the method of interacting with the environment. Sensors which need to send out a signal or pattern and measure the difference are known as active sensors (e.g. lasers, sonar) whereas sensors which only observe without any interaction are passive (e.g. thermal, cameras, multi-spectral).

2.2.1 Active 3D Sensors

Laser scanning devices are often used in robotics and industrial safety systems to provide highly accurate distance measurements. Laser-based ranging was first used for meteorological studies in the 1960s where an optical system was used to replace radar systems for cloud measurement [15] and atmospheric studies [16]. A single laser distance measure is achieved by projecting an infra-red or ultra-violet laser beam into the environment and measuring the time taken for it to return. A single range (point) measurement takes less than 100 nanoseconds to complete (given a target distance less than 10 metres). Laser scanning range finders such as those produced by SICK and Hokuyo produce readings in an arc about the scanning head. The measurements are captured in the same method as a laser ranger except once the result is captured the scanning head rotates and repeats the pulse and measure procedure until the head completes a full 180° arc from left to right. Some sensors offer even wider measurement angles allowing for a complete 360° scan, such as the RPLIDAR A2 360° Laser Scanner.

Another form of laser scanner that is used specifically for 3D modelling is a hand-held scanner that stripes a laser across the target using a visible laser. The device couples the laser readings with internal sensors such as an Inertial Measurement Unit (IMU) to re-project and rectify the readings into a 3D model. These systems

require the target to remain fixed and static during the scanning process.

When it comes to modelling, laser scanners are limited by the 2D scanning path. To scan larger targets such as landscapes, rock formations or buildings an alternate type of laser scanner is required. 3D LiDAR is a 3D laser scanning system which uses a 2D laser scanning head to sweep horizontally for a single rotation and the entire head is then rotated about the horizontal axis and the horizontal sweep is captured again. A 3D LiDAR system is able to generate a complete point-cloud sphere surrounding the scanning head. A LiDAR scan will typically produce over 20,000 3D points that are used to recreate the surface of the target.

Laser scanners use precise equipment to measure tiny points within the environment to generate the 3D information. An alternative to sampling a point in space at a time is to project a stripe and observe the distortion in the line, a technique devised by Agin and Highnam [17]. These structured-light systems are formed of a projector and camera which is mounted slightly offset from the projector. Chen and Kak built on these principles to utilise structured light striping for both calibration and modelling in 3D robotic vision utilising projective geometry [18]. Both of these systems utilise a narrow strip of light to recover the depth and move around the object. A faster approach is to project a pattern rather than a thin stripe of light. A range of patterns can be used such as pseudo-random points, however the most common visible pattern used is a series of vertical narrow coded stripes developed by Hall-Holt and Rusinkiewicz in 2001 [19]. The pattern is projected onto the target and captured by a camera. The images are fed back into a computer which uses sophisticated algorithms to process the distortions in the projection and measure the contours of the target. The advantage of using structured light sensors is that the system can capture the whole (visible) scene in one frame unlike laser scanning which has to pass over the target multiple times. Structured light systems are very fast and are able to produce 3D information in real-time [18,20] with accuracies of up to 100 μm.

Visible structured light systems are well-suited to fast, accurate reconstruction of

buildings and complex surfaces but can be harmful if used on humans or animals. Although structured light systems are non-contact and non-invasive, they utilise laser and/or bright DLP projectors which could affect a person's eyes due to the intense light. Structured light can also work when switching the DLP/laser projector to near-infra red or IR projectors. The most popular form of IR structured-light system can be seen in the Microsoft Kinect (Version 1) [21]. The Kinect is classed as an RGB-D sensor because of its ability to stream both colour video and depth information. The Kinect uses a Class 1 (IEC 60825-1:2007-03) infra-red laser projector to project a dot pattern into the environment. A monochrome CMOS camera sensor captures the IR pattern from the environment and computes the resulting 3D information. The Kinect is capable of tracking up to 6 people simultaneously and is able to perform full skeleton tracking of two people in the field of view. This is an extremely popular sensor with gamers and researchers alike. The release of this device to developers [22, 23] sparked a wave of new interest in field of 3D interaction [24–26]. Microsoft released a second version of the Kinect sensor which has improved tracking abilities (now up to 6 skeletons), a 1080p video camera and switched to an active IR camera. The developer communities have been able to choose between OpenNI, Microsoft Kinect SDK and LibFreeNect in terms of programming development kits to use the sensors. The developers behind the OpenNI framework have stopped work on it and now make their own RGB-D sensor called Structure.

Digital cameras are now widespread and often built into robots for use in remote sensing, surveillance, monitoring, film and TV applications. Cameras can be used in a range of configurations to provide the ground operator with the remote view in a safe area. The type of camera rig depends on the payload capacity and intended application of the vehicle. Typically most vehicles will be fitted with a forward facing camera or one with a wide field-of-view lens. In some configurations the vehicle is fitted with a stereo camera rig which is comprised of two identical cameras mounted precisely on a fixed rig. Multiple cameras can be arranged on-board to provide a wide view of the environment. By combing images from these cameras

and using algorithms designed for robotics such as Visual Simultaneous Localisation and Mapping (VSLAM) [27], one can construct a model of the environment and localise within it at the same time.

Other configurations include single camera systems such as a camera with a vertically facing fish-eye lens and mirror. The combination of fish-eye lens and mirrors provides a 360 degree view in the horizontal plane around the sensor. Fish-eye images are extremely distorted and require post-capture rectification before being used. Some 3D tracking solutions use two forward facing fish-eye cameras to view as much of the environment as possible.

2.2.2 Stereoscopic 3D

Stereoscopic 3D or 'binocular' vision uses two cameras with overlapping fields of view to recover depth information. The simplest stereo configuration features a pair of identical cameras which are securely mounted so that the optical axes are parallel to each other with a fixed distance between the centres (known as the baseline). The baseline is a key factor which must be known by the processing algorithms to recover the depth map. A stereo rig can be a single device, such as the Point Grey BumbleBee cameras or StereoLab's ZED 3D camera, which supports up to 2K video resolutions. Alternatively a stereo rig can be custom-made using two identical video cameras. Regardless of the physical configuration, a pair of images must be acquired simultaneously which is especially important if the rig and/or the target is moving. Once calibrated, a rectified stereo pair has the advantage of simpler depth recovery algorithms which are able to run at frame rate (input frame rate varies depending on factors such as bandwidth, sensor size, processing power and capture method). A stereo camera system produces a dense per-pixel depth map offering fast and reliable 3D information.

In order to demonstrate the principles of stereo depth recovery we begin by looking at the simplest case using the pinhole camera model shown in figure 2.1. In the

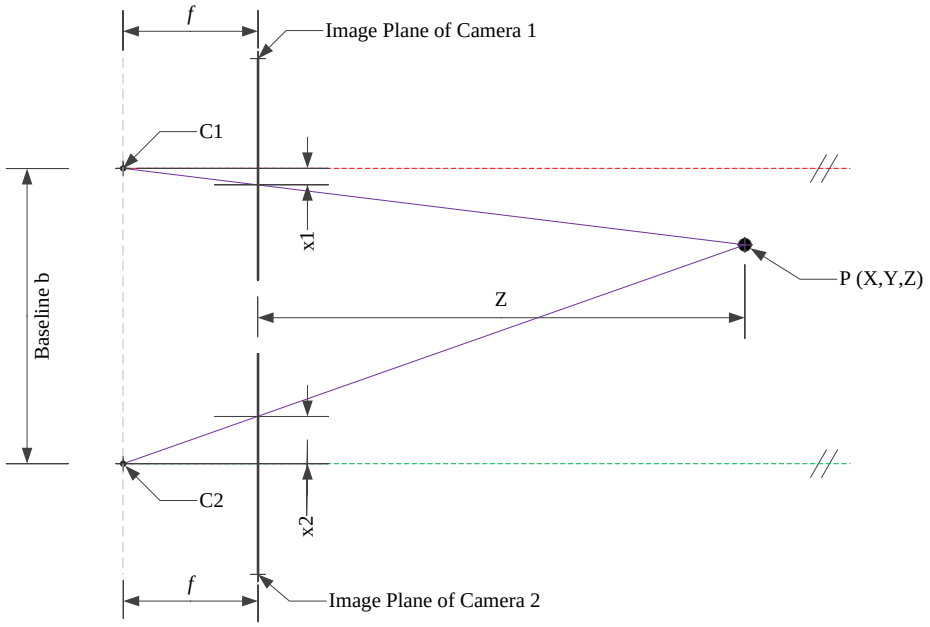


Figure 2.1: Stereo Pinhole Camera Rig - Looking down the camera's y-axis

diagram we can see that the optical centres $C1$ and $C2$ lie behind the image plane. A detailed explanation of the pinhole camera model and its relationship to stereo vision is covered in Chapter 3, however for the purpose of explaining depth from stereo here, it is important to note that the optical centre of a camera does not lie on the image plane. There is an optical ray that passes from P through the optical centre of each image. Point P can be expressed as $(X, Y, Z)^T$ as it is a 3D point in the world coordinate frame. We can map point P onto the image plane of a camera using the formula $(\frac{fX}{Z}, \frac{fY}{Z})$ [28]. Given this relationship we can begin to calculate the distance of the 3D point from the camera system.

The point P intersects the image plane, projecting an image point (x_1, y) in the left image and a point (x_2, y) in the right. If we place the origin of our coordinate system at the aperture of the right camera we can recover the distance between the camera and the real-world point. From figure 2.1 we have two equations, one for the left:

$$\frac{x_1}{f} = \frac{b - X}{Z} \tag{2.1}$$

and the right image

$$-\frac{x_2}{f} = \frac{X}{Z} \tag{2.2}$$

Combining (2.1) and (2.2) together to form a single equation gives:

$$\frac{x_1 - x_2}{f} = \frac{b - X + X}{Z} = -\frac{b}{Z} \tag{2.3}$$

where the minus sign is due to the different direction of the angle. This can be rearranged to give the depth of point P as the distance from the camera, Z, in terms of the disparity measured in the two images:

$$Z = \frac{f\,b}{x_1 - x_2} \tag{2.4}$$

Given a perfectly calibrated system such as the one presented in the pinhole model we can use equation 2.4 to calculate the distance from the centre of the baseline to the 3D point. There are couple of key caveats here: in computer vision, graphics and digital sensors the origin of an image is typically in the top left-hand corner and not at the optical centre therefore the processing algorithm must translate that to the centre of the image before measuring the location of the point. Secondly, the focal length f is normally given in millimetres whereas the image points will be returned in pixel units. The sensor dimensions must be known so that the measured image position is translated into millimetres.

In practice the pinhole camera model is too simple for real-world cameras as it does not account for lenses, distortion or cameras where the optical axes are not perfectly parallel. Additionally, the pinhole model assumes all points seen on the image plane are in focus which may not be the case with a real device. Stereo cameras are often used in robotics to aid navigation, planning and mapping due to the low computational cost.

2.2.3 Monocular 3D

Recovering 3D depth with just one camera opens up image-based 3D sensing to a wider range of applications, as robotic platforms often have strict payload limits but feature a built-in camera. A range of techniques have been developed to tackle the challenge of using only a single camera. One thing all of the techniques have in common is the need for the camera to travel through the environment about the region being reconstructed. In most approaches, two frames are processed at a time as though they were captured at the same time. The frames must have an overlapping view similar to that of a stereoscopic system except with larger baselines and the freedom to rotate about the target / optical axis.

2.2.4 Structure from Motion

Structure from Motion is a monocular stereo technique which is capable of recovering both the environment and the original camera locations. A Structure from Motion system uses several images from a range of viewpoints to recover the 3D environment in two phases: sparse, and then dense, reconstruction. Structure from Motion (SfM) can work with a single camera moving about a target or work with images captured from different cameras at different points of view [29] (or a combination of both).

Bundler [30] is the best known Structure from Motion system and is often the basis for benchmarking and algorithm comparison. Bundler was developed for a Photo Tourism project in 2006 [29] where the aim was to use large collections of photographs taken by tourists of a particular tourist attraction and then reconstruct these into a 3D model [31]. This work inspired the development of Microsoft's PhotoSynth service [11] for creating 3D panoramas and 3D scenes. PhotoSynth and Google's Street View do not produce 3D models and are not examples of Structure from Motion systems.

Bundler can form the central part of a 3D reconstruction work flow. A typical Structure from Motion system is comprised of a series of stages split into three phases, as shown in figure 2.2. The first stage is to detect important features within each of the input frames [32]; typically SIFT is used as the feature detector. SIFT [33] is protected and not freely available outside of academia, so many applications use an open-source implementation, VLFeat [34], which produces similar results to Lowe's implementation without the licensing restrictions. During the feature detection stage, a SfM application will also extract the focal length from the EXIF data contained in image meta-data, as explained further in chapter 3. Once the features have been extracted from each image, the next stage is to match features between each pair of images in the dataset. This stage can take some time to complete as each feature has to be compared with every feature of the other images in the dataset. After the dataset has been prepared for processing, the original images, features and image matches are then passed into a reconstruction system such as Bundler.

The resulting output from Bundler is a 3D point cloud which is correct up to scale. The drawback of a Structure from Motion approach is there in no way to know the scale. If a calibrated target is visible in the reconstruction the output can be scaled correctly. The results from a SfM pipeline are sparse and do not provide fine detail of the target. In order to increase the quality of the resulting model the output needs to be passed through additional processing steps to achieve a dense point cloud. The first step is to send the output through an algorithm such as Clustered Multi-View Stereo, CMVS [35, 36], which clusters nearby points into manageable chunks of data. These smaller clusters are then passed into an algorithm such as PMVS, Patch-based Multi-View Stereo [37] along with the original image dataset. PMVS produces a dense reconstruction from the clustered data; it can be run in parallel across multiple cores, dramatically reducing processing times (which are still long).

The dense point-cloud is significantly more detailed than the sparse one, however it is not immediately presentation ready. The final stages in the pipeline currently require human interaction to tidy up the model as there are often many erroneous matches which need to be deleted to reveal the final results. The final model

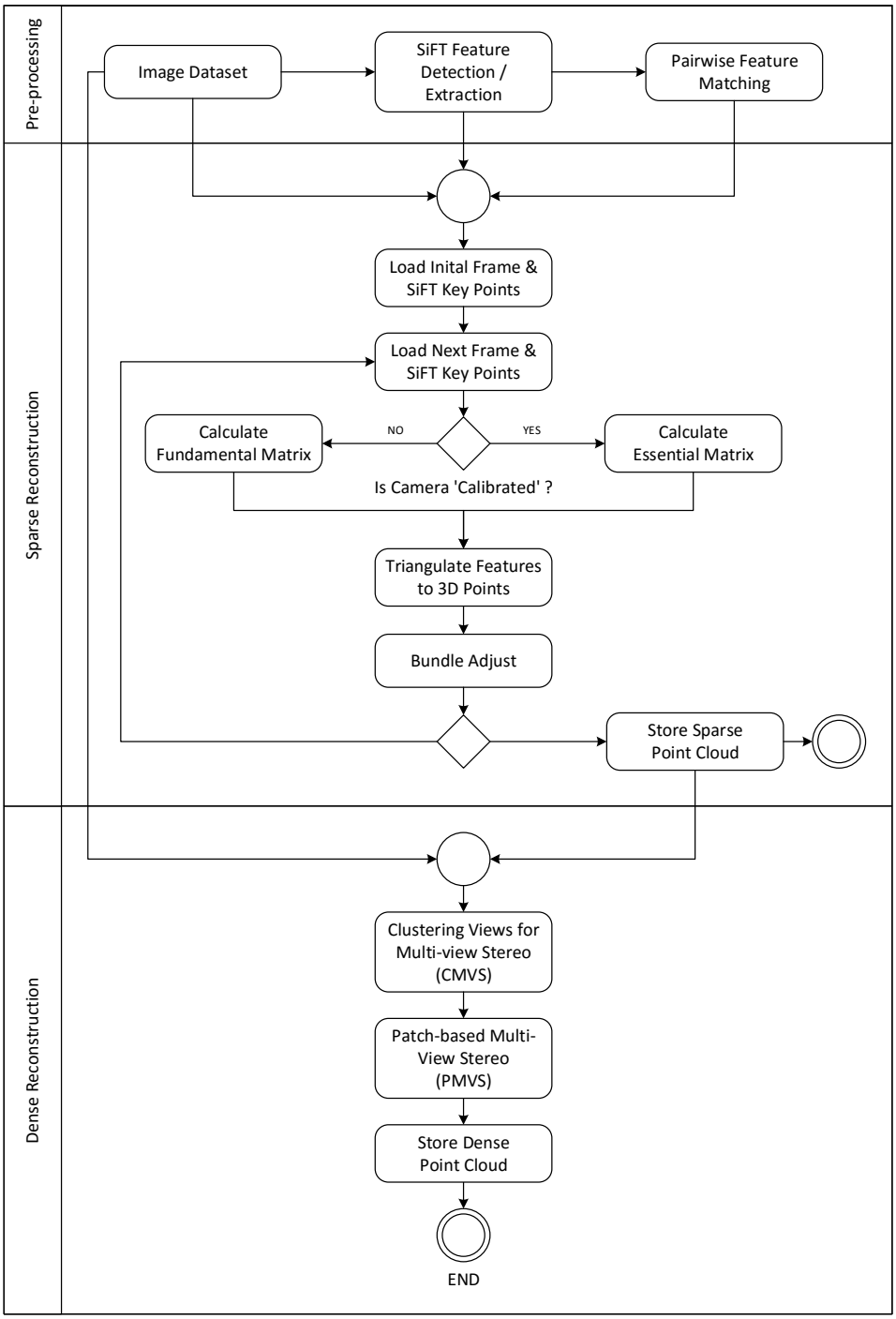


Figure 2.2: Complete Structure from Motion Workflow

provides a clear representation of the scene / target, though it may not be in a convenient form. Software such as MeshLab [38] can be used to process the collection of points into meshed surfaces allowing for the model to be used in other applications more easily. There are a number of triangulation and surface reconstruction algorithms for converting point clouds into meshes, such as Poisson [39] or Ball-pivoting [40]. Once a mesh is created it then needs to be textured using the original images, as the meshing operation removes the point colour information. An alternate approach, CMP-MVS [41], can be used which uses the sparse point cloud and produces a mesh rather than another point cloud thus skipping the manual manipulation stage. In [41] the authors are able to generate high quality surface models from the dense points.

The biggest drawback to Structure from Motion systems is that they require very lengthy processing times. Reconstructions can take several hundred minutes to compute and some tasks can take days, depending on the number of images to be processed and the hardware available. Improvements in General Purpose Graphics Processing Unit (GPGPU) technology has provided a significant impact on many computer vision tasks and especially Structure from Motion due to the increase in cores and speed compared with a CPU. A GPGPU is able to run a vast number of parallel threads with the aid of programming toolkits such as the NVidia CUDA framework. For example, a professional computation graphics card such as a NVidia Quadro K2100M (Mobile workstation GPU) has 576 parallel processing cores with 48 GB/s memory bandwidth to 2Gb of GDDR5 RAM, compared with 8 processing cores on the latest 3GHz Intel processor. Even an OEM gaming-grade graphics card such as a NVidia GeForce GTX55 has 288 parallel cores. A GPU implementation of SIFT [42] improves the performance of the detector a hundredfold. A full HD (1920px by 1080px) frame can take up to 30 seconds to pass through SIFT using a CPU implementation whereas the same image takes approximately 0.03 seconds using a GPU implementation.

VisualSFM [43] is Structure from Motion system which was built to explore the development of GPU-based SIFT [42], multi-core bundle adjustment [44] and gener-

ally further research into SfM. VisualSFM provides a GUI to allow the user to control the reconstruction process and evaluate the stages and tuning parameters to achieve the best reconstruction for a given dataset. Commercial alternatives are also available which provide detailed results and further processing workflows such as 3D PDF exports, Digital Elevation Models (DEM) and built-in camera calibration tools. AgiSoft's PhotoScan is the leading dedicated commercial solution, though leading 3D modelling and design companies such as AutoDesk are beginning to integrate this technology into their own products.

A Structure from Motion framework, Theia, has recently been published [45] which enables researchers to develop new algorithms and techniques without needing to build an entire system from the ground up. Until Theia became available researchers were often forced to either adapt Bundler or write SfM systems from scratch to integrate new functionality, which is what was done in the initial stages of the work described in this thesis.

2.2.5 SLAM: Simultaneous Localisation and Mapping

Simultaneous Localisation and Mapping (SLAM) [46, 47] is an important research area within the field of robotics. SLAM is the umbrella term for a process which maps an unknown environment and simultaneously localises the robot (or sensor/s) within the map. SLAM systems can use a range of sensors, from rotary / shaft encoders in odometers through to complex laser and LiDAR systems. A SLAM system can be categorised into two classes: off-line (batch) and on-line processing. On-line systems are used in robotic navigation due to the hard real-time constraints required to enable path planning. Typically a SLAM algorithm will involve some form of Kalman Filter (KF) or equivalent, with the Extended Kalman Filter (EKF) being most common. In [48] for example, the authors compare the effects of these filters with the addition of Compressed Extended Kalman Filter (CEKF) and Unscented Kalman filters (UKF). The findings showed that CEKF approaches produced better results when working with outdoor environments, as demonstrated by [49]. The

study evaluates SLAM performance on moderate notebook computers with standard CPUs.

2D laser range finders are one of the most common sensors used on mobile robots for SLAM as they provide high accuracy and measure over longer ranges than standard sensors such as sonar. Camera images may not be commonly associated with SLAM inputs; however this poses an interesting field of study, joining image processing and robotics to research Visual SLAM. In 2005, the first significant paper on Visual SLAM or VSLAM [27] was published, which demonstrated that a robot can use a camera instead of range-finding devices. SLAM and VSLAM can be used to generate either 2D or 3D maps of the environment, and the accuracy of the most recent ORB-SLAM [50] purportedly approaches that of SfM.

2.2.6 Visual SLAM Approaches

Dense Tracking and Mapping, DTAM [51], is a real-time camera tracking and reconstruction system which differs from Structure from Motion as it does not involve feature extraction. DTAM is able to reach real-time performance using a GPU and highly parallel algorithms. In [51] the authors refer to the problem of real-time structure from motion as "monocular SLAM". The authors discuss how DTAM works directly with a dense model, thus skipping all of the steps needed to identify and convert feature points into sparse point clouds and then recalculate those data points into dense point clouds. The DTAM algorithm works with the whole image to track the camera's motion against the developing dense model. The algorithm relies on a short baseline created by images featuring significant overlaps obtained by a short frame interval from a live camera. The authors discuss the difficulties of handling lighting changes that can affect the reconstruction process. DTAM certainly provides an interesting area for investigation as its authors have been able to achieve high-quality reconstruction in real-time.

Parallel Tracking And Mapping, PTAM [52] is another real-time ($\approx 30\,\text{Hz}$) monocu-

lar 3D camera tracking system designed primarily for use with augmented reality (AR). PTAM is also used in various robotics projects as the system tracks the environment, outputting camera positions. PTAM has recently been extended to work with multiple maps, PTAMM, so that the system can change between events based on the map in view. PTAMM is primarily aimed at AR games development.

Most of the techniques discussed so far rely on detecting and extracting features in input frame(s) with the aid of a feature detector such as SIFT [33,34] or SURF [53]. Although GPU implementations of SIFT and SURF exist, these detectors are often computationally expensive and can be slow. Another type of feature detector known as binary feature detectors has emerged, offering increased speed with similar accuracies. Binary detectors such as BRIEF [54, 55], BRISK [56] and ORB [57] are seeing increasing use in the field due to the lower requirements and ability to run in real-time.

ORB-SLAM [50] is a monocular SLAM solution which is able to compute in real-time both the camera trajectory and a sparse 3D reconstruction of the scene, thanks to the speed of the ORB feature detector. ORB-SLAM is able to close large loops and has been demonstrated producing sparse reconstructions of car-based video sequences in a wide variety of environments.

Structure from Motion systems and similar Visual SLAM systems rely on key-points produced by feature detectors to build sparse models of the environment. LSD-SLAM [58] is direct monocular SLAM technique. LSD-SLAM skips the feature detection / extraction and matching stages and works with the full image for both tracking and mapping. The result is a semi-dense reconstruction which can be used by navigation systems. Stereo LSD-SLAM [59] is an extension to enable the visual SLAM technique to work with dual cameras, enabling LSD-SLAM to run in real-time on a standard CPU. Both LSD-SLAM and Stereo LSD-SLAM work with rectified camera images so that any radial distortion and other lens effects are removed from the cameras. LSD-SLAM has been further extended to work directly with omni-directional cameras [60] where the system uses the raw input from an omni-directional camera

source to produce semi-dense 3D reconstructions. The results of [60] show that the omni LSD-SLAM is able to produce reliable results.

RatSLAM [61] is a different form of SLAM system to the visual systems discussed so far. RatSLAM's algorithms are inspired from the neutral processes within the brain. The advantage with RatSLAM is that the system is able to preform large scale mapping tasks with low-cost and low-grade cameras such as a typical computer web camera [62]. OpenRatSLAM [63] is the latest development in RatSLAM which was written from the ground up to provide a clean and open-source system for robotics researchers. OpenRatSLAM is comprised of four main components: a Pose Cell Network which manages the response to the odometry and local view components; Local View Cells which are responsible for determining whether a view is new or familiar; Experience Map Node which manages the graph building and path planning; and the Visual Odometry node which is used if the system is using just images as the source of odometry. OpenRatSLAM is designed to work with open frameworks including OpenCV and ROS (discussed in a later chapter).

A recent survey paper [64] identifies ORB-SLAM, LSD-SLAM, L-SLAM [65] and OpenRatSLAM as the most recent developments of SLAM for mobile platforms between 2013 and 2015. The authors of [64] evaluated these techniques against a benchmark produced by the researchers at TUM [66]: the results of the evaluation show that Visual SLAM is a complex and challenging task which often produces significant errors and requires post-processing to ameliorate them.

2.3 Visual Localisation and Mapping within Robotics

Humans are able to explore their environment without bumping into objects using just their vision. If a person is visually impaired then the brain is able to use other senses to help gauge the distance between them and surrounding objects. In the robotics field, engineers have added 3D cameras to robots to enable them to sense the world in 3D at a much lower cost than traditional ranging sensors. The robot

control software combines the data coming from the visual systems with an existing model to acquire an accurate map of the local environment.

VSLAM has been extended to address the problem of 3D texture mapping [67]. In this paper the authors discuss the difficulty with working with the six degrees of freedom of UAV odometry (a UAV is free to move horizontally and vertically as well as pitch, roll and yaw in any of the three axes). The authors highlight that VSLAM is not dense enough to form a texture map and overcome this by using calibrated stereo cameras and a feature detector to find key points; but the key is to create a uniform search grid in the event that few features are detected. Interestingly, the authors point out that "it is impossible to have dense reconstructed 3D points with SIFT/SURF only due to the real-time requirement" [67]. They have been able to work in real time using stereo cameras with the estimated positions from VSLAM combined with the SIFT or SURF features obtained during VSLAM which are used as the input of an inverse observation model. The author of the paper cites a previous work when referring to the "observation model". In [68], Nemra refers to the observation model as a way of describing the transformations from the real world coordinate system back through the image plane and then to the camera coordinate system. The resulting model from the work of [67] is a rough textured map similar to a digital surface model (DSM). There is a trade-off between the density of the recovered points and the accuracy of the reconstructed map.

2.4 Co-operative Robotics

Co-operative mobile robotics is a relatively new and rapidly expanding area of robotics which covers the challenges of working with multiple robots, in multiple teams. Multi-robot systems provide a new dimension to the field of robotics by expanding the application in areas where a single robot would be incapable of completing a task in a timely or costly fashion. The term 'co-operative robotics' is also used by industrial robotics manufacturers to mean robots which work with or alongside hu-

mans. The Robotics Industries Association of North America (RIA) focuses heavily on the safe integration between robots and operators. The principles of these robots are more motivated towards early studies into robotic pick and place devices which work alongside humans in industrial production facilities. These types of robots have specific applications in repetitive tasks such as vehicle manufacture. However for the purpose of this thesis, we are using the term 'cooperative robotics' in the more literal sense, as defined by the Oxford English dictionary to explore the relatively new area of multi-robot cooperation, also known as Multi-Robot Systems (MRS):

Definition 2.4.1. Co-operative — Having the quality or function of co-operating; working together or with others to the same end; of or pertaining to co-operation. – *Oxford English Dictionary*

The field of study into early multi-robot systems began in the late 1980s and early 1990s and was previously referred to as 'distributed robotics'. The early work focuses on three core areas: reconfigurable robot systems, known as cellular robotics [69–71], multi-robot motion planning [72, 73], and architectures for multi-robot cooperation cite724605OhkawaCooperation98, 772580FukudaMARS99. The field has since changed to include multiple forms of team-based approaches. Distributed and team-based robotics has a significant overlap with the field of Artificial Intelligence especially in the area of cellular systems. An editorial survey published in 2002 [74] identifies seven core topic areas within the domain of multi-robot systems covering: inspiration, communication, architectures, localisation and mapping, object transport, motion coordination, and reconfigurable systems.

In many applications, a single platform may not be sufficient to efficiently execute the task, especially in missions such as search and rescue or environmental monitoring, where a large environment needs to be mapped and surveyed. A single vehicle would not have enough resources or power to complete the mission, requiring complex planning and multiple data capture runs. Grocholsky *et al.* demonstrate a collaborative framework which uses multiple UAVs and UGVs in order to survey a

large area [75]: the framework uses the strengths of the UAVs to survey large areas and calls upon ground vehicles to provide accurate localisation and surveillance. The whole system is controlled from a central ground control station (GCS) which coordinates the robots and generates "information-driven" control points. The centralised controller issues commands to direct the vehicles to key targets to inspect the scene, based on the information generated from the air vehicles. The paper discusses the challenges in localising the ground vehicles based on the aerial information. The Robotic Operating System, ROS, is based on this principle of centralised data management via a master node. Centralising the data through a master seems like a logical solution, especially when attempting to synchronise data across multiple platforms. However, the centralised publishing leads to high network traffic. The large bandwidth requirement severely affects the performance of wireless inter-robot communications, often leading to control issues and data loss as detailed later in chapter 6.

Efficient communication is crucial in multi-robot teams to ensure that the task is completed efficiently and safely. An IEEE communications survey paper [76] identifies four types of communication. Type 1 communications, such as queries, can tolerate network and/or interrupt delays. Whereas 'Type 2' and 'Type 3' messages have hard, real-time demands as these messages encompass emergency states (Type 2) and high priority trajectory messaging such as collision avoidance (Type 3). Willke *et al.* define Type 4 messages as the mission-related messages such as swarming, global path planning etc. which can tolerate network delays, even though they are core to the mission. The survey focuses on inter-vehicle communications with a view to providing autonomous vehicles (cars, lorries) with a universal communications framework.

Situational awareness has a key impact on the decisions that one makes in the event of an emergency. The ability to understand the extent of a problem enables a commander to make effective decisions despite the natural instinct to tackle the emergency directly in front of them. UAVs and UGVs expands the level of situational awareness [77]. Phan *et al.* discuss the development of a cooperative platform for

wildfire detection and fighting. The paper lists the operational capabilities of each of the available robots and assigns them a rôle based on that information. The control system uses an airship as the main coordinator as it is able to stay on scene for longer than a helicopter-based drone such as a quadrocopter. The control system dispatches other devices to maintain the fire boundary. 'Time-on-scene' is a key term used to define the effectiveness of a particular vehicle. UAVs such as quadrocopters are limited to 10–15 minutes of flight time, whereas a fixed wing aircraft may be able to maintain a time on scene of greater than 30 minutes (depending on fuel type and payload). The flight time is offset by the capability to hover over a target. Fixed wing aircraft need to circle around a target location unlike a helicopter-based aircraft which can remain static directly over a target. Each vehicle has a set of strengths which, when combined with other vehicles, produces an effective team.

The United States Defence Advanced Research Projects Agency, DARPA, are known for their large-scale robotics challenges. The most famous of these being the DARPA Grand Challenge [78] held in November 2007. Another key DARPA challenge was the Software for Distributed Robotics programme which was developed to demonstrate the technical challenges of large robotic teams with a focus on mapping large unknown indoor environments. In [79] a team of 80 robots was created to navigate and explore a 600 m^2 indoor space. The authors created two classes of robot, a small number of highly capable 'leader' robots equipped with a scanning laser range-finder, camera and powerful on-board CPU. The second class of robots are a large number of simple platforms with just a moderate camera, microphone and low power CPU. The robots communicate over a standard ad-hoc wireless network and use vertical fiducials for robot identification. The vertical identifiers are akin to a bar-code wrapped around a pole which provide omni-directional readability from any of the ground vehicles. In order to expand the teams to introduce air vehicles additional fiducials would need to be added to the upper surface such as those explored later in this thesis.

Co-operative Visual Simultaneous Localisation and Mapping, C-VSLAM [80] expands on the work of 3D mapping from UAVs [67] and develops a new central

architecture for filtering the sensor data from multiple robots. C-VSLAM uses multiple aircraft to produce a 3D reconstruction of a target environment. The central node uses a non-linear H∞ (NH∞) Filter, which replaces the Extended Kalman Filter (EKF) used in [27]. Simulation results show that the approach of using two UAVs produces more accurate results than a single UAV, and considerably more accurate results than the internal navigation systems alone.

The approaches discussed so far all have similar robot-to-robot control structures, relying on a centralised control node for commanding the teams. This method works efficiently in typically hierarchical structures such as those in the armed forces; however biological influences have shown that there are alternate ways to create robotic teams to complete team-wide objectives.

Understanding the world around us is not just the end-goal for 3D reconstruction and remote sensing systems, it is often one of the largest inspirations for designing and developing new approaches. Nature has evolved multiple modes of movement, and robotics designers and engineers have sought to try to replicate these natural movements. Humanoid robots have been the subject of science fiction since 1927 [81] and perhaps more notably in 1942 from Asimov's three laws of robotics. Humanoid robotics are invaluable in expanding the understanding of the complexity of human movement and control. A recent land-based humanoid robot, SURENA III, has been developed by the University of Tehran and features 31 degrees of freedom (DOF) [82]. The robot is fully capable of walking over sloped surfaces, stairs and is stable across uneven surfaces. SURENA III may seem like something directly from science fiction with its tall stance of 1.9m and developing artificial intelligence modules. Perhaps the latest development in humanoid robotics comes from the Stanford Robotics Lab, namely OceanOne [83]. This is an underwater humanoid robot designed specifically for deep ocean exploration. OceanOne provides a pilot with haptic feedback, allowing for complex underwater operations that aid divers and also operating on tasks that are too dangerous for human divers to attempt.

Behaviour-based robotics and related areas of artificial intelligence have seen sig-

nificant inspiration from natural phenomena such as ant colonies, flocks of birds and fish. These events form the basis of the studies into swarm behaviours, which demonstrate how large groups of simple individuals interact to form complex behaviours and become collectively intelligent systems. The first example of a computer simulated swarm was published in 1987 by Craig Reynolds [84]. Reynolds' work, Boids, presented the concept of simple individual agents which use just three simple rules, cohesion, separation, and alignment, to form a computer animation of a flock of birds moving along a common path without programming the path of each bird explicitly. The resulting animations behave in a similar fashion to real birds. Since the introduction of swarm-like behaviours, the ant-colony problem has become an interesting research area for both those starting out in the field of artificial intelligence and researchers alike.

In a survey of swarm robotics approaches, Şahin outlines three key motivators for swarm robotics which are desirable for multi-robot systems [85]. The first motivation is robustness, a sensible requirement during any system design; however in the domain of swarm robotics, Şahin refers to the robust nature of the group: in the event that an individual is defective or dies, the group is able to continue towards the global goal. This makes individual robots dispensable. A key factor which separates swarm systems from other multi-robot systems is the control structure. Swarms use decentralised coordination architectures to develop complex behaviours. The swarm does not rely on one node to coordinate the group, which improves the team's robustness and reduces the chances of a task failure. The second motivation is flexibility within the group, allowing the swarm to generate different solutions based on the task. In all cases the swarm's behaviour should be scalable regardless of the number of agents in the swarm. These principles are confirmed in a more recent survey by Zhu and Tang, covering both biological and artificial inspirations of swarm robotics [86].

2.5 Concluding Remarks

This chapter began by exploring early 3D modelling systems and their move towards computer aided manufacturing and engineering. Starting with the pre-CAD origins and moving through to the cutting edge in 3D design and modelling systems, it is clear to see through the literature that 3D models have an important role to play in a range of disciplines. A wide range of sensors are available for recovering 3D information about the immediate environment, ranging from active methods which involve sending out a signal into the environment and measuring its response, through to passive sensing which observes the environment. This thesis looks specifically at modelling the real world in 3D point clouds through the use of computer vision, namely Structure from Motion, a monocular passive 3D sensing technique. Structure from Motion is clearly a well established 3D reconstruction technique which enables 3D models to be reconstructed using just a single camera (and its known focal length).

The literature review then focused on the relevant robotics aspects which would be necessary in developing a robotic capture platform later in the thesis. The review looks at individual operations such as SLAM/VSLAM before exploring using multiple robots as part of a team to collaboratively achieve a task such as mapping. The literature explores the use of visual techniques for localisation and mapping within robotics, which have tight performance constraints, yet do not produce the detail of the much slower Structure from Motion techniques. This poses the question, could a collaborative robotic system be used to capture a detailed 3D reconstruction of an unknown environment using vision-only techniques?

The next chapter explores the underlying theory and mathematics of Structure from Motion and implements a test solution to verify that the mathematics works correctly on a range of synthetic virtual camera views. Subsequent chapters then utilise these fundamental principles to understand the practicalities of obtaining 3D reconstructions using the given pipeline.

CHAPTER 3

3D Reconstruction from Multiple Robotic Sensors

The central subject of this research is 3D reconstruction from images captured from robotic platforms. Consequently the mathematics that underlies imaging and structure from motion is of critical importance, and this chapter presents that mathematics. Sections 3.1 and 3.2 describe the geometry of image formation within a camera by starting with a constrained formulation, and then successively removes those constraints. Section 3.3 presents the mathematics of multi-view geometry in some detail. A simple experiment is presented in 3.4 to confirm the validity of the mathematics and Section 3.5 draws conclusions.

3.1 Camera Geometry

The purpose of the following exposition is to produce an equation that describes how a real world point is mapped onto a pixel in an image. We shall start with the simplest possible model and successively generalise it.

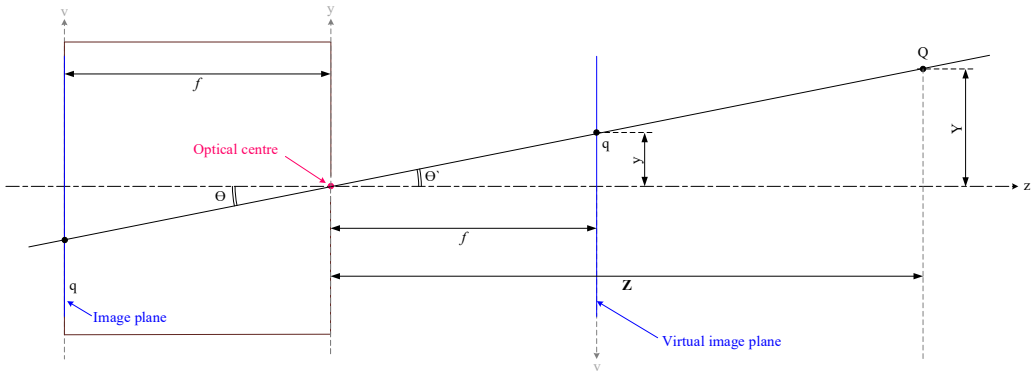


Figure 3.1: 2D diagram of the relationship between a 3D point and the image planes

3.1.1 The Pinhole Camera

We shall first analyse the simplest possible imaging configuration and model, a single pinhole camera. Figure 3.1 shows a schematic side elevation of a pinhole camera. Capitalised letters denote a 3D world coordinate while the corresponding lower case letter represents the corresponding coordinate inside the camera. A feature Q outside the camera is projected to the point q on the back focal plane of the camera. Points Q and q are connected by a straight line passing through the optical centre, the pinhole. This a consequence of the fact that light rays travel in straight lines. Consequently, triangles OAQ and Oaq are similar. It will be clear that the image in the camera is inverted, so many authors rotate the camera through 180° around the pinhole to produce a virtual focal plane in front of the camera, as illustrated in figure 3.2.

In order to calculate the real-world coordinates of Q we need to know the position of the corresponding point on the virtual image plane. Using the similar triangles OAQ and Oaq mentioned above:

$$\tan\theta = \frac{Q}{Z} = \frac{q}{z} \tag{3.1}$$

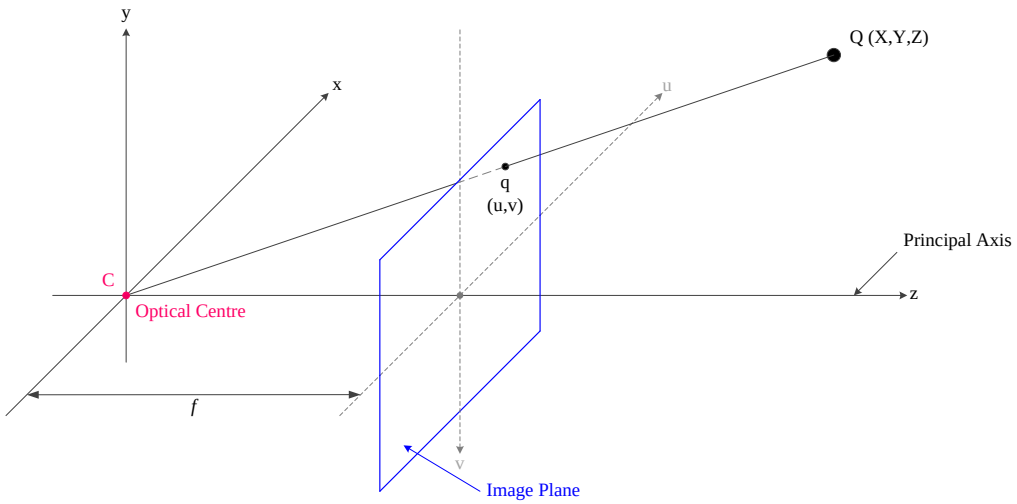


Figure 3.2: 3D diagram of a point, Q, seen in 3D space by a single camera, C

This relationship applies in both the y direction, as shown in the digram, and the x direction (out of the page). If we identify (u, v) as the 2D image coordinates corresponding to a feature q on the image plane we can write:

$$\begin{aligned} u &= \frac{fX}{Z} \\ v &= \frac{fY}{Z} \end{aligned} \tag{3.2}$$

Writing the pair of equations as homogeneous coordinates [87] we can represent the projection of the point Q at $(X, Y, Z)^T$ on $(u, v)^T$ as:

$$\begin{pmatrix} u \\ v \\ z \\ w \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.3}$$

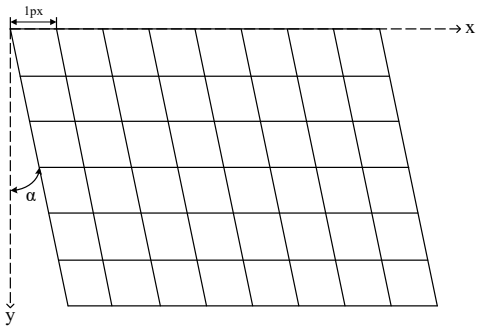


Figure 3.3: Skewed Pixel Array

3.1.2 Describing Real Cameras — The Intrinsic Camera Matrix

A pinhole camera is not usable in practice because exposure times are too long. A lens that replaces the pinhole allows for much shorter exposures but introduces some additional difficulties. Firstly, objects in the real world must be focused correctly onto the back focal plane by the lens. Lenses introduces aberrations [88] either through their manufacture or through the virtue of having finite apertures. We shall not cover these here, the reader is referred to a standard text [89].

The pinhole camera model assumes that the notional pixels on the back focal plane are square. In the case of CCD sensors, measuring image coordinates in pixels can introduce unequal scale factors in the horizontal and vertical directions which need to be accommodated. The effect of a non-rectangular sensing array is rarely discussed in the modern literature so we consider that effect here. This applies both to the individual sensors not being square and to them having a different pitch in x and y.

From figure 3.3 we can define a skew variable, S, as being:

$$S = (\tan \alpha)\frac{f}{P_y} \tag{3.4}$$

where α is the angle of the pixel, P_y its width, and f the focal length [90]. It is rare for skewed pixels to occur in modern systems, however it can arise in the event of capturing an image of an image, for example when taking an image of a printed photograph: skew will occur if the focal plane and the image are not parallel [87, 91].

Taking into account all of these effects, we can encapsulate them into a single matrix

$$K = \begin{bmatrix} f_u & S & t_u \\ 0 & f_v & t_v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

where f_u and f_v are the focal lengths in the u and v image directions and t_u and t_v describe any offset of the optical axis from the middle of the image. This matrix is normally known as the *camera calibration matrix* because these parameters are normally found through calibration. This matrix is also referred to as the *intrinsic matrix*.

In principle, full calibration is needed only once per device. However occasions such as temperature change, zooming, or changing the lens would prompt a re-calibration. There is a variety of methods for calibrating a camera [92–94]. The most common calibration technique uses a chessboard pattern as a calibration target, which has the advantage of providing highly contrasting intersecting squares. The calibration algorithm detects each of these intersections and measures the distance between the squares. The user enters the true dimensions of a square so that the algorithm can calculate the focal length and distortion values. The intersections between the squares provide a series of points that should form parallel lines in both the horizontal and vertical directions. If there is any warping in these planes then the lens has a radial distortion which can be calculated and stored to correct subsequent frames. A full calibration produces an additional matrix which can be used to remove the distortion in the frames before further processing.

3.2 The Camera In 3D Space — The Extrinsic Matrix

Knowing the intrinsic parameters of the camera provides only part of the information required for 3D reconstruction: it is also necessary to know where the camera lies in the world coordinate system. The position of the origin of the camera in the world coordinate system can always be written in the form:

$$C = \begin{bmatrix} R & t \end{bmatrix} \tag{3.6}$$

The rotation matrix, R, is a 3×3 matrix which can be thought of as being formed by the product of three rotation matrices around the axes (3.7, 3.8, 3.9):

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.7}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{3.8}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

The translation component of 3.6 is given by the t sub-matrix. C is commonly known as the *extrinsic camera matrix*.

Combining the intrinsic and extrinsic camera matrices, we can write:

$$C = K \begin{bmatrix} R & t \end{bmatrix} \tag{3.10}$$

which is known as the *camera matrix*.

The camera matrix describes the projection a point in 3D space onto a point in the image with respect to a reference coordinate system such as an object in the scene or the world coordinate system, so that the entire process can be written succinctly as:

$$q = K \begin{bmatrix} R & t \end{bmatrix} Q \tag{3.11}$$

3.3 Multi-View Geometry

The previous sections have shown that the properties of the camera itself can be modelled and the effect of placing the camera in 3D space described mathematically. This section goes on to present how these form the basis of reconstructing 3D structure from multiple images. Multi-view geometry underlies structure form motion, which this thesis explores.

Although a single frame can be used to extract a 3D model from an image given a specific set of conditions and known parameters [95,96], a single view is usually not enough to be used to reconstruct an environment. Instead, either multiple cameras are required or a single (monocular) camera must be moved through a static scene.

At each point an image is captured, the camera will have moved through a translation and/or rotation; these can be described by either the *essential* or *fundamental* matrices, discussed below. If the calibration matrix is available for the camera, the essential matrix can be used to estimate the correspondence between points on two images, allowing for 'keypoint' reconstruction. If the calibration matrix is unavailable (e.g. when using a live source) then there are two options: the first is to assume (or impose) the focal information and still work with an essential matrix, while the second is to employ the fundamental matrix instead.

The following paragraphs describe the essential matrix and how it encapsulates the correspondence between features in two images. Solutions that allow positions to be determined are then described. The more general case where the fundamental

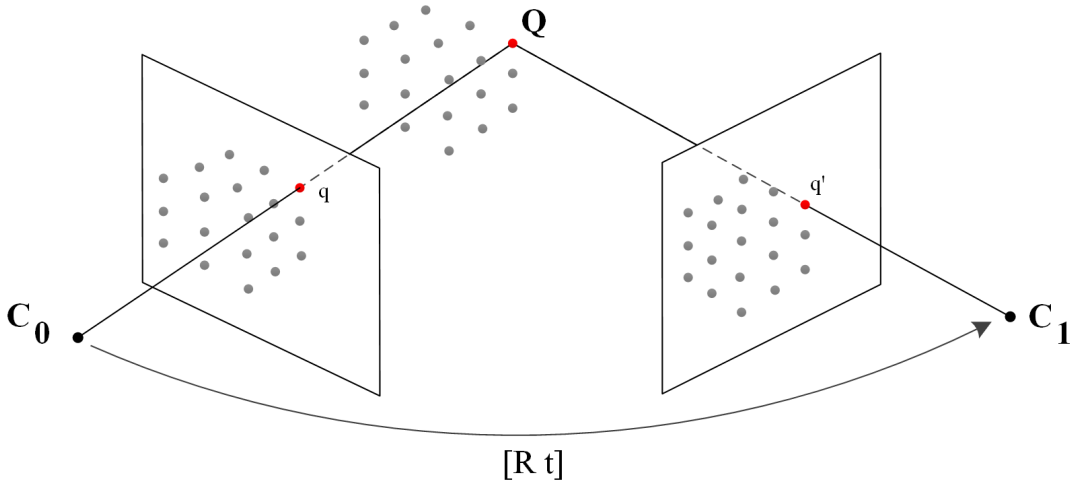


Figure 3.4: A series of 3D points as seen by two views

matrix has to be used is then discussed.

3.3.1 The Essential Matrix

The essential matrix is a 3×3 matrix (in non-homogeneous coordinates) which describes the correspondences between two related images. In figure 3.4, the motion from C_0 to C_1 can be described by the essential matrix as it encodes both the rotation and translation from one camera position to another. The essential matrix can be found from just the image points and thus enables reconstruction using triangulation without prior knowledge about the pose of the camera. The relationship between the two images can be modelled by equation 3.12 which introduces the essential matrix as E and a 3D point Q as seen from two views.

$$C_1^T E C_0 = 0 \tag{3.12}$$

where the essential matrix is written as:

$$E = R\Big[t\Big]_x \tag{3.13}$$

Here, R is the rotation matrix between the two camera centres and $\left[t\right]_x$ is the skew-symmetric matrix form of the vector t. The proof of the translation vector cross product is explained in detail in appendix A.1.

Following [97], the first step in calculating the essential matrix is to expand the constraint defined by equation 3.12 as shown below:

$$\begin{pmatrix} x_1 & y_1 & 1 \end{pmatrix} \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} = 0 \tag{3.14}$$

Multiplying the 3×3 matrix and C_0 together gives:

$$\begin{pmatrix} x_1 & y_1 & 1 \end{pmatrix} \begin{bmatrix} E_{11}x_0 & E_{12}y_0 & E_{13} \\ E_{21}x_0 & E_{22}y_0 & E_{23} \\ E_{31}x_0 & E_{32}y_0 & E_{33} \end{bmatrix} = 0 \tag{3.15}$$

Multiplying the two remaining matrices from equation 3.15 yields the following linear equation:

$$E_{11}x_1x_0+E_{12}x_1y_0+E_{13}x_1+E_{21}y_1x_0+E_{22}y_1y_0+E_{23}y_1+E_{31}x_0+E_{32}y_0+E_{33} = 0 \tag{3.16}$$

This equation is particularly useful for demonstrating the link between the unknowns of E against the matched keypoints Q_1, Q_2, Q_3. This can also be written

explicitly as:

$$\begin{pmatrix} E_{11} \\ E_{12} \\ E_{13} \\ E_{21} \\ E_{22} \\ E_{23} \\ E_{31} \\ E_{32} \\ E_{33} \end{pmatrix} \begin{pmatrix} x_1 x_0 \\ x_1 y_0 \\ x_1 \\ y_1 x_0 \\ y_1 y_0 \\ y_1 \\ x_0 \\ y_0 \\ 1 \end{pmatrix} = 0 \tag{3.17}$$

or

$$E.q = 0 \tag{3.18}$$

Equations 3.17 and 3.18 employ a single matched point in the images C_0 and C_1. A pair of real images will typically produce hundreds of matched points. Each pair of corresponding point matches generates a new vector q. The individual vectors $q_1, q_2, \cdots, q_n$ form the columns of a new matrix, Q, which can be used to solve for E.

$$E^T Q = 0 \tag{3.19}$$

As feature-based algorithms produce hundreds or even thousands of point matches, the challenge is then to pick the smallest number of keypoints that lead to a reliable answer. There are several ways of doing this.

For many years the minimal approach has been to use eight point-matches [98]. This algorithm does not work well on real image data as the points are subject to noise. Hence an improved version known as the normalised eight-point algorithm was developed [99, 100], which employs pre- and post-conditioning stages. More recently, it has been shown that the relative poses between cameras can be calculated with as few as five points [101, 102]. However even more recent work has shown that the perspective three point or P3P approach [103] is both faster and more stable than the five or eight-point algorithms.

When more than eight point matches are available there are two standard approaches for estimating the essential matrix. The most common approach is to use RANSAC, Random Sample Consensus [104], which is a robust model estimation algorithm. With RANSAC, one repeatedly calculates the essential matrix using (say) the eight-point algorithm until the matrix with the smallest residual error given the input pool is found. RANSAC has been used extensively throughout computer vision since its publication in 1981. However, many alternatives have been developed to address the run times associated with running through several hundred calculations. One example is ARRSAC, Adaptive Real-Time Consensus [105], which runs quickly and is suitable for use in real-time applications.

The second common approach is to use single value decomposition (SVD) to calculate the linear least squares best solution to the over-determined set of equations.

The fundamental matrix [106] is a 3×3 matrix of rank 2 which has the same role as the essential matrix, namely to describes the correspondence between the points in two views, but differs in that does not rely on any camera information. The fundamental matrix works without calibrated cameras to produce a projective reconstruction whereas the essential matrix is able to produce only a Euclidean reconstruction; hence the fundamental matrix is often explored in more detail in the literature [28, 97].

3.3.2 Epipolar Lines

Once the correspondence between two frames has been calculated, we move on to the task of recovering 3D points. Figure 3.5 shows a single 3D point Q being captured by two views. In this figure, it can be seen that there is a virtual ray joining the origin of both cameras. If the origin of the second camera can be 'seen' from the initial view, one could plot a virtual point in the frame corresponding to the optical centre of the other camera; this is known as the epipole. The epipole is often not visible within the bounds of the frame, unlike the example shown in figure

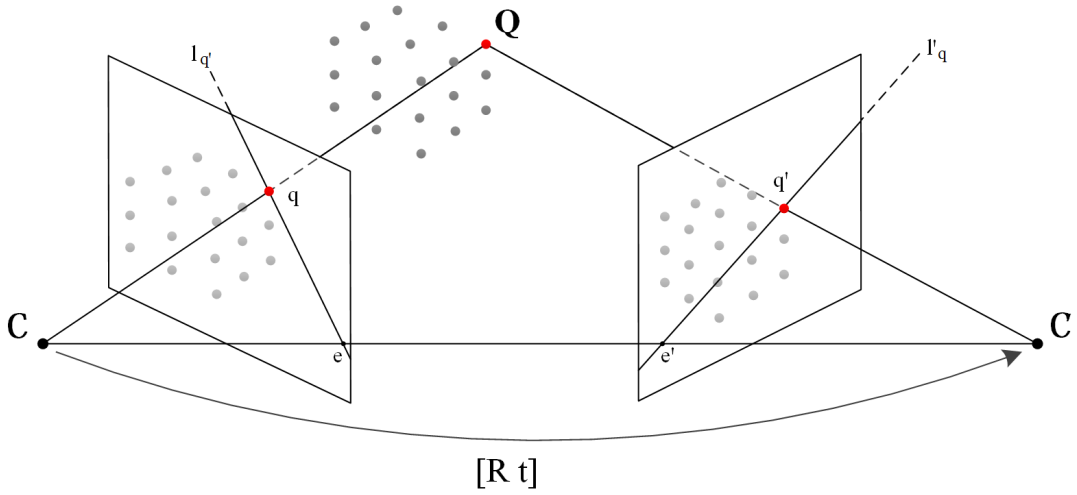


Figure 3.5: The epipolar lines for a key point as seen by two cameras

3.5, as it depends on the shape of the transformation from the initial view to the second.

Figure 3.5 also introduces a visualisation of the epipolar line. If we join the observed point q with the new epipole e we form the epipolar line for the first view, $l_{q'}$. The epipolar line enables one to look at the second view and narrow the search space when looking for the matching point. If the essential/fundamental matrix is correct (and the feature matcher has provided a correct matching keypoint) then the point Q as seen in the second view must lie on the corresponding epipolar line, l'_q. This is known as the epipolar constraint which is modelled as:

$$q'^T E q = 0 \tag{3.20}$$

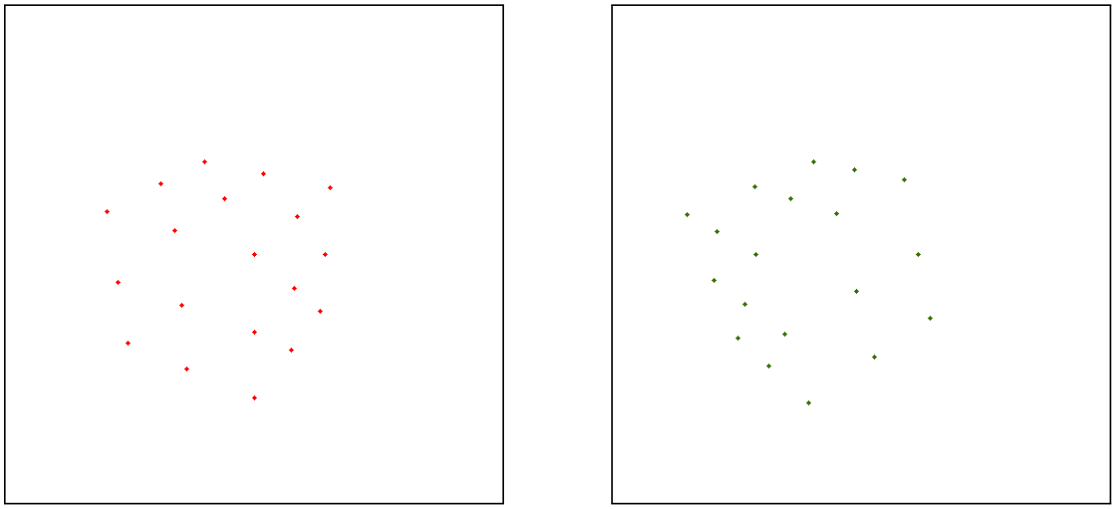
The epipolar constraint is an important relationship as it is used during the RANSAC process to verify that there exists a correspondence between the two images and evaluate the error of the estimated matrices as a function of the distance from the epipolar line, as well as reducing the search space for finding corresponding features during the later stages of dense reconstruction.

Joining the two epipoles e and e' forms the baseline that is used to perform triangulation to recover the depth of a given point. It can be seen that the 3D points of C, C' and Q form a plane. Therefore the corresponding point of q in image 2 must lie on the plane which intersects the image plane.

3.4 Experimental Confirmation

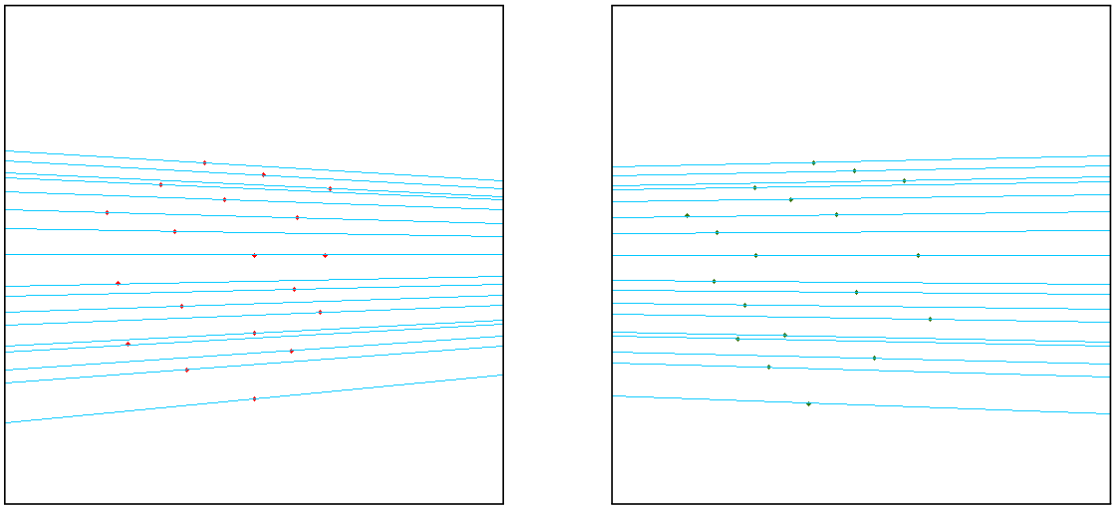
A virtual cube formed of 19 keypoints has been created to demonstrate this principle on a more realistic scene compared to the single point shown in figure 3.5. The cube resembles the intersections of a Rubix cube as seen in figure 3.6. Two virtual views of the cube have been generated by defining an arbitrary motion between the two virtual cameras and creating an image for each view.

Figure 3.7 shows the respective epipolar lines drawn on both views, based on the estimated essential matrix generated using SVD from all of the given points. The lines draw in closer towards the middle of the two views as there was a large rotation and only a small translation between the two viewpoints. It can be seen that if the viewpoints were rectified the lines would form a horizontal line across both images. The epipolar lines are generated based on a point in the opposite image. The quality of the essential matrix is measured based on the residual error obtained from equation 3.20. If the essential estimation is correct then every point will be on the respective epipolar line. Real-world images are rarely this clean due a range of factors such as lens distortions, feature detection/matching errors, noisy sensors and compression. Figure 3.8 shows the reconstructed 3D points based on the two input frames shown in figure 3.6.



(a) Left View (b) Right View

Figure 3.6: Two synthetic views of a virtual cube



(a) Left View with the epipolar lines generated from the right view

(b) Right View with the epipolar lines generated from the left view

Figure 3.7: The epipolar lines for the two views of the cube

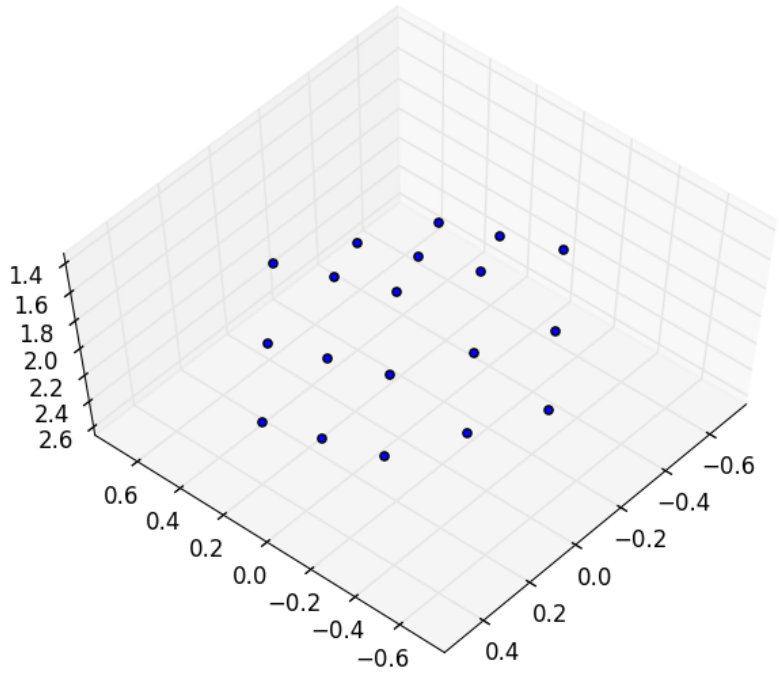


Figure 3.8: The output reconstruction based on the estimated essential matrix

3.4.1 Distance-Based Error Metric

Utilising the epipolar constraint as a metric for RANSAC produced unreliable results. Potential solutions for the essential matrix would produce wildly inaccurate models yet present a low residual error. A more reliable metric of calculating the residual error is to measure the distance of the observed point from the epipolar line. The first stage is to generate two points from the epipolar line:

$$A = \begin{bmatrix} -1 \\ \frac{-l_2-(l_0\times -1)}{l_1} \\ 1 \end{bmatrix} \qquad B = \begin{bmatrix} -1 \\ \frac{-l_2-l_0}{l_1} \\ 1 \end{bmatrix} \tag{3.21}$$

where l is the epipolar line. Pre-multiplying these two new matrices by the camera calibration matrix produces two points which can be used to produce a line in the image. Given these two points and the location of a keypoint (also pre-multiplied by the calibration matrix), we can calculate the distance from the respective epipolar line:

$$d = \frac{|(y_2-y_1)x_0-(x_2-x_1)y_0+x_2y_1-y_2x_1|}{\sqrt{(y_2-y_1)^2+(x_2-x_1)^2}} \tag{3.22}$$

where $A=(x_1,y_1)$, $B=(x_2,y_2)$ and the keypoint is (x_0,y_0).

3.4.2 Experimental Results

Four models have been prepared to test the reconstruction theories: a book-like edge of a cube containing just 15 keypoints, a Rubix-like cube containing 19 keypoints, a larger 6×6 cube and finally a larger 9×9 cube containing 217 points. These models test the stability and estimation accuracy of using SVD and RANSAC in the 3D reconstruction.

Table 3.1 shows that SVD is stable, producing consistent results on every run. For the purpose of these tests RANSAC was limited to a maximum of 1,000 iterations,

Run	Edge	Cube 1	Cube 2	Cube 3
1	1.6127	0.4873	0.4174	1.4242
2	1.6127	0.4873	0.4174	1.4242
3	1.6127	0.4873	0.4174	1.4242
4	1.6127	0.4873	0.4174	1.4242
5	1.6127	0.4873	0.4174	1.4242
Mean	1.6127	0.4873	0.4174	1.4242

Table 3.1: Mean Error Using SVD

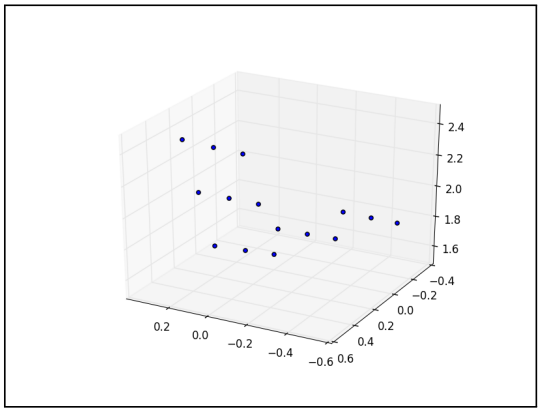
Run	Edge	Cube 1	Cube 2	Cube 3
1	0.4106	0.2926	0.1519	0.4397
2	0.9860	0.2882	0.1122	0.7184
3	0.9950	0.2795	0.1389	0.7516
4	1.0242	0.2996	0.1282	0.7776
5	0.9860	0.2869	0.1337	0.7099
Mean	0.8804	0.2894	0.1330	0.6794

Table 3.2: Mean Error Using RANSAC

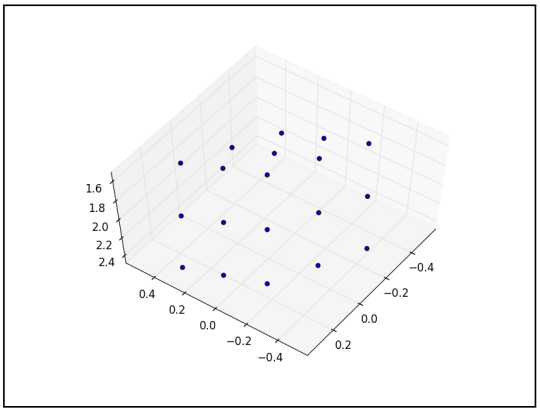
which it completes in just under two seconds on a mobile workstation. The test suite is programmed in Python. RANSAC produces more accurate results where the epipolar lines are closer to the original keypoints. The results shown in Table 3.2 show that RANSAC is not as consistent as SVD, yet it is able to produce a better estimation of the essential matrix. In all cases, the error is measured using the distance of the point from its respective epipolar line using equation 3.22. Figure 3.9 shows the resulting models generated using RANSAC.

3.5 Concluding Remarks

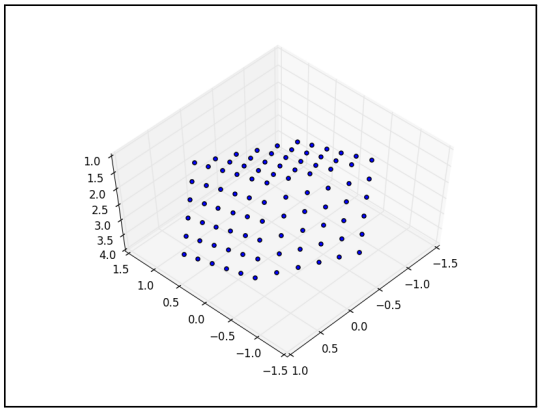
This chapter has presented the core mathematics necessary to produce a projective reconstruction from a pair of images. The chapter started by considering a single pinhole camera, which was then generalised into a calibration matrix suitable for a generic camera. An additional camera (viewpoint) was then introduced to explore 3D reconstruction given multiple points observed by two cameras. The calibration and essential matrices were evaluated using synthetic images to produce a model of a virtual cube. The results demonstrate that RANSAC produces more accurate results compared to SVD, though at the sacrifice of speed. RANSAC is more useful when real-world data points are used and noise is present. These steps form the core of structure from motion, which is the primary reconstruction technique used throughout this thesis.



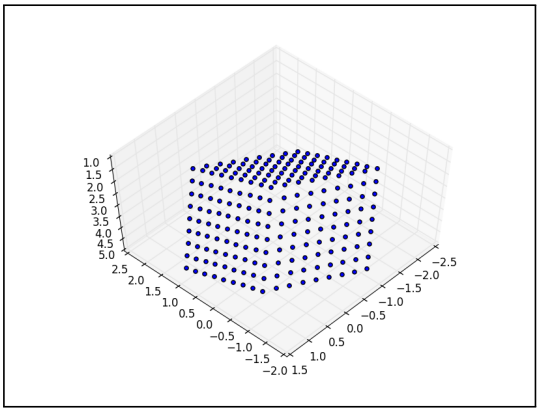
(a) Cube edge (15 Keypoints)



(b) Cube 1: 3×3 (19 Keypoints)



(c) Cube 2: 6×6 (91 Keypoints)



(d) Cube 3: 9×9 (217 Keypoints)

Figure 3.9: The Resulting Models Using RANSAC

Having developed code to verify that this SfM approach to model-building works, there are two paths that the research can take. One is to develop the remaining stages of the SfM pipeline and then explore algorithmic improvements and processing requirements; but this area is now fairly well established, to the extent that there are both free (open-source) and commercial offerings that purport to provide this functionality. The second path, which is the one that has been taken, is to use these tools and instead explore the practicalities of producing detailed and accurate reconstructions using real robots — and in particular heterogeneous systems that combine aerial and ground-based robots.

The next chapter presents a series of 3D reconstructions of increasing complexity, to investigate the main characteristics of the tools and to identify their shortcomings. Subsequent chapters explore the robotics aspects, ranging from having an aerial robot control ground robots to delivering high-resolution video frames suitable for presenting to the Structure from Motion toolchain.

CHAPTER 4

Assessing the Effectiveness of Structure from Motion Reconstructions

The experiments presented in this chapter are designed to ascertain the effectiveness, reliability, and performance of 3D reconstruction using SfM. They start by performing SfM reconstructions from a series of datasets of increasing difficulty. Rather than use the author's own code, which could be criticised for being one person's effort and lacking extensive verification by others, some widely-used implementations are employed, spanning open-source and commercial offerings. The rationale for doing this is that the thesis is exploring not a proof-of-concept implementation but rather trying to ascertain whether the best combination of hardware and software available today is able to yield accurate, dense models from images obtained from robots.

The leading commercial application for photogrammetry, AgiSoft Photoscan, uses Structure from Motion as its primary reconstruction method. It is widely used by archæologists and geologists and has also been used in forensic work. There are a handful of open-sourced non-commercial applications which are born out of research teams working on photogrammetry, most notably Bundler/CMVS/PMVS, VisualSFM and, more recently, Theia. Bundler is widely regarded as the benchmark

system for comparison when evaluating new algorithms and systems; however, VisualSFM has been used as the primary benchmarking tool here due the completeness of the application. This is because Bundler is focused on achieving a sparse model and leaves the latter stages to other tools, whereas VisualSFM is able to work with the complete workflow from raw images through to a dense model.

These are followed by a consideration of practical issues that affect the quality of reconstruction, the aim being to establish criteria that govern the way a robot capturing imagery intended for SfM should move.

4.1 Structure from Motion Experiments

4.1.1 Small-Scale Structure from Motion

The goal is produce a remote reconstruction system which is capable of providing detailed models and information to observers. A key element of reconnaissance work is often hidden in the detail of the reconstruction, so these experiments start with a fairly simple scene and work up to more complex and challenging tasks.

4.1.1.1 Simplistic Scene

After seeing the potential of Structure from Motion in a demonstration from the Oxford Archæology Digital Labs [107], a simple test was conceived using a digital camera and VisualSFM to reconstruct three books stacked with a pair of passive 3D glasses placed on top as seen in figure 4.1a. The visual texture on the covers of the three books and the non-repetitive nature of the shape of the glasses means that these components of the scene should be straightforward to reconstruct, though the floor may not.

Some 100 frames were captured by circling the target in increasing circles. The

(a) Simple scene (b) VisualSFM Sparse Model (c) VisualSFM Dense Model

Figure 4.1: Simplistic reconstruction target through to dense model

sparse result shows a fairly good estimate for the camera positions for each of the input frames. The camera positions can be visually inspected to determine whether the frames align with the estimated locations, however without external tracking these are recorded as just estimated positions. The sparse model (shown in figure 4.1b) shows a clearly visible hole where the footprint of the books would be in the floor. There is also some indication of the spines of the books and the edge of the glasses. These results become clearly visible when the dense reconstruction completes, revealing the full extent of the detailed model. To verify the accuracy of the model a point-cloud viewer was used to measure the size of the three books in the dense reconstruction. Table 4.1 shows the raw readings taken from the point cloud using CloudCompare [108]. The default unit of measure is in metres. Structure from Motion is said to be accurate up to an unknown scale. Table 4.2 compares the average width, A, length, B, and height, C, of each of the books with the real world measurements. The scale factor is fairly consistent across the range of measurements with a scale factor of 12.35 ± 0.75. The relative ratios in terms of the dimensions of the books demonstrate that a Structure from Motion based system will produce a model which is sufficiently accurate to capture further measurements from given that there is a known object in the scene to recover the scale factor.

Table 4.1: Book Reconstruction Measurements

Edge	Book 1 (m)	Book 2 (m)	Book 3 (m)
A_1	1.607522	1.975177	2.338155
A_2	1.611943	1.859780	2.406694
A_3	1.623558	1.921581	2.313572
A_4	1.614642	1.912326	2.411890
$A_{AVG.}$	1.614416	1.917216	2.367578
B_1	1.600952	2.915238	2.991217
B_2	1.609473	2.909721	2.985545
B_3	1.596283	2.908647	2.957458
B_4	1.593353	2.915244	3.010761
$B_{AVG.}$	1.600015	2.912213	2.986245
C_1	0.876692	0.627565	0.783491
C_2	0.881654	0.635173	0.746161
C_3	0.875103	0.661765	0.758704
C_4	0.871614	0.724290	0.753010
$C_{AVG.}$	0.876266	0.662198	0.760342

Table 4.2: Reconstruction vs Real Measurements

Book	Edge	Measured (cm)	Real (cm)	Scale
1	A	161.4	13.3	12.1
	B	160.0	13.3	12.0
	C	87.6	7.0	12.5
2	A	191.7	15.2	12.6
	B	291.2	23.1	12.6
	C	66.2	5.5	12.0
3	A	236.8	19.5	12.1
	B	298.6	24.4	12.2
	C	76.0	5.8	13.1

4.1.1.2 Cluttered Scene

A key element of evaluating the practical use of Structure from Motion is its ability to recover complex spaces and structures, to help an operator to understand how they are arranged. A collection of robot parts was piled together and the scene captured using a Fuji digital camera mounted on top of a Pioneer 3DX research robot. The robot was set to follow a pre-defined route plotted into ROS, the Robotic Operating System, which controls the robot. The camera was mounted so that lens was 90° to the direction of travel and to capture autonomously frames at a fixed time interval with autofocus enabled, producing a constant stream of images. The estimated camera positions can be seen in figure 4.2.

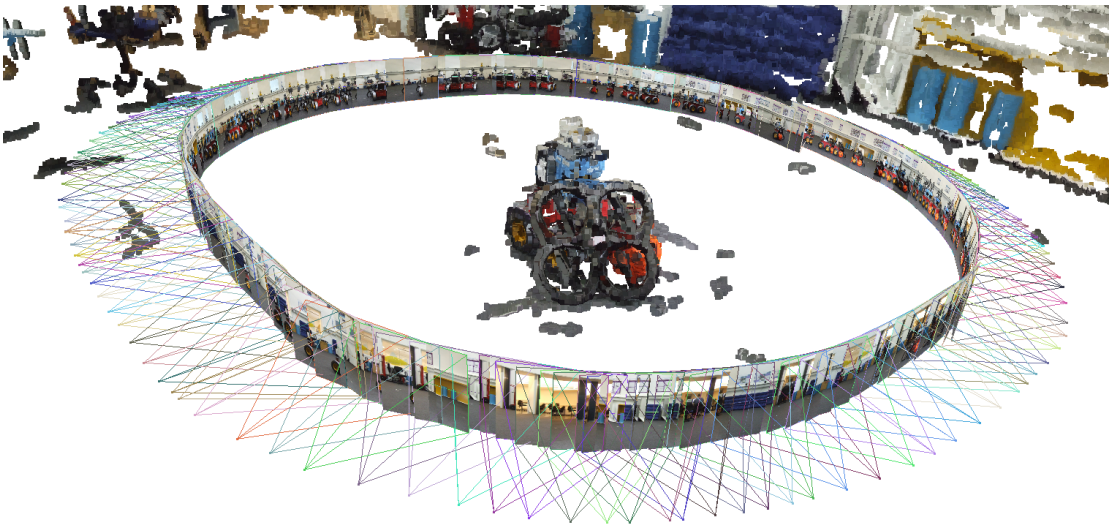


Figure 4.2: Estimated camera positions

The robot moved in a slow, steady motion around the room producing 150 images. The combination of the robot's speed and camera timing produced a smooth arc and evenly spaced camera angles with small baselines, the objective being to ensure over 60% overlap between frames, including translation and some rotation.

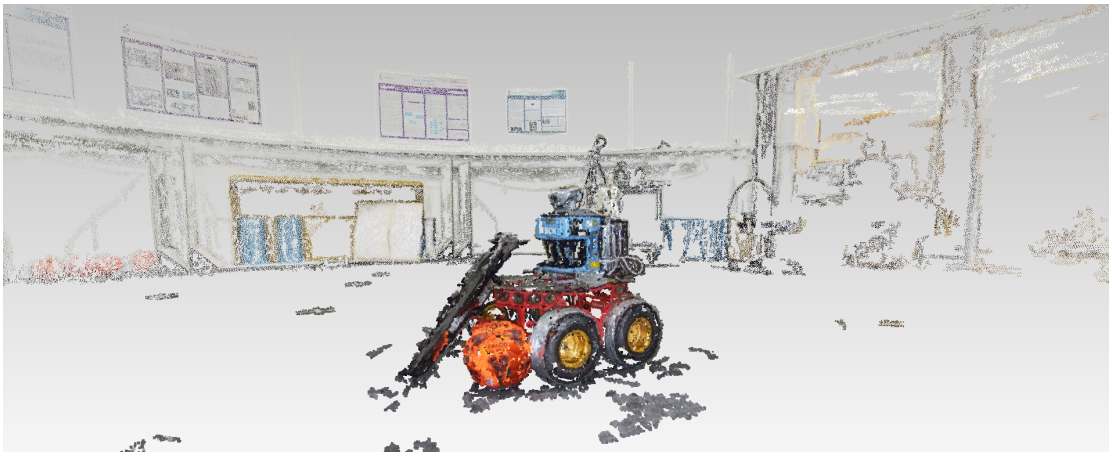


Figure 4.3: A 3D Reconstruction of a Complex Scene with VisualSFM

The feature-based basis of Structure from Motion as a reconstruction method is evident from the large holes in the resulting point-clouds: it has failed to recover and model *any* flat surfaces in the scene. This can be seen most clearly in figure 4.3 where the floor and walls produce few features and thus are not reconstructed. Some of the information lost during a SfM reconstruction can be intuitively identified, such as the circular nature of the room and the flat surfaces. Figure 4.4 shows a closer inspection of the resulting model compared with an original photograph at a similar point in its trajectory. It can be seen that the key features of the AR drone's indoor hull is present, with most of the orange football also being visible. There is however some loss of detail on the robot chassis, with most of the red frame missing.

The dense results produced by VisualSFM demonstrate that using a small robot to autonomously capture a scene produces a sufficient level of visual information to reconstruct the environment for use by a human operator, though not for producing a complete model.

Running this image set through Agisoft PhotoScan demonstrates the potential of using Structure from Motion's dense modelling: the models from Photoscan missed some of the finer details from the scene yet produced more of the surrounding environment. The propellers on the aircraft are missing from both the sparse and

(a) Cluttered Robotics Scene (b) VisualSFM Dense Model (c) PhotoScan Dense Model

Figure 4.4: Close inspection of a complex reconstruction target

dense reconstructions from PhotoScan, yet are visible in the models produced by VisualSFM as seen in figure 4.4.

4.1.1.3 The Fenwick Treasure

In 2014, a significant archæological find was made in Colchester, Essex during the ground works phase of a redevelopment of the Williams & Griffin store in the High Street. The Colchester Archæological Trust discovered a collection of fine gold and silver Roman jewellery dating back to the early stages of the Boudican Revolt in AD 61. The discovery has been named the Fenwick Treasure and, after restoration, can now be seen in Colchester Castle. More information about the Fenwick treasure can be found on-line on the trust's website [109].

The Colchester Archæological Trust kindly invited us over to view the jewellery and capture images of the find before the individual pieces were taken apart and cleaned. The Treasure was removed from the site on the day it was discovered (to avoid theft) by cutting out the entire piece of earth and securing it in a plastic container, which can be seen in the photos shown in figure 4.5a.

The target was photographed using a Canon 5D MII DSLR Camera equipped with a standard 24–105 mm F/4 Canon lens. The images were captured in RAW format,

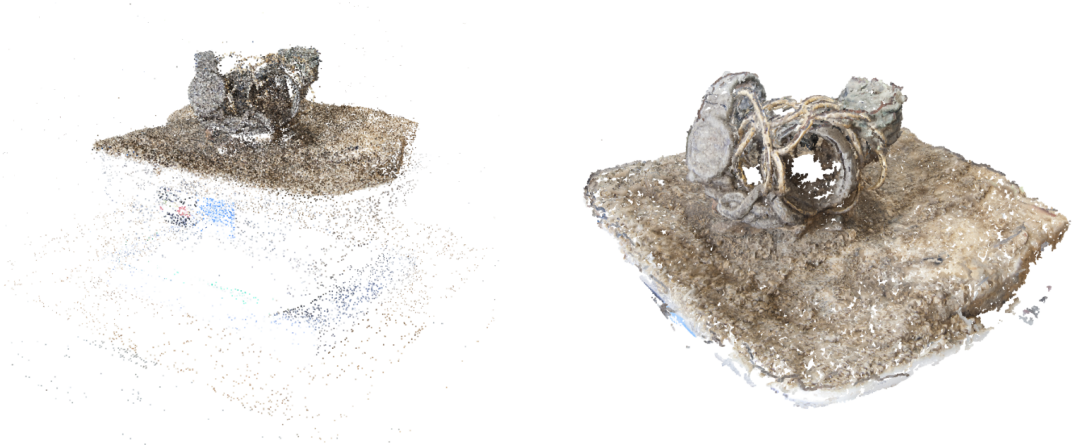
(a) A wide view of the target (b) A closer view (c) An overhead view

Figure 4.5: Photos of the Fenwick Treasure

preserving the high resolution capture with a frame size of 5616×3744 pixels. The camera was manually focused for every shot whilst moving around the container in circular passes and over the top. A total of 224 photographs were captured with 195 usable for processing (29 were discarded due to blur or exposure issues). The frames were batch-processed from the RAW camera format into JPEG images using the Canon LUT profile to produce usable images.

As an initial test, the images from a single pass around the target were collated and passed into VisualSFM running on a high-performance mobile workstation. A subset of 73 images proved to be enough in order to produce a detailed dense reconstruction in just under two hours. The resulting model, shown in figure 4.6b, has been manually cleaned up using MeshLab to remove erroneous points floating around the main object, leaving a model with over 2 million 3D points.

The full collection of valid images were then processed by VisualSFM and Photoscan Professional running on a computation server (24 cores with 96Gb RAM) to compare the effect of using multiple processors with cutting-edge software. The sparse model produced by VisualSFM is more detailed than that created by Photoscan, which can be seen by comparing the models shown in figures 4.6a and 4.7a respectively. The next step is to compare the dense models which would be used in a post mission documentation stage (offline to the robots) by testing the next part of reconstruction workflow. Both VisualSFM and Photoscan produce detailed reconstructions covering the key elements of the artefacts. PhotoScan was tweaked to



(a) VisualSFM Sparse Model (b) VisualSFM Dense Model

Figure 4.6: 3D Reconstruction of the Fenwick Treasure using VisualSFM

produce a dense model at its highest settings, yielding outstanding results. Measurements prove that the final model is quite accurate, and the amount of detail which can be seen in figure 4.7 is good: the fine structure of the bracelets, including the hinges, are clearly visible. The software also recreated the container, which was surprising as it is not visible in the sparse point cloud and was completely lost in the dense model created by VisualSFM.

The point cloud from PhotoScan comprises 58,853,562 coloured 3D data points and required 68Gb of RAM to support the dense modelling phase of the reconstruction pipeline, taking a total of 15 days to complete. It is clear that although the resulting models contain fine detail, a processing time measured in hours let alone days is completely unsuitable for robotics.

VisualSFM is designed to use GPU power in all stages where available, whereas PhotoScan is able to use the GPU only during the latter stages of dense point cloud estimation. Due to the fact that the processing server is a CPU-only device, both VisualSFM and Photoscan were equally limited. If a GPU is present on the workstation, VisualSFM is able to produce results significantly faster than PhotoScan: a typical frame will take approximately 4 seconds to find the SIFT features on a CPU

(a) PhotoScan Sparse Model (b) PhotoScan Dense Model (c) Dense Model Close-up

Figure 4.7: 3D Reconstruction of the Fenwick Treasure using PhotoScan

but only 0.64 seconds on a GPU [42]. Processing speed is an important consideration when evaluating Structure from Motion for robotics, as the robots themselves typically employ low-end processors due to power constraints. In order to cope with low-powered robot processors, the images need to be transferred to a remote machine for processing, with any consequent motion updates then being sent to the robot's navigation and planning modules. Realistically, a dense model is probably not be necessary for navigation, providing that the sparse model gives sufficient structure to enable the robot to sense and avoid obstacles. However, a delay of two hours is clearly unacceptable for robotic navigation, so other visual localisation techniques such as SLAM have to be explored, leaving Structure from Motion to provide a solution for preservation, documentation and analysis.

4.1.2 Large-Scale Structure from Motion

The experiments discussed to date were small scale tests, deliberately carried out indoors in tightly-controlled environments. A real-world robot will typically be faced with tougher outdoor environments and scenes featuring entire buildings or collections of structures. In the next series of experiments, the focus is on datasets exceeding 250 images and complex structures.

Figure 4.8: Square 2, University of Essex

4.1.2.1 University of Essex: Square 2

The University of Essex's Colchester campus proved to be a useful test location for 3D reconstruction experiments due the range of architectural elements featured around the campus. The campus is divided into five main squares surrounded by buildings. Square 2 has an array of visually challenging elements such as repeating textures, flat surfaces and a complex seating area in the centre of the square (with repetitive elements), as can be seen in figure 4.8. For the next experiment photographs of the square were captured by hand with a Fuji FinePix S5600 digital camera. A photographer looped around the main seating structure to form one pass around the square. The aim was to slowly move around by a "side-step" at a time to achieve a relatively large overlap between frames. The camera was kept facing inward towards the central seating structure, capturing a total of 292 images.

Although the images were captured in sequence, the reconstruction systems were left in their default configurations to evaluate the worst case scenario whereby the images could be out of sequence. The tool-chains first pass the images through SIFT feature extraction before matching. The matching process rebuilds the sequence in which the frames were captured before moving on to depth calculation and then building the sparse point cloud. VisualSFM, Bundler and Theia all follow this workflow; however PhotoScan groups all of these stages of the reconstruction into one

step called image alignment.

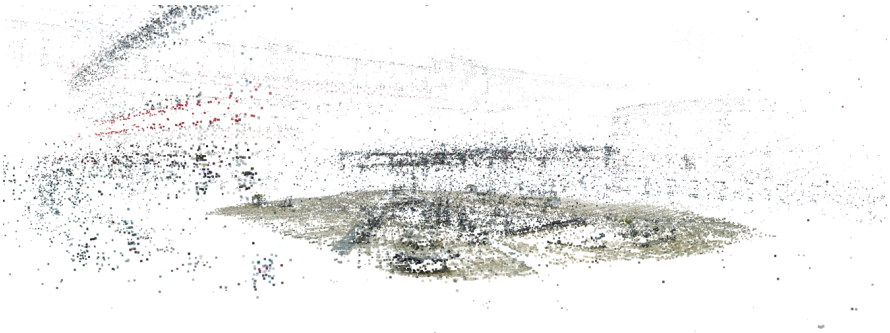
The Square 2 dataset has been used on a range of processing machines from CPU-only servers, through gaming desktops, and entry-level workstations up to a high performance mobile workstation. The results presented here are from a Dell T3600 workstation and a Dell Precision M4800 mobile workstation. The T3600 workstation is configured with a single hyper-threaded quad-core Xeon processor, 8Gb of RAM and two NVidia Quadro 600 graphics cards. The mobile workstation is configured with an Intel i7 processor, 16Gb of RAM and a high performance Nvidia Quadro K2100M graphics card.

VisualSFM produces a complete sparse reconstruction in 13,086 seconds on the T3600. When repeated on the mobile workstation the total time from input to sparse model takes just over an hour (3,682 seconds), proving that the GPU has an positive impact on the speed of processing. PhotoScan is unable to use the GPU until the dense cloud stage of the workflow therefore the package is limited by the number of CPU cores. Photoscan is further restricted as the package dedicates a CPU core per GPU to control the GPU processing, thus further increasing the camera alignment processing time. The image alignment phase (input to sparse model) for the same dataset took four hours to generate a sparse model.

The most notable difference in the results between Photoscan and VisualSFM is that the sparse point cloud produced by VisualSFM contains more information about the environment than that from Photoscan. The round building located off to the right side of the square (the rightmost building behind the seating structure) is completely missing in the Photoscan reconstructions and the majority of the surrounding buildings are incomplete. However, the dense model produced by Photoscan is generally able to cope better with flat surfaces, which is most notable in the paving slabs and benches.

Square 2 presents a series of challenges due to the architecture featuring several repeating sections. The 3D reconstruction algorithms use feature detectors to group frames together to find pairs of images for pair-wise matching. Repeating textures

can trip up feature detectors which results in large gaps in the resulting models and misaligned camera locations. The buildings that surround the square are formed of evenly spaced narrow vertical windows. The main seating area also features repeating vertical pillars. The results shown in figures 4.10 and 4.11 show the results from VisualSFM and PhotoScan respectively. VisualSFM outperforms PhotoScan in dense modelling with this scene. Several areas of the surrounding building are completely missing in the PhotoScan reconstruction of figure 4.11. Although PhotoScan produces a cleaner output than VisualSFM, this is at the expense of model detail and missing physical data.



(a) VisualSFM

(b) Agisoft PhotosScan

Figure 4.9: Sparse Reconstruction of 'Square 2'

Figure 4.10: Square 2 (VisualSFM) Dense Reconstruction

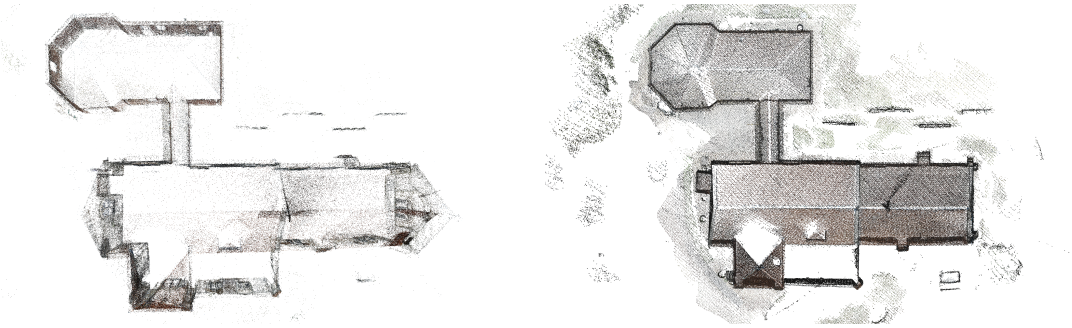
Figure 4.11: Square 2 (PhotoScan) Dense Reconstruction

4.1.2.2 Elmstead Church

Following the Robot Arena and the Square 2 reconstructions, a more complex experiment was conducted to evaluate how the algorithms scale up with a much larger image dataset captured using a mix of cameras. The principle here is to observe how the system behaves if two different robots capture imagery. One camera was the same Fuji model as in previous experiments and the second was intended to be the on-board camera of a Parrot AR Drone. Unfortunately, the weather was too windy to fly the quadrotor, so a different make of digital camera, a Nikon D300, was used to provide the system with high resolution images from a ground position in addition to the Fuji camera. In total, 934 photographs were captured of the church in Elmstead, Essex. The photos from both cameras were fed into VisualSFM for processing. The total time to go from raw images to a sparse reconstruction took 4 days, with GPU support where possible.

Figure 4.12: Sparse reconstruction of a Church in Elmsted Market, Colchester, Essex.

Dense reconstruction attempts have failed to complete in VisualSFM: there appears to be more point data than it can cope with in the CMVS/PMVS pipelines. The VisualSFM process sat at maximum CPU and RAM usage for over 2 weeks before being abandoned. Figure 4.12 shows the sparse point-cloud being displayed from a distance in MeshLab. The lack of a dense model is not really a disadvantage in this



(a) Overhead: Sparse (b) Overhead: (Ultra) Dense

Figure 4.13: Top-down View Comparing Sparse vs Dense Footprints

case because the sparse model of the church produces good detail of the building, with distinct edges when viewed from above as can be seen in figure 4.13. This is a particularly good result because all the photographs were captured from ground level. The church floor plan shows the corridor layout, pillars and windows without any photographs taken of the interior.

The dataset was also processed with PhotoScan on a server which had 24 cores with 96Gb RAM, allowing for considerably more data points. The configuration of PhotoScan was set to its default of 'high' quality dense modelling, given that VisualSFM failed to produce a dense model of the church. PhotoScan successfully reconstructed the dense model and was restarted with the 'ultra' settings enabled. The results of the initial dense model produces 96 million 3D points which can be seen in figure 4.14. The 'ultra' dense model comprises 331.7 million 3D points shown in figure 4.15, taking over 3 weeks (recall, the Fenwick treasure of figure 4.7b required 15 days to calculate the ultra dense model).

4.1.3 SfM Discussions

These experiments have shown that Structure from Motion is capable of producing highly detailed models given that the camera is moved through a controlled mo-

Figure 4.14: Dense model of the Elmsted Church (96 Million RGB 3D points)

Figure 4.15: Ultra Dense model of the Elmsted Church (331.7 Million RGB 3D points)

tion. The resulting measurements obtained from the simplistic scene in figure 4.1 demonstrate that the dense models are accurate if a known object is available to recover the scale factor. These experiments have shown that the Structure from Motion algorithms can cope with mixed cameras generating one model. All of the images must be sharp, well focused, detailed images. A robotics system would need to ensure that images are not captured close together as this increases the processing time without adding 'new' data. In some cases sparse point clouds are sufficient in providing detailed outlines, whereas the dense models produce much clearer models at the cost of time. Large datasets take significantly longer to process. We shall return to this in chapter 5.

4.2 Camera Orientation

The experiments discussed so far in this chapter highlight the potential of Structure from Motion. However, the reconstructions presented above show that the SfM algorithms are sensitive not only to the number of images but also to the capture method. The orientation and motion of the camera are key to the output. If the camera is moved in an incorrect fashion then the reconstruction algorithms will struggle to build a single 3D model of the environment. The camera must move through distinct translation between frames which must not be along the optical axis of the camera, e.g. walking (driving) forward. Moving the camera along the optical axis causes key points to move outside of the frame and therefore location information is lost. Conversely, each image needs to share a large number of the same features with the previous frame, producing the roughly 60% overlap criterion alluded to earlier. For a photographer, this entails moving around an environment and facing inward towards the centre of the scene — something which is normally more challenging for a robot entering an unknown environment.

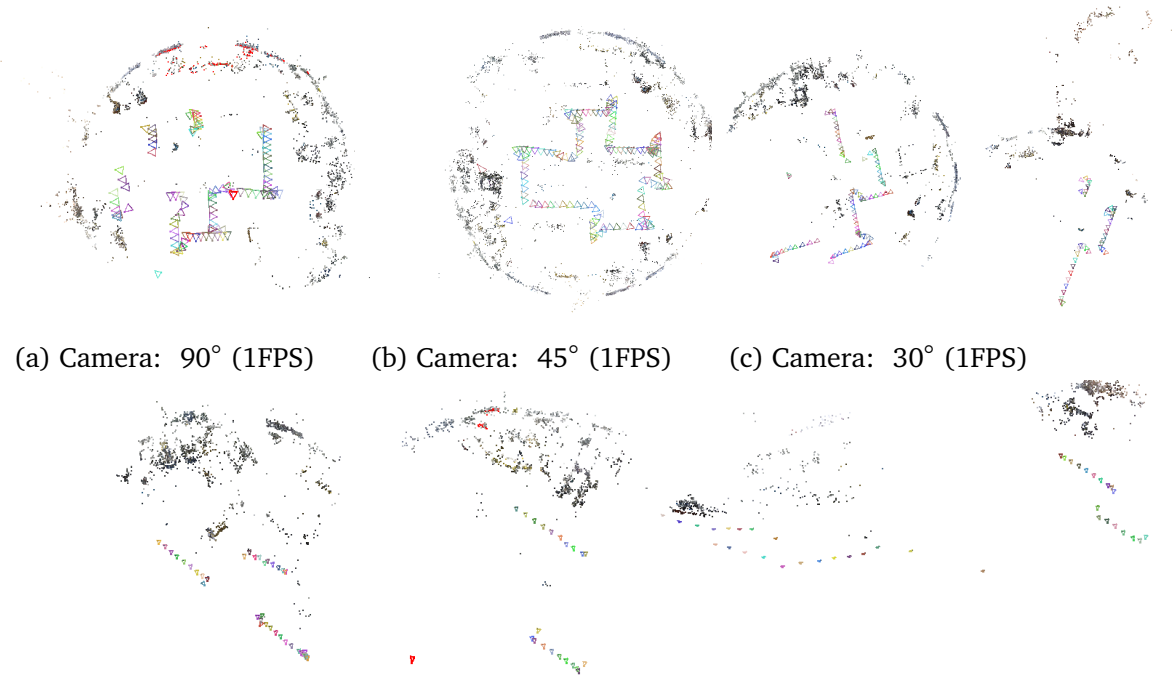
The following experiments seek to evaluate the effect of the orientation of the camera with respect to the direction of travel using a camera mounted on board a ground

robot. A typical, off-the-shelf robot features a single camera, with some platforms offering a second, first-person-view (FPV) camera; however these cameras are of considerably lower quality. These experiments seek to determine the trade-off between an operator's viewpoint and the quality of reconstruction. A complex, maze-like scene was created, comprising of a series of turns, flat surfaces and posts to navigate around, as seen in figure 4.16.

(a) Robot Arena: Starting point (b) Robot Arena: Reverse view

Figure 4.16: Orientation Test Environment

The first run was performed with the camera mounted facing forward, shown in figure 4.17d as the camera at 0°. The Structure from Motion algorithm is unable to compute a single model from the dataset, instead the output has been separated into four distinct models. Given that multiple models are produced it can be seen that this angle yields unsuitable results. Though it is interesting to note that the estimated camera positions are evenly spaced in the direction of travel, as they should be except that the tracks are lost and thus a new model is formed each time tracking is lost. Following this, the camera was rotated 30° towards the left, the centre of the scene. The reconstructed model contained more of the run but is still broken up into multiple models, showing that the trajectory was lost and the algorithms are unable to match the frames to previously-seen features. Rotating the camera further left to a 45° angle yielded the best result. The image sequence produces a clear and complete single model of the test environment. To complete the test, the camera was finally mounted at 90°, directly left of the robot. At this



(a) Camera: 90° (1FPS) (b) Camera: 45° (1FPS) (c) Camera: 30° (1FPS)

(d) Camera: 0° (1FPS)

Figure 4.17: Comparison of camera angle with respect to the robot

angle it can be seen that less data are reconstructed and the result is worse than those from the camera at 45 °.

With the camera mounted at 45°, the view is difficult for an operator to control the vehicle comfortably. Most robot platforms can be equipped with a dedicated first person view camera which transmits a fixed point of view aligned with the front of the vehicle; although this experiment shows that such a view is unsuitable for reconstruction, it *is* necessary for a safe manual override.

4.3 Concluding Remarks

This chapter has presented a series of 3D reconstructions of increasing complexity. The early reconstructions in the simplistic scene demonstrate the level of detail and accuracy which can be obtained purely from 2D images. These experiments have highlighted the importance of capturing high resolution frames in a suitable manor. This chapter has demonstrated that high quality models can be expected from Structure from Motion which can be used for display, documentation and measurement, if the dimensions of a known object is available. The large-scale experiments highlight the drawbacks of Structure from Motion, in that large number of frames lead to significant increases in processing times, even with the aid of powerful computational graphics cards.

The next chapter proposes a solution to the challenge of using video streams in Structure from Motion pipelines. Sampling video at video-rate generates a significant number of images which slows processing unnecessarily, wasting computation time. The solution presents a video filtering strategy to reduce the number of invalid frames and avoid unnecessary processing through rejecting duplicate and similar frames.

CHAPTER 5

Optimising Video for use in Structure from Motion Reconstructions

Structure from Motion systems rely on a series of frames with at least 60% overlap as seen in the previous experimental work. Robotic platforms often provide a monocular video feed which streams live video back to the control systems. Video poses a new set of challenges for 3D reconstruction systems, including how to select frames and, importantly, how to determine the focal length for use in reconstruction. Focal length is encoded in a still photograph's metadata but video streams do not carry this vital information. The simplest solution is to lock the focus of the video camera to ensure that the focal length remains constant during streaming, and then calibrate it. Although this solves the immediate problem, difficulties may well arise later with out-of-focus frames.

This chapter presents a solution in which frames are selected from either a pre-recorded or live video stream for use in a Structure from Motion workflow. The challenge of selecting frames from video is interesting as there is a number of ways in which it can be approached. Section 5.1 begins with direct rate-based frame extraction, effectively recreating the timed interval capture mode from a stills camera. Section 5.2 develops an alternative, more intelligent method, which analyses each

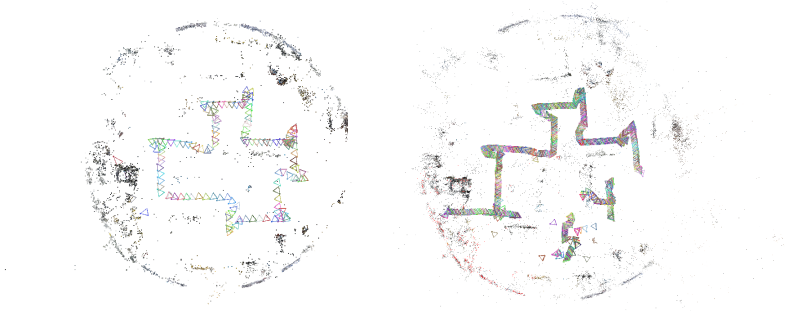
frame and waits for sufficient movement to be identified, filtering frames based on the correspondence percentage. Section 5.3 then adds image analysis to the process to consider sharpness and blur. Section 5.4 completes the pipeline by rejecting frames with excessive overlap. Section 5.5 draws conclusions.

5.1 Frame Rate Sampling

To evaluate frame rate sampling, a ground vehicle was fitted with a GoPro camera mounted 45° to the direction of the motion around an arbitrary scene, based on the results from the experiments in section 4.2. The vehicle made a single pass through the scene shown in figure 4.16. Structure from Motion requires that all frames be in focus (or feature-matching is poor); this rules out using standard web cameras, which are often poor at capturing motion, resulting in blurry frames and poor image quality. The GoPro does not suffer from these deficiencies and was set to record the movement at 50 frames per second in a progressive HD format — the progressive recording format is necessary to avoid the striping and motion-tearing artefacts present in interlaced formats. If interlaced video is unavoidable (e.g. broadcast streaming) then the source needs de-interlacing prior to frame extraction.

Once the robot completed its path through the scene, the video was temporally subsampled to produce sequences at 1, 5 and 10 frames per second. The video was also sampled at 1 frame every 2, 3 and 5 seconds. Figure 5.1 presents a direct comparison between the same video sequence segmented at 1 and then 5 FPS. The reconstruction system has no information about the path or pose of the robot and is working purely with the frames from the uncalibrated video sequence. It can be seen that the error increases as the number of frames increases. The duration of the video is 2:37, therefore if every frame was passed to the 3D reconstruction systems a total of 7,850 frames would require processing. The video sequence generates 158 images when sampled at 1 frame per second, 789 at 5 FPS and 1,578 at 10 FPS. The results vary significantly between the steps of 1, 5 and 10 FPS. In both

the 1 and 5 FPS datasets, the camera positions are generally correct but there is a large area of the path missing in the 5 FPS reconstruction, as seen in 5.1b. The reconstruction fails to construct a single model at 10 FPS; instead multiple chunks of the environment are created in a similar fashion to the results seen in figure 4.17d.



(a) Camera: 45° 1FPS (b) Camera: 45° 5FPS

Figure 5.1: Comparison of sampling rate from incoming video feed

The major drawback with decreasing the sampling rate below 1 FPS is the reduction in the number of frames: sampling once every 2 seconds halves the sequence to just 79 images. This becomes even more of an issue as the interval increases, with only 32 images being produced at 1 frame in every 5 seconds. There are large portions of the scene missing even though the vehicle is moving at a slow (roughly walking) pace. All of these sequences fail to produce a successful model other than the those in figures 5.1a & 5.1b.

5.2 Coherence Filtering

Although frame-rate controlled streaming as discussed in section 5.1 limits the number of incoming frames to a more manageable and reliable number, there is still the possibility of identical frames and over-imaging. These situations occur when capture agents (robot, aircraft or photographer) remain stationary and continue to stream video. Another drawback of the frame-rate controlled approach is that there

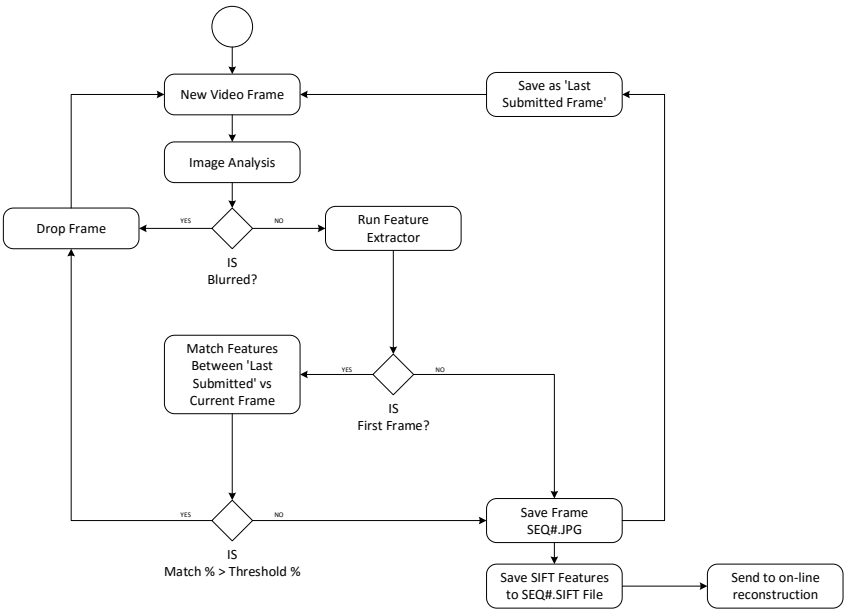


Figure 5.2: Video sequence pre-processor

may be large gaps of the scene missing in the image sequences when the capture agent moves at a non-steady pace or changes direction. Rotations typically happen much faster than the steady pace of straight line paths. As the camera rotates, a much larger area scene passes by the camera, necessitating the need for an different sampling method.

An alternate approach to image sequencing is to filter with the requirements of Structure from Motion in mind, the aim being to select images based on the percentage overlap and their quality. If image pairs are too similar (e.g. the overlapping number of features is greater 90%) then one of the near-duplicate frames can be discarded. Conversely, if the number of features is too low, the frames can also be discarded because the number of matches will be too low for a good reconstruction; instead, the image stream can be redirected to a new Structure from Motion workflow.

Figure 5.2 demonstrates the proposed solution to the coherence filtering approach.

The image analysis stage is used to detect and reject any heavily-blurred frames which arise through vibration or rapid movement. The image analyser utilises a focus measure operator (discussed in section 5.3) to determine the *blurriness* of the incoming frame. Once deemed suitably focused, the frame is then passed through a fast GPU implementation of SIFT [42] to extract the key points for comparison against the previous frame sent for processing. The SIFT analysis is the bottleneck in this pipeline: an Ultra-HD frame takes on average 640 ms to evaluate. Although considerably faster than with CPU implementations, this still restricts the maximum video analysis rate.

5.3 Image Analysis: Blurred Frame Rejection

A comprehensive study on focus measurement was published in 2012 detailing 36 different ways to determine the focus of an image [110]. The study concludes that Laplacian-based operators have the best overall performance in normal imaging conditions. It is important that any video filtering is capable of running as close to video rate as possible, to ensure that the system is able to keep up with an incoming video frame as supplied by the source and avoid buffering it into RAM. A broadcast camera delivers a complete frame every 40 ms (25 Hz). The Laplacian-based algorithms as evaluated by Pertuz *et al.* showed a runtime of 7.10–15 ms (with the exception of LAP5) on a quad-core pentium machine.

The variance of the Laplacian output frame is calculated, returning a single score for the level of blur in the image. A sharp, focused image will have a high variance whereas a soft or motion-blurred image will have a low variance. The variance or 'blurriness' threshold varies depending on the capture source. Implementing a fast blur detector with the Laplacian operation provided by OpenCV shows that, without some optimisation, the goal of 15 ms is difficult to achieve: the total time for the complete operation (Laplacian & two-pass variance) is 42.5 ms per frame. The two-pass variance calculation takes 29 ms to execute on a 1920 × 1080-pixel (Full HD)

image using an Intel i7 quad-core processor. Switching from the two-pass to a one-pass variance calculation method significantly reduces the computation time, to just 18 ms. The on-line method (shown in equation 5.1) produces a running estimate of the variance until the end of the image is reached, where the resulting variance matches that of the two-pass method.

$$\begin{aligned} M_{2,n} &= M_{2,n-1} + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n) \\ s_n^2 &= \frac{M_{2,n}}{n-1} \end{aligned} \tag{5.1}$$

Given that blur that from external influences or high-speed motion will exist across the whole image, one can sub-sample the image to reduce the number of pixels that require processing. Figure 5.3 shows the effects of sub-sampling the image at a range of intervals. The results are based on a full-HD video sequence recorded using a GoPro on board a ground robot during the earlier experiments in section 4.2. The processing time for each of the sampling methods and the percentage difference in the variance from the two-pass method have been plotted. The full list of sampling methods with their respective results are shown in table 5.1. Variance calculations are typically performed on vectors or arrays of data. An image is a two-dimensional array (a matrix), therefore each of the sampling methods select points based on a row-skip and column-skip parameter. The complete range from 1,1 (on-line) through to 7,7 are evaluated to explore the speed and error of each of these sampling approaches.

It was expected that there would be trade-off between execution time and the accuracy of the results. However, what is interesting is the variation in results based on the sampling parameters. Does skipping more rows than columns or vice-versa have an impact on either the speed or accuracy of the variance calculations? The graph shows an initially steep response for each step-up in performance. In the interest of accuracy, an upper limit of 4% difference has been applied when analysing the methods. The execution time drops from 29.88 ms to 11 ms before the error increase reaches a 1% difference from that of the two-pass method. Table 5.2 shows

the sampling methods within the region of 2–4% sorted by their percentage difference from the variance result produced by the two-pass method. Sampling pixels at every 2^{nd} row and every 3^{rd} column or *vice versa* produces results in the fastest time within this bracket. However the results from these two methods produce very different results. S15 (3,2) performs worse than S9 (2,3), suggesting that skipping more rows than columns has a detrimental effect on calculating the variance of a Laplacian image which can be seen across the rest of the sampling methods from the full table.

5.4 Frame Selection

The final stage of this 'video to 3D reconstruction' pipeline is to filter the frames to address the issues of overlap and video files generating too many similar/repeating frames. Live streaming behaves in the same way as a pre-recorded (Video on Demand, VoD) file in so much as the way in which OpenCV makes frames available to processing. This section will refer to the video source as a video stream (an ordered collection of frames) as the method of handling frame-wise and pair-wise operations is the same.

Once a clean, non-distorted frame is identified it is processed using a GPU implementation of ORB. [4] details a range of feature detectors, noting their respective performance and computational cost. SIFTGPU [42] produced 1,600 features per frame in 100 – 175 ms per frame whereas the GPU ORB detector produced 2,500 features in just 16.8ms per frame. Owing to the fact that most of the incoming frames will be rejected due to short baselines, and a frame rate of 25 frames per second was used, ORB was selected as the primary feature detector. Successful frames will be stored and passed through SIFTGPU when deemed suitable for 3D reconstruction. This approach of using two feature detectors enables the system to run much faster and still be able to produce SIFT key point files for successful frames.

In [4], a GPU-based Brute Force (BF) matcher is used to find the corresponding

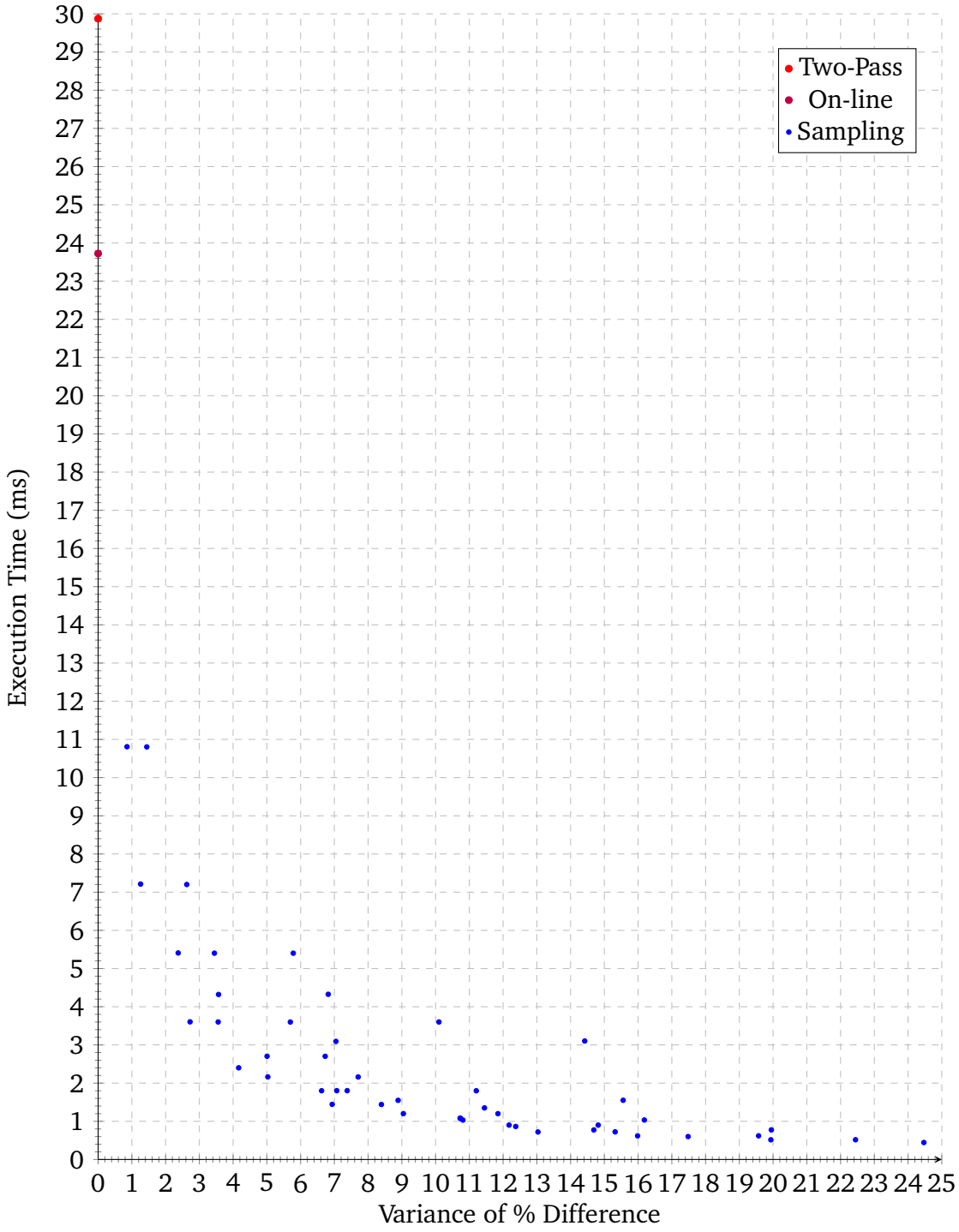


Figure 5.3: Variance By Sampling: Execution Time vs Error

Table 5.1: Variance Sampling Methods

Method	Difference (%)	Time (ms)	Method	Difference (%)	Time (ms)
S1 1,2	0.856017	10.8085	S25 4,5	10.7252	1.080030
S2 1,3	1.258290	7.21235	S26 4,6	12.1769	0.900096
S3 1,4	3.445640	5.40285	S27 4,7	14.6886	0.773537
S4 1,5	3.568700	4.32103	S28 5,1	6.81850	4.325490
S5 1,6	5.695250	3.59943	S29 5,2	7.70476	2.160930
S6 1,7	7.046780	3.09284	S30 5,3	8.39395	1.440020
S7 2,1	1.441870	10.8038	S31 5,4	10.7346	1.080040
S8 2,2	2.372150	5.40776	S32 5,5	12.3742	0.863978
S9 2,3	2.723820	3.60480	S33 5,6	13.0331	0.720042
S10 2,4	5.005760	2.70379	S34 5,7	15.9855	0.618766
S11 2,5	5.027850	2.16230	S35 6,1	10.0992	3.601140
S12 2,6	7.379320	1.80126	S36 6,2	11.2056	1.799110
S13 2,7	8.888480	1.54856	S37 6,3	11.8471	1.200030
S14 3,1	2.626160	7.20041	S38 6,4	14.8180	0.900414
S15 3,2	3.556670	3.60128	S39 6,5	15.3195	0.723496
S16 3,3	4.164170	2.40057	S40 6,6	17.4821	0.600258
S17 3,4	6.621560	1.79951	S41 6,7	19.9342	0.515760
S18 3,5	6.935560	1.44405	S42 7,1	14.4192	3.102760
S19 3,6	9.044720	1.20060	S43 7,2	15.5580	1.550110
S20 3,7	10.80980	1.03162	S44 7,3	16.1853	1.033310
S21 4,1	5.782080	5.40176	S45 7,4	19.9482	0.775279
S22 4,2	6.728080	2.70188	S46 7,5	19.5711	0.619957
S23 4,3	7.072760	1.80145	S47 7,6	22.4434	0.516807
S24 4,4	11.44840	1.35023	S48 7,7	24.4681	0.444182

Table 5.2: Sampling Methods: 2-4% Difference

Method (row, col)	Difference (%)	Time (ms)
S8: 2,2	2.37215	5.40776
S14: 3,1	2.62616	7.20041
S9: 2,3	2.72382	3.60480
S3: 1,4	3.44564	5.40285
S15: 3,2	3.55667	3.60128
S4: 1,5	3.56870	4.32103

matches between the key points from the pairs of frames. A CPU implementation takes up to 550 ms to complete based on 2,500 matches in a pair of HD frames. The GPU implementation reduces this time to 25 – 80 ms for the same pair of images. A metric for similarity is obtained by calculating the distance between the matched features. The distance measure depends on the type of feature detector used to extract the original features. If a floating point feature detector such as SIFT or SURF was used, then the Euclidean distance measure is normally used. Binary feature detectors such as ORB require the Hamming distance to be used instead. The results demonstrated by [4] show that the threshold of the Hamming distance needs to be manually tuned for each application: a distance of 25 resulted in a good model of the environment for the ground vehicle, yet a distance of 55 was required for the aerial footage.

5.5 Concluding Remarks

This chapter presented a video filtering strategy to reduce the number of invalid frames and avoid unnecessary processing caused by using raw video as input to Structure from Motion systems. This work produced a new pre-processing module optimised for Structure from Motion algorithms through rejecting duplicate and similar frames. The system requires 70.76 ms – 125.76 ms (depending on the number of key-points selected) to analyse each frame, resulting in a processing rate of 14 Hz. This filter would be suitable for use in off-line reconstruction with existing packages such as PhotoScan and VisualSFM. The filter could also be employed in live video systems, however the output would be limited to 14 Hz. Further investigation into matching strategies and alternate feature detectors would be required in order to reduce filtering time to reach a live video-rate (25 Hz).

The next chapter explores robotic aspects such as aerial observation, control structures to develop a framework for use in later experimental work in preparation for the penultimate chapter, which seeks to combine the work of these chapters together

to form a robotic remote reconstruction platform. The camera orientation and frame rate experiments feed into the design of the final system deployed in Chapter 8.

CHAPTER 6

Constructing a Collaborative Robot Demonstrator

The thesis so far has concentrated on the theoretical and practical aspects of 3D reconstruction. This chapter focuses on the robotics aspects needed to provide a platform for the vision systems. Section 6.1 explores the challenges of localising ground robots based on imagery from an aerial point-of-view. In this approach an aircraft is used to hover above a target to provide position feedback in GPS-limited environments. The research initially utilised a common robotic framework, ROS (Robotic Operating System) in combination with low-cost robotic systems. Section 6.2 covers the ROS architecture and the shift across to a dedicated approach, before introducing modifications to incorporate ground vehicles. Section 6.3 draws conclusions from these experiments.

6.1 Aerial Observation

Accurate position determination is a continual problem across the field of robotics. Although there are many competing technologies, all have shortcomings that restrict

their range of use. For example, conventional shaft encoder-based odometry may be acceptable for ground robots (though it suffers from problems such as wheel slippage) but is clearly inappropriate for aircraft or underwater robots. GPS, the Global Positioning System, can provide latitude and longitude but cannot be used successfully indoors and is too inaccurate for many uses with a root-mean-square (RMS) error of $\sim \pm 2\,\mathrm{m}$ [111]. Visual odometry [112, 113] and its related technique of visual SLAM [114], which both use video streams from a camera, can perform well but require the matching of visual features at video rate, quite a difficult task. The state of the art is arguably provided by time-of-flight systems; while these can operate in any environment, they are expensive and cumbersome and this makes them unattractive for robotics. Where one has the ability to instrument the environment, one can do better. Placing fiducials (markers) at known positions in the environment allows one to determine where one is relative to them (*e.g.* [115]). Landmarks can take the form of dedicated patterns or key beacons which, once identified, provide the system with an accurate location of the landmark. Position is then inferred through calculating the distance between the sensors and the landmarks. A marker can also be placed on the vehicle (or object to be tracked) so that the position of the object can be tracked from the fixed landmarks. Multiple-camera systems such as those produced by Vicon & Qualisys have become the industry standard for capturing human motion and involve attaching reflective markers to the objects being tracked [116].

A technology that is attractive in principle is to attach fiducials to the various robots, and then to determine position relative to those markers without any fixed landmarks. This approach employs only cameras, which can be used on ground or airborne vehicles, and is equally effective indoors or outside (providing the cameras are equipped with exposure control). The notion of using fudicials as landmarks is not new in robotics; however, in this work the landmarks are the robots themselves. The particular approach used here is to attach QR code fiducials [117] to the upper surfaces of ground robots so that an aerial observer (drone) may determine its position relative to one or more of them. When a ground robot needs to move through

a particular distance in a particular direction, the drone needs to hover (relative to other fiducials or to ground features) until it perceives the distance traversed to be correct. Clearly, there are inaccuracies in each stage of this scheme, and the principal purpose of the following experiments is to establish whether the errors introduced are random, and hence tend to cancel out over several movements, or systematic [118].

6.1.1 Visual Altitude Estimation

Acquiring depth from images is a well-established problem in Computer Vision as discussed in earlier chapters. Chapter 3 demonstrates a method of using one or more cameras to recover 3D information using feature matches for the purpose of 3D reconstruction. Structure from Motion is unsuitable as depth estimator due to its extensive processing requirements. A real-time approach is required to keep up with the fast update-rate of the PID controllers. The visual estimator must run at video-rate in order for a vision-based altitude approach to be usable on an aircraft.

The distance between the camera and and a known target can be calculated using:

$$\begin{aligned} Z_x &= \frac{Wf_x}{w_x} \\ Z_y &= \frac{Wf_y}{w_y} \end{aligned} \tag{6.1}$$

where W is the known width in millimetres, w_x, w_y measured width in pixels, f_x, f_y the focal length in pixels and Z_x, Z_y are the resulting distances to the object in millimetres. Z_x, Z_y are then averaged together to return a combined result to reduce the effect of a non-square image due to perspective effects.

6.1.2 Experimental Set-up

The aim of this experiment is to determine the reliability of using only the on-board camera and a QR code as a fiducial to calculate the altitude of an aircraft. Ground robots were equipped with QR ("Quick Response") 2D barcodes of size 17×17 cm on their upper surfaces. QR codes [117] are now in fairly widespread use because they contain only black and white regions and can be read (with appropriate software) using a conventional camera. They contain three 'finder' patterns in their corners which help distinguish the QR code from visual clutter, and the rest of the pattern encodes a series of characters. The 'finder' patterns allow the orientation of a QR code (and hence the vehicle to which it is attached) to be determined. An open source library, ZBar [119], was used for detecting and decoding QR codes. This interfaces to the popular OpenCV software, though it is necessary to use OpenCV to pre-process and manage the incoming image streams before presenting them to the routines that find and decode a bar code.

In order to limit the impact of random motion caused by working with live aircraft and robots, a camera crane was used to fix the aircraft in a static position. The crane provides a stable and repeatable method of setting the aircraft altitude without running the aircraft's motors. Fixing the position of the aircraft eliminates random behaviours caused by vibration and air turbulence. The crane keeps the aircraft (via the camera foot-plate) level through its entire range of motion. The ground robot is substituted with a camera track and dolly system which allows the target (QR Code) to glide back and fourth in the view of the camera in a perfectly straight motion. Figure 6.1 shows an overview of the physical experimental set-up.

The aircraft, a Parrot AR Drone 2.0, communicates with a laptop using the relevant modules within ROS, the Robotic Operating System [120]. ROS is a framework designed to work with a range of robots and provides and extensive range of modules designed to support the development of robotic platforms. In this work ROS was used both to control and utilise the on-board cameras of the UAVs, as well as capture the data being published from a Vicon system (to provide global 'ground truth')

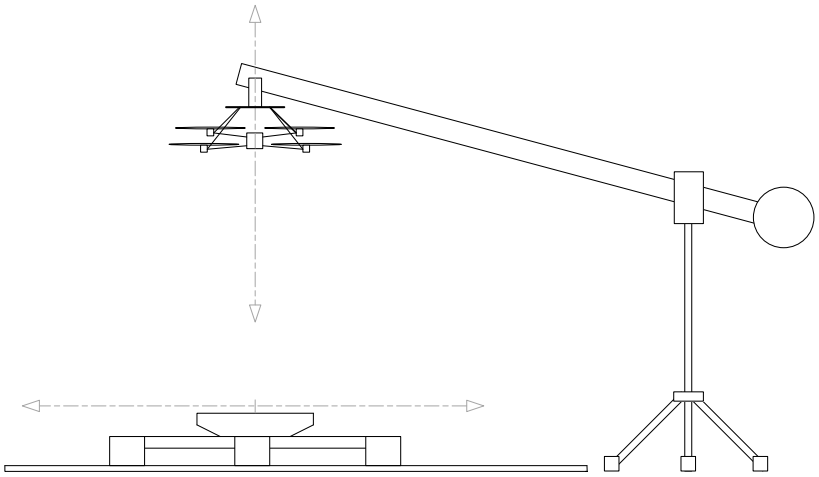


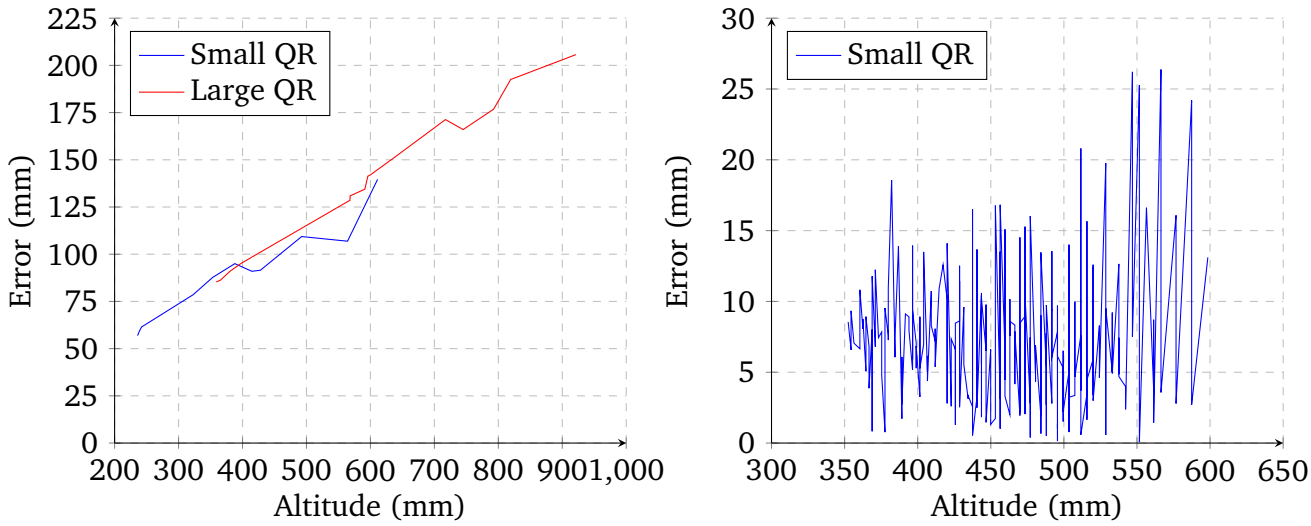
Figure 6.1: Aerial QR Measurement Test Environment

and the QR tracking software. The ROS configuration and modules are discussed in detail in Section 6.2.1.

6.1.3 Altitude Reading Results

The decoding library, ZBar, is designed to run at video rate; however during testing and development it was observed that the library was not always able to successfully decode a QR code when clearly visible in the image. In previous work [3] it was observed that sensor size has a significant impact on reading distance, however even the large (4K+ pixels) sensors struggled to decode the QR target beyond 2.8 m (1.8 m in the case of smaller targets). The resulting system is able to read and locate targets at an average of 20 Hz.

The QR size ultimately determines both the minimum and maximum altitudes of the aircraft due to the fixed field of view of the camera. Figure 6.2a clearly shows that there is a linear relationship between the altitude of the aircraft and the error in the height estimation from the QR code, regardless of the size of the QR code



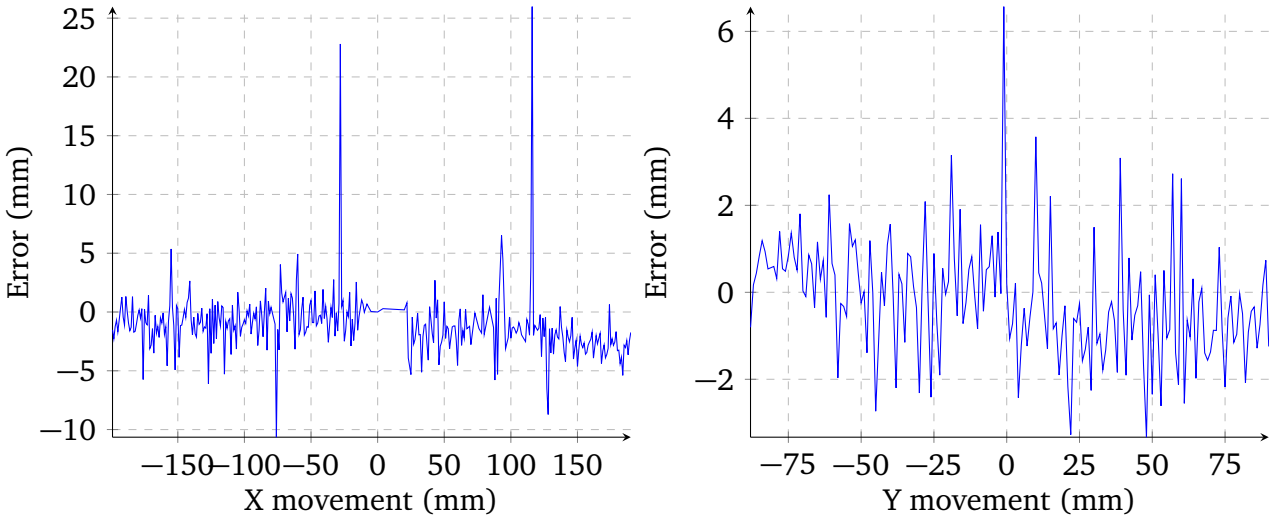
(a) (Uncalibrated) QR Measurement

(b) Small QR Error After Calibration

Figure 6.2: Altitude Estimation Error

used. The results were obtained from an uncalibrated camera. Interestingly there appears to be a consistent gain in error for every 100 mm of altitude. The minimum reading altitude for the smaller QR was observed at 235 mm, and at 360 mm for the larger one. Once the camera system is calibrated the results improve significantly. Figure 6.2b shows that after calibration and distortion compensation, the error reduces to a peak of 26 mm and is fairly consistent throughout the range. Calibration does not enable an increase the maximum reading altitude: this is limited by the contrast and resolution of the image.

To evaluate the accuracy of measuring ground movements for use in coordinating robots moving under the aircraft, the UAV was fixed at an altitude of 870 mm using the crane. This gives a view on the ground of 600 mm × 300 mm. The larger QR code (170 mm) was mounted to the camera dolly and moved through the camera's field of view. The results of these experiments with a static aircraft demonstrate that, providing that the aircraft can maintain a stable altitude of up to 1 metre, a QR code can provide an altitude measurement with an accuracy of ±25 mm in Z and relative position information with an accuracy of ±20 mm in X and Y. Figures 6.3a



(a) QR moving in the image x plane with a calibrated camera source

(b) QR moving in the image y plane with a calibrated camera source

Figure 6.3: Target Movement Estimation Error

and 6.3b demonstrate that the system becomes significantly more accurate when working with undistorted images. The error reduces to approximately ±5 mm error in the x plane of the image and ±2 mm in y.

These results have shown that, given a fairly stable aerial platform, QR codes can be used to provide relative position information with an accuracy up to 20 mm in the X-Y plane. Introducing pitch and roll skews the results, increasing the error as seen in figure 6.4. This approach is limited by the quality and resolution of the camera available; however, the results have shown that even a low-resolution sensor is able to achieve an accuracy of 20 mm from a height of one metre. Camera systems with longer focal lengths and higher contrast ratios will enable control to be achieved from higher altitudes; this is desirable because the aircraft needs to be able to operate above head height and cover a wider ground area. Additionally, it should be possible to utilise other sensors such as on-board Inertial Measurement Units (IMUs) or 3-axis accelerometers to compensate for the aircraft angle and hence reduce the errors seen through non-level measurements.

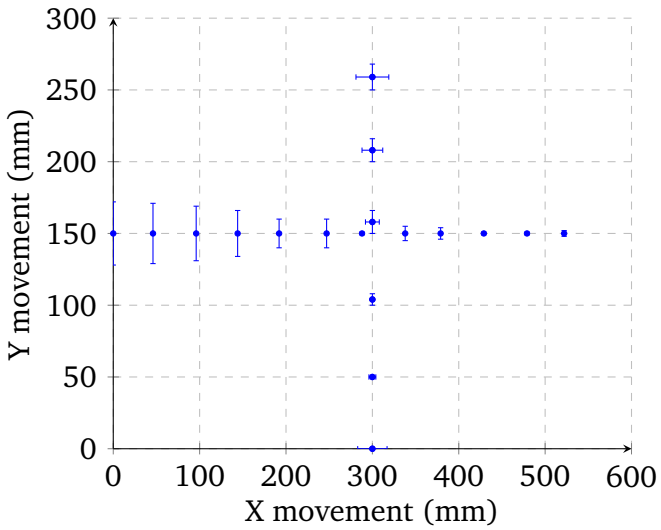


Figure 6.4: QR tacking with aircraft at +5° pitch and roll

6.2 Communication Frameworks

When investigating high-level robotic control such as inter-robot collaboration or mission planning it is inconvenient to have to reimplement low-level control of vehicles. A ground vehicle is a fairly easy case to work with as the consequences of failure are fairly minimal. However when working with more sophisticated and complex vehicles such as aircraft, underwater robots or large-scale robotic systems such as cars or trucks, the task of implementing low-level control can become extremely lengthy and require additional safety measures. Implementing low-level control systems takes time and careful work which has to be done before the research goals of the high-level algorithms can be developed.

Frameworks such as ROS, the Robotic Operating System [120], Paparazzi UAV [121] and MAVLink [122] all support the development of robotic systems by providing pre-built modules and community-driven development. These are contributed by developers and robotic engineers to produce a collection of tools that are peer reviewed and provide a solid foundation for further development. The aim of these

frameworks is to enable faster development of robotic systems and provide reusable modules. The experimental work in Section 6.1 utilised ROS with the ARDrone Autonomy [123] modules. ARDrone uses a modified version of the official Parrot AR2 SDK [124] to integrate a Parrot AR Drone 1 or 2 with ROS.

6.2.1 The Robotic Operating System (ROS)

The Robotic Operation System, ROS, is not an operating system in the traditional sense. In fact, ROS does not have low-level kernel or installable modules that would be akin to a traditional operating system such as Linux or Microsoft Windows. Instead, ROS is a combination of a framework and a library of pre-built modules which when combined together create a robotic system. ROS can run across multiple nodes which are networked together and communicate using a master node. This approach enables low-powered robots to utilise faster machines on the network for higher-level tasks such as visualisation.

The whole ROS architecture is built on a Publisher/Subscriber model [125]. Everything that needs to be passed through ROS is formed into a message and published on a topic. A topic is formed of a single publisher and can support multiple subscribers. ROS requires one master node to coordinate the topics and maintain / notify the subscribers. Only one master can exist on a ROS network at any one time; starting another master will cause the first to shut down or the attempt to fail.

The aerial observation experiments discussed in the previous section were configured with 3 core nodes: the ROS Master, AR Drone Driver and a Visual QR Processor (custom module). The driver node published a series of topics presented by the AR SDK. There was one topic for each of the statuses presented by the aircraft. For example to retrieve the battery level of the drone one would subscribe to the 'battery' topic which will publish a message on each update to the change in battery level. The driver package also listens to the 'cmd_vel' topic which communicates using a ROS Twist Message. A Twist message is formed of two vectors which encode

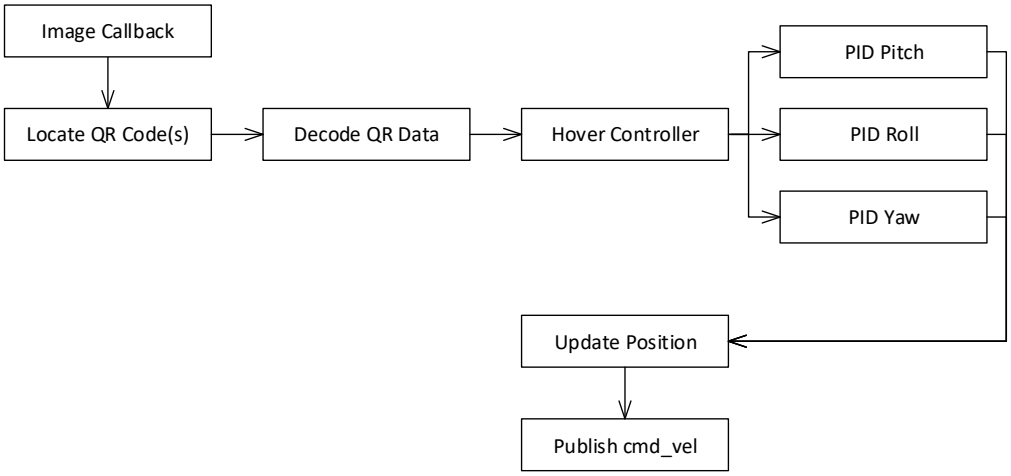


Figure 6.5: Aerial QR-Based Hover Control

the linear translation and angular rotation for each axis (x, y, z).

Extending the work of the aerial observations to enable autonomous flight adds an additional four nodes; three PID modules (one for roll, pitch and yaw) and a joystick driver. The 'joy' module is a generic joystick interface which enables a Linux-compatible gamepad to be used to provide control signals. The controller is mapped onto the command topic of the drone to provide operator control and a method to take over in the event of an emergency or fast approaching wall. The QR processor provides the location and rotation of the target with respect to the centre of the image. This information is sent to the PID controller in order to hover the aircraft over the target QR code. Figure 6.5 shows a component overview of an incoming frame being decoded through to the position of the aircraft being updated via a 'cmd_vel' message being published on the ROS network.

A ROS network can become complicated quickly, with information flowing at a high speed between nodes. Modules are often inter-weaved, with numerous messages flowing through the master. The physical network infrastructure needs to keep up with the large bandwidth demands of the information flow. Certain topics are updated at rates up to 250 Hz. The AR driver varies between 15 Hz – 200 Hz depending

on the topic and Vicon bridge (50 Hz – 250 Hz depending on capture rate). The large number of messages places a high demand on the network stack. 2.4 GHz (B/G/N) WiFi networks struggle with this level of throughput, leading to the need for wired gigabit networking adapters.

A consequence of a highly flexible platform is the cost in configuring the 'ideal' set-up. During development and experimentation with ROS, its core needed to be reconfigured and recompiled several times, consuming a significant amount of time, especially in the event of missing dependences or conflicting packages. The binary (repository) versions of ROS pull in many free open-source modules which may conflict with customised source versions, leading to tricky and sometimes unrecoverable states. ROS changed build pipelines and tool-chains between versions which has also led to issues, as some developers have not updated build processes to reflect the change in direction of the framework. The software configuration and compilation difficulties are, it can be argued, simply a reflection of the fact that ROS is immature and undergoing rapid development from the community. However, the combination of the publish-subscribe architecture and the need to route messages through a single master means that ROS is currently poorly suited to environments that use bandwidth-constrained wireless networks to transfer data and not consistent with the notions of 'swarms' of robots discussed in Chapter 2. Extensive tinkering with ROS was unable to produce a configuration that allowed the data necessary for control and processing to be transferred without significant losses across a WiFi link, even though the volume of data should be within its capability. It was therefore decided to develop a ROS-free solution, to ascertain whether, without the ROS overheads, this can be made to work.

6.2.2 Drone Handler — A Custom System

Another motivator for moving away from ROS was the release of the Parrot Bebop (version 1) aircraft. The Bebop is a more stable platform than the AR2 with improved optics and support for newer wireless network standards including 5 GHz.

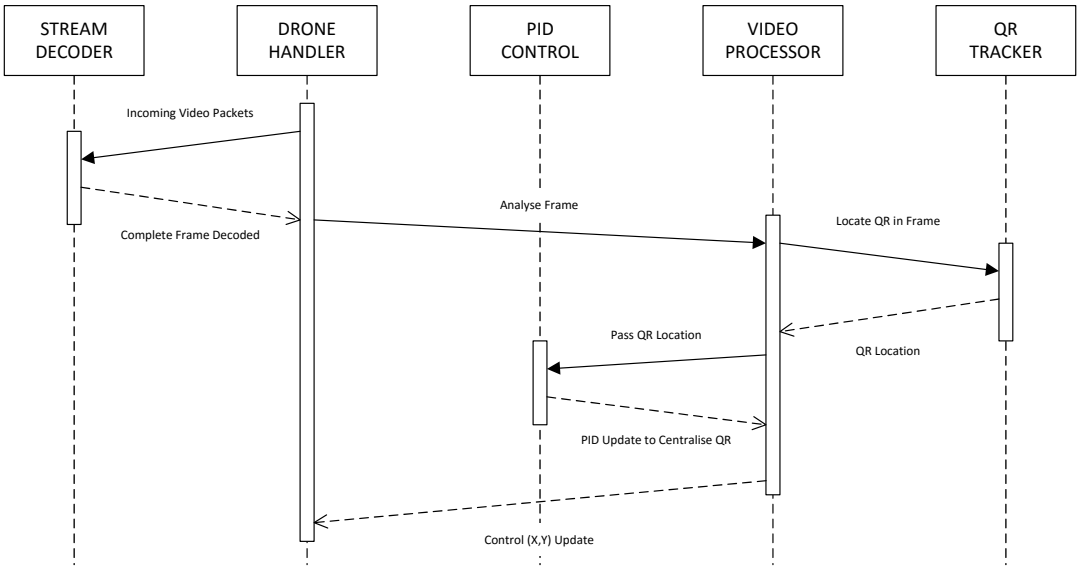


Figure 6.6: UML Sequence Diagram: UAV QR Position Hold

The Bebop replaces the AR's downward-facing low-resolution camera with a single fish-eye camera mounted at the front of the aircraft. The platform generates a software stabilised video feed with the aid of on-board sensors. The direction of the video view can be aimed using the control protocol. The Parrot AR3-SDK was in its early stages and there were no ROS modules for the newer Bebop aircraft at the time of investigation, though a ROS Bebop driver module started development in September 2015 [126].

Figure 6.6 documents the sequence of receiving a stream from the aircraft through to providing positioning commands using the communication and processing system developed by the author. The ground control station communicates with the aircraft using JavaScript Object Notation (JSON) messages. The SDK opens up two uni-directional streams, one from aircraft to controller (D2C) and the other from controller to aircraft (C2D). The D2C stream includes status information and video packets, while the C2D contains direct flight control instructions. Once the D2C stream is activated, the aircraft streams the video in the form of H.264 fragments which need to be passed to a video decoder such as FFMPEG in order to be rendered

(a) Bebop At 922 mm (b) Bebop At 2170 mm

Figure 6.7: Bebop Flight Tests

into frames. Once a complete frame is decoded it is converted into an OpenCV matrix for use in the visual processing pipelines and QR analysis.

Three PID controllers are used to update the control command packets for the aircraft. The PID controllers are set up to have the drone 'home in' on the QR code until it hovers directly above at a fixed altitude. The PID control loops are updated on successful identification of a target QR code. Figure 6.7 shows two still frames extracted by the QR processing code. The Bebop extends the operational altitude to just over 2 metres compared to 0.9 m with the AR2. The image in figure 6.7a shows the clear image presented by the on-board camera from the Bebop video stream at the limit of the AR2. The additional small QR targets become unreadable beyond 0.9 m as seen by the missing match of the lower QR in figure 6.7a.

One significant advantage that this approach has over ROS is that the system works a single entity. There are no messages passed through the IP stack which are critical to control. A 6 Degree of Freedom (6-DOF) input device is used for direct operator control to fly the aircraft into position as well as providing emergency control. The system also has fail-safes to cope with target loss and altitude limits. A TCP interface has been integrated to enable a remote process to inject control commands and

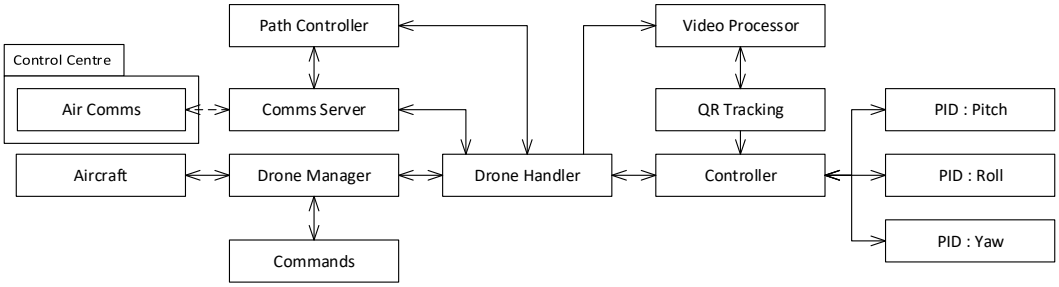


Figure 6.8: AR Drone Bebop Control — Component diagram

receive a subset of key information such as battery, flight data (state, altitude), and last seen QR data.

Figure 6.8 shows a breakdown of the drone control application. The drone control system has been entirely written from the ground up in C++, with the aid of the ARDrone SDK [127] and C++11 STL. The codebase is formed of 5,377 lines of code (1,148 of which are documentation and comments) across 22 classes. The solution took several months to implement and test. The drone handler has been designed to operate in both stand-alone and collaborative modes whereby the aircraft will automatically hover over a target QR code when it appears in the field of view which can be enabled and disabled via a command over the TCP interface. The drone handler makes up just over half of the C++ line count for the collaboration systems developed later in Chapter 8. The system uses a custom protocol to communicate between applications over the local (127.0.0.1) TCP/IP stack. The protocol is packetised using a C++11 JSON library, though any JSON library can be used to generate JSON messages to interface with this control application.

6.2.3 Collaboration

A single platform may not be sufficiently suitable to all types of reconnaissance task. Each platform has a series of features which have specific advantages. Ground

vehicles are ideal for carrying heavy payloads and large sensors over great distances whereas aircraft are more suited to observation-based tasks due to the limited lift and battery capacities. Depending on the type of aircraft deployed, the time-on-scene (ToS) can vary from 10 minutes through to several hours. A collaborative team made up from a selection of air and ground vehicles provides a useful resource which utilises the strengths of the individual platforms for a common goal.

Section 6.1 detailed how an aircraft can use ground data to localise itself for relative positioning. This work can be extended to provide information to the ground vehicles and thus create a higher level of situational awareness. Situational awareness is a term often used in command and control scenarios which refers to the level of perception of environmental factors. The US Coast Guard defines situational awareness as "the ability to identify, process, and comprehend the critical elements of information about what is happening to the team with regards to the mission" [128]. In the case of a robotic team, an aerial perspective produces information about the wider environment including the locations of targets, waypoints and the ground vehicles. Robotic ground vehicles produce specific information to aid the mission from dedicated sensors under the guidance of the aerial information. A single robot is able to sense its environment and begin to build a model of the world as it sees it, but this information is local to the robot, not the wider mission. The challenge arises when multiple robots are used to combine data to generate a wider model. Given a team of robots, can a task such as mapping be completed in a more accurate and faster way than using just a single vehicle?

Control philosophy is often broken down into two core models: centralised and distributed control. This partitioning is found across a range of disciplines, from software development (source control systems) through to industrial plant control. Inter-platform collaboration is possible with the ROS node model, but ROS is typically configured to work on a single robot. The ROS model is more akin to a centralised model as all of the nodes within a ROS network are required to talk through a ROS Master and are tightly linked to each other. For effective collaboration between robots and missions, the author contends that a more distributed approach is better.

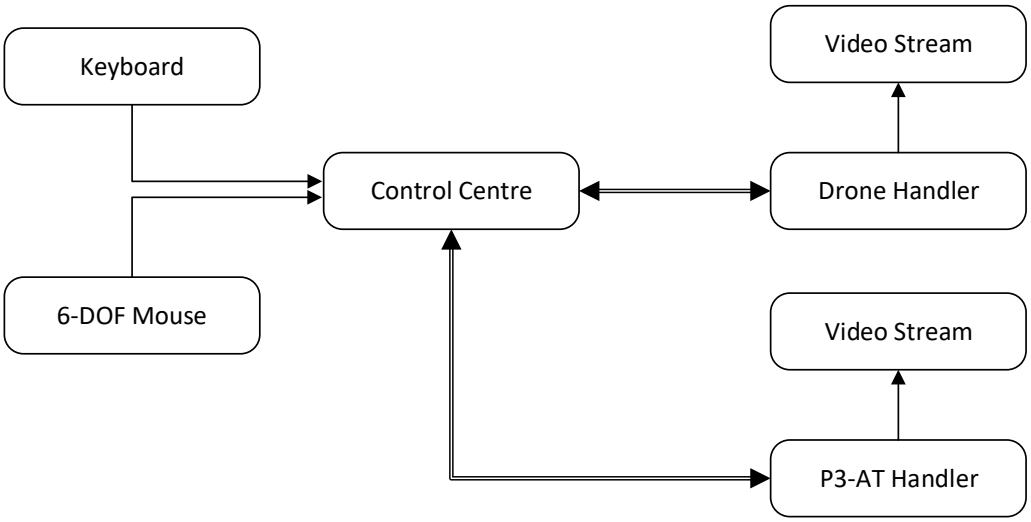


Figure 6.9: Multi-robot control overview

Each robot should be capable of performing a series of tasks (moving to way-points, location feedback, sensing, etc.) without direct interaction with the high-level control systems, fitting a distributed model. A centralised control system then looks after the mission planning and dynamic elements such as path-planning and computationally expensive areas such as 3D model generation. Hence the custom controller work from section 6.2.2 was extended to integrate 'nodes' for air, ground and central ground control.

The new approach creates separate control systems for each robot. The previous air controller has been modified to receive position commands over TCP and provide feedback over another TCP port. The ground vehicle used in these experiments was a Pioneer 3-AT (P3-AT), a versatile four wheel drive robotic platform. The P3-AT is equipped with a SICK 2D laser scanner and 16 ultrasonic sonar sensors, as well as a dedicated on-board computer to control the robot. The control application was developed using the same approach as the air controller in that a 3D mouse or joystick can be used to directly control the vehicle, while other tasks such as wander were implemented to enable the vehicle to operate in a stand-alone mode.

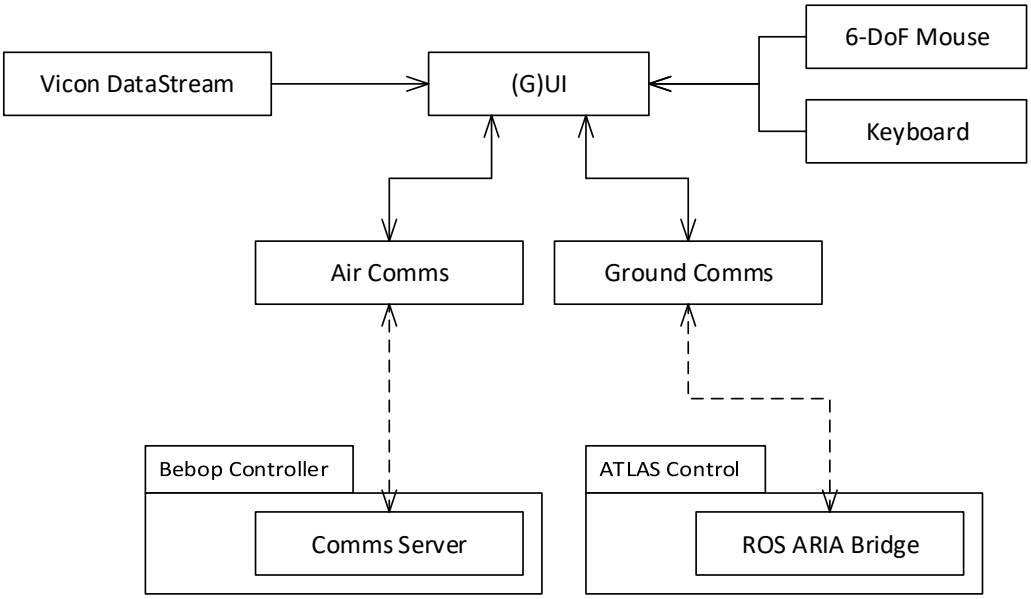


Figure 6.10: Control Centre - Component diagram

A separate application, shown as the 'Control Centre' in figure 6.9, presents a graphical command-line user interface (via ncurses [129] and a terminal) for operator control of the vehicles as well as presenting live data from each platform in one central location. Figure 6.10 shows a simplified block diagram of the modules that feed into the application. Figure 6.11 shows the user interface presented in a terminal window with the aid of the ncurses libraries. The control application allows the user to select an individual robot for manual control via a 6-DOF 3D mouse. This enables the operator to pilot the vehicle or send land/takeoff and emergency stop instructions. The controller can also enable / disable the automated tracking modules on the robots. When enabled, the aircraft will home in and hover directly over the QR target mounted to the robot. The controller can then pilot the ground vehicle (or request it to wander) in order to explore the area using the robotic team. Figure 6.10 shows a breakdown of the modules within the 'Control Centre' and the respective modules within the Bebop controller and the ground robot control applications. A

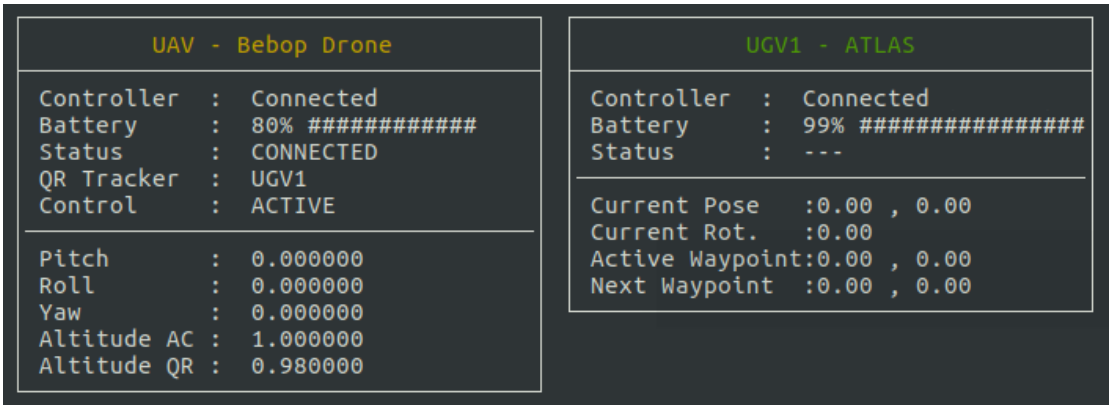


Figure 6.11: Control Centre - Terminal User Interface

detailed breakdown of the ground controller (ATLAS Control) is documented later in Chapter 8.

6.3 Concluding Remarks

This chapter has presented the practical elements and associated difficulties of using robotic systems for 3D reconstruction. The chapter began with using vision to provide a visual altitude measurement which feeds into the air control system to enable QR tracking from an aerial vehicle. The work was started under ROS as it provided a basis on which to develop the tracking modules without focusing on low-level vehicle control. Although ROS is clearly a useful framework, making the set-up work became so difficult that a custom implementation became more practical. The release of the Parrot Bebop inspired the development of the custom system which offered greater capability and extension to work alongside ground vehicles.

Chapter 8 builds on the frameworks developed in this chapter in order to form a system capable of an autonomous 3D reconstruction using air and ground vehicles. The next chapter pauses to consider the challenge of transmitting large format imagery across networks to provide greater detail and more accurate modelling.

CHAPTER 7

Transmitting Ultra-Large Imagery

The air and ground experimentation in the previous chapters have revealed a key limitation of vision-based robotic systems, in that all of the platforms have relatively low-resolution video transmission limited by the available bandwidth. The quality of the resulting models is directly related to the input image datasets. The visual altitude solutions presented previously in 6.1.1 are restricted by the quality of the image received from the aircraft which ultimately limits the operational altitude (or fiducial size) of such systems. Before the thesis moves to create a complete system comprising of aircraft and ground vehicles to generate a model of an unknown environment, the work pauses to explore the challenge of transmitting Ultra-High resolution (4K+) imagery from mobile platforms. Ultra-HD images contain over four times the number of pixels of Full-HD frames, and will ultimately become a requirement for robotic platforms to capture and transmit.

Section 7.1 introduces the area of Ultra-HD imaging and direction of the field of video coding with respect to video encoding of large formats. Ultra-HD, also known as 4K, is growing fast with the film and television industries, with 4K broadcasting leading the way in the sector. Section 7.2 describes a typical broadcasting set-up and how workflows can be adapted to integrate Ultra-HD into existing broadcasting studio spaces. Sections 7.3 and 7.4 explore the technical limitations and propose

a solution for transmitting live Ultra-HD video. This is followed by a real-world test of the solution in section 7.5, where the system was used to stream a series of graduation ceremonies to a world-wide audience. Section 7.7 completes the chapter through conclusions drawn from these experiments.

7.1 Ultra-High Resolution Imagery

Resolution, the number of pixels in an image, is often a controlling factor in computer vision tasks. Image quality and resolution are common problems faced by image processing systems and often the stumbling block preventing their successful use in real-world applications. The aerial observation work discussed in this chapter is severely affected by resolution, for example: the aircraft altitude is restricted by the quality of the image. The issue becomes more acute as the distance between the camera and the target increases. The contrast and overall definition of the QR code degrades to the point where the individual blocks of the QR code become blurred and unrecognisable. As the distance increases further, the large finder patterns in the QR code are also lost, losing the defined square. Ultra-High Definition (UHD) cameras can be used to capture extremely detailed images and provide large, sharp images allowing for greater resolution for use in image processing systems. Switching to higher resolution cameras introduces a series of new challenges in transport, processing and storage.

Ultra-HD images are considerably bigger than HD. 4K Ultra-HD has 3840 × 2160 pixels per frame, offering four times the definition of Full-HD (FHD, 1920 × 1080) and nine times the resolution of HD (720p) content. Transmitting live 4K Ultra-HD (4K-UHD) video requires heavy processing in order to deliver that content to end systems (and viewers). Before evaluating Ultra-HD for use in robotics it is worth looking into the delivery of Ultra-HD in broadcast environments such as live events and on-demand video content. Until recently the widely-accepted standard for streaming and storing video has been H.264 [130]. Superseding MPEG-2 and

MPEG-4. Generally MPEG is less efficient than that of H.264 in terms of compression, image quality and network efficiency. H.264 supports more advanced compression methods compared with the basic MPEG-4 encoders. H.264/AVC encoding is a widely used encoding format used on a range of systems from broadcast encoders through to smart phones, tablets and set-top boxes (and robots). The H.264 standard has support for a wide range of video frame sizes including 4K-UHD through the use of the highest levels of the encoding options (discussed in detail in section 7.4). H.264 is a mature encoding standard, though not well-optimised for the newer ultra-high resolutions.

Video coding standards have begun to move away from H.264/AVC to a new standard designed with Ultra-HD (4K/8K) in mind. H.265, also known as High Efficiency Video Coding (HEVC) is standardised by two main bodies, the International Telecommunication Union (ITU) as ITU-T H.265 v1 and the Motion Picture Experts Group (MPEG) as an international standard [131]. In both cases the first versions of the H.265 standard appeared in April 2013, which meant that HEVC-based live encoding and streaming was in its early stages at the time of investigating the feasibility of deploying a remote vehicle (ground or air) with a 4K video source. HEVC has since been updated to the latest revision as of April 2015 [132] with many hardware and software encoding packages now available. The video coding research community has widely adopted the challenge of working with UHD, leaving a gap in the area of H.264-based 4K transmission. This section follows an investigation into delivering live Ultra-HD using H.264 and a streaming approach similar to that of broadcast-grade HD. The aim of the investigation is to determine whether a vehicle could stream Ultra-HD footage back to the control station to provide the operators and 3D reconstruction pipelines with the highest level of detail of the remote environment without waiting for newer HEVC codecs to become available.

7.2 4K and the Broadcast Environment

4K-UHD is still in an experimental state for the broadcast industry with live UHD only just becoming available from the major broadcasters in mid-2016 [133–135]. On-line streaming companies such as Netflix, YouTube and Amazon are taking the lead in bringing 4K-UHD to consumers in their Video-on-Demand (VoD) services. The broadcast market is constantly evolving and adapting to technological changes. Consumers are ready to receive 4K content and are awaiting their transmission [136].

A live video source needs to be encoded in a relatively short time: anything more than a few minutes would be unacceptable for a live feed and will have a significant impact on the viewing experience, especially in the case of live sport. Small delays are generally expected by viewers. An example of the transmission delay can be observed by setting up two televisions, one with digital TV and another receiving the same channel over a satellite feed. The satellite feed lags a couple of seconds behind the digital television signal. Video on demand content, although streamed, is still a pre-recorded file and therefore is not restricted by encoding time.

Unlike the venture into 3D broadcasting which requires a complex change to the live production process, Ultra-HD fits into an existing broadcast environment in a similar fashion to HD workflows. Figure 7.1 demonstrates a simplistic schematic of a live UHD studio. In this example there are four different types of camera. The first is a QFHD (Quad Full-HD) camera which outputs a 4K signal over four HD-SDI lines such as a Sony F55/F65. The second camera uses HDMI 2.0 to output the live video; this can support uncompressed 4K video at a maximum of 30 frames per second. Consumer and prosumer devices typically feature HDMI ports for live output, while SDI-based output is found only on professional-grade equipment. The final camera shown in the figure is a professional Full HD video camera. A HD video signal can be up-scaled in real-time to be used as a 4K source through the use of a hardware upscaler, also known as an ‘up/down/cross’ converter. All of the video sources are

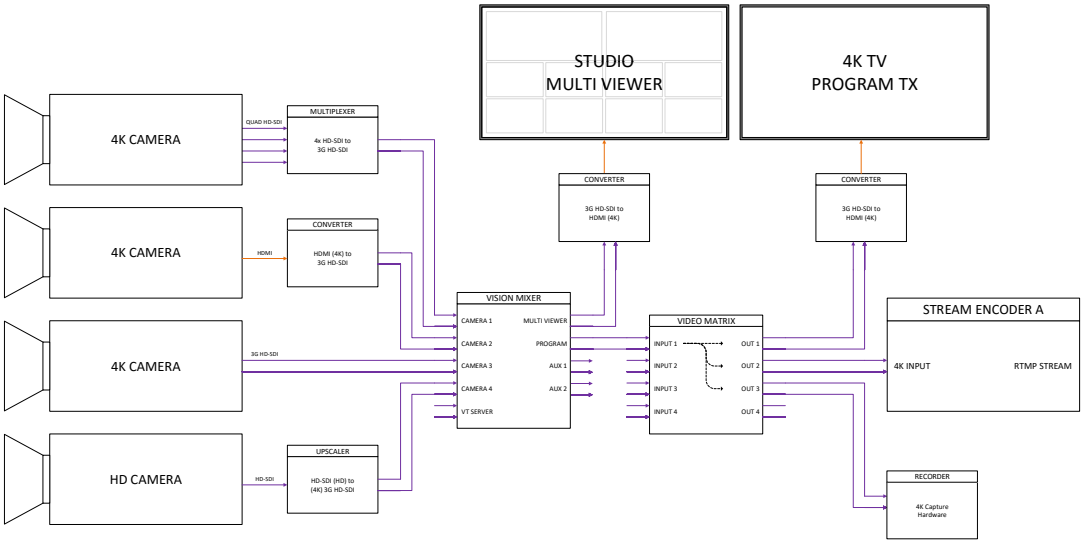


Figure 7.1: Example multi-camera, mixed format studio

fed into a video matrix which enables routing and source-sharing without additional splitters. The vision mixer is the heart of the operation. A vision mixer allows an operator (or director) to select the active camera signal that is shown (transmitted and recorded) whilst being able to see all of the incoming video signals at the same time. The mixing console also allows for additional effects to be added such as lower-thirds, also known as titles / straps. The output from the mixer is then sent across to recorders, transmitters and streaming systems.

In the example studio only two devices use HDMI. The reason behind this is the primary method of transport of video (and embedded audio) in a professional studio is SDI which runs over 75Ω coaxial BNC cable. SDI comes in a variety of standards designed to match the video standard of the equipment in use. A broadcast-grade Full-HD camera will use either HD-SDI (SMPTE 292M [137]) or 3G-SDI (SMPTE 424M [138]) which is capable of transmitting uncompressed video at 1.5 Gb/s and 3 Gb/s respectively. SDI is used in broadcast due its ability to carry high quality uncompressed video, audio and ancillary data in a reliable fashion. An SDI link is self-synchronising and has the capability to synchronise with other devices via

frame-locking, ensuring all of the signals are refreshing at exactly the same time — a crucial requirement when seamlessly switching between feeds.

To move into the UHD world, camera manufacturers offer Quad HD-SDI interfaces which produce uncompressed 4K video at 30 frames per second using four HD-SDI lines. The drawback with this approach is that a 4K broadcast camera (such as a Sony F55) requires 4 video lines to send the video signal back to the vision / transmission systems, adding 3 extra cables to run. This can be overcome by using a multiplexer which converts these 4 video feeds into 2 Dual-Link 3G-SDI video lines. Ultra-HD capture cards such as those by Blackmagic and AJA accept the 4K video over Dual-Link 3G-SDI. Newer SDI standards expand the maximum speeds for a single coaxial link up to 6 Gb/s with the introduction of 6G-SDI introduced by Blackmagic and later standardised by the Society of Motion Picture and Television Engineers (SMPTE) in 2015 [139], allowing broadcasters to return to running a single cable for the video. Newer standards are being worked on by SMPTE to support up to 24 Gb/s, enabling SDI to transmit progressive 4K at frame rates of up to 120 FPS.

The broadcast output to the streaming systems is using a 6 Gb/s bandwidth link; clearly this would be impossible to transmit to viewers. Viewers do not need the full uncompressed signal to enjoy the superior quality of the picture. In the context of using 4K on robotics, the question that is raised is how little bandwidth the signal requires in order to be of greater quality than HD and suitable for image processing tasks such as QR code discovery. The work as presented in this section focuses on the ability to ingest, compress and stream a UHD video source, whether that is a single camera or entire studio set-up, and transmit live.

7.3 Streaming Ultra-HD

The ultimate goal for this work is to be able to deliver broadcast-grade live coverage of an event in Ultra-HD. UHD places additional stress on the viewing hardware,

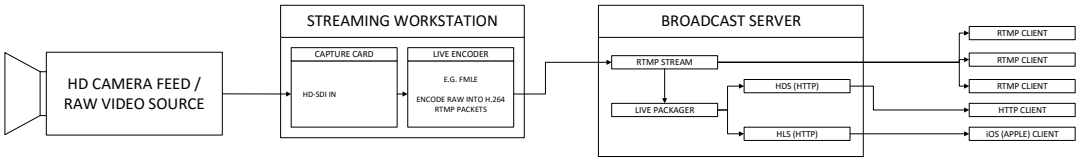


Figure 7.2: RTMP Streaming Block Diagram

requiring powerful GPUs to decode the incoming stream. A low-end GPU or software rendering would be out of the question here because of the quantity of data to be processed at video rate. Before setting target bit-rates and developing a streaming profile we first need to understand the methods the stream is 'viewed' by. Would a client be able to obtain the stream over WiFi or is a cabled connection required?

Previous work [140, 141] has shown that 4K video can be streamed over WiFi networks given the additional tuning discussed in section 7.4. In this work the author uses RTP, Real-time Transport Protocol [142], as the primary transport protocol for streaming the video from server to client; however there are newer real-time protocols (RTMP [143]) which offer better streaming support and are more commonly found in professional streaming solutions such as Adobe's Flash Media Live Encoder (FMLE). FMLE does not cap the streaming bit-rate, though the software is limited to a resolution of 1920×1080 pixels. This led to an investigation as to whether the limitation lies with the transmission protocol, broadcast server or the streaming software.

Figure 7.2 shows an overview of a typical RTMP-based streaming platform with a camera (or other SDI-based input) being ingested live before being transmitted to the broadcast server. The broadcast server serves two core purposes: streaming and repackaging content. The RTMP streaming functions handle live input and retransmission to viewers over TCP whilst monitoring each of the unicast connections. An RTMP stream contains the video, audio and meta-data including frame numbers and timestamps. Each RTMP stream will accept only one publisher, the encoder, and multiple clients all on the same TCP port, typically 1935.

The repackaging module (or 'Live Packager' as referred to by Adobe) is used to convert the RTMP stream into an HTTP-based stream. RTMP uses a bidirectional port to transmit, which may be blocked by firewalls, especially if it has been moved onto a different port. Web-based streaming uses TCP port 80, typically open to all in-bound traffic. A live packager converts the RTMP stream into video segments which are sent out over HTTP to a client. Traditional web-based streaming relies on progressive download, whereby the client will be downloading the content and watching as the video is saved. However, progressive download creates content rights issues as the viewer will end up with a complete copy of the original file on their local machine. HTTP Live streaming ensures that the client receives small chunks of video, facilitating smarter streaming (adaptive quality switching) and tighter content control.

In a benchmarking test a workstation was configured with Linux and a Blackmagic Decklink 4K capture card to evaluate the available streaming options under Linux. FFmpeg [144] is the *de facto* video processing library used in a myriad of streaming and video processing tasks. LibAV [145] is a fork of FFmpeg featuring native RTMP streaming support through the close integration with the LibRTMP project, making LibAV more suited to this work than the traditional FFmpeg framework. Initial experiments where conducted with an HD video feed to confirm that the system could connect and stream HD over RTMP via an Adobe Media Server (AMS) instance running on a dedicated server. An open-source flash-based video player is used to evaluate the streaming capability. The player is maintained by the Open Source Media Framework (OSMF) and offers full support for both live and on-demand RTMP, HTTP Dynamic Streaming (HDS) and Adaptive HDS stream playback. The total video latency as measured using a clapper board to 'mark' the start time measures at 4 seconds from the clap to it appearing on the client screen. For the HDS and HLS (Apple HTTP Live Streaming) the delay measures at 40 seconds. The HDS/HLS streams are behind the RTMP due to the repackaging. In the initial experiments the encoder, broadcast server and viewing clients are all on the same network. The broadcast server was then moved onto a separate network to test the response to

switching and routers. The round-trip time remained at 4 seconds for the RTMP and average of 40 seconds for the HDS/HLS streams based on a 2.8 Mb/s incoming RTMP stream.

Latency has a significant influence when designing video systems for use in robotics. Visual odometry and other core navigation tasks often require real-time imagery to feed into the control systems. Live control updates are restricted by the video-rate and transmission delay. Latency must be minimised to achieve fluid / seamless control. Direct control via tele-operation (remote control based on video feeds and sensor data) is a common approach to monitoring and interacting with a robot to react to dynamic events or hazards. In some cases latency is unavoidable for remote operation and vision-based navigation must run using on-board processing. An extreme example of this can been seen in the NASA Jet Propulsion Labs (JPL) Mars rover projects, where the full day's list of control instructions, known as 'the mission' are transmitted to the robots in one block once a day. The imagery is then downloaded back from the rover for review. The transmission can take between 10 to 20 minutes, therefore direct remote control over the rover's movements are not possible. Fortunately, robotics deployed on Earth do not suffer from these magnitudes of delay, however video transmission can still reach into the hundreds of seconds, posing a challenge for live remote robotic control.

Ultra-HD content is typically compressed to 18 Mb/s for services such as Netflix and Amazon, however this is still far too high for many consumers to be able to watch due to narrow bandwidth. The UK average broadband speed was 14.7 Mb/s [146] at the time of testing and has since raised to 22.8 Mb/s [147, 148]. Although the national average has risen, the speed across sub-urban and rural areas has remained relatively flat. The increase in average speed is due to the large-scale deployment of fibre to the cabinet (FTTC) and fibre to the premises (FTTP) within cities leaving rural areas behind at an average of 10 Mb/s. Given these figures, a target of 8 – 16 Mb/s was set, allowing for a range of 4K streams which UK-based viewers would be able to tune in to.

7.4 Pushing H.264

H.264 is split into 5 core levels. Each level determines the maximum video bit rate, frame size, and frame rates. Levels 5.1 and 5.2 (highest level) introduce 4K resolutions to the standard. In order to stream live 4K, the encoder needs to be tuned with the content, resources and viewing platforms in mind. FFmpeg and LibAV both use live presets which tune the encoding parameters to reduce computational cost and adjust compression quality. There are 10 presets: ultrafast, superfast, veryfast, faster, fast, medium, slow, slower, veryslow and placebo. Ultrafast offers the fastest encoding option preferring encoding speed over compression quality. Placebo offers the best encoding and compression quality which outputs the smallest file sizes for a given source. Placebo, slower and slow are the most computationally expensive presets and are not possible for use in live encoding (even for sub-high-definition content). When choosing a preset for live encoding there is a range of factors which needs to be assessed during testing, including frame-rate, CPU utilisation, RAM usage and network bandwidth.

During testing, one workstation was set up with a 16 core CPU (8 Core with Hyper-threading), 24 Gb RAM, Nvidia Quadro FX-series GPU and a Blackmagic 4K Decklink capture card. The ultrafast preset was used to facilitate live UHD at 25 frames per second. Using the superfast preset lowered the output frame rate to 20 frames per second whilst maxing out the CPU cores. If the encoder falls behind the live input the capture card begins to place the raw frames in RAM, which quickly fills up and causes a critical failure. For the large-scale test, three encoding workstations will be used.

Although the stock ultrafast preset will get a live UHD stream running, the output video will be challenging to watch, especially over wireless connections. Group of Pictures, GoP defines the encoding pattern for the H.264 encoder. H.264 sequences are formed of a sequence of I, P and B frames. An I-frame is a complete 'reference' frame which contains a complete copy of the image data. I-frames can

be considered as 'key-frames'. P-frames are prediction frames containing motion vectors and transform coefficients which predict where the content is moving to. Using P frames reduces the amount of data required to encode a frame, especially in dark areas and scenes with little or no movement. B-frames are also prediction frames however are bi-directional which allows the frame to refer to either the previous or next I-frame, whereas the P-frame can only predict motion based on the last I-frame. The order in which the sequence of I, B and P frames are encoded is determined by the encoder parameters. In [149], Adeyemi-Ejeye explored GoP structures for streaming ultra-HD over various wireless network environments, resulting in an optimal I-frame interval value of 40. It was also deemed necessary to disable B-frames as these are computationally expensive.

Small-scale laboratory experiments confirmed that the combination of these components and parameters produce a stable UHD stream. Through a series of trial and error tests it was discovered that the system could stream live 4K at a wide range of target bit-rates ranging from 8 Mb/s through to 20 Mb/s. Anything over 20 Mb/s became unstable with random buffering occurring at the client end. FFMpeg and LibAV both use Adaptive Bit-Rate (ABR) encoding strategies. ABR generates a smooth stream with a variable output bit-rate based on the content. For example if the source becomes dark (or black) the packets are smaller than those with complex colour. Constant Bit-Rate (CBR) encoding would encode every packet at the same rate which wastes bandwidth and computation time. ABR enables one to set a target rate, effectively a bandwidth limit to the encoding stream.

7.5 Live Ultra-HD Graduation

After a series of small-scale tests to confirm that the workstations and servers could cope with the workload and would remain consistently stable over long durations (24, 48 and 72 hour test runs), it was decided that the best way to demonstrate the effectiveness of the approach was to stream an entire university graduation series

in Ultra-HD, available globally. The University of Essex graduation week is formed of 11 ceremonies, lasting up to 2 hours each with a short break between sessions. Each day sees 3 ceremonies (with the exception of 2 on the last day). The University streams a live HD multi-camera broadcast of each ceremony which is viewed by a world-wide audience. The University's Information Systems Services (ISS) department offered support to enable a Ultra-HD trial to operate alongside the live HD service. Figure 7.3 shows the camera layout of the main venue. Unlike the HD service, only one 4K camera was available, restricting the camera placement. The UHD camera was mounted to a remotely operated pan & tilt head and placed in the centre of the venue to overcome what would be a fairly uninteresting static shot of the event. Fortunately the building infrastructure has two SDI lines run to strategic locations within the auditorium and two control rooms. The UHD camera is a Sony AX1 camera which outputs the video over HDMI. A BlackMagic HDMI to SDI 4K converter was required to pass the video through the building to the control room. The building's cable run is just within the limits of the 6G-SDI specification enabling the 4K Graduation test to go ahead.

The SDI signal was routed through three workstations, each responsible for outputting one stream. The three streams were set to 8 Mb/s, 14 Mb/s and 20 Mb/s. The aim was to demonstrate the steps in quality from a HD video stream to a 4K video stream. 20 Mb/s offers much deeper colours and finer encoding at the cost of higher bandwidth. It was anticipated that most viewers would be watching at 8 Mb/s with viewers on site opting to view the higher rate streams. The Ultra-HD streams proved to be a success with over 300 unique viewers tuning in from 35 countries across the world. The majority of the viewers were from within the UK (35%), however viewers were watching from Europe, Middle East, Asia, Australia and America.

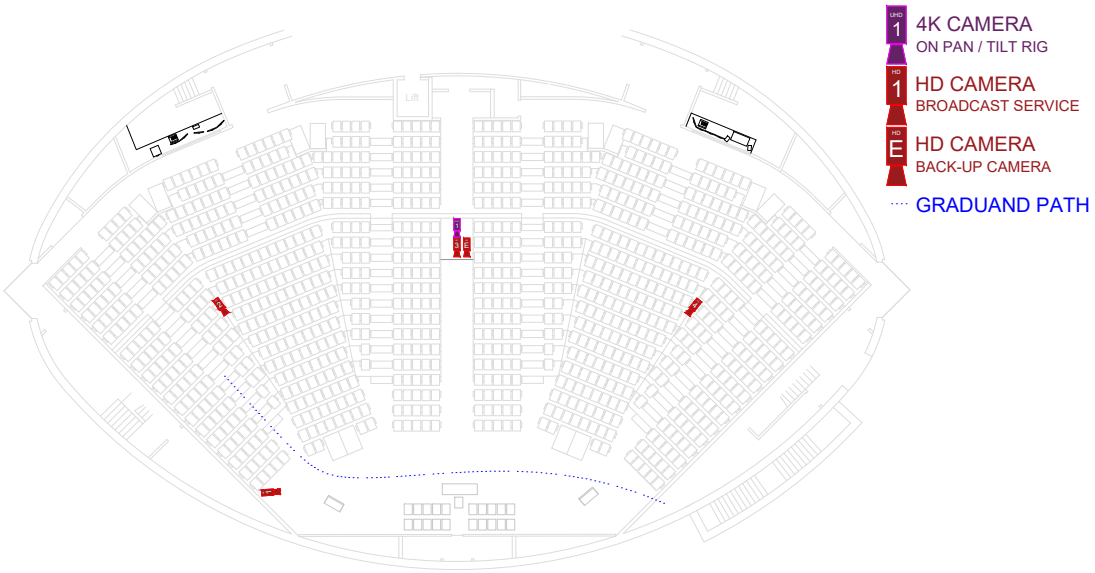


Figure 7.3: Graduation Broadcast Venue

7.6 Adaptive Delivery

Offering fixed streams where a viewer chooses which stream to watch relies on the viewer understanding their available bandwidth and is unable to react to changes in network conditions. Fixed streaming is a simple way of offering a streaming platform, however being able to provide streams where the player can automatically switch based on client's available bandwidth is a significant improvement to the overall experience, especially to end users. The player used in the graduation service provided clients with an option to select either 8 Mb/s, 14 Mb/s or 20 Mb/s. Adaptive streaming is widely used in web-based video platforms for both live and on-demand content. An adaptive service profiles the connection speed between a client and the server. The broadcast server will then select the appropriate quality for the current connection speed. In the case of Adobe Media Server (AMS) and Woza based-streaming platforms the connection is constantly monitored so that the stream can react to changes in network quality. The Graduation service used 3 encoding workstations to generate 3 streams being handled by an AMS server. To facilitate adaptive Ultra-HD a powerful 96-core server is deployed to provide live transcoding. Figure 7.4 shows an overview using a single encoding workstation providing a high-quality 20 Mb/s stream which is split into the various streams to support a fully adaptive streaming service covering standard definition (for mobile viewing) all the way up to Ultra-HD.

The diagram shows an 'entry server' which sits within a high-speed network labelled as a Transcoding Network (TN). The transcoding network can sit within the same machine or scaled up across multiple machines and even across multiple cloud servers depending on the available bandwidth. The encoding workstation produces a single high-quality feed into the RTMP entry server. Each of the transcoding nodes are then spun-up to pull the feed from the entry server. Each node also contains an RTMP server which is used to reduce the bandwidth between the entry server and the transcoding nodes. Each node is configured to transcode a series of substreams and is assigned a target bit-rate and frame size and a limited number of CPU

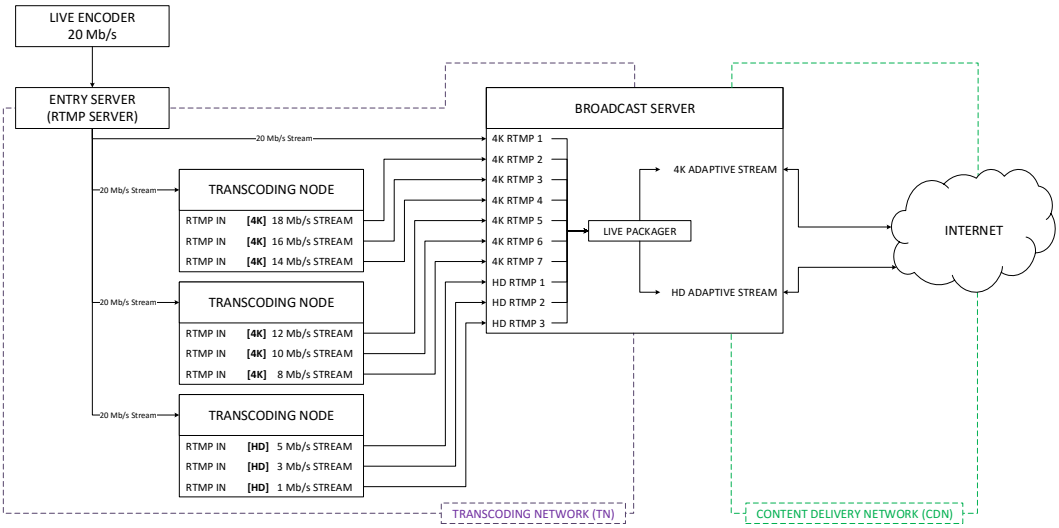


Figure 7.4: Adaptive RTMP/HDS Streaming Configuration

cores. Each of the transcoding nodes stream their RTMP video across to the broadcast server which then serves the content on to viewers. This approach enables one workstation to generate all of the require sub-streams necessary for adaptive delivery.

All RTMP streams are unicast TCP connections which ensures that the video packets always arrive in the correct order and in time. The broadcast server relies on the TCP protocol to guarantee that the packets are in-order for te seamless rate switching. The drawback to unicast connections is that each transcoding node will request a new copy of the 20 Mb/s feed which increases the load on the network infrastructure within the transcoding network. In the experiments a range of servers were utilised as transcoding servers ranging from 16-core workstations up to 96-core enterprise servers. The broadcast server used during the Graduation service has 24-cores and 96 Gb RAM. Later experiments showed that a 4-core machine could handle the 4K RTMP service.

7.7 Concluding Remarks

Experimentation identified resolution as a critical limitation of vision-based systems. Transmitting Ultra-HD imagery is an extremely computationally expensive task. The results of the Graduation broadcasting experiments demonstrate that 4K-UHD can be transmitted live over TCP/IP using H.264 encoding at a range of data rates scaling from 8 Mb/s – 20 Mb/s just within the range of existing wireless networks. Adaptive delivery methods enable a client to keep up with the live feed even in dynamic and variable bandwidth environments. Although the work proves that 4K can be transmitted using H.264, the computational demands are too high for a robotic platform. Newer research-grade aircraft such as the AscTec Pelican feature Intel i7 processors with USB 3.0 support which could lead to the possibility of transmitting UHD from the air using the newer H.265 standards. The newer encoders require less bandwidth, making them more suited to transmitting Ultra-HD imagery over wireless networks, expanding the applications of vision within robotics.

CHAPTER 8

Systems Integration

One of the principal aims of this thesis has been to understand the challenges of deploying visual 3D reconstruction techniques, namely Structure from Motion, in robotic systems. The thesis so far has tackled the individual components, targeting key areas which would make them more suited to an autonomous system. This chapter combines all of the previous findings into a single system to perform a complete remote 3D reconstruction. Given the results seen previously, can the components work together to produce a vision-only robotic control system with the goal of reconstructing an unknown environment?

8.1 Experimental Set-up

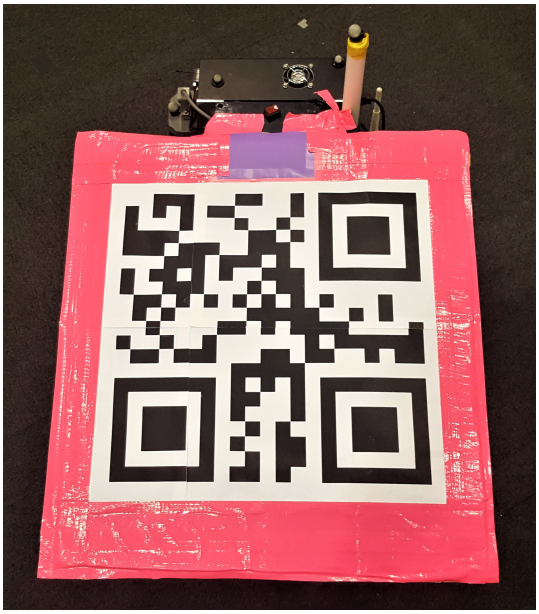
Continuing on from the custom implementation of the robotic communication system in section 6.2.2, the prototype system is formed of three core components: an aerial observer, ground vehicle and a Ground Control Station (GCS). The ground control station forms the main hub of the system as it receives positioning information from the aerial systems and in turn passes this information to high-level path planning tasks before relaying raw positioning data and control down to the ground

vehicle.

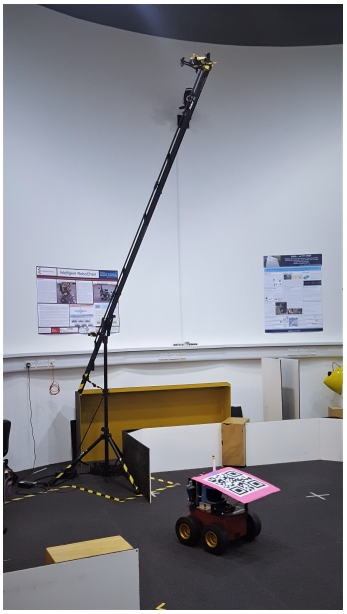
The 'position feedback' module within the GCS forms the core backbone of the overall system. This receives live position data from the aircraft handler alongside Vicon pose data. The Vicon motion capture system is used to record the ground truth for later analysis and is not used during the live work. An arbitrary anti-clockwise circular path was generated to drive the ground vehicle, monitored by the aircraft's visual tracking.

A method for identifying and locating robots using fiducials was presented earlier in Chapter 6, whereby QR codes are attached to the upper surface of ground vehicles and targets. However, the working altitude is limited by the optical resolution of the video feed transmitted from the aircraft. In the case of both the Parrot AR2 and Parrot Bebop the maximum working altitude is just over 1 metre from the 170 mm target used in previous experiments. Increasing the size of the QR code to 370 mm to match the width of the vehicle provided only a small increase in working altitude. The height of the camera system directly impacts the working area in which the ground vehicle can operate. Using the larger QR offers a working area of 4.1 m × 2.2 m, too narrow a space in which to move the ground vehicle. To enable aerial tracking from a UAV at a sensible altitude, an alternative solution is used to aid QR tracking by tracking a border colour. The QR code was surrounded with a bright pink border with a second square located at the centre of rotation, as seen in figure 8.1a. this enables the aircraft to track the pose and location of the ground vehicle in real-time beyond the QR readable altitude.

One approach to overcome the given hardware limitations would be to use the QR for vehicle and target identification when the aircraft flies at a low, 'inspection', altitude. Once the target and ground vehicle are identified, the aircraft then uses reference colours for colour tracking, enabling the aircraft to rise up to an 'operational' altitude and providing a significantly larger working area on the ground. In these experiments, testing revealed that using the colour tracking approach increased the visible ground work area to 6.3 m × 3.5 m whilst maintaining a stable lock on the



(a) Tracking Target: QR and Colour Border



(b) Bebop fixed to a static altitude of 3m

Figure 8.1: Air & Ground Vehicle Set-up

target vehicle. For the purpose of these experiments, the aircraft was mounted on a camera crane as seen in figure 8.1b and therefore the visual processing module within the aircraft handler used only the colour tracking method to provide the location and pose of the ground vehicle. The working altitude of the aircraft remained fixed at 3 m.

In previous work the ground vehicle was controlled using ROS with the aid of ROS-ARIA, a ROS bridge module which provides control to the Pioneer 3-AT (P3-AT). ROS is no longer used in this work as discussed earlier and thus a custom controller was written with the aid of the ARIA SDK framework to drive the vehicle in either manual (direct operator control) or autonomous mode, wherein the vehicle will drive to a given waypoint using the pose information provided from the ground control station. The controller requires a stable, regular stream of data from the GCS in order to smoothly drive autonomously. During development it was observed that standard wireless networking was too unreliable using the 2.4 GHz wireless

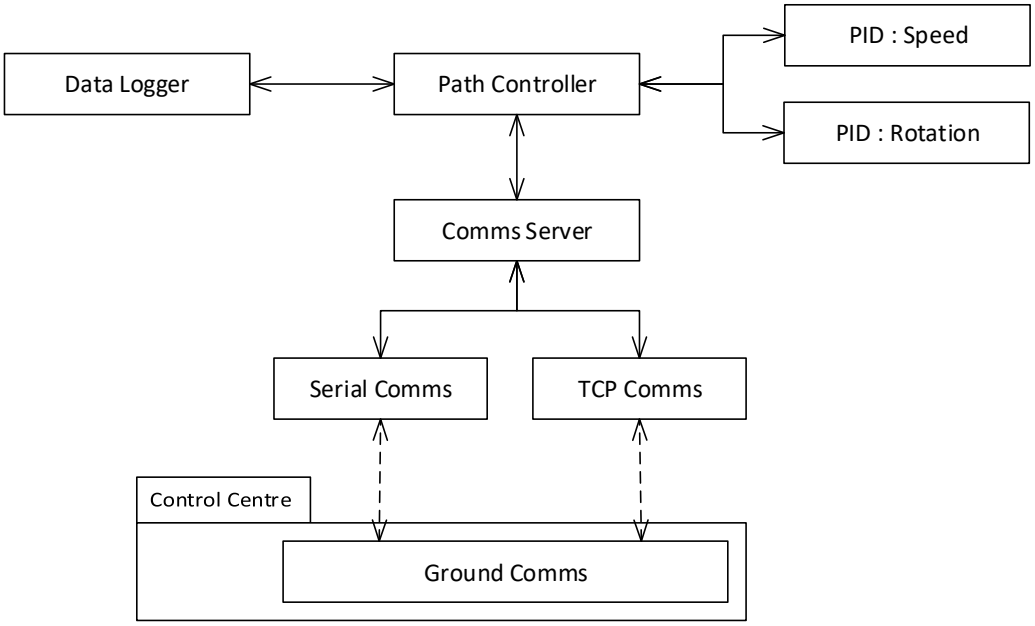


Figure 8.2: ATLAS Ground Vehicle — Component diagram

hardware: the robot experienced frequent drops in connection due to the congested nature of the institution's wireless infrastructure, even with a dedicated wireless access point to the point where standard SSH (remote terminal connection) was experiencing packet loss. Using an Xbee wireless 2.4 GHz serial module enabled a stable link to be maintained between the on-board computer and the ground control station. The Xbee modules provide a simple serial tunnel running at a baud rate of 115200 bits per second. The ground vehicle was able to navigate autonomously to waypoints even with a complete wireless connection drop, thanks to the dedicated wireless serial link. A watchdog was implemented into the ground vehicle to stop it if the data stream from the GCS suddenly stopped, to prevent a 'run away' or crash of an out-of-control vehicle.

The whole system uses a redesigned communication protocol based off of the control protocols used in Chapter 6. The system has been extended to work over TCP, UDP or serial link. The protocol uses JSON to exchange the messages and is grouped to ensure consistent transmission. The serial link can be flooded, whereby the packets collide and cause sync errors. The protocol has been tuned to fire every 50ms; this serves two purposes, the first being that all of the variables are transmitted in one block and the second is that the messages act as a timing pulse, akin to a 'heartbeat', whereby the connection can be monitored based on the frequency of packets. Should there be a link error, then the system can react before waiting on an OS-level link error to be presented. Table 8.1 describes the messaging protocol between the GCS and the ground robot. The messages are transmitted at a rate of 20 Hz. Table 8.2 shows the message structure transmitted from the Bebop controller to the control station. In the trails the applications run on the same workstation, however they could be run across multiple machines via an Ethernet or wireless connection.

The first version of the messaging protocol was designed to be fully event-driven, with messages being sent on demand. Messages were prefixed with a series of keywords to trigger actions, such as *{ABS x000 y000 rz000}* to update the global position. This method caused significant overhead due to unscheduled message tim-

Table 8.1: Inter-Control Communication Protocol: Ground Vehicles

Key	Data Type	Description
SMBL	Boolean	Left Mouse Trigger (Used to enable/disable motors)
SMBR	Boolean	Right Mouse Trigger (Used for Emergency Stop)
LinX	JSON Number	Manual drive, motor speed (forward)
RotZ	JSON Number	Manual drive, rotation amount (+/-)
Px	JSON Number	Vehicle pose, x
Py	JSON Number	Vehicle pose, y
PrZ	JSON Number	Vehicle rotation, rZ
WPCx	JSON Number	Current waypoint, x
WPCy	JSON Number	Current waypoint, y
WPNx	JSON Number	Next waypoint, x
WPNy	JSON Number	Next waypoint, y

Table 8.2: Inter-Control Communication Protocol: Aerial Feedback

Key	Data Type	Description
GV1x	JSON Number	Ground vehicle 1 pose, x
GV1y	JSON Number	Ground vehicle 1 pose, x
GV1r	JSON Number	Ground vehicle 1 rotation, r
GV2x	JSON Number	Ground vehicle 2 pose, x
GV2y	JSON Number	Ground vehicle 2 pose, x
GV2r	JSON Number	Ground vehicle 2 rotation, r
T1x	JSON Number	Capture target 1 pose, x
T1y	JSON Number	Capture target 1 pose, x
T1r	JSON Number	Capture target 1 rotation, r

ing and often resulted in the messaging stack collapsing due to timings and serial collisions. The revised version, shown in tables 8.1 and 8.2, transmit the full parameter set in one message, utilising mutexes to ensure consistency and prevent the stack from crashing. The updated methods also result in clearer, more concise C++ code due to the message packing with JSON. Listing 8.1 shows the readablity of the syntax. This is called every 50 ms by the communication thread to transmit the required data down to the ground robot.

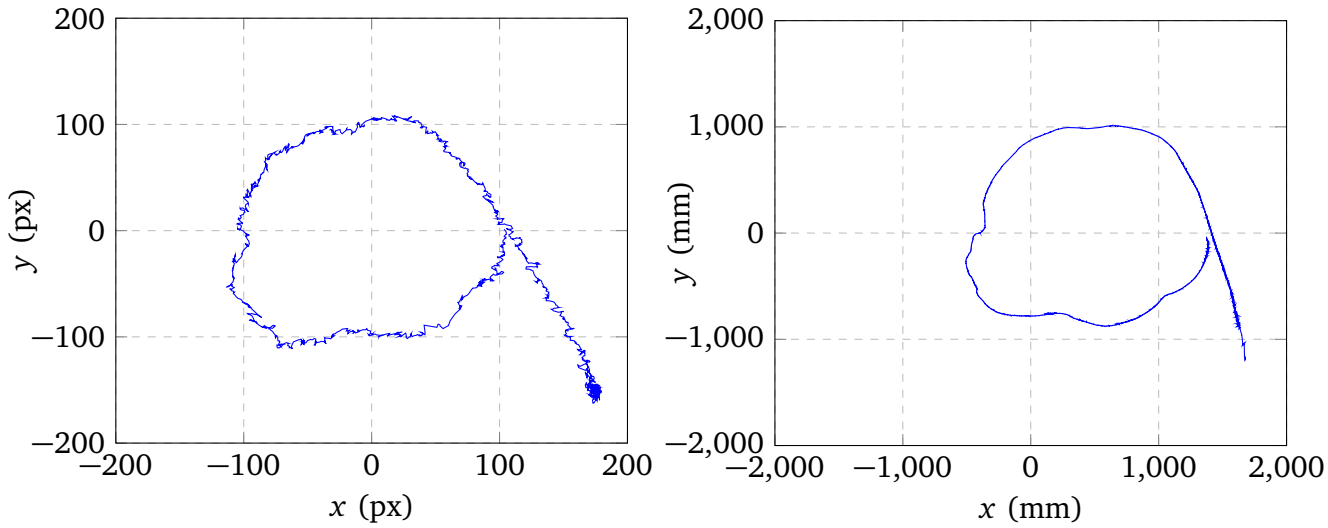
Listing 8.1: Transmitting with JSON Messages

```
// Generate a JSON string from the current data
json11::Json jsonObject = json11::Json::object
{
    {"SMBL",  m_smButtonL},
    {"SMBR",  m_smButtonR},
    {"LinX",  m_manualDriveLinear},
    {"RotZ",  m_manualDriveRotation},
    {"Px",    m_latestPose.x},
    {"Py",    m_latestPose.y},
    {"PrZ",   m_latestPose.r},
    {"WPCx",  m_currentWayPoint.x},
    {"WPCy",  m_currentWayPoint.y},
    {"WPNx",  m_nextWayPoint.x},
    {"WPNy",  m_nextWayPoint.y},
};

// Send JSON String
m_comms.send(jsonObject.dump());
```

8.2 Results

Figure 8.3 shows the motion of the ground vehicle driving under autonomous control using only the pose information provided by the aircraft handler. The GCS defined a circular path about the centre of the aircraft's camera. A Vicon 3D motion capture system recorded the true motion of the vehicle as it navigated the path. To facilitate more complex path-planning techniques in future work, the ground vehicle



(a) Visual Pose Tracking

(b) Vicon Ground Truth

Figure 8.3: Visual Ground Vehicle Position Tracking

receives only its current pose, its last waypoint and its next waypoint: the robot has no knowledge of the intended path. More sophisticated control algorithms such as PID or fuzzy logic controllers could be implemented to produce a smoother motion about the arc. However, for a simplistic controller, the robot is able to drive in a circle around the environment based purely on commands generated by the drone's view of the environment.

The ground vehicle was fitted with a GoPro HD camera which provides a live view of the motion. The live video frame extraction process described earlier in Chapter 5 was used to convert the incoming video into individual frames. The run took 1.5 minutes to complete, which would produce 93 frames if all frames are used prior to filtering. With such a small number of frames, a full dense reconstruction is possible within 45 minutes on a high performance mobile workstation equipped with a quad-core i7 (with hyper threading providing 8 logical cores). Figure 8.4 shows the resulting 'ultra' quality model produced using AgiSoft PhotoScan. The model contains 13.1 million 3D RGB points. Figure 8.5 shows the resulting point-cloud from above as would be seen from the aircraft. The white patch of missing

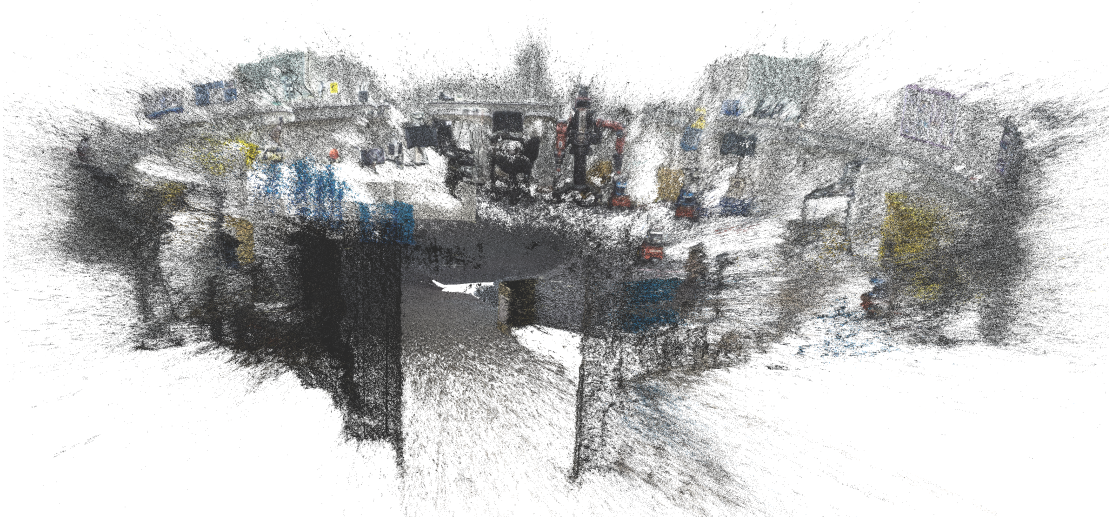


Figure 8.4: Completed Dense Reconstruction

data located just off-centre of the arena shows where the robot circled under the control of the aircraft. The model reveals the circular nature of the robotics arena which has been used throughout the experimental verification chapters of this thesis.

The point cloud still requires further processing to reveal a clear model of the environment. In previous chapters this process was done manually within the reconstruction software packages. However, through the use of the Point Cloud Library (PCL) [150] and CloudCompare [108], the dense point cloud was refined using a Statistical Outlier removal Filter (SOF) which reduced it to 12.6 million 3D RGB points and revealing a more detailed model of the capture environment, as seen in Figure 8.6.

The 'clean' dense point cloud provides a high degree of detail. There is a wide range of 3D data storage formats ranging from a simple X,Y,Z,R,G,B csv-style format through to more complex storage formats such as the Polygon File Format (.PLY), an open geometry definition file format (.OBJ), to application-specific formats such as AutoDesk Drawing Exchange Format (.DXF), and 3D Studio Max (.3D3). The

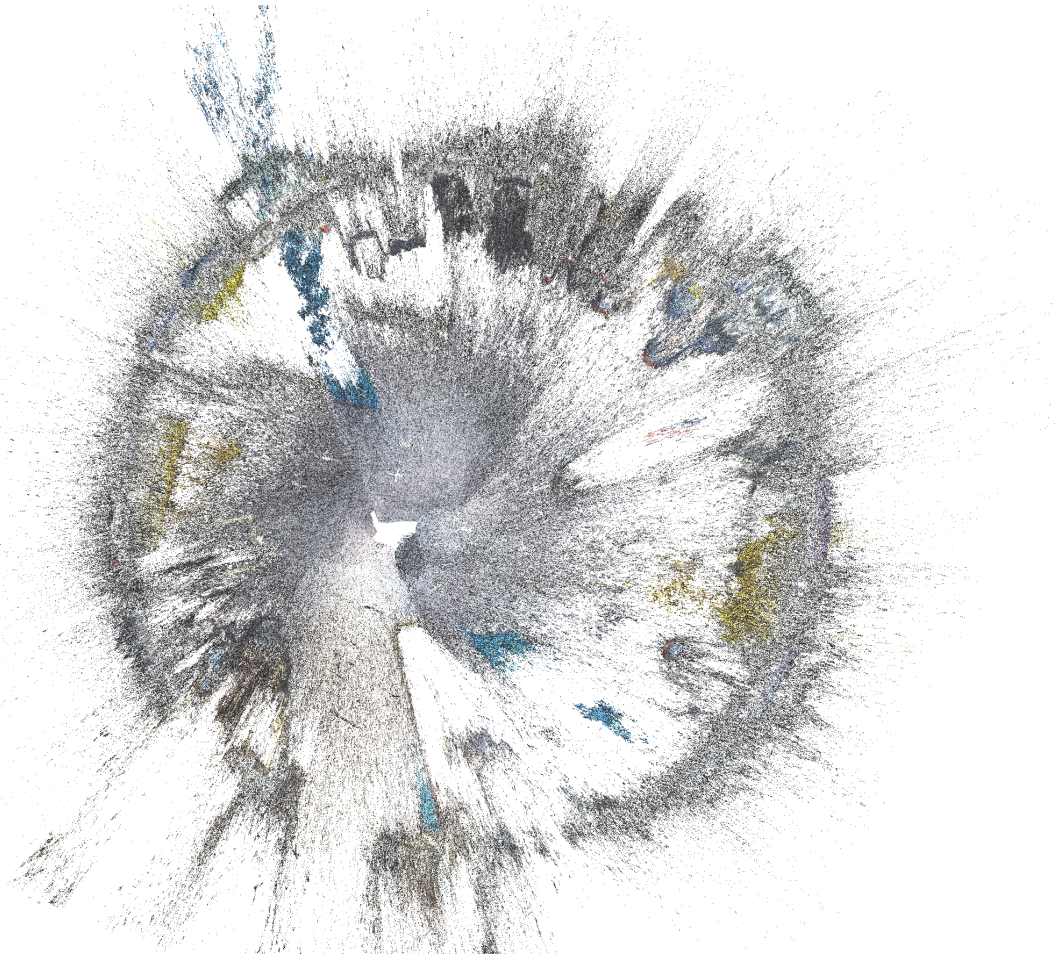


Figure 8.5: Experimental Overview

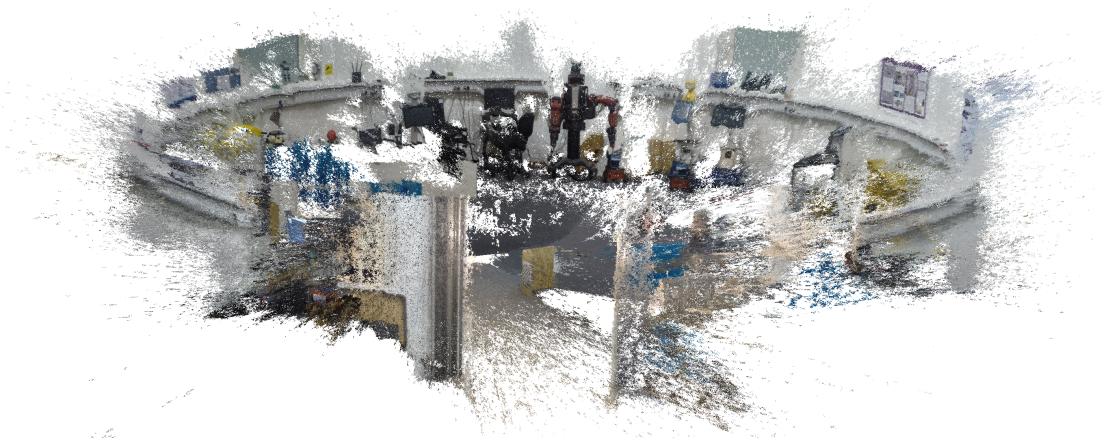


Figure 8.6: Dense Reconstruction After SOR Filtering

decision on which file format to export to depends on the target application. The simple XYZRGB format works well in robotic applications as there is no requirement for any of the extra information that is typically stored in the more advanced file formats. Some 3D reconstruction software, such as AgiSoft's PhotoScan, supports exporting to 3D PDF which enables models to be shared with non-technical users, although a 3D PDF is useful only for visualisation. A 3D reconstruction can also be printed via specialist 3D printers, however these devices require the point cloud to be converted to a solid mesh before being accepted by the printing systems. CloudCompare and other PCL-based tools enable designers and engineers to convert 3D point clouds into meshed-objects. Figures 8.7 and 8.8 show the conversion from the clean dense point cloud through to a meshed model of the robotics arena.

Further post-processing techniques such as Laplacian smoothing for noisy meshes [151, 152] can be applied to these mesh outputs to reduce noise, yielding a cleaner output. This can be seen when comparing the top-down view of the meshed models before (figure 8.8) and after the filter has been applied (figure 8.9).

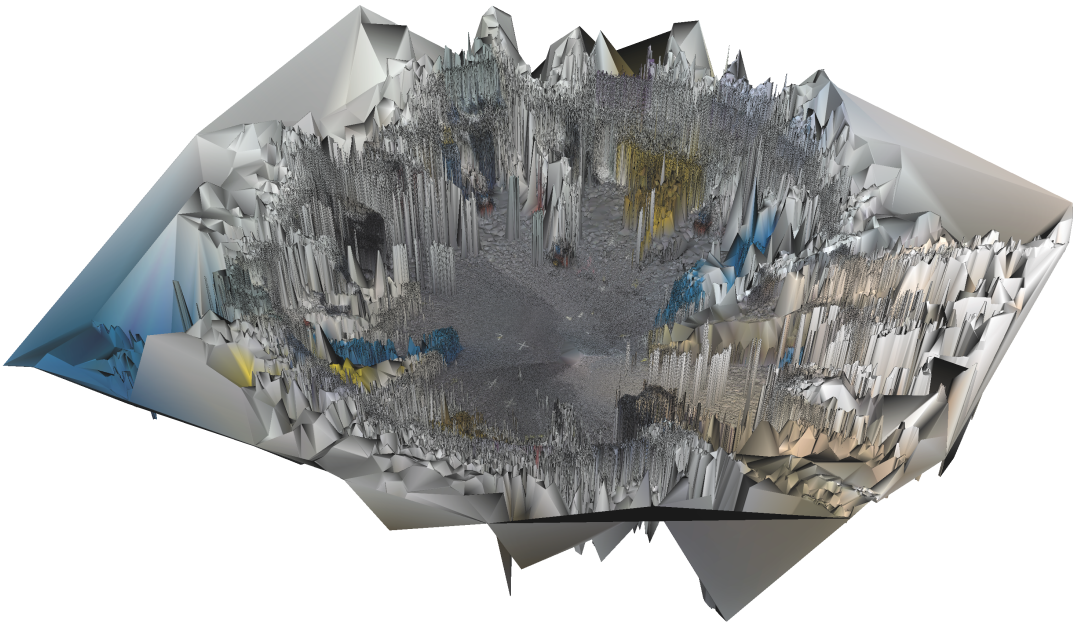


Figure 8.7: 3D Mesh generated from point cloud data

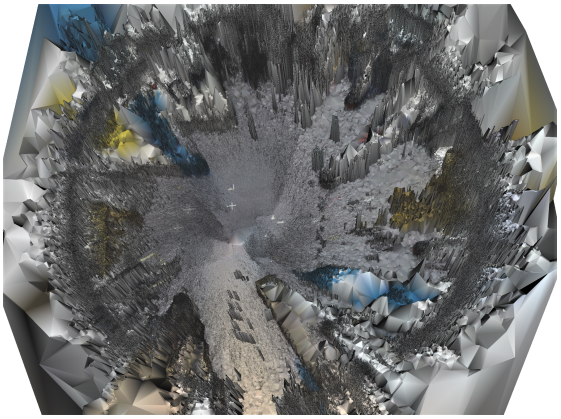


Figure 8.8: Top-down view

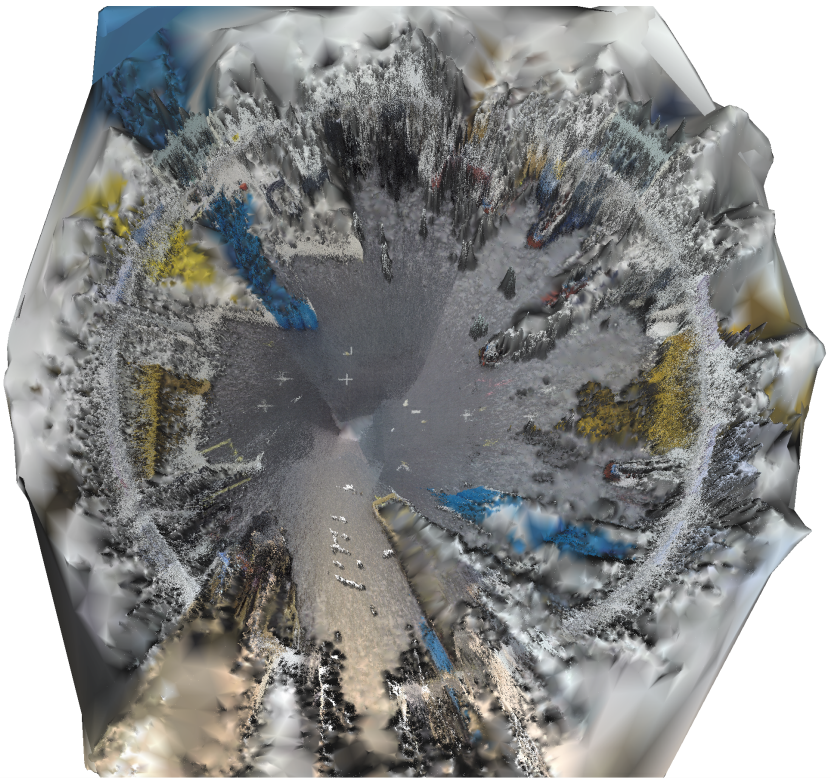


Figure 8.9: Smoothed 3D Mesh

8.3 Concluding Remarks

This chapter has presented a vision-only robotic solution capable of creating a 3D reconstruction of an unknown environment. The system utilises work from previous chapters to support the aerial observation and frame selection strategy. Although the aircraft remained static during the experiment, the ground vehicle was autonomously controlled using the positioning data provided by the aircraft's camera feed. A strategy for overcoming the altitude limit imposed by the stream resolution increased the ground work area by a factor of 1.5. The software components communicate over local serial pipes and the ground control system communicates over dedicated hardware with the ground robot to overcome the difficulties experienced when working with standard wireless networking hardware. Each of the vehicles can operate independently of the system whilst being able receive commands to contribute to the collaborative mission. The whole system has been implemented in C++ and is formed of 10,277 lines of code across three core applications: a drone handler (5,377 lines), robot controller (833 lines) and the central mission controller (4,067 lines), taking approximately six months of intensive development work to complete. The 3D models generated in this chapter show a successful reconstruction of the robotics arena. The modelling is taken to the next stage by converting the point cloud in to a meshed model. The meshing stages are beyond the scope of this thesis, as a meshed model is not necessary for robotics.

The next chapter draws the thesis to a close, reviewing the research that has been presented throughout this thesis and tying the contributions back to the original aims and objectives.

CHAPTER 9

Conclusions

This chapter draws the thesis to a close through a discussion of the key elements of the work with reference back to the relevant literature and research objectives. Section 9.1 discusses the main research, followed by a discussion on the limitations of this work in Section 9.2. Section 9.3 presents ideas for future work which would build upon the research presented in this thesis. Section 9.4 provides some closing remarks.

9.1 Discussions

The primary aim of this work has been to address the practicalities of using 3D reconstruction techniques in robotic applications. The literature discussed in Chapter 2 demonstrates that 3D models have an important role to play in a range of disciplines, from medicine through to engineering. A wide range of sensors has been discussed, ranging from active ones which involve sending out a signal into the environment and measuring its response, through to passive sensing which merely observes the environment. This thesis looked specifically at modelling the real world through the use of computer vision, in particular using Structure from Motion, a monocular

passive 3D sensing technique. The literature review then focused on the relevant robotics aspects which would be necessary in developing a collaborative robotic capture platform, discussed in Chapters 6 and 8.

Though this is at first a seemingly large task, once the layers are peeled back and the individual elements exposed, it can be seen that vision-based reconstruction systems can provide detailed and accurate models of the real world. The experimental results presented in Chapter 3 confirm the theory of Structure from Motion. Structure from Motion is a well established algorithm, so algorithmic advances have not been a focus of this work; rather the aim has been to understand practical issues of using Structure from Motion and obtaining imagery for 3D reconstruction. Chapter 4 followed on from the controlled experimental work presented in Chapter 3 by applying real-world imagery through a series of 3D reconstruction tasks of increasing complexity. The initial experiments verified the accuracy of the reconstruction technique, confirming that a Structure from Motion based reconstruction is correct up to an unknown scale factor. Once a known dimension is applied to the resulting models, a full set of measurements can be made by scaling the full model. The large-scale experiments such as the university buildings and Elmstead's church highlight some of the drawbacks of Structure from Motion, in that the large numbers of frames lead to significant increases in processing times: even with the aid of powerful computational graphics cards the church took over two weeks to produce the final results. The work thus far has shown that high quality, and even photo-realistic modelling, is possible with the selected methods; however work is required to make 3D reconstruction feasible in a more timely fashion.

In an attempt to reduce the computational cost of Structure from Motion, a more intelligent frame selection algorithm was devised [4] in Chapter 5. The frame selection algorithm seeks to make video sources suitable for application in 3D reconstruction. Frames extracted from video often present a range of technical faults, making them unusable in Structure from Motion. These flaws include large numbers of identical images (where a camera is stationary), sudden movement with varying baselines and blurring caused by movement. To tackle these problems a multi-stage

approach of filtering has be created. The first of these is to reject any blurred frames with the aid of the variance of the Laplacian. Each frame is then sub-sampled (for speed) before measuring the coherence between frames using feature-matching akin to the methods used in Structure from Motion. This produces a mechanism that allows useful frames to be identified in real time. This novel approach also benefits users of existing / commercial 3D reconstruction systems by allowing them to use video cameras as capture sources.

Attempting to understand and solve the problems associated with remote reconstruction has driven this research in a variety of areas including parallel / distributed computing, Ultra-HD video streaming, 3D modelling and robotic control systems. Each of these research areas seemed at first to be isolated from each other, yet as the project progressed and the components began to take shape it became clear that the fields of computer vision, graphics and robotics are intertwined together and provide an array of exciting research opportunities.

The models produced throughout this thesis demonstrate the level of detail that can be recovered from purely 2D imagery. Robots often have access to specialist sensors such as scanning lasers, depth sensors and GPS to provide accurate information about the environment and their own locations within it. Using vision alone offers new approaches to localisation in environments where external sensors are unavailable or blocked (as in the case of GPS). Vision-only is not a replacement to these sensors though experiments in Chapter 8 demonstrate that a vision-only system remote robotic 3D reconstruction system is feasible in princliple, and high quality models are possible, however using the cameras in real drones to locate and identify ground robots is difficult.

The nature of current robotics means that the 3D reconstructions must be done off-robot, but moving imagery from real robots using real wireless networks is a major problem. Getting live HD (720p) footage from robots to processing workstations proved to be extremely difficult. Even in the case of working with the Parrot Bebop, which features a 1080p sensor, the actual live images received over the live link top

out at 640 × 350 pixels. A change in networking hardware from 2.4 GHz (up to 450 Mb/s or 600 Mb/s) to 5 GHz (up to 1300 Mb/s) wireless slightly improved the frame quality but not its resolution. A 4K camera can provide over four times the resolution of the HD cameras; but transmitting these large frames requires a vast amount of processing. Working with ultra-high resolutions such as the images from 4K video sources led to the investigation of real-time streaming of live Ultra-HD using existing technology as the newer HEVC (H.265) codecs had not been fully developed at the time of experimentation. The work presented within chapter 7 defined a method of live streaming 4K-UHD video using H.264 and RTMP streaming protocols. The method showed that a latency as low as 4 seconds could be achieved at a range of bit-rates from 8 to 18 Mbps. While this is too slow for live robot control, these images can be used for building detailed 3D models.

Until this research [2], a global web-based Ultra-HD H.264 broadcast had not been attempted before, leading to the publication at the International Broadcast Convention in 2014. The newer HEVC encoders/decoders should give a significant improvement in video coding and will eventually filter down to the lower power devices needed for robotics. The most significant challenge faced when working with high resolution content is the high computational demand on the hardware. The hardware used in the 4K broadcasts would be impractical on a mobile platform. Live high resolution video encoding proved to be unsuitable for use on a robotic platform such as the ground vehicles and impossible on a small air vehicle: the encoding process requires an Ulta-HD (or Quad-Full HD, QFHD) capture card and powerful CPUs. In the live broadcasting of the University's graduation events we actually warped two of the 4K capture cards due to the intense heat being generated by the high-end workstations.

The Robotic Operating System, ROS, is a popular choice amongst robotic researchers due to large number of modules which can connect to the architecture to enable prototyping for new systems. ROS aims to be a generic framework which enables researchers to focus on specific tasks and then utilise pre-made software components such as path planning, positioning, low-level motor controllers, etc. to make

the rest of the robotic system. In principle this is an attractive system; however in practice the publish / subscribe model is poorly matched to a distributed system. The ROS network can work across real networks but require high-bandwidth Ethernet links. The systems developed during this work through chapters 6 and 8 use a custom protocol which works over TCP, UDP or serial links. The crucial advantage to this approach is that a tuned message can be exchanged across software applications either collectively on a single workstation (Bebop Driver $\leftrightarrows$ Mission Control) or across multiple machines (Mission Control $\leftrightarrows$ Ground Rovers (ATLAS)). This approach loses the flexibility of working with other external components in a simple manner, but gains the advantage of working with bespoke, fit-for-purpose functionality. The author recognises that ROS is a growing resource and that, in time, will overcome these issues; however until the ROS messaging system can be throttled and managed effectively for distributed control, a bespoke solution such as the one developed here will remain more suited to collaborative robotics applications employing wireless networks.

The 3D reconstructions performed in this work have been centred on ground and air vehicles with a core focus on the vision systems alone. There are other areas yet to be explored with this technique, underwater applications being the most obvious area (i.e. covering land, air and sea). Underwater vehicles pose a new series of constraints for a vision system such as lowlight, reduced visibility and sparse features. All of these factors add a new dimension to the application of computer vision for remote reconstruction.

9.2 Limitations

Structure from Motion is an extremely computationally expensive task requiring high quality imagery to be captured before processing. There are strict requirements relating to the motion of the camera: images must be captured with suitable translation and rotation between frames. These restrictions impose rules for nav-

igating a capture agent (ground vehicle) about an unknown environment, which is challenging as the shape of the environment influences the capture path.

The large-scale system presented in Chapter 8 makes the assumption that the aerial feed is stable and static above the target throughout. Although a rolling average is used to filter the live positioning data from the aircraft, the system does not account for any movement of the aircraft. A more mature system would need to account for the motion of the aircraft to compensate the offset in ground position. The likelihood is that sensor fusion would achieve this, in that IMU data from the aircraft would need to be calculated to add an offset to counter the drift of the aircraft.

Image quality was ultimately responsible for not being able to identify QR Codes or their respective 'finder' patterns. The distinct shape of the finder patterns become severely distorted and unable to provide location / pose information. A limitation of this experiment work is that other fiducials were not explored. A new experiment could have been devised to assess a range of fiducials, from simple coloured shapes through complex structures. Other targets may be easier to read from a low resolution feed than the complex QR code.

Another limitation of this work is the resolution of the imagery explored. The work explores the use of digital cameras ranging from Full-HD (1920 × 1080 pixels) through to 4K and 5K pixel stills. The Structure from Motion experiments within Chapter 4 did not explore the use of low resolution sources such as VGA cameras or standard web cameras. Web cameras were initially tested but proved to be far too blurry as soon as any motion was introduced, and therefore discarded and experiments began with HD or better sources. In any case, high resolution camera sources are necessary to produce detailed and accurate 3D reconstructions; low resolution cameras such web cameras and small first person view (FPV) cameras do not provide enough detail for use in 3D reconstruction.

9.3 Future Research

The bigger research question that sits at the heart of this work is to close the loop between sensing the real world through vision and the path-planning and SLAM systems. Other technologies such as LiDAR and RGB-D sensors enable robotic platforms to autonomously wander and explore unknown environments. Through this work it can be seen that although Structure from Motion produces detailed models and accurate information, the lack of scale and the significant processing time prevent it from being deployed on robotic platforms in the application of exploring remote unknown environments in its current form. Advances in hardware are enabling faster reconstructions, especially with the aid of general purpose graphics cards (GPGPU). Structure from Motion is not yet able to provide real-time information, though alternative approaches such as Visual SLAM and derivatives are narrowing in on live 3D telemetry. Once 3D reconstruction is available near real-time it can be integrated to close the loop between sensing, modelling and reacting to the modelled environment. Vision-only systems are not yet a drop-in replacement for sensing approaches such as LiDAR, however hopefully through this work it can be seen that computer vision is fast closing in on that complex goal and will soon be up to the task.

A new research question that arises from this work is can Structure from Motion be redesigned to take advantage of cloud-based and distributed processing? Throwing more resources at the task does make a difference to the processing time, so by redesigning the processing pipeline can one distribute the workload across a network of machines to enable near video-rate modelling? Additionally with the rise in GPGPU programming could the entire Structure from Motion system be implemented on the GPU? Computational GPUs are exceptionally powerful and are being used extensively in machine learning applications. Given the complex nature of Structure from Motion and the vast number of repetitive calculations could SfM on a GPGPU be the key to real-time modelling?

Throughout this work, the robotic platforms have remained standard commercial devices. Both the aerial and ground vehicles are stock products which have not been modified for the purposes of this work. An interesting research avenue could open up with the aid of more powerful unmanned aerial systems with on-board computation capabilities such as an AscTec Firefly or Pelican which feature a full Intel i7-based on-board computer with USB 3.0 support. These larger aircraft have longer operational times and more sophisticated flight controllers which enable the aircraft to be used in computer vision and SLAM tasks. An area which has not been explored within this thesis is utilising the processing whilst in the air. Processing on the aircraft removes the need to transmit full quality imagery down to the ground control system and produce accurate positioning information from the raw uncompressed video data. Working with larger UAS's also has the advantage of being able to attach higher resolution sensors such as 4K cameras given the larger payload capabilities; potentially enabling higher operational altitudes.

9.4 Concluding Remarks

Virtual and augmented reality systems are changing consumer markets, as these technologies become more refined and more widely used through consumer-level devices such as the Oculus Rift, PlayStation VR, HTC Vive, Google Cardboard, and Microsoft HoloLens, to name just a few of the key players. 3D reconstruction will play a significant role in these systems, especially in the area of augmented reality where the need to accurately and reliably map the virtual objects into the real-world are more demanding than enclosed virtual reality headsets. This work was driven by the desire of applying visual reconstruction in robotic applications. Simultaneous Localisation and Mapping (SLAM) is a well-established problem in the field of robotics. SLAM imposes time-bound constraints to enable application within on-line navigation systems. Although a well-established challenge, it is a non-trivial task which is enhanced by computer vision systems and approaches. 3D point-clouds such as those seen in this research and those of other 3D mapping systems provide

a model of the environment which could be used by robots in their path-planning and mission control modules.

The work has clearly shown that there is potential for a vision-only robotic reconstruction platform. In real-world applications the on-board sensors would be used to compliment the visual systems. Sensor fusion is an important part of modern robotic development and potentially holds the key to real-time visual 3D reconstructions. The big picture which inspired this research sees a collaborative team of robotic platforms travelling out into an unknown environment, whether that be a room or and entire complex, and work together to provide a live 3D model which can be used by operators to fulfil higher-level decision making tasks such as those found in search and rescue, fire management, disaster recovery or military applications.

Appendices

APPENDIX A

Derivations

There are some mathematical proofs which are needed to help understand the algorithms explained in the thesis, especially in chapter 4. This chapter will cover these blocks of maths.

A.1 Skew Symmetric Matrix

The essential matrix is defined by equation 3.13. This equation introduces a new notation of t which is expressed in a skew symmetric matrix. A vector cross product also can be expressed as the product of a skew-symmetric matrix and a vector. Given two vectors $\vec{a}$ & $\vec{b}$ we shall prove that $\vec{a}$ x $\vec{b} = \left[a\right]_x$ x $\vec{b}$.

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \tag{A.1}$$

$$\vec{b} = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} \tag{A.2}$$

The first step is to find the product of $\vec{a}$ and $\vec{b}$. This can also be expressed as the

union of $a \wedge b$ which gives a matrix determinate as seen below in equation A.3. $\hat{x}$, $\hat{y}$ & $\hat{z}$ are the unit vectors of each axis.

$$\vec{a} \times \vec{b} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} \tag{A.3}$$

Thus multiplying out gives:

$$\vec{a} \times \vec{b} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix} \tag{A.4}$$

Now that the answer for $\vec{a}$ x $\vec{b}$ is known the next step in the proof is to verify that the skew symmetric representation of a (as shown in A.5) produces the same result.

$$\Big[a\Big]_x = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \tag{A.5}$$

$$\Big[a\Big]_x \times \vec{b} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} -a_z b_y + a_y b_z \\ a_z b_x - a_x b_z \\ -a_y b_x + a_x b_y \end{pmatrix} \tag{A.6}$$

Rearranging the result of A.6 completes the proof that a skew symmetric matrix representation of the vector a.

Bibliography

[1] L. G. Clift and A. F. Clark, "Using UAVs and UGVs to Build 3D Models of Ground Features," *RSPSoc UAV Special Interest Group*, July 2013. [Online]. Available: http://www.rspsoc.org.uk/index.php/special-interest-groups/22-uav-sig.html

[2] L. G. Clift, A. O. Adeyemi-Ejeye, G. Koczian, S. D. Walker, and A. F. Clark, "Delivering live 4k broadcasting using today's technology," in *Technical Papers and Posters*. International Broadcasting Convention, September 2014.

[3] L. G. Clift and A. F. Clark, "Determining positions and distances using collaborative robots," in *Computer Science and Electronic Engineering Conference (CEEC), 2015 7th*, September 2015, pp. 189–194.

[4] ——, "Video frame extraction for 3d reconstruction," in *Computer Science and Electronic Engineering Conference (CEEC), 2016 8th*, September 2016.

[5] P. Waurzyniak, "Masters of Manufacturing: Patrick J. Hanratty," *Manufacturing Engineering*, 2010. [Online]. Available: http://www.sme.org/MEMagazine/Article.aspx?id=18986&taxid=1434

[6] I. E. Sutherland, "Sketch pad a man-machine graphical communication system," in *Proceedings of the SHARE design automation workshop*. ACM, 1964, pp. 6–329.

[7] F. Bruno, S. Bruno, G. D. Sensi, M.-L. Luchi, S. Mancuso, and M. Muzzupappa, "From 3d reconstruction to virtual reality: A complete methodology for digital archaeological exhibition," *Journal of Cultural Heritage*, vol. 11, no. 1, pp. 42 – 49, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1296207409000958

[8] E. Bagci, "Reverse engineering applications for recovery of broken or worn parts and re-manufacturing: Three case studies," *Advances in Engineering Software*, vol. 40, no. 6, pp. 407–418, 2009.

[9] NextEngine Inc., *Desktop 3D Laser Scanner*. [Online]. Available: http://www.nextengine.com/

[10] L. Galantucci, G. Percoco, G. Angelelli, C. Lopez, F. Introna, C. Liuzzi, and A. De Donno, "Reverse engineering techniques applied to a human skull, for cad 3d reconstruction and physical replication by rapid prototyping," *Journal of Medical Engineering & Technology*, vol. 30, no. 2, pp. 102–111, 2006.

[11] Microsoft, "Photosynth." [Online]. Available: http://photosynth.net/

[12] Google Inc., "Google Maps | Street View." [Online]. Available: http://maps.google.co.uk/intl/en/help/maps/streetview/

[13] "3D Doctor - Able Software Corp." [Online]. Available: http://www.ablesw.com/3d-doctor/3ddoctor.html

[14] P. Yecheng Wu and L. Yencharis, "Commercial 3-d imaging software migrates to pc medical diagnostics," *Advanced Imaging Magazine*, vol. October, pp. 16–21, 1998. [Online]. Available: http://www.ablesw.com/3d-doctor/aioct.html

[15] C. A. Northend, "Lidar, a laser radar for meteorological studies," *Naturwissenschaften*, vol. 54, no. 4, pp. 77–80, 1967. [Online]. Available: http://dx.doi.org/10.1007/BF00608760

[16] B. R. Clemesha, G. S. Kent, and R. W. H. Wright, "A laser radar for atmospheric studies," *Journal of Applied Meteorology*, vol. 6, no. 2, pp. 386–395, 1967. [Online]. Available: http://dx.doi.org/10.1175/1520-0450(1967)006<0386:ALRFAS>2.0.CO;2

[17] G. J. Agin and P. T. Highnam, "Movable light-stripe sensor for obtaining three-dimensional coordinate measurements," in *26th Annual Technical Symposium*. International Society for Optics and Photonics, 1983, pp. 326–333.

[18] C. Chen and A. Kak, "Modeling and calibration of a structured light scanner for 3-d robot vision," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4, Mar 1987, pp. 807–815.

[19] O. Hall-Holt and S. Rusinkiewicz, "Stripe boundary codes for real-time structured-light range scanning of moving objects," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, 2001, pp. 359–366 vol.2.

[20] K. Boyer and A. Kak, "Color-encoded structured light for rapid active ranging," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-9, no. 1, pp. 14–28, 1987.

[21] Microsoft. Microsoft kinect. Microsoft Corporation. [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/

[22] OpenNI. OpenNI - Open source framework for Natural Interaction (NI) devices. [Online]. Available: http://www.openni.org/

[23] OpenKinect & libfreenect - Open source Xbox Kinect software development. [Online]. Available: http://openkinect.org/

[24] K. Pfeil, S. L. Koh, and J. LaViola, "Exploring 3d gesture metaphors for interaction with unmanned aerial vehicles," in *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, ser. IUI '13.

New York, NY, USA: ACM, 2013, pp. 257–266. [Online]. Available: http://doi.acm.org/10.1145/2449396.2449429

[25] D. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras," *Multimedia, IEEE Transactions on*, vol. 15, no. 2, pp. 339–358, 2013.

[26] D. Alexiadis, G. Kordelas, K. C. Apostolakis, J. D. Agapito, J. Vegas, E. Izquierdo, and P. Daras, "Reconstruction for 3d immersive virtual environments," in *Image Analysis for Multimedia Interactive Services (WIAMIS), 2012 13th International Workshop on*, 2012, pp. 1–4.

[27] N. Karlsson, E. D. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vslam algorithm for robust localization and mapping," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.

[28] O. Faugeras, Q.-T. Luong, and T. Papadopoulou, *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. Cambridge, MA, USA: MIT Press, 2001.

[29] Noah Snavely, Steven M. Seitz, Richard Szeliski, "Photo Tourism: Exploring image collections in 3D," in *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)*, 2006. [Online]. Available: http://phototour.cs.washington.edu/Photo_Tourism.pdf

[30] N. Snavely, "Bundler: Structure from Motion (SfM) for Unordered Image Collections," Online. [Online]. Available: http://www.cs.cornell.edu/~snavely/bundler/

[31] R. S. Noah Snavely, Steven M. Seitz, "Modeling the world from internet photo collections," in *International Journal of Computer Vision*, 2007. [Online]. Available: http://phototour.cs.washington.edu/ModelingTheWorld_ijcv07.pdf

[32] P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Vision Algorithms: Theory and Practice*, ser. LNCS, W. Triggs, A. Zisserman, and R. Szeliski, Eds. Springer, 2000, vol. 1883, pp. 278–294.

[33] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. [Online]. Available: http://dx.doi.org/10.1023/B%3AVISI.0000029664.99615.94

[34] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[35] Furukawa, Yasutaka, "Clustering Views for Multi-view Stereo (CMVS)," Online. [Online]. Available: http://www.di.ens.fr/cmvs/

[36] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 8. IEEE, pp. 1362–1376. [Online]. Available: http://homes.cs.washington.edu/~furukawa/papers/pami08a.pdf

[37] Furukawa, Yasutaka and Ponce, Jean, "Patch-based Multi-view Stereo Software," Online. [Online]. Available: http://www.di.ens.fr/pmvs/

[38] MeshLab, University of Pisa, "MeshLab." [Online]. Available: http://meshlab.sourceforge.net/

[39] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[40] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 5, no. 4, pp. 349–359, 1999.

[41] T. P. M. Jancosek, "Multi-View Reconstruction Preserving Weakly-Supported Surfaces," in *IEEE Conference on Computer Vision and Pattern Recognition 2011*.

[42] Changchang Wu, ""SiftGPU: A GPU implementation of Scale Invaraint Feature Transform (SIFT)"." [Online]. Available: http://cs.unc.edu/~ccwu/siftgpu

[43] ——, "VisualSFM: A Visual Structure from Motion System." [Online]. Available: http://homes.cs.washington.edu/~ccwu/vsfm/

[44] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M. Seitz, "Multicore bundle adjustment," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2011.

[45] C. Sweeney, *Theia Multiview Geometry Library: Tutorial & Reference*, University of California Santa Barbara.

[46] M.W.M.G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte and M. Csorba, "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," in *IEEE Trans. on Robotics and Automation*, vol. 17, 2001, pp. 229–241.

[47] H. Durrant-Whyte, "Uncertain geometry in robotics," in *IEEE Trans. Robot and Automation*, vol. 4, 1988, pp. 23–31.

[48] G. Tuna, K. Gulez, V. C. Gungor, and T. V. Mumcu, "Evaluations of different simultaneous localization and mapping (slam) algorithms," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012, pp. 2693–2698.

[49] Y. Sukjune, P. Sung-Kee, C. H. Do, K. Soohyun, and K. Y. Keun, "Vision-based outdoor simultaneous localization and map building using compressed extended kalman filter," in *Control Conference (ECC), 2007 European*, July 2007, pp. 2819–2824.

[50] R. Mur-Artal, J. Montiel, and J. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *Robotics, IEEE Transactions on*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[51] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2320–2327.

[52] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 2007, pp. 225–234.

[53] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, 2006, pp. 404–417.

[54] C. F. Liew and T. Yairi, "Generalized brief: A novel fast feature extraction method for robust hand detection," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, Aug 2014, pp. 3014–3019.

[55] J. Liu and X. Liang, "I-brief: A fast feature point descriptor with more robust features," in *Signal-Image Technology and Internet-Based Systems (SITIS), 2011 Seventh International Conference on*, Nov 2011, pp. 322–328.

[56] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2548–2555.

[57] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2564–2571.

[58] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[59] J. Engel, J. Stuckler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 1935–1942.

[60] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[61] M. Milford, G. Wyeth, and D. Prasser, "Ratslam: a hippocampal model for simultaneous localization and mapping," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, April 2004, pp. 403–408 Vol.1.

[62] M. J. Milford and G. F. Wyeth, "Mapping a suburb with a single camera using a biologically inspired slam system," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1038–1053, 2008.

[63] D. Ball, S. Heath, J. Wiles, G. Wyeth, P. Corke, and M. Milford, "Openratslam: an open source brain-based slam system," *Autonomous Robots*, vol. 34, no. 3, pp. 149–176, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9317-9

[64] A. Huletski, D. Kartashov, and K. Krinkin, "Evaluation of the modern visual slam methods," in *Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT), 2015*, Nov 2015, pp. 19–25.

[65] V. Petridis and N. Zikos, "L-slam: Reduced dimensionality fastslam algorithms," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, July 2010, pp. 1–7.

[66] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[67] Xiaodong Li, Nabil Aouf and Abdelkrim Nemra, "3D Mapping based VSLAM for UAVs," in *20th Mediterranean Conference on Control & Automation (MED)*, July 2012.

[68] A. Nemra, "Robust airborne 3d visual simultaneous localisation and mapping," Ph.D. dissertation, Department of Informatics and Systems Engineering, Cranfield University, 2010.

[69] J. Wang and G. Beni, "Cellular robotic system with stationary robots and its application to manufacturing lattices," in *Intelligent Control, 1989. Proceedings., IEEE International Symposium on*, Sep 1989, pp. 132–137.

[70] T. Fukuda and S. Nakagawa, "Dynamically reconfigurable robotic system," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, Apr 1988, pp. 1581–1586 vol.3.

[71] T. Fukuda, Y. Kawauchi, and H. Asama, "Analysis and evaluation of cellular robotics (cebot) as a distributed intelligent system by communication information amount," in *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, Jul 1990, pp. 827–834 vol.2.

[72] E. Freund and H. Hoyer, "Pathfinding in multi-robot systems: Solution and applications," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, Apr 1986, pp. 103–111.

[73] H. Chu and H. A. ElMaraghy, "Real-time multi-robot path planner based on a heuristic approach," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, May 1992, pp. 475–480 vol.1.

[74] T. Arai, E. Pagello, and L. E. Parker, "Editorial: Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.

[75] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *Robotics Automation Magazine, IEEE*, vol. 13, no. 3, pp. 16–25, 2006.

[76] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, "A survey of inter-vehicle communication protocols and their applications," *IEEE Communications Surveys Tutorials*, vol. 11, no. 2, pp. 3–20, Second 2009.

[77] C. Phan and H. Liu, "A cooperative uav/ugv platform for wildfire detection and fighting," in *System Simulation and Scientific Computing, 2008. ICSC 2008. Asia Simulation Conference - 7th International Conference on*, 2008, pp. 494–498.

[78] D. A. R. P. Agency. (2007, November) Darpa urban challenge. [Online]. Available: http://archive.darpa.mil/grandchallenge/

[79] G. S. S. Andrew Howard, Lynne E. Parker, "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection," *International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 431–447, May 2006. [Online]. Available: http://journals.sagepub.com/doi/abs/10.1177/0278364906065378

[80] A. Nemra and N. Aouf, "Robust cooperative uav visual slam," in *Cybernetic Intelligent Systems (CIS), 2010 IEEE 9th International Conference on*. IEEE, 2010, pp. 1–6.

[81] F. Lang, "Metropolis," Film, March 1927.

[82] CAST - College of Engineering. SURENA III. University of Tehran. [Online]. Available: http://surenahumanoid.com

[83] Stanford Robotics Lab. Ocean One. Stanford University. [Online]. Available: http://cs.stanford.edu/groups/manips/ocean-one.html

[84] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *ACM SIGGRAPH computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 25–34.

[85] E. Şahin, *Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, ch. Swarm Robotics: From Sources of Inspiration to Domains of Application, pp. 10–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30552-1_2

[86] Y. fei Zhu and X. min Tang, "Overview of swarm intelligence," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 9, Oct 2010, pp. V9–400–V9–403.

[87] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, Mar. 2004.

[88] M. Born and E. Wolf, *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Cambridge University Press, 1999.

[89] B. Guenther, *Modern Optics*. Oxford University Press, 2015.

[90] M. Pollefeys, "Visual 3D Modeling from Images," in *Tutorial Notes*. University of North Carolina - Chapel Hill, 2004. [Online]. Available: http://www.cs.unc.edu/~marc/tutorial/tutorial02.html

[91] Y. Y. Morvan, "Acquisition, compression and rendering of depth and texture for multi-view video," Ph.D. dissertation, Technische Universiteit Eindhoven, 2009.

[92] O. Faugeras, Q.-T. Luong, and S. Maybank, "Camera self-calibration: Theory and experiments," in *Computer Vision ECCV'92*, ser. Lecture Notes in Computer Science, G. Sandini, Ed. Springer Berlin Heidelberg, 1992, vol. 588, pp. 321–334. [Online]. Available: http://dx.doi.org/10.1007/3-540-55426-2_37

[93] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, 1999, pp. 666–673 vol.1.

[94] ——, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.

[95] M. R. Oswald, E. Toeppe, C. Nieuwenhuis, and D. Cremers, "A survey on geometry recovery from a single image with focus on curved object reconstruction," in *Proceedings of the 2011 Conference on Innovations for Shape Analysis: Models and Algorithms*. Springer-Verlag, 2011.

[96] M. R. Oswald, E. Toeppe, and D. Cremers, "Fast and globally optimal single view reconstruction of curved objects," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, Rhode Island, Jun. 2012, pp. 534–541.

[97] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[98] H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pp. 61–62, 1987.

[99] W. Chojnacki, M. J. Brooks, A. V. D. Hengel, and D. Gawley, "Revisiting hartley's normalized eight-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1172–1177, 2003.

[100] R. Hartley, "In defense of the eight-point algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 6, pp. 580–593, Jun 1997.

[101] D. Nistér, "An efficient solution to the five-point relative pose problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6,

pp. 756–770, 2004.

[102] H. Stewenius, C. Engels, and D. Nistér, "Recent developments on direct relative orientation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 60, no. 4, pp. 284–294, 2006.

[103] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 2969–2976.

[104] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[105] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," *Computer Vision–ECCV 2008*, pp. 500–513, 2008.

[106] Q.-T. Luong and O. D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *International Journal of Computer Vision*, vol. 17, no. 1, pp. 43–75, 1996.

[107] OA Digital Labs, "Multi View 3D." [Online]. Available: http://oadigital.net/oalabsintro/oalabscv

[108] EDF R&D, "CloudCompare," 2009. [Online]. Available: http://www.cloudcompare.org/

[109] Colchester Archaeological Trust. (2014, September) The fenwick treasure at williams & griffin! Online. Colchester Archaeological Trust. [Online]. Available: http://www.thecolchesterarchaeologist.co.uk/?p=14844

[110] S. Pertuz, D. Puig, and M. A. Garcia, "Analysis of focus measure operators for shape-from-focus," *Pattern Recognition*, vol. 46, no. 5, pp. 1415 – 1432, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320312004736

[111] P. Ritsos and A. F. Clark, "Engineering an augmented reality tour guide," in *Proceedings of Eurowearables '03*, Sep. 2003, pp. 119–124.

[112] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa, "A robust visual odometry and precipice detection system using consumer-grade monocular vision," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Apr. 2005.

[113] M. Labbé and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[114] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[115] D. J. Johnston and A. F. Clark, "A vision-based location system using fiducials," in *Proceedings of the First International Conference on Vision, Video and Graphics*, Jul. 2003, pp. 159–166.

[116] "Vicon motion capture systems," http://www.vicon.com/products/camera-systems, accessed: 2015-06-25.

[117] "QR code features," http://www.qrcode.com/en/qrfeature.html, accessed: 2015-06-25.

[118] G. L. Squires, *Practical Physics*, 4th ed. Cambridge University Press, 2001.

[119] "Zbar bar code reader," http://zbar.sourceforge.net/index.html, accessed: 2015-06-25.

[120] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[121] P. Brisset, A. Drouin, M. Gorraz, P.-S. Huard, and J. Tyler, "The paparazzi solution," in *MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles*, 2006.

[122] L. Meier, "MAVLink Micro Air Vehicle Communication Protocol. Retrieved Online," Tech. Rep., 2013. [Online]. Available: http://qgroundcontrol.org/mavlink

[123] AutonomyLab, Simon Fraser University, "ROS ardrone_autonomy," 2014. [Online]. Available: http://wiki.ros.org/ardrone_autonomy

[124] Parrot, "Parrot for developers AR2 SDK," 2012. [Online]. Available: http://ardrone2.parrot.com/

[125] B. Bedi, M. Carter, and A. Stanford-Clark, "Control of publish/subscribe messaging," Mar. 2 2006, US Patent App. 11/209,445. [Online]. Available: https://www.google.com/patents/US20060047666

[126] AutonomyLab, Simon Fraser University, "ROS bebop_autonomy," 2015. [Online]. Available: http://wiki.ros.org/bebop_autonomy

[127] Parrot, "Parrot Bebop Drone for developers SDK3," 2015. [Online]. Available: http://developer.parrot.com/docs/SDK3

[128] United States Coast Guard, U.S. Department of Homeland Security, "Team Coordination Training (TCT)," 2004. [Online]. Available: https://www.uscg.mil/auxiliary/training/tct/

[129] Z. M. Ben-Halim, T. E. Dickey, J. Pfeifer, and E. S. Raymond. ncurses. [Online]. Available: http://www.gnu.org/software/ncurses/

[130] ISO, "Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding," International Organization for Standardization, Geneva, Switzerland, ISO/IEC 14496-10:2014, Sep. 2014.

[131] ——, "Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding," International Organization for Standardization, Geneva, Switzerland, ISO/IEC 23008-2:2013(E), Dec. 2013.

[132] ——, "Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding," International Organization for Standardization, Geneva, Switzerland, ISO/IEC 23008-2:2015(E), May 2015.

[133] C. Forrester, "BBC readies for UHD/HDR transmissions," *Advanced Television*, June 2016. [Online]. Available: http://advanced-television.com/2016/06/27/bbc-readies-for-uhdhdr-transmissions/

[134] L. Kelion, "Sky's 4k line-up includes james bond and premier league," *BBC News*, vol. Technology, July 2016, uK broadcaster will start showing ultra-high resolution content to some of its satellite customers from 13 August. [Online]. Available: http://www.bbc.co.uk/news/technology-36793510

[135] BT Ultra HD, "BT Ultra HD: Europe's first live sports Ultra HD (4K) Channel," July 2016. [Online]. Available: https://www.productsandservices.bt.com/products/ultra-hd/

[136] T. Moynihan, "4K TV's Biggest Content Problem: No Live Broadcasts Anytime Soon," *WIRED*, vol. GEAR, February 2015, http://www.wired.com/2015/02/live-4k-broadcasts/.

[137] SMPTE, "ST 292-1:2012 - SMPTE Standard - 1.5 Gb/s Signal/Data Serial Interface," *SMPTE ST 292-1:2012*, pp. 1–20, Jan 2012.

[138] ——, “ST 424:2012 - SMPTE Standard - 3 Gb/s Signal/Data Serial Interface,” *SMPTE ST 424:2012*, pp. 1–10, Oct 2012.

[139] ——, “ST 2081-10:2015 - SMPTE Standard - 2160-Line and 1080-Line Source Image and Ancillary Data Mapping for Single-Link 6G-SDI,” *SMPTE ST 2081-10:2015*, pp. 1–22, March 2015.

[140] A. O. Adeyemi-Ejeye and S. D. Walker, “Ultra-high definition wireless video transmission using h. 264 over 802.11 n wlan: Challenges and performance evaluation,” in *Telecommunications (ConTEL), 2013 12th International Conference on*. IEEE, 2013, pp. 109–114.

[141] A. O. Adeyemi-Ejeye and S. Walker, “4kUHD H264 wireless live video streaming using CUDA,” *Journal of Electrical and Computer Engineering*, vol. 2014, p. 2, 2014.

[142] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” Internet Requests for Comments, RFC Editor, RFC 3550, July 2003. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3550.txt

[143] H. S. Parmar and M. C. Thornburgh, “Real-Time Messaging Protocol (RTMP) specification,” Adobe Developer Connection, Adobe Systems Incorporated, Specification, December 2012. [Online]. Available: http://www.adobe.com/devnet/rtmp.html

[144] FFmpeg, “A complete, cross-platform solution to record, convert and stream audio and video.” [Online]. Available: http://www.ffmpeg.org

[145] LibAV, “Open source audio and video processing tools.” [Online]. Available: http://libav.org/

[146] Ofcom, “Average UK broadband speed continues to rise.” [Online]. Available: http://media.ofcom.org.uk/news/2013/average-uk-broadband-speed-continues-to-rise

[147] ——, "Nearly one in three UK broadband connections now superfast." [Online]. Available: http://media.ofcom.org.uk/content/posts/news/2015/one-in-three-uk-broadband-superfast

[148] M. Jackson, "Ofcom - Average UK Home Broadband Speeds Jump to Reach 22.8Mbps," ISPreview.co.uk. [Online]. Available: http://www.ispreview.co.uk/index.php/2015/02/ofcom-average-uk-home-broadband-speeds-slowly-reach-23mbps.html

[149] A. O. T. Adeyemi-Ejeye, "Ultra-high definition wireless video streaming," Ph.D. dissertation, University of Essex, 2015.

[150] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[151] J. Vollmer, R. Mencl, and H. Mueller, "Improved laplacian smoothing of noisy surface meshes," in *Computer Graphics Forum*, vol. 18, no. 3. Wiley Online Library, 1999, pp. 131–138.

[152] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*. ACM, 2006, pp. 381–389.