CRANFIELD UNIVERSITY
SCHOOL OF ENGINEERING

PhD Thesis

Academic Year 2013-2014

MIKAEL MANNBERG

Vision-based Navigation Using Landmark Recognition for Unmanned Aerial Vehicles

Supervisor: Dr. Al Savvaris

August 2014

Landmark Navigation for Unmanned Aircraft

Mikael Mannberg

School of Engineering

Cranfield University

July 2014

Abstract

This thesis describes a new approach for a vision-based positioning system for Unmanned Aerial Vehicles using a recognition method based on known, robust geographic landmarks. Landmarks are used to calculate a position estimate in a global coordinate frame without requiring external signals, such as GPS. Absolute systems are of interest as they provide a redundant positioning system, allow UAVs to operate when GPS-denied and can enable high-precision landings for spacecraft.

The core challenge with vision-based absolute positioning is recognition of landmarks. Most abundant landmarks, such as buildings, are visually similar and difficult to distinguish. Previous research in the area tends to focus on matching raw aerial image data to a set of reference images. While these methods can achieve acceptable results in specific conditions, they struggle with variations in lighting, seasonal changes and changing environments. This thesis presents a new multi-stage method that aims to solve this using a high-level matching framework where landmarks identified in an aerial image are matched to a reference database.

This has led to the development of a geometric feature descriptor that encodes the topography of landmarks. The proposed system therefore matches the arrangement of features rather than the appearance, which lets it distinguish individual landmarks in large sets (20,000+ features). Since the arrangement of landmarks often is semi-structured and ambiguous, in particular when considering man-made landmarks, a matching stage has been developed that uses a number of strategies to enable matching of individual landmarks to a full database.

The results have been evaluated for two conceptual vehicles with acceptable results, highlighting the strengths of the proposed system as well as areas for improvement.

Acknowledgements

A huge thanks to my friends, my family, my supervisor and most importantly my girlfriend who had to deal with this for way too long, then read it and fixed it without going crazy.

Time for a beer.

Contents

List of Figures

List of Tables

Nomenclature

c	Calibrated Image Centre
CCD	Charge Coupled Device
COTS	Commercial off-the-shelf
DOF	Degrees of Freedom
DSLR	Digital Single Lens Reflex Camera
EPSG	European Petroleum Surveyor's Group
f	Fingerprint Vector, Focal Length
FPGA	Field Programmable Gate Array
GIS	Geographic Information Systems
GML	Geographic Mark-up Language
GPGPU	General Purpose Graphical Processing Unit
GPS	Global Positioning System
k	Radial Distortion Parameters
OGP	International Association of Oil and Gas Producers
p	Tangential Distortion Parameters
s	Match Score
SFM	Structure From Motion
SIFT	Scale-Invariant Feature Transform
SQL	Structured Query Language
SURF	Speeded-Up Robust Features
T	Tanimoto Score
TOID	Topographic ID
U	Query Vector
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
V	Target Vector
VPS	Vision-based Positioning System
XML	Extensible Mark-up Language
WGS84	World Geodetic System 1984

Subscripts

f	Feature
i	Index
n	Neighbour
s	Score
x	Along X-axis
y	Along Y-axis

Chapter 1

Introduction

The aim of this project is to investigate the current state of visual positioning systems for unmanned aerial vehicles (UAVs) and to develop a new approach to vision-based positioning that can provide an absolute rather than relative position estimate. There are several advantages to such a system; in particular, it allows the precise positioning of UAVs without a dependency on external inputs, and the continuation of a mission if the primary positioning system fails.

As one of the primary sensory systems, vision is a logical and practical field to search for alternative positioning methods. Vision is used by humans, and many other creatures in the animal kingdom, to help determine their location relative to their surroundings; furthermore it helps determine position, judge distances, and even identify possible routing for navigation. The uses of vision scale from the very basic, such as in-flight stabilisation of a housefly to the highly sophisticated systems that allow humans to recognise and navigate the world. In literature, visual positioning systems have been studied extensively for vehicles across a variety of environments (ground, subsea & surface, and now aerial), in the promise that they will provide significant improvements to the current state-of-the-art systems, such as GPS and inertial navigation.

The use of vision as a positioning system provides several challenges; the core task, and thus area of interest, is how to incorporate recognition within the system. Humans are able to recognise features and objects through sight and relate them to memories and contextual information; a very relevant example of this is a map, which is a highly reduced topological representation of the world. By interpreting and understanding the topological representation of a city (including identifying the symbology used within the map, such as a cross for a place of worship), humans begin to recognize the city structure, including the layout of the roads, streets, buildings and landmarks. This helps us to anticipate how these features connect as we travel through the city, progressively building an internal mental map of the area. Even though maps are highly reduced representations of the world, they still allow humans to recognise areas from various viewpoints, whether on the ground or in the air. This includes, for example, passengers attempting to interpret location during a flight (particularly during take-off or landing) by searching for recognisable landmarks and features - or even their home - on the ground below.

This thesis will seek to develop a vision-based positioning system by focusing on reproducing the representation of topological structure. To support this development, the thesis will study computer vision methods that are currently used to recognise features and landmarks. These features, once identified, will be used to retrieve additional meta-data such as global location. This meta-data provides input into well-established algorithms for pose estimation that will enable the system to estimate its position in the world, helping this project to achieve its aim: to demonstrate a system using a map-like representation for positioning.

There are a number of technical motivations for a vision-based positioning system. Contrary to the positioning systems used today (such as GPS and navigation beacons), visual systems are based entirely on-board the vehicle. A visual system therefore allows vehicles to safely operate in areas where they may be out of range of external signals, where signals are not available, or where the signals may be interfered with. This includes several scenarios, such as a vehicle orbiting a planet or a Micro-UAV exploring a city.

1.1 Objectives

The following objectives were identified at the start of the project:

- Research optical sensors and models.
- Investigate computer vision algorithms for feature extraction, motion estimation and structure from motion (SFM).
- Investigate how visual data can be used to determine the true location of a vehicle and how it can be fused with other onboard sensors.
- Develop system architecture and research hardware and software methods required to provide real-time positioning data.

1.2 Contribution to Knowledge

The project primarily focuses on demonstrating the possibility of using a novel geometric fingerprinting algorithm to uniquely describe individual landmarks in an image by encoding geographic structure. The fingerprinting method is scale, rotation and translation invariant to enable localisation at a range of conditions with minimal additional computational overhead, a common problem with prior work in this area. The description method is minimal and invariant to the sensor and feature type, which allows the core algorithms to be used in a variety of missions and platforms with few modifications. The descriptor also has the advantage that it can be used in other types of problems where geometric patterns need to be matched or compared.

Fingerprinting methods have been used extensively in various fields such as audio recognition and chemical analysis. However, the methods have not been applied to geographical data due to ambiguity and comparatively low variety in geographical structure. This thesis demonstrates that it is indeed possible to uniquely identify individual landmarks and use them for a positioning system.

Secondly, the project also shows that, given only a number of descriptors extracted from a sensor measurement such as an image, it is possible to match detected landmarks to a known region and recover the locations of the detected features in a global coordinate frame. This is facilitated by using the new fingerprinting algorithm combined with a hypothesis-and-test matcher that evaluates a number of potential match candidates and selects the most suitable result. This process is dependent on the size of the target area region but tests have shown that accurate matching of individual features is possible with the help of contextual information.

Finally, the project is the first example of a system using Gabriel graphs for the selection and detection of poorly conditioned features. This is usually achieved by studying the feature vector itself but, since the fingerprints in this work are closely tied to geographic arrangements, the Gabriel graph provides an alternative way to carry out the process.

1.3 Publications

The following articles have been published during the course of this PhD:

- Landmark Fingerprinting and Matching for Aerial Positioning Systems[4]
  Mikael Mannberg & Al Savvaris
  AIAA Journal of Aerospace Information Systems, 2014

- A Visual Positioning System for UAVs Using Landmark Fingerprinting[5]
  Mikael Mannberg & Al Savvaris
  AIAA InfoTech@Aerospace Conference Proceedings, 2012

- Visual Odometry with Failure Detection for the Aegis UAV[1]
  Jose Roger-Verdeguer, Mikael Mannberg & Al Savvaris
  IEEE Imaging Systems and Techniques Conference Proceedings, 2012

- Automatic Pipeline Detection for UAVs[6]
  Hani Alqaan, Mikael Mannberg & Al Savvaris
  AIAA InfoTech@Aerospace Conference Proceedings, 2012

- High Precision Real-time 3D Tracking Using Cameras[7]
  Mikael Mannberg, Peter Silson, Antonios Tsourdos & Al Savvaris
  AIAA InfoTech Conference Proceedings, 2011

1.4 Thesis Structure

The thesis begins with a literature review (Chapter 2) that explores and outlines the current state-of-the-art methods used by both traditional and visual positioning systems for unmanned aerial vehicles. It proceeds to review current positioning methods and other sensors commonly available on unmanned vehicles. The review then describes and discusses methods for a higher-level visual navigation system, using feature description and matching methods based on work in other fields.

The literature review has two aims: the first is to demonstrate that the current work in the field of visual positioning is focused on approaches distinct from the method proposed by this thesis. The second aim is to demonstrate that the algorithms surrounding the feature descriptor and matcher, such as landmark extraction and pose estimation, are well studied and that the data required, such as geographical reference databases and efficient retrieval methods, are available and accessible. This thus allows the thesis to concentrate on the core task: the recognition problem.

Next, the System Overview chapter (Chapter 3) outlines the theory of operation and architecture of the proposed system. This includes a discussion of how the system operates and where it would fit among other systems onboard an autonomous vehicle. It also explains the proposed system architecture, including reasons behind the need for modularity and the various sub-systems that are required.

The following two chapters (Chapters 4 and 5) focus on the core recognition problem, starting with Feature Description then continuing on to Feature Matching. In Chapter 4 a new type of feature descriptor for geographical features is developed that is scale, rotation and translation invariant. Following this, Chapter 5 discusses various approaches to match the descriptor to ensure high quality results.

Chapter 6 studies the performance of two hypothetical configurations of the proposed system. One configuration is designed for a small and low-power fixed wing unmanned aerial vehicle that needs positioning updates at a high rate. The second configuration is a satellite in orbit which only requires occasional updates.

The thesis then concludes with Chapter 7, which reviews the work that has been conducted and provides a final discussion of the results. The chapter also discusses various areas where future work can be carried out. This includes both improvements to the positioning system as well as alternative uses of the new fingerprinting algorithms.

Chapter 2

Literature Review

2.1 Introduction

> "For the subject of vision, there is no single equation or view that explains everything. Each problem has to be addressed from several points of view - as a problem in representing information, as a computation capable of driving that representation, and as a problem in the architecture of a computer capable of carrying out both things quickly and reliably." - David Marr, 1982 [8]

David Marr's book "Vision" was published posthumously over 30 years ago, yet it is still considered one of the foundations for computer vision today. Marr's academic career began with an attempt to create a framework for how the human neural system can be simulated computationally but he soon realised that one of the most interesting challenges was to emulate the visual cortex, the part of the brain that understands sight. His main conclusion? The approach taken by earlier researchers had been wrong.

Computer vision research began as an extension of early artificial intelligence work, since the potential for computers that could understand imagery, concepts and make autonomous decisions was quite clear. At the time researchers assumed that intelligence and understanding could be modelled using simple algorithms and all that was needed to implement an artificial intelligence was to invent the correct algorithm. This belief, that intelligence was ultimately a simple problem that had a straightforward logical solution, was so strong that the Chilean government once attempted to develop a cybernetic system called CYBERSYN that would automate many of the country's governmental affairs[9].

This thinking led to the development and application of many fundamental computer vision methods algorithms in the 1960's and 1970's that are still frequently used today, such as Fourier transforms, the Harris corner detector [10] and histogram analysis. However, it soon became clear that these methods worked for very basic problems in strictly controlled conditions but they were unable to deliver the intelligence and autonomy that was imagined in the 1950's. While researchers began to realise that the problem was more complex than initially perceived, it was not until David Marr outlined his computational framework that it was clear that computer vision research needed a different approach.

Marr presented three main problems, all of which need to be solved to fully interpret a visual input:

1. Representation
   The most complex challenge deals with knowledge representation and the issue of retrieving timely and contextually relevant information for the task at hand. For example, given a task to locate a vehicle, humans understand not only the abstract concept of a vehicle (it is a method for transportation) but also where vehicles appear (roads, car parks), what components they consist of (wheels, doors, windows) and how they behave.

   Representing and retrieving all of this information is a challenging task as each concept has its own properties and relies on lower level concepts (for example, a wheel is round, has a tire and is usually in contact with the road). The result of this is that the brain can retrieve a vast amount of contextually relevant information before it even attempts to interpret a scene.

   > "Thus, there is a trade-off; any particular representation makes certain information explicit at the expense of information that is pushed into the background." - David Marr, 1982

   If one were to attempt an implementation of a robust computer vision system there would need to be a similar procedure in place that replicates the process of knowledge representation. This was a completely new field in Marr's time and progress has only been made recently, with much of the work being done by Google[11] who are developing what they refer to as a knowledge graph. The knowledge graph is used to assist their users to find information online by connecting simple search queries with larger concepts such as people and places. This has proven to be a very successful approach for Google and is one of the reasons it is currently the leading search engine.

However, the use of knowledge representation in computer vision is virtually nonexistant today since it is closely tied to the second problem Marr identified: image analysis.

2. Analysis
   When the relevant knowledge about the scene has been selected and distilled into its most useful state the next problem is to use this to extract information from the data captured by the imaging device.

   This area is where the bulk of the progress has been made in computer vision, which has led to the development of high-level classifiers, machine learning algorithms and more that helps locate and identify specific objects in an image. The goal is to use the information from the scene representation to select a suitable algorithm that can process the image and produce actionable outputs, such as the location of a vehicle in the scene.

   However, since today's systems lack contextual understanding of the scene, there is no feedback to an algorithm as to whether it is suitable for the task at hand. As a result, the algorithm will attempt to process the image and likely fail unless the very specific conditions it has been designed for are met. There is very rarely any feedback within the system to adapt or retry with a different object detection algorithm.

   This is one of the main limitations for computer vision today, in particular for unmanned aerial vehicles, as most systems continually experience variations in operating conditions (such as altitude, time of day, weather) and target scenes (urban, rural, desert, sea etc). Thus, most real-world use of computer vision methods are in strictly controlled environments (such as vehicle number plate recognition or facial scanners for border control[12]).

Computer vision has also seen extensive use in Visual Simultaneous Location And Mapping (Visual-SLAM or VSLAM) systems which can be utilised in a wider variety of environments. This flexibility is an advantage that is gained by using very basic image analysis methods and giving up any attempt to visually interpret the scene. Visual-SLAM systems often only track basic corner and edge fetures in an image, the algorithms rely on advanced sensor models and complex statistical filters to do its work.

3. System Implementation
   The last challenge that needs to be considered is the implementation of the computer vision system itself, in the context of the task at hand. This primarily deals with the hardware and software implementation of the methods and is highly dependent on the overall task and mission.

   This means that one has to consider the efficiency of the implementation, the time-sensitivity of the task, the available computational power and, particularly in the case of unmanned aerial vehicles, the electrical power available for the sensor system. In addition, computer vision systems used to be designed in isolation but are now often closely integrated with other sensors and sub-systems on a platform. Hence there is an additional need to consider the complete system as well as the interaction with other sub-systems.

Marr argues that to develop a successful computer vision system these three aspects need to be considered during the development, and not until all of the challenges are solved will we have a truly robust computer vision system.

Unfortunately, his vision was ahead of its time and even today computer vision is far from being able to solve these issues. Encouragingly, the research into these fields has been advancing in recent years, driven by the increase in computational power and renewed interest. Until these challenges have been solved we are limited to systems that will only operate in very specific scenarios and under certain conditions that need to be clearly defined before beginning the mission.

Taking a step back from Marr's ambitious vision, much progress has still been made in the computer vision field and the methods that are available today can deliver impressive results.

The aim of this thesis is to investigate computer vision from a positioning and navigation perspective for unmanned aircraft. The first part of this literature review will review some of the techniques used to determine the position of an aerial vehicle and discuss some current methods using visual techniques. The review then continues on to explore methods for a higher level visual navigation system using feature description and matching methods based on work in other fields.

2.2 Current Positioning Systems

Before reviewing visual methods it is useful to discuss the current state of positioning systems for unmanned aircraft. While they share many sensors with manned aircraft, such as airspeed, altimeters, angle of attack sensors and more, they also rely on additional positioning systems to ensure that they can operate autonomously. These systems can be divided into two types: relative and absolute. Relative systems provides an estimate of how the vehicle has moved relative to a starting point in a local coordinate system. Absolute systems will give a position estimate within a global reference frame.

2.2.1 Inertial Measurement Units (IMU)

Most UAVs are equipped with high performance inertial measurement units containing sensors such as accelerometers, gyros and magnetometers that are used to determine the relative position of the vehicle. Each sensor measures the change in acceleration or rotational rates in one axis and feeds it back to a system that integrates the measurements with respect to time, giving the vehicle's state estimate: position, velocity and orientation.

The technology used in these sensors, known as Micro Electro-Mechanical Sensors (MEMS), has recently advanced rapidly thanks to the inclusion of such sensors in personal electronics (such as smartphones and tablets). Since the mobile device industry has very specific requirements for low cost, small physical footprint and high quality data this has led to both a miniaturisation of the sensors and significant improvements in the data quality and rates. As a result of this, it is now possible for hobbyists and small UAV manufacturers to build a low cost IMU that rivals highly advanced commercial devices.

However, MEMS sensors have a few problems. For example, they are discrete, they are not measured continuously but rather queried at specific intervals. This leads to integration errors that gradually build up over time and causes what is known as drift on the sensor as the position and orientation errors increase. A way to overcome this is by polling the sensor at very high rates, usually hundreds or thousands of times per second, however this is not a long term solution as it will only delay the problem due to accumulation of errors.

In addition, the individual sensors themselves are not perfect as there are measurement errors, noise, bias and electrical interference that will give incorrect readings and directly offset the vehicle's reported position. For this reason much of the work on IMUs today is focused on the software side where sensor models and filters (most commonly Kalman filters[13]) are used to model these inaccuracies and improve the IMU's state estimate. These filters obtain estimates for the sensor errors and attempt to use statistical methods to correct the state estimate. However, even high precision IMUs with high quality sensors and well designed filters can build up enough drift error to be unusuable within a few minutes.

A final issue with relative sensors such as IMUs is that they are measuring movement in a local coordinate frame that is not aligned with the global frame that the vehicle is operating in. As such the vehicle must either operate in a body coordinate system, which is impractical for the user, or aligned with a global coordinate system somehow.

2.2.2 The Global Positioning System (GPS), Galileo and GLONASS

The Global Positioning System (GPS) is, as the name implies, a global positioning system that allows users to locate themselves accurately almost anywhere in the world.

GPS uses timing signals relayed from up to 24 satellites in orbit around the earth. Each satellite is equipped with an atomic clock which, when a receiver is synchronised to the signal, can be used to determine the time it takes for the signal to reach the receiver. When the receiver has obtained an accurate time measurement it can determine each satellite's position, and then its own position by triangulating the signals sent from the satellites. By using at least four satellites, it is possible to triangulate the location of the vehicle and obtain a 3D (latitude, longitude, altitude) solution with high accuracy. Usually the positional error is around five to ten meters for civilian use. The level of positioning accuracy is partially due to atmospheric effects and measurement errors, however the primary limitation is purely artificial. GPS was developed by the US Airforce and was primarily intended to give their vehicles a system that enables high precision positioning. Thus, there are two GPS bands, a military band with classified performance and a civilian band with slightly reduced accuracy.

After the GPS system was made available for civilian use, secondary systems have been developed to improve the accuracy. An example is differential GPS (DGPS) where the DGPS receiver obtains additional corrections sent from a ground station. This gives DGPS an accuracy of less than 20 cm[14] - as long as the user is within range of the ground station.

GPS is in many ways an excellent solution to the positioning problem and with a significant growth of GPS enabled products in the past decade, it has turned into a utility used in a vast number of systems. It also has further use as a highly accurate reference clock since each satellite carries an atomic clock that receivers can use to synchronise their internal clocks. This means GPS also has uses in fields such as finance, where it helps synchronise high speed transactions, and power grid management, where high accuracy clocks are used to phase-match power stations with the grid.

However GPS has a significant drawback, in particular from the point of view of an unmanned aircraft. GPS signals are emitted with comparatively low power transmitters, which means that it is easy to jam or otherwise interfere with the timing signals from the satellites. Since GPS is the only widely available absolute positioning system at the moment it is critical that it never fails, but it is easy to purchase a cheap GPS jammer that can disable receivers within a radius of several kilometers.

In addition, several nations claim to have developed more sophisticated jamming where the signals are not simply interrupted but rather modified, allowing them to gradually change the position data obtained by the receiver. For example, the US lost a classified unmanned aerial vehicle in Iran in 2011. Iran claims to have taken control of the vehicle by intercepting and modifying the GPS signals, tricking the UAV into crashing in the north of Iran. Similarly, a demonstration in 2012 showed an attacker gradually modifying GPS signals received by a ship. This allowed the attackers to effectively control the vessel by making it leave its intended route and navigate to the attacker's target instead.[15]

There are several alternatives to GPS either in development or in use, for example the Russian GLONASS system that is currently being upgraded and the European Galileo system. The Galileo project is significantly delayed but will provide similar or better performance than GPS and provide an alternative in case the US decides to close down the civilian channel. Since GPS, GLONASS and Galileo operate on very similar frequencies and protocols, it is possible to develop receivers that can use the three systems simultaneously for improved positioning accuracy and robustness. The drawback is that jammers can easily be extended to disable all the positioning systems at once.

2.2.3 GPS & IMU Data Fusion

The GPS and IMUs are complimentary sensors and are often combined to create a GPS-assisted IMU. This sensor kit uses relative inertial sensors as the primary data source to obtain the position and orientation of the vehicle and filters errors such as drift using absolute data obtained from the GPS. The GPS is also used to align the the coordinate systems and provide consistent positioning data.

This is normally carried out using a Kalman filter, which have been extensively proven to improve the accuracy of noisy measurements and be the optimal solution in certain situations. Kalman filters are predictive filters that estimate what the next measurement will be based on a process model, and then correct itself and the measurement depending on how well the model matched the result. In addition to the updated position estimate, the Kalman filter will also give a measure of the confidence in the estimate through the covariance matrix. This can be used to not only improve the accuracy of the direct measurements, it can also be used to correct for biases and other parameters in the system. A well designed Kalman filter gives very good results, but it can be difficult to model the process noise and they become complex for non-linear processes.

The issue with this system is that it suffers from the same vulnerabilities as GPS. If the GPS signal is lost for any reason then the positioning system falls back on to the

IMU and the position accuracy begins to deteroriate. This is one of the reasons for this PhD, to investigate alternative methods of providing GPS quality positioning as an alternative to GPS when it becomes unavailable.

2.3 External Sensors

The positioning sensor packages discussed in the previous section are a standard feature for unmanned vehicles today. However, UAVs are fundamentally a remote sensing platform and therefore they also carry many different types of external mission specific sensors to capture information about the operating region. External sensors range from electro-optical to radiation sensors, radars, air quality sensors and much more, and each vehicle is usually designed to accommodate multiple sensors. The sensor packages are also usually modular, allowing the ground crew to adapt the UAV for its next mission. These sensors are normally not used for positioning and navigation but in many cases they provide data that is invaluable for these tasks. Thus one of the objectives for this project is to explore ways to exploit the data captured by these sensors and apply it for the positioning task.

This section will discuss the most common types of sensors that are of interest for the visual navigation problem, describe how they operate and their benefits or trade-offs for our operating case.

2.3.1 Electro-Optical Sensors

Electro-optical sensors, widely known as cameras, use optics to project incoming light onto a light-sensitive device. This device, usually a charge-coupled device (CCD) for high quality image acquisition applications, converts the projected image into a digital readout that can be analysed by a computer. Specifically, the sensor contains a large number of light sensitive pixels that produce a voltage potential depending on the projected light intensity (more directly, it measures the number of photons converted into electrons and provides a voltage output). The CCD measures this voltage potential for each pixel and relays the digital measurements to a computer, which converts it into an array of intensities that can be analysed or displayed to the user. Cameras are by far the most popular sensor due to the wide variety in quality, performance and size, making it very easy to find commercial off-the-shelf (COTS) components that suit every type of vehicle.

Electro-optical sensors are normally divided into four types: ultraviolet, visual, near

Table 2.1: Electro-Optics Overview and Examples

Class	Wave-length range (nm)	Example usage
Ultra-Violet	30 - 400	Chemical composition analysis
Visible Light	400-700	Object detection, navigation
Short-wave Infrared	700-1200	Low-light imaging
Long-wave Infrared	1200-2000	Thermal imaging

or short-wave infrared and long-wave infrared (thermal imaging), but there are many other types.

By far the most commonly used sensor is the visual sensor and it is the primary focus for this project. Visual sensors capture light with wave lengths in the 400-700 nm range. This the same spectral range that the human eye is sensitive to, visual sensors are therefore carefully designed to provide the same spectral response as a human eye to ensure a natural and realistic colour representation.

The sensor tends to consist of three types of colour sensitive pixels: red, green and blue (creating an RGB image), that can be mixed together to represent all the colours within this colour space. The difficulty is when modelling the green response as humans are significantly more sensitive to light in the green range (around 525 nm). A CCD sensor's spectral response is different from the human eye (Figure 2.1), meaning that the green sensitivity needs to be increased.

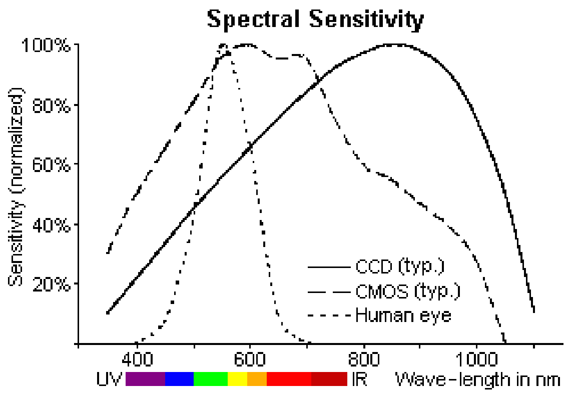


Figure 2.1: Spectral Sensitivity Comparison

While there are simple solutions to this problem, for example by adding a greater than one gain to the green pixels, or conversely applying a less than one gain to red and blue, most of these have a negative impact by increasing noise in the image or forcing increased exposure times (an increase in exposure time means that more light is captured but can cause blurring if the scene is not static). Therefore, most colour sensors today use a pixel layout known as the Bayer-pattern, which increases the sensitivity to green light by adding more green pixels. Instead of having an equal distribution of pixels (1/3 red, 1/3 green and 1/3 blue) the Bayer pattern weights the green to give 1/2 green, and 1/4 each for red and blue (Figure 2.2).[16]

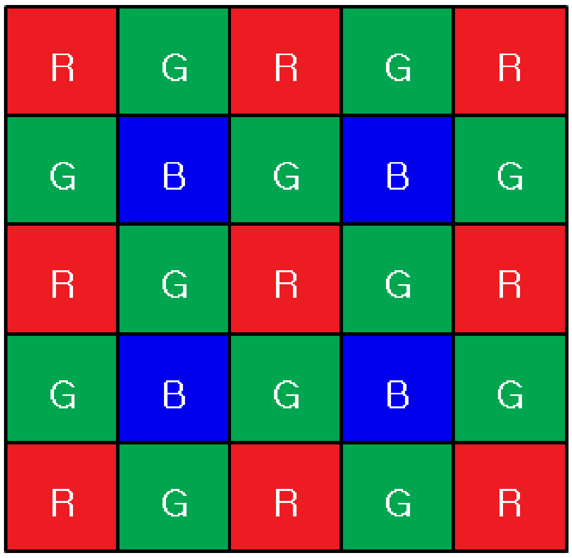


Figure 2.2: Bayer Pattern

Visual sensors come in a wide variety of physical sizes, which depend on the device they are designed for and the required performance. A general rule of thumb is that the larger the physical size of the sensor (and most importantly, the size of the pixels) the better the image quality as more light hits the sensor during exposure. Smaller sensors, such as those used in mobile phones (5.4 x 3.4 mm), are exposed to less light than a full frame professional sensor (51 x 39 mm), which has a number of effects on the image quality. During captures in bright scenes, such as outdoor scenes captured in daylight, the smaller sensors need a good de-noise algorithm to reduce the effects of noise and interference created by electronics surrounding the sensor. In darker scenes the image quality drops significantly as there is less light available, leading to a significant loss of information and a marked increase in noise due to the high digital gains used to squeeze every bit of light sensitivity out of the sensor. Meanwhile, larger sensors provide much cleaner images and are able to capture useful data in darker scenes.

The trade-off is that larger sensors will need optics with a longer focal length (and thus larger physical size) in order to be able to project and focus an image on to the sensor. This is particularly important when using tele-lenses that lets the operator zoom in on specific features at a long distance, where a high zoom factor can result in significantly heavier sensor payloads.

In addition to the sensor size, there are a large number of other factors that affect the resulting image quality of a camera system. These include exposure time, sensitivity, post-processing algorithms and the quality of the optical system. The lens can cause various forms of distortion in an image, most common are barrel distortion (where a wide angle image appears to be bulging), tangential distortion (where the sensor plane is at an angle to the projected image) and also various forms of chromatic aberrations. Chromatic aberration occur when light with varying wavelengths refract and separate, similar to the effect of a prism. This leads to blurring and distortion at the edges of an image and further degrades the image quality.

Finally, for long-range imaging applications such as UAVs and remote sensing, there are also external effects from the atmosphere that affect the image quality. The most obvious is weather, where clouds, rain and snow can significantly change the appearance of objects by changing the perceived colour through shading or by changing the reflective properties of materials.

Due to the large number of the factors that affect the image quality of an electro-optic sensor, it is clear that selecting the appropriate hardware for a mission is not a simple task. Optimally, one needs to know the exact expected operating conditions of the vehicle, the weather conditions and the mission parameters.

This has lead to several advancements in electro-optical hardware. One of the most interesting platforms is called the Argus-IS, a 1.8 gigapixel video system developed by the Defence Advanced Research Projects Agency (DARPA) in the United States. The Argus-IS creates a large, high pixel density sensor by combining 368 five megapixel mobile phone imaging sensors. Four custom-made lenses project an image of the ground onto the sensor plane, which is then stitched together from each individual sensor to create a 1.8 gigapixel mosaic. This enables the system to observe a much greater area at higher resolution than a normal electro-optical system would be capable of. In addition, the Argus-IS can also provide live high resolution video from up to 60 areas within the image simultaneously. This a very useful aspect of the system as it allows multiple operators and potentially multiple missions to use the same sensor data.

The Argus-IS system was first flown by BAE Systems in 2009, which successfully demonstrated the capabilities of the new platform (Figure 2.3. However, the system is still limited due to the vast amount of data that is being captured. To construct a full 1.8 gigapixel mosaic all 368 images must be captured, processed to correct for each sensor's individual colour and distortion characteristics, stitched and finally corrected for the individual distortions of the four lenses. The process requires very significant computational hardware to carry out within a useful time frame, which puts a large electric power demand on the UAV.

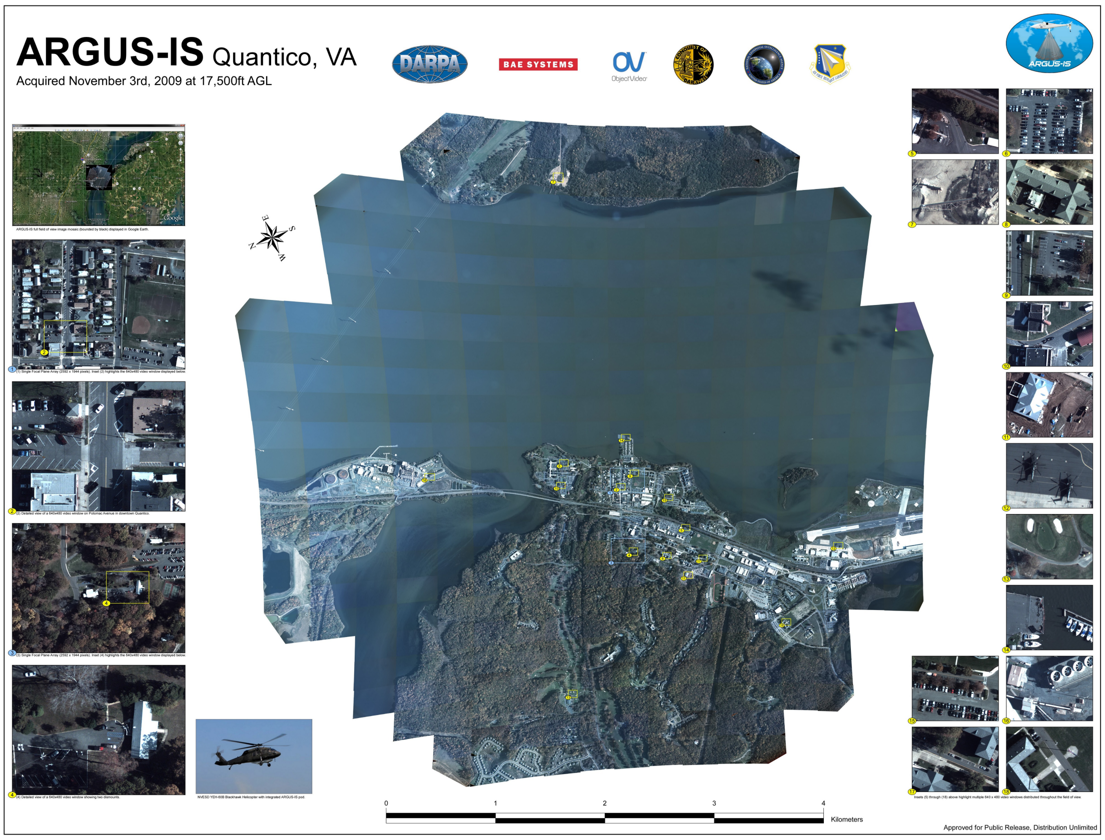

ARGUS-IS Quantico, VA
Acquired November 3rd, 2009 at 17,500ft AGL
DARPA
BAE SYSTEMS
OV ObjectVideo
ARGUS-IS
0
1
2
3
4
Kilometers
Approved for Public Release, Distribution Unlimited

Figure 2.3: Argus-IS Sample

Camera Calibration

While cameras provide large amounts of data and are an invaluable source of information, the raw data is of no use for positioning and navigation purposes due to the various distortion factors inherent in the camera system. Since the distortion factors can significantly change where features appear in the image, one has to accurately model and detemine these parameters prior to capturing data with a camera. A lot of work has been carried out in the area of camera calibration since errors in the calibration model directly impact the subsequent computer vision algorithms.

The calibration problem is generally divided into two issues: intrinsic and extrinsic calibration. Intrinsic calibration models the internal behaviour of the sensor and optics whilst the extrinsic calibration model obtains the camera's position and orientation in an external reference frame. While the extrinsic parameters are useful in certain scenarios (such as in motion capture systems[7] and stereo imaging systems), the intrinsic has an impact on virtually all computer vision problems and is critically important for detection and analysis accuracy.

Intrinsic calibration models the optical parameters of the camera and lens system. In general a pinhole camera model is used where the camera is seen as not having a lens and only a very small aperture. This simplifies the problem significantly and is a close approximation to most common lensed system (the biggest exceptions are fish-eye lenses and aspherical lenses). There are four main parameters to consider:

1. Optical axis centre (c_x and c_y): this parameter defines where the optical centre intersects with the image plane. Theoretically this should occur at the centre of the image in a perfect system ($c_x = width/2$ and $c_y = height/2$), however due to manufacturing inaccuracies it is often offset.

2. Focal length (f_x and f_y): the focal length is the distance from the focal point of the lens to the imaging plane. It is directly related to the field of view and is used to project image coordinates into space. Since most calibration methods do not consider the physical size of the sensor it is not possible to determine the true focal length (commonly measured in mm), thus the focal length is often reported as a relative measure of mm/pixel[17]. In most cases lenses are symmetrical, meaning that the focal length is similar in both the x

and y-direction, but cheap and poorly made lenses or very specific high field of view lenses can be asymmetrical.

3. Radial distortion coefficients (k): Virtually all lenses produce a barrel distortion to some degree. Barrel distortion causes an image to appear to be bulging out (or in certain cases pinched) in the centre and straight lines become curved due to the distortion. This effect is more visible in high field of view lenses and can be described as power function using three parameters, k_1, k_2 and k_3.

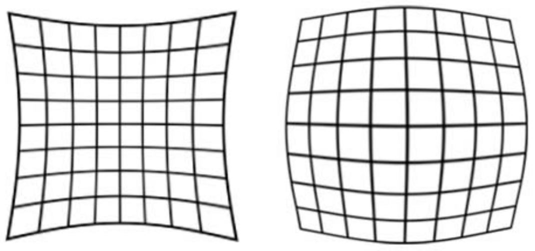


Figure 2.4: Effects of Barrel Distorion

4. Tangential Distortion (p): The final calibration parameter is used to correct for tangential distortion. Tangential distortion appears as a slight perspective transformation to the image and is caused by the sensor plane not begin parallel to the image plane projected by the lens. This is primarily caused by manufacturing defects where either the sensor is not mounted perfectly flat on the processing board or the optical mount is slightly angled relative to the sensor. The effects of a tangential distortion are normally not visible in an image, however it has a noticeable effect when the image is used for geo-referencing, 3D reconstruction and other high precision algorithms.

These parameters are usually determined by solving an optimisation problem in which an object is captured and a comparison is made between the true and predicted positions of certain features in the image. By minimising for the error the system retrieves the optimal parameters to model the lens. Since the calibration process needs a large number of features that cover as much of the field of view as possible it is common to use a chessboard pattern where each corner is a feature. Chessboards are used due to the ease of modelling, detection and association of points on them with sub-pixel accuracy[17], allowing more accurate calibration. This corner detection stage produces around 100 features spread through the image that can be analysed using a number of algorithms, such as Tsai, Zhang and Brown[18]. However, many other patterns and systems have been developed such as bundle adjustment based methods[19][20] that have the potential to obtain more accurate results, in particular when calibrating optics with very wide fields of view.

The resulting parameters are collected in the intrinsic calibration matrix M and are used extensively in other algorithms:

$$M = \begin{vmatrix} f_x & 0 & c_x \\ -0 & f_y & c_y \\ 0 & 0 & 1 \end{vmatrix} \tag{2.1}$$

These parameters are required in order to model the optical system and need to be determined with high accuracy if the camera is going to be used for positioning purposes. Most pose estimation methods, such as homography decomposition, require an undistorted image, otherwise the solution becomes poorly conditioned.

In addition, there are two other matrices that are often used in computer vision, the fundamental and essential matrices. These two matrices consider the projections of points in space onto the sensor plane in multiple view problems such as stereo imaging or monocular 3D reconstruction and rely on extrinsic information about the imaging system. They are not relevant for the work in this thesis, primarily due to the assumption that extrinsic data (position and orientation of the vehicle) is either unavailable or approximately estimated. Despite this, there are multiple view methods that can be used for navigation purposes, which will be discussed in Section 2.4.

2.3.2 SAR & LIDAR

Synthetic Aperture Radar (SAR) and Light Detection And Ranging (LIDAR) are two methods that, while technically different, provide similar results. SAR uses microwave radar and the movement of the imaging platform (such as a UAV) to compute a high resolution 3D map of the terrain (these models are widely used in remote sensing applications). LIDAR, on the other hand, uses lasers to measure the distance to objects, which can be combined to construct a dense 3D map similar to SAR. LIDAR systems are generally smaller and consume less power than SAR systems but have limited range due to light interference from the Sun.

Both of these systems are of interest since they are designed to be operated on airborne platforms and also deliver data that can be used for landmark navigation while carrying out other tasks simultaneously. Several methods have been proposed that detect buildings and roads with high accuracy[3].

2.3.3 The Data Problem

An important problem for most Unmanned Aerial Systems (UAS) today is dealing with the vast amount of data generated by the onboard sensors. A camera capturing high resolution video can generate hundreds of megabytes[1] of raw, uncompressed data every second. Currently, the data is compressed and relayed down to a human operator who reviews the data feed in real-time. This results in a costly and error-prone situation due to operator mistakes (caused by human errors) and results in the data having a significantly shorter useful life-span, as retrieval and correlation of past data is difficult.

This has resulted in two main developments that leads to higher levels of autonomy of the vehicle, more efficient data usage and, most importantly for the proposed visual navigation system, faster analysis of the data.

The first development, primarily driven by practical and economical reasons, is to automate the sensor analysis. Significant work is being carried out in areas such as image classification, detecting vehicles, locating survivors in natural disasters, identifying oil spills and precision farming. Furter work is also being done on incorporating additional meta-data to the sensor data such as when and where it was captured, what is in the data and more to enable faster recovery and more advanced correlation of past data. When the contents of the data is known, higher level analysis can be carried out such as threat detection, behavioural monitoring and process optimisation. This increases the autonomy of the system, reduces the operator workload, improves the efficiency of the system and produces valuable information for

[1] 1920 * 1080 pixels * 3 channels * 30 frames per second * 1 byte per pixel and channel = 186 MB/s

the operators of the vehicle.

The second aspect is more technical in nature and focuses on gradually moving sensor data processing from the ground control station and onto the vehicle. This has two effects; firstly by distributing the processing one can reduce the amount of data that needs to be relayed to the ground, reducing the load and dependency on communication links. Secondly, it increases the autonomy of the vehicle by providing faster and more up to date information to the flight management system. This enables the onboard systems to autonomously plan and make adjustments to the mission at hand and capture data that is of most use to the operators.

These two technical developments have been a fundamental requirement for a realistic visual positioning system. UAVs have generally been simple remotely controlled vehicles that relay a video feed to the ground, they have not been equipped with any additional hardware to enable onboard processing. Further, the hardware and software systems required for a visual positioning system have not been available but are progressively become better understood. The autonomy required for this type of positioning system is still a a number of years away[21] but UAV developments are steadily moving towards this goal.

2.4 Visual Positioning Systems

As with the other positioning systems (such as IMUs and GPS), computer vision positioning systems can be divided into two types: relative and absolute positioning methods. Relative methods determine the movement of the camera relative to its starting point while absolute methods obtain their position in a global reference frame.

Most of the research in the field has focused on relative positioning as most computer vision methods are currently better suited to this type of analysis. Relative methods generally require a simpler, lower level analysis of the imagery captured by the camera, and rely on significant simplifications (such as flat-earth). They also require an understanding of the camera and optics geometry to estimate the motion of the vehicle. Meanwhile absolute methods require a more complete vision system, as per Marr's definition, where there is a need for a higher level understanding of the information available in the image and some form of visual memory.

This leads to a difficulty in deciding which approach is more suitable. Absolute methods provide distinct advantages since they allow a vehicle to carry out longer missions in a greater variety of scenarios but the trade-off is an increase in complexity, as well as energy and computational requirements. Meanwhile, relative systems can be combined with other systems and act as a temporary failover in case a primary system encounters a problem. However, they generally do not perform well enough to be a primary positioning system for an entire mission.

2.4.1 Visual Odometry

Visual odometry is a comparatively old method that uses a technique known as optical flow to estimate the relative movement of the camera. Optical flow analyses the movement of various features and objects in an image from one frame to the next. By computing this movement vector for a large number of features it is possible to obtain a vector field that describes the overall movement of the features relative to the camera.

This presents a challenge when using optical flow, as the vector field contains move-

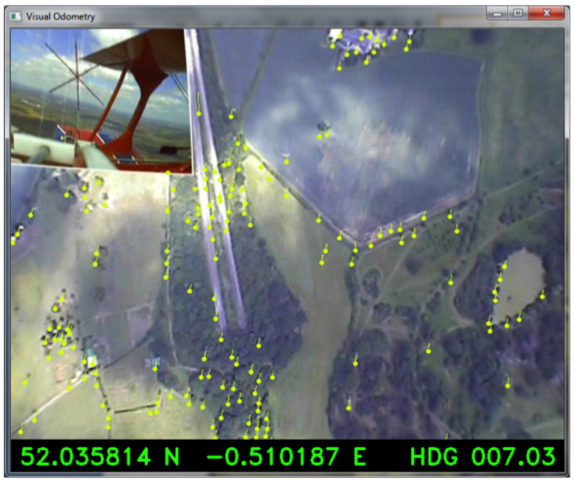


Figure 2.5: Homography-based Visual Odometry from an Aircraft[1]

ment that is caused by the camera displacement, as well as objects moving from one frame to the next. To overcome this optical flow systems are generally very targeted to specific missions and only applied in controlled conditions. For example, optical flow can easily be used to detect a vehicle moving through a road intersection using a stationary camera as the only movement vectors in the image will correspond to vehicles and pedestrians. However, if the camera is mounted on a moving platform, the flow field will include both the movement of the platform as well as the targets and make the detection more complex.

In the case of visual odometry the approach is the opposite; it is assumed that the camera is used to observe the ground and is the only object in the frame that moves significantly. There may be other objects in the frame that are moving but due to perspective and scale they only have a minimal influence on the overall flow field. In addition, these features can be rejected using an outlier elimination method such as RANSAC[22], which is discussed in Section 2.4.1.

Optical flow methods can be divided into two groups: dense and sparse methods. In a dense flow method, such as the popular method developed by Gunnar Farnebck[23], every single pixel is matched across to the next frame and the flow is usually presented as two new single channel images, one for each movement direction. The intensity of each pixel in the new movement images indicate how far the feature has moved in each axis.

Meanwhile, sparse methods detect a smaller number of features in each frame and attempt to match them to the next. This is very commonly done using a corner detector such as Harris[10] or Shi-Tomasi[24] and the Lucas-Kanade optical flow algorithm[25].

Which type of method is more suitable will depend on the task at hand. Dense methods provide more data (for a 640 by 480 pixel image it results in over 300,000 points) while sparse methods only produce small number of feature correspondences. This has a significant influence on the subsequent processing and for this reason time-critical applications such as visual odometry tend to rely on sparse methods. Dense methods are more commonly used to detect movement in a stationary image (such as a car driving through a red-light camera), while sparse methods are used to determine movement of the camera.

In addition, visual odometry for aerial vehicles usually relies on the flat earth assumption where the ground can be represented as a flat plane. This makes the process of estimating the movement much simpler, in particular in scenarios where the camera is mounted on a platform operating at high altitudes since the relative distance to the ground causes hills and other terrain features to appear flat.

By using this simplification it is possible to estimate a direct mapping for the points in the plane identified in the first frame to the plane found in the next frame where:

$$P_2 = HP_1 \tag{2.2}$$

H is known as the homography matrix, a 3 by 3 matrix that carries out the translation and rotation transformation from one plane to the next. It can be estimated by setting up a linear equation system that can be solved using a least squares method. This requires four known point correspondences that are coplanar (but not collinear), however optical flow will usually generate several hundred. For this reason it is very common to combine the homography estimation method with RANSAC (Section 2.4.1) to eliminate outliers and determine the best supported homography matrix.

Following estimation, the homography matrix can be decomposed into its rotational component, the directional cosine matrix R, and a translation vector T[26]:

$$H = R + T * N^T * d^{-1} \tag{2.3}$$

In addition, N^T is the normal vector of the plane being observed and is used to determine the correct translation since the decomposition yields multiple possible solutions. This is done by ensuring that the solution that produces a normal vector closest to $[0, 0, 1]^T$ is chosen, in the cases where the camera is assumed to be normal or close to normal to the plane. In non-nadir operating conditions the predicted normal vector for the estimated ground plane is used instead.

There is however a second problem. Decomposing the homography matrix yields a translation vector that needs to be multiplied by an unknown scale factor to determine the true translation in a global coordinate frame. This comes down to a problem known as the universal scale ambiguity. In a visual odometry system this can be visualised as the unknown distance to the object the camera is observing, which has a direct impact on the ground distance that is covered from one frame to the next. This is not only a problem in visual odometry (for example it occurs in monocular Structure From Motion (SFM) and Visual SLAM problems) and there are several ways to determine the scale factor[27][28]. Most methods revolve around a startup stage where the platform is moved a known, pre-determined distance, after which the scale factor is calcualted as the ratio between the true distance and the estimated value. This method is sensitive to variations in the course while moving and would ideally require either a very stable camera trajectory during calibration or a sensor system that can accurately measure where each frame is captured. This is a very effective solution for a UAV if the GPS and IMU is available, and highly accurate since it can be used to calibrate and correct the visual odometry system during the flight.

RANSAC

Since the overall flow likely contains incorrect movement vectors and other moving objects it is common to implement some form of outlier elimination method. The most common and robust method is known as RANSAC (RANdom Sample And Consensus), which is commonly used when an excess of data is available that needs to be reduced to the points that best fit a given model. For example, it is commonly used to estimate the homography matrix H, since several hundred point correspondences are available (some of which are outliers) but only a small number are needed to determine the matrix. However, the challenge is to select the right points so that the resulting matrix is supported by the majority of the dataset.

RANSAC attempts to estimate the parameters for a given model by generating a number of random hypotheses and verifying whether the candidate model is supported by a random subset of the dataset. This is an iterative process that continues until an appropriate model has been found, and since RANSAC generally is designed to aim for majority support for the model, it robustly handles even large numbers of outliers.

Two components are needed to construct a RANSAC algorithm; a process model that the data should match and a scoring function to evaluate the fit. Since the remainder of the algorithm depends heavily on the data type, model and scoring method it is difficult to create a general implementation of RANSAC (although the algorithm has been described in detail[29]). Below is a brief outline of the algorithm in a simple scenario where a straight line $y=kx+m$ is fitted to a set of 2D points:

1. Select a random sample from the data set.
2. Calculate the parameters (k and m) for the model by calculating the best fit line through the random set.
3. Select a random validation set and validate whether the proposed model is supported. This can be done by calculating the least squares distance from each point to the line determined in step 2.
4. If the parameters are good enough then save them and the related results.
5. Repeat steps 1-4 until a termination condition is encountered. A common termination condition is that a very large number (e.g. 90%) of the validation set confirms the model or that the number of iterations allowed has been exceeded. In case of a forced termination (as in the case of an exceeded iteration counter) the best parameters so far are returned.

2.4.2 Simultaneous-Location And Mapping (SLAM)

SLAM is a relative positioning method that is designed to be used in completely unknown environments. It is in many ways considered the holy grail of robotic positioning and navigation due to its ability to accurately determine a full, relative state estimate for the vehicle with minimal drift[30]. Further, it can be implemented on a wide range of sensors, from basic ultra-sonic range finders[31] to cameras (known as Visual SLAM)[32] and LIDAR[33]. Finally, the approach provides not only positioning data but also a map of the environment that can be used for path planning, obstacle avoidance and other mission tasks.

The SLAM process has been discussed since the 1986, with Smith and Cheeseman's paper[34] that developed a method for representing spatial uncertainty. SLAM has since been thoroughly reviewed and will thus only be discussed briefly in the literature review. Fundamentally, SLAM relies on the use of landmarks, specific features that can be detected in the scene and subsequently found and associated in later updates.

The first step of SLAM, location, uses these features to determine the vehicle pose in relation to the landmarks. If one momentarily, and naively, assumes that the landmark detection sensor is perfect then the relative movement of the features from one frame to the next is the inverse of the vehicle's movement. Therefore, it is clear that one can use the movement of landmarks to determine the pose of the vehicle independently of the vehicle's own sensors and therefore correct the internal navigation system.

The second step of SLAM, mapping, takes the opposite approach where the movement of the vehicle is assumed to be known with perfect certainty but the world is unknown. A mapping update is carried out by requesting a measurement from the landmark sensor. The vehicle then moves a known distance and attempts to detect the same landmarks. Due to uncertainties in the sensing process the detected landmarks will not appear in the same position but data association and fusion processes can be used to refine the location of features and gradually expand the map with new features.

While each step provides valuable data it is clear that neither process can realistically be carried out independently due to the assumptions made in each one. In fact, the two processes are co-dependent; the output from one process feeds directly into the other and can potentially be used to iteratively refine the results of the other. However, SLAM goes one step further by carrying out the two steps simultaneously using a Kalman filter. This allows the designer of a SLAM system to model the vehicle, sensors and other uncertainties and have the Kalman filter[35] gradually refine its estimates of the feature locations as well as the vehicle's states[36].

The Kalman filter provides an excellent solution to the SLAM problem but it also suffers from significant performance problems since each known feature in the world must be observed as a state in the filter. Thus, as more of the world is explored, the number of states increases and the number of calculation required for each update increases as a cubic function of landmarks. Since the Kalman filter relies on the inversion of a square matrix that consists of each feature the number of calculations increase non-linearly with feature count, causing the filter to rapidly increase in time to update.

There are various techniques to avoid this computational load. For example, the system might represent a history of states rather than explicit landmarks (known as delayed state SLAM [37]). This has the benefit that earlier frames can be permanently commited and marginalised out of the filter. This manages the size of the filter but causes drift in the long term. Another alternative is to use an information filter information filters are mathematically identical to Kalman filters but use a canonical form to represent the state and covariances instead of the moment form used in the Kalman filter. This gives an information matrix, which is the inverse of the covariance matrix, and an information vector, information matrix multiplied by the state vector) and simplified maths. However, one of the primary computational benefits of the information filter is that the information matrix only needs to be inverted once for each update (during the prediction step), while Kalman filters require an inversion for each landmark that has been observed by the sensor during the correction step.

Finally, SLAM often uses bundle adjustment methods to carry out loop closures[20], allowing the system to correct itself. Bundle adjustments are data association methods that are carried out when the system revisits an area which has previously been visited. Re-visiting an area allows the SLAM system to correct itself for drift by effectively recognizing well known features and improving the estimated states of system. While bundle adjustment is very useful to improve accuracy for systems that often traverse similar regions, such as a vehicle driving around a city[38], it is a very expensive data association method. A bundle adjustment is carried out by aligning the most recently observed features with all (or a subset of) the previous observation and finding the pose where the error between the two is minimised. This is a very computationally expensive methods, and it requires the SLAM system to maintain the landmark coordinates and covariances for the entire map.

For this reason bundle adjustments are often avoided. In a practical sense, a UAV flying a sortie is unlikely to pass over the same regions very often, which can lead to a very high computational overhead as bundle adjustments need to be attempted over a larger history of observed landmarks.

2.4.3 Visual SLAM

The previous discussion of SLAM have assumed that the algorithm is capable of determining a 3D position estimate of each feature in a single update. While this applies to scenarios where a vehicle is equipped with some form of ranging sensor such as an ultrasonic sonar, a radar or a stereo imager, it does not apply to cases where the vehicle is only equipped with a single electro-optic sensor. A monocular electro-optical system can only deliver the bearing and azimuth of a feature relative to the camera in a single update. Despite this, cameras would be very useful for SLAM since they provide a wealth of data about the world that can be utilised for navigation.

There are two approaches to Visual SLAM, monocular and stereo. Monocular SLAM uses a single camera and triangulates features over consecutive frames[39][26]. Stereo uses two synchronised cameras that triangulate features immediately[40]. Stereo systems are generally easier to work with and can be precalibrated to determine the baseline, distance and orientation between the cameras, and which lets the system determine the exact distance to features and avoids scale problems. However, stereo systems are not viable on a flying platform due to the distances involved, it is generally not feasible to construct a stereo imaging system on a UAV due to size and geometry restrictions.

Meanwhile, monocular systems obtain their baseline by using movement, similar to SAR. The difficulty is determining when a frame should be captured to carry out the triangulation. One consideration is the distance the aircraft needs to move to obtain a sufficient baseline, which is proportional to the flying altitude. Another issue is how the baseline is calculated, it is usually obtained using an IMU but at high altitudes it is likely that the aircraft will have to fly for long periods to obtain a sufficient baseline to accurately triangulate features on the ground. The question is whether the IMU is accurate enough to maintain a good track throughout the capture without relying on GPS, which often is not the case.

The motion can also be determined purely visually using egomotion estimation algorithms, which is commonly used in ground-based SLAM systems[17]. This approach works well in cluttered environments but degenerates when the tracked feature points are coplanar, an issue that occurs frequently in aerial imagery where all features are located on flat terrain.

Because of these reasons Visual SLAM has not been widely developed for aerial systems and is not yet considered a viable visual positioning approach.

2.4.4 Absolute Positioning Systems

The majority of work to date on visual navigation has been in the area of relative systems. While absolute systems would be more useful and suitable for most vehicles, they are also more complex to develop as they require a greater understanding of the information in an image and what the world looks like.

The motivation for an absolute system comes from observing how people use maps. Maps are visual representations of the world that describes how various features in the area are geographically related (the topology). Maps are generally very sparse and only contain the most critical features needed to obtain an accurate understanding of the topology of the region while still providing enough information to complete the task at hand (such as finding a museum in a new city).

A fascinating feature of maps is how sparse they can be and still provide relevant information to a user. For example a Londoner will be able to look at a minimal map without street names or labels of one of the world's largest cities and still be able to identify the exact street where they live. This indicates that maps are a very efficient method for representing topology and, more importantly, that there is some uniqueness in the structure that maps represent. In addition to this, it is possible for a person to not only look at a map of an area and instantly recognise where it is, but they can also determine the orientation of the map as well as get a sense of the scale of the map.

This suggests that there is some structure and uniqueness to man-made features in the world that can be exploited to determine location information. However, if you place a person in a completely unknown area of the world it will take them some time to determine where they are, indicating that there is also a familiarity aspect to the problem.

It is possible to fit this problem into Marr's computer vision model and outline a few initial points for an absolute visual positioning system:

1. The system will need some form of representation of the world that has been reduced to an efficient and suitable format. In addition, since the world is not static, it must be possible to update this representation (although this does not necessarily need to happen in real time).

2. The vehicle needs some form of analysis and recognition system. These systems need to be able to find certain features that are available in the world representation and recognise, or match, these features to the map given a current operational context.

3. The system must also be able to carry out an ego-state estimation after the region has been recognised. Maps are extremely useful tools as they ultimately let the user very accurately determine their own position within the map. The same applies to airborne vehicles which need to determine their position as well as orientation in the world.

Current State

One of the first attempts to demonstrate a visual positioning system with these capabilities comes from Conte[2]. Conte designed, implemented and tested a system onboard a remote controlled helicopter that captures an aerial view of the ground and matches it to a reference satellite image of the same area.

During the development of this system Conte encountered a number of issues, of which the most important challenge was to match the aerial image to the satellite image. His solution was to use a 2D cross-correlation (2DCC) based image matcher. A 2DCC matcher attempts to align a small query image within a larger target image by creating a correlation matrix. The correlation matrix itself is calculated by sliding the query image along the reference image and computing a per pixel score of the similarity. This score is calculated simply by selecting a slice from the reference image with the same dimensions as the query image and then summarising the total scores:

$$S = \sum I_{target}[x : x + w_{query}, y : y + h_{query}] - I_{query} \tag{2.4}$$

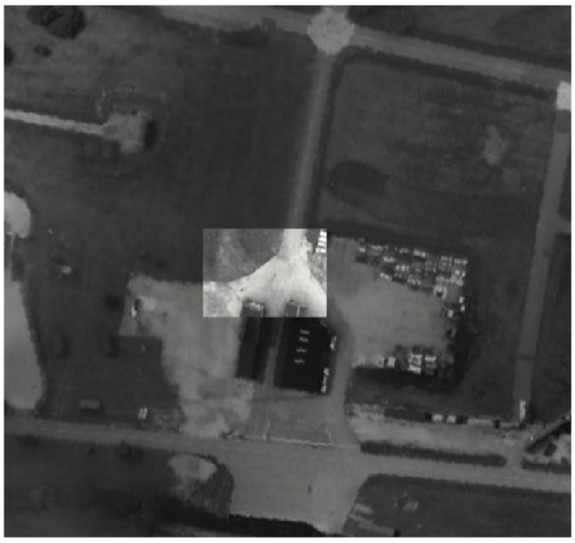


Figure 2.6: 2D Cross Correlation Search (query image overlaid on map)[2]

The query image is positioned at every single pixel coordinate within the target image, giving a two dimensional matrix that holds the score for every possible x and y position in the reference image. Peaks in the correlation matrix indicate a correlation between the query and target image. Thus, theoretically, the row and column with the highest correlation score indicates the most likely position of the query image. Assuming the target image was correctly geo-referenced, it is then possible to determine which region is observed and by extension the location of the vehicle when it was captured.

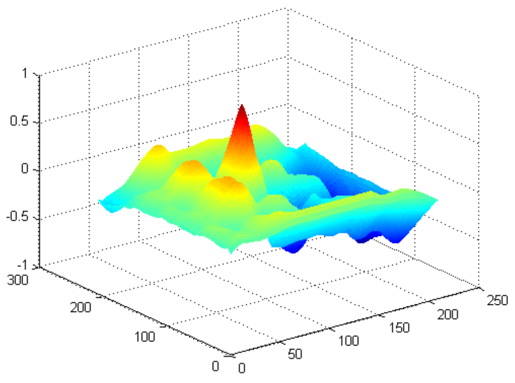


Figure 2.7: Sampel 2D Cross Correlation Matrix (peaks indicate high correlation)

One of the benefits of a 2DCC method is that it can be dramatically sped up by carrying out the correlation computation using Fast Fourier Transforms (FFT)[41], which theoretically allows it to match several hundred queries every second. This provides a very fast and simple matching process but it turns out that this approach is not as robust as it might seem.

Since the 2DCC method is pixel-based and uses the intensity values of the image it can easily get confused by changes in the scene. For example, Figure 2.8 shows what happens when a query image is rotated slightly relative to the reference image. The resulting correlation matrix is completely different, the scores are significantly lower and there is no clear peak indicating the correct alignment. To overcome this, the positioning problem needs to be redefined as an iterative optimisation problem where the query image is gradually rotated until the maximal correlation score is found. Since even a 1-2 $\circ$ rotation can cause the 2DCC to fail, this has to be done at very small steps, meaning that it takes several hundred correlation calculations to correctly handle rotation.

While the process is slow, it has the benefit of providing the system with more data since the orientation is now known. The bigger concern is for aerial vehicles which

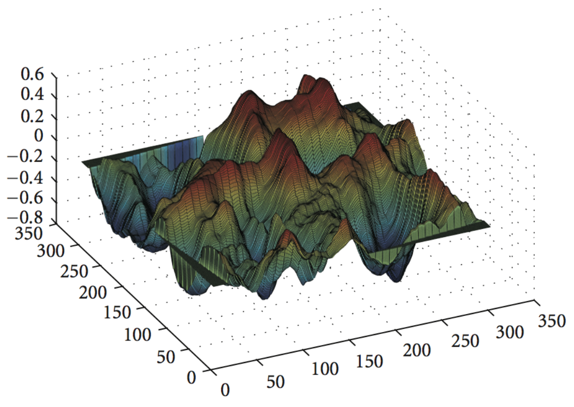


Figure 2.8: 2D Cross Correlation Matrix Affected by Rotation

operate in three dimensions, since two further transformations will be applied to the query image: scale and perspective. Again, if resolved, each of these transformations will provide additional data to the system and will allow it to determine its position in the full six states. Each transformation must be estimated and none of them are independent of the other.

This leads to an iterative process where for each rotation one has to evaluate each perspective transformation and scale combination. There are a total of six independent variables that have to be determined, which leads to a very slow multi-variable optimisation problem. Conte overcame this by obtaining initial position and orientation estimates from the IMU onboard the vehicle, which reduced the total number of matches to less than a thousand. In addition, he further sped up the system by using image pyramids (Figure 2.9) so that the first queries were carried out on lower resolution imagery that was gradually increased in quality as more accurate position estimates became available.

The trade-off for this approach is that the system is incapable of recovering from a lost position or incorrect position estimate due to the time it will take to complete a full query. Furthermore, the time sensitive nature of positioning systems mean that the data would have already expired, even if it was recovered. As a result the system must run continuously since even a small amount of drift or error, especially on the orientation sensors, will require a full match to recover.

Conte did however deal with these problems and successfully tested his method with acceptable results. While it is not a perfect system it remains one the only demon-

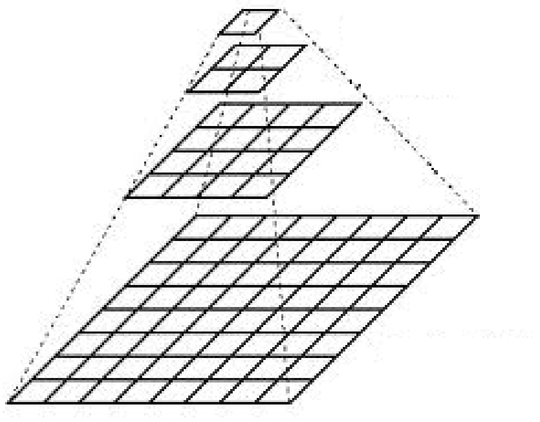


Figure 2.9: Image Pyramid Concept

strations of an absolute visual positioning system with acceptable performance.

There are several ways the system could be improved further, for example by replacing the 2DCC matcher with the Scale-Invariant Feature Transform (SIFT)[73]. SIFT describes individual features in an image in a scale, translation and (partially) illumination invariant way by transforming a visual patch into a feature vector that can be carefully matched from one frame to the next. Specifically it looks at the gradients within a 4 x 4 pixels patch and bins them into a 128 byte feature vector.

However, SIFT is ultimately another approach based on pixel intensities that has a more fundamental problem. These approaches assume that the scene is completely static and never changes. This makes them suited for frame by frame analysis used in tasks such as stereo vision and optical flow but they do not handle temporal changes very well. Since these descriptors and matchers are based on illumination, they are very sensitive to changes in the scene lighting and movement. This causes a number of problems when the methods are applied to real-world data sets, where the data is usually captured at distinctly different times.

A classic example of this is the sun's movement throughout the day, which causes a distinct change in the shading of an image. If the reference imagery is not captured in the same lighting conditions as the query image then suddenly the pixel-based matchers drop significantly in terms of success rates. In addition to this, weather can significantly change the appearance of features in an image by altering the reflectance of objects. For example, sun, rain and snow have very distinct and different appearances. As a result, query images captured in snowy conditions will not match successfully to a sunny reference and vice versa.

Thus, object-based detection methods are more robust, yet more complex, as they focus on detecting specific features in a variety of conditions. While work has been carried out on the detection methods themselves, not much has been done on the topic of using them for positioning. A few basic attempts have been made, for example Gu[42] used a distance and heading measure to attempt to describe a landmark based on its surrounding neighbours.

The descriptor uses the true distance and heading to the nearest neighbour as an anchor point, along with the distance to all other neighbours in a clockwise circle. Their work has partially shown the potential to match landmarks, however the method relies on very high quality feature detection. If the anchor point is incorrectly detected it will lead to a completely different descriptor for the feature since the anchor feature that the fingerprint is generated relative to is now different. In addition, the descriptor is not scale or rotation invariant as it operates in a global coordinate frame, requiring prior knowledge about the orientation and position of the camera relative to the terrain. While this approach has the potential to be tweaked to provide acceptable results if an accurate state estimate is available. It will fail in a fully-lost scenario and will likely suffer in a real-world operational situation.

Ultimately the vision-based absolute positioning problem remains unsolved since it requires a number of methods to be in place before they can be integrated into a complete system.

2.5 Landmark Detection

Since pixel-based methods are too sensitive and unreliable, in particular for matching between images captured at significantly different times, the vision-based system will need a robust object-based feature detection system.

These robust features should preferably be static features with long lifespans to avoid having to continuously update the system. They must be relatively easy to detect and the feature density must be high enough to provide sufficient data for positioning (see Section 2.7).

A brief study was carried out by the author to investigate which features are used by humans as aides for localisation. The study consisted of a simple test where a number of aerial images (rural and urban) were shown to a random selection of students and pilots at Cranfield University. Images were shown for a varying amount of time (100 ms - 5 seconds) to investigate whether there was a noticeable difference between "instinctive" attentional features found when an image was shown for a brief time versus more complex features found when the subject had more time to study the image.

In general, all subjects favoured the same types of features independent of time exposed to the image: man-made features such as roads and buildings. Other distinguishing landmarks were lakes, brightly coloured features or objects that uniquely identified that location. For example, the Xscape Centre in Milton Keynes is uniquely shaped and instantly recognisable.

While this test is useful to gain an understanding of the features could be used for positioning it is important to remember that it is a very basic experiment that was only used to generate ideas. Interpretation of aerial imagery is a highly skilled task that requires a complex understanding of the situation and that clearly improves with experience[43].

The features identified in the experiment can be divided into three classes, each of which would require their own method to analyse:

1. High quantity, low variety features

   These features include residential houses, road-intersections, roundabouts and other landmarks that are abundant and generally share similar characteristics such as shape, colour and size.

2. Low quantity, high variety features
   These features are the opposites to the previous class of features, they are highly unique but only exist in small numbers. These features benefit from the tacit knowledge of the observer, which provides them with substantial contextual information regarding location, orientation and more. Examples include the Xscape shopping centre, airports or monuments such as the Eiffel tower.

3. Attentional features
   The final group of features is unique since it does not rely on a high level analysis or understanding of the image. These features are often noticeable due to other characteristics that cause them to stand out from their surroundings. One such characteristic would be colour; when a brightly coloured warehouse was shown in an image it was consistently picked up as the main feature. Other characteristics include size, texture or patterns but they are dependent on the situation (for example, a large warehouse stands out among residential houses but not when surrounded by other industrial buildings).

The second type can quickly be ruled out for several reasons. First, while a successful match of the landmark would provide a wealth of data, it requires a very significant amount of knowledge to identify it and extract useful data. Second, a substantial amount of work has been made in this field to associate photos of similar landmarks. This work has shown that while it is fundamentally possible when given basic information about the image, a significant amount of time is needed due to the large number of processing steps required[44]. Additionally, these types of features are simply not available in the quantities and densities required for accurate positioning.

As discussed later in the pose estimation section of this chapter, the state estimation algorithms need at the very least four (correct) points to determine the pose of the vehicle. Realistically, a much larger number is desired to ensure that errors due to detection, matching and estimation are minimised. However, assuming a perfect scenario where only four points are needed, the vehicle is operating at approximately

10,000 ft / 3,300 meters and is equipped with a camera that has a field of view of 30° x 20°, then the footprint is:

$$f_x = 3.3km * 2 * tan(30°/2) = 1.77km \tag{2.5}$$

$$f_y = 3.3km * 2 * tan(20°/2) = 1.16km \tag{2.6}$$

and the feature density ρ:

$$\rho = \frac{n_{features}}{f_x * f_y} = \frac{4}{1.77 * 1.16} = 1.94 features/km^2 \tag{2.7}$$

giving a guideline minimum landmark density of 1.94 per square kilometre.

Meanwhile, the third type of features - attention-based features - are interesting due to their low computational cost. Itti-Koch[45] showed that it is possible to emulate the human visual attention response by filtering an image in four different ways (two edge responses, intensity and colour), then combining the normalised response of each filter into a saliency map. The saliency map highlights the most conspicuous features in the image and is very effective for finding certain visually outstanding features. While the filters proposed by Itti-Koch combine to detect colourful and textured features in the image (since these are found to be eliciting the strongest response in humans) it is possible to replace the filters to target other types of features. For example, it is potentially possible to replace or incorporate line or circle detection to identify features similar to roads and roundabouts.

However, there are three problems with this feature class. The first is that many of the attention-based features that humans find are context dependent. A large building will only stand out when surrounded by smaller buildings, to be able to account for this the detector needs a more sophisticated situational understanding of the image. Second, the proposed Itti-Koch method is scale and rotation dependent - texture and edge responses vary depending on the scale of features in the image, making it difficult to reliably extract the same features under varying conditions. This is particularly important if satellite reference data is used to construct the database as Itti-Koch can give very different results for the reference satellite image compared to a small segment of the same image. Third, it is difficult to ensure that the detector is designed to provide the required feature density for pose estimation.

This leaves the first class: abundant and visually similar features. These features are the easiest to detect reliably because of their visual similarity. However, an individual detection does not provide much information about the feature or where it might be geographically located.

There are approaches to deal with this lack of contextual data, which will be discussed in the Data Fingerprinting section. There is also a large number of algorithms available that can detect this feature class using non-spatial characteristics in a variety of data. For example, a simple approach is to utilise thermal imagery. Houses are normally heated and thus warmer than their surroundings, which makes it easy to extract them in aerial imagery (Figure 2.10). This is an approach that can be implemented with minimal computational cost.

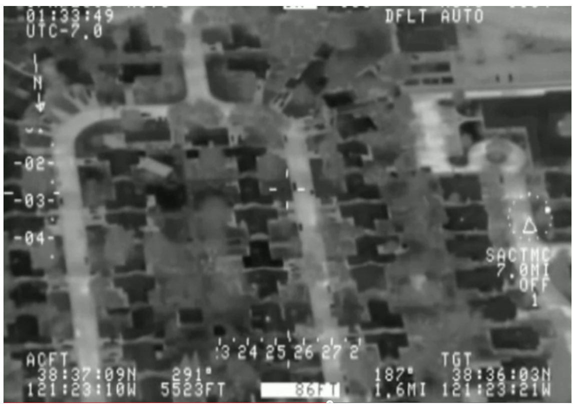


Figure 2.10: Thermal Image of Residential Area

Meanwhile, there are more sophisticated methods that can extract individual buildings and shapes with very high accuracy. These methods are often used in geographic information systems to automatically vectorise aerial and satellite footage but, while they are accurate, they are also slow[3]. It is not uncommon for the building extraction to take from several minutes up to hours for a one megapixel image. Methods exist for road and intersection detection as well, but with similar performance limitations[46].

There are also other examples that extend the use of a vision-based positioning system. Cheng[47] has shown a high quality crater detection method that can be

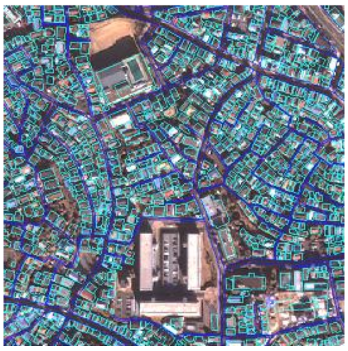


Figure 2.11: Building Detection Example[3]

used to locate a vehicle visually when orbiting an asteroid. The approach is fast and robust since craters are circular, a property that makes them visually distinguishable from their surroundings. Craters are also be excellent candidates for the proposed positioning system as they are completely random and abundant, in particular for vehicles in orbit or approach. Meanwhile, Leroy [48] has developed and tested an integrated inertial VSLAM system using craters to help vehicles navigate during de-orbit, but the system is relative, not absolute.

2.5.1 Reference Data

In moving towards the more robust and sophisticated approach that was presented by Marr we not only need the hardware and algorithms for detecting features, we also need a model. In the case of this project we need a model of the world the system will be operating in so that the positioning algorithms can obtain a position in a useful reference frame.

There are a number of things to keep in mind when dealing with this type of data. Some of the most important questions are, what kind of data do we need? How much data do we need (what is the size of the operating region) and is that data available in our data set? How recent and relevant does the data need to be? How can we efficiently store and retrieve this data? How is a lack of data handled?

The first question about what kind of data we need is the most critical when keeping the rest of the system in mind. The goal of this project is to develop a landmark based system, thus at the very minimum the system needs information about where each landmark is located in the world. It would be useful for the current project as well as further work if each landmark also has some additional meta data such as its type or feature class, geometric data and timestamps for when the feature was added to the database and when it was last observed or confirmed.

Data Sources

With the previous questions in mind it is useful to review some of the data sources that are available.

The UK Ordnance Survey's Master Map is an ideal choice as it contains a topography layer that not only holds landmark location data but also all of the additional meta-data (geometry, timestamps, type) that would be useful in the proposed positioning system.

In addition, the Ordnance Survey also makes extensive use of a topographic identifier known as TOID, now publicly available via the Ordnance Survey's Open Data initiative. The TOID is specific for a feature in the database and remains with it throughout its lifetime, allowing users to uniquely reference specific features. It would be a perfect descriptor to use for matching landmarks, however the TOID consists of a random string that is created and assigned when the feature is first added to the database. As a result, it is not possible to compute a matching TOID at a later stage. However, since the TOID is used by other databases in the Geographic Information field, it is useful to retain the TOID to enable alternative uses of the proposed system. See Section 7.4.5 for a discussion about this.

The Master Map data is provided as raw text in a format known as GML, the Geographic Markup Language. GML is an extension to XML, a very common markup language that can be used for a wide variety of data structure tasks. Since XML is a very popular format, it is very easy to parse these types of files. However there are no freely available parsers that analyses a GML file and outputs geographic objects, thus the end user has to implement this aspect to fit with their application.

While the Master Map is a specific product developed in the UK, virtually every government's mapping and geographic data agency supplies a similar data layer. For example, Sweden's Lantmteriet is selling a virtually identical product known as

Fastighetskartan providing the equivalent data to the Master Map. Most agencies provide data in either GML or Shape files, making it straightforward to acquire up-to-date geographical data for new mission areas.

In the case where building or road layers are not available or other landmark types are needed one has to fall back on satellite imagery that has to be processed prior to the flight. This imagery needs to be orthorectified (removing distortions caused by perspective and uneven terrain) and accurately geo-referenced. There are many providers of this type of imagery, the two most popular are GeoEye and DigitalGlobe, who provide a variety products with varying resolution and spectral bands. However, the most important factor is the age of the data and, depending on the desired landmarks, that the data is captured in conditions similar to mission conditions. For example, instead of using buildings it is possible to navigate using agricultural fields. In the UK, fields are generally unstructured and randomly distributed, making them perfect candidates for positioning in rural regions. However, when using this data it is critical to know what time of year the vehicle is operating since the fields change visually throughout the year and this needs to be considered when tuning the algorithm[49].

Similarly, a visual landmark based system could also be used to navigate on other planets using craters as reference points. The Mars Reconnaissance Orbiter (MRO) has mapped Mars with high accuracy[50], which would be an excellent data source for a high precision positioning system for future orbiters and landers.

However, to use this type of data one would need to develop a high quality classifier that can automatically analyse large image sets, and then ensure that the data is validated. Since this data would be mission critical one would have to ensure that very strict guidelines are in place so that the preprocessed data can be accurately detected by the algorithm operating on the vehicle. This could be done automatically, however it may be better to do this manually by using crowd-based websites such as Amazon's Mechanical Turk.

Coordinate Systems

There is also another aspect of the data that needs to be taken into consideration: the coordinate system. All of the data-sources that have been discussed so far use national coordinate systems to describe where features are located. For example, the Ordnance Survey's Master Map uses the British National Grid (reference OSGB36), where the British isles are divided into a number of grid squares. A location is specified by identifying the square it is located in using a two letter alphabetical code, along with the distance in meters from the lower right corner in the x & y directions. The datum of the grid is located in the Channel, near the Jersey and Guernsey islands.

Meanwhile, most satellite imagery is provided in the World Geodetic System (WGS), a global coordinate system that defines the position of features using an angular measure along the x and y axises (longitude and latitude) of the earth measured from a fixed datum. This datum is approximately located where the prime meridian (a latitudinal line located in Greenwich, London) intersects with the equator.

The WGS was initially developed by the Department of Defence in the United States as there was a need for a truly global coordinate system, in the early 1900's most countries were using national coordinate systems similar to the British National Grid. In the 1950's it was clear that these co-ordinate systems are not practical or accurate enough for demands of a modern global society, hence the need for a truly global coordinate system - the WGS. The initial standard was proposed in 1966, further refined in 1972 and finally completely reworked and ratified in its current form in 1984. WGS84 has since become the de-facto standard for global positioning and is the coordinate system used by virtually all positioning systems today, including GPS. As a result, it is also used extensively by unmanned vehicles since they often obtain their global position estimate from GPS.

However, WGS84 has an undesirable property from the point of view of a visual positioning system. The system is based on an angular measure of a location on the surface of a sphere, one degree in longitude (along the x-axis) covers different absolute distances depending on the latitude (y-location), since a longitudinal circle varies in radius with latitude. This also applies, althought to a lesser degree, to the latitude since the Earth is not perfectly spherical. This means that objects that are defined in WGS84 will distort in a metric coordinate frame at higher latitudes, as a result the data saved in the reference database would be different from what would be observed by an airborne vehicle (specifically it'd be compressed along the x-axis).

The solution to this problem is to introduce a third coordinate system that is fully metric. The Spherical Mercator is similar to WGS84 but defines points using an absolute distance (in meters) from the datum. Thus it is possible to avoid the distortions caused by WGS84 and ensure that what is observed matches the data that is available in the database. However, Spherical Mercator assumes a perfectly spherical Earth, an acceptable assumption in our proposed case as it would only cause very marginal differences to the projected points. It is important to keep this assumption in mind since it would have a much greater effect in a full 3D scenario and would require a more detailed model.

As the Spherical Mercator projection is a less accurate coordinate system it is recommended that the data is stored using WGS84 to avoid loss of precision. Projecting points from one coordinate frame to another is a straightforward task and there are several open source packages available that lets software developers easily carry out projections between hundreds of coordinate systems. One such example is PROJ.4[51], a package initially developed by the US Geological Survey.

The OGP (International Association of Oil & Gas Producers) Geomatics Committee maintains a list of most common coordinate systems, allowing users to refer to a standardised list of coordinate systems using a unique identifier. This system has been inherited from the now defunct European Petroleum Surveyers Group (EPSG). As a result the identifiers are of the format EPSG:XXXX where XXXX is a numeric serial number referencing the exact coordinate system. For example, WGS84 is also known as EPSG:4326.

2.6 Information Retrieval

An initial review shows that a navigation system based on matching raw visual data, such as Conte's system, is not viable approach for a robust system. The alternative is to extract and match individual features to a database but this presents a different challenge since the database will inevitably contain a vast number of features. It is likely that a complete feature database would contain several million features, in which case the process of finding and retrieving the matching features becomes nontrivial. Additionally, since the positioning system is time-sensitive, there is a strong emphasis on retrieving results with minimum delay.

It is also important to define exactly what the task for the retrieval system is. For example, there are many methods available in the computer vision world where one can submit a query image and retrieve similar or even identical images[52][53], even when parts of the image has been distorted or rescaled. These methods usually rely on content awareness, where the query image is analysed and reduced to a number of keywords (a technique known as *bag of words* or *bag of features*)[54]. The size of the target image set is reduced drastically by eliminating all images from the database that do not match the keywords in the query image. After this initial filtering it becomes feasible to use an alternative method such as SIFT to extract feature points in the query and target sets, thus making it possible to find the most likely match.

However, the task at hand differs since the system does not have to find a similar feature, it has to find the exact match, otherwise the positioning task will fail. It must also be robust to distortions and preferably carry out a reduction of the reference data in order to avoid having to upload several gigabytes of imagery to the vehicle prior to the flight.

2.6.1 Data Fingerprinting

The data fingerprinting field explores ways of describing data in an efficient and unique way, so that it can be matched to sets of known features or be used to identify similar features. Fingerprints can be very simple, a common and efficient example are binary feature vectors where each element in the vector indicates the presence of a certain property. Feature vectors have been used in a range of problems from identifying plants[55], to finding movie recommendations[56] and even determining the personal traits of people based on their social network data[57]. Meanwhile, more complex examples such as SIFT[58] compute a large, numeric feature vector that holds significantly more data to ensure uniqueness and precision when matching at the cost of performance.

Since this is an area of active research this project has limited its focus to methods that are used to finding specific matches. A significant amount of work has been done on similarity matching, prediction and suggestions using machine learning but these approaches are not of use for this project. Positioning requires exact matches to work.

Much work has been carried out in the field of specific matching, there are numerous research projects and commercial products available. However, very little of the work has been in the geographic field required for the proposed vision system. Instead the main focus has been on more commercially viable use cases.

One such example is audio fingerprinting where a song is identified based on a short recording made by a user's mobile phone. This is a challenging problem since the recording has been distorted by low quality audio hardware and is usually not particularly clean, the recording tends to be contaminated with sounds of the environment around the users. This audio clip has to be analysed in a way that lets a short (a few seconds) recording be matched against a vast database of known songs (one company can match over 50 million songs based on a five second clip) in just a few seconds. There are several types of methods for this problem created by companies such as Soundhound and Shazam.

Shazam's algorithm uses multiple steps to match a recording to its database[59]. The first is feature extraction where a spectrogram is generated from the recording. The goal of this step is to extract specific key points in the recording that are considered to be robust to noise based on intensity and frequency. This results in a 2D-plot that is due to its similarity to a star field is called a constellation map. The features are then further analysed for robustness and a number of these key points that satisfy certain criteria are then considered to be anchor points that will be matched in the database. Next, individual constellations are created around the anchor points by selecting nearby key points and a descriptor is calculated based on the frequency and relative temporal positions of the key points. This results in a number of unique descriptors that only one specific song within the database would match. However, matching all the raw descriptors to the database will take significant time, therefore the authors developed a fast hashing method that converts the fingerprints into a 32 bit long descriptor. Finally, a custom search algorithm is used that provides a 10,000 times faster search than normal methods by exploiting the specificity and high entropy of the fingerprints.

This allows the exact song to be identified in less than 10 ms for a radio quality recording (relatively poor quality with some noise interference) against a database of approximately 20,000 songs. This approach has proven to be very effective at matching even with compressed and noisy data, indicating that it would be a relevant method for an image based method as well. However, the audio algorithm has the benefit of only having to deal with two-dimensional data, time and frequency. Meanwhile, imagery and other mapping data is more complex, requiring greater care when constructing the descriptor and matching methods.

Audio fingerprinting is not the only use of data fingerprinting algorithms. These methods have also been used extensively in other fields such as chemical analysis where the bonds in chemical compounds can be described in a binary fingerprint and matched to a database[60].

2.6.2 Relational Mapping and Matching

Another common approach within information retrieval systems is relational mapping and matching. In relational mapping features are joined together in a graph network that is structured to group similar, or more closely related, features near each other. This approach is commonly used to find similar items or the most relevant results for a given query. For example, this is used extensively in social networks to map out connections between people and identify items that could be of interest to the user[61].

Meanwhile, graph theory is also used extensively in geographic tasks to map out networks such as transportation or power networks. Most importantly, they can also be used within cities to construct a graph that describes the appearance of the city and the physical relationships between features. An example of this is navigation and routing where intersections are turned into nodes in a graph.

It also allows graph matching methods to be used to match a small part of a graph to a reference. This approach is of interest as it does not rely directly on the physical location of features but rather describe the relationships between them and enables matching in a connectivity space rather than feature space. Thus, it would theoretically be possible to extract features from an aerial image, describe the relationship between them and match them to a known graph.

In addition to matching, graphs can also be analysed to evaluate properties of features in a network, such as the connectivity (number of neighbours), or as an input to a matcher. For example, it is possible to determine whether a region is likely to match based on the quality of the graph or whether the matching process should be delayed until higher quality data is available. This will be discussed further in Chapter 5.

In the end, information retrieval is an important aspect of a positioning system but the design of the retrieval system is highly dependent on the task at hand. The system selected for the visual positioning system is described in Chapters 3-5.

2.7 Pose Estimation

Once the features in the image are known and associated with their real-world locations the remaining task is to determine the pose of the camera (and thus the vehicle) when the image was captured.

Pose estimation is a well researched problem in computer vision, that ranges from simple planar pose estimation to VSLAM where the camera's position and location is gradually optimised and refined as more measurements are obtained. This is done using point correspondences, where a pixel location is matched to a known point in 3D space.

If the system is capable of obtaining exact global coordinate matches for features in the image and the imaging plane is parallel to the ground plane then the problem becomes very simple to solve since only four parameters need to be determined: x, y, z and heading. X, y and heading can be found using simple geometry and the height can be obtained using the camera model. If the image is undistorted and the field of view is known, then the altitude can be obtained by constructing a triangle with its base between the two ground points. By computing the angle between the two points it is then possible to determine the altitude using straightforward trigonometry.

While this works in principle it is a naive approach as small errors in detection will lead to substantial positioning errors. In addition, the assumption is that the vehicle is equiped with a perfect gimbal that always maintains nadir lock. This is highly unlikely in a real situation.

The easiest way to deal with this issue is by using homographies, which were described in Section 2.4.1. Homographies are mappings between two planes and thus require the flat earth assumption to generate valid results but it is a straight-forward and computationally cheap method to estimate the vehicle's position. The output of the homography decomposition will be the translation relative to the global coordinate system and the orientation relative to the ground plane, thus giving a full pose estimate.

A second alternative is full 3D model-based pose estimation, such as POSIT[62] or ObjPose[63]. These approaches require a better understanding of what is being observed, specifically they need an accurate camera model and an the exact 3D locations of the features on the object that is being observed. They then use iterative methods to minimise the reprojection error between the observed feature location and the predicted location based on the object and camera model. This allows the method to find the optimal, or least costly, orientation of the camera relative to an object. The benefit of these methods is that they support a 3D terrain model and can therefore estimate the position of the vehicle with much greater accuracy. They also provide their results in the global coordinate frame, meaning that terrain height is being accounted for. However, since the proposed system relies on the flat-earth assumption there is no benefit from these approaches at this stage.

Visual SLAM systems often take a different approach by solving the PnP problem[64][65], where n points (between 5 and 8, with varying numerical performance) are associated between two frames. Solving the PnP problem resolves the position of the points in 3D as well as the optimal camera poses for the two scenes. However, the solution from PnP is subject to an unknown scale factor that needs to be resolved using external sensors. This makes the PnP methods unsuitable for the proposed system.

2.8 Development Environment

A final area to review is the development environment. Having an environment that is closer to the performance of a real platform allows a better understanding of the behaviour of the new algorithms and makes it easier to investigate how it would be designed and implemented in a real scenario. While the real-world usage of these methods is still far away, it is not a good idea to develop new methods in "silos" without a consideration for how they would perform with real data and real hardware.

A secondary area to consider is the future work on this project. By selecting a portable, easy to learn, development environment and documenting the project well it is more likely that future work will be carried out without having to spend time on reimplementing functionality or repeating work in other ways.

There are currently three main programming languages or development environments used for computer vision algorithm development: Matlab, Python and C++. Each of these have a number of benefits and drawbacks that make them suitable for various stages of development.

Before reviewing them however it is worth briefly discussing the difference between compiled and interpreted languages as it has a very important impact on the performance of each language.

> Looking at a program written in machine language is vaguely comparable to looking at a DNA molecule atom by atom. D. Hofstadter, Gdel, Escher, Bach (1980). An Eternal Golden Braid. p. 290

Code is usually written in a human-readable language and syntax to simplify development. This code cannot be directly executed by a processor and therefore must be translated to machine code before it becomes useful. There are two ways to do this, using either a compiler or an interpreter.

A compiler is used to create a standalone executable that anyone (with the correct operating system and hardware platform) can run. It interprets the code written by the programmer and, via a number of stages, converts it into machine code that can be executed straight away. This results in machine code that is highly optimised for a specific processor, giving fast performance at the cost of flexibility. It is also generally a one-way process, while it is possible to decompile executables it is rarely done due to the complexity of a modern compiler and operating system.

Meanwhile, an interpreter is an application that sits between the code written by the programmer and the operating system. While a compiler prepares an executable once, the interpreter analyses and compiles the code every time the user is executing the application. It does so by analysing the code line by line and executing ready-made machine code fragments for each function, thereby executing the code on-the-fly. This has a few advantages because it makes development easier and the code becomes more portable. A compiled executable is locked down to one platform while an interpreted application can run on any platform as long as an interpreter is available. It also means that instead of having to port every application to a new platform one simply has to port the interpreter, which can often be much smaller in terms of lines of code. However, it also means that it is less likely to be optimised for the platform it is running on (there is a clear difference in performance between the two types of languages).

- Matlab & Simulink

  These two products were initially developed to model dynamic systems and design control systems using an in-house programming language Matlab. Matlab is designed to let users rapidly develop and test new algorithms and can be extended with new toolboxes that allow the user to work on a wide variety of problems. One of these toolboxes is the Computer Vision Toolbox and contains a number of functions for importing, manipulating and analysing imagery in either Matlab or Simulink. This makes Matlab very easy to get started with and means it is often used as a very first step to develop and test new methods, especially since it is a very popular product and many users are familiar with it.

  There are however two important issues with Matlab:

  1. Performance

     Matlab is an interpreted language meaning that the code is read, interpreted and executed on the processor line by line during runtime. In Matlab's case the interpreter is very poorly implemented leading to a significant performance hit, making realtime operation of anything beyond very simple tasks unrealistic. Similarly, threading and other methods to increase performance are very poorly supported.

2. Portability
   Matlab code can only be executed in its own interpreter and with the supporting toolboxes. These parts of the application are not easily portable to other platforms and can only be run on an x86 processor. Meanwhile, most platforms used in real-world missions run on low-power ARM processors, Digital Signal Processors (DSP) or FPGA arrays, all of which require distinctly different development tools. This means Matlab code has to be completely redesigned and reimplemented in a separate language before testing on real hardware.

- Python
  Python is an open source interpreted language that is designed to be efficient, capable and easy to learn. The syntax is similar to Matlab but the interpreter is highly optimised and can be several hundred times faster than Matlab. In addition, most underlying and supporting libraries are implemented in C/C++ for performance and Python itself has been ported to virtually every operating system and hardware platform in existence. It is very commonly used as a prototyping language among researchers but also has many uses in production environments. For example, it is used extensively in financial trading software and many web services rely on Python (with the help of various frameworks).

  Being an interpreted language it still suffers from a performance penalty when compared to C++. A particularly limiting factor is a design factor made when the Python interpreter was created that prevents it from running multiple threads. The language itself supports concurrent threads but the interpreter is incapable of executing them simultaneously on multiple processor cores due to a Global Interpreter Lock (GIL). This can be overcome to a degree by running multiple interpreters in parallel but performance will still suffer when data is transferred from one parallel process to another.

- C++
  While Matlab and Python are interpreted languages, C++ is a compiled lan-

guage where the code is converted into executable machine code by a compiler before runtime. The result of this is highly optimised and fast software that can take full advantage of the hardware it is running on. However, while interpreted languages manage memory and optimisations automatically, C++ does not and requires the developer to carefully test their code and ensure that it performs as expected. It is also platform specific, making it harder to move the codebase from one operating system or processor to another.

Both C++ and Python can be used for computer vision using one of many widely available computer vision libraries. The most common library is known as OpenCV, the Open-source Computer Vision library, maintained by the research company WillowGarage in the United States. OpenCV comes with a wide variety of functions to capture and analyse imagery including algorithms for calibration, optical flow, pose estimation, object recognition, 3D reconstruction and more.

A quick performance test comparing the three languages by creating an array and carrying out various simple calculations in memory (without copying or moving data) has shown that Matlab is approximately 200 times slower than the same implementation in C++, and 100 times slower than Python[7].

This leaves a choice between Matlab, Python with OpenCV or C++ with OpenCV. While Python is a little slower than C++ it results in code that is easy to understand and allows the developer to focus on the work that needs to be carried out rather than the specifics of the language. It can also easily be put on a real hardware platform for initial evaluation and give a reasonably accurate idea of how the overall system will perform in the field.

2.9 Conclusion

Vision-based positioning systems are more commonly used today thanks to an advancement in processing power and the maturation of the underlying algorithms.

As a result there are now substantial amount of techniques available, ranging from odometry to full SLAM systems capable of extensive operation with minimal drift. The common drawback for all of these methods is that they are estimating their position relative to a starting point and most forms of visual interruption will cause the system to fail.

This is one of the main arguments for the development of an absolute system, it would be of much greater use in a real mission since it can recover from visual interruptions. The second feature of an absolute system is that it can act as an alternative to GPS meaning it can be used as a replacement in environments where GPS is either being intentionally jammed or is simply not available. It can also act as a redundancy for GPS, thereby making the vehicle more robust and resistant to hardware failures.

Developing an absolute vision-based positioning system is not a trivial task and it requires an understanding of a large body of work before a meaningful contribution can be made. This literature review has explored some of the history behind computer vision and positioning systems and has reviewed a wide range of computer vision and computer science techniques that are relevant to the proposed system.

These techniques have loosesly been fitted to David Marr's framework for a robust vision system, which states that three problems need to be sovled in order to create a reliable system: representation, analysis and implementation. Most of the work in this thesis will focus on the first two problems, representation and analysis, but there will be strong emphasis on implementation considerations throughout.

In summary, four core topics have been identified that need to be addressed in order to develop an absolute vision-based positioning system:

1. Detection - identifying features within the image
2. Description - unqiuely describing each feature

3. Matching - matching the features to a reference data set
4. State Estimation - obtaining the position of the vehicle use feature matches

The core argument of this review is that an absolute vision-based positioning system is a challenging but not impossible task. A substantial amount of work has been made in all of the areas above, some of it directly applied to computer vision or positioning while other work has been in other fields such as information retrieval.

For an early prototype, the detection and state estimation problems can be considered to be solved. There are multiple methods for detecting various forms of landmarks with high accuracy and low computational cost. Similarly, given a number of image to world matches and an accurate camera model, it is possible to estimate a camera's position relative to the world with a high level of accuracy.

This leaves description and matching, uniquely describing a detected landmark and matching it to a set of reference data. These two areas have been widely researched in other fields and all of the required ingredients are available but there has not been an attempt to apply these methods in the vision-based positioning problem.

Description and matching form the core of the following chapters, which will begin by outlining an overall architecture for a vision-based system and then continue on to explore an initial solution to these two problems in detail. This includes a method to describe landmarks in a unique, scale, rotation and translation way, as well as the development of a geographic landmark matcher.

Chapter 3

System Overview

The system proposed in this thesis aims to overcome the challenges encountered in previous work by using a high level object-based matching strategy where individual landmarks (such as buildings in a city or craters on the Moon) are detected in an aerial image using feature classifiers and converted to single point features.

These landmarks can be uniquely matched to a global landmark database using a geometric descriptor, which enables the system to retrieve their real-world locations of the landmarks. The benefit of this approach is that it allows the system to discard a vast amount of the data that is captured by the sensor; data which is of no use in matching landmarks.

The matching process associates the landmarks with a corresponding global feature, giving a local-global feature correspondence. The camera's pose, and thus the vehicle's position, can be recovered once a sufficient number of feature correspondences have been found.

This system is envisaged to be used in conjunction with other positioning methods, where the proposed absolute system provides positioning updates at a rate near GPS frequency (1-5Hz) and other faster methods such as visual odometry and inertial units provide interim data for the flight control system. It is unlikely that a vision-based system would be a primary positioning system as long as GPS is available but it is a strong candidate for a redundant positioning system or for use in situations where GPS is unavailable.

The system described in this thesis has initially been designed to operate on a UAV equipped with a downward looking camera mounted on a perfect gimbal, allowing the vehicle to maintain the sensor plane parallel to the ground. This assumption simplifies the matching and pose estimation problems, but Chapter 7 will discuss how this can be improved in the future. However, there are cases where this assumption is valid, most notably in satellites, which makes this assumption valid for initial development.

3.1 Architecture

The core of the system is the control architecture that manages data from capture, through processing and on to external systems such as the autopilot. This process consists of four parts that have been designed to be modular and interchangeable so that the system can easily be modified for various usage scenarios. This modularity significantly simplifies development and testing and also allows the system to be used with a range of vehicles with only minor modifications. Finally, it also makes it easier to parallelise the tasks in the future, thereby speeding up the overall system on a multicore platform.

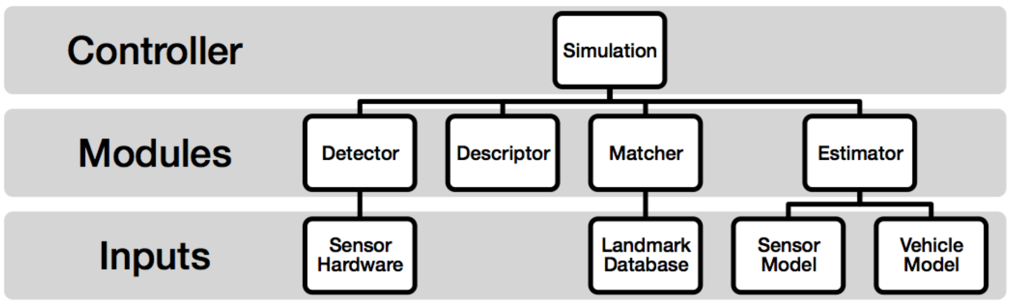


Figure 3.1: Overview of System Modules

Figure 3.1 shows a simple overview of the structure of the system. At the top of the system is a *Controller* that initialises hardware and loads the relevant modules for the current mission, which includes specific implementations of a *Detector*, *Descriptor*, *Matcher* and *Pose Estimator*.

In addition to these main modules, there are also other components such as feature selectors, database interfaces and communication modules required for full system integration. The controller manages these as well as the overall analysis and communication with other systems on the vehicle.

The entire system has been implemented in Python, "a programming language that lets you work quickly and integrate systems more effectively."[66], and depends on the following open source projects:

- Python 2.7[66] - programming language
- OpenCV 2.3[17] - computer vision library
- bitarray 0.1.0 - bit array manipulation library
- pyspatialite - geospatial database interface

However, this is a research development environment chosen for its flexibility and relative performance. It gives a good indication of the final performance of the system but due to the choice of an interpreted language the results in this thesis will not be fully representative of a real-world usage scenario.

3.1.1 Module Functionality

The system consists of a number of core modules that can be combined to create a suitable processing pipeline for a variety of missions. Below is a brief summary of the functionality required from each module as well as its inputs and outputs.

Controller

The controller manages all other subsystems and is essentially running the show. It handles initialisation, process flow, data flow and is envisaged to take care of error handling and parallelisation in the future.

Inputs

- Modules
- Processing script
- Hardware interfaces

Outputs

- Vehicle position

Detector

The detector analyses raw data and generates a list of local coordinates for features found in the data set. As discussed in the literature review, there are a wide range of techniques for this purpose and the correct algorithm will depend on the sensor, the mission and the available computational power. A common and efficient solution that will be used as an example in this thesis is the crater detector since it is a robust, efficient and proven algorithm.

Inputs

- Visual data
- Environmental and situational data

Outputs

- List of point features in local coordinate frame (parallel to ground-plane)

Descriptor

The descriptor processes the list of features identified by the detector and computes a unique identifier for each landmark. The development and performance of this process is described in Chapter 4. The system uses a geometric descriptor that encodes the feature's geometric relationship to surrounding features in a scale, rotation and translation invariant feature vector.

Inputs

- List of local point features

Outputs

- List of features in local coordinate frame with associated feature descriptor

Matcher

The matcher takes over once the descriptor has computed the identifier for each landmark. This module handles all steps required to associate features with their corresponding global coordinates. This includes eliminating weaker and less suitable features from the process, fencing the target global region, selecting appropriate matching strategies and final association of strong features (see Chapter 5).

Inputs

- Local features and descriptors
- Estimated position and error
- Interface to global feature database
- Descriptor scoring method
- Feature selection method

Outputs

- Associations between local and global features

Feature Selector

The feature selector is a submodule in the Matcher which uses a variety of techniques to identify weak or otherwise unsuitable features from the matching set. The selection process is highly dependent on the rest of the system as it needs to take the specifics of the sensor, descriptor and matcher into account. A discussion and initial development of this module can be found in Chapter 5.

Inputs

- Local features and descriptors

Outputs

- List of strong features and their descriptors (optionally with an associated score)

Pose Estimator

The last core module takes the point correspondences and determines the position of the system, given a model of the sensor.

Inputs

- Local and global feature correspondences
- Sensor model
- Estimated current / prior position

Outputs

- Position, altitude and heading of the vehicle in WGS84.

3.2 Data Management

The initial aim for this system is to manage a small number of features that can be expected to be found during a short mission (an estimated area of 20 km^2). However, the goal is for the system to be suitable for unmanned vehicles that carry out long endurance missions and can be expected to have nearly global coverage. This results in a vast number of features that need to be obtained from a suitable data source, be in a standard format and be accessible enough so that the matcher can retrieve a relevant subset with minimal delay.

3.2.1 Reference Data

This thesis makes extensive use of data from the Ordnance Survey's MasterMap as it contains a vectorised building layer. The building layer is a GML file that contains the outline of every man-made building in the selected region. This data can easily be reduced to point features by computing the centroid of each outline.

However, the MasterMap data presents a challenge as it separates parts of buildings into different entities with unique TOIDs if they were not constructed at the same time (such as an extension to a house). This can lead to problems during matching since the global database appears to contain additional features that are not present in the aerial capture. While the system is capable of handling a small number of unexpected or changed features, the MasterMap contains every modification to every building that requires planning permission. This means that if each raw shape in MasterMap is reduced to a point feature the number of features in the set nearly doubles, yet the feature locations are not matching what will be observed from the air.

To overcome this, intersecting or joining shapes have been merged before the centroid for the building is computed (this process assumes that the building detector can compute the complete outline of the building).

This is currently not an issue as the focus of this thesis is to demonstrate the use of geographical relationships to match landmarks, thus the structure of landmarks is of higher importance than fully accurate landmark detection. Once the description and matching techniques have been refined it is worth revisiting the detector and finding methods to obtain source data that closely matches what the detector will be observing in air.

Finally, the data has been further simplified by using a flat-earth assumption. All 3D data has been projected onto the ground plane as described by the WGS-84 global coordinate system. This results in a loss of elevation data, but since one of the main assumptions behind the initial development of the system is that the vehicle produces a nadir image, the effects of elevation is minimal in the image and can be removed completely.

To enable matching, the reference data has been also processed by the descriptor so that each landmark has a corresponding feature vector. However, the data in MasterMap is in the British National Grid system. Meanwhile, the descriptor uses a local metric system and the positioning system is designed to provide data to other systems in WGS84. To overcome this the raw MasterMap data is first reprojected to metric World Mercator (EPSG:3395) before being passed to the descriptor in order to maintain the correct aspect ratio of the region during the fingerprinting process.

3.2.2 Database

The reference data can produce millions of features that need to be stored and accessed quickly. In particular, the speed of retrieval is critical to a navigation system since the relevance of the results of the system is dependent on time. Thus, naive methods such as storing the features in a text or XML file are can immediately be ruled out, not only due to the difficulty of managing large versions of these files but also because of the difficulty in carrying out geographical queries (such as *retrieve all features in the region (15,10) and (20, 15)*) on the data.

This is a common problem within geographic information systems and there are several high-performance databases available that enable efficient access to features within the dataset. A classic example is the commerical application ArcGIS, while the work in this thesis has been developed using the open source database Spatialite.

Spatialite is a geospatial database built upon the SQLite database engine[67]. It extends SQLite by implementing a large number of geographical queries and an efficient method for managing geographical data. This means that data can quickly be accessed using geographic SQL queries (such as *SELECT centroids FROM landmarks WHERE MBR(...)*[1]). Spatialite and SQLite are designed to be portable and easy to use rather than optimal in terms of performance, but are easy to integrate into a new system and can be ported to a wide range of platforms. However, their performance may suffer when given very large datasets.

To further simplify and standardise development, the thesis uses an intermediate database interface that provides a standard API to the rest of the modules and translates the calls into a relevant database query. Similarly, the database interface translates the results from the database to a standard object oriented format used in the rest of the system. This has two benefits: it standardises and abstracts away access to the database and it makes it easy to modify the database interface, which lets the system use alternative databases in the future.

In addition, the database interface also implements a number of functions that are required by the matcher. For example, the matcher makes extensive use of not only the target feature but also its nearest neighbours. The list of nearest neighbours is populated when the database is generated, but retrieving all the neighbours can become a convoluted process if it is not carried out at a database level. Thus, the database interface handles these tasks as well, and provides simple functions for the matcher. An example of such a function is *getNeighboursForLandmarkID*, which retrieves all the neighbours surrounding the given landmark.

3.3 Conclusion

This chapter has given an overview of the most important aspects of the system architecture for the proposed visual navigation system. The core design considerations have been flexibility, modularity and ease of development. However, the design of the architecture and selection of external libraries has been made with some con-

[1]Minimum Bounding Rectangle - the query asks the database to return all centroids from the table *landmarks* that lie within the given bounding box.

sideration to performance as well in order to ensure the results in this thesis give a meaningful understanding of how a final implementation of the system would behave. The chapter has concluded with an overview of how the reference data used in the following chapters has been obtained, processed and stored in a geospatial database.

Chapter 4

Feature Description

4.1 Introduction

This chapter discusses the development of a scale, rotation and translation invariant feature descriptor that can be used to uniquely identify geographic landmarks. As previously discussed in the literature review, descriptors are a non-trivial problem for a number of reasons such as uniqueness (the ability to distinguish one feature from another), robustness (support for detection errors), repeatability (consistent descriptor for a given set of inputs) and computational performance (time to compute and memory requirements). They are also designed for specific use cases - generic descriptors do not exist, making it difficult to repurpose an existing descriptor.

The biggest challenge for a landmark descriptor is that landmarks have a very low variety in their appearance and properties, making it difficult to uniquely distinguish a specific landmark. To further complicate the problem, the proposed positioning system only has approximate information about the current position and orientation and is designed to ideally be able to operate with highly inaccurate positioning estimates. For this reason the descriptor must be scale and rotation invariant and cannot rely on any absolute reference points. This eliminates many previously used approaches where individual landmarks in an image are projected on to the ground plane (for example to use the estimated global feature coordinates for matching), such as discussed in [68]. As a result of this constraint the descriptor can only use information contained within the image itself and it must be independent of the relative scale, rotation and translation of features projected onto the image frame.

Several such descriptors have been developed in the computer vision field. A popular example is the Scale Invariant Feature Transform (SIFT) [58] that is often used for matching point features from one frame to the next. SIFT is heavily used in visual odometry, optical flow and other matching applications. However, SIFT and similar descriptors are unsuitable for this problem as they rely on the visual appearance of the feature. For example, SIFT will detect a specific feature such as the corner of a box and find this corner in a subsequent image, by matching a visual descriptor of the feature. This is not a viable approach when attempting to match landmarks such as houses and junctions due to the low visual variety in the set. Buildings generally look about the same when observed from the air, which would result in a large number of incorrect matches.

SIFT and its sibling SURF (Speeded Up Robust Features)[69] have been shown to be usable for relative positioning methods such as visual odometry but they lack higher level robustness to be able to support the task discussed in this thesis.

To overcome this, a new type of descriptor has been developed during this thesis. This new descriptor ignores the visual appearance of features and instead simply assumes that landmarks are point features. Thus, instead of focusing on the visual appearance, it describes the geometric patterns that the point features form. This is founded on the assumption that the world is relatively static; that there exists a region of features in the real world and database that is identical to the region that was just observed by the vehicle. One can therefore assume that a given landmark will be surrounded by the same neighbours in both the image and the real world. As a result the descriptor describes the geometric relationship between the feature and its neighbours, allowing it to encode an entire region into a binary vector referred to as a fingerprint. A similar approach has successfully been used in several other fields to uniquely identify songs [59], or to describe and match molecules in computational chemistry [70].

This descriptor comes with a number of benefits, of which the most important is flexibility. The proposed descriptor is fully decoupled from the detection method, unlike SIFT and SURF. Since the visual appearance is no longer needed to identify a feature the descriptor becomes independent of the detection method. This enables its use with a wide variety of sensor types and detection methods, such as optical, thermal and various forms of radar.

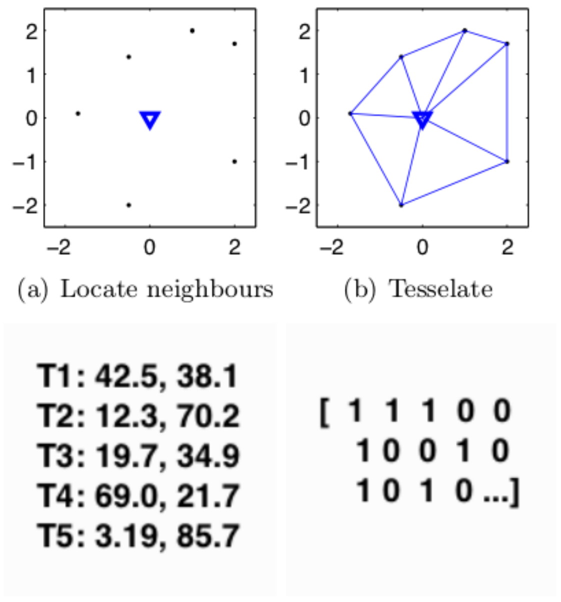
It can also relatively easily be extended to more complex environments. The current descriptor is only operating in 2D, where all features are assumed to lie in the same plane, however the approach can be extended to a fully 3D environment if such data can be delivered from the sensor. This type of description has not been explored in this thesis due to the assumption that the primary landmark sensor is a single optical sensor, which is unable to deliver 3D data in a single frame. Further, the fingerprint can be tagged with additional data to further improve performance in highly complex environments.

Finally, the process of generating a fingerprint is computationally a very light-weight process that can be implemented highly efficiently, thereby reducing the time per description to less than a millisecond.

4.2 The Fingerprinting Process

The process to generate a fingerprint for a feature begins by finding n (normally 5-10) of the feature's nearest neighbours (Figure 4.1a), based on the euclidean distance to the neighbours. When the neighbours are found they are sorted clockwise relative to the coordinate system's north (the starting point is arbitrary due to the rotation invariance of the descriptor) and tesselated as shown in Figure 4.1b. The tesselation process for this region is simple, a triangle is created between the points $P_{landmark}$, P_n and P_{n+1} where n is the index in the clockwise sorted list of neighbours.

Each triangle formed by the landmark and two of its neighbours encode the geometric relationship between the three features. This triangle can be described using two internal angles, giving a scale and rotation invariant measure of its geometric shape (Figure 4.1c). For a given landmark this results in a list of angles ($|angles| = 2 * nNeighbours$) that uniquely describe the pattern formed with its neighbours and can be used to match to a global database.



T1: 42.5, 38.1
T2: 12.3, 70.2
T3: 19.7, 34.9
T4: 69.0, 21.7
T5: 3.19, 85.7

[1 1 1 0 0
1 0 0 1 0
1 0 1 0 ...]

(a) Locate neighbours (b) Tesselate

(c) Region Encoding (d) Fingerprinting

Figure 4.1: The Fingerprinting Process

To increase the speed of matching and reduce the amount of data required to store each fingerprint, the list of angles is converted into a binary vector that is 180 bits long (Figure 4.1d), where each bit is a flag indicating the presence of an angle at that point (Equations 4.1-4.2).

$$f = zeros(180, 1) \tag{4.1}$$

$$f(angles) = 1 \tag{4.2}$$

As an example, the 69 degree angle found in triangle 4 in Figure 4.1c results in the following fingerprint (the subscript indicates the index in the vector):

$$f = [0_0, ..., 0_{67}, 0_{68}, 1_{69}, 0_{70}, 0_{71}, ..., 0_{180}] \tag{4.3}$$

The initial 180 bit vector size has been chosen since it allows every possible angle in a triangle to be encoded in the fingerprint with a resolution of 1 degree per bit. Higher or lower resolutions can be used at the cost of computational time for an increase in resolution due to the additional data, or reduced matching accuracy for lower resolution due to the decreased variety in the vector. There is no benefit from increasing the resolution of the vector unless the detector can register landmarks with very high accuracy (for a 640 by 480 image with approximately 100 features the detection error would need to be less than 1 pixel from the truth).

The result of creating a binary representation of the fingerprint is that each fingerprint and its associated data can be stored in less than 50 bytes, allowing memory efficient storage and fast retrieval of the data. The descriptor also implements a scale factor that allows fingerprints to be down-sampled to further decrease the size of the feature vectors, but it is currently not used and set to be 1.0.

It is worth noting that this final step has a significant trade-off by being a destructive one-way process as it is not possible to reconstruct a region given a binary fingerprint. Since the actual region shape is not required at any point in the proposed system this is considered an acceptable trade-off given the increase in performance.

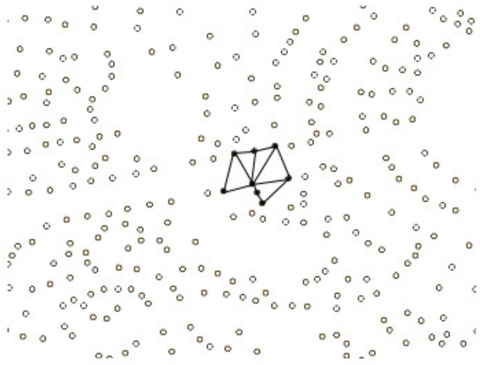
4.3 Scale, Translation and Rotation Invariance

Scale, translation and rotation (STR) invariance is one of the key properties that provides the power and versatility of the proposed fingerprinting method. By being STR invariant a feature can be observed from any altitude, position and heading (in case of an air vehicle), while the descriptor consistently computes the same descriptor for each feature. The one condition for this is that the complete region (landmark and neighbours) must be available in the captured frame, this will be discussed further in the Feature Selection section of Chapter 5.

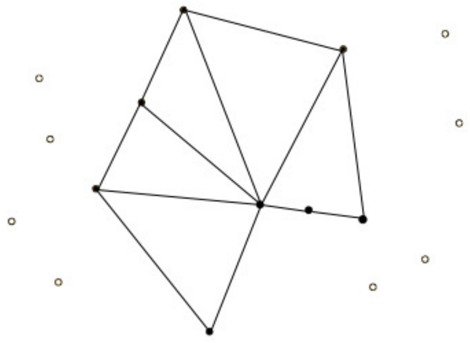
The invariance significantly reduces the computational requirements for the system compared to previous approaches and makes the proposed descriptor a viable candidate for a real positioning system.

Figure 4.2 illustrates the effects of scale and rotation on a sample fingerprint. The figure shows a specific feature in a region that is observed from various altitudes and rotation angles. In each case the descriptor produces an identical fingerprint as long as the same features can be extracted from the source image. Thus the limitation of the system is based on the limits of the feature detection method. As long as the sensor is capable of robustly registering landmarks, the descriptor will be able to compute a repeatable unique identifier for the landmark.

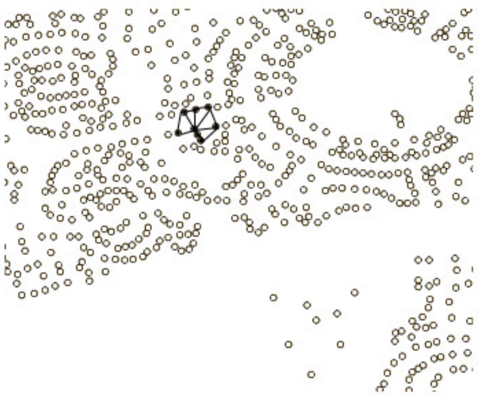
(a) Original image



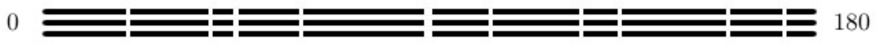
(b) Selected feature and neighbours for fingerprinting



(c) Effect of zooming in and rotating region



(d) Effect of zooming out



(e) Fingerprint vector comparison (black = 0, white = 1)

Figure 4.2: Effects of Scale and Rotation on a Fingerprint.

However, while the descriptor is scale, rotation and translation invariant it is worth keeping in mind that the invariance is in the 2D image plane, not in the global 3D frame. This means that in a 3D scenario the descriptor is invariant to vehicle heading and altitude. However, rotations in pitch and roll distort the fingerprint and require an iterative approach to resolve. Nonetheless this is a significant improvement over the approach used by Conte [2] as the number of degrees of freedom that needs to be resolved iteratively have been reduced from six (x, y, z, yaw, pitch and roll) to two (pitch and roll).

4.4 Uniqueness & Robustness

A very important factor in any descriptor is the level of uniqueness and robustness of the feature vectors. Stronger and more unique feature vectors reduce amount of work required to find a correct match. This can easily be achieved by including additional information in the vector but it often results in a trade-off where other aspects of the system have to give. For example, one way to improve the uniqueness of a feature descriptor is to include a patch of the visual appearance of the feature, such as in SIFT, but this would remove most of the benefits of the descriptor (resolution and rotation independence). Similarly, adding information about the feature type could be beneficial but it would depend on the detector being able to obtain this information accurately. Since this is difficult to assess for a relatively generic descriptor the system will only use the the angular vector but the underlying code has been developed so that it can easily be extended to improve results with specific detectors.

4.4.1 Multilayer Descriptors

A more likely way to improve uniqueness and robustness would be to add a second fingerprint angle layer. This layer could be computed slightly differently, for example by skipping certain features or by constructing other types of polygons such as quads instead of triangles. However, when doing this one has to be careful to ensure that new information is added to the descriptor, there is no benefit in simply multiplying data since it will not increase matching accuracy but will decrease performance due to the increased overhead. This has not been explored in detail since it does not seem to result in an increase in quality of the fingerprints. The current method fully encodes the shape of a region well and adding another representation of the same region does not solve any of the underlying problems caused by poor detection.

4.4.2 Variance

Theoretically there are 2^{180} (approximately $1.5 * 10^{53}$) combinations of data that can be stored in the proposed fingerprint vector, which vastly outnumbers to number of features in the world. However, this would only be achievable if the world consisted of a completely random and non-repeating arrangement of landmarks. Unfortunately that is not the case, modern societies have a preference for constructing grid-based cities for example, which increases the probability of finding angles within the 45-135 degree region in a fingerprint since neighbouring buildings are located at roughly north, south east and west of the target building. Similarly, it is very rare for a region to contain very sharp or very shallow angles (where the angle is less than about 20 degrees or greater than 160). The effect of this can be seen in Figure 4.3 where a sample of 40,000 fingerprints generated from the Ordnance Survey's MasterMap have been normalised and binned into a histogram.

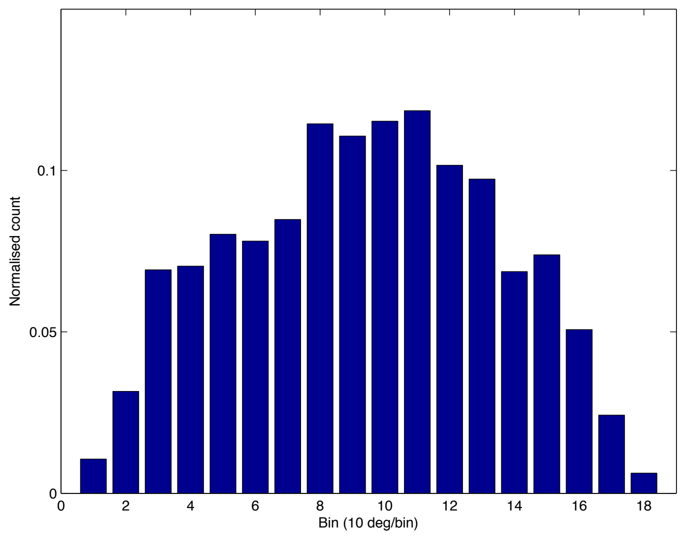


Figure 4.3: Fingerprint Uniqueness with OS MasterMap Data

In addition, while it might be possible to generate close to perfect fingerprints for landmarks using professionally created datasets, an automatic detector on a moving vehicle will inevitably cause a further loss in accuracy. As such a more relevant measure for uniqueness is how accurately one can match a given feature from a noisy source to a reference database. The first part of this, how individual fingerprint can be matched one to one will be discussed in the next section, while a more complete matching approach will be developed in Chapter 5.

4.5 Fingerprint Similarity Scoring

The first part of matching a fingerprint involves computing a similarity score that provides a numerical measure of the similarity between two fingerprints (usually between 0-1). Scorers are used extensively in the matcher to determine whether two landmarks are matching and, as discussed in the next chapter, sometimes need to be computed several thousand times per second.

4.5.1 Simplistic Methods

Because of this performance requirement it might be tempting to use a naive method, such as simply checking whether the query fingerprint is identical to the target:

$$F_q == F_t \tag{4.4}$$

This approach will yield extremely fast matching but it will fail almost instantly because a difference in a single bit will cause the comparison to fail. It also does not provide a measure of how similar two fingerprints are in case the comparison fails. As a result this method will only work in perfect scenarios where there is a one to one match for a query feature.

One way to overcome this is to use a logical exclusive OR (XOR):

$$K = F_q \oplus F_t \tag{4.5}$$

This logical operation can only be used on binary fingerprints and checks whether a bit in a given position in the query is equal to the bit in the target. If the two bits are equal (ie 0, 0 or 1, 1) the result is 0 but if the two bits are different (1, 0 or 0, 1) the result is 1. Thus the result of running an XOR operation on two fingerprints is a new vector that reveals all the differences between the two fingerprints. This is a good improvement over a simple comparison as the scoring method now can be used to determine how similar the fingerprints are by simply adding up the bits in the XOR vector. In a perfect scenario where two fingerprints are identical the XOR vector will be a zero vector, giving a sum of the XOR vector to be zero. As the differences increase between the two vectors the more bits will be turned on, resulting in an increasingly higher score.

Testing of XOR scoring shows that it is an extremely fast (since it is a very simple operation built into the CPU) but very sensitive scoring method. Even small amounts noise in the detection method will result in bit-offsets in the fingerprint, which are propagated through to the final score. Since it is very likely that feature observations are noisy, the scoring method must be more directly measuring the similarity of the feature, however the proposed XOR matcher can give the same score for two very similar and two very different fingerprint vectors.

4.5.2 Cosine-based Methods

This leads to the most common approach to comparing two feature vectors, cosine similarity[71]:

$$K = cos(\theta) = \frac{F_q \cdot F_t}{|F_q||F_t} \tag{4.6}$$

This measure computes the cosine of the angle between two vectors, where two very similar vectors will have a small angle giving a score of one, while completely dissimilar (orthogonal vectors) will have a score of zero. The cosine measure is notably different from the previous methods since it supports non-binary values for fingerprints while still being independent of the total magnitude of the vector. This is a very useful property that will be exploited later, but it has other uses in information retrieval systems as well. For example documents can be compared by creating a word-frequency vector for two documents. Thanks to the magnitude invariance two documents of varying length can be compared to determine similarities in writing styles and content, thereby providing a measure of relevance or relatedness.

Meanwhile, since cosine similarity scoring supports numeric fingerprint vectors, it would be possible to define a degree of confidence in a certain landmark observation. For example, a certain landmark gives rise to two angles (α_1 and α_2) that will be encoded in the fingerprint vector. In a binary fingerprint the bits at positions α_1 and α_2 would be switched on, however, since the detection method is noisy, it is possible that this measurement is incorrect. Due to the resolution of the fingerprint vector (1 deg/bit) it is very likely that even a few pixels noise in the detector will cause a mismatch with the database vector.

4.5.3 Error Models

One way to overcome this is to incorporate a probability distribution that describes the estimated error and encode this distribution in the fingerprint. Thus, instead of simply switching on the bit at location α, the distribution is encoded in the elements surrounding α:

$$F = [0_0, ..., 0.05_{67}, 0.2_{68}, 0.5_{69}, 0.2_{70}, 0.05_{71}, ..., 0_{180}] \tag{4.7}$$

This results in two difficulties: how are overlapping error distributions dealt with and what noise model should be used to represent the performance of the detector?

Starting with the overlapping distributions, what should be done if the features F_1 and F_2 intersect in the fingerprint? Since the two features are independent the two distributions cannot simply be added together. A way to to avoid this would be give up the fingerprint vector model and instead store angles and their respective estimated error models, but this would make matching significantly more difficult and slower than using a single feature vector. Another way would be to create independent feature vectors for each angle but this would force a discretisation and loss of accuracy of the error model and still result in an increased computational expense.

As for the error models themselves, it would be possible to assume a simple hat, linear or Gaussian model but this error will be dependent on the internals of the detection algorithm as well as any transformations to the image (such as perspective and scale due to altitude and orientation, terrain effects and more). This will either require additional inputs and a bespoke error model for the sensor and detector or a very simplistic error model.

A flat error model has been chosen for this work for two reasons. First, a simple model makes the descriptor generic and means that minimal work is required to adapt it to a variety of sensors and detection methods. Second, calculating cosine scores with numerical values is a very costly computation since it involves a large amount of floating point operations. Further, a numeric fingerprint with 4 byte float values for each element occupies 720 bytes of memory (excluding additional metadata about position, ID and neighbours), nearly 30 times more memory than a binary fingerprint.

4.5.4 Efficient Scoring

However, it is possible to exploit the magnitude invariance of the cosine scorer to avoid the additional computational overhead and memory usage. To do this, we first use a simple flat "hat" error distribution with an equal probability for each possible angle element surrounding the measured angle. This is the equivalent of saying that the detector has an equal probability of registering a landmark at any point within a given region around a real landmark. The detection error is purely random:

$$error_{detection} = landmark_{truth} + rand(-maxError : maxError) \tag{4.8}$$

Secondly, we simplify further by assuming that each detected feature in the image has the same error distribution. This means that the detector is unaffected by lens distortion, terrain effects and other local visual distortions. This simplification is required to avoid dependencies on additional information such as position, orientation, terrain type, elevation maps and more, which would otherwise be required to account for these effects appropriately.

The process for incorporating a landmark observation into a fingerprint then becomes very straight forward. If a feature is detected at a position α with an estimated error of $\pm 2deg$ and an equal probability of being anywhere within this region, then the resulting data in the fingerprint vector becomes:

$$[0.0_0, ..., 0.2_{\alpha-2}, 0.2_{\alpha-1}, 0.2_{\alpha}, 0.2_{\alpha+1}, 0.2_{\alpha+2}..., 0.0_{180}] \tag{4.9}$$

Finally, the cosine similarity measure is magnitude invariant, which means that the numerical fingerprint vector is equivalent to a binary fingerprint (since the numeric fingerprint simply is a scaled version of the binary fingerprint):

$$cosineSimilarity(n * F_q, F_t) = cosineSimilarity(F_q, F_t), n > 0 \tag{4.10}$$

thus:

$$\begin{aligned} &[0.0_0, ..., 0.2_{\alpha-2}, 0.2_{\alpha-1}, 0.2_{\alpha}, 0.2_{\alpha+1}, 0.2_{\alpha+2}..., 0.0_{180}] \\ &= 0.2 * [0_0, ..., 1_{\alpha-2}, 1_{\alpha-1}, 1_{\alpha}, 1_{\alpha+1}, 1_{\alpha+2}..., 0_{180}] \\ &\equiv [0_0, ..., 1_{\alpha-2}, 1_{\alpha-1}, 1_{\alpha}, 1_{\alpha+1}, 1_{\alpha+2}..., 0_{180}] \end{aligned} \tag{4.11}$$

This reverts the memory requirements back to the expected binary memory usage but the computational issue remains (although to a lesser degree). To overcome this, the original cosine similarity scorer can be replaced by the Ochiai measure[72]. The Ochiai is a measure that has primarily been used in biology to study similarities between animal species in different geographical regions, however it is identical to the cosine similarity measure when used with binary feature vectors. The Ochiai measure is defined as:

$$K(F_q, F_t) = \frac{n(F_q \cap F_t)}{\sqrt{n(F_q) * n(F_t)}} \tag{4.12}$$

where n(F) is the number of enabled bits in the bit-vector F. This is a dramatically faster computation since the binary set intersection (AND) is much quicker to compute than the dot product that is required in the cosine similarity measure.

4.5.5 Jaccard Index

The Jaccard index is an alternative binary similarity measure that was first used by Paul Jaccard. It is defined as:

$$T_s(U,V) = \frac{\sum_i (U_i \wedge V_i)}{\sum_i (U_i \vee V_i)} \tag{4.13}$$

This index is very similar to the Ochiai measure but it falls off linearly with respect to similarity, while the Ochiai measure follows a cosine. As a result the Jaccard index penalises small differences between vectors harder, which enforces a stricter similarity requirement between vectors than the Ochiai index. This is preferable in the proposed system since the variety between vectors in the database of landmarks is relatively low (a challenge that will be discussed in chapter 5), meaning that two different landmarks can have very similar fingerprint vectors. By allowing the score to fall of quicker only the most similar vectors will remain in the matching set, making it less likely that an incorrect feature is matched. Conversely, this has the effect that correct features will be rejected due to unexpected detection errors. However, this is ultimately preferable for the system since a small number of high confidence matches are more reliable and will give a better final positioning estimate than a large number of potentially incorrect matches.

4.6 Results

4.6.1 Synthetic Data Generation

The following results have been generated by randomly distributing 500 features in a simulated world. This data does not reflect a real-world scenario (such as a city layout), it is only used to evaluate the performance of the different scoring methods. A simulated aerial capture is generated by selecting a subregion, then gradually distorting it to evaluate which scorer results in the highest number of accurate matches for a given distortion level.

The distortion has been designed to simulate realistic errors introduced by the detection stage and consists of three types:

1. Shift - a bit in the vector is shifted either left or right to simulate a noisy detection
2. Insertion - a new angle is introduced in the vector to simulate a false detection
3. Deletion - an existing angle is removed from the vector to simulate a failed detection

These are randomly applied to the features, each with equal probability, with a final check to ensure that a correct number of angles are present in the fingerprint. If not (e.g. if the deletion distortion has been applied multiple times), random angles are added or removed to the vector.

The number of distortions is gradually increased while the performance is measured by selecting 100 features at random and scoring them against the true set. A correct match is recorded if the distorted fingerprint scores the highest when matched against its truth.

For each match a "true" vector quality score is also calculated which is determined by calculating the cosine similarity score between the original and the distorted fingerprints. This is used to bin fingerprints by their level of distortion, since the quality of a distorted vector depends strongly on which type of distortion was applied. Thus, even if the same number of distortions were applied to a fingerprint the resulting quality can vary significantly, which is accounted for by binning by quality.

4.6.2 Accuracy

Results from this test are shown in Figure 4.4. For reference, since the vector quality is based on the cosine of the angle, a score of 0.707 means that there is a 45 degree angle between the query and target fingerprint.

As expected the equality condition and XOR scorer does not fare well, with the former failing as soon as any distortion is introduced. The XOR scorer performance drops off very quickly as well and is not useful in any form of realistic scenario. This leaves the cosine (Ochiai) and Jaccard index, both of which are much more robust to distortions. These maintain their matching accuracy up to high distortion

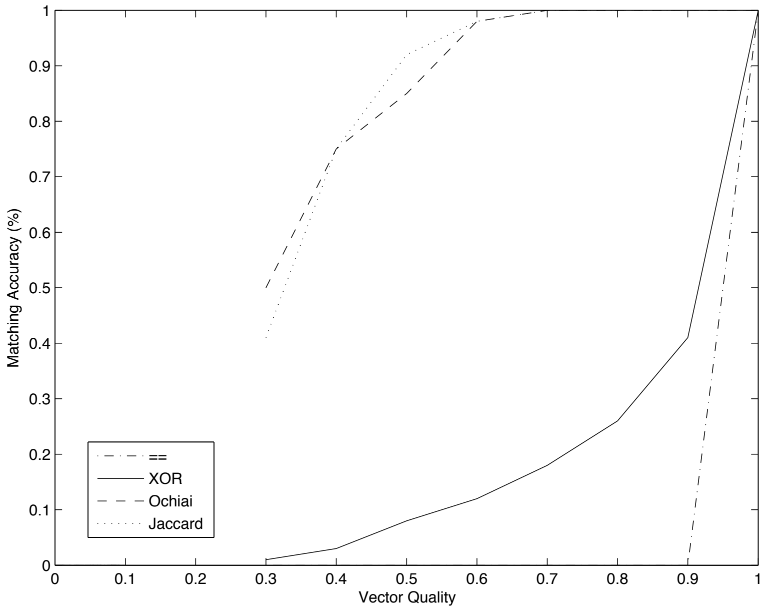


Figure 4.4: Fingerprint Scorer Performance (Vector Quality 0.0 = fully destroyed, 1.0 = perfect and undistorted)

levels (vector qualities of 0.4-0.6) but then fail rapidly as fingerprints are further distorted.

Since the detector is expected to be noisy it is also possible to estimate the resulting quality level of a certain detector. For a detector that detects features within ±2 pixels of the truth, with an average distance from landmark to neighbours of 50 pixels, the resulting angular error in the vector is approximately 2 degrees. Given this error the estimated vector quality is 0.5.

As such, the scorers must be able to perform at this level. Both the Jaccard index and the Ochiai similarity scorer perform very well at this range with greater than 97% accuracy, with the Jaccard scorer marginally outperforming the Ochiai.

These results come with a significant caveat though since they have been carried out on a relatively small sample set where there are few ambiguous features. The number of incorrect matches will increase as the feature set increases due to the relatively low variety in the feature vectors. As such, a simple one to one scorer is useful for determining the similarity between a small number of candidates but it is unusable for large scale matching using tens or hundreds of thousands of features. This

Table 4.1: Scoring Methods Performance

Scorer	OPS
==	970,000
XOR	8,200
Cosine (numeric)	350
Ochiai (binary)	3,000
Jaccard (binary)	2,900

will require a higher level matcher that can reduce the target set to a manageable number and exploit additional information that is available in the data set. This will be discussed in more detail in Chapter 5.

4.6.3 Performance

In addition, each method was evaluated for performance. To do this, two fingerprints were generated then each scorer was repeatedly calculating the score for these for 60 seconds while the number of executions was recorded. This gives the number of Operations Per Second (OPS), which can be compared to give the relative speed of each method. The true performance is highly dependent on the implementation of the scorer, the programming language and the speed of the processor that is executing the code. In this case the test was carried out using 32bit Python 2.7.4 on Mac OS X 10.9.1, running on a 3.5 GHz Intel i7 processor. The results also include examples for a numeric cosine scorer to illustrate the computational cost of dealing with numerical vectors instead of binary.

While the equal and XOR scorer are much faster than the other scoring methods, results in Figure 4.4 has shown that these two methods are nearly useless. More interestingly, the performance results show that the binary scorers are nearly ten times faster than numeric scorer. Since these have been tested in an interpreted environment (Python) rather than as a compiled application it is reasonable to expect an additional overall 50-100x speed up in a compiled implementation. However, for prototyping purposes the current implementation is fast enough.

4.7 Conclusion

This chapter has discussed the challenges of feature description (robustness, uniqueness, repeatability and computational performance) and new type of feature descriptor for geographical landmarks has been developed. The new descriptor uses the pattern constructed by a landmark and its surrounding neighbours and encodes it in a scale, rotation and translation invariant way. By adding a simple error model and using a binary scorer it is possible to obtain better than 97% matching accuracy for small sets in the target operating conditions, with the best scorers taking less than 0.33 milliseconds to compute a score.

However, these results also demonstrate the need for a higher level matcher that can reduce the target set to avoid mismatches. There are several benefits to using a higher level matcher that is independent of the scoring method and this matching strategy will be discussed in the next chapter.

Chapter 5

Feature Matching

Chapter 4 discussed the development of a feature descriptor for landmarks that is scale, rotation and translation invariant. It also described a number of one-to-one scoring methods that computes how similar two fingerprints are and evaluated their performance under varying distortion levels. While the best matchers performed respectably well under distortion the results showed that a bigger problem is finding a match for a single fingerprint in a large set (10,000+ features).

There are two main reasons for why the descriptor fails in large sets:

1. Region similarity - man-made landmarks are semi-structured. While there is variety in the placement of individual landmarks, they tend to be placed in regular grid-like layouts, such as cities and along roads. This causes a non-uniform distribution of features in the fingerprint and makes it much more likely that two geographically distant regions have a very similar geometric appearance.

2. Data reduction - the proposed feature descriptor discards a substantial amount of data about landmarks (such as appearance and feature class, which is discarded since it will not aid the matching process) and then further reduces the amount of data available by irreversibly compressing the region into a binary fingerprint. This gives a very memory efficient descriptor that is still capable of uniquely describing the landmark.

However, these two combine to make the descriptor very sensitive. Even small errors such as detection noise can have a big impact on the fingerprint vector and can cause wildly incorrect matches. Since the goal of the project is to match a small number of landmarks detected in an aerial image to a vast database of landmarks with high accuracy (or at the very least low rates of false positives), it is clear that a more sophisticated matching approach is needed. This chapter describes a solution to this matching problem that improves matching accuracy under normal operating conditions. Following this will be a discussion of alternative matching methods as well as a number of optimisations that can be done to further speed up or otherwise improve the matching accuracy and performance.

5.1 Contextual Information

In the previous chapter, a single feature was matched to a large set of features. This provided adequate results but was found to not be accurate enough to be used in a real system since there are too many ambiguous matches. Thus, to improve the matching accuracy the system needs a way to reduce the number of target features in the database and thereby reduce the risk of a failed match. An obvious example of this would be to use the current best estimated of the vehicle's position, speed, altitude and field of view to predict which part of the database is likely to be observed next. Another would be to compute a metric that is specific to the feature (such as roundness of a crater), that can be used to further narrow down the target set. However, there are other, more powerful, methods that can further improve the accuracy and speed of matching, which do not directly use the descriptor of the fingerprint itself but instead rely on other contextual information about the scene.

One effective approach is to analyse of the arrangement of features. Due to the nature of the descriptor, it is impossible to construct a fingerprint for a single landmark since its surrounding neighbours are required to compute the descriptor. This means that each aerial image capture must contain multiple features, which is additional information that can be used to improve the matching quality significantly.

A quick calculation shows that it is realistic to assume that an image taken over a densely populated area can contain approximately a hundred landmarks [1]. Matching each of these individually to an entire database will yield a very low success rate, but there are several ways to compute metrics about the query set that can help increase the matching performance. A simple example is elimination of poorly conditioned features and ranking of features by their quality (and thus probability to match correctly).

These methods rely on a simple assumption, that the landmarks are static and permanent. This means that, assuming the detector is performing correctly, the UAV will observe a number of local landmarks for which there are corresponding matches in the database that are arranged in a similar way relative to each other (with the exception of scale and rotation). An alternative way to express this assumption is that each query landmark will be surrounded by the same neighbours in both the aerial capture and the database. A certain amount of flexibility must be permitted to allow some changes in the environment, such as newly constructed buildings that are not available in the database, or failures in the feature detector but these must not make up more than a small percentage of the total number of features.

5.1.1 Feature Selectors

During the development of the geometric descriptor it became apparent that certain landmarks result in higher quality descriptors than others. More importantly, the reverse was also found to be an issue, the descriptors for certain landmarks can be too badly conditioned to be of any value during the matching process and can even result in a degradation of the overall result. This degradation can occur when poorly conditioned features are used to identify a target region using RANSAC since the features can match to a large number of incorrect landmarks, thereby throwing off the algorithm. In a worst case scenario a large number of poorly conditioned features are used in the same update in RANSAC, which can result in a completely random target region being selected. At this point even high quality features that normally should match if the correct region is identified will fail.

[1] Assuming a landmark size of ten by ten meters, landmark density of 70%, captured from 500 meters altitude with a regular 20 x 30 degree field of view lens

However, the weak landmarks can also cause problems even if the correct region is found, by causing incorrect matches within the local region. Since the correspondences between local and global features are used to compute the position of the vehicle it would be preferable to have a high quality rather than high quantity of matches. Weak features can therefore mistakenly be used to compute the position of the vehicle, which can result in a number of effects depending on the algorithm chosen.

In an ideal scenario the false match will be rejected using an outlier elimination method while the correct matches are kept, in which case there is no effect on positioning. However, this approach is risky as outlier elimination methods require that at least a majority of correspondences are correct (i.e. fit the desired model). In many scenarios this is true but as we will soon see it is not unlikely for a visual positioning system to obtain inputs that can be very badly conditioned. In some cases it is possible to obtain a poorly conditioned capture where a large majority of the features are unsuitable for matching, at which point it is important to either prevent these features from being matched or alternatively reject the entire image and capture a new input.

If the outlier rejection fails or is found to be unsuitable then the incorrect matches will give rise to other problems. The effect depends on the pose estimation algorithm but will likely lead to either gradual drift or a complete failure. Both of these should be corrected once well conditioned data is captured, but if the matching errors are not caught it can result in issues in other systems that make use of the positioning data.

In short, the pose estimation performs better and becomes more robust with a small number of high quality features rather than a large number of low quality features. Because of this, the system strongly benefits from having a method to reject poor features.

In the context of the geometric feature descriptor poor features can be defined as features that either have an incomplete descriptor (because insufficient neighbours are identified) or a descriptor that is too similar to others. There are three main cases where this can occur:

1. Single features

   In cases where only a small number of features are found in an image (less than the number of neighbours required for fingerprinting) the fingerprints are too weak to match reliably. For example, this can occur when the system relies on buildings but is currently operating over a sparsely populated area. When this is the case the system should temporarily disable itself until the feature count increases above a required threshold.

2. Line features

   In cases where a set of features form a line, such as buildings placed along a single road, the region encoded in the fingerprint will be too similar as the neighbours used are placed directly to the left and right of the query feature. This leads to nearly identical fingerprints for a number of features, causing matching to fail.

3. Edge features

   Features placed near the edge of the image will fail to match as their fingerprints rely on neighbours that fall outside the image. The fingerprints for these features will be incomplete and will therefore not be strong enough to match.

Two methods have been found to effectively identify and eliminate these types of features. Both methods produce a normalised score between zero and one, which lets the matcher select a small number of the highest ranking features for matching and eliminates any potentially difficult, low scoring, candidates.

Central Features

Starting with simplest failures, failure case 3 - edge features, the problem is that features located near the edge of the image likely depend on features that lie outside the captured region. Therefore the fingerprinting process for these features will produce incomplete fingerprints and the subsequent analysis for these features will fail.

A simple way to overcome this is to add a selection criteria where features closer to the centre of the image are favoured. This is implemented as a distance measure where the distance from the centre of the image is calculated for each feature, if the feature falls outside a limiting distance it is not considered for the matching. Further, features closer to the centre are weighted higher than outlying features and are therefore given priority in the matching process.

This approach is currently favoured over the use of a hard threshold distance where all features that lie within the limiting distance are equally scored. While dense scenes may contain features that are excellent for matching but do not lie near the centre of the image, sparse scenes are suffering from the opposite effect. By introducing a linear score based on the normalised distance, poorly conditioned and outlying features are automatically removed from the matching process.

This might adversely affect densely populated scenes by favouring more central features over better conditions but outlying features. However, there are additional selectors that analyse other metrics and the combined score can be used to identify a sufficient number of strong candidates even given this weakness.

The final output of this method is a list of all features that fall within the safe region, sorted by their proximity score.

Connectivity Analysis

The second selector method is more complex as it analyses the connectivity between features. Connectivity analysis deals with the other two types of difficult features: line features and single features. These features are easily detected due to their poor connectivity with their neighbours, for example features on a road are only connected to its two nearest neighbours and single features have a very low number (often zero) of neighbours.

Additionally, since the descriptor is based on the surrounding neighbours there is a direct relationship between the connectivity of a feature and the strength of its descriptor. Connectivity analysis therefore provides a good measure of how well connected and thus how likely a specific feature is to be described and matched successfully. Similarly to the central feature descriptor, this is used to rank a query set and limit the matching task to the n highest scoring features.

The proposed selector is based on the Gabriel graph, a graph that connects first degree nearest neighbours. A first degree connection is defined as a connection that can not be made using shorter segments via other features. To determine whether a connection between two points, A and B, is a valid first degree connection, one constructs a circle with its centre at the midpoint between A and B with a diameter of the distance between the two points (Figure 5.1). If no other points fall within this circle the connection is valid, otherwise the graph is refined by adding the intermediate point.

The result of applying this to a real geographical region is shown in Figure 5.2. A Gabriel graph makes it very easy to identify line features as they only have two connections to neighbouring features, the two immediate neighbours. Therefore by eliminating features with a low connection count in a Gabriel graph one can avoid line features and other weak regions where the descriptors are likely to be weak.

The Gabriel graph has one final benefit, it not only identifies weak features it also locates the strongest features in the set, the features with the most number of strong first degree connections. This allows the matcher to start the process with the strongest features that are the most likely to match and gradually refine the results using weaker features.

Figure 5.2 shows the result of constructing a Gabriel graph for a small region. Green features have been accepted while red features have been rejected. In addition to this the connections between features have been coloured to indicate whether that connection will be encoded in the fingerprint, green indicates encoded connections while black indicates lost data. An interesting point to note is that even though a large number of features have been eliminated from the direct matching process most of the data they represent, the connections they are based on, is still encoded in the fingerprints. Thus, very little information is lost by eliminating troublesome features but the matching accuracy improves significantly.

While this approach provides excellent selection results it can be a computationally intensive method since it first requires the computation of the Delaunay triangulation (an O(n log n) function) for the set, which is then further processed with an O(n) function to obtain the Gabriel graph. This means the computational time requirement primarily logarithmically as the number of features increase, which could potentially cause problems if a very large number of features are suddenly detected.

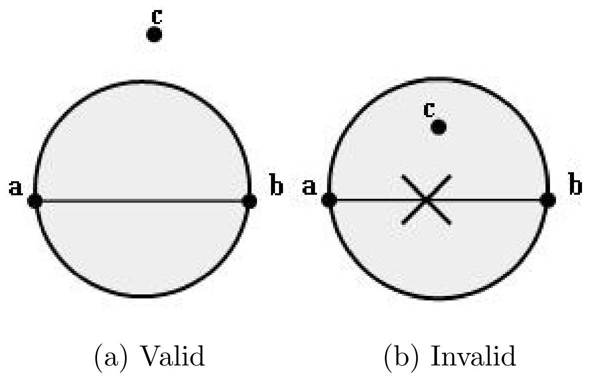


Figure 5.1: Gabriel Graph Connectivity Rules

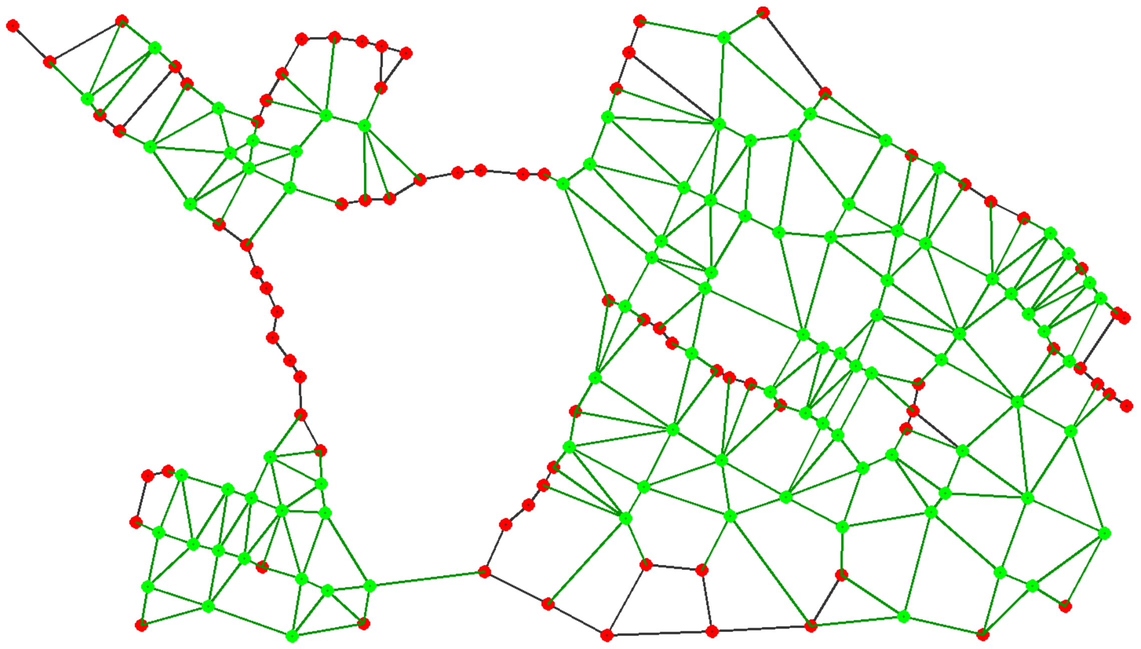


Figure 5.2: Gabriel Graph for East Cranfield

A simple solution to this is to restrict the Gabriel graph to only work on the set of features selected by the Central Selector, which makes the Gabriel graph work as an additional refinement of the selector set rather than an independent selector.

5.1.2 Neighbours

As previously discussed, matching a single feature is difficult as there can be a number of ambiguous matches in the region, even if the feature itself is very well conditioned and has a robust set of neighbouring landmarks. The most effective way to reduce this number is to include additional information that further distinguishes the correct feature from the incorrect matches. Fortunately there is a very cheap and easy source for this information: the feature's neighbours.

The neighbours surrounding the query feature are already known since they are required to compute the fingerprint. Thus, the information contained in these neighbouring features can easily be added to the current fingerprint to effectively encode a larger region for the feature. This means that the fingerprint will now encode not only first degree neighbours but also second degree, with the result that third degree features will be affecting the descriptor (the the neighbouring features encode their neighbours in their fingerprint). This increases the size of the region drastically and, depending on the number of shared neighbours, up to a quadratic increase in features.

However, simply concatenating the fingerprint vectors and forming one large feature vector is for many reasons not a good idea. The first issue is that the process would be order dependent. Changing the order (such as concatenating the fingerprints clockwise instead of counterclockwise or changing the starting point) will result in a completely different fingerprint. It would also be easy to accidentally create an ordering process that is dependent on additional variables. For example, the starting point for a clockwise ordering process adds a heading requirement, which we would like to avoid in the system.

In addition, if the fingerprint is order dependent then a failure to visually detect one of the neighbours can cause a partially incorrect fingerprint where first half might match but the second half does not since the remainder of the vector has been shifted one feature step. It may be possible to resolve this , but it causes further ambiguities and adds complexity to the system.

Finally, concatenating the fingerprints causes a large duplication of data as a feature's fingerprint is repeated in all of its neighbours. This increases the computational cost at all points in the system and makes it harder to update the database in case a new feature needs to be added.

A more efficient solution is to add an ID number to each feature and during construction of the reference database store the ID numbers of the neighbours associated with the current feature. This allows the matcher to retrieve the target feature and, if required, come back and fetch the neighbouring features for validation, as these fingerprints should match the neighbours observed in the aerial capture. It also reduces the cost of modifying the database as only the features directly affected by the change need to be updated.

However, this requires an assignment method to work since the order of the neighbours is unknown. Thus, each query neighbour feature is matched to each potential target neighbour to determine their match. This gives a cost matrix that can be evaluated using the Hungarian algorithm to determine the optimal assignment that maximises the overall score, and thus the (likely) correct matches for each neighbour.

Incorporating this information significantly improves the system's tolerance to noise and poor detections. In effect it shifts the descriptor matching results in Section 4.6.2 - Figure 4.4 left, meaning the required fingerprint quality is reduced.

Given the same parameters, when only the query fingerprint is degraded and the neighbours are unchanged the required matching quality drops to 0.3. In a more realistic degradation scenario where both the query feature and the neighbours are degraded the matching quality settles at 0.39 as the neighbours can not be relied on to the same degree.

5.2 Matching Strategies

Ultimately, carrying out the matching process as a one to one task on a global scale is futile. The world is simply too ambiguous to allow correct matching on a large scale, even given very well conditioned features and neighbourhood matching. However, while the end result requires a one to one match for each feature, the matching processes does not necessarily have to work this way. This section discusses various approaches to the matching problem, which help obtain high quality results.

The following methods can be seen more as filters and refinements rather than full matching as they process and validate an initial set of potential matches for a query feature. This set is obtained by carrying out a one to one match for each feature in a certain region but rather than saving only the best match, the n best matches are kept. The assumption is that the operating conditions are such that the quality of the feature fingerprints exceed the threshold to obtain a successful direct match in at least 50% of the cases. This means that the correct match may not be the best scoring match in the candidate set but it is likely to be a member of the set, thus the problem to a degree becomes an assignment problem where the best solution is the one that assigns each query feature to a target feature so that the overall fit between the query and target regions is maximised. This is a difficult computation as each possible combination needs to be evaluated to determine the best assignment.

5.2.1 Constraints

The most effective way to filter the results is to match the entire set of selected features at once and apply a number of constraints before the best matches is selected. For example, while the likelihood of a single feature having several ambiguous matches is quite high, the probability of several features having incorrect matches in the same region is much lower.

Internal Distance

A simple constraint method is to minimise the internal distance of the target set. This is the simplest interpretation of the example above, where it is assumed that the correct assignment of features is the one where all features lie near each other.

This metric is very easily evaluated for an assignment by calculating the distance matrix from the selected features. The score of the assignment is then obtained by calculating the the total sum of the elements in the distance matrix. Theoretically, the correct assignment is the one that minimises this sum.

Since the distance matrix is symmetric only half of the elements in the matrix needs to be computed, making this a quite cheap method. However, it is a quadratic function ($O(n^2)$) of the number of candidate features, so if the system is configured to use a larger number of candidates the workload increases significantly.

The downside of this constraint is that it does not actually consider whether the assignments are correct and fit any sort of model - it simply attempts to find the assignments that place the features as close together as possible. If there are false matches in the same region it is possible to end up with an incorrect assignment, which, as previously discussed, can cause issues in the state estimator and should be avoided.

An alternative to this method is to use a clustering method such as Qualitative Threshold (QT) clustering[74]. QT clustering is a method that identifies clusters of specific sizes in a dataset, but it can also be used as a method to determine whether a set of points form a cluster of a given size. Thus it is possible to set a size limit for the cluster based on the estimated size of the query set[2] and score the quality of the match depending on the number of clusters found in the target set. If more than one cluster is returned then the assignment involves a target region that is larger than the query set.

However, this is a risky assumption due to false matches. It is likely that certain features in the query set will not match to the query set for various reasons. These will be randomly placed in the search region and will therefore form separate clusters. Because of this, it is not only the number of clusters that is relevant, it is also dependent on the number of features in each cluster. A simple way to overcome this issue is to instead rely on the number of features in the largest cluster and computing the final score as the percentage of features that fall in this cluster.

Unfortunately, QT clustering is currently a prohibitively expensive method. The initial part of the clustering process is similar to the previously discussed distance minimisation method, since both approaches compute a distance matrix. However, the QT clustering algorithm then creates each cluster by iteratively attempting every cluster combination to determine the largest one. If additional clusters are needed the process is repeated until every feature has been assigned to a cluster. This effectively increases the computational cost of evaluating the QT cluster score compared to internal distance from $O(n^2)$ to $O(n^3)$, which will greatly increase the time it takes to evaluate an assignment.

[2]This is possible if an approximate altitude is known, see Section 5.2.2 for a discussion on the use of positioning data.

Region Similarity

The region similarity approach is a different type of constraint which tries to ensure that the assignments match a slightly more complex model, thus making it more likely to obtain the correct assignments (unlike the internal distance constraint). It makes use of the previously discussed static world assumption to identify whether the region constructed from the proposed assignment matches the region constructed from the query features.

An obvious way to do this would be to compute a fingerprint for the query features and then attempt to replicate the fingerprint using each assignment combination of the candidate features.

This approach is difficult to implement in reality as the fingerprinting algorithm is based on a specific feature. Thus this approach requires an anchor point, a feature which is central in the shape, around which the remaining features are treated as neighbours. However, selecting an anchor point is not easy as that feature must be guaranteed to be correct in order for the fingerprints to match when all the correct assignments are made. If the anchor point is matched incorrectly the fingerprint for the target region will be completely different from the query region. Since it is not possible to make this guarantee an alternative approach is needed.

An successful approach has been to construct simple shapes from a random selection of features. If a polygon is constructed from a number of features in the query set then the correct assignment in the target set should produce the same polygon. Once a set of assignments has been identified a new polygon is constructed with a random set of features and the process is repeated. This allows the system to keep track of features that continuously become selected as matches in the polygons, the candidate features which are most likely to be the correct assignments.

There are a number of benefits to this approach, first the polygon matching (based on the internal angles similar to the geometric descriptor) does not need to be very precise as it is unlikely that an incorrect arrangement of features will produce a matching shape, in particular considering that the candidate matches are often spatially very distant. Since the only error, theoretically, on the polygon shapes will be due to detection errors, the matching conditions can also be fairly tight to ensure only a very similar arrangement is accepted.

Secondly, the polygon matching approach is much more efficient than the internal distance measure. Instead of attempting every single assignment combination among all the target features only the features that are known to be match candidates for the vertices in the polygon are used. This could also in the future allow the system to lock down features it is particularly confident about, and continue constructing polygons where some of the vertices in the polygon are known with good confidence while others still need to be resolved.

The implementation of this approach is very similar to the fingerprint matching. For simplicity and speed the current implementation uses triangles, so three vertices are picked from the query set (that has been filtered by the feature selector). A triangle is constructed from these features and the internal angles are computed to avoid scale and rotation dependencies. However, instead of producing a fingerprint vector the three angles are kept as numerical values and sorted in ascending order.

A candidate triangle can then be evaluated against the query shape by checking whether each angle falls within a certain threshold of the angle in the query. The threshold is primarily a function of the expected error from the detector, similar to the thresholds used in the geometric descriptor.

5.2.2 Positioning Data

The goal of this project has been to avoid using as much external information as possible in order to develop a system that is independent from other positioning methods. This has led to the development of a new type of descriptor and a large amount of work on improving the matching performance for very large queries. These techniques have shown promising results but the decision to not make use of information that would be readily available in a flying vehicle would unnecessarily rule out some of the most effective methods for improving matching accuracy.

The most important additional piece of information is the vehicle's approximate location and altitude. It is reasonable to assume that the vehicle will have at least a rough estimate of its current location in order for critical systems such as the autopilot to function correctly. Ignoring that this information is available only serves to make the proposed visual positioning system harder to design, unnecessarily complex and less robust without providing any tangible benefits.

Depending on this information does mean that certain positioning scenarios are no longer possible, the main issue being a fully lost scenario where the vehicle has no knowledge of its position. However, an aerial or space vehicle will not begin a mission without this information and reasonably accurate estimates of the position are required to maintain safe flight. As such, it is fair to assume an operating scenario where the starting position and orientation is provided along with a secondary positioning system such as GPS and/or a dynamic model that allows the vehicle to predict its position with acceptable accuracy until it has reached sufficient altitude to enable the vision-based positioning system.

Similarly, it is likely that a vehicle will operate using a more power efficient system such as GPS until a failure or interference is detected, at which point the VPS can be initialised with the last known or the current predicted position and take over positioning duties until the primary system becomes available again.

Region Selection

The most important use of positioning data is to restrict the matching process to a smaller region of the database. As a result the matching process is carried out against a much smaller set which will significantly increase the likelihood that correct matches are identified. It will also provide a very drastic performance improvement as the initial candidate search is directly dependent on the number of features in the target set.

Ensuring that the correct geographic region is selected from the database is not quite as simple as selecting a region around the last known position since the correct region is not only dependent on the position of the vehicle, it is also affected by the altitude, velocity and heading of the vehicle. In addition, it is useful to know the estimated error for each of these parameters as this allows a very targeted search in the database.

As long as control inputs and a vehicle model is known it is also possible to gradually extend the search region as a function of the estimated errors in case the VPS fails to obtain a position estimate (such as in the case of operation in an area low in landmarks).

The effect of this on the performance of the visual positioning system is significant. If the vehicle's position is known to GPS accuracy it is possible to extract a region that matches the exact footprint of the sensor. This lets the matching process be simplified to the point where it effectively confirms the matches found by the sensor. This is done by computing the fingerprints for features in the field of view of the sensor, passing them through the feature selector and confirming that the selected features are present in the search region. In the future this can also be extended so that position data is fed back to the detector in order to provide an initial guess of where landmarks may appear. This feedback loop should provide improved performance for the landmark detector and improve the overall speed of the system.

However, this is only the case when good positioning data and a well conditioned view of the landmarks is available. If there are problems with either of these the system must fall back on a larger regional search and use the previously discussed methods to confirm the matches.

5.3 The Matching Process

This chapter has discussed a number of approaches to identify suitable features, correctly match these to a database and to verify this assignment. Figure 5.3 shows how these fit together to form the complete matching process. Each step has one or more inputs that can be tailored to the specific task at hand, but overall the system has been designed to follow a predictable and repeatable process.

As previously discussed, the process begins by extracting and processing the local query features. A new set of data is captured from the sensor, which is processed to extract specific landmarks, such as buildings or craters. These point features are processed to generate a feature vector, or fingerprint, for each landmark and then passed through a feature selector that identifies the features that are most likely (or not) to match.

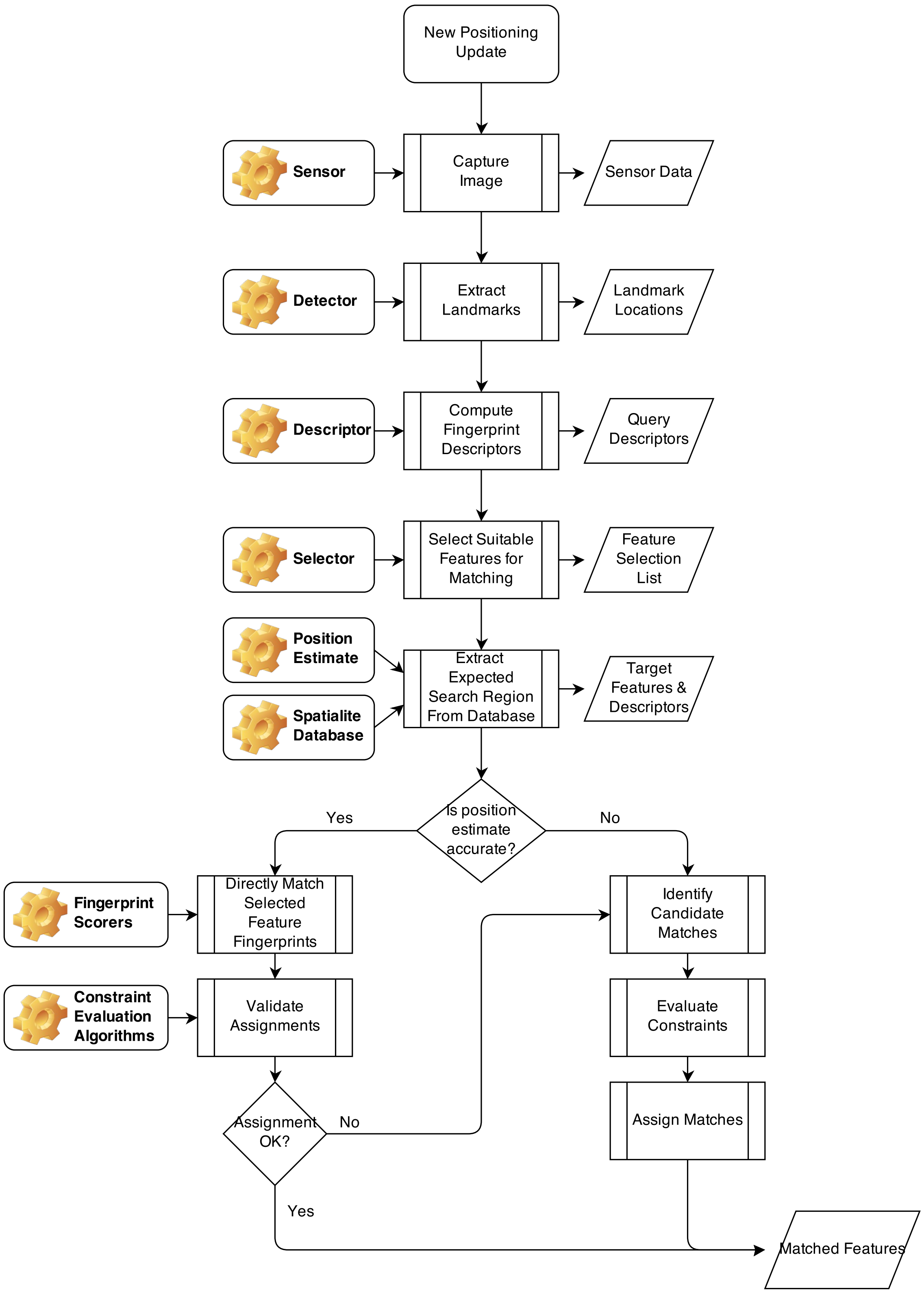


Figure 5.3: Matching Flowchart

Once the local features are ready for matching the system begins the matching process. The first step is to obtain the latest position estimate and retrieve the corresponding region from the database. The size of this region is dependent on the estimated accuracy of the position data.

At this stage the system reaches the first of two decision points. If the position estimate is good enough, most easily evaluated by checking the number of target features that have been retrieved from the database, it starts an optimised and simplified matching process. In this case the selected query features are directly matched to the target set and the best match is simply considered to be the best scoring match. Once each match has been identified it is validated using the region similarity method (the internal distance method is of no use in this case since the correct region already has been identified, it would effectively attempt to minimise the size of the same cluster over and over).

This is the second decision point, if the assignments pass validation then the correct matches are returned to the controller, otherwise it the system falls back on a full match. This is the same process that is carried out in case the position estimate is too poor, resulting in a larger search region.

In this case the system obtains the n best matches for a feature and uses them as candidates. Each combination of these candidates will be validated using a constraint method that produces a fitness score, indicating whether the current assignment clusters the features close together (internal distance) and whether the regions match (region similarity). This results in a much slower but more accurate process, although it can still fail occasionally. A failure is determined to occur in case the best fitness score is low, in which case the matcher informs the controller of a failure and starts over from the beginning. The difference is that on the next update the search region is deliberately expanded to increase the chance of recovery in case of drift.

5.4 Matching and Energy Costs

So far this chapter has discussed a number of methods to identify the correct match for a query feature. The overall goal has been matching accuracy, a correct assignment is critically important in order to obtain an accurate pose estimate. A small temporal delay of the positioning data due to computational overhead is acceptable and data fusion with lagging sensors is a well researched area[75]. However, a more important issue is the time to match, since it is directly dependent on the selected processing chain and the processing platform (such as CPU, GPGPU or FPGA) which can consume a significant amount of power.

The overall goal of this thesis is to investigate and develop algorithms for unmanned aerial vehicles and space vehicles, two types of vehicles that typically operate with strict power limitations. The available power varies from vehicle to vehicle but typically it ranges from the low hundreds of Watts for a small fixed wing UAV to about 2,000 Watts for a satellite in orbit. A a number of systems such as flight control, propulsion and communications all compete for this power which leaves a small portion available for mission tasks, such as sensors and analysis. The proposed visual positioning system takes advantage of the mission sensors and is therefore constrained by their the power limitations.

As such, the actual design and selection of sensors is directly dependent on the type of vehicle, the available sensors and the available processing power. Fortunately, the UAV world is moving towards a scenario where more and more processing is carried out on the vehicle in order to increase the level of autonomy of the vehicle. Meanwhile, advances in processor design and manufacturing is continuously following Moore's law, which dictates that the number of transistors in a processor approximately doubles every two years. Similarly, studies have shown that the electrical efficiency of processors has doubled every 18 months for the past 60 years[76].

This means that methods that currently are too slow to be used on a real platform due to power constraints are likely to be applicable in the future. There is also a rise in parallel computing and the use of GPGPUs (General Purpose Graphics Processing Units), which are highly efficient massively parallel processing platforms. These would enable the positioning system to operate in parallel, either to increase the update rate of the system or by reducing the time to match. In particular, this would speed up the scoring methods which match a small set of query features against a large target set since this process is highly repeatable and each computation is independent of the result of the others.

It would also allow the system to parallelise one of the currently slowest parts of the matching process: the constraint evaluation. Since the algorithm is evaluating each constraint for each assignment combination this involves a large number of computations, which again are repeatable and independent. These algorithms would suit a GPGPU perfectly.

GPGPUs give a large performance boost and compare favourably with CPUs in terms of computations per Watt for parallel tasks[77], but they still add a significant power demand to the system. A current high-performance GPU can consume over 250 Watts[78], which could be suitable for a crater matching satellite in orbit that only needs to correct its position occasionally, while a different low power design would be required for a small fixed-wing UAV. The benefit of the proposed architecture and methods is that the algorithms are interchangeable and can easily be targeted to a specific vehicle with particular power requirements.

5.5 Conclusion

This chapter has presented the two main problems with matching the geometric descriptor: multiple regions around the world can have a very similar appearance and the descriptor uses an aggressive data reduction method in order to efficiently store the geometric region. These issues mean that a single feature vector is too ambiguous to be matched directly to a large database and a more sophisticated approach is required.

Feature matching, in particular for positioning, is a difficult process that is constrained by a number of factors. One is the pose estimation following the matching process. Pose estimation algorithms usually do not require a large number of feature matches, but they require high accuracy for those matches. Another is time to match, an issue that is highly dependent on the platform and the operational scenario.

Because of this a number of methods have been developed that helps increase the chance of identifying correct matches. These include feature selectors that identify poorly conditioned query features and constraint methods that identify whether a match is correct.

In addition, the system takes advantage of the current estimated position. This estimate is critical for various systems onboard unmanned vehicles and is a very effective way to speed up the matching process. In practice it is used to extract a specific region from the database, thereby limiting the size of the target set drastically but speeding up the matching process and increasing the accuracy.

In cases where the position estimate is highly accurate, i.e. the extracted target region matches the query region very closely, the matcher can be further optimised by bypassing the candidate search and validation. This drastically increases the speed of matching and lets the system run at a higher rate once a position ”fix” has been obtained.

Ultimately, the matching process is a computationally intensive process that is dependent on the operational parameters, the type of vehicle and the available power. For that reason the matcher has been designed to be modular, allowing various algorithms to be enabled or disabled as needed.

The following chapter will look at two matcher configurations in order to evaluate the performance of the descriptor and matcher system.

Chapter 6

System Performance

The previous chapters have developed a new type of descriptor and matcher for landmarks but the algorithms have been discussed in relatively conceptual terms. This chapter aims to put some numbers behind the algorithms by creating two potential implementations of the system and analysing how these perform. The implementations have been chosen to simulate two vehicles at opposite ends of the unmanned vehicle spectrum, one being a small fixed-wing UAV needing continuous positioning updates in a small operating region and the other being a Mars lander that uses the visual positioning system for a short period of the flight to enable high accuracy landings on a remote planet.

Additionally, the two vehicles would be operating using two distinctly different landmark arrangements. The UAV uses buildings, meaning it relies on man-made semi-structured landmarks. Meanwhile, the lander will be designed to use craters that are nearly randomly distributed on the surface of the planet, which negates the benefits of some of the algorithms developed in the previous chapters.

Each vehicle will be discussed in detail, starting with assumptions about the operating scenario and mission as well as the performance of external systems or algorithms. After this, each implementation will be detailed, relevant parameters will be defined along with a motivation for the values they are assigned, and finally the performance of the system will be evaluated.

These tests aim to show some of the strengths of the system, primarily by being able to match landmarks to known references even given relatively large initial positioning errors. Further, the computational performance and the effects of computational constraints imposed by the two vehicles will be discussed.

Since these vehicles are made up for the performance evaluation most of the vehicle parameters are based on realistic assumptions. Actual vehicles, in particular future vehicles that the VPS may be implemented on, will likely perform better than assumed as well as be equipped with more robust systems.

6.1 VPS System Design for Mini-UAV

The first of the two platforms is a hypothetical low-cost Mini-UAV that is carrying out a short mission (around 2-3 hours) in a small region of about 100 km^2. As an example, the missions this vehicle could carry out would be to monitor events in a city or being launched from a military vehicle to assess a city or village ahead. This vehicle is used to demonstrate the functionality of the system for low cost and low performance UAVs. In addition, the operating regions are fairly small but challenging due to the structured nature of cities.

6.1.1 Imaging System and Landmarks

The vehicle can be assumed to be operating at an altitude of 1,500 ft (500 m) and equipped with an imaging system with a field of view of 20 x 30 degrees. This gives a footprint for a nadir camera of 260 m x 175 which, as previously discussed in Section 5.1, gives a maximum of approximately a hundred landmarks per capture. The sensor is assumed to have a resolution 2000 by 1300 pixels, giving a spatial resolution of 0.13 meters per pixel and an estimated average landmark size of approximately 80 by 80 pixels. These landmarks have an estimated detection error of two pixels and a failure rate of 2%.

Both of the proposed missions involve small regions that can be very dense in terms of landmarks. These landmarks are likely to be either structured (placed along a road or in distinct repeating patterns) or semi-structured (generally aligned to the vertices in a grid but with some variety in their arrangement). The reference database has

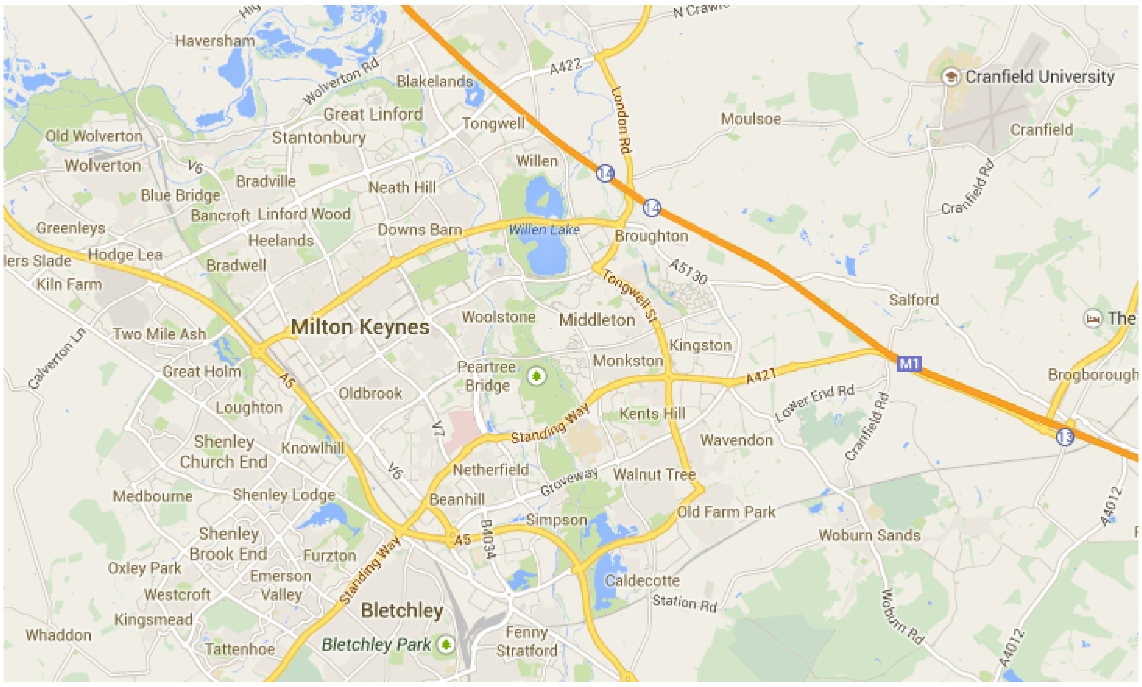


Figure 6.1: Map of Region Covering Milton Keynes and Cranfield Used for Mini-UAV Tests (captured from Google Maps)

been constructed using data from the Ordnance Survey's MasterMap (as discussed in Chapter 3). A region covering the city of Milton Keynes and Cranfield (Figure 6.1) has been extracted from the MasterMap, giving a total of 50,000 features from a 15 km x 10 km region, that includes both urban and rural areas.

6.1.2 Current Positioning Systems

Small unmanned vehicles of this type are generally using low-cost GPS-aided inertial systems such as the Xsens MTI-G-700. These units provide position and attitude in global coordinates (WGS84) but are dependent on the GPS to correctly estimate these parameters. The proposed VPS is intended to replace GPS in case of a failure, thus the time between the failure of the GPS and the loss vehicle control is critical parameter since the VPS must obtain a position estimate followed by full rate positioning updates before this time has passed.

The Xsense IMUs are often chosen as they provide a high accuracy estimate of the vehicle's attitude for a long period of time (the MTI-G-700 claims a gyro drift of 10 degrees per hour). Unlike acceleration, which only affects the estimated position of the vehicle, gyros are used to estimate the attitude of the vehicle and ensure that the vehicle maintains stable flight. This means that an unmanned fixed-wing vehicle can continue to operate without a position estimate for a period of time but drift or loss of attitude data is a critical failure.

Normally the drift is caused by biases on the gyros, which can be estimated by fusing IMU and GPS data as discussed in Chapter 2. Therefore the failure point is dependent on the autopilot's ability to control the vehicle given slightly incorrect attitude data. This is dependent on the robustness of the control system and it is reasonable to assume that a few degrees is acceptable as other critical and controllable parameters such as airspeed, angle of attack and altitude are available from other sensors.

However, while the vehicle might not need a positioning estimate to continue flight, the VPS will need an approximate starting location. When GPS fails this estimate is maintained by integration of IMU data, including accelerometers. These accelerometers will not only be measuring the motion of the vehicle but also gravity, which must be subtracted from IMU the readings in order to provide an accurate positioning estimate. This gives rise to the most critical problem when using IMUs for positioning; if the gyros drift by even a small amount the gravity which normally acts along the Z-axis will begin to contaminate the X and Y axes. As a result the velocity and position estimates will begin to drift approximately quadratically as seen in Figure 6.2, where the effect of a linearly increasing (from zero to one degree) gyro error in one axis on the position estimate is shown with respect to time. In reality there will be drift on all three axes, giving an even larger final error.

Figure 6.2 shows a quadratic positioning error with respect to time, which is directly related to the size of the search region the VPS matcher will have to process. The time taken to match in the VPS is in turn a non-linear function of the number of features in the search area. As a result, the exact time when the VPS is no longer able to match due to time constraints is dependent on the processing power of the vehicle, but a reasonable estimate is around a hundred seconds, giving a positioning error of up to 150 meters per axis assuming the control inputs are known.

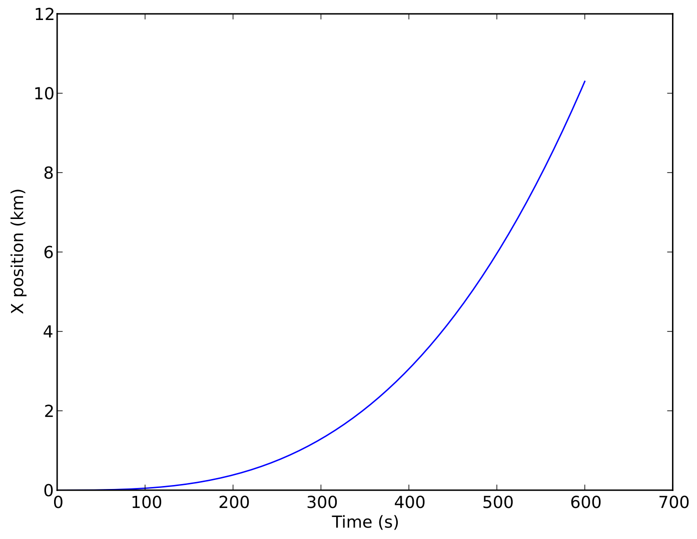


Figure 6.2: Position Drift due to Increasing Gyro Bias Error (linear increase from zero to one degree)

6.1.3 Reference Data

Figure 6.1 shows the area that is being used for this test. The extracted landmarks have been processed to join intersecting or overlapping features and the centroid for each feature has been computed. This reduced the database to approximately 40,000 features, which have been reprojected from British National Grid coordinates to Spherical Mercator in order to preserve the aspect ratio of the landmarks and provide the same view as observed by a UAV (Figure 6.3). Finally, these features have been fingerprinted using the geometric descriptor with a resolution of one degree per bit and the ten nearest neighbours have been stored for each fingerprint.

The use of a landmark database instead of raw imagery results in a significant reduction in the size of the database, allowing the vehicle to carry a larger regional database. The geospatial database used for testing contains over 40,000 fingerprinted features distributed over a 150 km^2 area and is 15 MB in size. By comparison, the aerial dataset covering the same area at 0.5 m/pixel resolution and using standard JPEG compression is 400 MB.

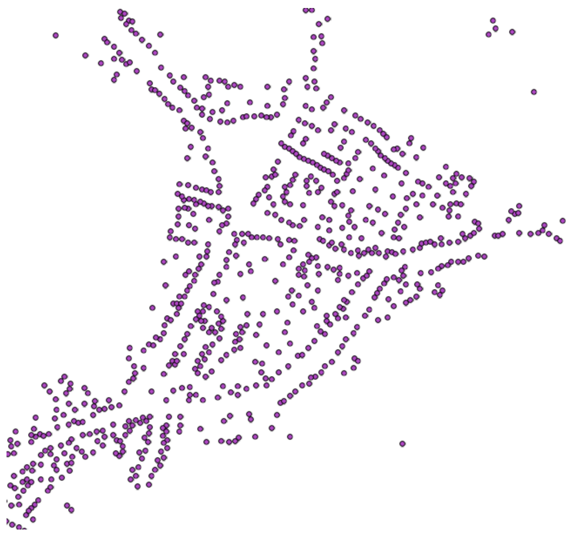


Figure 6.3: Result of Processing Building Features from the OS MasterMap of Cranfield

6.1.4 Visual Positioning System Configuration

Since the VPS will be operating in a semi-structured environment it will be needing a full processing chain, including feature selection, neighbourhood scoring and constraint validation in order to obtain an initial match. Once the first match has been correctly obtained the system can optimise itself by switching over to feature verification, which will significantly speed up the time to match and provide positioning data at a higher rate.

Descriptor

Given a landmark size of 80 x 80 pixels, a spacing of approximately 80 pixels between each landmark and a detection error of two pixels, the estimated angular error in the fingerprint is ± 1 degree. In order to match the database, each fingerprint encodes its ten nearest neighbours for validation.

Feature Selectors

As the environment is semi-structured, the VPS will use both the Gabriel Graph and Central feature selector. The Central feature selector is used to avoid matching incomplete features near the edges of the frame, while the Gabriel graph eliminates poorly conditioned features such as features placed along a road.

The Central feature selector has been configured to use a linear scale, that provides a score from zero to one where zero is at the x or y limits of the image and one is at the centre of the image. As previously discussed in Section 5.1.1, this approach has been chosen as it favours more complete features in sparse scenes.

Meanwhile, the Gabriel graph requires very little configuration as it only requires a integer specifying the number of features to return. Thus the Gabriel graph algorithm will always return the same number of features but the scores for the features are normalised to the set, the first feature returned will always have a score of one, then the n next best features are returned to the matcher. In the UAV configuration the VPS will be time constrained, so the Gabriel graph has been configured to return the 20 best features in each scene.

Database Region Extraction

The system needs a number of parameters in order to extract the expected region from the database. This depends on the last known position, the vehicle's velocity, control inputs from the autopilot and the drift on the inertial sensors. In order to simplify the problem the vehicle's movement and control inputs have been taken out of the system. Once the simulated vehicle looses GPS it maintains the last known position with an increasing positioning error.

While it is possible to add the control inputs from the autopilot, these parameters have no meaningful effect on the performance of the descriptor and matcher, which are benchmarked on the recall rate (see Section 6.3 for further details). In addition, the control inputs add further errors which cannot be modelled without more information about the vehicle and autopilot.

The region selection is thus computed as the last known position plus the estimated distance travelled according to the IMU. In addition, this region is expanded by the estimated drift in X and Y as a function of time seen in Figure 6.2. The drift has been scaled up by a factor of two to cover other unknown in the system, such as changes in altitude.

Thus, if a new image is captured dt seconds after the loss of GPS the search region is:

$$X_c = X_0 + \Delta X_{IMU} \tag{6.1}$$

where X_c is the centre of the search region, X_0 is a 1D vector of the last known position and ΔX_{IMU} is the change in position integrated from the IMU. The size of the search region is:

$$X_s = S_f + 2 * D(dt) \tag{6.2}$$

where X_s is a 1D vector composed of the width and height of the search region, S_f is a vector of the sensor footprint and $D(dt)$ is the drift error as a function of time since last GPS update.

Optimisation Threshold

If the estimated drift is low then the system can disable the initial candidate search. This threshold is dependent on the number of estimated additional features in the image, thus if the drift is low there will only be a small number of additional features, while a high drift error yields a large number of features and causes the direct matcher to fail. In this test the threshold has been set to 25 meters which, given the estimated feature density, gives up to 40 additional features in each frame depending on the type of scene.

Constraints

The system will only use the region similarity constraint since the search region is small and the last known position is known (the internal distance constraint will not be able to provide meaningful data when used with small search regions). Region similarity returns a score for the fit of each assignment and the highest scoring assignment is assumed to be correct.

When constraints are used for validation in the optimised matching method, the system returns the percentage of triangles that matched between the query and target set. The assignment is considered to be correct if at least 95% of the triangles match.

6.2 VPS System Design for Mars Lander

The second platform falls into the other end of the unmanned vehicle spectrum by being a high risk and high cost vehicle where the success of the mission is critically dependent on the positioning system. The aim of a lander is to set down as close as possible to a designated area of scientific interest on another planet. Failure to land at the correct region can at worst lead to a complete failure of the mission, at a cost of several hundred million pounds.

Another aspect to consider is the available computational power. Computers operating in space need to be hardened to handle radiation that would otherwise cause memory corruption and crashes (this is also a common issue on earth, critical computer systems such as servers use error checking and hardened memory modules to prevent data loss caused by background radiation). Hardening processors and memory modules for operation in space causes the hardware to lag behind the current state of the art by up to ten years, thus severely restricting which algorithms can be used on the vehicle.

By design, landers can provide short periods of high power, for example for critical parts of the flight such as landing. This is supplied from batteries that are charged by solar panels during the cruise phase and means that processors can run at the highest possible clockspeeds and high power components can be considered if they provide a signficant benefit to the mission.

6.2.1 Imaging System and Landmark Detection

As previously discussed in Section 2.5, there has been a signficant amount of work done on detecting craters, primarily by researchers at NASA's Jet Propulsion Laboratory. These detection methods are efficient and have been paired with SLAM algorithms to provide an additional odometry estimate during landing, which has been validated using real drop tests on earth [47]. SLAM gives a relative position estimate and helps ensure that the vehicle maintains the planned trajectory during atmospheric entry but does not provide an absolute position needed to ensure an accurate landing.

The VPS would be of benefit since it can locate the vehicle in the planet's local coordinate system and help ensure a highly accurate descent to the intended scientific target. VPS benefits from having a highly unstructured and robust environment to navigate by, craters can be assumed to be randomly and uniformly distributed on the planet's surface and they are long lasting features. Further, due to the thin atmosphere there are no clouds that can obscure the view during descent and dust storms tend to be local events that does not affect the visibility of large landmarks such as craters.

Determining the properties of the imaging system is a difficult task, since it requires a system that is capable of detecting features at high altitude and throughout the descent. This will inevitably require a system with a variable focal length and, since craters are of varying sizes, a contextual database that can switch between different detection levels as features scale in and out of the view. In order to simplify this study the performance of the system is based on a vehicle currently in orbit and equipped with a detector based on the work by Cheng [47]. In this report, craters are detected with greater than 95% accuracy and subpixel accuracy, given an image size of 830 by 470.

Each capture by the system has been designed so that it captures approximately 150 features, which is the equivalent of a 7% by 7% capture of the database.

6.2.2 Current Positioning Systems

Landers fuse several data sources during the flight. During the cruise phase the vehicle is kept on its intended trajectory using IMUs, which are corrected using star trackers that provide attitude to within 0.1 degree. In addition, the vehicles are affected by the planets' spin rates, thermal and ration pressure from the sun and even the wobble of each planet's magnetic axis, all of which need to be modelled and taken into account for the navigation solution. These models allow the vehicle to reach the planet and enter orbit but additional corrections are required to ensure an accurate landing on target.

In recent years radio signals transmitted via the Deep Space Network (DSN) have been used to help estimate Mars landers' position and velocity in orbit, which has resulted in highly accurate insertions at the top of the atmosphere (the Mars Exploration Rover was within 200 meters of its target). Radio-based methods require a large amount of human interaction and can not be used to maintain the vehicle's position estimate during the descent where primarily drag and wind can cause large positional errors.

6.2.3 Reference Data

Mars has been extensively mapped in the past twenty years, in particular using the Mars Reconnaisance Orbiter's HiRISE camera. This has provided a highly accurate terrain map of Mars with a planar resolution of approximately 0.3 meters per pixel and an elevation map with up to 0.25 meter resolution. This map can be processed using a crater detector at different scales in order to provide a reference database for the VPS.

Fortunately it is very easy to emulate this, unlike features on earth craters are randomly distributed on the surface of the planet. This has been simulated by creating 30,000 features in the database, each of which has been fingerprinted with one degree per bit resolution and assigned ten neighbours.

6.2.4 Visual Positioning System Configuration

The VPS configuration for the lander is generally the same as for the Mini-UAV with a few exceptions in order to account for the crater detector, larger search region and unstructured landmark distribution. Unless otherwise specified this configuration is using the same parameters as described in Section 6.1.4.

Descriptor

The detector is capable of detecting landmarks with subpixel accuracy, which gives an angular error of less than one degree. This is lower than the design resolution for a fingerprint (one degree per bit), which has therefore been modified to provide a resolution of 0.5 degrees per bit and one degree error. This gives a 360 bits long feature vector with a flat error of ± 2 bits per angle.

Feature Selectors

Unlike the Mini-UAV, the lander is operating in an unstructured environment. This results in a very low level of weak features which means that the Gabriel graph becomes ineffective and, to a great extent, unneccessary. As a result the lander has been configured to use the same central feature selector as the Mini-UAV while the Gabriel graph selector has been disabled. The matcher selects the 20 highest scoring features from the central feature selector and then proceeds with the matching process.

This has a side-effect of matching a majority of features near the centre of the image. Large numbers of central features can have a detrimental effect on the pose estimation algorithm - in particular on the elevation estimate, but this is dependent on the pose estimation and data fusion methods used.

Database Region Extraction

Since the database has been randomly generated it does not have a specific scale and the details of the capture location and altitude have not been defined. As discussed in 6.2.1, the detection has been designed to simulate a capture of a 7% by 7% square from the database. If the database is the equivalent to a hemisphere it is reasonable to assume that the initial position is known to within 15 percent of the database. As such, the search region is 40% by 40% square, giving approximately 4,500 features in the target set.

Optimisation Threshold

The lander is equipped with high performance inertial sensors and startrackers that are able to maintain the attitude of the vehicle for an extended period (the MRO was estimated to have drifted 3.7 km in ten days without corrections). As such, there is no reason to run the VPS at high rates, and slower but more accurate algorithms are favourable.

Constraints

Both constraint methods will be applied for this method since the target region contains significantly more features than the region matched by the Mini-UAV. The internal distance metric is therefore a very effective method to ensure that the assignment of the candidate features results in a cluster in the same region.

The system is also using 20 candidates instead of ten, in order to improve the chances that a correct match will be identified.

6.3 Matching Performance

The primary performance indicator of the descriptor and matching system is the recall rate. The recall rate gives a measure of how successful a query was and is computed as the ratio of the number of features that were successfully matched to the number of features in the query. The query features are the features initially extracted by the landmark detector unless a feature selection algorithm is used, in which case the query features are the features selected by the algorithm.

In each evaluation 500 random queries are matched and averaged to give the estimated recall rate. Since both systems use a reference database it is possible to extract a truth from the database and modify it to simulate the different parameters studied in the following sections and still maintain a correct ID for each feature.

Each section will evaluate the effect of various parameters compared to the baseline system configurations described in the previous sections.

6.3.1 Detector

Table 6.1 shows the effect of detection noise on the recall rate. These results have been produced by adding random noise to each feature in the query set before generating the fingerprint. As expected, the system performs very well as long as the noise matches the expected parameters (recall that the descriptor for the Mini-UAV was designed for a detection error of two pixels while the lander system expects an error of less than one pixel). Once the error increases far beyond the intended detection error the recall rate drops off signficantly and becomes unusable for positioning.

Further, Table 6.2 show the performance of the system with two pixels noise but gradually decreasing detection rates. In this case 5% and 10% of the features were either moved or removed from the query set prior to fingerprinting and show that a poor detection method has a very detrimental effect on the system. These results emphasizes how important it is to have a robust and predictable detector, that is less likely to suffer from false or failed detections.

While the lander has a technically more accurate detector since there is less variety in the set of features it is trying to detect, it also suffers more when there is an increase in ambiguity due to incorrect fingerprints.

Meanwhile, Figure 6.4 shows the effect of a change in view angle. The project has so far assumed a fully top-down view of the terrain below but the built in noise model, neighbourhood matching and constraint evaluation makes it possible to offset the camera relative to the ground and still obtain relevant matching results (assuming the detector is capable of dealing with the perspective change). While the angular limit for acceptable matching is relatively low (approximately 13 degrees), the results show that small changes on the camera's orientation do not pose a significant problem for the matcher.

6.3.2 Feature Selector

Table 6.3 shows the impact of the feature selector. Eliminating incomplete features with the central feature selector has a significant impact on the recall rate and the

Table 6.1: Effect of Detection Noise on Recall Rate

Noise (pixels)	Mini-UAV	Lander
0	99%	99%
2	96%	72%
4	92%	34%
6	68%	15%

Table 6.2: Effect of Detection Rate on Recall Rate

Detection Rate	Mini-UAV	Lander
100%	96%	97%
95%	85%	78%
90%	67%	54%

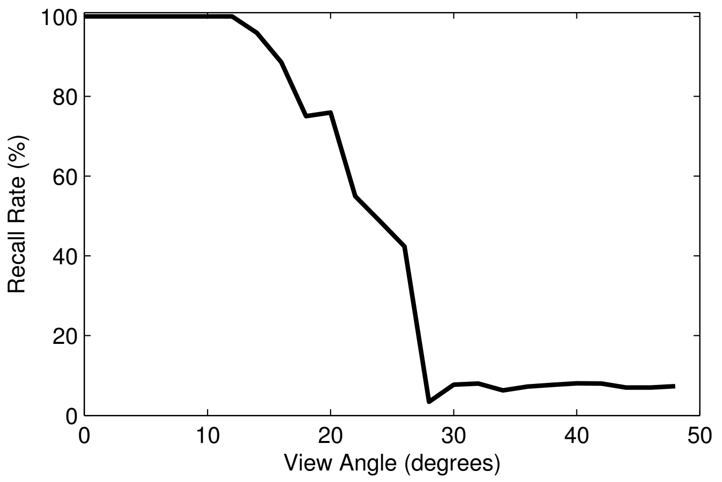


Figure 6.4: Recall Rate for Increasing View Angles

Table 6.3: Effect of Feature Selector on Recall Rate

Test Condition	Mini-UAV	Lander
No selector	79%	84%
Central only	89%	96%
Central + Gabriel	93%	95%

Mini-UAV benefits from the additional use of the Gabriel selector. However, the Gabriel selector has little effect on the lander due to the unstructured landmarks - weak, ambiguous or poorly conditioned features are much rarer in unstructured environments - which is reflected in the recall rate. The Grabriel graph has only been applied to illustrate this issue.

6.3.3 Matcher and Constraints

Table 6.4 shows the effect of varying the number of candidates on the recall rate for the two vehicles. The Mini-UAV outperforms the Lander since it is matching to a smaller region and is able to further reduce the number of features in its target set thanks to a very targeted search region.

As a result, it is more likely that the correct match will be one of the best scoring candidates once the neighbourhood verification has been carried out. This can also be observed by looking at the effect of increasing the number of candidates for the Mini-UAV further, which does not result in a meaningful increase in the success rate. Since the correct assignment is likely to be a high scoring match additional features simply increase the time to match, and can occasionally result in a failed match as more false matches are introduced into the constraint algorithm.

However, increasing the number of candidates is noticeably improving the recall rate for the lander since it has a larger search region resulting in a higher number of ambiguous features. By obtaining more candidates the system increases the chance that the correct match is a member of the candidate set.

The Mini-UAV benefits from the lower candidate count since it reduces the computational overhead. Table 6.5 shows the effect of increasing the number of candidates

Table 6.4: Effect of Number of Match Candidates on Recall Rate

Number of Candidates	Mini-UAV	Lander
5	84%	64%
10	93%	82%
20	94%	95%

Table 6.5: Effect of Number of Match Candidates on Matching Time

Number of Candidates	Mini-UAV	Lander
5	220 ms	353 ms
10	532 ms	814 ms
20	1236 ms	1,828 ms

on the time to match, once the query set has been passed through the feature selection.

The lander takes longer to carry out a match since it uses a 360 bit long feature vector versus the 180 bit feature vector for the UAV. In addition, it uses both the internal distance and region similarity constraints to evaluate assignments, unlike the Mini-UAV which only uses region similarity. This results in a relatively long time required to match but it is worth keeping in mind that these times have been obtained using a research version of the VPS implemented in Python and can be reduced significantly with proper software implementation and optimisation.

Table 6.6 shows the results of the constraints on the recall rate. As a comparison, a direct match without candidates and validation is shown as well, which results in poor matching accuracy for both vehicles. The lander is doing much worse due to

Table 6.6: Effect of Constraints on Recall Rate

Test Condition	Mini-UAV	Lander
Highest Candidate Score	56%	28%
Internal Distance (ID)	76%	73%
Region Similarity (RS)	94%	85%
ID + RS	94%	95%

the larger target region but at these matching rates either system is unusuable for positioning.

Introducing the constraints improves the results drastically though. Both constraints increase the recall rate sufficiently to use a positioning method that uses outlier rejection, but the region similarity is much more effective for the Mini-UAV.

Combining the two constraints drastically improves the performance for the lander as the two methods effectively reject invalid assignments in a large region. The Mini-UAV meanwhile does not benefit from the same increase in performance, since the internal distance constraint is only effective on large regions. On small regions it provides similar, but more ambiguous results than the region similarity method.

6.3.4 Optimised Matcher

The Mini-UAV (and to a lesser degree the Lander) is able to use an optimised matching process when the position is known with high confidence that effectively lets the system lock on to a position. This matching process does not use candidates or constraints, instead it relies solely on the neighbourhood score to identify correct matches. The method requires a low likelyhood of ambiguous matches and is therefore only applicable in highly targeted matches but it removes one of the most computationally expensive stages of the matching process.

Table 6.7 shows the performance of the matcher with increasing positioning errors when the candidate and constraint matching is disabled. When the target region is known with less than 25% error the optimised matcher performs as well the full matcher but once the search region grows due to uncertainties then the performance tapers off very quickly. However, for a 25% uncertainty the matching time reduces to 83 ms per match, compared to 532 ms for for the full matcher.

6.4 Conclusion

This chapter has developed two configurations of the VPS for two distinctly different types of missions. One is a Mini-UAV designed to operate in an urban region with

Table 6.7: Recall Rate for Optimised Matcher

Test Condition	Recall Rate
Exact Region	95%
25% Region Growth	91%
50% Region Growth	82%
100% Region Growth	47%

semi-structured landmarks while the other is a Mars lander designed to operate with unstructured landmarks.

The differences in landmark structure gives two different systems since certain algorithms and methods are not applicable in each case. For example, the Mini-UAV does not use the internal distance measure since it is ineffective in targeted search regions. Meanwhile, the lander does not use an optimised matcher once a position has been obtained with high confidence. This is due to the performance of other on-board positioning systems which can maintain positioning estimates with sufficient accuracy. As a result it is preferrable to carry out a full match and ensure higher matching performance but with a longer matching time.

In addition, differences in the performance of sensors and detection methods on each vehicle affect various aspects of the VPS. These changes have been discussed in detail and each system has been validated by adjusting various parameters and studying the performance of the VPS.

The performance has been evaluated by analysing the recall rate, the number of features correctly matched as a ratio of the number of features in the query for each parameter as well as the optimised matching strategy.

Results show that the VPS performs well sa long as it is given scenes that are within the design of the system. However, over-designing the system in order to cover for unknowns can have a detrimental effect, with reduced recall rates and an increase in false matches. As a result, it is important to fully understand the performance of each sensor on the vehicle and extensively test the system prior to deployment.

The following chapter will review the results of this thesis and discuss the validity of the overall system.

Chapter 7

Discussion

7.1 Background

This project has aimed to investigate the current state of visual positioning systems for unmanned aerial vehicles and to develop a new approach to vision-based positioning that can provide an absolute rather than relative position estimate. The primary advantage of utilising vision for positioning is that visual systems can operate independently of external signals and thus provide a robust and independent positioning system. Current common positioning systems, such as GPS, rely on external signals and thus can fail due to interference (e.g. urban canyons in cities) or lack of signals (e.g. GPS is unavailable for a lander on Mars) and are also susceptible to tampering, either through spoofing or jamming. Furthermore, a vision-based positioning system can complement other positioning systems by providing a fallback system in the case of failure, improving the overall robustness and reliability of the vehicles positioning system as a whole.

Vision is an information-rich data source. Since the late 1960s, the potential of vision as a sensor within computer science has been explored; however, the problem has proven to be more complex than initially expected. Vision is not simply a visual task; it requires both the identification and interpretation of features within the environment. Vision thus requires recognition; for example, the human brain combines both visual data and contextual information to provide a high level understanding of what is seen. In recognition of this, David Marr developed a framework where he aimed to outline the challenges that are required to be overcome in computer vision in order to mimic the brain's abilities to recognize and interpret visual information.

Whilst these challenges have yet to be fulfilled, the field of computer vision continues to be extensively researched and a vast number of algorithms have since been developed, ranging from corner detection and camera calibration algorithms to complex 3D modeling and facial recognition methods. In addition, vision has proven to be an effective method for solving certain navigation problems; for example, visual odometry can in certain scenarios provide a highly accurate position estimate over long periods of time.

Each of these tasks, however, need to be carefully controlled in order to achieve the desired results. Whilst Marr's model has encouraged researchers to explore and study methods that enables vision systems to incorporate higher level information, the work is still in its infancy.

As a result, the majority of the current work on visual positioning systems has focused on relative positioning. Relative systems rely on a frame-to-frame analysis, where a feature is matched from one frame to the next. This enables algorithms, such as optical flow and SLAM, to track features that can then be used to determine the motion of the vehicle. Whilst relative positioning attempts to demonstrate Marrs model, the frame-to-frame analysis requires very little understanding of the scene, and thus simplifies the problem significantly.

The main challenge with relative positioning methods is that they cannot be used to correct the true position of the vehicle. For example, a lander that has flown to Mars will only be able to determine its position relative to its starting location, and will not be able to align itself with the local coordinate system and/or perform a landing at a specific location. The systems are also highly vulnerable to visual interruptions, such as if the vehicle flies through a cloud. This temporary obscuration of the camera forces the vehicle to fall back on less accurate sensors and introduces considerable positioning errors that cannot be recovered once visual tracking is resumed.

To overcome these issues, the literature review was used to investigate the current state of the art for absolute visual positioning systems. The review identified that absolute positioning is currently studied and used extensively for localised problems, such as augmented reality applications. In an augmented reality application, a 3D graphic is overlaid on a display of the real world. The graphic can then be viewed from different perspectives by using an algorithm that tracks the position of the camera relative to the target. The system that tracks the position consists of two stages. First, specific features on a marker are detected and associated with known 3D coordinates (feature detection) and then the position and orientation of the camera relative to the target is determined by the pose estimation algorithm (pose estimation).

As it is unrealistic and impractical to distribute markers around the world, a marker-based system is not sufficiently scalable to assist UAVs with navigation. Thus research in the field of absolute visual positioning has required the development of other methods. The most promising work has been carried out by Conte[2], who used a 2D cross-correlation method to match an image captured from a vehicle to a satellite image of the region. Pose estimation is carried out by identifying to optimal alignment of the aerial image within the geo-referenced satellite image. This method has been flight-tested but struggles under anything but perfect conditions.

As a result, this literature review has taken a step back to the methods used in augmented reality applications and found that these algorithms could be applied on a global scale by using landmarks as known reference features (rather than markers). In order for this to work, three different areas needed to be explored: landmark detection, feature matching and pose estimation.

The review demonstrated that the detection and pose estimation problems have been extensively studied and, to a large degree, solved. To facilitate feature detection, for example, there are algorithms for that enable accurate detection of buildings, roads and other landmarks in satellite and aerial imagery, as well as methods to detect craters in imagery captured by satellites and landers. Similarly, pose estimation algorithms have been developed that allow accurate positioning of cameras relative to a target once the camera and vehicle model has been defined. The accuracy and reliability of the pose estimation depends on obtaining a sufficient number of point correspondences, where the features in the aerial image can be associated with their global reference matches.

The literature has shown extensive work on matching features to a reference, such as song matching, but none of this work has been applied to geographic features. Consequently, it was identified that a primary aim for the thesis should be to investigate the possibility of developing a geographic feature descriptor and matcher that enables pose recovery from aerial images.

A key issue to consider during the development of a localisation system is the reliance on other data. The aim has been to avoid relying on prior knowledge about position and orientation of the vehicle in order to develop a system that is independent or, at most, loosely coupled with other systems on the vehicle. If the proposed system is overly reliant on inputs from other systems it will fail as they fail, making the system largely unusable. However, it has been assumed that the vehicle has an approximate position and accurate attitude estimate since these are required for critical flight control tasks.

7.2 Landmark Detection and Availability

To locate a vehicle from aerial and satellite imagery, as this thesis has shown, requires a distinct type of topography: landmarks (features) that are abundant and reliably detectable. The primary feature of focus for this thesis has been residential buildings, identified due to their consistent appearance (usually slanted roofs and rectangular shapes), an attribute which makes them relatively easy to detect reliably. Furthermore, as building detection is used extensively in many applications within Geographic Information Systems (GIS) and mapping, a significant amount of research has been carried out in this field.

The use of residential buildings as a primary feature however can limit the applicability of the system, and thus operational scenario, to urban areas. To overcome this limitation, the system could incorporate additional feature detectors that can run in parallel or contextually, depending on the current mission. For example, buildings can be used in urban regions while farms and forests are used in rural areas. Similarly, road-junctions can be used in either scenario. Ultimately, the system has been designed to support any class of point feature that can be reliably detected from the vehicle.

One requirement the system design cannot account for is the availability of features to detect. If the requisite features are not present or not available in the minimum quantities required to match, then the proposed system will be unable to compute a position estimate. Furthermore, visual sensors are susceptible to obscuration by clouds or inclement weather, thus making it unsuitable for use as a primary positioning system. However, unlike a relative system, clouds do not pose a significant problem for the proposed system as the position of the vehicle can be recovered on exiting a formation, whilst being maintained using other systems. In addition, from a practical viewpoint, it is unlikely that unmanned vehicles would operate extensively in poor weather conditions and heavy cloud as often the primary mission is to capture imagery of ground targets. Clouds and poor weather would prevent adequate imagery from being captured and thus it is unlikely that the mission would be launched.

The vision sensor that captures the aerial imagery may also be supplemented by other sensor data that can help detail the ground below. For example, many vehicles that operate at high altitudes use other types of sensors including synthetic aperture radars, which can penetrate clouds to build a map of the terrain below. This data can be used to identify landmarks and provide an input to the proposed positioning system. As a result, the visual positioning system is not strictly visual and can be modified to work with a number of sensors.

Finally, the current work is exploratory, to determine whether it is possible to match semi-structured landmarks accurately for use in positioning. Due to this the sensing aspect of the problem has taken a back-seat to the description and matching issues.

7.3 Description and Matching

The majority of previous work studying feature description and matching has focused on matching of visual appearance between aerial captures and reference imagery, such as the work by Conte. This type of matching restricts the performance of the system as it requires recent reference imagery that matches the current weather and lighting conditions. To overcome the limitations brought on by temporal and/or seasonal changes, the proposed system uses widely available but visually similar features, such as buildings, to help locate the vehicle.

Further, landmarks cannot be matched by their individual appearance - a residential home seen from the air in London can appear virtually identical to a home in Aberdeen. This rules out a significant number of the current state of the art feature descriptors in computer vision, such as SIFT and SURF, which match features based on their appearance. Instead the system takes an alternative approach by matching the features collectively; thus an alternative descriptor was developed that encodes the geographic relationship between features.

The descriptor is based on the concept that landmarks can be recognised not by their individual appearance but how they are related to other surrounding features. For example, most people can identify their home among a number of houses along a road in a satellite image, even if the house is identical to the others.

To replicate this interpretation, the new descriptor encodes the relationship between a feature and its immediate neighbours. The relationship is encoded in a binary feature vector, known as a fingerprint, that is scale, rotation and translation invariant. For performance reasons, the descriptor is aggressive and discards most of the data associated with the feature that is irrelevant to interpreting location. The shape of the region is encoded in the feature irreversibly, making it a strictly one-way process. However, the descriptor is fully repeatable for varying scales and rotations, making it suitable for the detection of geographic landmarks from various altitudes and rotations. This ensures that the features identified by the detector produced the same fingerprints as those within the reference database, whether the latter was extracted from aerial or satellite imagery.

As a result, the operational envelope scope of the system is not limited by the visual positioning system itself, but rather the landmark detection algorithm. In some scenarios this is not an issue, for example, craters can be detected at a variety of sizes, but in building detection amongst others the envelope is much tighter.

Since landmarks, in particular man-made landmarks, tend to be semi-structured the descriptor on its own is not strong enough to match landmarks on a large scale. Semi-structured regions mean that fingerprints for two features can be very similar, even if they are geographically distant, which becomes a significant issue in the pose estimation algorithm.

This led to the development of a number of strategies in order to reduce the risk of incorrect matches. These include the elimination of poorly conditioned features, high-level region matching and methods to reduce the potential target set extracted from the database. This improved matching performance, giving up to 95% matching accuracy in both test scenarios: urban, semi-structured environments and unstructured crater matching.

The results of the project also demonstrated the effect of sub-optimal matching situations, caused by a poorly designed and configured system. In these situations, the matching performance was greatly reduced, with certain parameters rapidly reducing the matching accuracy to less than 30%. It is critically important that any real world implementation of a vision-based positioning system considers the precise requirements in order to optimise the performance of the landmark sensor and associated algorithms. For example, a fundamental assumption in the proposed system is that landmarks can be detected with a low false positive rate and a specific estimated error. If the detection rate of the landmark algorithm is only slightly lower than expected in flight, it will have a dramatic impact on the performance of the overall system.

The results of the project have shown that vision-based positioning systems are highly sensitive; changes in parameters and inputs can have both positive and negative effects on the overall performance. Consequently any proposed system will require extensive tuning tailored for the prescribed task. This is not unique to this project. For example, Visual-SLAM systems, which are among the most robust visual positioning systems currently available, have a multitude of parameters that need to be adjusted depending on the vehicle, trajectory type and likelihood of loop closures.

In addition, the processing platform on which the proposed vision-based positioning system is implemented can limit the overall functionality of the system. For example, some of the tasks executed by the system, such as landmark detection and the constraint stages of the matcher, can be very computationally expensive and therefore restrict the use of the system in real vehicles. Fortunately, processing hardware is becoming faster and more energy efficient every year; at the moment the mobile devices industry is aggressively advancing technology in this field. It is thus likely that by the time VPS is ready for flight testing, the technology will have sufficiently developed to meet the computation demands.

Ultimately, the performance results for the VPS have merit and show the potential for the descriptor and matcher when used in a complete positioning system. However, current work is still in the proof of concept stage and a significant amount of work is required to move towards flight testing.

7.4 Future Work

There are a number of areas where progress is required in order to advance the system to the stage where it can be flight tested and validated. Some of these areas are critical for accurate positioning in the real world, while others are optional. There are also a number of opportunities for areas where the current descriptor can be applied to add new or improved functionality.

7.4.1 3D Terrain

The flat earth assumption is one of the core assumptions behind the current work. This has helped simplify the problem to reach a state where the initial idea can be validated and proven. However, the world is not two-dimensional and the third dimension needs to be incorporated into the system to obtain a true position estimate.

This should be a straight-forward task since reference maps such as the Ordnance Survey contain elevation data and the database can associate any form of metadata with a feature. As such, the core issue is to obtain accurate elevation maps and associate each building feature with elevation data. This data can immediately be used in the pose estimation algorithms to determine not only the latitude and longitude but also the elevation of the vehicle above mean sea level (AMSL) as measured in WGS84.

7.4.2 Non-nadir Sensors

The second assumption in this project is that the sensor is mounted on a gimball in lock-down mode, thus making the sensor plane parallel with the ground plane at all times. As a result the image is similar to what would be observed by a satellite, making the features easier to match by avoiding perspective distortions and minimising the effect of uneven terrain.

In a real operational scenario, this would require an an additional sensor and gimbal dedicated purely to positioning, thus doubling the payload a vehicle would need to carry; a less than ideal situation when considering the potential increase in weight and power demands of the vehicle. Thus a better approach would be to take advantage of the sensor already mounted for the mission and correct for perspective distortions in the VPS software. The results obtained in this thesis has shown that the system can cope with perspective distortions of up to 13 degrees (see Section 6.3.1). This would allow the gimbal's own state estimates to be used to approximately transform the image coordinates from the sensor plane to an intermediate plane parallel to the ground.

This approach may face difficulties if the ground is not planar; for example, if some of the features lie on a hill that is being observed at an angle. It is possible to make use of a known elevation map however this introduces an additional external dependence and requirement to the system. This is highly impractical and defeats the purpose of the system: to create a positioning system that locates a vehicle without the use of external inputs.

An alternative method may be to generate an estimated elevation map using either a structure from motion algorithm or a simplified version thereof. For example, it is trivial to track simple features such as Harris corners[10] between frames. If an aircraft travels far enough between frames to produce a reference baseline then it becomes possible to triangulate features and determine their approximate position relative to the camera. This is a very light-weight method to obtain approximate elevation data for landmarks and handle unexpected view angles. Furthermore, since this approach is independent from the feature detector, it can run at a high rate to ensure accurate feature tracking and improved elevation recovery. Meanwhile, the landmark detector can be cued on key frames at low rates to conserve energy.

A final difficulty faced with non-nadir sensors is the change in appearance of 3D features. A simple example is buildings; whilst they may have very similar rooftops, their facades can be completely different. This makes detection a much harder problem and thus requires a more sophisticated feature detection algorithm. Fortunately this is only applicable to 3D features; 2D features, such as craters, are only distorted due to the perspective (e.g. circles become ovals), which can be handled by the detector.

7.4.3 Parallelisation and Optimisation

The current processing chain is completely linear which results in a longer time to find a match. However, some of the slowest tasks such as constraint validation are independent and can be carried out in parallel if the processing platform supports it. This is covered in more detail in Section 5.4.

7.4.4 Altitude layers

The proposed visual positioning is ultimately constrained by the landmark detection algorithm. If the detection algorithm is unable to detect features due to altitude then the entire system fails. In some cases, such as building detection, there are not many options other than tuning the detector and providing feedback to it to compensate for altitude changes.

However, in the Mars lander scenario, the problem is exacerbated since the vehicle descends from orbit to ground level in a matter of minutes. During this descent, the features undergo a massive scale change relative to the vehicle; large features that were previously visible leave the view, while small features now occupy enough space in the image to be detected.

To account for these scale changes, it would be useful to construct a multi-layer database for vehicles that carry out this type of descent where the system intelligently switches between layers as the vehicle descends.

Similarly, to optimize the detection of features, it would be advantageous for the system within an aerial vehicle to be capable of switching the context or reference layer depending on the current location. This would allow a vehicle to navigate using buildings in urban areas but switch over to agricultural fields once it reaches rural areas.

7.4.5 Alternative Uses

There are a number of areas where either part or all of the proposed system can be used in alternative ways. Note, these are ideas that have arisen during the work on this thesis and are not in any way an exhaustive list.

Descriptor

The detector is a basic but efficient approach to uniquely describing geometric relationships between features. It is robust to tracking errors, whilst scale, rotation and translation invariant, and can work with any type of point feature. As a result, the descriptor can be applied to a number of problems outside landmark description:

- Face recognition - Locations of key features in a face, such as corners of eyes and mouth, can be extracted and encoded in a fingerprint.
- Data encoding - There are use cases where QR codes are required but the code itself is too complex to be detected reliably. The descriptor could be used to encode data, such as developing a serial number that can reliably/readily be detected from all angles and distances.
- Star trackers - Star trackers are mounted on satellites and continuously track stars by matching them to a reference database. This task is remarkably similar to the proposed system, although it only obtains attitude. Star trackers also use comparatively complex methods to match star patterns. The descriptor could be used to quickly encode star patterns, giving either higher rate attitude estimates or lower power consumption.

VPS

Developing the VPS for real-time vehicle positioning is difficult; a single positioning error can have significant repercussions. However, there are other reasons to obtain positioning information where real-time matches are not required. For example, as discussed in Section 2.3.3, UAVs produce vast amounts of data that needs to be geo-referenced and tagged in order to be of value.

The VPS can thus be used to automatically geo-reference both UAV and satellite imagery, but it has the additional benefit of being able to identify specific features in an image. This allows the processing of historic data from vehicles and enables searches for landmarks instead of coordinates. For example, it becomes possible to search for all aerial views of a specific building between two dates.

This feature-based approach to mapping can be further expanded by referencing the features to other databases. The best example of this is the Ordnance Survey's TOID, which is a unique identifier for all features in the UK. The TOID is used in a wide range of databases, from planning permissions to energy usage and crime reports. This data can be pulled in and associated with each feature in the VPS, allowing high-level querying of UAV data.

Finally, it can be used for systems such as Argus-IS (see Section 2.3.1) that provide a persistent eye in the sky, as it simplifies the identification of particular areas of interest and helps manage the vast amounts of data generated by the sensor.

Chapter 8

Conclusion

This thesis has developed a new approach to visual positioning for unmanned aerial vehicles which, unlike previous methods, is designed from the bottom up to provide absolute positioning data within a global coordinate frame. The aim of the thesis is to provide an alternative to GPS, which enables positioning in GPS-denied environments (due to interference or jamming) or in areas where GPS is not available.

The majority of previous work within vision-based positioning systems has focused on relative positioning methods, such as visual odometry and Self-Location And Mapping (SLAM), whilst comparatively little work has examined and explored the potential for absolute positioning. This disparity is propagated because relative positioning tends to be a simpler task to solve; it relies solely on a frame-by-frame analysis, however does suffer from visual hiatuses or discontinuities (such as clouds). Absolute methods, whilst expected to be more complicated to develop, can recover their position once the hiatus is over and allow vehicles to continue their mission, thus making the system a viable alternative to GPS.

Absolute methods use feature correspondences between an aerial capture and a reference database to obtain global positioning. The challenge with this approach is that even the most robust and reliable landmarks, such as buildings and junctions (often chosen for their low visual variety and long lifespans), can be organised in semi-structured ways (such as in grid-like arrangements or other repeating patterns) that can be difficult to match

To overcome this challenge, this thesis has presented a new framework for an absolute positioning system that consists of four main tasks:

- Landmark Detection
- Feature Description
- Landmark Matching
- Pose Estimation

Through a comprehensive review of current literature, it is acknowledged that two of these tasks within the framework, landmark detection and pose estimation, have been discussed extensively, and in some cases solved. The two outstanding tasks, feature description and landmark matching would need to be solved in order to fulfill the requirements of the framework, and the aim of this thesis.

These requirements have led to the development of a feature descriptor that uniquely describes landmarks based on their geographic arrangement relative to their neighbours. The descriptor is scale, rotation and translation invariant and allows the system to accurately distinguish similar or ambiguous features within the same region. It encodes the appearance of the region in a binary fingerprint that allows memory efficient storage of a large number of features.

The results from the initial tests of the descriptor showed that it can accurately match in small regions, however its performance suffers as the size of regions increases. As a result, a more complex matching approach was necessitated. This approach involved several stages: feature selection, candidate matching, neighbourhood verification and constraint validation.

In feature selection, the query features are down-selected to ensure only well-conditioned features are matched, using a new approach based on connectivity graphs. Candidate matching is a straight-forward scoring matcher; it uses neighbourhood verification to determine a set of potential matches for each query feature. Finally, constraint validation ensures that the selected best matches among the candidates fit with an overall model. The model is based on calculating/matching/comparing the region similarity between the aerial capture and the correct target region in the database.

The results show that in two test scenarios, a mini-UAV operating over a city and a Mars lander preparing for a precision descent, the system can achieve a better than 95% matching accuracy, given a well configured system and good conditions. While these results are promising, it also revealed that the system is sensitive to incorrect estimates of certain parameters, such as the performance of the feature detector. As a result, it is critical for a designer of a future visual positioning system to fully understand the performance and limitations of not only the sensor but also the algorithms and the complete system.

Whilst these sensitivities and limitations are not inconsequential, it is shown that the system can still operate successfully. As shown in the literature review, most visual positioning systems today suffer from similar issues, which, at the core. Is due to an overreliance on vision alone. Although pure vision is an information-rich data source, it requires additional contextual information to truly achieve robustness and reliability.

In addition, it should be noted that in order to prove the concept a number of assumptions were made, which should be addressed in any opportunity of further work. The system currently relies on a flat earth model and does not support changes in elevation. It also assumes that the sensor plane is parallel with the ground plane in order to avoid perspective distortions and distortions due to uneven terrain. This requires the use of a lock-down gimbal and a postioning-dedicated sensor in addition to the mission sensor, which both would add further weight and power requirements.

As discussed, these problems can be overcome and would allow the system to work in parallel with the mission task. Further, since the core of the positioning system relies on point features, any sensor that is capable of detecting point features can be used, such as visible light and thermal cameras or synthetic aperture radars. In particular, the incorporation of radar would allow the system to operate in poor weather conditions.

Finally, there are other uses for the new descriptor besides its implementation within the visual positioning system. The most noteworthy is the potential of the descriptor to process and automatically geo-reference imagery captured by UAVs and satellites whilst tagging landmarks present within the scene. These landmarks can then be associated with external databases, allowing high-level queries in visual data.

Bibliography

[1] Jose Roger-Verdeguer, Mikael Mannberg and Al Savvaris, “Visual Odometry with Failure Detection for the Aegis UAV,” *IEEE Imaging Systems and Techniques Conference Proceedings*, 2012.

[2] Conte, Gianpaolo and Doherty, Patrick, “Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information,” *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1–19, 2009.

[3] Hiroyuki Miura, Saburoh Midorikawa and Kazuo Fujimoto, “Automated Building Detection From High-Resolution Satellite Image for Updating GIS Building Inventory Data,” *13th World Conference on Earthquake Engineering*, 2004.

[4] Mikael Mannberg and Al Savvaris, “Landmark Fingerprinting and Matching for Aerial Positioning Systems,” *AIAA Journal of Aerospace Information Systems*, 2014.

[5] Mikael Mannberg and Al Savvaris, “A Visual Positioning System for UAVs Using Landmark Fingerprinting,” *AIAA InfoTech@Aerospace Conference Proceedings*, 2012.

[6] Mikael Mannberg and Al Savvaris, “Automatic Pipeline Detection for UAVs,” *AIAA InfoTech@Aerospace Conference Proceedings*, 2012.

[7] Mikael Mannberg, Peter Silson, Antonios Tsourdos and Al Savvaris, “High Precision Real-time 3D Tracking Using Cameras,” *AIAA InfoTech@Aerospace Conference Proceedings*, 2011.

[8] Marr, David, “Vision: A computational investigation into the human representation and processing of visual information,” *Henry Holt and Co*, 1982.

[9] Eden Media, *Cybernetic Revolutionaries.* The MIT Press, 1 ed., 2011.

[10] Chris Harris, Mike Stephens, "A combined corner and edge detector.," *Alvey vision conference, Vol. 15*, 1988.

[11] Kasneci, G. and Suchanek, F.M. and Ifrim, G. and Ramanath, M. and Weikum, G., "NAGA: Searching and Ranking Knowledge," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pp. 953–962, April 2008.

[12] Jain, A.K. and Ross, A. and Prabhakar, S., "An introduction to biometric recognition," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, pp. 4–20, Jan 2004.

[13] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe, "Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects," *Information Fusion*, vol. 7, no. 2, pp. 221 – 230, 2006.

[14] Misra, Pratap and Enge, Per, *Global Positioning System: Signals, Measurements and Performance Second Edition.* Lincoln, MA: Ganga-Jamuna Press, 2006.

[15] A.J. Kerns, D.P. Shepard, J.A. Bhatti and T.E. Humphreys, "Unmanned aircraft capture and control via GPS spoofing.," *Journal of Field Robotics*, 2014.

[16] Bryce Bayer, "Color imaging array," 07 1976.

[17] Bradski, Gary and Kaehler, Adrian, *Learning OpenCV.* O'Reilly Media, 2008.

[18] Tsai, R., "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, pp. 323–344, Aug. 1987.

[19] Triggs, Bill and McLauchlan, Philip F. and Hartley, Richard I. and Fitzgibbon, Andrew W., "Bundle Adjustment - A Modern Synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, (London, UK, UK), pp. 298–372, Springer-Verlag, 2000.

[20] M. A. Lourakis and A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," *ACM Trans. Math. Software*, vol. 36, no. 1, pp. 1–30, 2009.

[21] Hai Chen and Xin-Min Wang and Yan Li, "A Survey of Autonomous Control for UAV," in *Artificial Intelligence and Computational Intelligence, 2009. AICI '09. International Conference on*, vol. 2, pp. 267–271, Nov 2009.

[22] Scaramuzza, D. and Fraundorfer, F. and Siegwart, R., "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 4293–4299, May 2009.

[23] Farneback, Gunnar, "Two-frame Motion Estimation Based on Polynomial Expansion," in *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, (Berlin, Heidelberg), pp. 363–370, Springer-Verlag, 2003.

[24] Shi, J. and Tomasi, C., "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pp. 593–600, Jun 1994.

[25] Lucas, Bruce D. and Kanade, Takeo, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, (San Francisco, CA, USA), pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.

[26] Hartley, Richard and Zisserman, Andrew, *Multiple view geometry in computer vision*. Cambridge Univ Pr, 2 ed., 2003.

[27] Denis Chekhlov, "Robust real-time visual slam using scale prediction and exemplar based feature description.," *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[28] Xiaojing Song and Seneviratne, L.D. and Althoefer, K. and Zibin Song and Zweiri, Y.H., "Visual Odometry for Velocity Estimation of UGVs," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pp. 1611–1616, Aug 2007.

[29] Fischler, Martin A. and Bolles, Robert C., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[30] Dissanayake, M. W M G and Newman, P. and Clark, S. and Durrant-Whyte, H.F. and Csorba, M., “A solution to the simultaneous localization and map building (SLAM) problem,” *Robotics and Automation, IEEE Transactions on*, vol. 17, pp. 229–241, Jun 2001.

[31] Yang, P., “Efficient particle filter algorithm for ultrasonic sensor-based 2D range-only simultaneous localisation and mapping application,” *Wireless Sensor Systems, IET*, vol. 2, pp. 394–401, December 2012.

[32] Steder, B. and Grisetti, G. and Stachniss, C. and Burgard, W., “Visual SLAM for Flying Vehicles,” *Robotics, IEEE Transactions on*, vol. 24, pp. 1088–1093, Oct 2008.

[33] Cole, D.M. and Newman, P.M., “Using laser range data for 3D SLAM in outdoor environments,” in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1556–1563, May 2006.

[34] Smith, Randall C. and Cheeseman, Peter, “On the Representation and Estimation of Spatial Uncertainty,” *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[35] R. E. Kalman and R. S. Bucy, “New results in linear filtering and prediction theory,” *Trans. ASME, Ser. D, J. Basic Eng*, p. 109, 1961.

[36] Shoudong Huang and Dissanayake, G., “Convergence analysis for extended Kalman filter based SLAM,” in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 412–417, May 2006.

[37] Ryan M. Eustice and Hanumant Singh and John J. Leonard, “Exactly sparse delayed-state filters for view-based SLAM,” *IEEE Transactions on Robotics*, vol. 22, pp. 1100–1114, 2006.

[38] Lothe, P. and Bourgeois, S. and Dekeyser, F. and Royer, E. and Dhome, M., “Towards geographical referencing of monocular SLAM reconstruction using 3D city models: Application to real-time accurate vision-based localization,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2882–2889, June 2009.

[39] Chekhlov, Denis and Pupilli, Mark and Mayol-cuevas, Walterio and Calway, Andrew, "Real-Time and Robust Monocular SLAM Using Predictive Multi-resolution Descriptors," *Computing*, 2006.

[40] Lim, Young-Chul and Lee, Chung-Hee and Kwon, Soon and Jung, Woo-Young, "Distance estimation algorithm for both long and short ranges based on stereo vision system," *2008 IEEE Intelligent Vehicles Symposium*, pp. 841–846, June 2008.

[41] Good, I.J., "Introduction to Cooley and Tukey (1965) An Algorithm for the Machine Calculation of Complex Fourier Series," *Breakthroughs in Statistics*, pp. 201–216, 1997.

[42] Gu, D.Y. and Zhu, C.F. and Guo, Jiang and Li, S.X. and Chang, H.X., "Vision-aided UAV navigation using GIS data," in *Vehicular Electronics and Safety (ICVES), 2010 IEEE International Conference on*, pp. 78–82, IEEE, 2010.

[43] C. Davies, W. Tompkinson, N. Donnelly, L. Gordon, and K. Cave, "Visual saliency as an aid to updating digital maps," *Computers in Human Behavior*, vol. 22, no. 4, pp. 672 – 684, 2006. Attention aware systems Special issue: Attention aware systems.

[44] Zheng, Yan-Tao and Zhao, Ming and Song, Yang and Adam, Hartwig and Buddemeier, Ulrich and Bissacco, Alessandro and Brucher, Fernando and Chua, Tat-Seng and Neven, Hartmut and Yagnik, Jay, "Tour the World: A Technical Demonstration of a Web-scale Landmark Recognition Engine," in *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, (New York, NY, USA), pp. 961–962, ACM, 2009.

[45] Itti, L. and Koch, C. and Niebur, E., "A model of saliency-based visual attention for rapid scene analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 1254–1259, Nov 1998.

[46] Steger, C., "An unbiased detector of curvilinear structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 2, pp. 113–125, 1998.

[47] Yang Cheng and Andrew E. Johnson and Larry H. Matthies and Clark F. Olson, "Optical landmark detection for spacecraft navigation," in *In Proceedings of the*

13th Annual AAS/AIAA Space Flight Mechanics Meeting, 2003.

[48] B. Leroy, G. Medioni, E. Johnson, and L. Matthies, “Crater detection for autonomous landing on asteroids,” *Image and Vision Computing*, vol. 19, no. 11, pp. 787 – 792, 2001.

[49] Butenuth, Matthias and Straub, Bernd-Michael and Heipke, Christian, “Automatic extraction of field boundaries from aerial imagery,” *KDNet Symposium on Knowledge-Based Services for the Public Sector*, pp. 3–4, 2004.

[50] McEwen, Alfred S. and Eliason, Eric M. and Bergstrom, James W. and Bridges, Nathan T. and Hansen, Candice J. and Delamere, W. Alan and Grant, John A. and Gulick, Virginia C. and Herkenhoff, Kenneth E. and Keszthelyi, Laszlo and Kirk, Randolph L. and Mellon, Michael T. and Squyres, Steven W. and Thomas, Nicolas and Weitz, Catherine M., “Mars Reconnaissance Orbiter’s High Resolution Imaging Science Experiment (HiRISE),” *Journal of Geophysical Research: Planets*, vol. 112, no. E5, pp. n/a–n/a, 2007.

[51] Evenden, Gerald and Warmerdam, Frank, “Proj.4 - Cartographic Projections Library,” *Source code and documentation available from trac.osgeo.org/proj*, 1990.

[52] Jegou, Herve and Douze, Matthijs and Schmid, Cordelia, “Hamming embedding and weak geometric consistency for large scale image search,” in *Computer Vision–ECCV 2008*, pp. 304–317, Springer, 2008.

[53] Yee, Ka-Ping and Swearingen, Kirsten and Li, Kevin and Hearst, Marti, “Faceted metadata for image search and browsing,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 401–408, ACM, 2003.

[54] Jégou, Hervé and Douze, Matthijs and Schmid, Cordelia, “Improving bag-of-features for large scale image search,” *International Journal of Computer Vision*, vol. 87, no. 3, pp. 316–336, 2010.

[55] Rogers, David J. and Tanimoto, Taffee T., “A Computer Program for Classifying Plants,” *Science*, vol. 132, no. 3434, pp. 1115–1118, 1960.

[56] James Bennett and Stan Lanning, “The netflix prize,” *Proceedings of KDD cup and workshop*, 2007.

[57] Kosinski, Michal and Stillwell, David and Graepel, Thore, “Private traits and attributes are predictable from digital records of human behavior,” *Proceedings of the National Academy of Sciences*, 2013.

[58] Lowe, David G, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[59] Wang, A., “An industrial strength audio search algorithm,” in *International Conference on Music Information Retrieval (ISMIR)*, 2003.

[60] Xue, Ling and Godden, Jeffrey W. and Stahura, Florence L. and Bajorath, Jrgen, “Design and Evaluation of a Molecular Fingerprint Involving the Transformation of Property Descriptor Values into a Binary Classification Scheme,” *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 4, pp. 1151–1157, 2003.

[61] Ugander, Johan and Karrer, Brian and Backstrom, Lars and Marlow, Cameron, “The anatomy of the Facebook social graph,” *arXiv preprint arXiv:1111.4503*, 2011.

[62] Dementhon, Daniel F and Davis, Larry S, “Model-based object pose in 25 lines of code,” *International journal of computer vision*, vol. 15, no. 1-2, pp. 123–141, 1995.

[63] Schweighofer, G. and Pinz, A., “Robust Pose Estimation from a Planar Target,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 2024–2030, Dec 2006.

[64] Moreno-Noguer, F. and Lepetit, V. and Fua, P., “Accurate Non-Iterative O(n) Solution to the PnP Problem,” in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, Oct. 2007.

[65] Nist´er, D., “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.

[66] Thorne, Brian, "Introduction to Computer Vision in Python," *The Python Papers Monograph*, vol. 1, p. 13, 2009.

[67] Owens, Mike, *The Definitive Guide to SQLite (Definitive Guide).* Berkely, CA, USA: Apress, 2006.

[68] Ross and Geiger, "Vision-Based Target Geolocation and Optimal Surveillance on an Unmanned Aerial Vehicle," *AIAA Guidance Navigation and Control Conference*, 2008.

[69] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[70] Yap, "PaDEL-descriptor: An open source software to calculate molecular descriptors and fingerprints," *Journal of Computational Chemistry, Volume 32, Issue 7*, 2011.

[71] Jackson, Donald A and Somers, Keith M and Harvey, Harold H, "Similarity coefficients: measures of co-occurrence and association or simply measures of occurrence?," *American Naturalist*, pp. 436–453, 1989.

[72] A. Ochiai, "Zoogeographical studies on the soleoid fishes found japan and its neighboring regions," *Bulletin of Japanese Society of Fisheries*, vol. 22, pp. 526–530, 1957.

[73] Cesetti, A. and Frontoni, E. and Mancini, A. and Zingaretti, P. and Longhi, S., "A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks," *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, U.S.A. June 810, 2009*, pp. 233–257, 2010.

[74] Heyer, L J and Kruglyak, S and Yooseph, S, "Exploring expression data: identification and analysis of coexpressed genes.," *Genome Research*, vol. 9, pp. 1106–15, Nov. 1999.

[75] Larsen, Thomas Dall and Andersen, Nils A and Ravn, Ole and Poulsen, Niels Kjølstad, "Incorporation of time delayed measurements in a discrete-time Kalman filter," in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 4, pp. 3972–3977, IEEE, 1998.

[76] Jonathan G. Koomey and Stephen Berard and Marla Sanchez and Henry Wong, "Implications of Historical Trends in the Electrical Efficiency of Computing," *IEEE Annals of the History of Computing*, vol. 33, no. 3, pp. 46–54, 2011.

[77] Luebke, David and Harris, Mark and Govindaraju, Naga and Lefohn, Aaron and Houston, Mike and Owens, John and Segal, Mark and Papakipos, Matthew and Buck, Ian, "GPGPU: General-purpose Computation on Graphics Hardware," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, (New York, NY, USA), ACM, 2006.

[78] Hong, Sunpyo and Kim, Hyesoon, "An Integrated GPU Power and Performance Model," *SIGARCH Comput. Archit. News*, vol. 38, pp. 280–289, June 2010.