# SALIENCY GROUPED LANDMARKS FOR USE IN VISION-BASED SIMULTANEOUS LOCALISATION AND MAPPING

by

**Deon Joubert**

Submitted in partial fulfilment of the requirements for the degree

Master of Engineering (Electronic Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology
UNIVERSITY OF PRETORIA

December 2013

# SUMMARY

**SALIENCY GROUPED LANDMARKS FOR USE IN VISION-BASED SIMULTANEOUS LOCALISATION AND MAPPING**

by

**Deon Joubert**

| | |
|---|---|
| Supervisor(s): | Mr H. Grobler |
| Department: | Electrical, Electronic and Computer Engineering |
| University: | University of Pretoria |
| Degree: | Master of Engineering (Electronic Engineering) |
| Keywords: | Robotics, vision-based SLAM, computer vision, landmark extraction, data association, landmark management, saliency, Kinect, 3D dataset |

The effective application of mobile robotics requires that robots be able to perform tasks with an extended degree of autonomy. Simultaneous localisation and mapping (SLAM) aids automation by providing a robot with the means of exploring an unknown environment while being able to position itself within this environment. Vision-based SLAM benefits from the large amounts of data produced by cameras but requires intensive processing of these data to obtain useful information. In this dissertation it is proposed that, as the saliency content of an image distils a large amount of the information present, it can be used to benefit vision-based SLAM implementations.

The proposal is investigated by developing a new landmark for use in SLAM. Image keypoints are grouped together according to the saliency content of an image to form the new landmark. A SLAM system utilising this new landmark is implemented in order to demonstrate the viability of using the landmark. The landmark extraction, data filtering and data association routines necessary to make use of the landmark are discussed in detail. A Microsoft Kinect is used to obtain video images as well as 3D information of a viewed scene. The system is evaluated using computer simulations and real-world datasets from indoor structured environments. The datasets used are both newly generated and freely available benchmarking ones.

# OPSOMMING

## OPVALLENDHEIDSGEGROEPEERDE BAKENS VIR GEBRUIK IN VISIEGEBASEERDE GELYKTYDIGE KARTERING EN LOKALISERING

deur

**Deon Joubert**

| | |
|---|---|
| Studieleier(s): | Mnr H. Grobler |
| Departement: | Elektriese, Elektroniese en Rekenaar-Ingenieurswese |
| Universiteit: | Universiteit van Pretoria |
| Graad: | Magister in Ingenieurswese (Elektroniese Ingenieurswese) |
| Sleutelwoorde: | Robotika, visie-gebaseerde GKL, rekenaarvisie, bakenontginning, data-assosiasie, bakenbestuur, opvallendheid, Kinect, 3D datastel |

Die effektiewe toepassing van mobiele robotika vereis dat robotte take moet kan uitvoer met ’n uitgebreide mate van outonomie. Gelyktydige kartering en lokalisering (GKL) ondersteun automatisering deur aan robotte die vermoë te verskaf om ’n onbekende omgewing te kan verken terwyl die robot daartoe in staat is om te lokaliseer binne hierdie omgewing. Visie-gebaseerde GKL trek voordeel uit die groot hoeveelhede data wat deur kameras gegenereer word, maar vereis intensiewe verwerking van hierdie data om bruikbare inligting te verkry. In hierdie verhandeling word voorgestel dat, omdat die opvallende inhoud van ’n beeld baie van die beskikbare inligting saamvat, dit gebruik kan word om GKL-stelsels wat visuele data gebruik, te bevoordeel.

Die voorstel word ondersoek deur ’n nuwe baken te ontwikkel vir gebruik in GKL. Kernpunte in ’n beeld word saamgegroepeer volgens die opvallende inhoud van ’n beeld om die bakens te vorm. ’n GKL-stelsel wat gebruik maak van hierdie nuwe baken word geïmplementeer ten einde die lewensvatbaarheid van die gebruik van die baken te demonstreer. Die bakenontginning, datafilter en data-assosiasieroetines wat nodig is om gebruik te maak van die baken word in detail bespreek. ’n Microsoft Kinect word gebruik as sensor om videobeelde sowel as 3D-inligting van ’n besigtigde toneel te verkry. Die stelsel is geëvalueer met behulp van rekenaarsimulasies en werklike-wêrelddatastelle van binnenshuise gestruktureerde omgewings. Die datastelle wat gebruik word, bestaan uit beide nuut gegenereerde en vrylik beskikbaar toetsdatastelle.

# ACKNOWLEDGEMENTS

I would like to express my gratitude to the following people:

- My supervisor, Hans Grobler - through his advice and guidance I have become a better researcher and engineer. Thank you for keeping me on track.
- My family - for their endless encouragement and support. Thank you for being there when the nights were long.
- My grandfather, Prof. W. H. Gerneke - for showing me the way.
- My friends - for remembering me when I forgot myself.
- My colleagues - for your advice, support and patience.

The work contained in this dissertation has been funded in part by the Technology and Human Resources Programme of the National Research Foundation and in part by the Council for Scientific and Industrial Research. I am grateful for the trust and faith that these organisations have placed in me.

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ATE | Absolute trajectory error |
| DFT | Discrete Fourier transform |
| DoF | Degrees of freedom |
| DoG | Difference of Gaussian |
| EIF | Extended information filter |
| EKF | Extended Kalman filter |
| FAST | Features from accelerated segment test |
| GL | Grouped landmark |
| GPS | Global positioning system |
| IMU | Inertial measurement unit |
| IF | Infrared |
| LTS | Long term support |
| NARF | Normally aligned radial features |
| NEES | Normalised estimation error-squared |
| PCA | Principal component analysis |
| PCL | Point cloud library |
| PFT | Phase Fourier transform |
| PQFT | Phase Quaternion Fourier transform |
| QFT | Quaternion Fourier transform |
| ROS | Robot operating system |
| RMSE | Root mean square error |
| SIFT | Scale-invariant feature transform |
| SL | Singular landmark |
| SLAM | Simultaneous localisation and mapping |
| SPmodel | Symmetries and perturbations model |
| SR | Spectral residual |
| Std dev | Standard deviation |
| SURF | Speeded up robust features |
| ToF | Time-of-flight |

# TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

In the first quarter of 2012 it was announced that Amazon.com would purchase Kiva Systems Incorporated for $775 million. Kiva Systems is a company that specialises in improving the efficiency of distribution centres through the use of multitudes of mobile, autonomous robots [1]. The transaction was by far one of the largest investments by a commercial entity into the recently emerged research field of service robotics. The investment signified a change in how robots were perceived. Before, robotics had been seen as an isolated science, restricted to research laboratories or assembly lines. The research field was now viewed as dynamic and maturing, with real-world applications for the present. Furthermore, the transaction highlighted the potential value of the innovative implementation of extensively automated robots.

Providing mobile robots with the capacity to operate autonomously is a difficult and extensive problem. The research described in this dissertation investigates how visual data and specifically the concept of saliency could be used advantageously in the automation of mobile robots. The context of this problem is presented in this chapter, as well as the specific focus for which a solution was formulated. The approach used to develop a solution is also presented. Finally, the condensed layout of the rest of the dissertation is given.

## 1.1 PROBLEM STATEMENT

### 1.1.1 Context of the problem

Robotic tasks are often defined as dull, dangerous or dirty. Robots can operate in environments which would be hazardous or even lethal to humans, such as the depths of the ocean or on the surface of

Mars. Robots are capable of tirelessly performing the same, repetitive task with a far lower rate of failure than humans. Industrial robotics have played a significant role in improving manufacturing and have turned factories into efficient, adaptable and safe production facilities. However, these facilities still require that the operational space of the robots be highly restricted and controlled, with most of the robots statically mounted. For robots to develop and become even more useful, it is necessary to endow them with the capacity to operate alongside humans in the dynamic and uncontrolled environments humans inhabit. In short, robots need to become more autonomous and capable of self-navigation.

One of the requirements for the realisation of complete autonomy in mobile robotics is the capability of robots to determine their position within an environment. Global positioning satellite (GPS) systems have been used for such localisation, however a GPS signal can be lost or be completely unavailable [2], such as in an underground or indoor environment. Dead reckoning systems, based on inertial measurement units (IMUs), encoders, visual odometry or registration techniques, have been used to overcome this problem, but suffer from drift due to error accumulation [3]. A way in which to overcome this problem is to record a history of what a robot has sensed within its environment and to localise the robot in terms of this record. A known map of an environment would simplify the application of this method. Unfortunately, creating such a map for use by a robot is time-consuming or impractical in cases where the robot must explore an unknown environment. Therefore autonomous robotics requires that the problem of simultaneous localisation and mapping (SLAM) be solved.

The identification of unique points of interest in an environment, or landmarks, from sensory data enables a robot to create a concise, short-hand map of the environment. The first tractable solution to the problem of SLAM is presented in [4], where the positions of the robot and landmarks are described using a probabilistic framework. The correlation between the estimates of the positions are quantified through the use of a covariance matrix. Estimates of the landmark and robot positions can thus be refined as the robot continues to move and gain more information about its environment.

To implement SLAM, a robot must have access to a sensor that produces data which relates to the environment. Visual sensors are well suited to robotic applications in view of their relative low cost, low weight and the high rate of data that cameras can generate. The disadvantage of these sensors is the large amount of processing that is required to convert the data into useful information.

### 1.1.2 Research opportunity

As will be seen in Chapter 2, much research has been conducted with regard to the development of vision-based SLAM systems. However, most of these approaches are limited to using landmarks based on point features and choose to filter out vast amounts of data that are present in images. Information sources such as colour, edge composition, texture variance and the relationship between pixels all influence how humans identify what regions of an image are important. If the ability of humans to effortlessly identify such salient regions within an image, which can be seen as a distillation of all these information sources, could be synthesised by robots, it could lead to an improvement in the application of SLAM and thus in the automation of robots.

## 1.2 RESEARCH OBJECTIVE AND APPROACH

The research objective was to investigate how saliency, a concept which arises naturally from an image for human beings, could be used to realise a vision-based SLAM system for use in different types of robots. Several questions were raised by this objective:

- How should the saliency of an image be determined in an efficient manner?
- How can the definition of a landmark make use of the salient content of an image?
- How should the landmark-handling routines of a SLAM system be implemented?
- Identifying regions allows for the use of groups of features as landmarks. Would this be more beneficial than using single features?
- How should the implemented SLAM systems be evaluated to determine whether an improvement was realised?

The approach that was used to answer these questions was to implement a general-purpose SLAM system, using a Microsoft Kinect, and then to define a landmark and the landmark-handling modules based on this system. The Kinect was used, not as a point cloud generating device, but as a means to relate visual data directly to 3D spatial information. The use of the Kinect allowed the focus of the landmark development to be the use of saliency as opposed to the inference of 3D information.

## 1.3 RESEARCH CONTRIBUTION

The most important contribution was the new methodology developed for the extraction of reliable, differentiable landmarks from visual data, thereby improving robot localisation from a map within a local window. The core idea of this new method was that by grouping individually sensed image features into larger landmarks the system would be less affected by the observational noise of each feature. Features were grouped according to a saliency map of the video image, which indicated regions in an environment that were rich in visual features. If a static environment could be assumed, these salient regions would also remain consistent.

Another contribution was the new dataset that was generated for evaluation purposes. The new dataset can in future be used as a comparative test bench for new SLAM implementations. The dataset contains data from several sequences where the Kinect was moved along a specific path a number of times. The data comprises camera images from the Kinect together with the ground truth positions of specific positions along the path. A short study on SLAM evaluation methods is presented, which can guide future researchers in defining test protocols.

## 1.4 OVERVIEW

A literature study was conducted to investigate the current state of research of the SLAM problem, with special attention given to vision-based solutions. A summation of this study is given in Chapter 2. The pre-existing theory used to implement the SLAM system and particularly the saliency detection methods is presented in Chapter 3. The development of the landmark and the components required to implement the SLAM system that used this landmark is shown in Chapter 4, which also contains details on how the Kinect sensor was used and the software environment used. The complete SLAM system was tested to determine the accuracy and success of the implementation. The experimental methodology, results and interpretation of these results are detailed in Chapter 5. Chapter 6 contains a summary discussion of the system, system limitations and ideas on how the work can be improved upon and extended.

# CHAPTER 2

# LITERATURE STUDY

## 2.1 CHAPTER OVERVIEW

In this chapter the development of SLAM to become an almost ubiquitous solution to the problem of robot localisation in an unknown environment is shown. First the fundamental difficulty of solving SLAM is discussed, after which different approaches to solving SLAM are investigated, highlighting the use of visual data where appropriate. Thereafter the way in which the intended operational environment and available robot vehicle and sensors influence how SLAM is implemented is discussed. Finally, the use of features in visual SLAM is investigated, as well as how saliency detection can be incorporated into a vision-based SLAM system.

## 2.2 APPROACHES TO SOLVING THE SLAM PROBLEM

As has been argued in the introduction, the localisation of a mobile robot and mapping of the surrounding environment are two problems that must be solved to realise independent robot operation. Each of these problems on its own can be solved given the solution to the other. In a known environment a robot can be localised by determining the relative position of the robot with regards to the known positions of landmarks within the environment. Conversely, if the position of the robot is known, a map can be created of the surroundings by determining landmark positions relative to the robot. Each of these scenarios limits the scope in which robots can be used to environments that have been mapped beforehand or where exact localisation information is continuously available. Therefore the two problems must be solved simultaneously. Two common approaches to solve SLAM is to use either landmarks or scan-matching.

### 2.2.1 Landmark-based SLAM

The classical approach to implementing SLAM [4] is to use an estimation algorithm to keep track of the pose of the robot together with the sensed positions of environmental features, or landmarks, in a state vector. With landmark-based SLAM the accurate detection of correspondences between newly extracted landmarks and previously mapped landmarks is vital to the success of the implementation. Landmark-based SLAM requires the reliable identification of landmarks within a large set of data and the accurate matching of newly detected landmarks to previously mapped landmarks. The benefit of this method is that the memory and computational requirements of the core SLAM estimator are decreased as the state vector does not have to encompass all of the sensed environmental data. On the other hand, preprocessing of the data is required to extract the landmarks, which adds to the overall computational expense of the system. Another disadvantage of landmark-based SLAM is that it is more difficult to implement in an outdoor environment as it is more difficult to find suitable landmarks. The extended Kalman filter (EKF) framework is commonly used as estimation algorithm in landmark-based SLAM [5].

The assumption that the error of the pose and landmark estimates will diverge over time reduced the expected applicability of landmark-based SLAM. In [6] it is proven that the positional estimates will converge to a lower bound error dependent on the error of the first observation as landmarks are re-observed and new landmarks are detected. The formulated proof, albeit for the restricted case of a two-dimensional vehicle, has guaranteed a continued interest from the research community in SLAM as a vital component in the overall automation of robots.

### 2.2.2 Scan-matching SLAM

An alternative approach to landmark-based SLAM is trajectory tracking or scan-matching SLAM [7], where the state vector consists of the current and past poses of the robot. Each recorded pose has an associated set of sensory data. These data scans are typically 2D laser scans or 3D point clouds. The scans are then matched and merged to form the required map of the environment. The matching process, or registration, provides an approximation of the robot pose. The best estimate of the positions that generated the sensor inputs then forms a non-linear optimisation problem [8].

An important part of registration is first to determine the corresponding points in the data scans that are to be matched. The most commonly used registration algorithm is iterative closest point (ICP)

[9] where correspondences are found using nearest neighbour matching. If an initial estimate of the pose approximation to be determined can be provided, the computation time and pose error can be decreased. In [10] the 3D coordinates of visual features, matched using a normalised cross-correlation algorithm, were used to produce such an initial estimate. In [11] a stereo camera system was used instead of a laser scanner to produce a 3D point cloud. Each 3D point was easily identifiable not only by its spatial position but also by a visual descriptor. A descriptor can be viewed as the condensing of the local image information surrounding a pixel. As can be seen, the use of visual information can be beneficial to finding a solution to a 3D spatial problem.

The grouping of singular data points into larger, more complex forms, is investigated in [12]. Here singular points were grouped into planar patches and used in a planar-adapted ICP. The advantage of this approach is that the size of the data is reduced (from 100 000 points to 150 planes, on average) and that more information, namely the orientation of the plane, is available to identify each data entry.

### 2.2.3 Combination of approaches

The use of particle filters to track multiple hypotheses of possible robot trajectories together with associated mapped landmarks can be interpreted as a combination of the scan-matching and landmark-based SLAM approaches. Much success has been achieved by using a Rao-Blackwellized particle filter, as seen in the FastSLAM implementation [13]. Even though this method is more robust against incorrect matching of landmarks, such errors should still be avoided. The method can still benefit from improved landmark formulation.

A recent implementation [14] can also, in a simplified manner, be seen as a combination of scan-matching and landmark-based SLAM approaches. In this implementation the overall motion was separated into smaller local windows, where landmarks were detected and matched to form the best estimate within each window. The motion between windows then formed a separate estimation problem to solve.

For the SLAM implementation that was developed as part of this research, landmark-based SLAM was used, as the impact of the developed landmark was more apparent than with scan-matching SLAM. However, many of the methods and routines that have been developed, be it the grouping of features into more complex landmarks or the use of visual feature descriptors, can be applied to aid in finding correspondences for the registration process on which scan-matching SLAM is reliant, or any combination of approaches.

## 2.3 SLAM IMPLEMENTATION CONSIDERATIONS

The development of a new landmark is dependent on the SLAM algorithm to be used. For the design of the algorithm a number of factors need to be considered, such as the type of robot to be used, the sensors available to the robot and the environment in which the robot will operate. The choice of estimation algorithm that is used is also influenced by these factors.

### 2.3.1 Environment

SLAM was first developed for application in a highly controlled indoor environment. As the capabilities of SLAM were expanded, it was implemented in vehicles intended for outdoor environments [2, 3, 15]. Outdoor environments usually require a greater area to be mapped, which entails an increase in the number of landmarks to be processed. More landmarks leads to an increase in computation expense and memory usage. Outdoor landmarks are also harder to detect because the environment is less structured and distinct. A landmark for such an environment must be less dependent on geometric features and more adaptable to changing scenery. Although the developed system was not designed for an outdoor environment, it was deemed useful to include such considerations in the design of the landmark to promote future applications of it.

### 2.3.2 Vehicle type

SLAM has been implemented in the navigation and automation of various types of robotic vehicles. These vehicles include ground-based wheeled vehicles [15], small fixed-wing unmanned aerial vehicles (UAVs) [2, 16], rotor-wing UAVs [3], autonomous underwater vehicles [17] and blimps [18]. In [19] a vision-based SLAM implementation was tested on a humanoid robot, though it was not specifically designed for it. The implementation in [19] is interesting because the system utilised a constant velocity movement model, which is not reliant on any additional sensors other than the camera used. The system can therefore easily be adapted for use by any type of robot.

### 2.3.3 Sensors

Irrespective of the approach used, SLAM relies on the establishment of correspondences between sets of data produced by exteroceptive sensor. The manner in which these correspondences are obtained is dependent on the type of environmental sensory information a robot receives. Some sensors that have been used in SLAM implementations are laser scanners [5], sonar sensors [20], infrared (IR) cameras [21] and time-of-flight (ToF) cameras [22].

Standard cameras are also used to implement SLAM. Vision-based SLAM systems requires the large amount of data produced by cameras to be processed into usable information. The extraction of the landmark depth information from images is an especially challenging problem that requires an accurate and efficient solution. Two methods are typically used to obtain depth data from visual data, monocular vision [19] and stereo vision [23].

A single image from a camera only provides bearing data and cannot be used to provide depth information of landmarks [16]. Monocular depth perception uses a sequence of images from a single camera and the concurrent vehicle movement to determine the distance to a landmark. A very effective implementation of monocular vision is shown in [19] where an active vision approach was used with a mobile camera head. Stereo vision overcomes the inability of a single camera to provide immediate depth information. The method uses two or more cameras with an offset in position and angle to provide depth information of detected features. Stereo vision is a simpler process, as the offset is fixed and robot positional information need not be included to determine the offset in images. A system that used an active vision approach while using a stereo vision camera configuration is shown in [24].

The most direct way in which to obtain 3D data is to use a 3D laser scanner [25]. Such scanners that are capable of capturing 3D data in a single scan are usually heavy and expensive. An alternative is to mount a 2D laser scanner onto an actuator and then to combine a number of 2D scans into a point cloud [5, 12, 26]. Stereo vision usually produces a cloud that is not as dense as that of a laser scanner because only points of interest within an image are incorporated into a point cloud.

With the release of the Microsoft Kinect [27] an inexpensive, fast 3D sensor is available for use in robotics. The Kinect, shown in Figure 2.1, can be viewed as a combination of a 3D sensor and a standard camera. The Kinect consists of a normal colour camera, an IR laser and IR receiver. The

Kinect extracts depth data from a scene by projecting a set pattern into the scene using the IR laser. The receiver then records the reflected pattern and from the deformation in the pattern the Kinect is then able to determine the spatial distribution of the points within the environment [28]. The Kinect is able to produce data at a rate of 30 frames per second with a spatial resolution of 3 mm and a depth resolution of 10 mm. One of the limitations that the Kinect suffers from is that depth measurements are only accurate within a small range band, between 0.8 m and 3.5 m. The Kinect has already been used by a number of visual odometry systems [29] and in the RGB-D SLAM implementation [30].

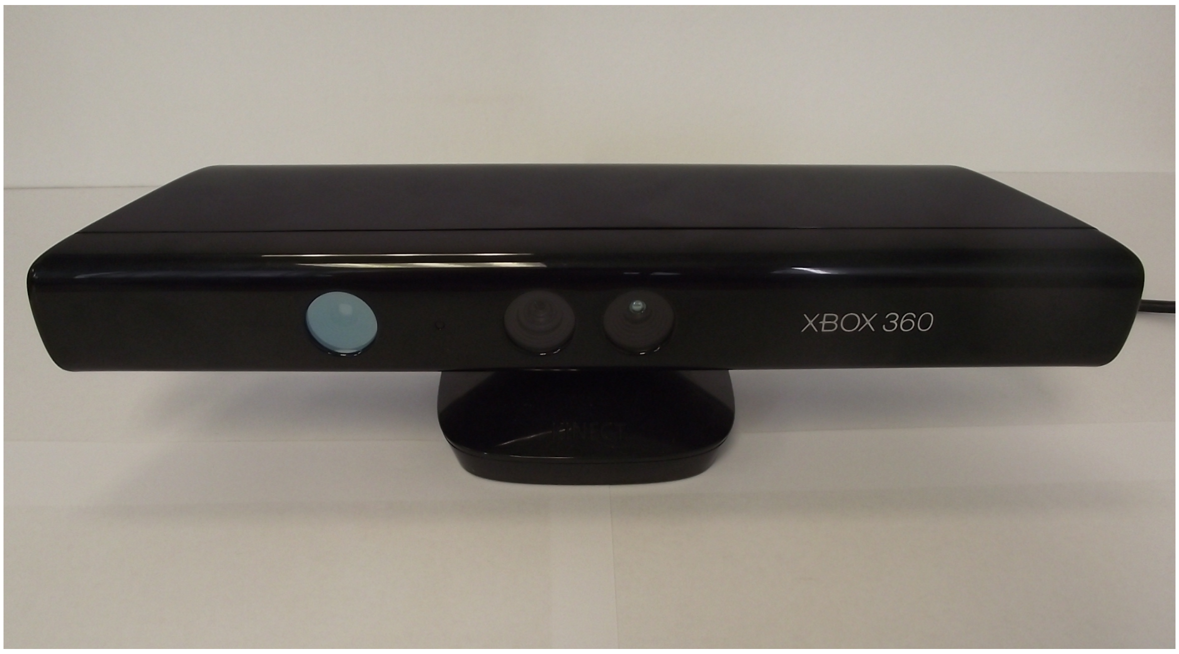


**Figure 2.1:** Microsoft Kinect. The first circular recess on the left houses the IR laser, the second recess the standard video camera and the third the IR receiver.

Robot systems are often equipped with proprioceptive sensors that provide information on the condition of the robot itself. These sensors can be used by a SLAM algorithm to improve the estimation accuracy further. Proprioceptive sensors that have been used include GPS receivers [31], IMUs, barometers [3] and wheel encoders. The disadvantage of SLAM systems using these sensors is that the systems are restricted to robots equipped with these sensors and to environments in which these sensors can function; GPS is not available underground and tracked (tank-tread) robots do not provide useful odometry data.

### 2.3.4 Estimation algorithm

A robot will typically have access to several sources of positional information: the various sensors, the map being generated and various mathematical models of the robot. The combination of all of these sources using a probabilistic filter is advantageous and often required because the filter can then be used to determine the best positional estimates of the robot and landmarks. There are various types of probabilistic filters that can be used.

The EKF has been used in many implementations and in many variations [15, 16, 18, 32]. The filter has become a standard in the SLAM research community. Using it in the development of the SLAM system has allowed more attention to be paid to the design of the landmark. The greatest drawback of the EKF is the inherent inaccuracy due to the use of linearised dynamic and observation models. Another disadvantage is the quadratic increase in computational expense as more landmarks are included in the map. The latter problem can be overcome by using efficient map management methods, reducing the number of landmarks processed for each cycle.

As more accurate positional estimates were desired, researchers began to investigate the use of particle filters in SLAM. Particle filters are able to process more complicated observation and dynamic models. As has been mentioned, a popular particle filter based algorithm is the factored solution to SLAM, or FastSLAM [13]. One of the advantages of this algorithm is that the computational expense does not increase as much as EKF SLAM as the number of landmarks increased. The disadvantage of FastSLAM is that the positional uncertainty is underestimated as time elapses. In [33] it is shown that FastSLAM is not consistent in the long term owing to the necessary deletion of particles and the resultant loss of historical pose information. However, the implementation is regarded as useful in the short term as part of a larger navigational system.

There are also many other estimation algorithms that have are used to solve SLAM. One of these is the extended information filter (EIF) [17]. The use of the filter produces a naturally sparse information matrix, which eases the computational and memory problems associated with SLAM implementations. The covariance matrix is not explicitly used but is incorporated in the information matrix. A problem of the EIF is that recovery of the covariance matrix, often used for data association and other purposes, is cumbersome and computationally expensive. Another interesting approach is the use of an artificial neural network, based on the computational model of a rodent hippocampus, as an estimation algorithm in [34]. The algorithm was shown to map an entire suburb effectively.

## 2.4 LANDMARK DEFINITION AND EXTRACTION

In most of the literature pertaining to SLAM the terms feature and landmark are used interchangeably because a landmark is usually defined as a singular detected feature. As interest points within an image are also referred to as features in computer vision terminology, it was decided not to equate these terms in this dissertation, as this can lead to ambiguity. The term feature is used to refer to visual points of interest and the term landmark is used to refer to the already processed spatial point or region mapped by the SLAM system.

Landmark extraction refers to the process in which features are detected in large batches of sensory data and processed for use in SLAM. Examples of features often used include interesting and unique points, or keypoints, in images or corners in laser scans. Data association of landmarks refers to the process through which the correspondence between previously mapped landmarks and newly detected landmarks is found. The process can involve using information on the underlying features and the distance between computed landmark spatial positions. Accurate data association enables a SLAM system to correct any drift that has accumulated as a result of previous estimation errors. Sometimes the drift is too large (because of a very large distance travelled) and data association fails. The described scenario is referred to as the loop closure problem. In such cases additional methods are used to find correspondences. Although vision has been used to accomplish this goal [35], it was decided to restrict the implemented system to only methods based on the developed landmark.

The landmark definition determines how both landmark extraction and data association can be conducted. As a Kinect is a source of both visual and 3D data, the rest of this section pertains to landmarks extracted from these types of data.

### 2.4.1 Landmarks from 3D data

There are a number of geometric shapes that have been used as landmarks in 3D SLAM. Of these shapes, planes have been used the most often [5, 36]. The symmetries and perturbations model (SPmodel) is a structure that enables the use of multiple types of landmarks in the same state vector [37]. The SPmodel was used in [38] to manage lines, in [5] to track planes and in [39] to keep track of planes, spheres and cylinders. A very similar structure to SPmodels was used in [40] where landmarks, initially sensed as points, were grouped into more complex geometric structures. The system extracted line and plane structures from the detected landmark points. The points that formed part of each structure were replaced with a reduced coordinate that described their position within that structure: a 1D coordinate if the point was part of a line or a 2D vector if it was part of a plane. The advantage of grouping landmarks in this way is that the cross-correlation of the landmarks is increased while the size of the state vector is reduced.

When fitting data to predefined geometric models a fitting bias is introduced into the system [41]. Furthermore, points are ignored that do not fit the model but that can provide useful information on the environment. A landmark-based approach that avoided using geometric models is presented in [41]. Locally interesting or salient regions were identified based on the entropy, change in statistics and the size of all regions. Clumps of data that surrounded these regions were used as landmarks. In this manner the problems of a geometric model were avoided while a repeatable and uniquely identifiable feature was still defined.

### 2.4.2 Feature detection for 3D data

The Kinect presents the opportunity to combine visual feature detection techniques and 3D point cloud data in innovative ways. The combination of different fields of research is desirable, as the experiences obtained in computer vision can be transferred to the relatively new field of 3D point cloud processing. In [42] 2D corners were detected in an intensity image generated by a ToF camera for SLAM. Corners at the edge of occlusion boundaries that would have led to inconsistent landmarks were then detected and discounted. Various other possibilities have also been investigated. In [43] a point cloud was compressed into a range image, which allowd for the detection of normally aligned radial features. Regional point descriptors have also been used, which are feature descriptors that are calculated directly from the 3D data [44]. In [36] the extracted planar patches were large enough to allow 2D computer vision techniques to be carried out on these patches.

### 2.4.3 Landmarks from visual data

Generally speaking, vision-based SLAM uses computer vision techniques to find distinguishable features in an image that are then tracked in landmark-based SLAM or are used as correspondence points in scan-matching SLAM [11]. The benefit of identifying interest points (or shapes) within large sets of data is that the memory and computational requirements of subsequent processes, such as the estimation algorithm, are decreased owing to the reduced amount of data that needs to be processed. However, to extract the features requires additional processing of the data, which adds to the overall computational expense of the system. Visual features that have been used as landmarks in SLAM implementations include the Harris corner [3, 18, 32], scale-invariant feature transform (SIFT) [21], speeded up robust features (SURF) [17] and Lucas-Kanade optical flow [45].

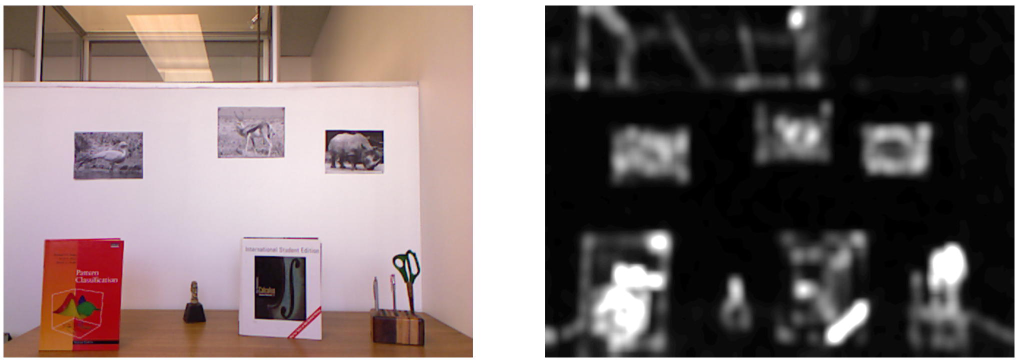
For most of the methods mentioned above the landmarks consist of only those visual features and are extracted using the standard detection algorithms for those features. In [19] the landmark was augmented by additional extraction of an image patch centred on a detected corner. The positional estimates of the features were then used to predict where the corner would be visible in subsequent frames. An image patch of the expected corner position was then transformed using the positional information of the robot to maximise the chances of matching the previously mapped landmark. The described method and results in [19] show that there is benefit in using more complex landmarks.

As SLAM is a computationally expensive process, reducing the number of landmarks that need to be handled will often lead to a reduction in computation time and an improvement in the accuracy of the estimates. For vision-based SLAM implementation, point features can be grouped together to make use of the geometric relationships between the features to improve feature detection, matching and tracking. In [46] groups of SIFT keypoints called fingerprints were used as a way to identify sub-maps in a global map, which improved mapping efficiency. Groups of multiscale Harris corners were matched in [18] and more corners were matched based on the predicted position of these corners relative to the matched group. Groups of visual features were used in [47] and [48] to identify and recognise objects used as landmarks in SLAM. Object recognition requires the isolation of regions possibly containing objects and in [48] saliency detection was used to find such regions.

## 2.5 SALIENCY DETECTION

When presented with an image, humans are typically capable of automatically and quickly identifying people, objects or shapes that stand out from the rest of the image. The quality of regions within an image to draw the attention of a viewer naturally is defined as saliency. The isolation of salient regions within a viewed scene aids humans when dealing with the large amount of information present by restricting the area on which higher-level cognitive processes need to be enacted [49]. Saliency detection can also help robots utilise visual sensors more efficiently in a similar fashion. Furthermore, having robots respond to the same visual cues as humans can facilitate better interaction between robots and humans [50].

The saliency content of an image is often represented as a saliency map. The map indicates the spatial location of salient regions within an image. An example can be seen in Figure 2.2. A taxonomy of the saliency methods reviewed can be seen in Figure 2.3. In general, there are two approaches to computing a saliency map. The first is the bottom-up approach, where regions are identified as conspicuous because of the unusual (in a local sense) occurrence of low complexity image characteristics, for example colours and edges. The second is the top-down approach where an image is analysed and saliency is assigned according to criteria based on the intended application. Frequency-based approaches can be seen as an extension of the bottom-up approach, as these methods still aim to implement task-independent saliency detection.



**(a)** **(b)**

**Figure 2.2:** (a) Input image. (b) Saliency map. The map indicates the location of salient regions within an image, with brighter regions having a higher saliency value.

### 2.5.1 Top-down approaches

An example of top-down saliency detection can be seen in [51], where the intended application was the detection of faces or hands. The purpose of the saliency map was to aid in the isolation of regions most likely to contain the type of object. Principal component analysis (PCA) was applied to a large dataset of examples of the target object class to obtain a feature vector best describing the class. Regions within the image were then defined as salient based on probabilistic density estimates computed using the feature vector. A saliency map showing the possible location of people was generated in [52], also using a feature vector obtained through PCA. Here the visual context of a scene, or where people were most likely to be found within an image, was used to improve the results of a local or bottom-up saliency detection method. In both [51] and [52] the aim was the detection of a class of object. Landmarks defined as belonging to a specific object class will restrict

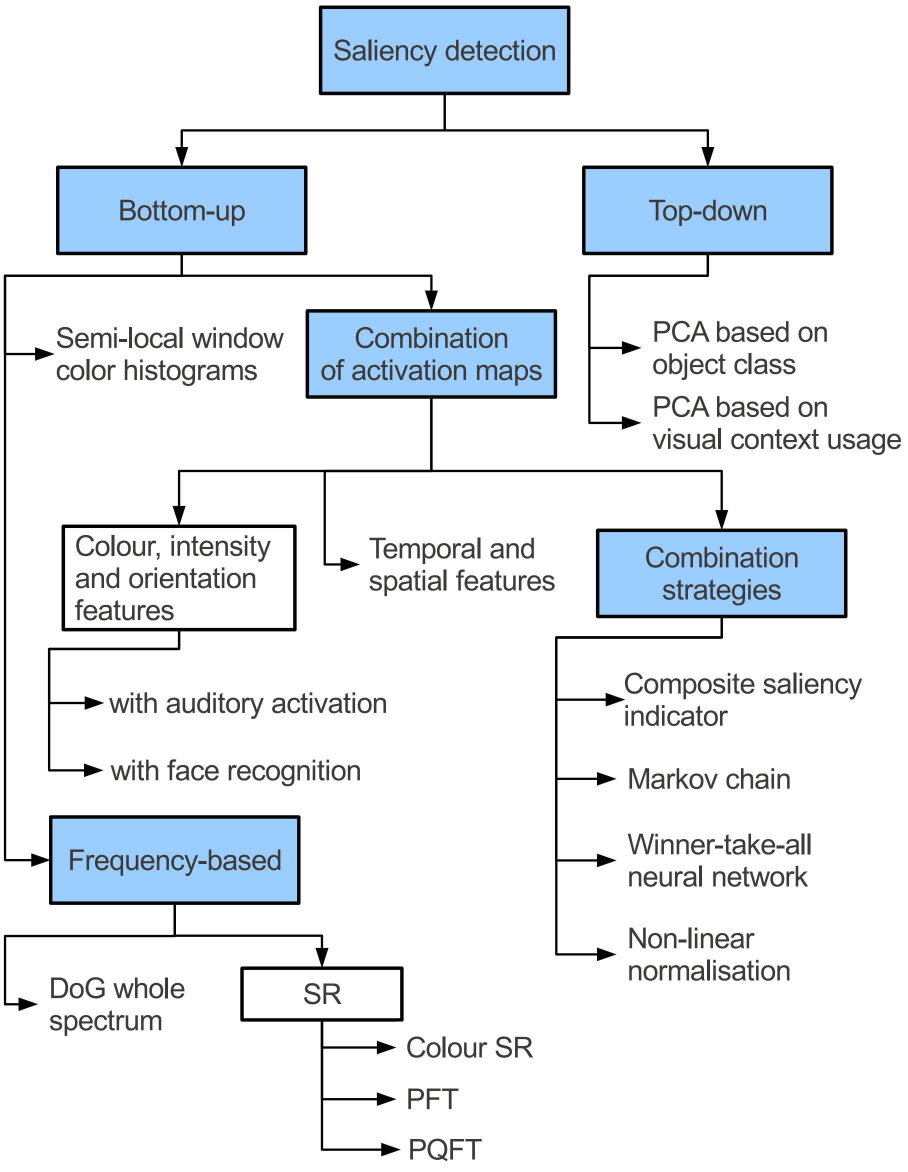

Saliency detection
Bottom-up
Top-down
Semi-local window color histograms
Combination of activation maps
PCA based on object class
PCA based on visual context usage
Colour, intensity and orientation features
Temporal and spatial features
Combination strategies
with auditory activation
with face recognition
Composite saliency indicator
Markov chain
Winner-take-all neural network
Non-linear normalisation
Frequency-based
DoG whole spectrum
SR
Colour SR
PFT
PQFT

**Figure 2.3:** Taxonomy of saliency methods reviewed.

the accompanying SLAM implementation to areas where the type of object is commonly expected to be found.

### 2.5.2 Bottom-up approaches

Saliency detection methods based on the bottom-up approach are typically stated to be inspired by the biological study of primate vision. Primate saliency detection is often ascribed to the differences detected in the foveal (focused) and peripheral vision [53]. Such peripheral-foveal vision can be imitated by using two types of cameras [50] or by using difference of Gaussian (DoG) and Gabor filters (which operate on differences of scale) when extracting features [54]. Parallel computation [55] and combination of low-level features [53] to form a saliency map are also based on biological modelling.

Typical bottom-up saliency detection consists of three steps [55]. The first step is the extraction of low-level features and characteristics from images. The second step is the translation of these features into an activation map. The third step is the combination of several activation maps into a final saliency map. The final step is only applicable if more than one activation map is used, which is often not the case, as seen in [56], [57] and [58]. As the first two steps are almost always used in bottom-up saliency detection methods, only the presence of activation map combination is used to further classify bottom-up methods in the taxonomy shown in Figure 2.3.

In [53] activation maps based on colour, intensity and orientation were generated and then combined into a single saliency map. The saliency of each pixel was determined in [56] by evaluating a posterior probability model based on only the colour contrast between semi-local windows in an image. As only one feature was used, there was no combination step. In [59] it is argued that face detection should form a separate activation map, as humans have a natural tendency to be drawn to real or perceived faces. An auditory activation map was combined with a visual activation map in [60] to produce a saliency map used to determine the gaze direction of a robot head. Spatial and temporal features were combined in [61] to find salient regions in videos. In [48] SIFT features, contours, hue, saturation and intensity information were combined into a saliency map. Rectangular salient regions were expanded to encompass whole objects, which were then used as landmarks to complement an IR scanner grid-based SLAM implementation.

Various combination strategies are investigated in the literature. In [62] activation maps with a greater composite saliency indicator, based on the compactness and density of salient regions, were weighted more heavily before combination. Activation maps that were too dissimilar to other maps were not used at all. Markov chains were used in [55] to combine activation maps, with the similarity and saliency between pixels in different activation maps used as edge weights. A winner-takes-all neural network is proposed in [63] to combine activation maps. The resultant saliency map was used to direct the movement of a robot head. In [54] several strategies were investigated and two methods based on non-linear normalisation were found to provide the best performance while retaining generality.

There are a number of disadvantages to bottom-up saliency detection. Firstly, the choice and number of features to use are not directly apparent. As stated in [62], a single feature might not be robust enough while using an increasing number of features will not necessarily result in a better solution. Secondly, the extraction of features and generation of numerous activation maps can be computationally expensive. Thirdly, the optimal manner in which feature maps should be combined is not obvious, as the relative importance of features is difficult to determine.

### 2.5.3 Frequency-based approaches

An analysis of the frequency spectrum in which several saliency detection methods, most of which are bottom-up methods, operate is shown in [57]. Most of these methods are shown not to utilise the full image spectrum, resulting in saliency maps that typically identified only the edges of objects. A DoG filter was formulated with the specific aim of utilising the full spectrum and was used to produce saliency maps covering the whole area of objects viewed.

#### 2.5.3.1 SR saliency detection method

Another frequency-based approach is the spectral residual (SR) saliency detection method [58]. The method was developed as a preliminary segmentation routine for an object detector. As it was required that the object detector be expandable to new types of objects, the method could not rely on the features of any specific object class. The bottom-up approach proposed in [53] was deemed unsuitable as it was too computationally expensive and because it was dependent on the shared image features of salient regions.

The novel approach used in [58] to formulate a task-independent saliency detector was to investigate what ascribed to a region the quality of not being salient. In other words, it was attempted to determine what made a region a part of the background of an image. Images were theorised to be generated by predictable probability distributions. The information in images could then be defined as consisting of an innovative (salient) component and a redundant (background) component. The latter corresponded to the statistically invariant properties present in all images.

The logarithm of the amplitude spectrum, or log spectrum, of an image is often used in the study of the underlying statistics of scenes. In [58] the log spectra of a large number of images were studied and it was found that there existed a general trend in the average of these log spectra. The similarity between the spectra was regarded as the redundant component contained within all images, while the deviations from this general trend in each individual image were seen as part of the innovative component. If the general trend was removed from the spectrum of an image, it was postulated that the SR produced would contain the innovative or salient content of the image. As the general trend could be interpreted as an averaging of a large number of log spectra to remove the individual deviations, the redundant component was approximated by the convolution of the log spectrum of an image with a local averaging filter. To generate a saliency map, the input signal was first "reconstructed" using the original phase spectrum and the computed SR as the magnitude of the signal. Thereafter the square of the magnitude of the saliency image was blurred using a Gaussian filter to produce the final map.

A comparison between the SR saliency detection and the bottom-up approach described in [53] is shown in [58]. Saliency maps were computed for both methods for a number of images and compared to human produced saliency maps. The SR saliency detector was found to be fifteen times faster and produced maps more closely aligned to the human produced maps than the bottom-up approach. Test results from [57] also showed that the method was faster than several other bottom-up approaches.

One of the advantages of the SR saliency detection method that can be of great benefit to a landmark extraction routine is that it does not require a complex activation map combination strategy, reducing the number of parameters that need to be set. Furthermore, it is fast. For these reasons it was used in the design of an object-searching robot in [50]. The robot employed the SR method together with structure from stereo, object recognition and FastSLAM to create a map of object locations in a previously unknown environment. Usually, SR saliency detection is only applied to the intensity (grey-scale version) of an image. Here it was also applied to the colour channels of an image. In [64] it is argued that saliency should be based on both shape-distinctiveness (normally detected in intensity images) and colour-distinctiveness to improve the information content of saliency maps.

#### 2.5.3.2 PFT saliency detection

In [65] it is argued that the generation of a frequency-based saliency map as described in [58] only requires the phase spectrum of the input image. Various one dimensional waveforms were analysed to investigate this claim. Reconstruction of a waveform consisting of a single square wave pulse using only the phase spectrum of the signal showed a peak disturbance at the location of the original rising edge of the pulse. A similar reconstruction of a uniform sinusoidal waveform showed no such obvious disturbance. From these observations the deduction was made that locations within a signal that were less homogeneous in comparison to the rest of the signal resulted in greater disturbances at the same location in the phase spectrum reconstruction of the signal. The signal reconstruction method used was based on the one presented in [58].

Based on these observations, a Phase Fourier transform (PFT) saliency detection method was developed. The method was very similar to the SR method, except that the input image spectrum was normalised to set the amplitude spectrum to one so as to maintain only the phase spectrum during the reconstruction of the input image to form the saliency map [66]. In [65] a comparison of the saliency maps produced by the two methods showed that the maps are remarkably similar, with the PFT method map produced in a third of the time of the SR map. Another advantage of the PFT method was that, because it did not require the averaging filter for the approximation of the image redundant component, it had one less parameter to set.

#### 2.5.3.3 PQFT saliency detection

The final goal of the work presented in [65] was the development of a spatio-temporal saliency detector for use in the analysis of video. The detector was developed by extending the PFT method to use both the colour and motion information present in video. Image pixels were represented as quaternions composed of intensity, a motion value and two colour opponency pairings. The phase of this quaternion image was then obtained using a quaternion Fourier transform and a saliency map was generated using the same methodology as the PFT and SR methods.

A performance comparison of the SR method, the PFT method and the newly developed phase quaternion Fourier transform (PQFT) method as well as the bottom-up method developed in [53] is presented in [65]. The frequency-based methods were all faster and detected more correct objects than the bottom-up method. The PQFT was shown to be the best at detecting salient objects, but was slower than both the SR and PFT methods. However, the PQFT still outperformed the bottom-up method when white-coloured noise was added to the test images while the SR and PFT methods both failed during this test. As the last test showed, the incorporation of colour-distinctiveness into a saliency detector did improve performance, as argued in [64].

The frequency-based methods that have been discussed are ideal as a basis for a SLAM landmark extraction routine because the methods provide good region of interest extraction capability at fast speeds. The latter feature is important because, given the complexity of a SLAM system, it is desirable to reduce computational expense throughout the system. The PQFT, although slower, is still a viable candidate because of the additional robustness gained from the inclusion of colour information. However, the PQFT method had to be adapted before it could be used. The motion feature used by the PQFT method served to highlight regions that were subject to distinctive change across video frames. For landmark extraction, it was more important to find regions that remained stable between frames. Therefore the motion feature was set to zero as suggested in [65] in the application of the method.

The salient regions detected by the described frequency-based methods are stated in [58] to form proto-objects as defined in [49]. Proto-objects are defined as the combination of low-level features rapidly detected by a human viewer. Such a proto-object is not stable in space and time and is continuously replaced with newly formed proto-objects. Proto-objects are said to draw the attention of the viewer and through longer study can be perceived as solid objects and become coherent in space and time. In a similar fashion, the developed system described in Chapters 3 and 4 used a saliency detector to identify potential landmarks and through further study extracted landmarks and kept track of them using an EKF-based SLAM system.

# CHAPTER 3

# GENERAL THEORY

## 3.1 CHAPTER OVERVIEW

The approach taken to answer the research questions identified in Chapter 1 was to implement a landmark-based SLAM system using a Kinect and then to develop a new landmark based on the grouping of visual features according to the saliency content of a video image. A high-level representation of such a SLAM system can be seen in Figure 3.1. The Kinect provides the landmark extraction method with visual data. Landmarks are extracted and matched with previously mapped landmarks during data association. The matches are then evaluated by the EKF-SLAM system and an estimate of the current robot position and orientation, as well as the landmark positions, is produced. Spurious landmarks are removed in the landmark management routine.

The following chapter describes the pre-existing general theory used to implement the described vision-based SLAM system. In Section 3.2 the EKF-SLAM system based on the constant velocity model is explained. The saliency detection methods used during landmarks extraction are formulated in Section 3.3. The formulation and rationale for using the Mahalanobis distance in the data association routine is discussed in Section 3.4.

## 3.2 EKF SLAM

A Kalman filter [67] is an algorithm that produces estimates of unknown variables using a series of observed measurements and mathematical modelling of a system. An EKF is an extension of the Kalman filter for nonlinear systems. The EKF linearises the mathematical models around the current mean. When applied to the SLAM problem, the EKF estimates the position of the robot and

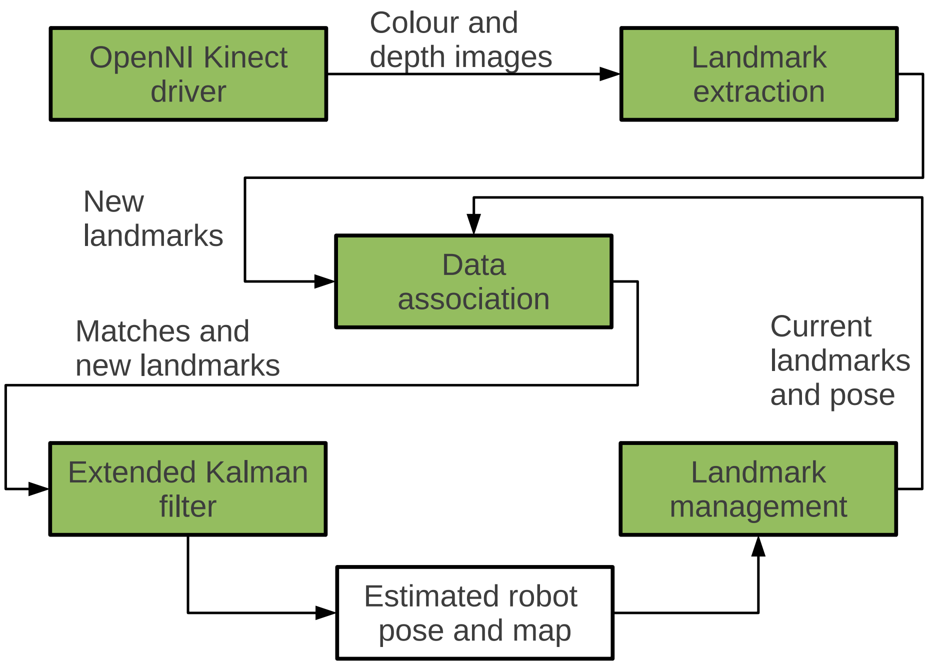


**Figure 3.1:** Block diagram of a vision-based EKF-SLAM system.

the landmarks contained in the state vector. A standard EKF consists of a prediction and an update phase, but as SLAM requires the addition of new landmarks to the map, an augmentation phase is added.

The prediction phase of the EKF computes the current position of the robot based on a mathematical motion model and the previous known position. After newly sensed landmarks have been extracted from sensor data and matched to previously mapped landmarks, the matches are then used by the EKF update phase to modify the robot pose and the mapped landmark positions. The augmentation phase adds those sensed landmarks that have not been matched to the state vector. The prediction, update and augmentation equations that are presented in this section are adopted from [19], [68] and [69].

In the formulation of equations it has been attempted to structure equations and variables according to the following convention. Subscripts in variables indicate the time step at which the variable is being evaluated. A vertical bar indicates that the variable is based on specific prior information. Superscripts indicate the frame of the variable or the coordinate axes the variable is based in. Two frames are of importance in this implementation: The robot frame $R$ and the world frame $W$. The robot frame is the coordinate system situated in the centre of the Kinect while the origin of the world frame is set to the initial robot position. A combination of superscripts shows that a variable indicates the transformation from one frame to another. For example, the notation $\hat{\mathbf{x}}^{mn}_{k|k-1}$ denotes the estimate of the transform from frame $m$ to $n$ called $\mathbf{x}$ at time step $k$ given the values of $\mathbf{x}$ at time step $k-1$, that is to say the previous time step.

### 3.2.1 State vector formulation

The state vector that is tracked is expressed as

$$\mathbf{x} = \begin{pmatrix} \mathbf{r} \\ \mathbf{m} \end{pmatrix}, \tag{3.1}$$

where $\mathbf{r}$ is the robot pose and $\mathbf{m}$ is the map of the environment, consisting of the 3D positions of the landmarks. Furthermore, the EKF also computes the covariance of the state vector $\mathbf{P}$. The covariance matrix can be seen as indicating the uncertainty of the estimates.

The robot pose is described by 13 parameters and is represented as

$$\mathbf{r} = \begin{pmatrix} \mathbf{p}^W \\ \mathbf{q}^{WR} \\ \mathbf{v}^W \\ \mathbf{w}^R \end{pmatrix}, \tag{3.2}$$

where $\mathbf{p}^W$ is the 3D position of the robot within the world frame, $\mathbf{q}^{WR}$ is a quaternion describing the rotation between the world frame to the robot frame, $\mathbf{v}^W$ is the translational velocity of the robot relative to the world frame and $\mathbf{w}^R$ is the rotational velocity of the robot relative to the frame of the robot. The last-named is expressed in compressed angle-axis form where the unit vector of $\mathbf{w}^R$ indicates the axis around which an angular rotation equal to the size of the vector is made.

### 3.2.2 Noise covariance formulation

Sensors are not perfect and will produce measurements corrupted by noise. Mathematical models are approximations of real-world operations and will suffer from inaccuracies. The EKF attempts to minimise the effect of the sensor and model uncertainties by including an estimate of these uncertainties as noise covariances into the positional computations.

The motion model noise covariances consist of the translational velocity $\mathbf{V}$ and the rotational velocity $\boldsymbol{\Omega}$ noise covariance matrices. Both are dependent on the length of the time step $\Delta t$, as the inaccuracy associated with the motion model increases with time. The noise matrices are combined into a single movement noise covariance matrix $\mathbf{R}(\Delta t)$ as

$$\mathbf{R}(\Delta t) = \begin{pmatrix} V_x\Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & V_y\Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & V_z\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & \Omega_x\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & \Omega_y\Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & \Omega_z\Delta t \end{pmatrix}. \tag{3.3}$$

The sensor noise covariance ($\mathbf{Q}$) is not dependent on $\Delta t$ and is expressed as

$$\mathbf{Q} = \begin{pmatrix} Q_x & 0 & 0 \\ 0 & Q_y & 0 \\ 0 & 0 & Q_z \end{pmatrix}. \tag{3.4}$$

### 3.2.3 Prediction phase

The prediction phase of an EKF involves the estimation of the robot pose for a specific time step using a mathematical model defining the motion of the robot as well as the previous robot pose estimate. The motion model often used when formulating vision-based SLAM systems is the constant velocity motion model [68]. In this model it is assumed that there is no control input affecting the system and that the translational and angular velocities affecting the system remain constant. Any change in the velocities is assumed to result from noise affecting the system. Given a sensor with a relatively high data rate, such as the Kinect sensor (30 Hz), and the assumption that the sensor is moved slowly and consistently, an EKF SLAM system based on this assumption can quite accurately track the overall movement, as has been shown in [42, 70, 71]. To accommodate this model, the standard EKF prediction equation was reformulated to consist of only the motion model $\mathbf{f}_r$ and the expected noise:

$$\mathbf{r}_k = \mathbf{f}_r(\mathbf{r}_{k-1}) + \mathbf{R}(\Delta t). \tag{3.5}$$

From this formulation the equations for determining the estimates, if the noise is assumed to be Gaussian and zero mean, are

$$\hat{\mathbf{r}}_{k|k-1} = \mathbf{f}_r\left(\hat{\mathbf{r}}_{k-1|k-1}\right) \tag{3.6}$$

$$\mathbf{P}_{k|k-1} = \nabla\mathbf{F}_r\mathbf{P}_{k-1|k-1}\nabla\mathbf{F}_r' + \nabla\mathbf{F}_n\mathbf{R}\nabla\mathbf{F}_n', \tag{3.7}$$

where $\nabla\mathbf{F}_r$ is the Jacobian of the movement model based on the robot pose and $\nabla\mathbf{F}_n$ is the Jacobian based on the motion model noise. Both Jacobians are resolved using the current value of $\hat{\mathbf{x}}_{k-1|k-1}$. The formulation of the Jacobian matrices is extensive and is shown in Appendix A.

The constant velocity movement model is given by

$$\mathbf{f}_r = \begin{pmatrix} \mathbf{p}^W + \left(\mathbf{v}^W + \mathbf{V}\right)\Delta t \\ \mathbf{q}^{WR} \otimes \mathbf{q}\left(\left(\mathbf{w}^R + \boldsymbol{\Omega}\right)\Delta t\right) \\ \mathbf{v}^W + \mathbf{V} \\ \mathbf{w}^R + \boldsymbol{\Omega} \end{pmatrix}. \tag{3.8}$$

As can be seen, the model is not dependent on any odometric information. Changes in the movement velocity and direction are captured in the update step where the external sensory information is processed. The model allows the EKF SLAM algorithm to be applicable to any robot vehicle using a visual sensor system. Using this model, Equation 3.6 becomes

$$\hat{\mathbf{r}}_{k|k-1} = \begin{pmatrix} \mathbf{p}^W_{k-1|k-1} + \left(\mathbf{v}^W_{k-1|k-1}\right)\Delta t \\ \mathbf{q}^{WR}_{k-1|k-1} \otimes \mathbf{q}\left(\left(\mathbf{w}^R_{k-1|k-1}\right)\Delta t\right) \\ \mathbf{v}^W_{k-1|k-1} \\ \mathbf{w}^R_{k-1|k-1} \end{pmatrix}. \tag{3.9}$$

Landmarks are assumed to remain static, therefore no new estimates are computed for the landmark positions. Furthermore, only a part of the state covariance matrix needs to be subjected to the whole of equation 3.7. The following formula from [69] is used to improve the speed of computation.

$$\mathbf{P}_{k|k-1} = \begin{pmatrix} \nabla\mathbf{F}_r\mathbf{P}_{\mathbf{rr}k-1|k-1}\nabla\mathbf{F}_r' + \nabla\mathbf{F}_n\mathbf{R}\nabla\mathbf{F}_n' & \nabla\mathbf{F}_r\mathbf{P}_{\mathbf{rm}} \\ \left(\nabla\mathbf{F}_r\mathbf{P}_{\mathbf{rm}}\right)' & \mathbf{P}_{\mathbf{mm}k-1|k-1} \end{pmatrix}, \tag{3.10}$$

where $\mathbf{P}_{\mathbf{rr}}$ indicates those covariance variables involving only the robot pose variables, $\mathbf{P}_{\mathbf{rm}}$ those variables involving the robot pose and the mapped landmarks and $\mathbf{P}_{\mathbf{mm}}$ those variables involving only the mapped landmarks.

### 3.2.4 Update phase

The motion model used does not incorporate any odometric or control information. The SLAM system is therefore heavily dependent on the processing of the sensory information so that the update phase can adapt the estimates to changes in camera movement. The standard EKF formulation to determine how an already mapped landmark is observed by the sensor is

$$\mathbf{z}_{k|k-1} = \mathbf{h}\left(\mathbf{x}_{k|k-1}\right) + \mathbf{Q}, \tag{3.11}$$

where $\mathbf{h}$ is the observation model, which describes how each mapped landmark is expected to be observed by the robot and $\mathbf{z}_{k|k-1}$ is the expected sensed landmark positions, which are based upon the prediction phase pose estimate. The observation model is dependent on the sensor used in the experiments and is therefore formulated in Section 4.5.

It is assumed that the sensor noise is Gaussian and has a zero mean. The state estimate and covariance matrix are computed using

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}\nu_k, \tag{3.12}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{KSK}', \tag{3.13}$$

for which

$$\nu_k = \mathbf{o}_k - \mathbf{z}_{k|k-1}, \tag{3.14}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{KSK}', \tag{3.15}$$

$$\mathbf{K} = \mathbf{P}_{k|k-1}\nabla\mathbf{H}_x'\mathbf{S}^{-1}, \tag{3.16}$$

$$\mathbf{S} = \mathbf{H}_x\mathbf{P}_{k|k-1}\mathbf{H}_x' + \mathbf{Q}, \tag{3.17}$$

and where $\mathbf{o}_k$ is the latest sensed and matched landmarks from the data association routine. $\mathbf{K}$ is denoted as the Kalman gain and indicates how much confidence there is in the sensed landmark positions. $\nu_k$ is the innovation or the difference between the sensed and expected landmark positions. $\mathbf{S}$ is called the innovation covariance.

$\mathbf{H}_x$ is the Jacobian of the observation model in terms of the state and is expressed as

$$\mathbf{H}_x = \frac{\delta\mathbf{h}\left(\mathbf{x}\right)}{\delta\mathbf{x}}. \tag{3.18}$$

The complete formulation is extensive and is shown in Appendix A. The Jacobian is resolved using the current value of $\hat{\mathbf{x}}_{k|k-1}$

After the covariance matrix is computed, it is transformed to ensure the symmetry of the matrix [69] according to

$$\mathbf{P} = \frac{1}{2}\left(\mathbf{P} + \mathbf{P}'\right). \tag{3.19}$$

### 3.2.5 Augmentation phase

As part of the SLAM process it is necessary for the system to incorporate new landmarks into the state and covariance matrix and thereby expand the map of the environment. Adding a landmark to the state requires the transformation of the landmark from the robot frame to the world frame using **g**, the mapping function. As the mapping function implemented is specific to the EKF-SLAM system using the Kinect, it is formulated in Section 4.5.

Expansion of the state vector is then a simple concatenation that is represented as

$$\mathbf{x}_a = \begin{pmatrix} \mathbf{x} \\ \mathbf{m}^W \end{pmatrix}, \tag{3.20}$$

where $\mathbf{x}_a$ denotes the augmented state vector.

The covariance needs to be manipulated in a more extensive manner, as the correlation between the new landmark and the robot pose needs to be captured. The augmented covariance matrix $\mathbf{P}_a$ is expressed as

$$\mathbf{P}_a = \nabla\mathbf{Y}\mathbf{P}_e\nabla\mathbf{Y}', \tag{3.21}$$

where

$$\mathbf{P}_e = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}, \tag{3.22}$$

$$\nabla\mathbf{Y} = \begin{bmatrix} \mathbf{I}_{n\times n} & \mathbf{0}_{n\times 3} \\ \nabla\mathbf{G}_r & \nabla\mathbf{G}_z \end{bmatrix}, \tag{3.23}$$

and where $\mathbf{I}_{n\times n}$ is the identity matrix with dimensions equal to the number of variables already contained in the state vector and $\mathbf{0}_{n\times 3}$ is a null matrix. $\nabla\mathbf{G}_r$ and $\nabla\mathbf{G}_z$ are the Jacobians of the mapping function in terms of the robot pose and the landmark position and are formulated as

$$\nabla\mathbf{G}_r = \frac{\mathbf{g}(\mathbf{m}^R)}{\mathbf{r}}, \tag{3.24}$$

$$\nabla\mathbf{G}_z = \frac{\mathbf{g}(\mathbf{m}^R)}{\mathbf{m}}. \tag{3.25}$$

These Jacobians are defined in Appendix A and are resolved using the final state vector estimate $\hat{\mathbf{x}}_{k|k}$.

## 3.3 SALIENCY DETECTION

The newly developed landmark consists of visual features that are grouped together according to the saliency content of an image. Therefore, the first step in landmark extraction was the computation of a saliency map indicating locally unique and interesting regions. Saliency detection methods based on SR, PFT and PQFT were implemented.

### 3.3.1 SR saliency detection

As described in Section 2.5.3.1, the SR saliency detection method isolates salient regions within an image by removing the redundant information common to all images. The redundant component is formulated in [58] to be a general trend that exists in the logarithms of the amplitude spectra of all images.

The convolution of the log spectrum of the input image with a local average filter $\mathbf{h}_n$ is used as an approximation for this redundant component and is expressed as

$$\mathbf{h}_n = \frac{1}{n^2}\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}, \tag{3.26}$$

where $n$ determines the size of the filter. The SR is then obtained by subtracting the approximation from the original log spectra in the following manner

$$\mathbf{L} = \log\left(||\mathscr{F}(\mathbf{I}_g)||\right), \tag{3.27}$$

$$\mathbf{SR} = \mathbf{L} - \mathbf{h}_n * \mathbf{L}, \tag{3.28}$$

where $\mathbf{I}_g$ is the grey-scale version of the input image.

An inverse Fourier transform is applied to the SR together with the phase spectrum of the original image to construct a new saliency image. The final saliency map is generated by squaring the magnitude of the new image and then blurring it with a Gaussian filter for reasons that will be explained in Section 3.3.3.

### 3.3.2 PFT saliency detection

The PFT method is relatively simple and is based on the formulation in [65]. As is stated in Section 2.5, this method produces similar results to the SR method but is faster.

The method involves isolating the phase spectrum of the grey-scale input image and constructing the new saliency image based on only this spectrum. The amplitude spectrum is set to one by normalising the frequency spectrum before the construction of the saliency image. The saliency map is then constructed by also applying a Gaussian blurring filter to the square of the magnitude of the new image.

### 3.3.3 PQFT saliency detection

For the PQFT method, the saliency map is generated by applying a QFT to the input image using the method described in [65]. The input image is first expressed as consisting of quaternion pixels composed of intensity and colour information. To do so, the input image is formulated as the sum of four orthogonal sub-images, such that

$$\mathbf{I} = \mathbf{I_0} + \mathbf{I_1}\mu_1 + \mathbf{I_2}\mu_2 + \mathbf{I_3}\mu_3, \tag{3.29}$$

where $\mu_i$, $i = 1,2,3$, satisfies the conditions $\mu_i^2 = -1$, $\mu_1 \perp \mu_2$, $\mu_2 \perp \mu_3$, $\mu_1 \perp \mu_3$ and $\mu_3 = \mu_1\mu_2$. Using these conditions, Equation 3.29 is written in symplectic form as

$$\mathbf{I} = \mathbf{f_1} + \mathbf{f_2}\mu_2, \tag{3.30}$$

$$\mathbf{f_1} = \mathbf{I_0} + \mathbf{I_1}\mu_1, \tag{3.31}$$

$$\mathbf{f_2} = \mathbf{I_1} + \mathbf{I_2}\mu_1. \tag{3.32}$$

As can be seen, four information sources can be used to determine the saliency map. Typically, the colour and intensity information of a video image are used as inputs to the QFT, as seen in [65]. Four broadly-tuned colour channels are created and combined into blue-yellow and red-green colour opponency pairings, using Equations 3.33, 3.37 and 3.38.

$$\mathbf{R} = \mathbf{r} - \frac{\mathbf{g}+\mathbf{b}}{2}, \tag{3.33}$$

$$\mathbf{G} = \mathbf{g} - \frac{\mathbf{r}+\mathbf{b}}{2}, \tag{3.34}$$

$$\mathbf{B} = \mathbf{b} - \frac{\mathbf{r}+\mathbf{g}}{2}, \tag{3.35}$$

$$\mathbf{Y} = \frac{\mathbf{r}+\mathbf{g}}{2} - \frac{|\mathbf{r}-\mathbf{g}|}{2} - \mathbf{b}, \tag{3.36}$$

$$\mathbf{RG} = \mathbf{R} - \mathbf{G}, \tag{3.37}$$

$$\mathbf{BY} = \mathbf{B} - \mathbf{Y}. \tag{3.38}$$

The **R**, **B**, **G** and **Y** colour channels are first normalised before producing the colour opponency pairings. These pairings are also used in the SR-based saliency detection method described in [50].

The intensity of the image is also often used as an input and is computed using

$$\mathbf{In} = \frac{\mathbf{r}+\mathbf{g}+\mathbf{b}}{3}. \tag{3.39}$$

In order to obtain the QFT of the video image the Fourier transform of each symplectic component is computed:

$$\mathbf{Q} = \mathscr{F}(\mathbf{f_1}) + \mathscr{F}(\mathbf{f_2})\mu_2 = \mathbf{F_1} + \mathbf{F_2}\mu_2. \tag{3.40}$$

The saliency map is based on the phase spectrum of the image. The effect of the magnitude of the quaternion transformed image **Q** is reduced by normalising the image in the frequency domain and the new symplectic components each undergoes an inverse Fourier transform. The resulting image is then expressed as

$$\mathbf{q}' = \mathbf{a} + \mathbf{b}\mu_1 + \mathbf{c}\mu_2 + \mathbf{d}\mu_3. \tag{3.41}$$

The saliency map is then computed by taking the square of the magnitude of this new image and blurring it using a Gaussian filter, as is done for the SR and PFT methods:

$$Map = g * ||\mathbf{q}'||^2. \tag{3.42}$$

The blurring is necessary as the magnitude of the new image contains individual points of high value in an otherwise low-valued image and is unsuited for segmentation, as can be seen in Figure 3.2.

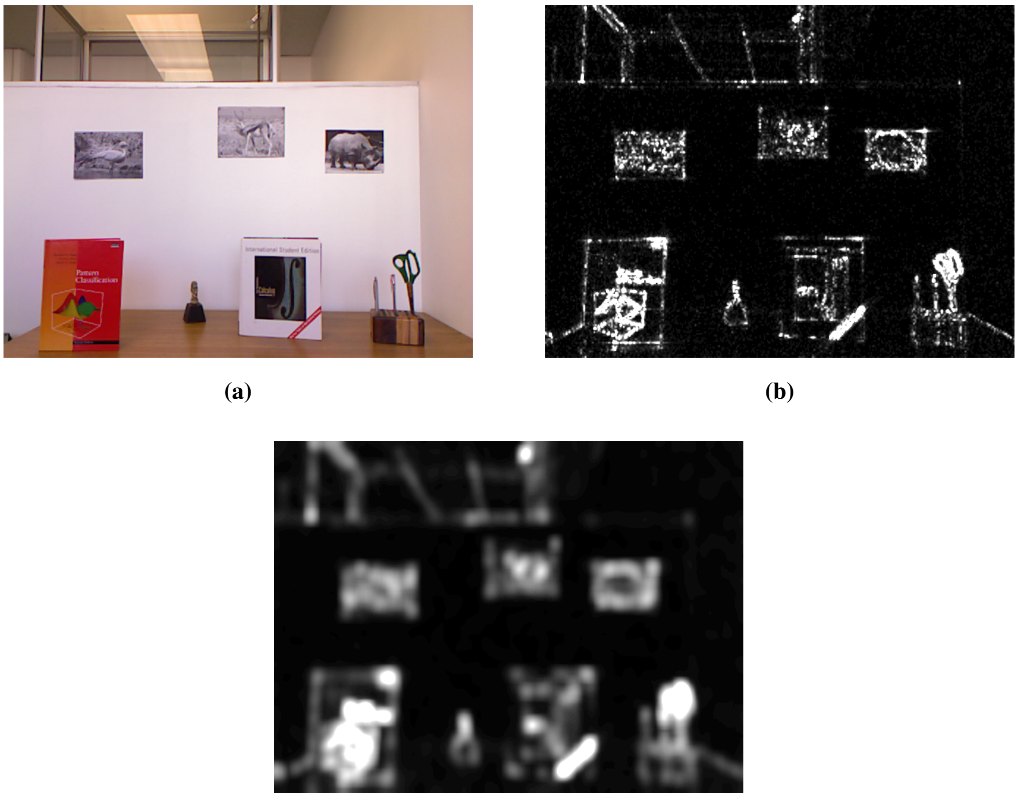


**Figure 3.2:** The saliency map blurring process. (a) Input image. (b) Saliency map produced before blurring; note how only points are identified as salient. (c) Final saliency map after blurring with regions now shown as salient.

## 3.4 DATA ASSOCIATION

The data association step in SLAM indicates the discrepancy between the modelled and the sensed external environment. The discrepancy is the key piece of information used in the EKF update phase to adjust the constant velocity model to account for the changes in the robot movement. The precise matching of extracted landmarks is vital to maintain the accuracy of subsequent processes. Therefore, potential matches between previously matched landmarks and newly detected ones are usually subjected to filtering before the best matches are selected.

A metric that is used frequently in data assocation is the Mahalanobis distance. The Mahalanobis distance incorporates the covariance matrix of the state vector and provides an estimate of how much confidence there is in the positional estimates of the previously mapped landmark and the robot itself.

Inclusion of this confidence estimate into the metric allows for intelligent filtering of matches: If the position of a previously mapped landmark is uncertain then a match with a (reasonably) distant newly detected landmark can perhaps be correct and should not be filtered out before it can be further evaluated. Alternatively, if the position of the mapped landmark and the robot is certain, then the detected landmark will have to be close to the mapped landmark for it to be a plausible match.

The Mahalanobis distance formula, as typically used in SLAM implementations, is given by

$$d_m = v'\mathbf{S}^{-1}v, \tag{3.43}$$

where

$$\mathbf{S} = \mathbf{H}_x\mathbf{P}\mathbf{H}_x' + \mathbf{Q}, \tag{3.44}$$

and where $v$ is the difference between the potentially matching landmarks' 3D world coordinates, **H** is the Jacobian of the observation model, **P** is the state covariance matrix and **Q** is the sensor noise covariance matrix (see section 3.2).

# CHAPTER 4

# SYSTEM IMPLEMENTATION

## 4.1 CHAPTER OVERVIEW

To implement the vision-based SLAM system using saliency grouped landmarks, certain components of the system had to be modified or custom developed. The EKF-SLAM formulations had to be adapted to use the Kinect as a data source. Landmark extraction and data association modules had to be implemented that were capable of processing the new landmark. These modules, sometimes referred to as the visual front-end of a SLAM system, had to be tailored for the implemented EKF-SLAM formulation and the Kinect. The flow chart in Figure 4.1 shows the system components with their associated sub-modules, which will be explored further in this chapter. Finally, the whole system had to be implemented in a software environment on a specific hardware platform.

The chapter commences with a description of the software environment in Section 4.2. The hardware specifications are discussed in Section 4.3 and the usage of the Kinect is described in Section 4.4. In Section 4.5 the modification of the EKF SLAM formulation is explained. The composition of the landmark as well as the implementation details of the landmark extraction sub-modules are explored in Section 4.6. The data association sub-modules and the landmark management module are discussed in Sections 4.7 and 4.8, respectively.

## 4.2 SOFTWARE ENVIRONMENT

A number of open source software packages were used to implement the SLAM system. These packages are widely used in the robotics community and have achieved an acceptable level of maturity while remaining flexible for use in research. The system was implemented using the Ubuntu 10.04

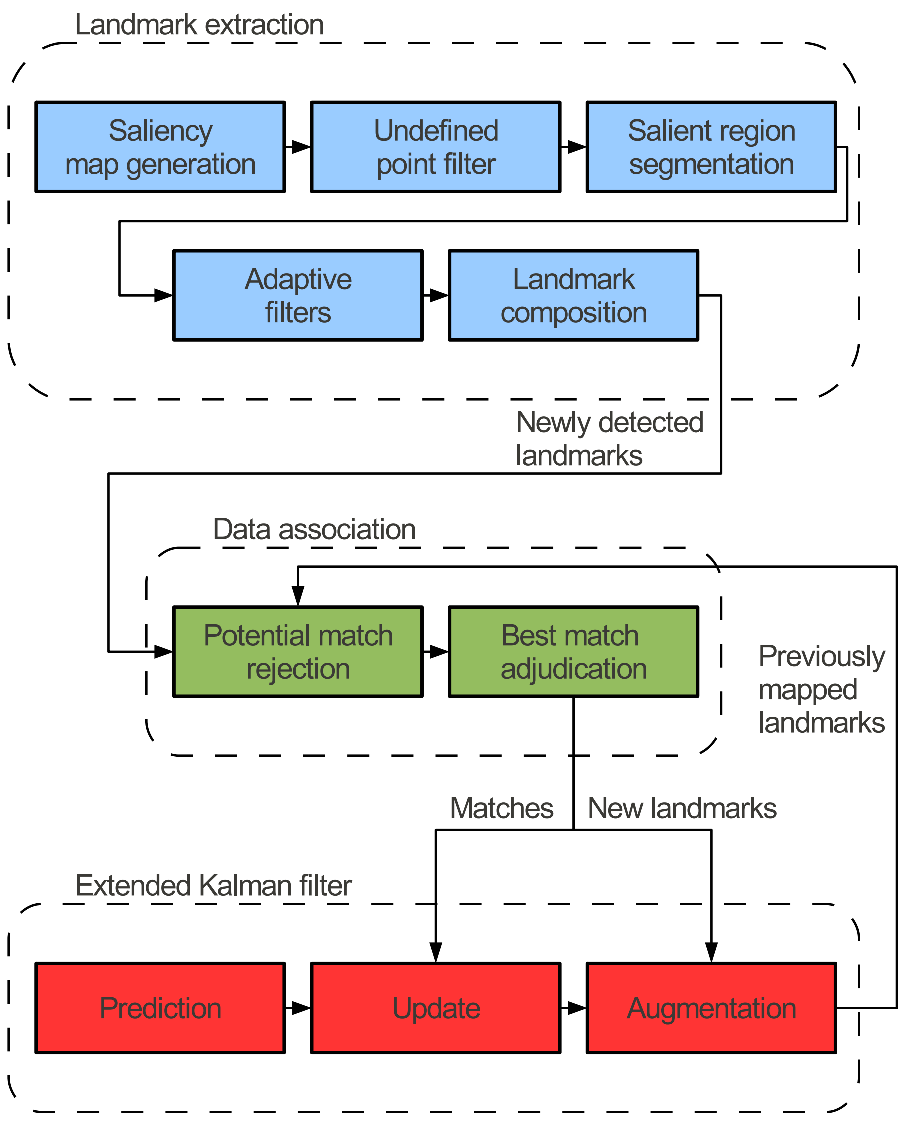


**Figure 4.1:** Flow chart illustrating the interaction between the landmark extraction, data association and EKF (with omission of landmark management).

Lucid Lynx operating system. The C++ programming language was used to implement the system, while MATLAB [72] was used to process the results produced by the system and to plot graphs from these results.

### 4.2.1 ROS

Robot Operating System (ROS) is a meta-operating system developed and maintained by Willow Garage [73]. ROS offers a plethora of drivers for sensor devices (such as cameras and the Kinect) and a data transfer structure, which enable faster development of robotic systems. Furthermore, various useful libraries, such as OpenCV and the Point Cloud Library (PCL), have already been integrated into ROS. ROS also offers several dataset management tools, such as *rosbag* and *rxbag*, to ensure that datasets are recorded correctly and easily used for evaluation purposes. The system also offers various data visualisation tools, such as *rviz*. The ROS Diamondback release was used to develop and evaluate the current system.

### 4.2.2 OpenCV

As has been mentioned, the OpenCV computer vision library [74] has already been integrated into ROS. The library, developed by Intel but now maintained by Willow Garage, contains a variety of image-processing and machine algorithms. A number of these algorithms, especially the feature detection and fast Fourier transform functions, were extensively used in the implementation of landmark extraction and data association routines. OpenCV version 2.1 was used in the current implementation.

### 4.2.3 Eigen

Eigen is a C++ template library designed to provide optimised linear algebra operations such as matrix manipulations, inversions, determinant calculation and transpositions [75]. The EKF SLAM algorithm was implemented almost exclusively using Eigen matrices and vectors. Eigen 3.0 was used as it is integrated as a package into ROS Diamondback.

## 4.3 HARDWARE SPECIFICATIONS

A Dell Precision MP4600 laptop computer was used to develop the system, record the dataset and run the simulations and evaluations. The computer was equipped with an Intel Quad Core I7-2820QM processor (which had a clock speed of 2.30 GHz speed) and 16 GB Random Access Memory. The laptop was also equipped with a 256 GB solid state hard drive. The developed system components were not programmed to use multiple-core processing techniques.

## 4.4 KINECT UTILISATION

In order to use the Kinect for robotic applications, a software driver was needed to access the data from the Kinect via the USB interface. The OpenNi Kinect driver was already implemented as a ROS package [73] and was therefore used for this implementation. During the processing of the Kinect data the OpenNi driver matched each point from the IR camera to an image pixel from the normal colour camera. Both the IR and normal video cameras had a 640x480 resolution; the Kinect was capable of producing a point cloud of 307200 points. Not all of these points were valid or of interest. Therefore the following equations, taken from [76], were used to obtain the 3D coordinates for specific pixels:

$$Z = I_d(v,u)d_s, \tag{4.1}$$

$$X = Z\frac{(u-c_x)}{f_x}, \tag{4.2}$$

$$Y = Z\frac{(v-c_y)}{f_y}, \tag{4.3}$$

where X, Y and Z are the 3D coordinates, $I_d$ is the depth image from the IR receiver, $v$ and $u$ are the pixel coordinates within that image, $f_x$ and $f_y$ are the focal lengths, $c_x$ and $c_y$ are the optical centres and $d_s$ is an experimentally determined scaling factor. These intrinsic camera parameters were different for every Kinect and were determined using OpenCV calibration routines. As the depth image was pre-registered to the normal video camera image received, the images were not rectified, as this would have invalidated the correspondence between the pixels and spatial positions calculated in Equations 4.2 and 4.3.

## 4.5 EKF-SLAM MODIFICATIONS

The EKF-SLAM systems described in [19] and [68] are for camera-based SLAM implementations which use point feature landmarks. Therefore, the observation model from the update phase and the mapping function from the augmentation phase had to be modified as these functions manipulate saliency-based grouped landmarks as extracted from Kinect data.

Three factors influenced the formulation of the observation model. Firstly, the Kinect sensor provided the means to express the sensed landmark positions as 3D coordinates. Secondly, landmarks were stored as positions relative to the world frame during the augmentation step. Lastly, the observation model was not applied to the robot pose in the state vector. Therefore, the observation model merely consisted of transforming the stored coordinates for each mapped landmark $i$ from the world frame ($\mathbf{m}^W_{i,k|k-1}$) to the frame of view of the robot ($\mathbf{m}^R_{i,k|k-1}$). The observation model can be expressed as

$$\mathbf{m}^R_{i,k|k-1} = \mathbf{h}\left(\mathbf{x}_{k|k-1}\right) = \left(\mathbf{q}^{WR}_{k|k-1}\right)^* \otimes \mathbf{q}\left(\mathbf{m}^W_{i,k|k-1} - \mathbf{p}^W_{k|k-1}\right) \otimes \mathbf{q}^{WR}_{k|k-1}, \tag{4.4}$$

where $\otimes$ indicates quaternion multiplication and $x^*$ the conjugate of $x$.

The role of the mapping function is to transform newly observed and unmatched landmarks from the robot frame to the world frame and can be seen as the inverse of the observation model:

$$\mathbf{m}^W_{i,k|k} = \mathbf{g}(\mathbf{m}^R_{i,k|k}) = \mathbf{q}^{WR}_{k|k} \otimes \mathbf{q}\left(\mathbf{m}^R_{i,k|k}\right) \otimes \left(\mathbf{q}^{WR}_{i,k|k}\right)^* + \mathbf{p}^W_{i,k|k}. \tag{4.5}$$

## 4.6 LANDMARK EXTRACTION

As can be seen in Figure 4.1, the implemented landmark extraction routine consisted of a number of sub-modules, of which the operation can be summarised as follows: Firstly, a saliency map was generated using one of the method described in Section 3.3. Thereafter, the pixels corresponding to undefined 3D points were removed from this map. Suitable contours indicating regions of high saliency were then found in the filtered map. Once the contours had been found the visual features, or keypoints, could be detected in the image and grouped according to the salient regions. The landmarks were then finalised by obtaining descriptors of the grouped keypoints and the average 3D position of all the pixels within the regions.

### 4.6.1 Landmark composition

Each selected salient region, with the accompanying keypoints, was used to form a landmark. The 3D position of the landmark was determined by averaging across all of the pixels' corresponding 3D points within the region. The mean position was stored together with pixel positions for the FAST keypoints within the region along with the accompanying SURF descriptor for each keypoint. A descriptor is a feature vector representation of the image region surrounding a keypoint [77]. Descriptors were used to uniquely identify keypoints in such a way that keypoints could be distinguished from one another while also providing the means to match specific keypoints between video frames. A keypoint on its own did not provide enough information to robustly identify it between frames.

SURF descriptors are able to match keypoints across frames even if there are large scale and rotational changes [77]. Although there are a variety of different descriptors that also exhibit such invariance (such as SIFT and the FAST descriptor), SURF descriptors were found to provide the required balance between computational speed and keypoint distinctiveness for the SLAM system.

The SURF descriptors, together with the FAST keypoint positions and the landmark 3D position were used to match previously mapped landmarks with newly detected ones.

The rationale for using the described approach to forumalate a new landmark was that the 3D positions of salient regions were more stable than those of singular keypoints and that using the descriptors of multiple keypoints led to a more robust matching process.

### 4.6.2 Saliency detection method implementation notes

The saliency detection methods described in Section 3.3 frequently use the Fourier transform or its inverse. The discrete Fourier transform (DFT) function from the OpenCV library was used whenever a Fourier transform was required.

The PQFT method allows the usage of four different inputs to the QFT to produce a saliency map. The implemented system made use of two colour-opponency pairings and the intensity of the image, formulated in Equations 3.37, 3.38 and 3.39. The difference between sequential intensity images is used in [65] as the fourth input to the QFT. The intensity difference is used in an attempt to assign saliency to dynamic regions within an image, which would be inappropriate for the implemented SLAM system where it is desired to identify stable regions as potential landmarks. Therefore, the fourth input to the QFT was set to zero, as suggested in [65]. Future studies can focus on using additional information through this channel to improve the saliency map generation. By substituting the described inputs into Equations 3.31 and 3.32, the following symplectic components were obtained:

$$\mathbf{f_1} = \mathbf{RG}\mu_1, \tag{4.6}$$

$$\mathbf{f_2} = \mathbf{BY} + \mathbf{In}\mu_1, \tag{4.7}$$

which were then applied to Equation 3.40.

In [58] it was found that the input image size affects the size of identified salient regions. A large input image would result in the isolation of smaller regions containing finer details, while a smaller input image would result in finer details being ignored in favour of large objects in a scene. For the implemented SLAM system, the input images were resized to 260x200 using bilinear interpolation. The OpenCV *resize* function was used. The images were resized for all three saliency detection methods.

The resize value was found by means of experimentation with the datasets described in Chapter 5. The particular size was found to identify regions sufficiently large to accommodate numerous keypoints while not encompassing the whole of the viewed scene. Small variations of the resize value were not found to have a significant influence on the size of salient regions produced, while large variations led to unusable saliency maps. The invariability to small changes can be attributed to the adaptive threshold filter, described in Section 4.6.5, that isolates regions according to the local saliency of each region. The local saliency remains relatively constant for small changes in input image size.

### 4.6.3 Undefined point filter

The point cloud generated by the Kinect 3D sensor usually contained a number of undefined points. These were points for which 3D positions could not be computed because they were located beyond the range of the sensor, reflected the IR laser pattern in an unexpected manner or were overexposed to other sources of IR light. As these points could corrupt the estimation of the landmark position, the corresponding pixels of these points and the surrounding regions needed to be filtered from the saliency map.

A filtering mask was generated according to the position of these corresponding pixels. A zero-initialised image was assigned large values at each of the positions, whereafter the image was blurred to form the mask. The filtering operation involved setting pixels in the saliency map to zero if a corresponding pixel in the mask had a non-zero value. Figure 4.2 illustrates this process. Note that this filtering was applied before the map was blurred.

### 4.6.4 Salient region segmentation

After the saliency map had been obtained and filtered, the input video image needed to be segmented into high-saliency regions. The saliency map was first filtered using an adaptive threshold to form a segmentation mask. The adaptive threshold is described in Section 4.6.5. Thereafter contours were detected within this mask using the standard OpenCV $findContours$ algorithm. Contours that were too small were rejected, as well as contours that were too close to the borders of an image. The latter

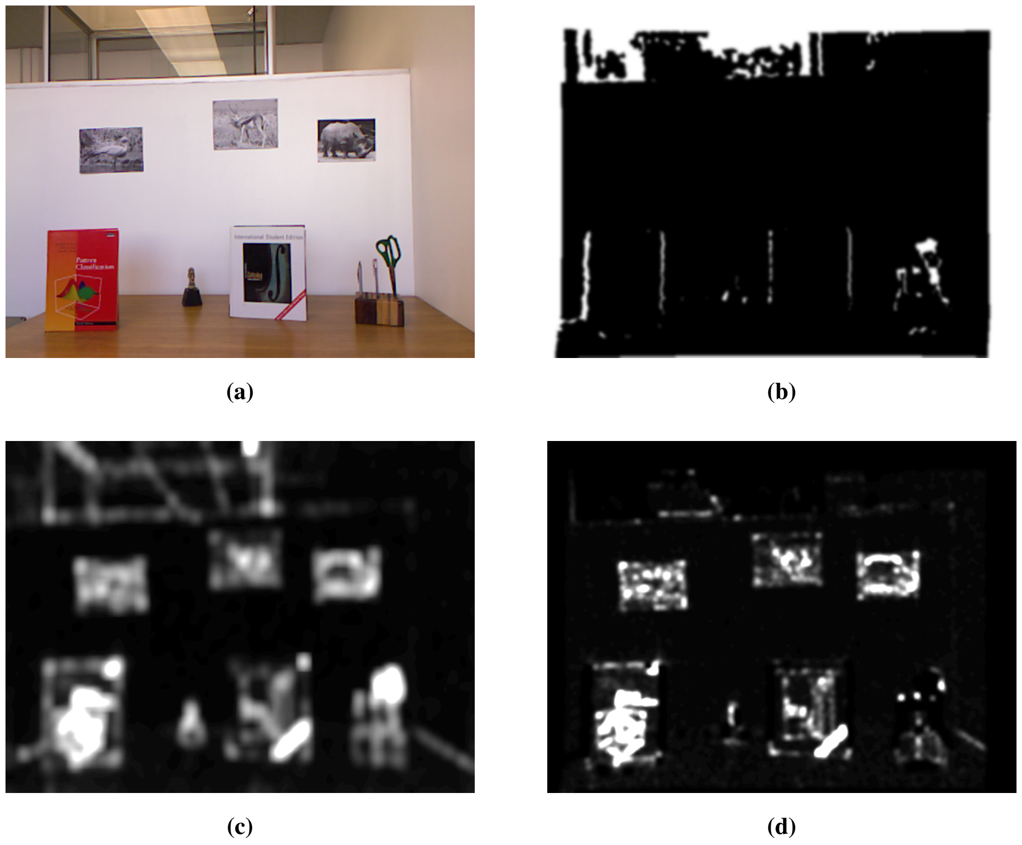

(a) (b) (c) (d)

**Figure 4.2:** Filtering of undefined 3D points. (a) Input image.(b) Filtering mask. (c) Saliency map before filtering. (d) Filtered saliency map. Note the absence of the scissors (bottom right corner) and the reflections from the glass case (top left and centre) from the final salient map, both of which correspond to a number of undefined points.

contours were most likely only partially observed regions and would grow or shrink as the Kinect was moved.

After this filter, features from accelerated segment test (FAST) keypoints were detected across the whole of the image and then sorted according to the contour in which the keypoints were found. Before being sorted, the detected keypoints were also subjected to a threshold filter to remove keypoints of low quality. After some experimentation with the available OpenCV feature detection routines, such as SURF, SIFT, and Shi-Tomasi corners, it was decided to use FAST keypoint detection because it maintained a balance between the number of keypoints detected, the stability of keypoint positions and computational efficiency. A final filtering phase rejected contours with too few keypoints according to an adaptive parameter, also described in 4.6.5.

### 4.6.5 Adaptive filters

Judicious selection and filtering of landmarks was key to promoting accurate position estimation. Superfluous landmarks would also overburden the estimation algorithm unnecessarily. However, not having any landmarks for an extended period would lead to even more inaccurate estimation. The estimation error would increase because changes in the movement of the Kinect would not be incorporated into the estimated movement model parameters, which in turn would cause an unbridgeable discrepancy in the modelled movement and the real movement of the Kinect.

An absolute threshold was used to filter out regions that were obviously not salient enough for use as landmarks. The threshold value was determined by processing the images from a number of scenes and changing the threshold value so that regions associated with a floor, wall or tabletop were removed. If used in isolation, an absolute threshold would inevitably lead to too few landmarks in texture-sparse regions of an image, while proliferating the number of landmarks in texture-rich regions. Therefore an adaptive thresholding algorithm was also applied on the normalised saliency map, which emphasised locally interesting regions.

The OpenCV $adaptiveThreshold$ algorithm was used to find locally interesting regions. Pixels within the saliency map were deemed locally salient if the pixel value was above the mean of a surrounding block of pixels. Thereby regions were isolated that had a high saliency relative to a local section of the map. Use of the adaptive threshold resulted in separated regions being found in high-textured sections where otherwise only a very large region would have been identified using a global threshold. Regions were also isolated in low-texture regions where no regions would have been found. After some experimentation using the datasets described in Section 5, a block size of 51 was found to isolate regions large enough to contain a suitable number of keypoints but also small enough not to vary in size across image frames owing to too large a background area being used to compute the mean. The choice of the block size would be dependent on the resolution of the camera as a higher resolution would require a larger block size to extract suitably large regions. A change in the image resize value used by the saliency detection method would also affect the block size. Local saliency, after the removal of globally inadequate regions, also provided a more consistent segmentation mechanism for landmarks, as the global saliency of a fixed area would vary between camera poses, while the local saliency would stay mostly the same.

The third threshold that that was automatically adapted to suit the environment was the minimum number of keypoints within a contour required for that contour to be acceptable. The keypoint threshold was adapted according to the number of contours that were valid for the viewed scene. Contours were first extracted using no threshold. Thereafter the contours were counted that had more keypoints than the threshold value. If the number of contours was less than the minimum number of contours required, the keypoint threshold value was lowered by two until the needed number of contours or the minimum threshold value was reached. After a threshold value was determined, the contours were filtered according to this value. The threshold was reset to an initial value before every image was analysed.

## 4.7 DATA ASSOCIATION

During the data association phase, each newly detected landmark was matched to all previously mapped landmarks. Potential matches were evaluated so as to retain only quality matches with a high probability of being correct. Thereafter, the best match for each newly detected landmark was selected from all the potential matches. Once the best matches were found, the descriptors and keypoints of the previously mapped landmarks were replaced by those from the newly sensed landmarks.

### 4.7.1 Potential match rejection

Potential matches were filtered using two criteria: the 3D spatial distance between the landmarks and the line length variance. The use of spatial distance measures for data association required that the estimates produced by the SLAM system remained sufficiently accurate to provide good positional tracking throughout the whole operational time span of the system. If this could not be guaranteed it would have been required to use loop closure to realign the estimations. In this implementation it was assumed that the relative movement of the Kinect between video frames as well as the total distance travelled for each dataset was sufficiently small for the system to remain accurate.

If potentially matching landmarks were very far apart in terms of their Euclidean distance, the match was rejected outright. The position of the previously mapped landmark was of course projected into the robot frame to facilitate distance computation. After the rough Euclidean distance filter was applied, the Mahalanobis distance $d_m$ was computed and if it was above a set threshold, the match was also rejected.

The next filter was based on the line length variance and evaluated the similarity in the physical distribution of the keypoints in each of the matched landmarks. The filter served to improve the robustness of the data association process by using the geometric relations between the pixel positions of keypoints. The line length variance is similar to the lengths scatter used in [46]. The FAST keypoints of the previously mapped landmark were matched to the keypoints in the newly detected landmark according to the SURF descriptors of the keypoints. The variance of the length of the lines connecting each matched keypoint pair was then computed and if it exceeded a threshold, the match was rejected. The aim of this filter was to ensure that the keypoints of the matched landmarks were distributed in a similar fashion within the video frames. A large line length variance would indicate that the keypoints in the previously mapped landmark were arranged vastly differently to those in the newly detected landmark and the match between the landmarks would most likely be incorrect.

The angle of the keypoint pair connecting lines could also have been used as a filtering metric as well the differences in the 3D density of the physical positions of the landmark keypoints. However, these methods were more computationally expensive than the implemented line length variance. The alternative methods should be investigated in future studies.

### 4.7.2 Best match adjudication

The filtering of incorrect matches was required, but it was also necessary to find the best match in all the acceptable matches between a single mapped landmark and different newly detected landmarks. To compute the best match metric, the keypoints in the newly detected landmark **n** were matched to those in the previously mapped landmark **m** and vice versa. During this process the descriptor distance between each keypoint match pair was also computed. The descriptor matching measure $d_d$ was computed using the following formula:

$$d_d = \frac{\sum_{i=0}^{N} d_i^{nm} + \sum_{j=0}^{M} d_j^{mn}}{N+M}, \tag{4.8}$$

where $d_z^{nm}$ indicates the Euclidean distance between the SURF descriptors of the $i_{th}$ keypoint pair match in **n** to **m**, $N$ is the number of keypoints in **n** and $M$ is the number of keypoints in **m**. The Euclidean distance was used, although the SURF descriptor space is not metric in nature, because it provides an efficient and discriminatory measure to compare the quality of matches.

The computed best match metrics were also subjected to a set threshold, so that only quality matches were propagated through the system. If more than one mapped landmark was matched to a new landmark, the matching pair with the smallest distance measure was selected as the correct match.

## 4.8 LANDMARK MANAGEMENT

As a robot explores an environment, the number of tracked landmarks increases, which affects the computation time of the system. Such an increase is expected as a robot enters unexplored regions. However, sometimes the state vector can become augmented with landmarks that burden the system while remaining unmatched. These landmarks are not reliably extracted or remain unmatched although the robot is in the proximity of their recorded location. Such problematic landmarks are the result of a failure of the sensoral front-end of the system to extract only quality landmarks, but sensor noise (and motion blur if the sensor used is a camera) can also lead to the extraction of landmarks that can not be easily redetected in other video frames. The problematic landmarks have to be removed to ensure the efficient operation of the system.

A landmark management method similar to that in [68] was used: A landmark was selected for removal if the number of times it had been within range of being detected ($n_r$) was less than ten and the number of times it had been detected ($n_d$) was lower than a quarter of $n_r$. The landmark was removed by deleting the landmark's position state vector and removing all covariance values associated with it from the state covariance matrix. The threshold values were determined by evaluating the SLAM system for different threshold values using the *XYZ* dataset (described in Section 5.5). The specific values were selected because a reasonable number of landmarks were removed and the system accuracy was not affected by the landmark management.

A quick timing test to illustrate the need for a landmark management routine was conducted. The complete EKF SLAM system with visual front-end was tested using the *RPY* (described in 4.6.5) dataset sequence and parameters described in Section 5.5. Without the use of landmark management the processing time to analyse the whole of the 2 minutes and 3 seconds sequence was 9 minutes and 26 seconds. With the landmark sequence it only took 5 minutes and 57 seconds.

# CHAPTER 5

# RESULTS

## 5.1 CHAPTER OVERVIEW

A number of simulations and real-world experiments were carried out to determine whether the vision-based SLAM implementation using saliency grouped landmarks succeeded at localising a robot in an unknown environment. An additional aim of the tests was to investigate whether a SLAM system using the newly developed multiple-feature landmarks would produce better results than a SLAM system using single feature landmarks. Within the SLAM research community a variety of methods and measures are used to evaluate implementations. A short investigation was conducted to determine which evaluation methodology was most applicable to the developed system and this investigation is summarised in Section 5.2. The manner in which the statistical significance of a comparison between SLAM methods should be evaluated was also investigated. Upon the conclusion of the short investigation it was decided to evaluate the SLAM system using three methods.

In Section 5.3 the simulations that were used to validate two important fundamental assumptions of the system are described. The first assumption is that the core EKF SLAM implementation, based on the constant velocity movement model, produces behaviour that is correct in relation to other standard SLAM implementations. The second assumption is that the combination of singular features into grouped landmarks can lead to an improvement in the accuracy of SLAM.

After the system was validated using simulation, the complete system was tested using two datasets of recorded Kinect data. In Section 5.4 a newly generated dataset was used to test the performance of the system in repeatably determining the position of specific waypoints, in a manner similar to that described in [19]. Specific factors that have an impact on the accuracy of the system were identified and experiments were conducted to further investigate the effect of these factors.

Thereafter, in Section 5.5, the localisation capability of the system across the whole of the path travelled was evaluated using a "test bench" dataset created by the Technical University of Munich [78]. The performances of three SLAM implementations, each using a different frequency-based saliency detection method, were compared to determine the method producing the best landmarks. The dataset was also used to illustrate the advantages of the current system over a similar system using singular feature landmarks in Section 5.6. A short computational analysis comparing these two implementations was also conducted. The overall performance of the system was analysed and is discussed in Section 5.7.

## 5.2 PREVIOUS APPROACHES TO EXPERIMENTAL EVALUATION

### 5.2.1 Validation techniques

As SLAM is a well-studied problem, it is common to evaluate a new implementation to investigate whether behaviour that is considered typical, correct and required of a SLAM system can be observed. In [79] it is proved that a nonlinear 2D EKF-based SLAM system using point landmarks will always converge. A corollary to this proof is that any determinant of the uncertainties, that is to say the covariance matrix entries, of the mapped landmarks will always be monotonically decreasing. The observation of this phenomenon in the determinants of the covariance matrix has become a measure by which a SLAM algorithm can be validated. Such testing can be conducted by using simulations [80, 81] and real-world data [6].

Another validation technique is to examine the uncertainty of newly mapped features, as seen in [70, 82, 83]. As the robot or camera progresses through an environment these uncertainties will continue to increase until a previously mapped feature is reobserved, after which there is a noticeable drop in the uncertainties of all mapped features. The drop in uncertainty is indicative of the ability of the system to correct for drift and to realign the estimated map accordingly [48, 84]. Such behaviour is commonly illustrated using ellipses surrounding landmarks of which the size is dependent on the uncertainty in a graphical representation of the recorded map.

### 5.2.2 Evaluation methods

To evaluate the accuracy of a SLAM system it is desirable to compare the estimated path and map to a ground truth. GPS data can be used to provide the ground truth for outdoor applications [32] but typically not for indoor implementations. For the latter other means of establishing a ground truth must be used. In [19] a camera was manually moved along a measured rectangular path. The corner points of the path were used as waypoints and the results were stated as the average of the system estimates at these points. A grid was marked on a floor in [24] to measure the positions of a robot along the path that it was travelling. The orientation of the robot was measured using a laser pointer. A laser sensor was used in [85, 86] to construct a ground truth. Simulations of the algorithm and operational environment can be used to provide an easy source of ground truth [87, 88, 89].

A number of implementations have been evaluated without using any kind of ground truth. In such cases the robot is made to travel along a path and return to its initial position. The difference between the initial and final estimates is then seen as a very rough indication that the system does not diverge and is consistent [18, 24, 47, 83]. In [90] a six degrees of freedom SLAM system was tested on a flat level surface. The estimated distances from the surface and vertical angle were used as error measures, as these were supposed to be zero. SLAM system performances are often qualitatively compared to methods using odometry only [8, 18, 91]. In such cases the consistency of the SLAM maps and the divergence of the odometry method maps are often illustrated.

Most methods only use the robot pose estimates, but the estimated position of landmarks can also be used to evaluate a SLAM system. The estimated landmark positions can be compared to manually measured ground truths [71, 92, 93]. In [8] detected landmark points were expected to be co-planar and the root mean square error (RMSE) of the fitted plane and point positions were used. In [83] the stability of landmark tracking across video frames was evaluated by comparing scale scores and orientation of matches for different view points of the landmarks. A particle filter SLAM implementation was evaluated in [88] using the log-likelihood of the observation data given the best estimate of the robot trajectory and map. In [89] the normalised estimation error-squared (NEES) for the mapped landmarks was also calculated. Although the landmark positions offer another way to evaluate a system, these results are usually not comparable between implementations because of differences in sensors used and landmark extraction algorithms [94].

### 5.2.3 Error metrics

To evaluate the accuracy of the pose estimated, the difference between the pose estimate and the ground truth, or error, is often plotted over time [32]. In [86] and [95] the calculated error was shown to be within a $2\sigma$ bound and was therefore deemed acceptable. The average and mean of the quadratic error was reported as an overall result in [91], while in [96] the RMSE was used. The mean error together with the maximum error and percentage mean error was shown in [68]. These overall metrics attempted to indicate the consistency of the SLAM system. A more thorough analysis of consistency can be achieved using the NEES, as in [97] and [89]. The NEES measure is usually difficult to use, as it requires multiple algorithm runs and the formula uses the information matrix, which is not always readily available [94]. In [89] this measure was used to analyse the results generated by a simulation and only the robot pose covariance matrix was inverted instead of the whole covariance matrix.

All of the previously mentioned metrics evaluate the absolute difference for each pose estimate. In [94] it was argued that this approach can lead to an inaccurate evaluation of an algorithm. Initial errors (typically encountered when a system starts to add features to its map [32]) would affect the map on which subsequent estimates were to be based. Errors of the same size as such initial errors would have less of an impact on subsequent measures but would lead to a more optimistic absolute error result. The relative displacement between pose estimates was proposed in [94] as an alternative. A similar approach can be seen in [8] where it was reasoned that the advantage of SLAM over pure odometry methods lies in the improved consistency of positional estimates where there was an overlap in detected features. To evaluate this property, the error in the relative pose of every pair of estimates was plotted as a function of the distance between the pair of estimates used. The relative displacement error was also useful in situations where a point of origin could not be fixed [14]. However, for the current implementation it was felt that an absolute error measure would capture the improvement in the overall map and positional estimates that would result from the compensation characteristic of a SLAM system that was reacting to a previous erroneous positional estimate far more effectively.

Throughout this chapter the absolute trajectory error (ATE) is used to evaluate the performance of the SLAM algorithm with regard to the real pose of the robot. The formulation of the ATE that is used is very similar to that provided in [76]. The error measure evaluated the estimation accuracy of the system for the whole of the traversed trajectory and allowed for various statistics to be computed which enabled comparison between SLAM implementations. The ATE is calculated by determining the error at each time step, which is computed as the Euclidean distance between the 3D positional estimate and the real 3D position. The orientation estimation capability of the system was tested separately in Section 5.4.

### 5.2.4 Statistical significance of results and comparisons

Various statistical parameters based on the ATE are typically computed to quantify the overall performance of a SLAM system for a specific dataset. Commonly used parameters are the mean, standard deviation and RMSE [30]. As these parameters are computed from a limited sample set, the parameters are only estimates of the true parameter value. Therefore, it is necessary to consider a confidence interval surrounding a parameter estimate which indicates the expected range in which the true value lies.

SLAM systems can be evaluated using the overall mean of the averages from a set of experiments. If the averages of the experiments are assumed to be normally distributed, the fact that 95% of the averages lie within two standard deviations from the overall mean can be used as a confidence interval. Such confidence intervals were calculated for the results of two different localisation methods in [98], where the confidence interval indicated the clear superiority of one method over the other in certain circumstances. In [99] a 68% confidence interval (equal to one standard deviation separation from the mean) was calculated of the overall mean information loss of different landmark management strategies but only to indicate the spread of results for each strategy.

When comparing the statistical parameters of different SLAM algorithms, the parameters can be found to be quite close in value to each other. As the parameters are only estimates of the true values, it is necessary to consider with what degree of confidence it can be stated that the true values are significantly different. In such cases the null hypothesis that the two parameters are the same must first be evaluated before any conclusions can be drawn from the comparison.

The Student's $t$-test is an often used statistical method for evaluating a parameter estimate or the comparison between two sets of results (through a null hypothesis evaluation at a set confidence value) [100]. $t$-tests were used to compare different EKF-based orientation estimators in [101] and different PF-based localisation algorithms in [102]. Vision-based localisation methods were evaluated using the $t$-test in [103] and [104]. The latter method is based on biological principals and uses the saliency of a scene to identify landmarks.

One of the assumptions of the Student's $t$-test is that the parameters under evaluation are drawn from normal distributions. An alternative method that does not assume any distribution is the Mann-Whitney-U test, also know as the Wilcoxon rank-sum test. In [105] it is used to compare different versions of a semantic place classification system that fuses visual cues and laser scan data. The performance of a laser-based SLAM system was evaluated for different settings using a Wilcoxon test in [106]. In [107] both the $t$-test and the Wilcoxon test are used to compare global localisation methods due to uncertainty regarding the normality of the results evaluated.

Although the $t$-test and the Wilcoxon test are typically applied to results gathered from multiple simulations or real-world experiments, the tests can also be used to compare different methods applied to a single dataset. A $t$-test was used in [108] to evaluate the performance of a PF-based localisation system, using different types of maps, on a single set of data collected by a robot equipped with a laser scanner. In [109] a dataset was generated using a simulated vehicle operating in a virtual environment. A two-tailed $t$-test was then used to compare localisation methods based on odometry, a Kalman filter and a PF. A Gaussian Sum Filter localisation system is compared to a PF-based system in [110] using a paired $t$-test and a single dataset as gathered by an autonomous vehicle operating in an urban environment.

The Student's $t$-test is used to evaluate the results presented in this chapter, as the $t$-test is commonly used in the literature. As the results were often found to fail the Shapiro-Wilk test for normality, the Wilcoxon test is used to verify the $t$-test result, as was done in [107]. The appropriate version of either test is used depending on whether the samples were considered dependent or independent. Only results exceeding the 95% confidence level are considered as statistically significant. Both Microsoft Excel and the SciPy library in Python are used to calculate the statistical significance. The use of Python allows for direct computation of the confidence level at which a comparison is statistically significant. To enable the visual comparison of results, the $t$-test 95% confidence intervals for various statistical parameters are computed and are visualised using error bars.

## 5.3 SIMULATIONS

Simulations were used for two purposes: To validate the implementation of the constant velocity model EKF SLAM formulation and to validate the theory that grouped landmarks were better for use in SLAM than singular landmarks.

### 5.3.1 EKF SLAM validation

As has been discussed in Section 5.2, two typical observable phenomena are typically used to evaluate SLAM systems. The first is consistent growth in the relative uncertainty of newly detected landmarks, followed by a sharp decrease in uncertainties when a previously mapped landmark is reobserved. The second is continuous reduction in any of the determinants of the mapped landmark covariance matrix entries. To determine if these phenomena could be observed in the developed system, a C++ computer simulation was used. An important point to note is that the simulation did not test the landmark extraction and data association methods, only the core EKF SLAM algorithm.

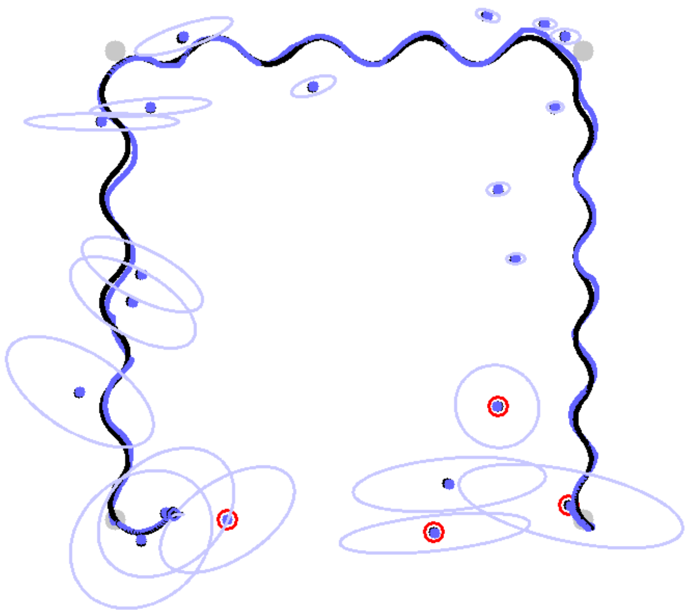
In the simulation a robot moved between six preset waypoints in a 3D space and observed point landmarks as it moved. Waypoints were defined by a 3D position as well as an orientation unit vector. Landmarks were defined only by a 3D position. The robot velocities were manipulated so as to change the pose of the robot to match that of the waypoint pose. Note that only the 3D position was evaluated to determine if a waypoint had been reached. The robot movement was made to correspond with the constant velocity movement model by imposing certain constraints: The robot translational and rotational velocities were changed gradually and the robot was forced to remain in motion and never come to a complete stop. Furthermore, the maximum velocities of the robot were limited to 0.1 m/s and 0.025 radians/s respectively. The size of all time steps was set to 0.01 seconds.

The manner in which landmarks would be observed by a robot equipped with a noisy sensor with a restricted viewing angle was simulated in the following manner: Landmarks were observed if they fell within a spherical sector with a radius of 8 m and an angle of 70 degrees, as projected from the robot's position and in accordance with its orientation. The sensed landmark positions were transformed from the world frame to the robot frame and then corrupted with zero-mean Gaussian noise with a 0.02 standard deviation. Perfect data association was used to match the newly sensed landmarks with previously mapped landmarks. These matched landmarks were then input into the EKF SLAM algorithm. There were 20 landmarks in the simulated environment. The diagonal values of the EKF expected noise covariance matrices were set to values found to work well in experimental testing, while those for the observation noise matrix were set to match the simulated noise.

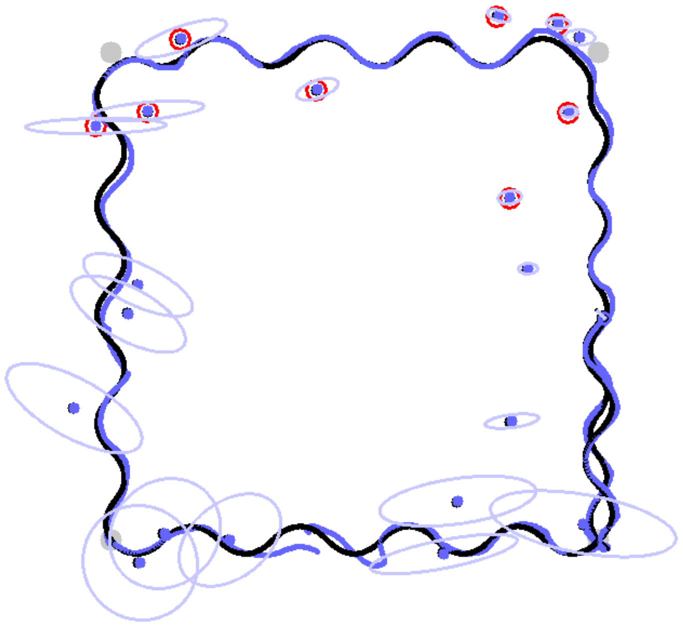
The simulation was visualised using the OpenCV library drawing functions [74]. Although the visualisations were only in two dimensions, they represented a cross-section of the entire simulated 3D space. Figure 5.1 shows the state of the simulation before and after several previously mapped landmarks were reobserved. The black line indicates the real path of the robot while the lighter or blue line indicates the estimated path of the robot. The black and lighter/blue circles with lines through them show the real and estimated pose of the robot. Large light (grey) coloured dots are the waypoints of the robot. Black and lighter (blue) smaller dots are the real and estimated landmark positions. Landmarks are surrounded by a light grey covariance ellipse if they have been added to the SLAM map, and by a close-fitting red circle if they are actively being observed. The robot was moved in a counter-clockwise direction.

In Figure 5.1a it can be seen that the landmarks that had been observed later in the simulation had a larger covariance ellipse than landmarks that had been observed at the start of the simulation. The newer covariance ellipses were larger because the uncertainty of a new landmark was dependent on the uncertainty the estimation algorithm had in the accuracy of the pose of the robot at the time of observation. As no landmarks had been reobserved at this stage, no information which could have led to a decrease in the pose uncertainty had been collected and it continued to grow. Figure 5.1b shows the simulation after the landmarks that had been observed at the start of the simulation were reobserved. As can be seen, the landmark covariance ellipses had decreased in size. Figure 5.2 shows the covariance value of the last observed landmark's x-axis estimate together with the observation instances of one of the landmarks that had been mapped earlier in the simulation. There was a marked decrease in the last landmark uncertainty at the point where the older landmark was reobserved, at about time step 2100. When the robot reobserved a previously mapped landmark, the estimation algorithm updated the robot pose and landmark estimates using this information. The correction of the estimates showed that the implemented SLAM algorithm can compensate for drift typical of odometry and movement estimation systems.

The determinant of various landmark position covariance values can be seen in Figure 5.3. The determinants were calculated and plotted in MATLAB from the covariance values produced by the simulation software. Plots were vertically magnified until it could be satisfactorily determined that the graph was monotonically decreasing. The time step values differ between the graphs because determinants could only be calculated when the specific landmark had been observed. As can be seen, the determinants were monotonically decreasing, which indicated that the EKF SLAM algorithm is convergent. The convergence of the algorithm, together with the drop in landmark uncertainty after reobservation, validates the constant velocity model EKF SLAM formulation.



(a)



(b)

**Figure 5.1:** True and estimated robot trajectory and map (a) before and (b) after reobservation of previously mapped landmarks.

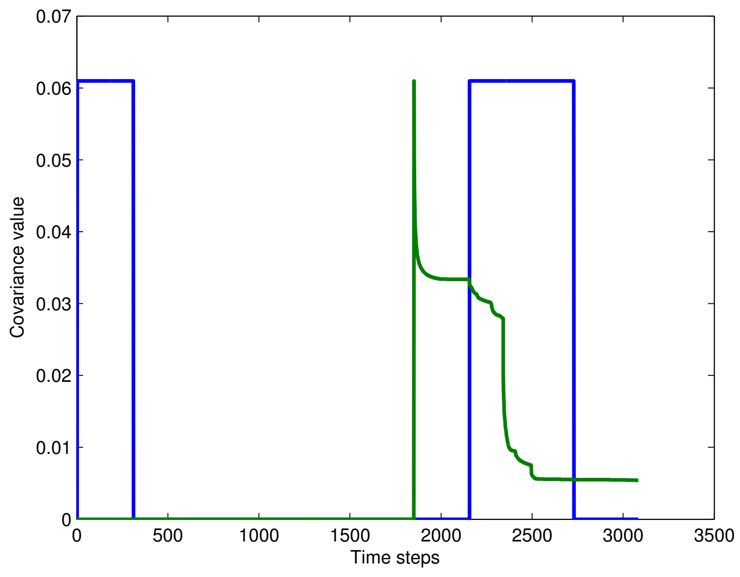


**Figure 5.2:** $P_{xx}$ of $20^{th}$ landmark with observation instances of 1st landmark.

### 5.3.2 Grouped landmark assumption testing

A C++ computer simulation was used to verify the assumption that the grouping of features would lead to better estimation results. As the simulation was very similar to the one described in Section 5.3.1, only the differences will be described. Observable features in the simulation were grouped before the start of the simulation into six clusters of five features. The extracted landmarks consisted of either singular features or a group of features. The two SLAM systems using these methods were named singular landmark (SL) SLAM and grouped landmark (GL) SLAM. In the latter method the mean of the features was used as the position of the landmark. The features were all continuously observed by the robot so as to simplify the comparison between the two methods. Zero-mean Gaussian noise with a standard deviation of 0.05 was added to the sensed position of each feature. The observational noise covariance value of the EKF was set to match the simulated noise. The number of waypoints had been reduced to five to speed up the computation of results.

The simulation was executed ten times for each feature-sensing method and for a varying numbers of features. The average and standard deviation of the ATE RMSE for these experiments can be seen in Table 5.1. The difference between the averages of the two methods is expressed as a percentage of the SL method average. The table also shows the $t$-values of the two sample $t$-tests conducted to verify the statistical significance of any observation of improved performance. A two sample $t$-test

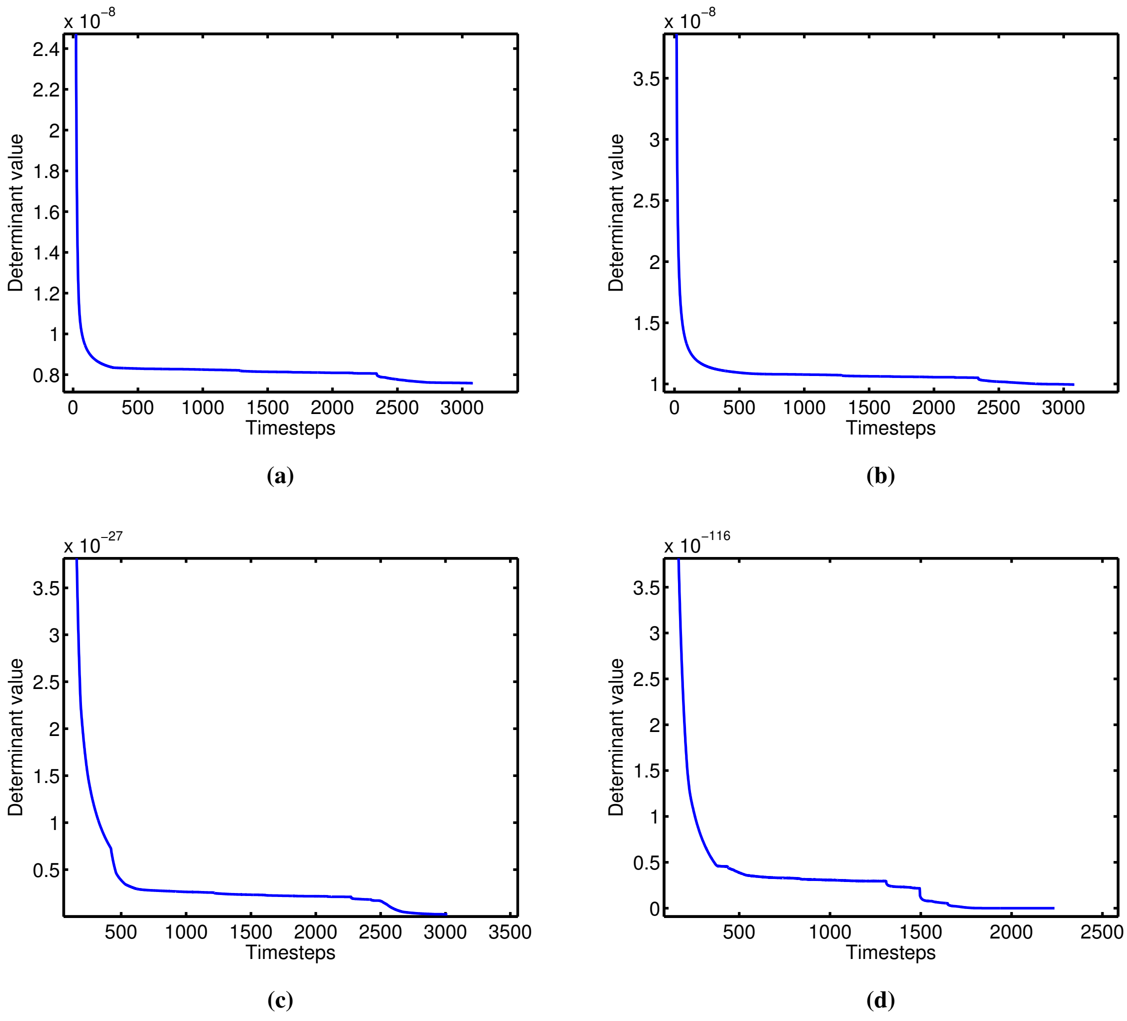


**Figure 5.3:** Determinants of various state covariance submatrices. (a) $1_{st}$ observed landmark submatrix. (b) $3_{rd}$ observed landmark. (c) $5_{th}$, $6_{th}$ and $7_{th}$ observed landmarks submatrix. (d) Submatrix of the first ten landmarks.

was used as the number of samples are relatively small, the standard deviations are not the same and the simulations can be seen as unmatched random samples due to the random noise added for each simulation. The null hypotheses that was formulated for the $t$-test was that the RMSE averages of the two methods for a particular number of feature groups are equal. The null hypothesis was rejected if the computed $t$-value was larger than critical $t$-value for a confidence level of 95%, which for the number of samples in this experiment is 1.734. Upon rejection of the null hypotheses the alternative hypothesis, that the mean of one sample is greater (and thus less accurate) than the other, was accepted. The Wilcoxon rank sum test was used to confirm the $t$-test result.

The SL sensing method was the more accurate method when fewer feature groups were available but the GL method was the more successful method when enough groups were available. As can be seen, the differences between the methods are all statistically significant except when the number of groups is equal to 4, which can be interpreted as the point where the methods are almost equal in accuracy. The Wilcoxon rank-sum test found all comparisons to be statistically significant. Landmark-based SLAM systems typically require a minimum number of landmarks to produce an accurate estimate. The standard deviation results indicated that the GL SLAM method was also the more consistent of the two methods. The improvement in accuracy and consistency was believed to be due to the fact that the error on the average of a group of sensed landmarks was lower than that of an individual landmark. The confidence interval surrounding the RMSE average are not shown as the standard deviation of most of the results are too small relative to the average to be meaningfully represented within a graph.

**Table 5.1:** RMSE averages and standard deviations of the simulated SL and GL SLAM methods with varying number of feature groups. Each group has 5 features. The difference is given as a percentage value. A difference was considered statistically significant if the $t$-value was larger than $t_{0.05} = 1.734$.

| Groups | SL avg. | SL std dev | GL avg. | GL std dev | Avg. difference | $t$-value |
|---|---|---|---|---|---|---|
| 1 | 0.0973 | 9.78E-04 | 4.3244 | 1.12E-01 | -4344.40 | 126.1019 |
| 2 | 0.0522 | 5.05E-02 | 4.2364 | 2.41E-02 | -8015.71 | 249.3076 |
| 3 | 0.0312 | 8.28E-03 | 0.0416 | 2.02E-05 | -33.34 | 4.1574 |
| 4 | 0.0371 | 1.39E-02 | 0.0452 | 1.41E-02 | -21.83 | 1.3658 |
| 5 | 0.0434 | 1.81E-02 | 0.0329 | 2.03E-05 | 24.19 | 1.9319 |
| 6 | 0.0473 | 1.29E-02 | 0.0295 | 1.48E-05 | 37.63 | 4.6147 |

## 5.4 EVALUATION USING WAYPOINT GROUND TRUTH

A series of simple test sequences were used to evaluate the complete system, which comprised the landmark extraction and data association modules (based on the PQFT saliency detection method) together with the EKF SLAM algorithm. A new dataset of Kinect movement sequences was recorded for which the ground truth at specific waypoints was known. The intrinsic camera parameters for the particular Kinect used were also determined.

### 5.4.1 Tabletop dataset

The generated dataset consisted of three sequences: *Clockwise*, *C.Clockwise* and *Extended*. The first two sequences had very simple motions, designed to focus on testing the translational estimation capability of the system and to test for any bias in the operation of the system. In the *C.Clockwise* sequence the Kinect was moved with a less smooth motion than in the *Clockwise* sequence. The purpose of the variation in motion was to determine the effect on the estimation accuracy of having a motion which was less compliant with the constant velocity movement model. The *Extended* sequence was a more complex test to evaluate both the rotational and translational estimation accuracies of the system. The ROS data recording tool, *rosbag*, was used to record data from the Kinect to *bag* files. These files contained the normal video and IR camera images, the time stamp information of these images, the calibration information for the camera as well as the transform information between the camera frames.

For the *Clockwise* sequence, a 1 m by 0.5 m rectangular path was marked out on a table. When the SLAM system was activated, the first landmarks to be sensed were included directly into the SLAM map. After that newly sensed landmarks were subjected to data association. At the beginning of the sequence the Kinect was subjected to a small movement in which an initial map was built. Thereafter the Kinect was moved from corner to corner of the marked path in a clockwise direction, constantly facing the same scene - an office table covered with various objects. The Kinect was slowly brought to a halt at each corner (so that the movement would comply with the motion model) and the 3D positional estimates for those points were recorded in a similar fashion as the tests carried out in [19]. The loop was completed three times. The loops were completed in 3 minutes and 42 seconds. With a total distance of 9 m travelled, the average speed was 0.0498 m/s. A photo of the Kinect viewing the cluttered desk can be seen in Figure 5.4.

A second sequence, *C.Clockwise*, was recorded where the camera was moved in a counterclockwise direction to investigate the possibility of a bias introduced by the specific direction of motion of the camera. The rectangular path marked out was the same as in the first sequence but there were slight variations in the scene viewed by the camera. The movement in the second test was not as smooth as in the first. The loops were completed in 3 minutes and 28 seconds. The average speed was 0.0404 m/s.

A final recorded sequence, *Extended*, consisted of a more complicated path to evaluate the estimation

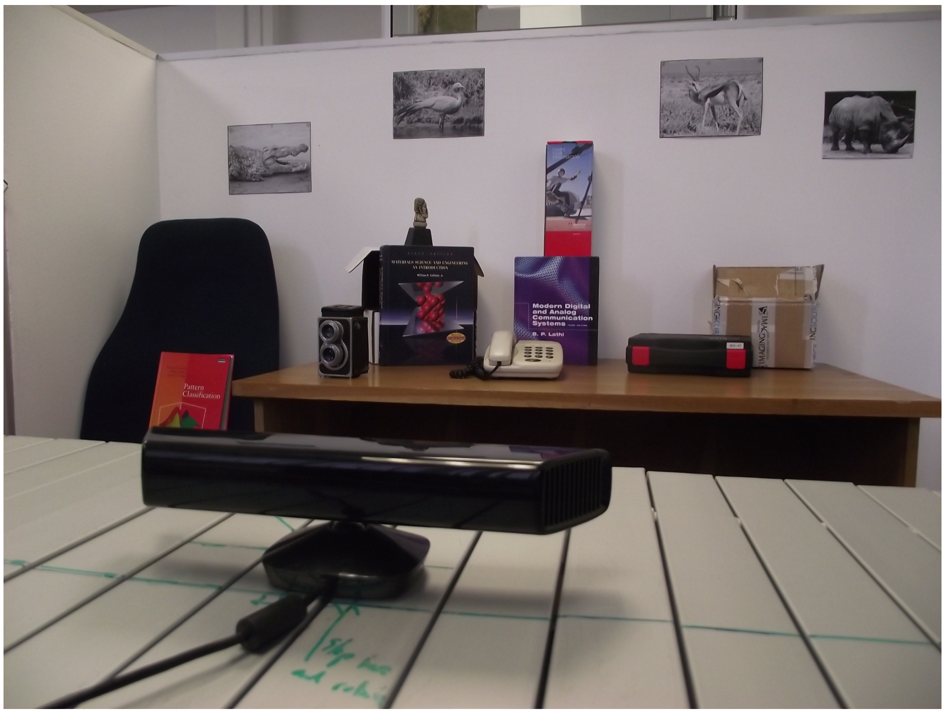


**Figure 5.4:** Setup for the recording of the Tabletop dataset. Here the Kinect can be seen placed upon the marked path and pointed toward the cluttered desk scene.

of a change in camera height and rotation. The camera was again moved in a clockwise direction while following waypoints and the camera was kept pointed towards a scene. The second waypoint was situated on a box placed on the marked table. The motion from the first waypoint to the raised second waypoint was diagonally upwards (the movement was kept consistent with the aid of a wire). To reach the third waypoint, the camera was first moved forward to clear the box, then directly downward to the table, and then directly forward to reach the waypoint. The camera was then moved to the fourth waypoint, where it was rotated 45 degrees at the waypoint. The new orientation was preserved during the motion towards the last waypoint, where it was only rotated back when it had reached the waypoint. One complete loop was 3.32 m long, so the total distance travelled was 9.96 m. The total time elapsed was 3 minutes and 48 seconds. The average speed of the Kinect in the sequence was 0.0395 m/s.

### 5.4.2 Kinect calibration

The calibration routines from the OpenCV library [74] were used to obtain the intrinsic camera parameters for the specific Kinect used. The parameters are shown in Table 5.2.

**Table 5.2:** Intrinsic parameters of the Kinect colour camera used for the experiments described in Section 5.4.

| Parameter | Value |
|---|---|
| Focal length, x axis | 520.8437 |
| Focal length, y axis | 519.1293 |
| Principal point, x axis | 322.2108 |
| Principal point, y axis | 253.8037 |

To determine the $d_s$ for the Kinect used, the real distance and camera-measured distance to a number of points in a scene were recorded. The average of the ratio between the real and measured distances was used as the scaling value. The selected points were all within 1.5 m and 2.2 m from the Kinect, which is the same range for features used in the experiments. The $d_s$ was calculated to be 1.004845.

### 5.4.3 Results and discussion

After the dataset had been recorded the data from the *bag* files were loaded into the SLAM implementation in such a way that every image was processed by the algorithm. The initial pose for the robot was set to zero. These values were chosen based on previous experience in evaluating the system. The estimation averages and standard deviations for the waypoints for the *Clockwise* sequence are shown in Table 5.3 and for the *C.Clockwise* sequence in Table 5.4. The translational estimates of the waypoints for the *Extended* sequence are shown in Table 5.5 and the rotational estimates in Table 5.6.

The estimated positions were computed by subtracting the positional estimation error after the initial mapping movement from all subsequent estimates. The purpose of the adjustment was to compensate for the lack of a map initiation procedure, as in [19] where the system was initialised using four known landmarks. To inspect whether the adjustment was necessary and beneficial, a quick test was conducted. In this test a recording was made where the Kinect was kept stationary for approximately 13 seconds, underwent some initial movements, returned to the starting position and was kept stationary again for about 13 seconds. The standard deviation for the first stationary phase was 0.0138 while for the second stationary phase it was 0.0057. The difference in standard deviations indicate that the system was more stable after an initial mapping movement and subsequent measurements based on the stabilised position would be more accurate.

**Table 5.3:** Position estimation results from the *Clockwise* sequence. All values are in metres.

| Ground Truth | | | Estimate Averages with Standard Deviation | | | | | |
|---|---|---|---|---|---|---|---|---|
| x | y | z | $x_{mean}$ | $x_{stddev}$ | $y_{mean}$ | $y_{stddev}$ | $z_{mean}$ | $z_{stddev}$ |
| 0 | 0 | 0 | 0.043 | 0.020 | 0.019 | 0.059 | -0.003 | 0.019 |
| -1 | 0 | 0 | -0.941 | 0.008 | 0.037 | 0.044 | -0.024 | 0.015 |
| -1 | 0 | 0.5 | -1.016 | 0.016 | 0.020 | 0.021 | 0.499 | 0.010 |
| 0 | 0 | 0.5 | 0.057 | 0.021 | 0.015 | 0.019 | 0.519 | 0.011 |

**Table 5.4:** Position estimation results for the *C.Clockwise* sequence. All values are in metres.

| Ground Truth | | | Estimate Averages with Standard Deviation | | | | | |
|---|---|---|---|---|---|---|---|---|
| x | y | z | $x_{mean}$ | $x_{stddev}$ | $y_{mean}$ | $y_{stddev}$ | $z_{mean}$ | $z_{stddev}$ |
| 0 | 0 | 0 | 0.014 | 0.014 | -0.020 | 0.010 | 0.000 | 0.010 |
| 0 | 0 | 0.5 | 0.058 | 0.018 | -0.044 | 0.017 | 0.513 | 0.001 |
| -1 | 0 | 0.5 | -0.992 | 0.023 | 0.014 | 0.008 | 0.520 | 0.010 |
| -1 | 0 | 0 | -0.958 | 0.018 | 0.014 | 0.018 | 0.008 | 0.009 |

As can be seen from the results in Tables 5.3 and 5.4, the system was capable of providing a realistic estimate of the position of the camera during the *Clockwise* and *C.Clockwise* sequences. In comparing the two tables, it can be seen that the system performed far better on the *C.Clockwise* sequence. The disparity is especially noticeable when the Kinect was at the bottom left corner of the table (the second waypoint in Table 5.3 and the final waypoint in Table 5.4). The overall average positional error for the *Clockwise* sequence was 0.1015, which was greater than that of the *C.Clockwise* sequence, 0.0467. the system performed better with the *C.Clockwise* sequence at a confidence level of more than 95%. The statistical significance of the comparison was evaluated as in Section 5.3.2, except that the matched-sample dependent *t*-test and Wilcoxon signed-rank test was used as a single SLAM system was tested on the same waypoints and the samples are therefore dependent. The SciPy library in Python was used and therefore the $t_{0.05}$ is not stated. The confidence intervals shown in Figure 5.5 also indicate that there is no overlap and therefore the comparison of the two sequences is valid.

The expected result had been that the system would have performed better with the slower *Clockwise* sequence. However, although the average speed during the *C.Clockwise* sequence was faster and this would indicate a more rough motion, it could be that the resultant motion was in the end more aligned with the constant velocity motion model. The difference in performance between the two sequences could be attributed to a number of factors, most probable among them a bias inherent in the system, the differences in the motion of the Kinect and variations in the sensory noise levels. Nevertheless, the standard deviation of the points indicate that these estimates, though not completely correct, were consistent.

**Table 5.5:** Position estimation results for the *Extended* sequence. All values are in metres.

| **Ground Truth** | | | **Estimate Averages with Standard Deviation** | | | | | |
|---|---|---|---|---|---|---|---|---|
| **x** | **y** | **z** | $\mathbf{x_{mean}}$ | $\mathbf{x_{stddev}}$ | $\mathbf{y_{mean}}$ | $\mathbf{y_{stddev}}$ | $\mathbf{z_{mean}}$ | $\mathbf{z_{stddev}}$ |
| 0 | 0 | 0 | -0.017 | 0.019 | 0.013 | 0.006 | 0.015 | 0.008 |
| -1 | -0.27 | 0 | -1.026 | 0.003 | -0.248 | 0.006 | 0.019 | 0.006 |
| -1 | 0 | 0.5 | -1.008 | 0.020 | -0.004 | 0.008 | 0.545 | 0.012 |
| -0.5 | 0 | 0.5 | -0.506 | 0.025 | -0.084 | 0.030 | 0.519 | 0.004 |
| 0 | 0 | 0.5 | -0.019 | 0.015 | -0.019 | 0.004 | 0.520 | 0.004 |

**Table 5.6:** Angle estimation results for the *Extended* sequence. All values are in degrees.

| **Ground Truth** | **Estimate Average** | **Standard deviation** |
|---|---|---|
| 0 | 0.755 | 0.983 |
| 0 | 1.362 | 0.136 |
| 0 | 0.829 | 0.861 |
| 45 | -40.509 | 2.295 |
| 0 | 2.732 | 0.541 |

From the standard deviations for the *Extended* sequence, shown in Table 5.5 and Table 5.6, it can be seen that the position estimates were stable, with a marked increase in error at the fourth waypoint, where the rotation occurs. The standard deviation for the angular estimates was also highest at the fourth waypoint. The larger error would indicate that the system was not as accurate at estimating rotational as translational movement. It is difficult to compare the results from the *Extended* sequence to the other two sequences due to the differences in motion paths and waypoints, but the following qualitative conclusions could be made: Firstly, a more complex motion did lead to an increase in estimation error. Secondly, the difference in error of the first two sequences was unlikely to have been caused by only a movement bias because the *Extended* sequence was also in a clockwise direction and did not produce errors as large as the *Clockwise* sequence.

### 5.4.4 Effect of camera movement and illumination

As has been previously discussed, it was believed that the difference in estimation error produced by the system for the *Clockwise* and *C.Clockwise* sequences could be produced by bias, different types of Kinect movement and variations in sensory noise levels. System bias to movement direction had already been discounted as unlikely, therefore it was required to investigate the last-mentioned two sources.

The cause of differences in the way the Kinect was moved was self-evident: it was the person moving the Kinect. To determine what the effect of a different movement would be, a new sequence, *Clockwise*2 was recorded, in the same manner as *Clockwise*. For the new sequence, the Kinect was handled more gently at an average speed of 0.0395 m/s. The RMSE and other ATE statistics of the known waypoints are shown in Table 5.7. As can be seen, there was a decrease when the Kinect was handled in a smoother fashion. However, the comparison was only statistically significant at a 90% confidence level (computed using the methods described in Section 5.4.3) and further experimentation is required to investigate the effect of motion on the accuracy of the system. The overlap in confidence intervals shown in Figure 5.5 further illustrates this point.

**Table 5.7:** Statistics of the absolute trajectory error produced by the system for the *Clockwise* and *C.Clockwise*2 sequences.

| Statistic | Clockwise | Clockwise-2 |
|---|---|---|
| RMSE | 0.1063 | 0.0855 |
| Mean | 0.1015 | 0.0787 |
| Median | 0.1129 | 0.0824 |
| Std Dev | 0.0332 | 0.0349 |
| Minimum | 0.048 | 0.0139 |
| Maximum | 0.1457 | 0.1246 |

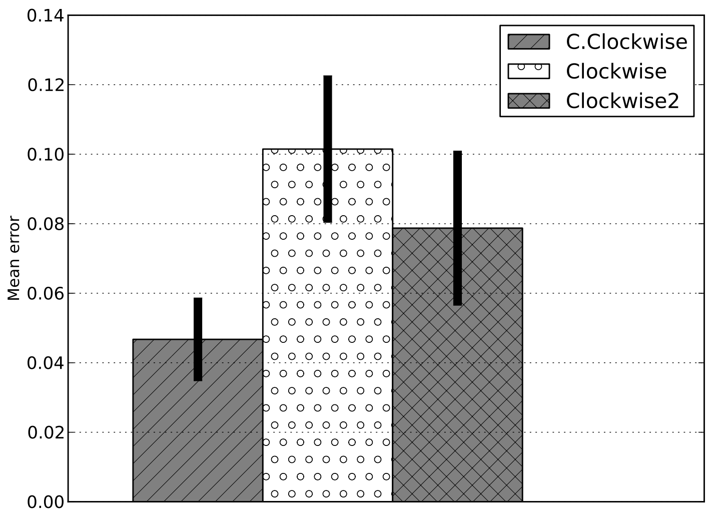


**Figure 5.5:** 95% Confidence intervals on the mean error for the C.Clockwise, Clockwise and Clockwise2 sequences. The overlap in intervals between the latter two sequences is evident.

The cause of variations in sensory noise levels was less obvious. Investigation of the images from the sequences shows that the *C.Clockwise* sequence was more brightly lit than the *Clockwise* sequence. The Kinect standard video camera automatically adjusts its exposure level to suit the environment viewed. Lower levels of illumination would have lead to a higher exposure time, which would have lead to higher levels of motion blur in the images. The *Clockwise*2 illumination levels were very similar to those found in *Clockwise*. To determine how the sensor noise levels affected the system, the expected noise covariance values were adjusted from the initial value of 0.002. The value was chosen after some experimentation. These results are shown in Table 5.8 and indicated that when the system was adjusted to handle the greater sensor noise the results increased in accuracy but only up to a point. The increase in accuracy from expected noise levels of 0.002 to 0.003 was found to be statistically significant, while the difference between levels of 0.002 and 0.004 was too small. For comparison purposes, the same was done for the better illuminated *C.Clockwise* sequence and shown in Table 5.9. Here it can be seen that adjusting the expected sensor noise led to a slight increase in error (which was found not to be statistically significant), as there was no actual increase in sensor noise. From the results of these expected noise level experiments, it can be seen that the system is sensitive to the amount of illumination in a scene. However, due to complex manner in which the expected noise covariance values affects the rest of the system, it is difficult to predict which value will produce the optimal result.

**Table 5.8:** Statistics of the absolute trajectory error produced by the system for the *Clockwise* sequence, for differing values of sensory noise covariance.

| **Statistic** | **Q = 0.002** | **Q = 0.003** | **Q = 0.004** |
|---|---|---|---|
| RMSE | 0.1063 | 0.0905 | 0.1124 |
| Mean | 0.1015 | 0.0857 | 0.1047 |
| Median | 0.1129 | 0.0829 | 0.108 |
| Std Dev | 0.0332 | 0.0303 | 0.0426 |
| Minimum | 0.048 | 0.132 | 0.1903 |
| Maximum | 0.1457 | 0.0347 | 0.0334 |

**Table 5.9:** Statistics of the absolute trajectory error produced by the system for the *C.Clockwise* sequence, for differing values of sensory noise covariance.

| **Statistic** | **Q = 0.002** | **Q = 0.003** | **Q = 0.004** |
|---|---|---|---|
| RMSE | 0.05 | 0.0655 | 0.0593 |
| Mean | 0.0467 | 0.0599 | 0.0557 |
| Median | 0.051 | 0.0619 | 0.0559 |
| Std Dev | 0.0187 | 0.0277 | 0.0214 |
| Minimum | 0.0134 | 0.0099 | 0.021 |
| Maximum | 0.0721 | 0.1039 | 0.0932 |

## 5.5 EVALUATION USING COMPLETE PATH GROUND TRUTH

The previous experiments, though useful to indicate whether the system was working as desired, were somewhat contrived in that the camera was stopped at each waypoint. The type of movement described artificially influenced the position estimates and did not represent how the system would be used in a real application. Therefore a test using a dataset where the complete information regarding the ground truth was available was used to provide a more in-depth and realistic evaluation of the system. The SLAM system was implemented using all three of the saliency mapping methods to determine which method was most suitable.

### 5.5.1 Freiburg 2 dataset

As the current implementation used a Kinect sensor, the dataset developed by the Technical University of Munich, described in [78] and available from [76], could be used to obtain a complete ground truth evaluation of the system. The dataset contained various recordings from a Kinect as it was being moved through an indoor environment. These *bag* files consisted of the normal camera images, depth camera images, the time stamp information for these images, the transformation between the two cameras and the IMU data from the Kinect. Ground truth for the path of the Kinect was recorded at 100 Hz by a MotionAnalysis motion capture system and could be downloaded separately. The dataset has been used to evaluate a number of real time point cloud registration systems [29] and a point cloud based SLAM implementation [30]. The creators of the dataset also provided an error measurement tool based on the ATE as well as the work in [94].

The dataset was divided into two sub-datasets called Freiburg 1 and Freiburg 2. Sequences from the latter were used, as the camera motions at the start of these sets were slow enough to test the current implementation, which assumed zero initial velocity. The following sequences were used: *XYZ*, *RPY* and *Desk*. The first two had very simple, slow motions to test the operation of a system for translational and rotational motion, respectively. The total length of camera motion for these datasets were 7.029 m and 1.506 m. The *Desk* dataset had a Kinect being moved around a large desk environment to complete a loop. Of all the sequences that had been used so far, *Desk* most resembled an actual usage case of the SLAM system, as the Kinect was moved in a natural fashion, not limiting itself to specific waypoints or movement styles. As the loop was quite large, it offered an opportunity to evaluate the loop closure capability of the system. The total length travelled by the Kinect was 18.88 m. The camera parameters in Table 5.10, taken from [76], were used.

As has been seen in Section 5.4.4, changing the expected EKF noise covariance values had a noticeable effect on the parameters. Therefore, the parameters for each Freiburg 2 sequence were set after evaluating the average rotational and translational velocities of the sequences and through experimentation with various values.

### 5.5.2 Results and discussion

The data from the downloaded Freiburg 2 *bag* files were input into the SLAM implementation in such a way that every image was processed by the algorithm. The experiments were conducted using the

**Table 5.10:** Intrinsic parameters of the Kinect colour camera and scaling factor value for the Freiburg 2 sequences.

| Parameter | Value |
|---|---|
| Focal length, x axis | 520.9 |
| Focal length, y axis | 521.0 |
| Principal point, x axis | 325.1 |
| Principal point, y axis | 249.7 |
| Scaling factor | 1.031 |

SR, PFT and PQFT saliency detection methods. All the landmark extraction and data association routines as described in Chapters 3 and 4 were applied for each method. The initial pose for the algorithm was set according to the ground truth file for each sequence. The ATE was computed for each sequence and method and various statistics on it were extracted. The number of landmarks tracked were also reported. The time reported was not an absolute measure, but served as an indication of the relative computation time. Output video images were generated, which showed the landmarks extracted from an input image to aid analysis of the results further.

The automated analysis tool available from [76] was not used in this evaluation. The tool attempted to align the estimated positions to the ground truth, which was not necessary for this evaluation, as the ground truth values had been used to set the initial positions for each sequence. The statistical significance of the comparisons between SLAM systems was investigated using the same paired-sample tests as described in Section 5.4.3, as different methods are applied to the same datasets and the samples are considered dependent. The results of these tests are presented in Section 5.5.3. However, to summarise it can be stated that all comparisons are statistically significant up to a 95% confidence level, except for the comparison between the SR and PQFT methods for the *Desk* sequence.

#### 5.5.2.1 XYZ sequence

The results for the *XYZ* sequence are provided in Table 5.11. As can be seen, the SLAM system using the PQFT saliency detector outperformed the other two methods for all measured values except the maximum error. In addition, it did so in approximately the same computational time with the same number of landmarks. From this it could be conjectured that the PQFT method was better at generating saliency maps (which would be in agreement with the results reported in [65]) that led to the extraction of reliable landmarks.

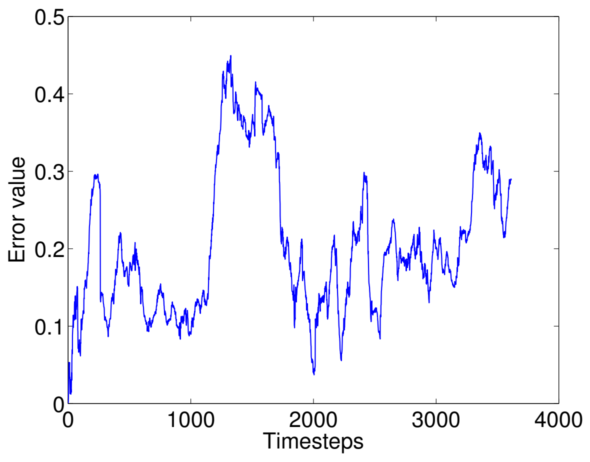
**Table 5.11:** ATE statistics of the SLAM system using different saliency detection methods for the $XYZ$ sequence.

| **Statistic** | **SR** | **PFT** | **PQFT** |
|---|---|---|---|
| RMSE | 0.224 | 0.190 | 0.166 |
| Mean | 0.203 | 0.170 | 0.142 |
| Median | 0.184 | 0.148 | 0.124 |
| Std Dev | 0.093 | 0.084 | 0.085 |
| Maximum | 0.450 | 0.434 | 0.497 |
| Time | 4:37 | 4:31 | 4:38 |
| Landmarks | 22 | 22 | 22 |

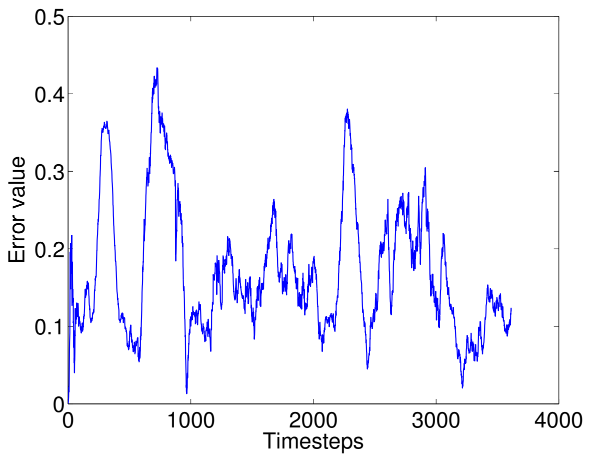
Figure 5.6 shows the ATE graphs of the three saliency method implementations. Although the different implementations performed well across the entire sequence, there were specific instances in which large errors occurred, which required further investigation. Figure 5.6c shows that the largest error occurred for the PQFT around time step 2800, where the camera underwent a large z-axis motion with regard to the frame of the camera. A movement along the z-axis translated into a movement toward and away from the main source of landmarks. The deduction was made that the landmark extraction system was not capable of handling such large changes in image scale. The large maximum error as seen in Table 5.11 could also be attributed to the system not being invariant to scale changes.

The SR detector did not appear to suffer from the same problem, as seen in Figure 5.6a. The reason for this was that the SR method detected smaller regions, which were more invariant to scale. A side-by-side comparison, seen in Figure 5.7, shows the differences in the extracted landmark sizes. The PQFT method tended to identify larger regions as being salient because it used additional colour information. The SR method isolated separated, more compact regions. Though not shown, the PFT method finds regions that were well separated but not to the same extent as the SR method. The PQFT method was perhaps not as invariant to scale changes as either the SR or PFT method.

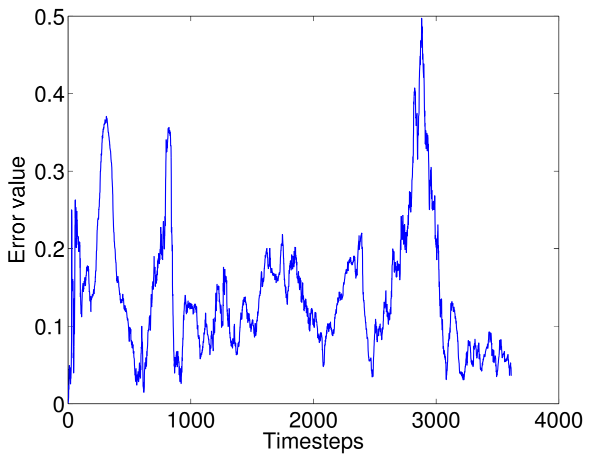
The largest error for the SR method occurred at approximately time step 800, which involved a simple upward motion of the camera showing less of the office table area. The SR method was not able to find



**(a)**

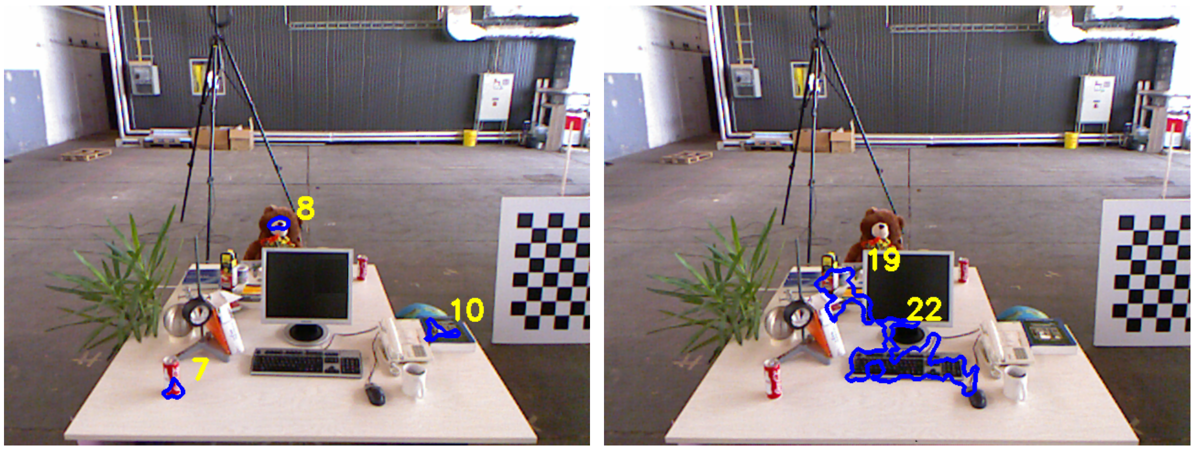


**(b)**



**(c)**

**Figure 5.6:** ATE graphs of the SLAM system using the (a) SR, (b) PFT and (c) PQFT saliency detection methods for the $XYZ$ sequence.



(a) (b)

**Figure 5.7:** Landmark extraction showing the effect of scale on the (a) SR and (b) PQFT saliency method.

enough sizable landmarks in this region and failed to provide the EKF with information to update the state estimate. Thus, although smaller sized landmarks were useful to avoid errors caused by image scaling, these landmarks could also be difficult to detect reliably.

#### 5.5.2.2 RPY sequence

The results for the *RPY* sequence can be seen in Table 5.12. Here it can be seen that the SR method was by far the better method for this sequence while the PFT marginally outperformed the PQFT. The dominance of the SR method strongly contradicted the results from the $XYZ$ sequence. Examining the ATE graphs of the methods, shown in Figure 5.8, it can be seen that a major error occurred at time step 1000 for both the PQFT and PFT methods. At this time the camera underwent a yaw motion and views a large calibration checkerboard, as seen in Figure 5.9. The particular scene produced low-quality landmarks for two reasons. Firstly, the landmarks found on the checkerboard were ambiguous because the repetitive pattern affected the feature descriptors, while the proximity of the landmarks prevented the filters based on distance to exclude the incorrect matches. Secondly, some of the landmarks were detected beyond 3.5 m, which was the limit of accurate depth measurement for the Kinect. The landmark below the transparent plastic cover marked by a -1 was approximately 5 m away from the Kinect. The identified low-quality landmarks affected the accuracy with which the pose was tracked.

The ATE graph for the SR method shows no such error occurring at time step 1000. The output video at that time was examined and it was seen that the SR method was failing to detect any landmarks. As no landmarks were being matched or added to the state vector, the EKF was not updated with new information at the time. In the sequence the camera was rotated back along the same path and thus the EKF could continue to use the same estimate without a too large error. If the camera had not returned to the same position, the EKF would most likely have lost track of the camera pose. Therefore the success of the SR method was specific to the *RPY* sequence.

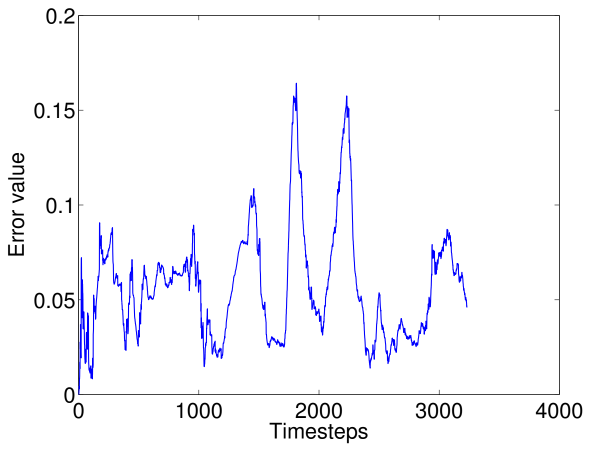
**Table 5.12:** ATE statistics of the SLAM system using different saliency detection methods for the *RPY* sequence.

| **Statistic** | **SR** | **PFT** | **PQFT** |
|---|---|---|---|
| RMSE | 0.064 | 0.140 | 0.149 |
| Mean | 0.057 | 0.113 | 0.120 |
| Median | 0.055 | 0.081 | 0.094 |
| Std Dev | 0.030 | 0.083 | 0.088 |
| Maximum | 0.164 | 0.354 | 0.420 |
| Time | 4:12 | 4:41 | 5:57 |
| Landmarks | 30 | 76 | 113 |

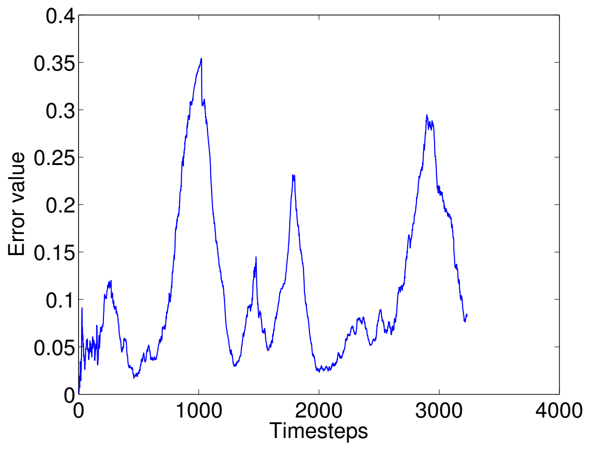
#### 5.5.2.3 Desk sequence

The results of the three different saliency methods for the *Desk* sequence can be seen in Table 5.13. Here the PQFT implementation again outperformed the other methods but at a far greater computational time cost. The greater computation time could easily be attributed to the large number of landmarks that were being tracked. The margin between the SR and PQFT implementations was relatively small, although the difference in standard deviation and maximum error was noticeable (the statistical significance of this comparison is evaluated in Section 5.5.3) The PFT method performs substantially worse than the other methods.

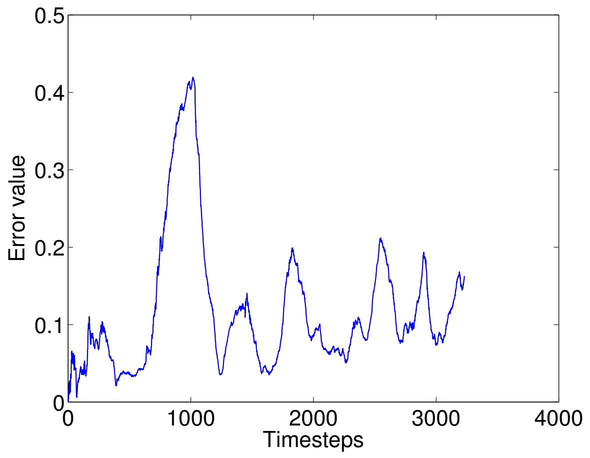
The ATE graphs and the estimated camera paths, shown in Figure 5.10, show that the error decreased as the camera reached the starting point again, for all three methods.The error reduction indicated that loop closure had occurred and that the system had corrected the pose and map estimates. A screen capture produced by the PQFT method at the end of the sequence, as seen in Figure 5.11, seemed to



**(a)**



**(b)**



**(c)**

**Figure 5.8:** ATE graphs of the SLAM system using the (a) SR, (b) PFT and (c) PQFT saliency detection methods for the *RPY* sequence.

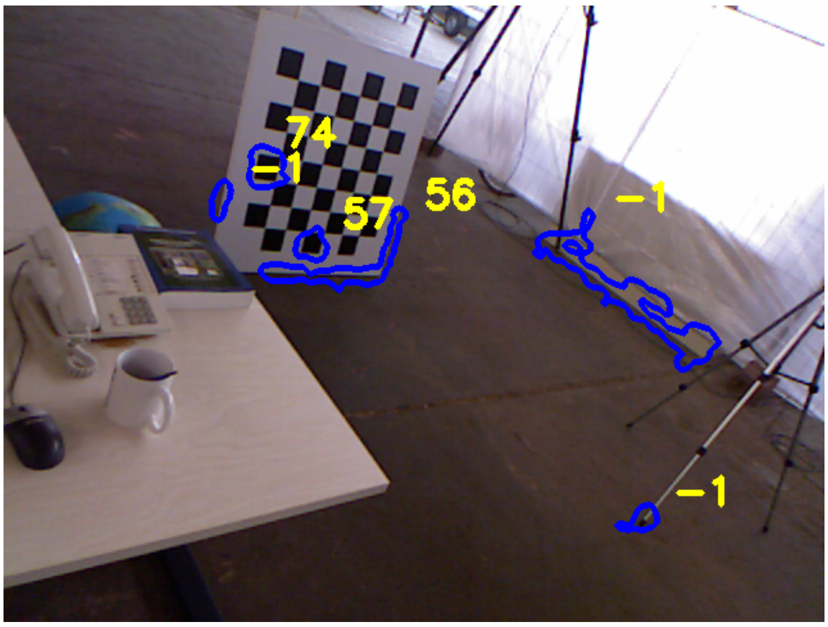


**Figure 5.9:** Screen capture at approximately time step 1000 of PQFT implementation for the *RPY* sequence. Outlines indicate matched landmarks, while the numbers indicate the position of the landmarks within the state vector. Note the checkerboard and distance to some of the landmarks.

substantiate this claim. Similar results can be seen for the other methods. A number of landmarks extracted at the start of the sequence were reobserved (as indicated by the yellow number, which signifies the landmark index within the EKF state vector). The occurrence of loop closure when only data association had been used was a good result for such a large loop.

The SR method implementation produced a smaller error at the end of the sequence than the SLAM system using the PQFT method, but it was less successful at tracking the camera during the overall movement. The PQFT implementation suffered from less drift in the pose estimation and did close the loop equally effective. All of these results could indicate that data association was more effective with the landmarks extracted by the SR method but that the positional information provided by the PQFT method landmarks was of better quality.

The SLAM implementation using the SR method was very fast in comparison to the other methods. During testing it was noted that many video frames were not processed as not enough landmarks were found, which would have benefited the computational speed but would have hampered the accuracy of the pose estimate. The difference in the accuracy of the SR and PQFT methods could also have been caused by frames that were not processed.

All three systems encountered a large error at approximately time step 800. The cause of this error was not as obvious as in previous cases, though it seemed that the camera view was of the highly cluttered desk, which perhaps could have led to a number of incorrect landmark matches. The PQFT method SLAM implementation was able to recover from this error, while the PFT method never fully recovered. The PFT performs poorly overall because it could not correct for the error sufficiently.

**Table 5.13:** ATE statistics of the SLAM system using different saliency detection methods for the *Desk* sequence.

| **Statistic** | **SR** | **PFT** | **PQFT** |
|---|---|---|---|
| RMSE | 0.786 | 1.170 | 0.740 |
| Mean | 0.618 | 1.010 | 0.628 |
| Median | 0.499 | 1.021 | 0.585 |
| Std Dev | 0.485 | 0.590 | 0.391 |
| Maximum | 1.910 | 2.323 | 1.377 |
| Time | 3:53 | 9:07 | 14:29 |
| Landmarks | 101 | 188 | 227 |

### 5.5.3 Statistical significance

All comparisons between the SR, PFT and PQFT methods were subjected to a matched-pairs $t$-test as well as a Wilcoxon rank-sum test. All of the comparisons where found to be statistically significant with at least a 95% confidence level, except for the comparison between the PQFT and SR method for the *Desk* sequence. Figure 5.12 shows the 95% confidence intervals of the mean error for the different methods on the *XYZ* and *RPY* sequences. As can be seen, there are no overlaps in intervals which further validates the comparisons between the methods. The confidence intervals for various statistics of the ATE of the SR and PQFT methods on the *Desk* sequence is shown in Figure 5.13. Here it can be seen that there is an overlap in the intervals of the mean estimate, that the PQFT has the smaller RMSE and the SR has the lower Median. Furthermore, during the matched-pairs $t$-test for this comparison is was found that the null hypothesis (that both methods have the same mean) can only be rejected with a 63% confidence level, which is far below the threshold of 95%. Therefore it cannot be stated whether either method is more suited to the *Desk* test set and that more experimentation is required for determining the superior method.

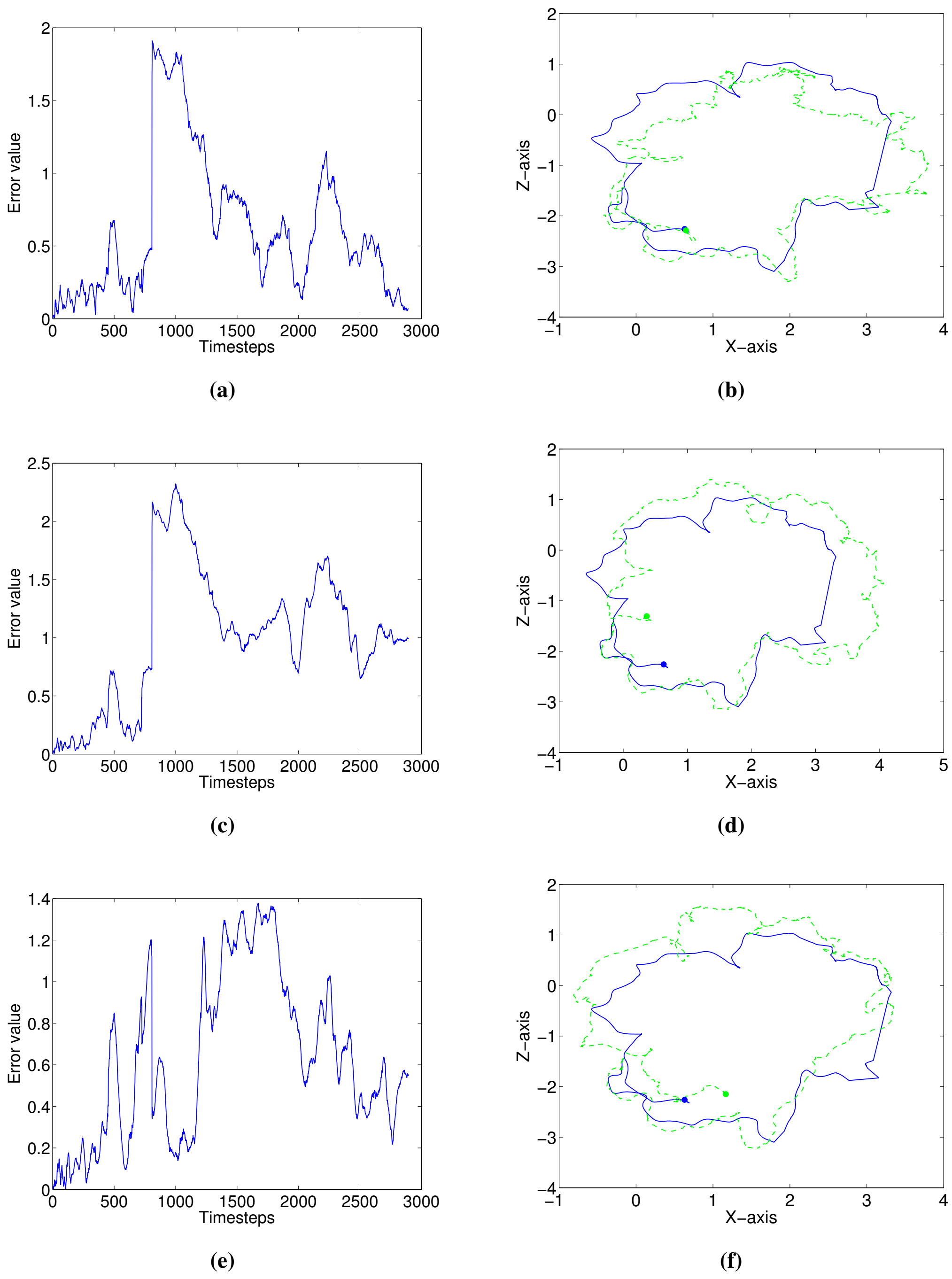


**Figure 5.10:** ATE graphs and estimated paths of the SLAM system for the *Desk* sequence. (a) and (b) are from the SR implementation, (c) and (d) are from the PFT implementation while (e) and (f) are from the PQFT implementation. The true path of the camera is indicated by the solid blue line while the estimated path is indicated by the dashed green line. The large dot is the end of each path.

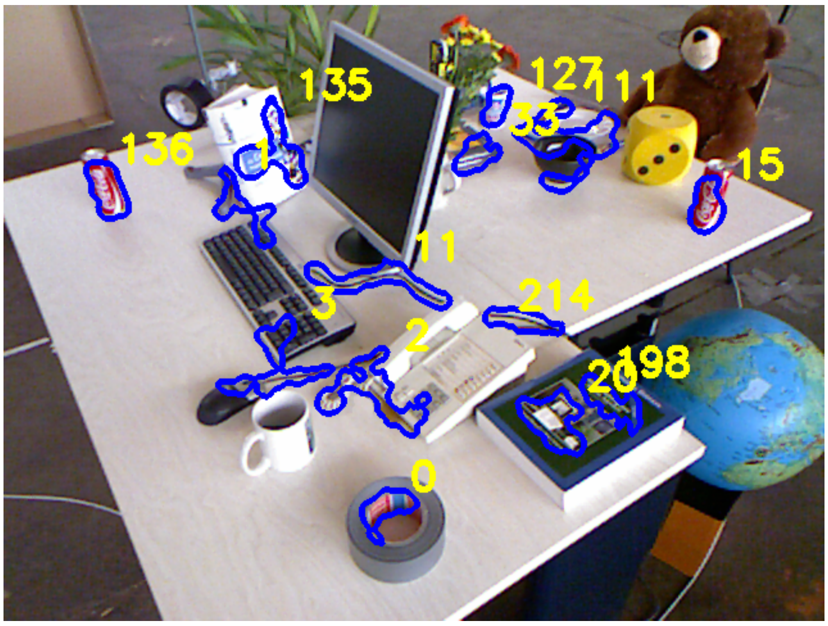


**Figure 5.11:** Screen capture at the end of the *Desk* sequence. Low numbers indicate that old landmarks have been reobserved and thus loop closure has occurred.

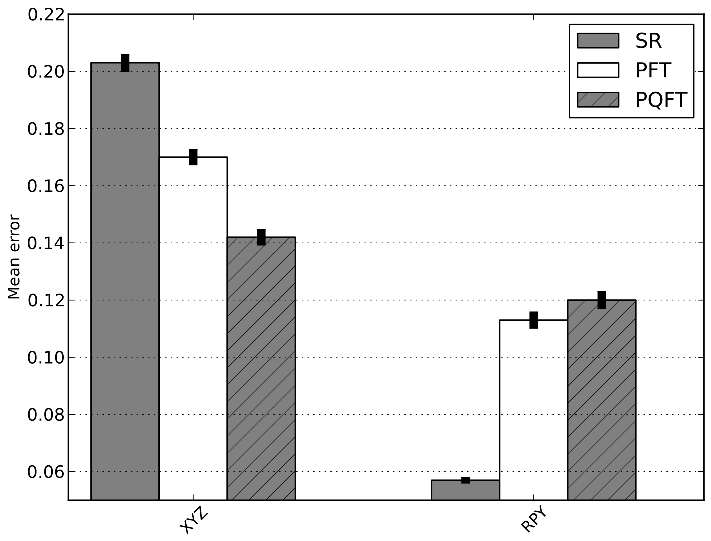


**Figure 5.12:** 95% Confidence intervals showing no overlap between the mean errors of the different SLAM methods for the *XYZ* and *RPY* sequences.

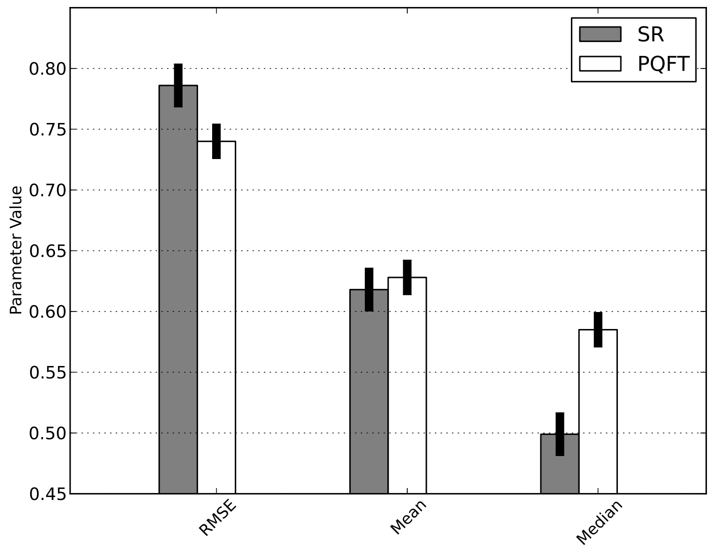


**Figure 5.13:** 95% Confidence intervals for different statistical parameters of the SR and PQFT SLAM systems for the *Desk* sequence. The PFT method is not shown as it performed substantially worse than the other two methods.

## 5.6 COMPARISON BETWEEN SINGULAR FEATURE AND GROUPED FEATURES AS LANDMARKS

A comparative study was conducted to examine the advantages and disadvantages of using the newly developed landmark instead of singular feature landmarks in SLAM. The SLAM system using the PQFT saliency detection method, hereafter referred to as grouped landmark SLAM or GL SLAM, was used. A singular landmark or SL SLAM implementation was developed using many of the routines and formulations used for the GL SLAM implementation. The use of singular landmarks was somewhat similar to the SLAM implementation in [87]. The same EKF SLAM framework as discussed in Sections 3.2 and 4.5 was used. The standard OpenCV FAST keypoint detection and SURF descriptor extraction routines were used for landmark extraction. A FAST keypoint threshold filter was used to eliminate ambiguous features. Potential matches between newly detected keypoints and previously mapped landmarks were first filtered using the methods described in Section 4.7.1, except that the line length variance was not evaluated. Matches were found using a brute force matching technique and the best matches were chosen using the method described in Section 4.7.2. Landmarks were also deleted using the approach described in Section 4.8.

A direct comparison between the two SLAM implementations was difficult even when using the same dataset: There were usually a number of parameters that needed to be set for each implementation and these parameters affected each implementation differently. The most notable difference in the parameter settings was that the SL SLAM FAST keypoint threshold and landmark deletion parameters were set much higher than for the GL SLAM implementation, as far more landmarks were extracted in SL SLAM and could have overwhelmed the system.

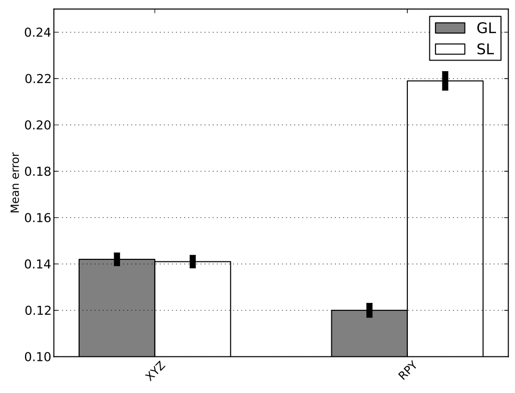
### 5.6.1 Results and discussion

The same procedures described in Section 5.5.2 were used to evaluate the SL SLAM implementation. The GL SLAM results from the same section were used for the comparison. Various statistics on the resultant ATEs of both implementations were computed. In addition, the final number of tracked landmarks and computation time were also recorded. The computation time reported was the time it took the system to analyse all the images in a sequence. Although real-time operation was not a goal of this implementation, the completion time provided a useful measure to evaluate the differences between the systems. Statistical significance was evaluated using the tests described in Section 5.4.3. The results of this evaluation can be seen in Table 5.14.

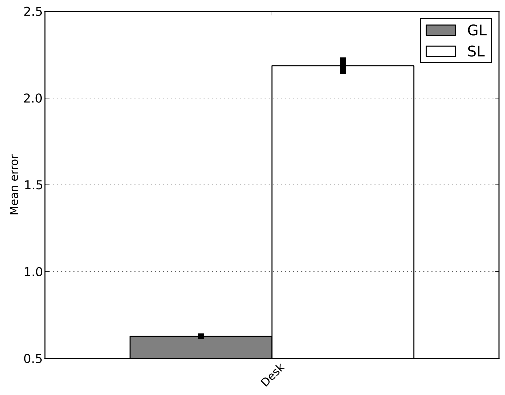
**Table 5.14:** Statistics of the ATE produced by the SL and GL SLAM systems for the Freiburg 2 sequences.

| | **XYZ** | | **RPY** | | **Desk** | |
|---|---|---|---|---|---|---|
| **Statistic** | **GL** | **SL** | **GL** | **SL** | **GL** | **SL** |
| RMSE | 0.166 | 0.164 | 0.149 | 0.248 | 0.740 | 2.528 |
| Mean | 0.142 | 0.141 | 0.120 | 0.219 | 0.628 | 2.186 |
| Median | 0.124 | 0.119 | 0.094 | 0.211 | 0.585 | 2.539 |
| Std Dev | 0.085 | 0.084 | 0.088 | 0.117 | 0.391 | 1.270 |
| Maximum | 0.497 | 0.466 | 0.420 | 0.579 | 1.377 | 4.204 |
| Time (min:sec) | 4:27 | 4:17 | 5:57 | 4:10 | 14:29 | 55:27 |
| Landmarks | 22 | 35 | 113 | 78 | 227 | 299 |

SL SLAM performed slightly better and more efficiently than GL SLAM for the *XYZ* sequence. However, the increase in performance is not large enough to be considered statistically significant. More landmarks were tracked by SL SLAM, which is to be expected. The *XYZ* sequence was very simple,



**(a)**



**(b)**

**Figure 5.14:** 95% Confidence intervals of the mean errors for the SL and GL SLAM systems for the (a) *XYZ* and *RPY* sequences and (b) *Desk* sequence.

with very basic motions and a consistent view of a texture-rich cluttered desk. For the *RPY* sequence, GL SLAM outperformed SL SLAM in terms of the estimation accuracy by a prodigious margin which is statistically significant. However, GL SLAM tracked fewer landmarks than SL SLAM but still took longer to complete the sequence. The discrepancy in computational time is further investigated in Section 5.6.2. From the results the deduction was made that the additional overhead of the GL SLAM landmark extraction and data association consumed more time than the reduction in EKF SLAM computations owing to the lower number of landmarks.

Figure 5.15 shows the ATE of SL SLAM for the *RPY* sequence. The graph shows that a large estimation error was encountered at time step 1000, similar to that seen in Figure 5.8c for GL SLAM. However, an even larger error was encountered at approximately time step 250. Figure 5.14 shows the confidence intervals for the mean errors of the SL and GL SLAM methods for the all the complete ground-truth sequences. Here it can be seen that there is a definite overlap for the *XYZ* sequence while there is no overlap for the *RPY* sequence, substantiating the statistical significance of the latter comparison. As can be seen in Figure 5.14b, the difference in performance of SL and GL SLAM methods for the *Desk* sequence is evidently statistically significant.

A screen capture for both implementations at this point can be seen in Figure 5.16. Before time step 250, the camera pans far enough to the left that the camera view is of only the large box lid and the cord on the floor, as seen in the left hand side of Figure 5.16. There were a number of keypoints in this part of the *RPY* sequence, but because of the homogeneous texture of the scene these keypoints were all of too low a response and were filtered by the SL SLAM implementation. Even the plant and desk objects seen in Figure 5.16 produced keypoints with responses falling below the threshold value. As no suitable keypoints were detected, the EKF was not updated with sensory information. Thereafter the EKF could not adapt to the changes in rotational velocities and this caused the large error. In contrast, the GL SLAM implementation detected enough landmarks to maintain a good pose estimate.

The capability of GL SLAM to operate in this problematic section of the *RPY* section was attributed to two causes. Firstly, the GL SLAM landmark extractor used adaptive filtering. Secondly, more keypoints could be detected because the threshold for FAST keypoints could be set far lower. A lower threshold value could be used because the grouping of landmarks allowed for lower quality features to be used, as individual keypoint mismatches did not have as catastrophic an impact in GL SLAM as in SL SLAM.

To determine whether lowering the keypoint response threshold would improve the results of the SL SLAM implementation, the ATE was computed for the first 350 time steps of the *RPY* dataset for different threshold settings. The original threshold was 150. The statistics for these results, seen in Table 5.15, indicated that more keypoints could be detected if the threshold was lowered, which led to more accurate estimation results. The same threshold values were evaluated using the whole of the *XYZ* sequence. The results are shown in Table 5.16 and showed an increase in error as the threshold was lowered despite an increase in the number of landmarks tracked. The results shown in Tables 5.15 and 5.16 emphasised the utility of adaptive thresholding.

**Table 5.15:** ATE statistics of the GL SLAM system and SL SLAM system for the first 350 images of the *RPY* sequence. The number after SL indicates the FAST keypoint response threshold used for SL SLAM

| **Statistic** | **GL SLAM** | **SL 150** | **SL 130** | **SL 110** |
|---|---|---|---|---|
| RMSE | 0.066 | 0.357 | 0.268 | 0.101 |
| Landmarks | 63 | 16 | 23 | 38 |

The results for the *Desk* sequence show that SL SLAM was incapable of dealing with the large loop of the sequence. The singular feature implementation completely failed to provide significant

**Table 5.16:** ATE statistics of the GL SLAM system and SL SLAM system for the *XYZ* sequence. The number after SL indicates the FAST keypoint response threshold used for SL SLAM

| **Statistic** | **GL SLAM** | **SL 150** | **SL 130** | **SL 110** |
|---|---|---|---|---|
| RMSE | 0.166 | 0.164 | 0.220 | 0.404 |
| Landmarks | 22 | 35 | 55 | 110 |

loop closure, as can be seen in Figure 5.17. Here it can be seen that the ATE increased almost unceasingly. The dips in error at about time steps 2000 and 2800 could perhaps have resulted from correct matching, but the system had lost track of the camera pose and could not provide enough correction to recover it. Such a loss of tracking led to a steep increase in the number of landmarks, as seen in Table 5.14. The increase was a result of the previously mapped landmarks not being matched to the newly detected landmarks because of the difference in distance between matches, which was caused by the erroneous pose estimate. The large number of landmarks led to the very long completion time. The large difference in performance of the SL and GL SLAM methods for the *Desk* sequence is of course statistically significant at a 95% confidence level.

In summation, SL SLAM could provide better results for simple environments with rich texture and when the Kinect was moved in a very simplistic manner. GL SLAM, on the other hand, could operate in environments with lower quality keypoints and with more dynamic and extensive Kinect movements because of the adaptive thresholding that it applied and the more extensive data association method that it employed.

### 5.6.2 Computational analysis

The GL SLAM system produced more accurate positional estimates than the SL SLAM system. However, it was found to have taken much longer to process the *RPY* sequence even though it had tracked fewer landmarks. Although the focus of the implemented system was not on computational performance, it will always be a consideration in the deployment of robotic systems and therefore deserved attention.

In previously reported results, the total time the system took to process the data within a sequence was reported as a measure of the computational cost of an implementation. To provide a more in-depth analysis of the computational performance, the *Callgrind* profiler tool from the *Valgrind* instrumen-

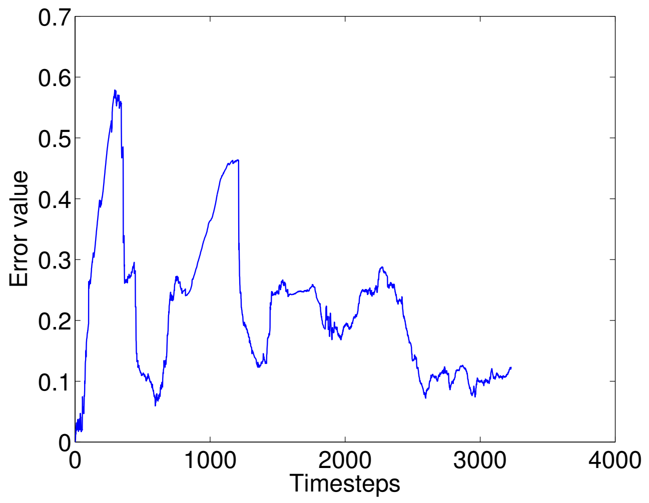


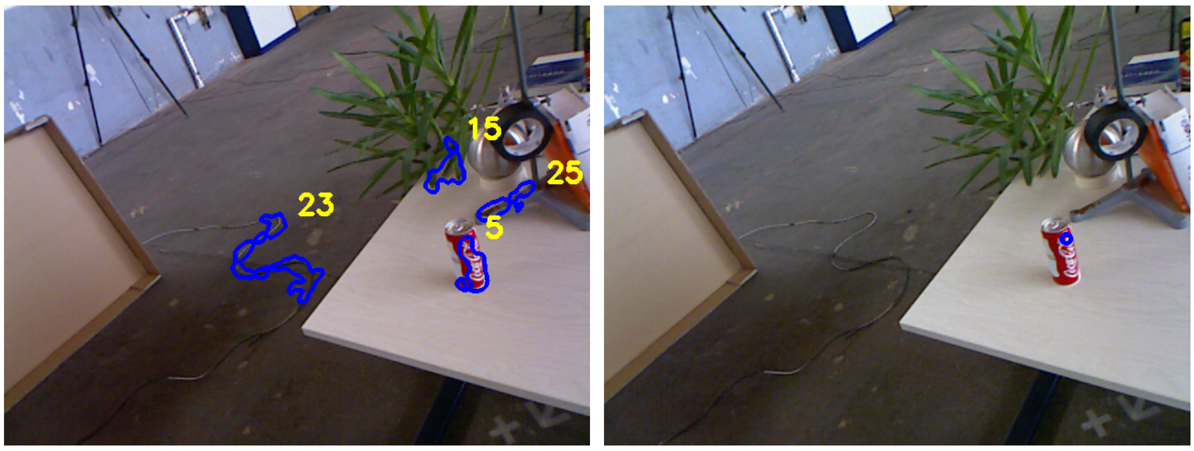
**Figure 5.15:** ATE graph of SL SLAM for the *RPY* sequence. Note the large error at step 250, which corresponded to a region containing keypoints with low responses.

tation framework was used [111]. *Callgrind* analyses the behaviour of an executable binary and records the call history of the functions within the program. The call history contains the exact number of instructions executed, the number of function calls and the relationship between functions. The information within the call history can be used to compute the relative length of time spent within a function. Unfortunately, the emulation process slows down the execution of the binary significantly. Therefore, the SLAM implementations were only evaluated for the first 50 time steps of the *RPY* sequence. The *Kcachegrind* application [112] was used to visualise and organise the text-based results produced by *Callgrind*.

The SL SLAM and GL SLAM systems both used the same EKF estimation algorithm. Therefore, the difference between the systems was the type of landmark used, which determined the landmark extraction and data association routines implemented. By comparing the two systems, the additional computational cost of the saliency-grouped landmark was investigated.

#### 5.6.2.1 Results

By analysing the results produced by the *Callgrind* profiler tool, it was seen that 70.52% of the total time was spent on the landmark extraction step and 26.57% on the data association step in the GL SLAM system. The GL SLAM system spent a total of 97.09% of the total time extracting and matching landmarks. The SL SLAM system only spent 47.63% on these steps. The sizeable



(a) (b)

**Figure 5.16:** Screen captures of the SL and GL SLAM at around time step 250 for the *RPY* sequence showing how scenes containing low-quality keypoints affected both algorithms.

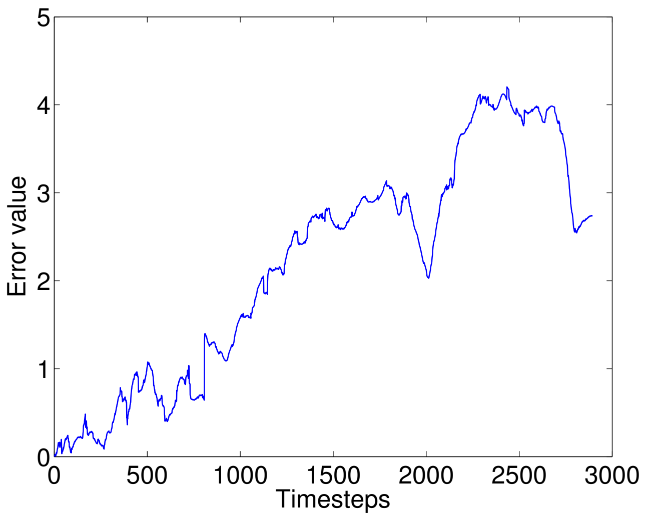


**Figure 5.17:** ATE graph of SL SLAM for the *Desk* sequence. The error increased almost constantly.

difference is to be expected given the given the greater complexity of the saliency-grouped landmark with regards to the simpler, single feature landmark used by SL SLAM.

#### 5.6.2.2 Optimisation

The computational cost of the saliency-grouped landmark was large, but given the increases in accuracy and robustness it would still be desirable to use it for SLAM. There were several ways that could be used to optimise the landmark extraction and data association steps associated with the new landmark. More advanced programming techniques or hardware, such as general-purpose computing on graphics processing units, multi-threaded programming or higher-end processors, could have been used. More efficient, but possibly less accurate, procedures and algorithms could have been used. Finally, the general implementation of the system could have been improved.

The GL SLAM *Callgrind* output was used to determine the parts of the system that would benefit the system the most by being optimised. The following are the most expensive functions and represent 61.16% of the total time spent:

- *BruteForceMatcher*, the function used to match keypoints according to their descriptors.
- *SURFInvoker*, SURF descriptor extractor.
- *DFT*, the Discrete Fast Fourier Transform function.
- *at*, the OpenCV matrix entry accessor.
- *cvtColor*, the algorithm used to convert colour images to grey-scale.

As can be seen, all of the listed functions are from the OpenCV library and would be difficult to optimise directly. Therefore the application of these functions would need to be optimised. One of the reasons that the *BruteForceMatcher* was so computationally expensive was that the Euclidean distance of the SURF descriptors had to be computed. If the SURF descriptor was replaced by a descriptor using a binary string (such as the Fast Retina Keypoint (FREAK) descriptor [113]), then the *BruteForceMatcher* could use the Hamming distance instead of the Euclidean distance. Computing and matching the FREAK descriptor in this way is two orders faster than for the SURF descriptor [113]. The *DFT* is fundamental to the PQFT saliency detection method and therefore very difficult to replace or apply in a different manner.

For the purpose of demonstrating how optimisation could affect the performance of the system, the usage of the last two listed functions was optimised. The *at* method was replaced by a method utilising an incrementing pointer, as described in [114]. The *cvtColor* routine was being called by both the FAST keypoint detector and SURF descriptor extractor and this was prevented by providing the functions with a grey-scale image as input. Implementing these simple optimisations resulted in a 9.89% reduction in the total computation time for the *RPY* sequence. Though the improvement in computational performance was a relatively small, it showed that optimisation of the landmark extraction and data association steps could make the saliency-grouped landmark a viable candidate for future visual SLAM implementations.

## 5.7 OVERALL SYSTEM PERFORMANCE

To evaluate the overall system performance, various statistics on the ATE for most of the sequences described in this chapter are shown in Tables 5.17 and 5.18. The ATE for the Tabletop dataset was computed by using the adjusted estimated waypoint positions. The Freiburg 2 sequence results were taken from the PQFT saliency detection method implementation.

**Table 5.17:** Statistics of the ATE produced by the system for the Tabletop and Freiburg 2 datasets. All values are in metres.

| **Statistic** | **Clockwise** | **C. Clockwise** | **Extended** | **XYZ** | **RPY** | **Desk** |
|---|---|---|---|---|---|---|
| RMSE | 0.106 | 0.050 | 0.069 | 0.166 | 0.149 | 0.740 |
| Mean | 0.102 | 0.047 | 0.059 | 0.142 | 0.120 | 0.628 |
| Median | 0.113 | 0.051 | 0.056 | 0.124 | 0.094 | 0.585 |
| Std Dev | 0.033 | 0.019 | 0.035 | 0.085 | 0.088 | 0.391 |
| Maximum | 0.146 | 0.072 | 0.156 | 0.497 | 0.420 | 1.377 |

**Table 5.18:** Statistics of the ATE produced by the system for the Tabletop and Freiburg 2 sequences, expressed as percentages of the total distance travelled.

| **Statistic** | **Clockwise** | **C. Clockwise** | **Extended** | **XYZ** | **RPY** | **Desk** |
|---|---|---|---|---|---|---|
| Distance (m) | 9.000 | 9.000 | 9.960 | 7.029 | 1.506 | 18.880 |
| RMSE | 1.182 | 0.556 | 0.688 | 2.356 | 9.880 | 3.918 |
| Mean | 1.128 | 0.519 | 0.597 | 2.023 | 7.968 | 3.326 |
| Median | 1.254 | 0.567 | 0.558 | 1.770 | 6.242 | 3.097 |
| Std Dev | 0.369 | 0.208 | 0.354 | 1.295 | 1.208 | 5.837 |
| Maximum | 1.619 | 0.801 | 1.566 | 7.074 | 27.855 | 7.293 |

Direct comparison between the Freiburg 2 and Tabletop dataset was difficult because of the fundamental differences in the way the ground truths were determined. However, general and qualitative conclusions could be drawn. From the computed percentages it can be seen that the *Extended* sequence produced less accurate estimates than the *C.Clockwise* sequence and the estimates for the *Desk* sequence were less accurate than for the *XYZ* sequence. Therefore it can be stated that the more complicated and longer camera paths led to an increase in the system estimation error. On the other hand, this error was restricted by the loop closing effect, as demonstrated by the *Desk* sequence. Furthermore, when the errors between the *Clockwise* and *C.Clockwise* sequences and between the *XYZ* and *RPY* were considered, it is possible that the differences in illumination levels and Kinect movement produced greater errors than the larger paths did. From this it can be concluded that the implemented SLAM system utilising saliency grouped landmarks was an effective and expandable solution for robot localisation but it could benefit from being made more robust.

# CHAPTER 6

# CONCLUSION

In this dissertation the use of visual saliency in SLAM was investigated. To do so, a SLAM system using a saliency-based landmark was implemented. The newly developed landmark consisted of image features grouped together according to the saliency content of an image. The implemented SLAM system used an EKF to track landmarks extracted from visual and depth data generated by a Kinect. The implemented SLAM system was capable of estimating the position and orientation of a Kinect as it was moved around, as well as the location of landmarks in the viewed environment. The following chapter discusses the results obtained in Chapter 5 and the research questions raised in Chapter 1. Aspects warranting further research are also identified.

## 6.1 LANDMARK DEFINITION AND HANDLING

The definition of the landmarks was an important research question as it defined how saliency was used to implement vision-based SLAM. The landmarks were defined as consisting of FAST keypoints, with associated SURF descriptors, found within an isolated high-saliency region. The position of a landmark was the mean of the 3D spatial location of all pixels in the region.

A number of landmark-handling routines had to be developed to make use of the new landmark. The landmark extraction routine was augmented with filters that removed landmarks with low saliency content and spurious spatial positions. The filters were also capable of adapting to the texture sparsity of a scene so that enough landmarks could be produced to provide enough update information to the EKF. The data association routine used both the geometry and the descriptors of the multiple image features in each landmark to find quality matches. A landmark management routine was required to eliminate non-performing landmarks so that the efficiency of the system could be maintained.

In terms of the coherence theory described in [49], the regions isolated by the saliency detection method could be interpreted as proto-objects. Suitable regions were tracked by the EKF-SLAM algorithm through the use of the SURF descriptors of the FAST image features. Thus, the proto-objects were made coherent in time and space, which elevated them to the status of objects [49].

## 6.2 EVALUATION OF RESULTS

The manner in which the implemented SLAM system was evaluated was an important question that determined how the results were to be interpreted. A short investigation was conducted to determine the most appropriate evaluation methodology, which was then used.

The EKF-SLAM system was first validated in simulation according to the observation of two phenomena: A reduction in position uncertainties after landmark reobservation and the continuous decrease of the determinants of landmark covariance matrix entries. Both of these phenomena were observed, thus indicating that the developed system was a viable SLAM platform for further development and experimentation.

Thereafter two datasets of recorded Kinect movement sequences were used. The two datasets differed in the type of ground truth provided with each dataset. Recorded datasets allowed for the generation of repeatable and easily comparable results, as well as the isolation of specific events and causes.

A new dataset, Tabletop, containing the ground truth for specific waypoints was recorded. The performance of the SLAM system on these simple sequences showed that the system was capable of producing good robot pose and landmark position estimates. However, experiments indicated that rough movement of the Kinect, as well as low illumination levels in an environment, had a detrimental effect on the estimation results. Unfortunately, a portion of the results were not statistically significant and further experimentation would be required to comprehensively prove the effect of movement on the system.

The Freiburg 2 dataset provided the ground truth for the complete path, which allowed for a more thorough analysis of the system. Experiments conducted using the dataset showed that the system estimated translational movements better than rotational movements. The results also showed that the system was capable of estimating motion during a large loop and recognising that the starting point had been reached. The observed occurrence of loop closure caused a reduction in the estimation error.

The error correction showed that the system was capable of more than a standard dead-reckoning odometry method, which would have suffered from continual error growth due to drift.

## 6.3 COMPARISON OF SALIENCY DETECTION METHODS

One of the questions encountered when designing a saliency-based SLAM implementation was how the saliency content of an image was to be determined effectively. Frequency-based saliency detection methods were used because of their ease of implementation, impressive performance in isolating regions of possible importance, speed and the fact that the methods have relatively few parameters to set. SLAM systems that used the SR, PFT and PQFT methods were implemented and the estimation results were compared to determine the most effective method. Overall, the PQFT implementation was found to be very accurate but also the slowest of all the methods. The PQFT implementation also struggled to maintain landmark coherence during large-scale changes. The SR implementation was able to cope with these changes and was the fastest method. There was also no statistically significant difference in the performance of the SR and PQFT methods on the *Desk* sequence. However, the SR SLAM implementation failed to maintain an accurate estimate when viewing texture-sparse scenes. Both the PQFT and SR implementations were able to achieve loop closure successfully, while the PFT implementation was not as successful. From the experiments conducted it can be concluded that their are certain distinct advantages to using the PQFT or SR methods and that these should be considered before applying either method.

## 6.4 ADVANTAGES OF GROUPED FEATURES AS LANDMARKS

Whether the use of multiple features to form a landmark would be more beneficial than a singular feature landmark was an important research question that arose from the landmark definition. A comparison between a SLAM implementation that used singular features as landmarks and one that used the new saliency-based grouped feature landmark was conducted. In both simulation and real-world experiments it was found that the use of groups of features as landmarks led to more accurate and stable positional estimates. The simulation showed that a minimum number of landmarks was required for the SLAM system to produce usable results. The need for a minimum number of landmarks was also indirectly observed in the dataset experiments. The dataset experiments also showed that grouped landmarks were more robust to texture-sparse regions and to large camera movements.

A short computational analysis comparing the two SLAM implementations showed that the landmark extraction and data association routines associated with the new saliency grouped landmark were far more computationally expensive those associated with the singular landmark. The larger expense was to be expected given the far greater complexity of the newly developed landmark. However, it was shown that the computation time of the saliency grouped landmark routines can be reduced using basic profiling and optimisation methods.

## 6.5 FUTURE RESEARCH POSSIBILITIES

Though the current, landmark-based SLAM implementation was most suitable for application in restricted, structured indoor environments, the landmarks themselves can be used in a system operating in a large, unstructured outdoor space. The increase in operational scope is possible because the landmarks were based on the saliency content of images, which is an effective measure by which to segment coherent landmarks from natural environments. The general graph optimisation system shown in [14] presents an opportunity to test the landmark using a large-scale SLAM implementation.

However, for application in a large-scale SLAM system the computational expense of the landmark extraction and data association routines would have to be reduced, preferably with a large enough factor to allow for implementation as part of a real-time SLAM system. Great reductions in the time needed to process incoming video images could be realised by replacing bottleneck functions with the latest developed methods, such as the FREAK descriptor [113], or by using more advanced programming techniques and hardware.

The cloud of 3D points produced by the Kinect has been underutilised in this implementation. Although this was done on purpose so as to focus on using visual saliency to produce landmarks, there are a number of possibilities that could be investigated. The PCL [115] contains a variety of filtering and geometric fitting algorithms, which could be used to relate the environment and the landmarks to the robot in interesting ways. The most direct application of the saliency-based landmark is to aid in the finding of correspondences in point clouds for a scan-matching SLAM system.

The landmark composition can be extended to make it even more effective. The normal vector of each salient region could form part of the landmark as a means of incorporating more orientation information into the system. The landmark could be made more invariant to scale change by using a multiscale saliency detector such as that proposed in [66]. The fourth input to the QFT could be used to add additional environmental information to increase the saliency information of sparsely-textured scenes.

The EKF formulation used could also be improved. The update phase can be reformulated to use the 2D pixel to 3D spatial point conversion Equations 4.2 and 4.3. Using these equations could lead to some interesting possibilities, such as actively searching for landmarks in the image space as described in [19]. A more complex sensor noise model can be implemented to use landmarks better in non-ideal circumstances. For example, the expected sensor noise can be made a function of landmark distance and illumination level.

Future implementation and acceptance of robots is dependent on the extent of their autonomous and self-navigational capabilities. Visual sensors can provide robots with a large amount of information that, if properly processed, can improve how robots orientate themselves in an environment. Saliency detection offers an effective mechanism to prioritise regions for processing. In this dissertation saliency detection was used to improve vision-based SLAM, which can serve as the foundation for navigation systems and further automation.

# REFERENCES

[1] (2012, August) Kiva Systems Incorporated: Kiva Systems. [Online]. Available: http://www.kivasystems.com/.

[2] J. Kim and S. Sukkarieh, "Real-time implementation of airborne inertial-SLAM," *Robot. Auton. Syst.*, vol. 55, no. 1, pp. 62–71, 2007.

[3] D. Törnqvist, T. Schön, R. Karlsson, and F. Gustafsson, "Particle filter SLAM with high dimensional vehicle model," *J. of Intell. Robot. Syst.*, vol. 55, no. 4, pp. 249–266, 2009.

[4] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Auton. Robot Veh.*, vol. 1, pp. 167–193, 1990.

[5] J. Weingarten and R. Siegwart, "EKF-based 3D SLAM for structured environment reconstruction," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3834–3839.

[6] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 229–241, 2001.

[7] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[8] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1066–1077, 2008.

[9] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 239–256, 1992.

[10] A. Milella and R. Siegwart, “Stereo-based ego-motion estimation using pixel tracking and iterative closest point,” in *IEEE International Conference on Computer Vision Systems*. IEEE, 2006, p. 21.

[11] F. Bertolli, P. Jensfelt, and H. Christensen, “SLAM using visual scan-matching with distinguishable 3D points,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 4042–4047.

[12] D. Viejo and M. Cazorla, “3D plane-based egomotion for SLAM on semi-structured environment,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 2761–2766.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” in *Proceedings of the National conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002, pp. 593–598.

[14] H. Strasdat, A. Davison, J. Montiel, and K. Konolige, “Double window optimisation for constant time visual SLAM,” in *2011 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2352–2359.

[15] J. Nieto, T. Bailey, and E. Nebot, “Recursive scan-matching SLAM,” *Robot. Auton. Systems*, vol. 55, no. 1, pp. 39–49, 2007.

[16] M. Bryson and S. Sukkarieh, “Building a robust implementation of bearing-only inertial SLAM for a UAV,” *J. Field Robot.*, vol. 24, no. 1-2, pp. 113–143, 2007.

[17] I. Mahon, S. Williams, O. Pizarro, and M. Johnson-Roberson, “Efficient view-based SLAM using visual loop closures,” *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1002–1014, 2008.

[18] T. Lemaire, C. Berger, I. Jung, and S. Lacroix, “Vision-based SLAM: Stereo and monocular approaches,” *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 343–364, 2007.

[19] A. Davison, I. Reid, N. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1052–1067, 2007.

[20] S. Barkby, S. Williams, O. Pizarro, and M. Jakuba, "A featureless approach to efficient bathymetric SLAM using distributed particle mapping," *J. Field Robot.*, vol. 28, no. 1, pp. 19–39, 2011.

[21] W. Zhou, J. Miró, and G. Dissanayake, "Information-efficient 3-D visual SLAM for unstructured domains," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1078–1087, 2008.

[22] S. May, D. Droeschel, S. Fuchs, D. Holz, and A. Nuchter, "Robust 3D-mapping with time-of-flight cameras," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1673–1678.

[23] J. Saez and F. Escolano, "Entropy minimization SLAM using stereo vision," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2006, pp. 36–43.

[24] A. Davison and D. Murray, "Simultaneous localization and map-building using active vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 865–880, 2002.

[25] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer, "Avenue: Automated site modeling in urban environments," in *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 2001, pp. 357–364.

[26] H. Surmann, A. Nuchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robot. Auton. Syst.*, vol. 45, no. 3-4, pp. 181–198, 2003.

[27] (2012, November) Xbox 360 + Kinect. [Online]. Available: http://www.xbox.com/en-US/kinect?xr=shellnav.

[28] (2011, February) PrimeSense: Reference Design. [Online]. Available: http://www.primesense.com/?p=514

[29] F. Steinbrucker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *2011 IEEE International Conference on Computer Vision Workshops*. IEEE, 2011, pp. 719–722.

[30] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*. IEEE, 2012.

[31] G. Dissanayake, S. Williams, H. Durrant-Whyte, and T. Bailey, "Map management for efficient simultaneous localization and mapping (SLAM)," *Auton. Robot.*, vol. 12, no. 3, pp. 267–286, 2002.

[32] J. Artieda, J. Sebastian, P. Campoy, J. Correa, I. Mondragón, C. Martínez, and M. Olivares, "Visual 3-D SLAM from UAVs," *J. Intell. Robot. Syst.*, vol. 55, no. 4, pp. 299–321, 2009.

[33] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the FastSLAM algorithm," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 424–429.

[34] M. Milford and G. Wyeth, "Mapping a suburb with a single camera using a biologically inspired SLAM system," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1038–1053, 2008.

[35] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*. IEEE, 2006, pp. 1180–1187.

[36] R. Lakaemper and L. Jan Latecki, "Using extended EM to segment planar structures in 3D," in *Proceedings of the 18th International Conference on Pattern Recognition*. IEEE Computer Society, 2006, pp. 1077–1082.

[37] J. Castellanos, J. Montiel, J. Neira, and J. Tardós, "The SPmap: A probabilistic framework for simultaneous localization and map building," *IEEE Trans. Robot. Autom.*, vol. 15, no. 5, pp. 948–952, 2002.

[38] O. Wulf, K. Arras, H. Christensen, and B. Wagner, "2D mapping of cluttered indoor environments by means of 3D perception," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, vol. 4. IEEE, 2004, pp. 4204–4209.

[39] P. de la Puente, D. Rodriguez-Losada, A. Valero, and F. Matia, "3D feature based mapping towards mobile robots' enhanced performance in rescue missions," in *IEEE/RSJ International*

*Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1138–1143.

[40] A. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas, “Discovering higher level structure in visual SLAM,” *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 980–990, 2008.

[41] D. Cole, A. Harrison, and P. Newman, “Using naturally salient regions for SLAM with 3D laser data,” in *International Conference on Robotics and Automation, SLAM Workshop*. Citeseer, 2005.

[42] P. Gemeiner, P. Jojic, and M. Vincze, “Selecting good corners for structure and motion recovery using a time-of-flight camera,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009*. IEEE, 2009, pp. 5711–5716.

[43] B. Steder, R. Rusu, K. Konolige, and W. Burgard, “Point feature extraction on 3D range scans taking into account object boundaries.” *J. Auton. Robots*, vol. 30, no. 1, pp. 25–39, 2011.

[44] M. Bosse and R. Zlot, “Place recognition using regional point descriptors for 3D mapping,” in *Field and Service Robotics: Results of the 7th International Conference*, vol. 62. Springer Verlag, 2010, pp. 195–204.

[45] M. Elmogy and J. Zhang, “Robust real-time landmark recognition for humanoid robot navigation,” in *IEEE International Conference on Robotics and Biomimetics, 2008*. IEEE, 2009, pp. 572–577.

[46] D. Schleicher, L. Bergasa, R. Barea, E. Lopez, M. Ocaa, J. Nuevo, and P. Fernandez, “Real-time stereo visual SLAM in large-scale environments based on SIFT fingerprints,” in *IEEE International Symposium on Intelligent Signal Processing*. IEEE, 2007, pp. 1–6.

[47] J. Choi, K. Lee, S. Ahn, M. Choi, and W. Chung, “A practical solution to SLAM and navigation in home environment,” in *SICE-ICASE, 2006. International Joint Conference*. IEEE, 2006, pp. 2015–2021.

[48] Y. Lee and J. Song, “Visual SLAM in indoor environments using autonomous detection and registration of objects,” in *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008, pp. 301–314.

[49] R. Rensink, “The dynamic representation of scenes,” *Visual Cognition*, vol. 7, no. 1-3, pp. 17–42, 2000.

[50] D. Meger, P. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. Little, and D. Lowe, “Curious George: An attentive semantic robot,” *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 503–511, 2008.

[51] B. Moghaddam and A. Pentland, “Probabilistic visual learning for object representation,” *IEEE Trans. Pattern Anal. Machin. Intell.*, vol. 19, no. 7, pp. 696–710, 1997.

[52] A. Oliva, A. Torralba, M. Castelhano, and J. Henderson, “Top-down control of visual attention in object detection,” in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 1. IEEE, 2003, pp. I–253.

[53] L. Itti, C. Koch, and E. Niebur, “A model of saliency-based visual attention for rapid scene analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1254–1259, 1998.

[54] L. Itti and C. Koch, “A comparison of feature combination strategies for saliency-based visual attention systems,” *SPIE human vision and electronic imaging IV (HVEIâĂŹ99)*, vol. 3644, pp. 373–382, 1999.

[55] J. Harel, C. Koch, and P. Perona, “Graph-based visual saliency,” in *Advances in Neural Information Processing Systems*, vol. 19, 2007.

[56] E. Rahtu and J. Heikkila, “A simple and efficient saliency detector for background subtraction,” in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1137–1144.

[57] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, “Frequency-tuned salient region detection,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1597–1604.

[58] X. Hou and L. Zhang, “Saliency detection: A spectral residual approach,” in *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[59] M. Cerf, J. Harel, W. Einhäuser, and C. Koch, “Predicting human gaze using low-level saliency

combined with face detection," *Advances in Neural Information Processing Systems*, vol. 20, 2008.

[60] J. Ruesch, M. Lopes, A. Bernardino, J. Hornstein, J. Santos-Victor, and R. Pfeifer, "Multimodal saliency-based bottom-up attention a framework for the humanoid robot iCub," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 962–967.

[61] Y. Zhai and M. Shah, "Visual attention detection in video sequences using spatiotemporal cues," in *Proceedings of the 14th annual ACM international conference on Multimedia*. ACM, 2006, pp. 815–824.

[62] Y. Hu, X. Xie, W. Ma, L. Chia, and D. Rajan, "Salient region detection using weighted feature maps based on the human visual attention model," *Advances in Multimedia Information Processing-PCM 2004*, pp. 993–1000, 2005.

[63] S. Vijayakumar, J. Conradt, T. Shibata, and S. Schaal, "Overt visual attention for a humanoid robot," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2001, pp. 2332–2337.

[64] J. Van De Weijer, T. Gevers, and A. Bagdanov, "Boosting color saliency in image feature detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 1, pp. 150–156, 2006.

[65] C. Guo, Q. Ma, and L. Zhang, "Spatio-temporal saliency detection using phase spectrum of quaternion Fourier transform," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[66] M. Holtzman-Gazit, L. Zelnik-Manor, and I. Yavneh, "Salient edges: A multi scale approach," in *ECCV 2010 Workshop on Vision for Cognitive Tasks*, 2010.

[67] R. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[68] J. Civera, O. Grasa, A. Davison, and J. Montiel, "1-point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry," *J. Field Robot.*, vol. 27, no. 5, pp. 609–631, 2010.

[69] P. Newman, “EKF based navigation and SLAM. SLAM summer school 2006,” 2006.

[70] L. Paz, P. Piniés, J. Tardós, and J. Neira, “Large-scale 6-DOF SLAM with stereo-in-hand,” *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 946–957, 2008.

[71] J. Sola, A. Monin, M. Devy, and T. Vidal-Calleja, “Fusing monocular information in multi-camera SLAM,” *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 958–968, 2008.

[72] (2012, November) MATLAB The language of technical computing. [Online]. Available: http://www.mathworks.com/products/matlab/.

[73] (2012, July) Willow Garage: ROS Documentation. [Online]. Available: http://www.ros.org/wiki/

[74] (2012, July) Willow Garage: OpenCV 2.1 C++ Reference. [Online]. Available: http://opencv.willowgarage.com/documentation/cpp/index.html

[75] (2012, August) G. Guennebaud: Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. [Online]. Available: http://eigen.tuxfamily.org

[76] (2012, July) J. Sturm: RGB-D SLAM dataset and benchmark. [Online]. Available: http://vision.in.tum.de/data/datasets/rgbd-dataset

[77] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF),” *Comput. Vis. Image Und.*, vol. 110, no. 3, pp. 346–359, 2008.

[78] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, W. Burgard, D. Cremers, and R. Siegwart, “Towards a benchmark for RGB-D SLAM evaluation,” in *Proceedings of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf.(RSS), Los Angeles, USA*, vol. 2, 2011, p. 3.

[79] S. Huang and G. Dissanayake, “Convergence and consistency analysis for extended Kalman filter based SLAM,” *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 1036–1049, Oct. 2007.

[80] J. Kim and S. Sukkarieh, “Autonomous airborne navigation in unknown terrain environments,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 40, no. 3, pp. 1031–1045, Jul. 2004.

[81] V. Sazdovski and P. M. G. Silson, “Inertial navigation aided by vision-based simultaneous localization and mapping,” *IEEE Sensors J.*, vol. 11, no. 8, pp. 1646–1656, Aug. 2011.

[82] A. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *Proceedings of the Ninth IEEE International Conference on Computer Vision*. IEEE, 2003, pp. 1403–1410.

[83] S. Se, D. Lowe, and J. Little, “Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks,” *Intl. J. Robot. Res.*, vol. 21, no. 8, pp. 735–758, 2002.

[84] J. Rogers, A. Trevor, C. Nieto-Granda, and H. Christensen, “Simultaneous localization and mapping with learned object recognition and semantic data association,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1264–1270.

[85] S. Ahn, M. Choi, J. Choi, and W. Chung, “Data association using visual object recognition for EKF-SLAM in home environment,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 2588–2594.

[86] S. Takezawa, D. Herath, and G. Dissanayake, “SLAM in indoor environments with stereo vision,” in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.(IROS 2004)*, vol. 2. IEEE, 2004, pp. 1866–1871.

[87] Z. Dai-xian, “Binocular vision-SLAM using improved SIFT algorithm,” in *2010 2nd International Workshop on Intelligent Systems and Applications*. IEEE, 2010, pp. 1–4.

[88] M. Dailey and M. Parnichkun, “Simultaneous localization and mapping with stereo vision,” in *9th International Conference on Control, Automation, Robotics and Vision, 2006*. IEEE, 2006, pp. 1–6.

[89] A. Gee, D. Chekhlov, W. Mayol, and A. Calway, “Discovering planes and collapsing the state space in visual SLAM,” in *British Machine Vision Conference*, vol. 2007, 2007.

[90] T. Botterill, S. Mills, and R. Green, “Bag-of-words-driven, single-camera simultaneous localization and mapping,” *J. Field Robot.*, vol. 28, no. 2, pp. 204–226, 2011.

[91] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich, “The

vSLAM algorithm for robust localization and mapping," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005*. IEEE, 2005, pp. 24–29.

[92] H. Morioka, S. Yi, and O. Hasegawa, "Vision-based mobile robot's SLAM and navigation in crowded environments," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3998–4005.

[93] X. Zhang, A. Rad, and Y. Wong, "Sensor fusion of monocular cameras and laser rangefinders for line-based simultaneous localization and mapping (SLAM) tasks in autonomous mobile robots," *Sensors*, vol. 12, no. 1, pp. 429–452, 2012.

[94] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Auton. Robots*, vol. 27, no. 4, pp. 387–407, 2009.

[95] G. Hu, S. Huang, and G. Dissanayake, "Evaluation of pose only SLAM," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 3732–3737.

[96] A. Chatterjee and F. Matsuno, "A geese PSO tuned fuzzy supervisor for EKF based solutions of simultaneous localization and mapping (SLAM) problems in mobile robots," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 5542–5548, 2010.

[97] J. Gutmann, E. Eade, P. Fong, M. Munich *et al.*, "Vector field SLAM localization by learning the spatial variation of continuous signals," *IEEE Trans. Robot.*, vol. 28, no. 3, pp. 650–667, 2012.

[98] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige, "An experimental comparison of localization methods," in *1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 1998, pp. 736–743.

[99] J.-L. Blanco, J. González, and J.-A. Fernández-Madrigal, "Subjective local maps for hybrid metric-topological SLAM," *Robot. Auton. Syst.*, vol. 57, no. 1, pp. 64–74, 2009.

[100] R. Johnson, I. Miller, and J. Freund, *Miller and Freund's Probability and Statistics for Engineers*. Prentice Hall PTR, 2004.

[101] A. M. Sabatini, "Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation," *Sensors*, vol. 11, no. 10, pp. 9182–9206, 2011.

[102] P. Beeson, A. Murarka, and B. Kuipers, "Adapting proposal distributions for accurate, efficient mobile robot localization," in *2006 IEEE International Conference on Robotics and Automation*, 2006, pp. 49–55.

[103] D. Stronger and P. Stone, "A comparison of two approaches for vision and self-localization on a mobile robot," in *2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3915–3920.

[104] C. Siagian and L. Itti, "Biologically inspired mobile robot vision localization," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 861–873, 2009.

[105] A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 298–320, 2010.

[106] S. Magnenat, V. Longchamp, M. Bonani, P. Rétornaz, P. Germano, H. Bleuler, and F. Mondada, "Affordable SLAM through the co-design of hardware and methodology," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 5395–5401.

[107] A. Goldhoorn, "Solving ambiguity in global localization of autonomous robots," Ph.D. dissertation, Masters thesis, University of Groningen, 2008.

[108] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, "Monte carlo localization in outdoor terrains using multilevel surface maps," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 346–359, 2008.

[109] S. Limsoonthrakul, M. N. Dailey, and M. Parnichkun, "Intelligent vehicle localization using GPS, compass, and machine vision," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 3981–3986.

[110] J. R. Schoenberg, M. Campbell, and I. Miller, "Localization with multi-modal vision measurements in limited GPS environments using Gaussian sum filters," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1423–1428.

[111] (2012, December) Callgrind: Call-graph generating cache and branch prediction profiler. [Online]. Available: http://valgrind.org/docs/manual/cl-manual.html.

[112] (2012, December) Kcachegrind call graph viewer. [Online]. Available: http://kcachegrind.sourceforge.net/html/Home.html.

[113] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 510–517.

[114] R. Laganière, *OpenCV 2 computer vision application programming cookbook*. Packt Pub Limited, 2011.

[115] (2012, August) Willow Garage: PCL what is PCL? [Online]. Available: http://www.pointclouds.org

# APPENDIX A

# DERIVATION OF JACOBIAN MATRICES

In this appendix the formulation of the various Jacobian matrices used in the SLAM system developed in Section 3.2 are presented. These formulations are based on the MATLAB code accompanying [68]. First the formulation of the Jacobian for the EKF prediction phase are given, followed by the update phase Jacobian. The augment phase Jacobians are the last to be presented.

## A.1 CONSTANT VELOCITY MOVEMENT MODEL

The formula for the complete constant velocity model Jacobian is

$$\nabla \mathbf{F}_r = \frac{\delta \mathbf{f}_r(\mathbf{r}, \Delta t)}{\delta \mathbf{r}} = \begin{pmatrix} \frac{\delta \mathbf{f}_p(\mathbf{p}^W, \mathbf{v}^W, \Delta t)}{\delta \mathbf{r}} \\ \frac{\delta \mathbf{f}_q(\mathbf{q}^{WR}, \mathbf{w}^R, \Delta t)}{\delta \mathbf{r}} \\ \frac{\delta \mathbf{f}_v(\mathbf{v}^W)}{\delta \mathbf{r}} \\ \frac{\delta \mathbf{f}_v(\mathbf{w}^R)}{\delta \mathbf{r}} \end{pmatrix} = \begin{pmatrix} \frac{\delta}{\delta \mathbf{r}} \left(\mathbf{p}^W + \left(\mathbf{v}^W + \mathbf{V}\right) \Delta t\right) \\ \frac{\delta}{\delta \mathbf{r}} \left(\mathbf{q}^{WR} \otimes \mathbf{q}\left(\left(\mathbf{w}^R + \mathbf{\Omega}\right) \Delta t\right)\right) \\ \frac{\delta}{\delta \mathbf{r}} \left(\mathbf{v}^W + \mathbf{V}\right) \\ \frac{\delta}{\delta \mathbf{r}} \left(\mathbf{w}^R + \mathbf{\Omega}\right) \end{pmatrix}. \tag{A.1}$$

Each of these derivatives are formulated in turn. The first derivative is

$$\frac{\delta \mathbf{f}_p(\mathbf{p}^W, \mathbf{v}^W, \Delta t)}{\delta \mathbf{r}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \end{pmatrix}. \tag{A.2}$$

The formulation of the derivative of the quaternion equation is more complicated and requires a number of steps. First the formula for determining the quaternion of the angular velocity vector and time step needs to be specified. As the angular velocity vector is a compressed angle-axis rotation, the formula is

$$\mathbf{q}\left(\left(\mathbf{w}^R+\mathbf{W}\right)\Delta t\right)=\mathbf{q}^{w\Delta t}=\begin{pmatrix}\cos(\frac{|\mathbf{w}\Delta t|}{2})\\ \sin(\frac{|\mathbf{w}\Delta t|}{2})\frac{w_x}{|\mathbf{w}\Delta t|}\\ \sin(\frac{|\mathbf{w}\Delta t|}{2})\frac{w_y}{|\mathbf{w}\Delta t|}\\ \sin(\frac{|\mathbf{w}\Delta t|}{2})\frac{w_z}{|\mathbf{w}\Delta t|}\end{pmatrix}. \tag{A.3}$$

The new quaternion is written as $\mathbf{q}^{wt}$ to simplify notation. The product of this quaternion with $\mathbf{q}^{WR}$t will be written as $\mathbf{q}^p$ for the same reason. Subscripts indicate which of the components of the quaternion are used. $w_i$ indicates which component of the angular velocity vector is used. The derivative of the quaternion equation is

$$\frac{\delta\mathbf{f}_q(\mathbf{q}^{WR},\mathbf{w}^R,\Delta t)}{\delta\mathbf{r}}=\frac{\delta\left(\mathbf{q}^{WR}\otimes\mathbf{q}^{w\Delta t}\right)}{\delta\mathbf{r}} \tag{A.4}$$

$$=\begin{pmatrix}0&0&0&q_w^{w\Delta t}&-q_x^{w\Delta t}&-q_y^{w\Delta t}&-q_z^{w\Delta t}&0&0&0&\frac{\delta q_w^p}{\delta w_x}&\frac{\delta q_w^p}{\delta w_y}&\frac{\delta q_w^p}{\delta w_z}\\ 0&0&0&q_x^{w\Delta t}&q_w^{w\Delta t}&q_z^{w\Delta t}&-q_y^{w\Delta t}&0&0&0&\frac{\delta q_x^p}{\delta w_x}&\frac{\delta q_x^p}{\delta w_y}&\frac{\delta q_x^p}{\delta w_z}\\ 0&0&0&q_y^{w\Delta t}&-q_z^{w\Delta t}&q_w^{w\Delta t}&q_x^{w\Delta t}&0&0&0&\frac{\delta q_y^p}{\delta w_x}&\frac{\delta q_y^p}{\delta w_y}&\frac{\delta q_y^p}{\delta w_z}\\ 0&0&0&q_z^{w\Delta t}&q_y^{w\Delta t}&-q_x^{w\Delta t}&q_w^{w\Delta t}&0&0&0&\frac{\delta q_z^p}{\delta w_x}&\frac{\delta q_z^p}{\delta w_y}&\frac{\delta q_z^p}{\delta w_z}\end{pmatrix}. \tag{A.5}$$

To obtain the last three columns, the derivative of $\mathbf{q}^p$ by the angular velocity vector needs to be determined. The chain rule of differentiation is used to get

$$\frac{\delta\left(\mathbf{q}^{WR}\otimes\mathbf{q}^{w\Delta t}\right)}{\delta\mathbf{r}}=\frac{\delta\left(\mathbf{q}^{WR}\otimes\mathbf{q}^{w\Delta t}\right)}{\delta\mathbf{q}^{w\Delta t}}\frac{\delta\mathbf{q}^{w\Delta t}}{\delta\mathbf{w}}. \tag{A.6}$$

The first term of this product is given by

$$\frac{\delta\left(\mathbf{q}^{WR}\otimes\mathbf{q}^{w\Delta t}\right)}{\delta\mathbf{q}^{w\Delta t}}=\begin{pmatrix}q_w^{WR}&-q_x^{WR}&-q_y^{WR}&-q_z^{WR}\\ q_x^{WR}&q_w^{WR}&-q_z^{WR}&q_y^{WR}\\ q_y^{WR}&q_z^{WR}&q_w^{WR}&-q_x^{WR}\\ q_z^{WR}&-q_y^{WR}&q_x^{WR}&q_w^{WR}\end{pmatrix}, \tag{A.7}$$

while the second term is

$$\frac{\delta \mathbf{q}^{w\Delta t}}{\delta \mathbf{w}} = \begin{pmatrix} -\frac{\Delta t}{2}\frac{w_x}{|\mathbf{w}|}\sin\left(\frac{|\mathbf{w}|\Delta t}{2}\right) & -\frac{\Delta t}{2}\frac{w_y}{|\mathbf{w}|}\sin\left(\frac{|\mathbf{w}|\Delta t}{2}\right) & -\frac{\Delta t}{2}\frac{w_z}{|\mathbf{w}|}\sin\left(\frac{|\mathbf{w}|\Delta t}{2}\right) \\ dq_A(w_x) & dq_B(w_x, w_y) & dq_B(w_x, w_z) \\ dq_B(w_y, w_x) & dq_A(w_y) & dq_B(w_y, w_z) \\ dq_B(w_z, w_x) & dq_B(w_z, w_y) & dq_A(w_z)) \end{pmatrix} \tag{A.8}$$

$$dq_A(w_i) = \frac{\Delta t}{2}\frac{w_i^2}{|\mathbf{w}|^2}\cos\left(\frac{|\mathbf{w}|\Delta t}{2}\right) + \frac{1}{|\mathbf{w}|}\left(1 - \frac{w_i^2}{|\mathbf{w}|^2}\right)\sin\left(\frac{|\mathbf{w}|\Delta t}{2}\right) \tag{A.9}$$

$$dq_B(w_i, w_j) = \frac{w_i w_j}{|\mathbf{w}|^2}\left(\frac{\Delta t}{2}\cos\left(\frac{|\mathbf{w}|\Delta t}{2}\right) - \frac{1}{|\mathbf{w}|}\sin\left(\frac{|\mathbf{w}|\Delta t}{2}\right)\right). \tag{A.10}$$

The remaining derivatives of Equation A.1 are

$$\frac{\delta \mathbf{f}_v(\mathbf{v}^W)}{\delta \mathbf{r}} = \begin{pmatrix} 0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0 \end{pmatrix} \tag{A.11}$$

$$\frac{\delta \mathbf{f}_v(\mathbf{w}^R)}{\delta \mathbf{r}} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix}. \tag{A.12}$$

## A.2 OBSERVATION MODEL

For ease of reference, the observation equation is restated

$$\mathbf{m}^R_{i,k|k-1} = \mathbf{h}\left(\mathbf{x}_{k|k-1}\right) = \left(\mathbf{q}^{WR}_{k|k-1}\right)^* \otimes \mathbf{q}\left(\mathbf{m}^W_{i,k|k-1} - \mathbf{p}^W_{k|k-1}\right) \otimes \mathbf{q}^{WR}_{k|k-1}. \tag{A.13}$$

As the landmarks have no influence on one another within the observation model, the Jacobian approximation of the observation equation is

$$\mathbf{H}_x = \frac{\delta \mathbf{h}(\mathbf{x})}{\delta \mathbf{x}} = \begin{pmatrix} \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_1)}{\delta \mathbf{x}} \\ \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_2)}{\delta \mathbf{x}} \\ \vdots \\ \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_n)}{\delta \mathbf{x}} \end{pmatrix}, \tag{A.14}$$

where $n$ is the number of landmarks that are in the map. The entries can be formulated as

$$\frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{x}} = \begin{pmatrix} \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{p}} & \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{q}} & \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{v}} & \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{w}} & \mathbf{0} & \mathbf{0} & \dots & \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{m}_i} \dots & \mathbf{0} & \dots \end{pmatrix}. \tag{A.15}$$

Given the observation model, the derivatives with regard to the angular and translational velocities are known to be zero. The derivative in terms of the quaternion is

$$\frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{q}} = \begin{pmatrix} dq_{11} & dq_{12} & dq_{13} & dq_{14} \\ dq_{21} & dq_{22} & dq_{23} & dq_{24} \\ dq_{31} & dq_{32} & dq_{33} & dq_{34} \end{pmatrix}, \tag{A.16}$$

where

$$\begin{aligned}
dq_{11} &= 2q_w^{WR}\left(p_x^W - m_ix\right) + 2q_z^{WR}\left(p_y^W - m_iy\right) - 2q_y^{WR}\left(p_z^W - m_iz\right) \\
dq_{12} &= 2q_z^{WR}\left(p_z^W - m_iz\right) + 2q_x^{WR}\left(p_x^W - m_ix\right) + 2q_y^{WR}\left(p_y^W - m_iy\right) \\
dq_{13} &= 2q_x^{WR}\left(p_y^W - m_iy\right) - 2q_y^{WR}\left(p_x^W - m_ix\right) - 2q_w^{WR}\left(p_z^W - m_iz\right) \\
dq_{14} &= 2q_z^{WR}\left(p_y^W - m_iy\right) + 2q_x^{WR}\left(p_z^W - m_iz\right) - 2q_z^{WR}\left(p_x^W - m_ix\right) \\
dq_{21} &= 2q_z^{WR}\left(p_y^W - m_iy\right) + 2q_x^{WR}\left(p_z^W - m_iz\right) - 2q_z^{WR}\left(p_x^W - m_ix\right) \\
dq_{22} &= -2q_x^{WR}\left(p_y^W - m_iy\right) + 2q_y^{WR}\left(p_x^W - m_ix\right) + 2q_w^{WR}\left(p_z^W - m_iz\right) \\
dq_{23} &= 2q_z^{WR}\left(p_z^W - m_iz\right) + 2q_x^{WR}\left(p_x^W - m_ix\right) + 2q_y^{WR}\left(p_y^W - m_iy\right) \\
dq_{24} &= -2q_w^{WR}\left(p_x^W - m_ix\right) - 2q_z^{WR}\left(p_y^W - m_iy\right) + 2q_y^{WR}\left(p_z^W - m_iz\right) \\
dq_{31} &= -2q_x^{WR}\left(p_y^W - m_iy\right) + 2q_y^{WR}\left(p_x^W - m_ix\right) + 2q_w^{WR}\left(p_z^W - m_iz\right) \\
dq_{32} &= -2q_z^{WR}\left(p_y^W - m_iy\right) - 2q_x^{WR}\left(p_z^W - m_iz\right) + 2q_z^{WR}\left(p_x^W - m_ix\right) \\
dq_{33} &= 2q_w^{WR}\left(p_x^W - m_ix\right) + 2q_z^{WR}\left(p_y^W - m_iy\right) q_z^{WR} - 2q_y^{WR}\left(p_z^W - m_iz\right) \\
dq_{34} &= 2q_z^{WR}\left(p_z^W - m_iz\right) + 2q_x^{WR}\left(p_x^W - m_ix\right) q_x^{WR} + 2q_y^{WR}\left(p_y^W - m_iy\right).
\end{aligned}$$

The derivative of the observation model with regard to each landmark is

$$\frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{m}_i} = \begin{pmatrix} dm_1 & 2q_y^{WR}q_x^{WR}+2q_z^{WR}q_w^{WR} & 2q_z^{WR}q_x^{WR}-2q_w^{WR}q_y^{WR} \\ 2q_y^{WR}q_x^{WR}-2q_z^{WR}q_w^{WR} & dm_2 & 2q_z^{WR}q_y^{WR}+2q_w^{WR}q_x^{WR} \\ 2q_z^{WR}q_x^{WR}+2q_w^{WR}q_y^{WR} & 2q_z^{WR}q_y^{WR}-2q_w^{WR}q_x^{WR} & dm_3 \end{pmatrix}, \quad \text{(A.17)}$$

where

$$dm_1 = \left(q_x^{WR}\right)^2 + \left(q_w^{WR}\right)^2 - \left(q_z^{WR}\right)^2 - \left(q_y^{WR}\right)^2$$
$$dm_2 = -\left(q_x^{WR}\right)^2 + \left(q_w^{WR}\right)^2 - \left(q_z^{WR}\right)^2 + \left(q_y^{WR}\right)^2$$
$$dm_3 = -\left(q_x^{WR}\right)^2 + \left(q_w^{WR}\right)^2 + \left(q_z^{WR}\right)^2 - \left(q_y^{WR}\right)^2.$$

The derivative of the observation model with regard to the pose can be stated as

$$\frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{p}} = \frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{m}_i}\frac{\delta \mathbf{m}_i}{\delta \mathbf{p}} = -\frac{\delta \mathbf{h}(\mathbf{r},\mathbf{m}_i)}{\delta \mathbf{m}_i}. \quad \text{(A.18)}$$

## A.3 MAPPING FUNCTION

The mapping function is restated here for ease of reference

$$\mathbf{m}^W = \mathbf{g}(\mathbf{m}^R) = \mathbf{q}^{WR} \otimes \mathbf{q}\left(\mathbf{m}^R\right) \otimes \left(\mathbf{q}^{WR}\right)^* + \mathbf{p}^W, \quad \text{(A.19)}$$

where the Jacobians are given by

$$\nabla \mathbf{G}_r = \frac{\mathbf{g}(\mathbf{m}^R)}{\mathbf{r}} \quad \text{(A.20)}$$
$$\nabla \mathbf{G}_z = \frac{\mathbf{g}(\mathbf{m}^R)}{\mathbf{m}}. \quad \text{(A.21)}$$

The $\nabla \mathbf{G}_r$ is simple to formulate

$$\nabla \mathbf{G}_r = \begin{pmatrix} 1 & 0 & 0 & dqgr_{11} & dqgr_{12} & dqgr_{13} & dqgr_{14} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & dqgr_{21} & dqgr_{22} & dqgr_{23} & dqgr_{24} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dqgr_{31} & dqgr_{32} & dqgr_{33} & dqgr_{34} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \text{(A.22)}$$

where

$$
\begin{aligned}
dqgr_{11} &= 2q_y^{WR}m_z^R - 2m_y^R q_z^{WR} + 2m_x^R q_w^{WR}\\
dqgr_{12} &= 2m_x^R q_x^{WR} + 2m_z^R q_z^{WR} + 2q_y^{WR}m_y^R\\
dqgr_{13} &= -2m_x^R q_y^{WR} + 2q_x^{WR}m_y^R + 2m_z^R q_w^{WR}\\
dqgr_{14} &= -2m_y^R q_w^{WR} + 2m_z^R q_x^{WR} - 2m_x^R q_z^{WR}\\
dqgr_{21} &= 2m_y^R q_w^{WR} - 2m_z^R q_x^{WR} + 2m_x^R q_z^{WR}\\
dqgr_{22} &= 2m_x^R q_y^{WR} - 2q_x^{WR}m_y^R - 2m_z^R q_w^{WR}\\
dqgr_{23} &= 2m_x^R q_x^{WR} + 2m_z^R q_z^{WR} + 2q_y^{WR}m_y^R\\
dqgr_{24} &= 2q_y^{WR}m_z^R - 2m_y^R q_z^{WR} + 2m_x^R q_w^{WR}\\
dqgr_{31} &= -2m_x^R q_y^{WR} + 2q_x^{WR}m_y^R + 2m_z^R q_w^{WR}\\
dqgr_{32} &= 2m_y^R q_w^{WR} - 2m_z^R q_x^{WR} + 2m_x^R q_z^{WR}\\
dqgr_{33} &= -2q_y^{WR}m_z^R + 2m_y^R q_z^{WR} - 2m_x^R q_w^{WR}\\
dqgr_{34} &= 2m_x^R q_x^{WR} + 2m_z^R q_z^{WR} + 2q_y^{WR}m_y^R.
\end{aligned}
$$

Finally, the second Jacobian is

$$
\nabla \mathbf{G}_z = \begin{pmatrix} dgz_1 & 2q_y^{WR}q_x^{WR} - 2q_z^{WR}q_w^{WR} & 2q_y^{WR}q_w^{WR} + 2q_x^{WR}q_z^{WR} \\ 2q_y^{WR}q_x^{WR} + 2q_z^{WR}q_w^{WR} & dgz_2 & 2q_y^{WR}q_z^{WR} - 2q_x^{WR}q_w^{WR} \\ -2q_y^{WR}q_w^{WR} + 2q_x^{WR}q_z^{WR} & 2q_y^{WR}q_z^{WR} + 2q_x^{WR}q_w^{WR} & dgz_3 \end{pmatrix}, \tag{A.23}
$$

where

$$
\begin{aligned}
dgz_1 &= -\left(q_y^{WR}\right)^2 + \left(q_x^{WR}\right)^2 + \left(q_w^{WR}\right)^2 - \left(q_z^{WR}\right)^2\\
dgz_2 &= \left(q_y^{WR}\right)^2 - \left(q_x^{WR}\right)^2 + \left(q_w^{WR}\right)^2 - \left(q_z^{WR}\right)^2\\
dgz_3 &= -\left(q_y^{WR}\right)^2 - \left(q_x^{WR}\right)^2 + \left(q_w^{WR}\right)^2 + \left(q_z^{WR}\right)^2.
\end{aligned}
$$