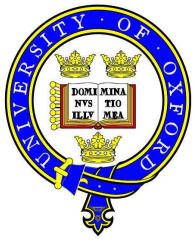
Active Vision Laboratory

Simultaneous Localisation and Mapping Using a Single Camera



Brian P. Williams
Pembroke College

D.Phil. Thesis
Supervised by Dr. Ian D. Reid

Robotics Research Group
Department of Engineering Science
University of Oxford

2009

Brian P. Williams
Pembroke College
University of Oxford

Doctor of Philosophy
Trinity Term
2009

Simultaneous Localisation and Mapping Using a Single Camera

Abstract

This thesis describes a system which is able to track the pose of a hand-held camera as it moves around a scene. The system builds a 3D map of point landmarks in the world while tracking the pose of the camera relative to this map using a process called simultaneous localisation and mapping (SLAM). To achieve real-time performance, the map must be kept sparse, but rather than observing only the mapped landmarks like previous systems, observations are made of features across the entire image. Their deviation from the predicted epipolar geometry is used to further constrain the estimated inter-frame motion and so improves the overall accuracy. The consistency of the estimation is also improved by performing the estimation in a camera-centred coordinate frame.

As with any such system, tracking failure is inevitable due to occlusion or sudden motion of the camera. A relocalisation module is presented which monitors the SLAM system, detects tracking failure, and then resumes tracking as soon as the conditions have improved. This relocalisation process is achieved using a new landmark recognition algorithm which is trained on-line and provides high recall and a fast recognition time.

The relocalisation module can also be used to achieve place recognition for a loop closure detection system. By taking into account both the geometry and appearance information when determining a loop closure this module is able to outperform previous loop closure detection techniques used in monocular SLAM. After recognising an overlap, the map is then corrected using a novel trajectory alignment technique that is able to cope with the inherent scale ambiguity in monocular SLAM.

By incorporating all of these new techniques, the system presented can perform as a robust augmented reality system, or act as a navigation tool which could be used on a mobile robot in indoor and outdoor environments.

Acknowledgements

Firstly, many thanks go to my supervisor Ian Reid who has been a wonderful source of inspiration and help throughout my DPhil. I greatly appreciated his encouragement and eloquence when it came to producing papers for publication and his help in guiding my research directions.

I would also like to thank Paul Newman and José Neira for examining and assessing this work. The discussion resulting from their questions and criticisms during the viva was both interesting and insightful. The final thesis has greatly benefitted from their comments and suggestions.

Thanks also go to all the members of the Active Vision group and the Mobile Robotics group at Oxford. They have always been there to discuss any problems I have had in my research over a cup of coffee or a pint at the Royal Oak. In particular, I would like to thank Paul Smith who helped me through the first year of my DPhil and provided an updated version of the SceneLib library which formed a basis for my own system. I would like to thank Georg Klein for his collaboration in the development of the relocalisation system described in this thesis. Thanks also go to Mark Cummins for sharing his FabMap code and allowing me to compare its performance to my own loop closure algorithm.

I would like to thank Andrew Davison. It was his monocular SLAM system which provided the inspiration and the starting point for the work in this thesis. Seeing his research on the Oxford website encouraged me to leave Imperial to do my DPhil at Oxford. Little did I know, he was already on his way to Imperial to start a new research group.

I am very grateful to all those at the Real Time Robotics group at the University of Zaragoza who generously invited me to visit their lab and have offered support throughout my DPhil. In particular I would like to thank Mingo Tardós, José Neira, and José María Montiel whose advice I benefitted from to ensure that my

ideas were implemented correctly. I would also like to thank Javier Civera who helped me work though the bugs in my code and also generously shared some of his own code which has been incorporated into my system. I am also grateful to the group in Zaragoza for sharing the code for their loop closure detection method and allowing me to compare it to my own technique. I also thank them for sharing the Keble College sequence on which I test my system.

For allowing me to film some of the other sequences used in this thesis, I would like to thank the Ashmolean Museum, the Natural History Museum, and Pembroke College. Something more interesting than 'an office environment' was needed when I knew I would spend hours looking at these same videos over and over.

Finally, I would like to thank my family and friends for their love and support throughout my DPhil. I could not have done it without you.

Mathematical Notation

General Notation

s	Scalars are roman
$\mathbf{v}$	Vectors are roman bold
$\mathtt{M}$	Matrices are courier bold

Extended Kalman Filter Notation

$\hat{\mathbf{x}}$	Mean of an estimate of the variable $\mathbf{x}$
$\mathtt{P}_x$	Covariance of the estimate of the variable $\mathbf{x}$
$\mathbf{C}$	Pose of the camera
$\mathbf{v}$	Linear velocity of the camera
ω	Angular velocity of the camera
$\mathbf{L}$	All landmarks in the map
$\mathbf{L}_i$	Position of the i^{th} landmark
$\mathbf{W}$	World reference frame
$\mathbf{M}$	Inter-frame motion of the camera
$\hat{\mathbf{x}}^W$	Mean of an estimate of $\mathbf{x}$ represented in the world coordinate frame
$\hat{\mathbf{x}}^C$	Mean of an estimate of $\mathbf{x}$ represented in the camera coordinate frame
k	timestep number
$\hat{\mathbf{x}}_{k\|k-1}$	Mean of an estimate of $\mathbf{x}$ at timetep k calculated using only the information up to timestep previous timestep, $k-1$. The result of the Prediction Step.
$\hat{\mathbf{x}}_{k\|k}$	Mean of an estimate of $\mathbf{x}$ at timetep k calculated using all information up to timestep k. The result of the Update Step.
$\mathbf{z}_k$	The measurements made at timestep k
$\mathbf{z}_{i,k}$	A measurement made of the i^{th} landmark at timestep k

Contents

1

Introduction

Vision is our most important sense for understanding the world around us and giving this ability to computers opens a vast range of beneficial new applications. A digital camera can provide a computer with huge amounts of information. The challenge lies in processing this data to extract meaningful information within a timeframe where it is still relevant.

This thesis focuses on using the data provided by a single video camera to build a geometric representation of the scene while estimating the motion of the camera as it moves through that scene. This process, called monocular simultaneous localisation and mapping (SLAM), has clear applications for robotics. A mobile robot attempting to safely traverse a room must continually determine its position

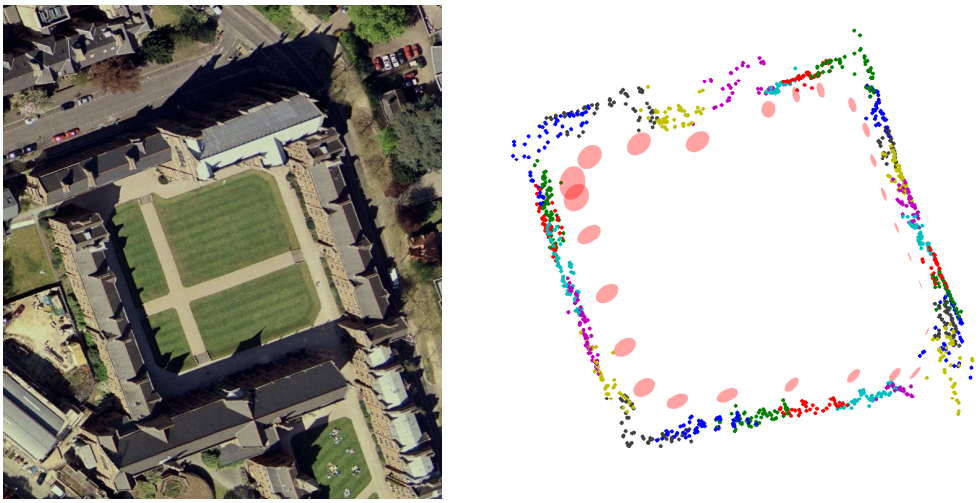


Figure 1.1: **Map Building and Localisation:** The monocular SLAM system presented in this thesis has applications in mobile robot navigation. To safely move around an environment, a robot can autonomously build a map of the environment and localise itself within that map. Using the monocular SLAM system presented in this thesis, a map of this university courtyard was built using the observations made with a single camera. The geometric map on the right can be compared to an aerial view of the courtyard on the left. To build a map of this size (approximately 70 metres on a side) our system automatically divided the environment into manageable submaps shown here in different colours. These submaps are made up of the estimated 3D position of natural point features in the scene observed by the camera. When the user walked completely around the courtyard, the system recognised that the original submap was coming back into view and was able to improve the overall estimate of the map. However, since imperfect sensors are used, some uncertainty in the estimate still exists and is shown here as one standard deviation uncertainty ellipses (pink) for the origin of each submap relative to the first submap in the lower right. Since the pose of the camera is tracked in real-time as the map is being built, the system could be used for navigation of a mobile robot fitted with a camera as it explores a new environment.

relative to the objects around it. Monocular SLAM allows metric maps of a scene to be built which could be used by robots for navigation purposes (Figure 1.1).

Other applications would also benefit from the inexpensive vision-based pose sensor that such a system would provide. One such example is augmented reality. This is the process of adding virtual objects to live video in such a way that they appear to be part of the real scene. The user is free to move the camera around the environment and view the virtual objects along with their real counterparts from any position. To achieve this effect, the pose of the camera relative to the real scene must be determined as each new frame arrives. Real-time monocular SLAM can provide this information (Figure 1.2).

In this thesis, we present a complete monocular SLAM system. We focus on achieving robust, real-time performance using a single, hand-held camera. Several new techniques are developed in this thesis which address some of the limitations of previous systems.

First, previous approaches to the problem of monocular SLAM will be discussed in Chapter 2. This allows us to discuss in detail the systems upon which this work is based as well as present their limitations which this work strives to overcome. Here, we also describe the underlying monocular SLAM system which we use. This system incorporates ideas from previous systems but also some new methods. It acts as a base for the principal new techniques presented in this thesis.

In Chapter 3, we present a method to improve the quality of the estimate produced by the monocular SLAM system. This is achieved by including a greater

Figure 1.2: **Augmented Reality:** An important application for monocular SLAM is augmented reality. This technique allows virtual objects to be added to live video using the pose of the camera relative to real objects in the world. Since our system provides the relative camera pose as each new frame arrives, the virtual character in this computer game (shown in colour) can be rendered on live video to make it appear to be running across on the real desk (shown in greyscale). The user is able to move the camera around the scene to view the virtual objects in the real setting from different angles.

number of observations to improve the estimate of the inter-frame camera motion. It is also shown that a robocentric framework can be applied to monocular SLAM to achieve improved consistency in the estimated camera position and map.

Then, in Chapter 4, a relocalisation module is presented which can be used to detect and recover from tracking failure. This can occur when the camera is occluded or the motion is too fast – a common occurrence when working with a hand-held camera. It is important that these situations are detected to prevent corruption of the map being built. The inclusion of the relocalisation module makes the overall system much more robust and is important in bringing these systems closer to use outside the lab.

Chapter 5 then discusses how this relocalisation system can be modified for use as a place recognition algorithm for loop closure detection. Without the ability to recognise when the camera has returned to a previously mapped region of the world, the system would be unable to accurately build large maps. A method is presented which can recognise when it has traversed a loop and determine the transformation needed to correct the map. This technique is also used to join independent maps built of the same environment. This loop closure detection method is shown to outperform previous loop closure detection methods used in monocular SLAM systems. Using this technique, the complete monocular SLAM system is shown to be able to reliably build maps of large multi-loop environments in real time.

Finally, in Chapter 6, conclusions are presented discussing the overall performance of the monocular SLAM system and the performance of its different components

presented in this thesis.

1.1 Contributions

The main contributions in this thesis are as follows:

- A novel method for utilising a large number of observations to improve camera motion estimation accuracy while retaining a just sparse map to facilitate real-time drift-free operation.
- The reformulation of the robocentric mapping framework for use in monocular SLAM to achieve a more consistent estimate.
- A new landmark recognition algorithm developed specifically for monocular SLAM which learns the appearance of landmarks online so they can be quickly found again after tracking failure.
- A relocalisation system which can detect tracking failure, prevent map corruption, and then restart tracking after determining the camera pose using our landmark recognition system and the map geometry.
- A method for recognising loop closure events well suited to monocular SLAM which establishes correspondences between the image and the appearance and geometry of the map built by the system.
- A trajectory based map alignment technique which is able to cope with the scale ambiguity inherent in the monocular SLAM problem. The alignment

is used to correct the map estimate in large maps after a loop closure has been detected.

In later chapters, our monocular SLAM system will be presented and these contributions will be further detailed. First though, we will discuss the related work done in this field, specifically highlighting the methods which have been incorporated into our own system.

2

Related Work

2.1 Introduction

Our goal is to track the pose of a camera as it moves through the world using the images recorded by the camera. The information needed to determine the camera motion must be extracted from only these images. As the camera moves, static features in the world will appear to move in the image. If the apparent motion of these features can be tracked, the motion of the camera can be inferred and a representation of the world can be created. Such a system is called a monocular SLAM system.

Many choices are available in creating a monocular SLAM system though. Which

features in the world should be selected to be tracked? How will they be recognised in subsequent images? How should the trajectory and world be represented? Which estimation algorithm should be used?

In this chapter we will discuss the design choices made in previous monocular SLAM systems in response to these questions. We will attempt to cover the broad spectrum of techniques used in previous systems, however, extra attention will be given to Davison's system [24]. His system was the first real-time monocular SLAM system and it forms the basis for the system described in this thesis.

First, we will discuss the many different ways that have been used to represent the world being observed by a monocular SLAM system. Then, we will show the different techniques used to recognise these landmarks in subsequent images. Finally, the different estimation algorithms used to determine the camera motion from the observations of the world will be examined.

2.2 World Representation

The appearance and geometry of most scenes in the world is very complex. However, with the vast amount of information provided by a video sequence, it is possible to build quite faithful representations of even these complex scenes. In the field of visual reconstruction, systems have been designed which can take high resolution images of a scene from several views and build a dense and accurate three dimensional model of the scene [69, 73]. Often, every pixel in each image is used to build a representation good enough to render photorealistic views of the

scene from novel camera poses. However, to represent and estimate such complex models of the world, vast amounts of memory and computation are needed.

For some applications, such dense representations are not needed and so are sacrificed for decreased computation time. A robot navigating a building for example needs to know the position of a potted plant blocking its path but not the detailed position of each leaf on the plant. Several sparse world representations have been used to try to capture just the essence of the structure of the environment while still providing good landmarks so that the camera pose can be tracked. In man-made environments planes and straight lines are common and are typically able to describe the broad structure of a scene very succinctly. Planar regions in the world can be recognised when viewed from multiple positions since all points on a plane can be mapped between the two views via a homography [36]. Alternatively, the 3D position of points in the world can first be determined and then planes can be robustly fit to this point cloud to find planes in the world [13]. A planar region must have sufficient texture for either of these techniques to work.

Straight lines can also be used in monocular SLAM to capture useful information about a scene. Edges can be detected in an image and then tracked in subsequent frames to determine their 3D position. Unlike planes, the estimate of an edge is not fully constrained during tracking as the motion along the edge is unobservable (unless additional texture information is used). Furthermore, some edge features in an image do not have a single 3D position in the world. The apparent edge of a curved surface seems to move as camera moves. Despite these difficulties, some good results have been obtained in including edges in the world representation in monocular SLAM systems [27, 41, 74].

For virtually all monocular SLAM systems point features have played an important role in the world representation. The memory and computation costs required to maintain a point feature estimate are very small since a point corresponds to a single 3D position in the world. These features are selected to be easily trackable in the image. They are typically chosen to have strong image intensity gradient changes in two orthogonal directions and thus motion in any direction in the image is observable.

Several interest point detectors to find such easily trackable point features have been proposed. The detector of Harris [35] and that of Shi and Tomasi [71] detect so called corner features. FAST [67] detects similar features but is optimised for speed. Blob-like features can be detected using the difference of Gaussian detector [46]. Blob-like features have an intrinsic scale which can make later recognition of the feature more straightforward. However, blob-like features are typically much slower to compute than corner features as the image must be analysed across several scales to find them.

Much work has been done on the detection, recognition, and tracking of point features. They can provide a good set of landmark observations to allow the pose of a camera to be accurately tracked. However, the representation of the world when using only points is far from complete. A set of points indicates discrete locations where an interest points exist in the world, but a lack of a point at a location does not imply that no object is present. This makes point-only representations insufficient for tasks like obstacle avoidance. Furthermore, occlusion of landmarks cannot be handled in point-only based representations as it can in dense visual reconstruction systems. For many applications though,

where the camera trajectory estimate is more important that the map, point representations are sufficient. Furthermore, when both the appearance and the geometry of a map of point landmarks is used, the representation can be rich enough to recognise and distinguish different locations in the scene as will be shown in Chapter 4 and Chapter 5. Point landmarks are used for the world representation in the system described in this thesis.

For some estimation algorithms, newly initialised landmarks require a different representation. When using a single camera as the only sensor, the depth of any point in the scene is unobservable in the first view. A new point feature seen in the image is only constrained to lie on the ray from the optical centre of the camera. The direction of this ray determines only two of the three degrees of freedom needed to represent a 3D point. To get around this difficulty, some systems have been forced to use a more complex representation for new landmarks, until the initially unobservable state can be estimated. Davison [23] used a particle filter to estimate the depth of a new point landmark before it was converted to a 3D representation. This technique was slow and the system could only initialise one or two new landmarks at a time. Montiel *et al.* [51] proposed parameterising a new landmark with inverse depth instead, since this probability distribution more closely matches the Gaussian assumptions used by their estimation algorithm. This representation requires six parameters and is fast enough to initialise several features at once. We use their method in our system to initialise new map landmarks. With other estimation algorithms, a point feature is not used until it is observed from multiple camera positions when its depth becomes observable. In these systems, a different initial parameterisation is not necessary, but they

do initially miss out on the information provided by these early observations of a new landmark.

Even sparse world representations like point-based ones can become computationally expensive to maintain as new regions are explored and the size of the map begins to grow. If the goal is simply to estimate the camera motion, then old map landmarks can be deleted as soon as they pass out of view of the camera. This is the approach used in visual odometry techniques [55]. These algorithms can provide estimates of arbitrarily long camera trajectories but the estimate can drift over time since the system is unable to recognise when it has returned to a previously seen area. In effect, the system believes it is continually exploring new territory and so it suffers from the drift which is common to all estimation algorithms during exploration if only local observations are available.

Another technique to reduce the computational burden of a growing map is to divide the map into submaps. Since the computation required for many estimation algorithms grows with the number of landmarks jointly estimated, this decoupling of the estimate into manageable pieces allows the algorithm to map larger areas with only a small reduction in optimality. The submapping technique used in our system is based on the approach of Clemente *et al.* [20], but several others have been proposed for monocular SLAM systems [12, 17, 30, 59]. Submapping will be discussed further in Chapter 5. Next, we will discuss methods used to recognise map landmarks when they are viewed by the camera.

2.3 Landmark Recognition

To measure the apparent motion of a map landmark in the image, the landmark must be recognised in subsequent frames. This process of matching an observation to a particular landmark is called data association. The appearance of a landmark can change due to changes in lighting, viewpoint, or noise, but methods have been developed to recognise landmarks despite these difficulties. Here we will focus only on recognition methods for point landmarks since these are used in our system. More complex landmarks like planes or lines require specialist recognition techniques.

A simple method for recognising a landmark is normalised cross correlation (NCC). When the landmark is first observed, a small image patch around the landmark is saved. In subsequent frames, provided the appearance has not changed too much, the landmark will still give a strong response to NCC. The normalisation makes this method invariant to uniform changes in brightness and with a suitable threshold, it is somewhat invariant to out-of-plane rotations. This simple method was used successfully in Davison's system [23]. However, if the patch is planar, and the normal has been estimated, then a homography can be used to warp the patch to match the predicted camera viewpoint. When warping is included NCC matching becomes quite invariant to viewpoint, allowing much more freedom for the camera motion as demonstrated in the system by Molton *et al.* [50]. The normal does not even need to be accurately estimated. In our system, the patches for point landmarks are warped to account for camera rotation assuming the patch normal is along the ray to the feature from which it

was first initialised. This makes the NCC matching invariant to a large enough range of camera viewpoints without the need of estimating a normal for each patch.

Unfortunately, matching based on NCC is susceptible to both false positives and false negatives. False negatives result in unsuccessful observations of landmarks and so the threshold is usually tuned to make sufficient successful landmark observations per frame quite likely. Of course, this increases the frequency of false positives too. If incorrect data association occurs, it can irreversibly corrupt the estimate produced by the estimation algorithm. All monocular SLAM systems must take steps to avoid this. Davison used a process called active search [23] to minimise the chances of NCC returning a false positive. In active search, the current estimate of the landmark relative to the camera is used to predict the current image location for that landmark. The uncertainty in this estimate is then used to determine a 99% confidence region in which to search for the landmark. NCC is only performed within this small region and the location with the best score is used for the observation. This process can be seen in Davison's system in Figure 2.3 on page 29.

Though active search greatly reduces the chance of false matches, they can still occur. In a region of the image with repeating texture, two locations within the search region may both give a strong NCC response. A single observation on its own can be insufficient to determine which is the correct match. By gating using a 99% confidence search region, active search performs an individual compatibility test on each observation. However, by analysing the position of the entire set of observations in each image, a jointly compatible set can be found and incorrect

matches can be identified and ignored. The joint compatibility branch and bound algorithm (JCBB) [52] can achieve this for sets of a few tens of observations. Beyond this, it becomes too computationally expensive as it scales exponentially with the number of observations. Joint compatibility is tested in a similar way to the individual compatibility test in active search but instead of using each the landmark position estimate individually, the correlations between the estimates of the entire landmark set are taken into account. The joint measurement of a set of landmarks together can be gated at a desired confidence level to identify incompatible sets of observations. A branch and bound interpretation tree search can then be used to find the largest jointly compatible set of observations within a chosen confidence level. The Mahalanobis distance is used as the criterion for branching so that the hypotheses with a higher degree of joint compatibility are explored first. JCBB is performed after the observations have been made to find the largest jointly compatible subset of these observations exclude any spurious measurements. JCBB was first used in monocular SLAM in [20]. Both active search and the JCBB algorithm are used in our SLAM system to recognise incorrect data association and prevent these observations from corrupting the estimation process.

An even more sophisticated observation method is the Active Matching technique of Chli & Davison [16]. They move the landmark observation and the search for a jointly consistent data association set into the same loop. In this loop, the next landmark to observe is chosen based on the expected Shannon information gain. The information gained as a result of this measurement reduces the size of the search region for the subsequent landmark measurements. This decreases

the overall image processing time for landmark observations. If any of the observations results in multiple potential matches to a landmark then each of these hypotheses are explored in an interpretation tree using a mixture of Gaussians approach. In their most recent work [15], instead of considering the complete correlation structure between the landmarks as encoded in the map estimate, they approximate this distribution with a Chow-Liu tree [18]. This tree is the optimal first order approximation of a joint distribution and it allows them to perform the search much faster with only a small reduction in accuracy.

NCC works for frame-to-frame tracking in monocular SLAM because the position of the camera relative to each landmark can be predicted allowing small search regions and patch warping. More sophisticated point landmark recognition algorithms have been developed to work in more challenging, wider baseline matching where a good prediction may not be available. Many of these techniques calculate a descriptor for each landmark which is designed to be invariant to viewpoint and lighting changes [7, 46, 48, 49]. To determine matches, a nearest neighbour search is used to find a match in descriptor space. These engineered descriptors provide robust matching even when the landmark's position and appearance cannot be predicted. They are unnecessary for frame-to-frame matching in monocular SLAM but, if the tracking fails, and the camera becomes lost, they can be far more useful. They will be discussed further in Chapter 4.

Another technique for wide baseline matching is to learn the appearance of a landmark. Rather than engineering invariance to viewpoint and lighting changes like descriptor based methods, this invariance can be learned if the appearance of the landmark in these situations is included in the training examples. These examples

can be extracted automatically from a sequence in a process called harvesting [57]. Randomised trees provide a framework for learning the appearance of a landmark over time. Once trained, the landmark can be quickly recognised again in any image. We have adapted the randomised trees method of Lepetit *et al.* [44] for use in monocular SLAM to allow our system to quickly relocalise after tracking fails. This landmark recognition technique will be discussed further in Chapter 4.

2.4 Estimation Algorithms

To estimate the pose of a moving camera, observations of the world relative to the camera are made using the images produced. Represented in homogeneous coordinates, a 3D point in the world, $\mathbf{X}$, is projected to a point in the image, $\mathbf{x}$, via the projection matrix, $\mathtt{P}$.

$$\mathbf{x} = \mathtt{P}\mathbf{X} \tag{2.1}$$

The projection matrix for a camera is defined by the camera's intrinsic and extrinsic parameters. The extrinsic parameters define the pose of the camera with respect to the scene using a rotation matrix, $\mathtt{R}$, and a translation vector, $\mathbf{t}$. The intrinsic parameters define the projective properties of the camera such as the focal length. These parameters are contained within the calibration matrix, $\mathtt{K}$.

$$\mathtt{P} = \mathtt{K}[\mathtt{R}|\mathbf{t}] \tag{2.2}$$

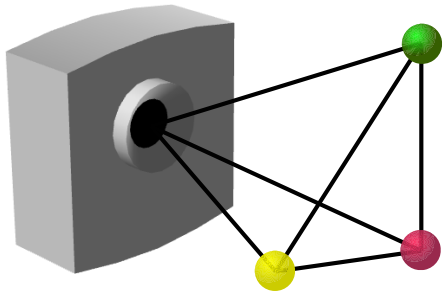


Figure 2.1: **Three-point-pose:** With just three observations of points with known 3D geometry, the camera pose is restricted to just four solutions. A fourth point observation can be used to disambiguate and find the true camera pose.

In the same way that the pose of a moving camera can be estimated using observations, the intrinsic properties of the camera can also be estimated. However, in the system presented in this thesis, the intrinsic properties of the camera do not change and so the camera can be calibrated beforehand. This reduces the number of parameters that need to be jointly estimated online.

The camera we use also has a wide angle lens which causes non-linear radial distortion. We use the Tsai camera model [81] with second order radial distortion modelling to account for radial distortion in observations. The code used for this model was written by Javier Civera[1] and the parameters are determined during the calibration phase before the monocular SLAM system is used.

As a camera moves through the world, the images captured show the scene from different views. A complete description of the geometry associated with multiple views can be found in the book by Hartley & Zisserman [36] but we will provide a brief summary here starting with the simplest cases.

[1]University of Zaragoza, Spain

If the geometry of the world is known, then the pose of a camera can often be determined from just a single view. Several algorithms exist to determine the pose from the image observations of many points, for which [45] provides a good review. In principle, the image measurements of four non-coplanar points, of known 3D position can fully constrain the camera pose relative to those points. In fact, with just three points, a discreet number of camera poses can be calculated (Figure 2.1). A fourth point can then be used to disambiguate between the (up to four) solutions given. A survey of three–point–pose algorithms can be found in [34]. This technique is used in the relocalisation process in our monocular SLAM system and will be further discussed in Chapter 4.

In other cases, when the geometry of the world is unknown, much can still be inferred from images of a scene. Two cameras, C and C', defined by their projection matrices, $\mathtt{P}$ and $\mathtt{P}'$, observe the same point in the world, $\mathbf{X}$, expressed in homogeneous coordinates. Without loss of generality, the first camera is assumed to be at the origin so that the extrinsic properties of the second camera define the pose of this camera relative to the first.

$$\mathtt{P} = \mathtt{K}[\mathtt{I}|\mathbf{0}] \quad \mathtt{P}' = \mathtt{K}'[\mathtt{R}|\mathbf{t}] \tag{2.3}$$

The 3D point, $\mathbf{X}$, is first projected into the image of the first camera yielding the 2D point, $\mathbf{x}$. The image of $\mathbf{X}$ in the second camera, $\mathbf{x}'$, is constrained to lie on a line in the image plane of the second camera called the epipolar line, $\mathbf{l}'$. All such lines begin at the epipole, $\mathbf{e}'$, which is a point in the image corresponding to the projection of the optical centre of the first camera. This geometry is illustrated in

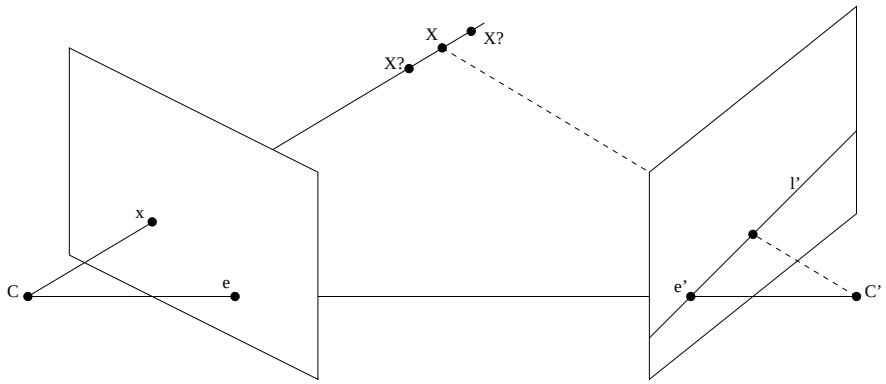


Figure 2.2: **Epipolar Geometry:** A 3D point, $\mathbf{X}$, observed in the first image at $\mathbf{x}$ must lie on the epipolar line, $\mathbf{l}'$, in the second image. All epipolar lines pass through the epipole, $\mathbf{e}'$, which is the projection of the optical centre of the first camera into the second image plane.

Figure 2.2. Both the epipole and the epipolar lines are defined by the fundamental matrix, $\mathtt{F}$, which is determined by the two cameras projection matrices.

$$\mathtt{F} = \mathtt{K}'^{-\top}[\mathbf{t}]_{\times}\mathtt{R}\mathtt{K}^{-1} \tag{2.4}$$

where $[\mathbf{t}]_{\times}$ represents the skew symmetric matrix formed form the components of the translation, $\mathbf{t}$.

The epipolar line is given by

$$\mathbf{l}' = \mathtt{F}\mathbf{x} \tag{2.5}$$

and the epipole by

$$\mathbf{e}'^{\top}\mathtt{F} = \mathbf{0}. \tag{2.6}$$

The epipolar geometry provides the constraint that for every pair of points imaged

by both cameras,

$$\mathbf{x}'^{\top}\mathtt{F}\mathbf{x} = 0. \tag{2.7}$$

If the intrinsic parameters are known, then the image points can be transformed into the ideal camera frame, where $\mathtt{K} = \mathtt{I}$. When ideal cameras are used, the essential matrix, $\mathtt{E}$, plays the role of the fundamental matrix for defining epipoles and epipolar lines.

$$\mathtt{E} = [\mathbf{t}]_{\times}\mathtt{R} \tag{2.8}$$

If the two image points are in the ideal camera frame then,

$$\mathbf{x}'^{\top}\mathtt{E}\mathbf{x} = 0. \tag{2.9}$$

Using this equation, the essential matrix can be computed from five point correspondences in two images. Typically, a larger number of correspondences are found and the essential matrix is determined via RANSAC [32] followed by a least squares optimisation [54].

The camera matrices can then be determined from the essential matrix in the following way. Without loss of generality, we will assume that the first camera matrix is $\mathtt{P} = [\mathtt{I}|\mathbf{0}]$ and the second camera matrix is $\mathtt{P}' = [\mathtt{R}|\mathbf{t}]$. First, the singular value decomposition (SVD) of the essential matrix is determined yielding

$$\mathtt{E} = \mathtt{U}\Sigma\mathtt{V}^{\top} \tag{2.10}$$

where $\mathtt{U}$ and $\mathtt{V}$ are orthogonal matrices and

$$\Sigma = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.11}$$

The diagonal entries of Σ are the singular values of $\mathtt{E}$ which must, according to the constraints of the essential matrix, consist of two identical values and one zero value. The camera matrices can only be determined from E up to an unknown scale, s.

The solutions for the translation and rotation specifying the second camera pose relative to the first is then given as the skew symmetric matrix formed from the translation

$$[\mathbf{t}]_\times = \mathtt{U}\mathtt{Z}\mathtt{U}^\top \tag{2.12}$$

and the rotation matrix which has two solutions

$$\mathtt{R} = \mathtt{U}\mathtt{W}\mathtt{V}^\top \text{ or } \mathtt{U}\mathtt{W}^\top\mathtt{V}^\top \tag{2.13}$$

where

$$\mathtt{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

and

$$\mathtt{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.15}$$

The solution is only given up to scale and a four-fold ambiguity. The ambiguity is due to the two soltuions for $\mathtt{R}$ and the unknown sign of s. This ambiguity can be resolved by noting that the point must be in front of both cameras, but the scale ambiguity is inherent in the problem and cannot be resolved. Another issue in this decomposition is so called degenerate motions which cause singularities. When the camera undergoes one of these motions (e.g. a pure rotation), the epipolar geometry does not fully constrain the motion.

Visual odometry systems like those developed by Nistér *et al.* [55] use these techniques to track the pose of a camera as it moves through a scene. Point correspondences are found between each pair of consecutive images using matching techniques like those discussed in Section 2.3. The essential matrix is then estimated and decomposed to determine the inter-frame motion at each timestep. The scale is kept consistent over the trajectory by observing some points across more than two frames. The goal in visual odometry is to estimate the trajectory of the camera. A temporary point cloud representing points in the world is simply a by product. Once a point passes out of view, it is forgotten. If it comes back into view and is measured again, it is treated as a new landmark. Over time, this causes drift in the pose estimation despite each incremental motion being very well estimated.

Unlike visual odometry systems, we wish to jointly estimate both the camera trajectory and a persistent representation of the scene structure. Only if landmarks are remembered and recognised when they come back into view, can drift in the camera trajectory estimate be avoided. However, estimating both at once can be seen as a chicken and egg problem. It is straightforward to navigate using a known map and it is straightforward to build a map using a sensor at known positions. If both are initially unknown though, and they are estimated simultaneously, the estimates become correlated. An uncertainty in the position of landmarks causes a corresponding uncertainty in the position of the sensor and vice versa. This problem is solved by estimating the two jointly in an estimation algorithm that can account for this coupling.

The earliest methods for determining scene structure from images were developed by the photogrammetry community [43]. For example, the structure of scenes in aerial photographs could be determined without even the use of computer algorithms. Later, when computers became available, the field of computer vision also tackled the problem (calling it 'structure from motion') and redeveloped some of the same techniques.

The key algorithm for structure from motion is called 'bundle adjustment' [80]. First, point landmarks are found in each image in a sequence. Correspondences are then found between the landmarks in different frames using a landmark recognition algorithm like those discussed in Section 2.3. Every landmark observed in more than one image provides a constraint both on the position of that landmark and on the pose of each camera observing the landmark. The goal then is to find a set of camera poses, $\mathtt{C}$, and 3D landmark positions, $\mathbf{X}$, which minimises the

reprojection error of the observations, $\mathbf{x}$. The projection of a 3D point into the image is represented by the function Π. The objective function (Equation 2.16) shows this for each of the m cameras and n observations where $v_{i,j}$ is a binary variable equal to 1 if the i^{th} landmark is visible in the j^{th} camera and 0 otherwise.

$$\min_{\mathtt{C}_j, \mathbf{X}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{i,j} (\Pi(\mathtt{C}_j, \mathbf{X_i}) - \mathbf{x}_i)^\top (\Pi(\mathtt{C}_j, \mathbf{X_i}) - \mathbf{x}_i) \tag{2.16}$$

Typically, this least-squares problem is solved using the Levenberg-Marquardt algorithm [62] which interpolates between the Gauss-Newton algorithm and gradient descent.

Bundle adjustment has had great success. One notable application is its use in the film industry to add computer generated special effects post production [26]. It can even work with uncalibrated cameras by including the intrinsic parameters in the estimation process [60]. Care must be taken though to detect both incorrect observations which can corrupt the estimate and degenerate motions where the estimate is under constrained [79]. A full review of bundle adjustment techniques can be found in [80].

Traditional bundle adjustment is a batch method. All of the images are processed in parallel off-line after the sequence has been recorded. This is fine for adding special effects for Hollywood movies but is unsuitable for on-line systems which require an estimate for the camera pose as the images are being acquired. The processing time constraints for off-line systems are also less strict and processing every frame of a sequence using bundle adjustment can be quite time consuming.

The robotics community has also tackled the problem of estimating the structure of the environment from a moving sensor but with the goal of real-time on-line processing. This is necessary so that the robot can use the information while navigating around an environment. In robotics, the problem is called simultaneous localisation and mapping (SLAM) and typically sequential, rather than batch techniques are used. At each timestep, the previous estimate of the scene structure and the current camera pose is updated using the new information gathered at that timestep.

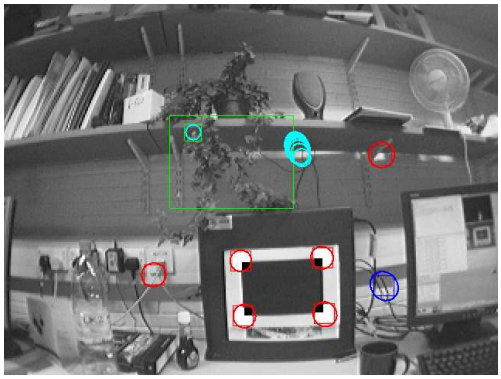
The importance of taking into account the correlations between the estimate for landmarks and the robot pose was shown in the seminal paper by Smith, Self & Cheeseman [75]. In this paper, they presented the Extended Kalman Filter(EKF) estimation method. The EKF allows a robot in an initially unknown environment to build a map of that environment while simultaneously navigating within the map being constructed. This ability is very important for a robot since it allows it to safely perform many tasks which would otherwise be impossible.

Much of the early work on robot navigation used sensors like ultrasound and lasers which provide both range and bearing measurements and were relatively easy to process in real time on the computers of the day. Vision offers much richer information but requires more sophisticated processing to get this information. The first system to achieve real-time SLAM in full 3D using a single camera as its only sensor was developed by Davison [24, 23]. His system uses the EKF framework to sequentially update an estimate for the camera pose and a map of point landmarks observed by the camera as it moves through a scene. Since his system provides a basis for the system presented in this thesis, it will now be

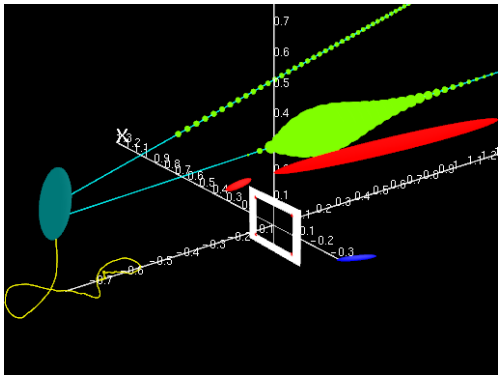
discussed in some depth.

2.4.1 Davison's System

Davison's system makes observations of point features using a hand-held calibrated camera with a wide angle lens. The system is initialised by observing a known target with four initial landmarks. This fixes the scale of the world and also provides landmarks to track the pose of the camera while new landmarks are chosen from the world and initialised. New landmarks are selected by placing a box in the image in a region currently without landmarks with preference given to regions the camera is moving towards. The pixel with the best Shi-Tomasi score [71], which will make it easy to track, is selected from within this box and is initialised as a new landmark. An 11×11 image patch centred on the landmark is saved to recognise the landmark in later images using NCC. Since no patch warping is done, the system is unable to recognise landmarks over large viewpoint changes. To account for this, the system notes the camera position when the landmark is initialised and does not attempt to observe that landmark if the current estimate for the camera pose is too different from this. As landmarks pass out of view, the system initialises new ones to try to maintain a sparse set of around ten visible landmarks at all times. Any landmark for which too many attempted observations fail is removed from the map to allow more promising landmarks to be initialised. The resulting sparse map is sufficient to provide an accurate estimate of the camera pose while keeping the total state to be estimated as small as possible so that the processing can be done at framerate.



(a) **Image View:** The system predicts the position of each landmark in the image and searches within the three standard deviation ellipse around this point. Six landmarks (red) have been successfully matched using NCC between the image and the patch stored for each landmark (□). One observation has failed (blue). A new landmark has been selected in this frame from within the green box. This box is placed to ensure a good distribution of landmarks across the image. A second landmark is still in the process of being initialised and the cyan ellipses show the search region for each depth hypothesis along the ray.



(b) **World View:** The 3D state estimate corresponding to the same timestep as the frame in (b) is shown. The camera position uncertainty ellipsoid (cyan) and the estimated trajectory (yellow) are on the left. Fully initialised landmarks are indicated by uncertainty ellipsoids whose colour corresponds to the success of the latest observation. The two partially initialised landmarks are indicated by a ray (cyan) with a green sphere for each depth hypothesis. Size of the sphere indicates the weight for that hypothesis. The new landmark has equal weight for every hypothesis whereas the older landmark is almost ready to be converted.

Figure 2.3: **Davison's System:** This system demonstrated the first real-time SLAM with a hand-held camera [24, 23]. The system is initialised with the camera in front of a known target with a landmark in each corner of the black rectangle. The system then adds to this initial map by selecting and initialising new natural landmarks in the scene. The observation of these landmarks allows the camera to be tracked as it moves out of view of the known target. The landmarks being observed in the image frame are shown in (a) and the 3D state estimate is shown in (b).

The EKF is a linear-Gaussian estimation algorithm and so approximates the state being estimated as a multi-variate Gaussian probability distribution. All states being estimated therefore should at least approximately conform to the Gaussian uncertainty assumptions used. Unfortunately, the probability distribution for the depth of a new landmark being initialised is highly non-Gaussian because the camera provides bearing only observations of landmarks. Therefore, the full state for new landmarks cannot be estimated initially by the EKF algorithm. Davison instead estimates the depth using a particle filter as discussed in Section 2.2. Only after the probability distribution becomes approximately Gaussian is the landmark fully estimated by the EKF. This initialisation process is costly and so only one or two new landmarks are initialised at a time. Figure 2.3 shows this process of landmark initialisation.

The EKF is used to maintain the estimate for the pose of the camera, $\hat{\mathbf{C}}^W$, and the 3D position vector, $\hat{\mathbf{L}}^W$, for each of the n initialised landmarks relative to the world reference frame W. Additionally, a constant velocity motion model is used so both the linear velocity relative to the world, $\hat{\mathbf{v}}^W$, and the angular velocity, $\hat{\omega}^C$, in the camera frame, C, are also estimated. The EKF maintains this joint estimate of the state, $\mathbf{x}$, as a multi-variate Gaussian probability distribution with mean, $\hat{\mathbf{x}}$, and covariance, $\mathtt{P}$.

$$\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \mathtt{P}) \tag{2.17}$$

The state vector and covariance matrix contain entries for each parameter being

estimated.

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{C}}^W \\ \hat{\mathbf{v}}^W \\ \hat{\omega}^C \\ \hat{\mathbf{L}}_1^W \\ \vdots \\ \hat{\mathbf{L}}_n^W \end{bmatrix} \tag{2.18}$$

$$\mathtt{P} = \begin{bmatrix} \mathtt{P}_C & \mathtt{P}_{Cv} & \mathtt{P}_{C\omega} & \mathtt{P}_{CL_1} & \dots & \mathtt{P}_{CL_n} \\ \mathtt{P}_{vC} & \mathtt{P}_v & \mathtt{P}_{v\omega} & \mathtt{P}_{vL_1} & \dots & \mathtt{P}_{vL_n} \\ \mathtt{P}_{\omega C} & \mathtt{P}_{\omega v} & \mathtt{P}_{\omega} & \mathtt{P}_{\omega L_1} & \dots & \mathtt{P}_{\omega L_n} \\ \mathtt{P}_{L_1 C} & \mathtt{P}_{L_1 v} & \mathtt{P}_{L_1 \omega} & \mathtt{P}_{L_1} & \dots & \mathtt{P}_{L_1 Ln} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathtt{P}_{L_n C} & \mathtt{P}_{L_n v} & \mathtt{P}_{L_n \omega} & \mathtt{P}_{L_n L_1} & \dots & \mathtt{P}_{L_n} \end{bmatrix} \tag{2.19}$$

The pose of the camera is represented using a 3D position vector for the translation and a quaternion for the rotation.

$$\hat{\mathbf{C}}^W = \begin{bmatrix} \hat{\mathbf{r}}^W \\ \hat{\mathbf{q}}^W \end{bmatrix} \tag{2.20}$$

Prediction Step

At each timestep, the new camera state is first predicted by applying a constant velocity motion model using the camera velocities being estimated by the EKF. In this model, it is assumed that linear and angular velocities are perturbed by uncoupled zero-mean Gaussian noise, $\mathbf{n}$, produced by unknown linear and angular

accelerations, $\mathbf{a}$ and α respectively. These accelerations over the inter-frame time, Δt, cause the instantaneous velocity noise $\mathbf{V}$ and Ω with covariance $\mathtt{Q}$.

$$\mathbf{n} = \begin{bmatrix} \mathbf{V}^W \\ \Omega^C \end{bmatrix} = \begin{bmatrix} \mathbf{a}^W \Delta t \\ \alpha^C \Delta t \end{bmatrix} \tag{2.21}$$

The predicted state at timestep k using all the information up to timestep $k-1$, is calculated according to this motion model using Equation 2.22.

$$\hat{\mathbf{x}}_{k|k-1} = \begin{bmatrix} \hat{\mathbf{r}}^W_{k|k-1} \\ \hat{\mathbf{q}}^W_{k|k-1} \\ \hat{\mathbf{v}}^W_{k|k-1} \\ \hat{\omega}^C_{k|k-1} \\ \hat{\mathbf{L}}^W_{1,k|k-1} \\ \vdots \\ \hat{\mathbf{L}}^W_{n,k|k-1} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{r}}^W_{k-1|k-1} + (\hat{\mathbf{v}}^W_{k-1|k-1} + \mathbf{V}^W) \triangle t \\ \hat{\mathbf{q}}^W_{k-1|k-1} \times q\left(\left(\hat{\omega}^C_{k-1|k-1} + \Omega^C\right) \triangle t\right) \\ \hat{\mathbf{v}}^W_{k-1|k-1} + \mathbf{V}^W \\ \hat{\omega}^C_{k-1|k-1} + \Omega^C \\ \hat{\mathbf{L}}^W_{1,k-1|k-1} \\ \vdots \\ \hat{\mathbf{L}}^W_{n,k-1|k-1} \end{bmatrix} \tag{2.22}$$

where $q\left(\left(\hat{\omega}^C_{k-1|k-1} + \Omega^C\right) \triangle t\right)$ denotes the quaternion defined by the axis-angle rotation about $\left(\hat{\omega}^C_{k-1|k-1} + \Omega^C\right) \triangle t$.

The covariance matrix is also affected by the motion model prediction to reflect the increase in uncertainty caused by the noisy accelerations.

$$\mathtt{P}_{k|k-1} = \mathtt{F}^\top \mathtt{P}_{k-1|k-1} \mathtt{F} + \mathtt{G}^\top \mathtt{Q} \mathtt{G} \tag{2.23}$$

where

$$\mathtt{F} = \frac{\partial \hat{\mathbf{x}}_{k|k-1}}{\partial \hat{\mathbf{x}}_{k-1|k-1}} \tag{2.24}$$

and

$$\mathtt{G} = \frac{\partial \hat{\mathbf{x}}_{k|k-1}}{\partial \mathbf{n}} \tag{2.25}$$

Since the map landmarks are assumed to be static, the map estimate is unaffected by the motion model prediction stage.

Observation Step

Next, observations of landmarks are made in the image using active search. The predicted location in the image, $\mathbf{z}_{k|k-1}$, for all the landmarks is calculated using the observation model h.

$$\mathbf{z}_{k|k-1} = h(\hat{\mathbf{x}}_{k|k-1}) \tag{2.26}$$

This projects the landmark into the image using the camera calibration parameters and the estimated relative position between the landmark and the camera.

The observation model also assumes that measurements are subject to noise, $\mathtt{R}_K$, which is assumed to have a variance of 1 pixel. Using this noise estimate and the uncertainty in the relative position estimate of the camera and the landmark, a covariance, $\mathtt{S}_k$, for the predicted observation can be calculated.

$$\mathtt{S}_k = \mathtt{H}_k^\top \mathtt{P}_{k|k-1} \mathtt{H}_k + \mathtt{R}_k \tag{2.27}$$

where

$$\mathtt{H}_k = \frac{\partial \mathbf{z}_{k|k-1}}{\partial \hat{\mathbf{x}}_{k|k-1}} \tag{2.28}$$

The system uses $\mathtt{S}_k$ to define an elliptical search region for each landmark where the true observation is expected to be made. This search region is used for the active search process to make the observations.

Update Step

After the observations of landmarks have been made, the measured values, $\mathbf{z}_k$, are compared to the values predicted based on the state estimate. The innovation, $\boldsymbol{\nu}$, is the difference between the prediction and the true observations.

$$\boldsymbol{\nu}_k = \mathbf{z}_k - \mathbf{z}_{k|k-1} \tag{2.29}$$

This innovation is then used to update the state estimate based on the latest observations.

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathtt{W}_k \boldsymbol{\nu}_k \tag{2.30}$$

and

$$\mathtt{P}_{k|k} = \mathtt{P}_{k|k-1} - \mathtt{W}_k \mathtt{S}_k \mathtt{W}_k^\top \tag{2.31}$$

where

$$\mathtt{W}_k = \mathtt{P}_{k|k-1} \mathtt{H}_k^\top \mathtt{S}_k^{-1} \tag{2.32}$$

Each new frame is processed in this way to sequentially update the estimate for

the camera pose and map structure. Crucially, Davison's system was able to perform all the steps within the 33ms budget required for framerate operation. His system was the first to achieve this goal of framerate tracking in previously unknown environments.

2.4.2 Shortcomings of Davison's System

Davison's system performs well in ideal conditions but there are several issues which limit the performance of the system in more general use. To keep processing time small, the map is kept as sparse as possible. The update time for the EKF algorithm is $O(N^2)$ where N is size of the state being estimated. Typically, a dozen landmarks are in view in each frame and only the observations of these landmarks can constrain the pose estimate of the camera. If more observations could be used, the estimate for the camera trajectory could be improved, but a dense map of landmarks cannot be maintained in this framework if real-time operation is required. In Chapter 3 we present our method for improving the camera motion estimate by increasing the number of observations made without increasing the size of the state being estimated.

Others have also presented systems which allow larger numbers of observations per frame by using other estimation frameworks. If the FastSLAM algorithm [78] is used rather than the EKF, a much larger number of map landmarks can be used. This algorithm conditions the map on the camera trajectory, decoupling each landmark estimate so that each can be estimated by an individual EKF. A probability distribution for the camera trajectory is then maintained using

a particle filter. This algorithm was applied to monocular SLAM by Eade & Drummond [28] and their system was able to maintain much denser maps of point landmarks in real-time. Unfortunately, over large trajectories, FastSLAM requires an infeasible number of particles to provide a good estimate [4].

Klein & Murray [40] were also able to use a large number of observations per frame by using a different estimation framework. Their system combines the batch bundle adjustment techniques with a tracking system which can estimate the camera pose with respect to a fixed map. As the camera explores the scene, keyframes are selected if the latest view is sufficiently different from the current set of keyframes. The poses of the camera and landmarks visible in this keyframe set are estimated using Levenberg-Marquardt bundle adjustment. This time consuming process is done in a background processing thread whenever new keyframes are added to the set. In a parallel thread, the camera pose is tracked at each frame using the current map estimated by the bundle adjustment system. This approach allows a very dense landmark set leading to a very stable estimate for the camera trajectory. However, the update time for the bundle adjustment grows as new keyframes are added. This limits the size of the environment the system can work in.

Davison's system is also limited in the size of an environment it can map before it begins to run below framerate. A map of less than 70 landmarks can be maintained on modern computers which limits the system to environments about the size of a single room. Submapping techniques allow larger scenes to be explored. Our own system uses submapping to build larger maps and addresses the loop closure problem which becomes apparent in larger environments (Chapter 5).

Submapping has been demonstrated in several other monocular SLAM systems using the EKF algorithm to maintain each submap [17, 20, 59].

In another system by Eade & Drummond [29], a different estimation algorithm is used to build maps of larger environments. In this system, new observations of landmarks are incorporated into a graph of local maps. Nodes are chosen to minimise error resulting from the non-linear observation model in monocular SLAM. The number of landmarks in each local map is limited which allows real-time operation over large environments and very consistent maps are built by optimising the graph structure using shared landmarks between the nodes. This computation is made tractable by constructing the submaps in such a way as to ensure statistical independence between the local maps. Both the local and global map structure is optimised using the Levenberg-Marquardt algorithm from bundle adjustment. Their system was later extended to make use of loop closure and map joining techniques, further improving the performance of the system [30]. These aspects of their system will be discussed further in Chapter 5 where our own loop closing and map joining method is presented.

The EKF estimation algorithm used in Davison's system has been shown to be inconsistent [3]. The algorithm tends to be overconfident in the estimate calculated due to linearization errors. As well as improving the computational efficiency of mapping large environments, submapping also improves the quality of the estimate by decreasing these linearization errors [3, 11]. Other monocular SLAM systems have also attempted to improve the quality of the estimate by using different estimation algorithms.

Chekhlov *et al.* [14] used the Unscented Kalman Filter(UKF) for their monocular SLAM system. This algorithm improves the consistency of the estimate by making more accurate approximations to the non-linearities in the observation process. The improvement in consistency comes at a cost in complexity ($O(N^3)$ vs. $O(N^2)$ for the EKF). Recently, Holmes *et al.* [38] showed that using the square root UKF algorithm [84] a more consistent estimate can be obtained at $O(N^2)$ complexity in a monocular SLAM system. However, the actual processing time required for the implementation was an order of magnitude greater than for the EKF on reasonably sized maps due to a large constant factor on the computational cost in their implementation.

Our own system improves the consistency of the estimate by using the robocentric framework [11]. This method reduces linearization errors by representing the map in a frame centred on the current camera position. This method will be discussed in Chapter 3.

The final problem with Davison's system is the fragility of the tracking system. The EKF algorithm relies on getting good observations at each frame to track the pose of the camera. When these observations fail, his system is unable to recover. Unfortunately, in all but ideal circumstances, the failure of observations for some frames is inevitable. The camera can easily be occluded or a sudden motion can cause motion blur. During these situations, the motion model is unlikely to accurately predict the camera trajectory for more than a couple frames and the active search process relies on a good prediction for the pose of the camera. It's true that without observations, the search regions would grow to reflect this uncertainty but this also increases the chances of an observation with

incorrect data association. These observations corrupt the map and the system must be restarted from scratch. Our system addresses this problem and can detect and recover from tracking failure, leading to more robust tracking performance. Like ours, other systems since Davison have also included recovery from tracking failure[14, 30, 41, 63]. This problem is discussed further in Chapter 4.

In the next section, we present our own monocular SLAM system which acts as a base upon which we have developed the new techniques presented in this thesis. Our base system uses some components and techniques developed by other researchers in monocular SLAM and these aspects will be highlighted.

2.5 Our Monocular SLAM System

Our system builds upon the system of Davison [24]. We track the pose of a hand-held camera as it is moved through a previously unknown scene by jointly estimating the pose of the camera and the structure of a sparse map of point landmarks in 3D using the EKF framework. Landmarks are represented using the inverse depth parameterisation [51] allowing multiple landmarks to be initialised simultaneously. This allows the system to begin tracking without the initial known landmarks used in Davison's system. They can still be used however, if the true scale of the map is required.

The map is kept sparse to allow framerate (30Hz at 640×480 pixels per frame) update of the estimate. Landmarks for which observations fail too often are removed from the map so that the limited processing time can be used for more reliably

observed landmarks. If the number of visible landmarks falls below a threshold (usually 10-25), new landmarks are initialised in the image in locations without any landmarks. New landmarks are selected from easily tracked interest points detected in the image using FAST corner detector [67] followed by thresholding of the Shi and Tomasi [71] score. We also aim to choose landmarks distinctive from those already in the map to aid relocalisation (Chapter 4). Map landmarks are then observed in each frame using normalised cross correlation (NCC) after warping the initial patch to match the view from the predicted camera location. The warping is done assuming the patch is planar and that the normal lies along the ray between the landmark and camera pose when the landmark was initialised. This is sufficient to allow the patch to be tracked over a wide range of views. To limit the chance of false matches, active search [24] and the JCBB algorithm [52] are used. Incompatible observations are marked as false and are not incorporated into the state update. This allows these unreliable landmark to be safely deleted. If any observations of a landmark after its initialisation are used to update the state estimate, the information from these observations will influence all subsequent estimates of the entire state. Removing the landmark from the state vector does not remove the influence of its past observations on the state estimate. Instead, we try to recognise incorrect measurements and prevent them from being included in any state updates.

The image locations of observed landmarks are measured to sub-pixel accuracy using the method of Kanade & Lucas [5]. The registration is performed using a Newton style (2^{nd} order) optimisation. Like Davison, we assume a symmetric measurement noise, $\mathtt{R}$, of 1 pixel standard deviation. A more accurate approach

would be to determine the uncertainty of the registration on a per match basis. The Hessian, $\mathtt{H}$ of the optimisation can be used to infer the covariance of the registration ($\mathtt{R} = 2\mathtt{H}^{-1}$). This calculation would properly account for the different measurement uncertainty of different measurements based on the landmark appearance. For instance if the image patch for a landmark contained only a straight gradient edge then the measurement would be more certain in the direction perpendicular to the edge than along it. In our system though all new landmarks are selected at corner points with strong gradient changes in two roughly perpendicular directions and so the uncertainty of the registration will generally be more symmetric. Our use of 1 pixel standard deviation is an overestimate though. In an experiment to determine the measurement noise of observations of typical landmarks using the Hessian of the registration we found a mean standard deviation of 0.05 pixels. In practice though, this overestimate in the measurement noise makes only a small difference to the system as the largest part of the innovation covariance, $\mathtt{S}$, comes from the uncertainty in the estimated position of the landmark from the camera pose predicted using the motion model (Equation 2.27).

In subsequent chapters we present new techniques we have developed to address the problems of estimation quality, performance in larger environments, and robustness to tracking failure. These use the base monocular SLAM system described here. In each chapter, we provide more detail about work related to these problems to supplement the overview given in this chapter.

3

Robocentric Monocular SLAM with Visual Odometry

The results in this chapter have been published in the proceedings of the International Conference on Robotics and Automation (ICRA), 2010 [89].

3.1 Introduction

The accuracy of a monocular SLAM system depends both on the observations made by the system, and on the algorithm used to perform the estimation. In this chapter, we propose a new system which achieves accurate tracking through

the use of both a greater density of observations and a framework which provides higher quality estimates. This is achieved while maintaining a persistent map of landmarks which is important for drift-free tracking and recovery from tracking failures (Chapter 4).

Visual SLAM systems, like Davison's [24], provide drift-free motion estimation because the motion is estimated relative to a persistent map of landmarks. The same landmarks are recognised and reobserved each time they come back into view. However, due to the complexity of jointly estimating both the camera pose and the map, only a sparse map can be maintained in real-time. Typically only about 10-20 landmarks are visible in any one frame which limits the number of constraints on the motion. Even these observations are not guaranteed to be successful and when too many of these privileged observations fail, the camera pose can be under constrained causing a decrease in estimation accuracy.

In contrast, visual odometry systems, like that of Nistér *et al.* [55], attempt to estimate the motion of a moving camera as accurately as possible using every available point match between frames. These systems achieve real-time motion estimation with a much larger number of observations per frame than systems like [24] because a persistent map is not maintained. Landmarks are forgotten as soon as they pass out of view which allows constant time updates but can cause drift in the estimation over time. Despite this, the relative local motion is usually very well estimated due to the large number of measurements used. These systems use these frame-to-frame matches to estimate the epipolar geometry and from this, determine the relative camera motion between frames. This approach can fail when the camera undergoes degenerate motions (such as a pure rotation)

for which the epipolar geometry does not fully constrain the motion.

We aim to produce a monocular SLAM system which retains the drift-free properties of visual SLAM systems with a persistent map while incorporating the additional information provided by visual odometry style measurements into the filter. By observing apparent image motion of many features, rather than only a select few from the map, we improve the accuracy of ego-motion estimation in monocular SLAM, both by the effect of noise cancellation from many measurements, and also by overcoming failure-modes of monocular SLAM, such as when there are too few map-to-image matches to constrain the ego-motion.

While landmark observations provide constraints on the absolute position of the camera in the map, two-frame point matches only provide constraints, via the epipolar geometry, on the relative motion of the camera. Such constraints are naturally incorporated into a filter recast from a world-centric frame into a camera-centric frame. To that end we derive the appropriate formulation of robocentric SLAM [11] for a visual sensor, and show that (as expected) this also yields a more consistent estimate of the filter's uncertainty.

Recently, Civera *et al.* [19] have also presented a monocular SLAM system which uses the robocentric framework and visual odometry style observations. The observation of all point features are handled by including them as temporary landmarks in a transient map. Once the landmark passes out of view of the camera, it is removed from the map and so their system cannot benefit from the reobservation of old landmarks to prevent drift. They obtain accurate estimates for the motion of the camera over long sequences (using GPS as ground truth)

but they do not return to previously mapped areas where drift would become apparent. To detect outliers amongst the large number of observations they present a new technique called 1-point RANSAC since the JCBB algorithm [52] would be too slow. Worldcentric mapping is impractical in a system with a transient map, so their system is the first to report the use of robocentric mapping for a visual sensor. Nevertheless they do not report experiments to verify the benefits of this framework in terms of filter consistency as we will here. Their system is currently unable to achieve real-time operation, requiring about one second to process each frame.

In this chapter, we will present our monocular SLAM system which uses both visual odometry and the observations of 3D landmarks within a sparse persistent map to produce an accurate drift-free estimate of the camera trajectory and map structure. We also show how the quality of the estimate is improved by using the robocentric rather than traditional worldcentric estimation framework through experiments on both real and simulated data.

3.1.1 Contributions

The main contributions presented in this chapter are as follows:

- *Reformulate of the monocular SLAM filter into a robocentric framework.* This naturally leads to a more consistent estimate where the estimated uncertainty more closely matches the true uncertainty.
- *Combined visual odometry with sparse map based EKF monocular SLAM.*

This allows the system to benefit from many more observations while the update time remains $O(n^2)$ with the size of the sparse map. This leads to less noisy estimates for the camera pose and more accurate tracking of long trajectories.

3.1.2 Outline

In this chapter we present two key innovations to improve the quality of the estimate produced by our monocular SLAM system; the robocentric framework and the addition of visual odometry observations to a sparse EKF estimation framework. Each of these will be discussed in turn, and then results will be given on the performance of the system. These results include both real-world scenarios and simulations to asses the quality of the estimate provided. Finally, conclusions will be given.

3.2 Robocentric Mapping

Estimation algorithms like the EKF provide both an estimate of a state and a measure of the uncertainty of that estimate. Since the observations used to estimate the state are noisy, the estimate given by the algorithm cannot be expected to perfectly match the true value of the state. However, a good estimation algorithm should provide an unbiased estimate with an uncertainty which reflects the true uncertainty in that estimate. An estimation algorithm with these properties

is said to be consistent.

Unfortunately, the EKF algorithm has been shown to suffer from the problem of inconsistency. By running simulations so that the estimate can be compared to the ground truth, Bailey *et al.* [3] showed that the EKF algorithm greatly underestimates the uncertainty in the estimate it provides. This overconfidence causes problems for several aspects of SLAM. In techniques like active search, the relative uncertainty between a landmark and the camera is used to define a search region for the landmark observation. A three standard deviation uncertainty ellipse used as a search region would be expected to have a 99% probability of truly containing the landmark, but if the estimate is inconsistent this will not be the case. Furthermore, an overconfident filter is prone to large jumps in the state estimate. If the camera measures a landmark with an underestimated uncertainty, the too small innovation covariance will cause a correspondingly large update in the state estimate rather than the smooth update that a consistent algorithm would provide.

Bailey *et al.* found that the EKF becomes inconsistent after the angular uncertainty grows beyond a few degrees. Before the uncertainty grows to this point, the estimate and its uncertainty match the ideal values for a Gaussian estimator. Once the angular uncertainty grows too large though, the linearization errors in the EKF algorithm begin to corrupt the estimate leading to an overconfident result.

Castellanos *et al.* [11] have proposed a more consistent EKF-based SLAM algorithm called robocentric SLAM. In their approach, the state is represented in

a frame relative to the current position of the robot. With this framework, the robot is always positioned at the origin with zero uncertainty in the pose. All map landmarks and their uncertainties are represented relative to this. When the estimates are in the robot's reference frame, the level of angular uncertainty is low for the nearby landmarks being observed. This reduces the linearization errors and so the state estimate for robocentric SLAM is more consistent.

We have applied the robocentric mapping technique to monocular SLAM to produce a more consistent monocular SLAM system than the worldcentric approach used in Davison's original system. Recently, the robocentric approach has also been used in the monocular SLAM system of Civera *et al.* [19]. Their work does not provide results on the consistency of the resulting system. We will provide results for the consistency of our own system later in this chapter.

Figure 3.1 shows our monocular SLAM system operating in the robocentric frame. As the camera moves around the world observing different regions of the map, the position of the camera in the robocentric representation remains fixed at the origin while the map of landmarks moves around it. The uncertainty ellipses in the figure show the small uncertainty in the estimated position for the nearby landmarks and the growth in uncertainty as the camera moves to another part of the map. Relative to the camera, the position of distant landmarks not being observed is more uncertain than those currently in front of the camera. It is this property of the robocentric approach which leads to the decrease in linearization error.

In the following sections, we outline the steps of the robocentric mapping method

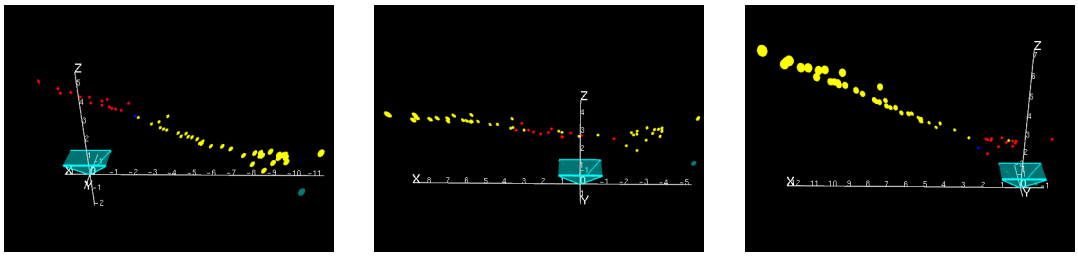


Figure 3.1: **Robocentric Mapping:** In the robocentric frame the camera pose estimate remains at the origin with zero uncertainty. Landmarks near the camera currently being observed tend to have low uncertainty while landmarks in other parts of the map have larger uncertainty. This change in uncertainty can be seen in the relative size of the landmark uncertainty ellipsoids as the camera is moved to different locations relative to this map in these three images. The origin of the base reference frame relative to the camera position can be seen with its respective uncertainty as a cyan ellipse in the lower right.

for monocular SLAM. Unlike Castellanos *et al.*, we use a hand-held camera as our only sensor rather than a wheeled robot with odometry. This leads to differences in the implementation of robocentric mapping which will be discussed. As well as providing a more consistent estimate for monocular SLAM, this framework is well suited to visual odometry observations which will be discussed afterwards in Section 3.3.

3.2.1 Robocentric State Representation

In the robocentric framework, the state, $\mathbf{x}$, is parameterised as a multi-dimensional Gaussian distribution represented in the coordinate frame centred on the pose of the camera, C.

$$\mathbf{x}^C \sim \mathcal{N}(\hat{\mathbf{x}}^C, \mathtt{P}^C) \tag{3.1}$$

where a superscript indicates the reference frame for the estimate.

Unlike in the worldcentric framework, a state vector entry for the camera pose is not needed. In the reference frame of the camera, the camera pose is known with certainty. It is simply located at the origin with no rotation. An entry for the set of landmarks, $\hat{\mathbf{L}}^C$, is still needed but is now estimated relative to the current camera pose. Since we use the constant velocity motion model, an entry for estimating the linear, $\hat{\mathbf{v}}^C$, and angular velocities, $\hat{\omega}^C$, is also needed. Finally, to allow conversion back to the worldcentric frame, the origin and orientation of the world reference frame, $\hat{\mathbf{W}}^C$, must also be estimated and so it is placed in the state vector relative to the camera as a non observable feature. The state vector, $\hat{\mathbf{x}}$, for monocular SLAM in the robocentric framework is

$$\hat{\mathbf{x}}^C = \begin{bmatrix} \hat{\mathbf{W}}^C \\ \hat{\mathbf{v}}^C \\ \hat{\omega}^C \\ \hat{\mathbf{L}}^C \end{bmatrix} \tag{3.2}$$

and the corresponding covariance matrix, $\mathtt{P}$, is

$$\mathtt{P}^C = \begin{bmatrix} \mathtt{P}^C_W & \mathtt{P}^C_{Wv} & \mathtt{P}^C_{W\omega} & \mathtt{P}^C_{WL} \\ \mathtt{P}^C_{vW} & \mathtt{P}^C_{v} & \mathtt{P}^C_{v\omega} & \mathtt{P}^C_{vL} \\ \mathtt{P}^C_{\omega W} & \mathtt{P}^C_{\omega v} & \mathtt{P}^C_{\omega} & \mathtt{P}^C_{\omega L} \\ \mathtt{P}^C_{LW} & \mathtt{P}^C_{Lv} & \mathtt{P}^C_{L\omega} & \mathtt{P}^C_{L} \end{bmatrix}. \tag{3.3}$$

3.2.2 Prediction Step

Like the worldcentric approach, the first step in the robocentric EKF is to predict the motion of the camera at timestep k using the information up until the previous timestep, $k-1$. Using the motion model, the incremental motion, $\hat{\mathbf{M}}^C_{k|k-1}$, between timestep $k-1$ and timestep k is calculated.

In the *worldcentric* approach, the predicted pose for timestep k is then calculated by composing the old camera pose estimate, $\hat{\mathbf{C}}^W_{k-1|k-1}$, with the incremental motion.

$$\hat{\mathbf{C}}^W_{k|k-1} = \hat{\mathbf{C}}^W_{k-1|k-1} \oplus \hat{\mathbf{M}}^W_{k|k-1} \tag{3.4}$$

where $\oplus$ denotes the coordinate frame composition operation.

At the same time, the uncertainty in the motion prediction, $\mathtt{P}^W_M$, is transferred onto the uncertainty of the camera pose estimate. Typically, these two steps (incremental motion calculation and composition) are combined into a single function. The incremental motion is predicted and then immediately marginalised out in updating the camera position.

In the *robocentric* framework however, the camera remains at the origin. So, for the prediction step, it is instead the structure of the entire stochastic map which is transformed to reflect the incremental motion. The i^{th} landmark can be transformed in this way by composing the estimate of the landmark $\hat{\mathbf{L}}^C_{i,k-1|k-1}$ with the inverse of the incremental motion prediction.

$$\hat{\mathbf{L}}^C_{i,k|k-1} = \ominus\hat{\mathbf{M}}^C_{k|k-1} \oplus \hat{\mathbf{L}}^C_{i,k-1|k-1} \tag{3.5}$$

where $\ominus$ denotes the inverse operation for a frame transformation.

As well as moving the landmark position estimate into the predicted camera frame, the marginalisation process also transfers the uncertainty in the motion estimate onto the landmark position uncertainty. This is in contrast to the world-centric framework where only the camera pose uncertainty is inflated at the prediction step.

However, rather than performing the composition described in Equation 3.5 at this stage, Castellanos *et al.* propose to delay it. The predicted motion is usually quite uncertain and to use it to transform the map here could introduce linearization errors. Instead, the motion prediction itself is included in the state vector. Only after observations have been made and the update is performed, will the then refined motion estimate be marginalised out and used to transform the map into the new reference frame. The prediction step for robocentric SLAM therefore, is to predict the incremental motion since the previous timestep, and to place it in the state vector

$$\hat{\mathbf{x}}^C_{k|k-1} = \begin{bmatrix} \hat{\mathbf{x}}^C_{k-1|k-1} \\ \hat{\mathbf{M}}^C_{k|k-1} \end{bmatrix} \tag{3.6}$$

with corresponding covariance

$$\mathtt{P}^C_{k|k-1} = \begin{bmatrix} \mathtt{P}^C_{k-1|k-1} & \mathtt{0} \\ \mathtt{0} & \mathtt{P}^C_{M_{k|k-1}} \end{bmatrix}. \tag{3.7}$$

In our system we use a hand-held camera as the sole sensor and therefore have no odometry. Instead, we get the inter-frame incremental motion prediction from the linear and angular velocity which is also estimated by the filter. We use the same constant velocity motion model used by Davison [24].

The inter-frame motion in the time $\triangle t$ is represented by a translation, $\mathbf{r}$, and a quaternion, $\mathbf{q}$. The velocities used to predict this motion are represented in the state vector by a three component linear velocity vector, $\hat{\mathbf{v}}^C$, and a three component angular velocity vector, $\hat{\omega}^C$. With a constant velocity motion model it is assumed that these velocities are perturbed by uncoupled zero-mean Gaussian noise, $\mathbf{n}$, produced by unknown linear and angular accelerations, $\mathbf{a}$ and α respectively. These accelerations cause the instantaneous velocity noise $\mathbf{V}$ and Ω with covariance $\mathtt{Q}$.

$$\mathbf{n} = \begin{bmatrix} \mathbf{V}^C \\ \Omega^C \end{bmatrix} = \begin{bmatrix} \mathbf{a}^C \triangle t \\ \alpha^C \triangle t \end{bmatrix} \tag{3.8}$$

The velocity is predicted to be perturbed by the noise $\mathbf{n}$.

$$\begin{bmatrix} \hat{\mathbf{v}}^C_{k|k-1} \\ \hat{\omega}^C_{k|k-1} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{v}}^C_{k-1|k-1} + \mathbf{V}^C \\ \hat{\omega}^C_{k-1|k-1} + \Omega^C \end{bmatrix} \tag{3.9}$$

At each timestep, the estimated velocity from the previous timestep is used to predict the inter-frame incremental motion.

$$\hat{\mathbf{M}}^C_{k|k-1} = \begin{bmatrix} \hat{\mathbf{r}}^C_{k|k-1} \\ \hat{\mathbf{q}}^C_{k|k-1} \end{bmatrix} = \begin{bmatrix} \left(\hat{\mathbf{v}}^C_{k-1|k-1} + \mathbf{V}^C\right) \triangle t \\ q\left(\left(\hat{\omega}^C_{k-1|k-1} + \Omega^C\right) \triangle t\right) \end{bmatrix} \tag{3.10}$$

where $q\left(\left(\hat{\omega}^C_{k-1|k-1} + \Omega^C\right) \triangle t\right)$ denotes the quaternion defined by the axis angle rotation about $\left(\hat{\omega}^C_{k-1|k-1} + \Omega^C\right) \triangle t$.

The incremental motion is placed in the vector for the predicted state at time k given the observations up to time $k-1$.

$$\hat{\mathbf{x}}^C_{k|k-1} = \begin{bmatrix} \hat{\mathbf{W}}^C_{k|k-1} \\ \hat{\mathbf{v}}^C_{k|k-1} \\ \hat{\omega}^C_{k|k-1} \\ \hat{\mathbf{L}}^C_{k|k-1} \\ \hat{\mathbf{M}}^C_{k|k-1} \end{bmatrix} \tag{3.11}$$

The covariance matrix must also be updated according to the motion model prediction as follows.

$$\mathtt{P}^C_{k|k-1} = \mathtt{F}_k \mathtt{P}^C_{k-1|k-1} \mathtt{F}^\top_k + \mathtt{G}_k \mathtt{Q} \mathtt{G}^\top_k \tag{3.12}$$

where

$$\mathtt{F}_k = \frac{\partial \hat{\mathbf{x}}^C_{k|k-1}}{\partial \hat{\mathbf{x}}^C_{k-1|k-1}} = \begin{bmatrix} \mathtt{I} & 0 & 0 & 0 \\ 0 & \frac{\partial \hat{\mathbf{v}}^C_{k|k-1}}{\partial \hat{\mathbf{v}}^C_{k-1|k-1}} & 0 & 0 \\ 0 & 0 & \frac{\partial \hat{\omega}^C_{k|k-1}}{\partial \hat{\omega}^C_{k-1|k-1}} & 0 \\ 0 & 0 & 0 & \mathtt{I} \\ 0 & \frac{\partial \hat{\mathbf{M}}^C_{k|k-1}}{\partial \hat{\mathbf{v}}^C_{k-1|k-1}} & \frac{\partial \hat{\mathbf{M}}^C_{k|k-1}}{\partial \hat{\omega}^C_{k-1|k-1}} & 0 \end{bmatrix} \tag{3.13}$$

and

$$\mathtt{G}_k = \frac{\partial \hat{\mathbf{x}}^C_{k|k-1}}{\partial \mathbf{n}} = \begin{bmatrix} 0 \\ \frac{\partial \hat{\mathbf{v}}^C_{k|k-1}}{\partial \mathbf{n}} \\ \frac{\partial \hat{\omega}^C_{k|k-1}}{\partial \mathbf{n}} \\ 0 \\ \frac{\partial \hat{\mathbf{M}}^C_{k|k-1}}{\partial \mathbf{n}} \end{bmatrix} \tag{3.14}$$

When using the constant velocity motion model, the incremental motion estimate will be correlated with the rest of the state after the prediction stage. This is in contrast to the case of a robot with odometry measurements presented in [11] (Equation 3.7). The correlations appear in our case because the uncertain velocity estimates in the state vector are used to predict the incremental motion. This was misreported in [19].

3.2.3 Update Step

The update step in Robocentric Mapping is the same as that of the ordinary EKF. The difference between measured observations, $\mathbf{z}_k$, of landmarks and their

predicted measurement, $\mathbf{z}_{k|k-1}$, according to the observation model gives the innovation, $\boldsymbol{\nu}_k$.

$$\boldsymbol{\nu}_k = \mathbf{z}_k - \mathbf{z}_{k|k-1} \tag{3.15}$$

This is used to update the state and covariance as follows.

$$\hat{\mathbf{x}}^C_{k|k} = \hat{\mathbf{x}}^C_{k|k-1} + \mathtt{W}_k \boldsymbol{\nu}_k \tag{3.16}$$

and

$$\mathtt{P}^C_{k|k} = \mathtt{P}^C_{k|k-1} - \mathtt{W}_K \mathtt{S}_k \mathtt{W}_k^\top \tag{3.17}$$

where

$$\mathtt{W}_k = \mathtt{P}^C_{k|k-1} \mathtt{H}_k^\top \mathtt{S}_k^{-1} \tag{3.18}$$

and

$$\mathtt{S}_k = \mathtt{H}_k \mathtt{P}^C_{k|k-1} \mathtt{H}_k^\top + \mathtt{R}_k \tag{3.19}$$

where R_K is the measurement uncertainty and H_k is the Jacobian calculated by linearising the measurement model for each observation, i, about the predicted state as follows

$$\mathtt{H}_{i,k} = \frac{\partial \mathbf{z}_{i,k|k-1}}{\partial \hat{\mathbf{x}}^C_{k|k-1}} = \begin{bmatrix} \mathtt{0} & \mathtt{0} & \mathtt{0} & \mathtt{0} \dots \mathtt{H}_{L_{i,k}} \dots \mathtt{0} & \mathtt{H}_{M_{i,k}} \end{bmatrix} \tag{3.20}$$

where

$$\mathtt{H}_{L_{i,k}} = \frac{\partial \mathbf{z}_{i,k|k-1}}{\partial \hat{\mathbf{L}}^C_{i,k|k-1}} \tag{3.21}$$

and

$$\mathtt{H}_{M_{i,k}} = \frac{\partial \mathbf{z}_{i,k|k-1}}{\partial \hat{\mathbf{M}}^{C}_{k|k-1}} \tag{3.22}$$

After this update is completed, the system can initialise new landmarks if there are currently too few visible. The system selects new landmarks from the image and adds them to the map. It is important to do this before the composition step so that these new landmarks become properly correlated with the rest of the map during the composition step.

3.2.4 Composition Step

The final stage of robocentric mapping is to transform the entire stochastic map so that the new camera pose estimate is centred at the origin. This is done using the (now refined) incremental motion estimate. The incremental motion and its uncertainty are effectively transferred to the landmark estimates as the motion is marginalised out of the state.

The estimate for each part of the state is calculated through composition with the refined motion

$$\hat{\mathbf{x}}^{C*}_{k|k} = \begin{bmatrix} \ominus \hat{\mathbf{M}}^{C}_{k|k} \oplus \hat{\mathbf{W}}^{C}_{k|k} \\ \ominus \hat{\mathbf{M}}^{C}_{k|k} \oplus \hat{\mathbf{v}}^{C}_{k|k} \\ \ominus \hat{\mathbf{M}}^{C}_{k|k} \oplus \hat{\omega}^{C}_{k|k} \\ \ominus \hat{\mathbf{M}}^{C}_{k|k} \oplus \hat{\mathbf{L}}^{C}_{k|k} \end{bmatrix} \tag{3.23}$$

and the covariance is then transformed using the Jacobian of this transforma-

tion, $\mathtt{J}_{C \to C^*}$.

$$\mathtt{P}^{C^*}_{k|k} = \mathtt{J}_{C \to C^*} \mathtt{P}^{C}_{k|k} \mathtt{J}^{\top}_{C \to C^*} \tag{3.24}$$

Continually keeping the map representation relative to the current camera pose in this way leads to a more consistent estimate. However, it does come with a cost. As well as the $O(n^2)$ update step present in the worldcentric EKF algorithm, the robocentric framework has a second $O(n^2)$ step, the final composition.

In the next section, a new observation method is presented which provides extra measurements to improve the estimate of the inter-frame motion at each timestep leading to a more accurate map and trajectory estimation. These observations do not require extra entries in the state vector however so they avoid increasing the $O(n^2)$ cost in the composition step.

3.3 Visual Odometry

In order to estimate the motion of the camera more accurately, as many landmarks as possible should be tracked from frame to frame. However, the update time of the EKF algorithm scales quadratically with the number of entries in the state vector. Instead, we will keep the sparse map used by Davison with typically $10-20$ landmarks visible in each frame but we will observe the inter-frame motion of the remaining corner features. These extra observations will greatly improve the estimate of the inter-frame motion before the final robocentric composition step.

3.3.1 Observation Step

As each new frame arrives, we detect corner points in the image using the FAST corner detector [66]. An 11×11 pixel patch around each corner is stored. Then, in the subsequent frame, the image location for each feature is found using normalised sum-of-squared-difference correlation followed by subpixel refinement [5].

The set of all these correspondences defines the epipolar geometry between the two frames. However, some of the matches may be incorrect. Joint compatibility is infeasible for outlier detection on such a large number of observations since it scales exponentially with the number of observations. The randomised joint compatibility algorithm of Paz *et al.* [58] could possibly handle this large observation set by performing joint compatibility on smaller samples in a RANSACstyle approach. Instead, we apply a simpler two view geometry RANSAC algorithm to detect any outliers. An essential matrix is calculated using the RANSAC method outlined in [76]. Using this essential matrix we can threshold to detect outliers and remove them from the set of visual odometry measurements. At least five visual odometry measurements must be made to allow an essential matrix to be calculated so that outliers can be detected. The essential matrix calculated here is used only for outlier rejection and plays no further role in the inter-frame motion estimation.

An example of these visual odometry features being tracked alongside the map landmarks is shown in Figure 3.2. The black lines indicate the image motion for each visual odometry feature since the previous frame. Observations rejected as outliers are indicated in magenta.

Figure 3.2: **Monocular SLAM with Visual Odometry:** While tracking the ordinary map landmarks, up to 200 visual odometry features can be used to refine the estimate of the inter-frame motion. The black lines here show the visual odometry features motion from position in the previous frame (•). Two observations have been rejected as outliers and are shown in magenta (one of these is easy to find in the centre of the image). The observed map landmarks are shown in red if the measurement was successful, blue when no match was found, and magenta if the observation was rejected by JCBB. Ellipses indicate the active search region. Landmarks not selected for observation in this frame are yellow. The 3D state estimate at this timestep can be seen in Figure 3.4.

3.3.2 Update Step

The prediction step of the robocentric EKF algorithm (Equation 3.10) provides a prediction for the inter-frame motion of the camera using the estimated velocity and the motion model. This inter-frame motion prediction defines a prediction for the epipolar geometry between the frames. Using the translation and rotation prediction, the predicted Essential matrix, $\mathtt{E}_{k|k-1}$, describing this geometry can be calculated.

$$\mathtt{E}_{k|k-1} = \left[\hat{\mathbf{r}}^C_{k|k-1}\right]_\times R\left(\hat{\mathbf{q}}^C_{k|k-1}\right) \tag{3.25}$$

Here $\left[\hat{\mathbf{r}}^C_{k|k-1}\right]_\times$ represents the skew symmetric matrix formed from the translation estimate and $R\left(\hat{\mathbf{q}}^C_{k|k-1}\right)$ represents the rotation matrix formed from the estimated quaternion.

For each visual odometry feature that was initialised at timestep $k-1$ and matched in timestep k, the ideal homogeneous image coordinates $\mathbf{p}_k$ and $\mathbf{p}_{k-1}$ are found using the camera calibration parameters. Then, the predicted epipolar line, $\mathbf{l}_i$, for the i^{th} feature is calculated.

$$\mathbf{l}_{i,k|k-1} = \mathtt{E}_{k|k-1}\mathbf{p}_{i,k-1} \tag{3.26}$$

If the predicted inter-frame camera motion is correct, and the observed image locations of the visual odometry feature is accurate then $\mathbf{p}_{i,k}$ will lie on the epipolar line, $\mathbf{l}_i$. In practice though, there is noise in the observation of the point corre-

spondence and the inter-frame incremental motion estimate is unlikely to be precisely correct. In fact, it is the predicted inter-frame incremental motion which we are trying to refine. The perpendicular distance from the predicted epipolar to the measured image location in ideal homogeneous coordinates is used as the measured observation, $\mathbf{z}_{i,k}$, for the EKF update step.

$$\mathbf{z}_{i,k} = \mathbf{p}_{i,k}^{\top}\mathbf{l}_{i,k|k-1} \tag{3.27}$$

The predicted observation, $\mathbf{z}_{i,k|k-1}$, is always zero (the point lying directly on the predicted epipolar line defined by the predicted camera motion).

Before $\mathbf{z}_{i,k}$ is calculated, the homogeneous point, $\mathbf{p}_{i,k}$, and the predicted homogenous line, $\mathbf{l}_{i,k|k-1}$, should be rescaled to provide an accurate measurement of the perpendicular distance. A good scaling for a homogeneous point, $(u, v, w)^{\top}$, is

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \rightarrow \begin{pmatrix} \frac{u}{w} \\ \frac{v}{w} \\ 1 \end{pmatrix} \tag{3.28}$$

and for a homogeneous line, $(a, b, c)^{\top}$,

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \rightarrow \begin{pmatrix} \frac{a}{\sqrt{a^2+b^2}} \\ \frac{b}{\sqrt{a^2+b^2}} \\ \frac{c}{\sqrt{a^2+b^2}} \end{pmatrix}. \tag{3.29}$$

The measurement noise, $\mathtt{R}_{VO}$, for visual odometry observations is estimated using

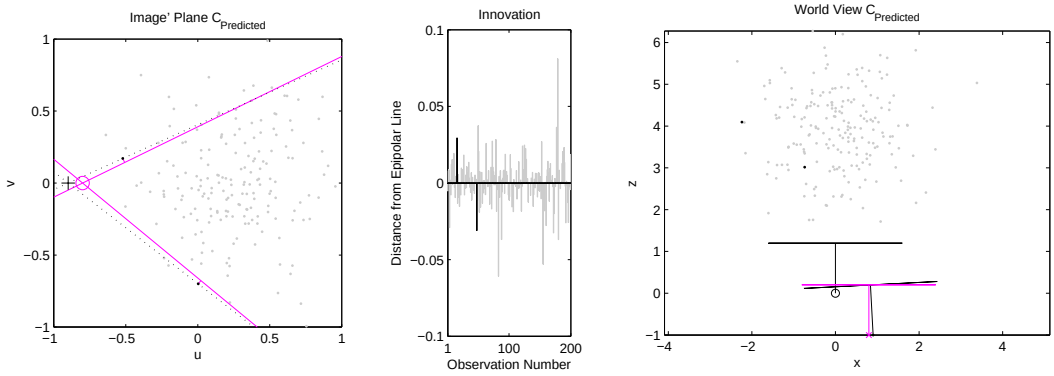
the 1 pixel image noise standard deviation assumed for normal map landmark observations, $\mathtt{R}_{image}$.

$$\mathtt{R}_{VO} = \mathbf{l}^\top J_{homog} J_{ideal} \mathtt{R}_{image} J_{ideal}^\top J_{homog}^\top \mathbf{l} \tag{3.30}$$

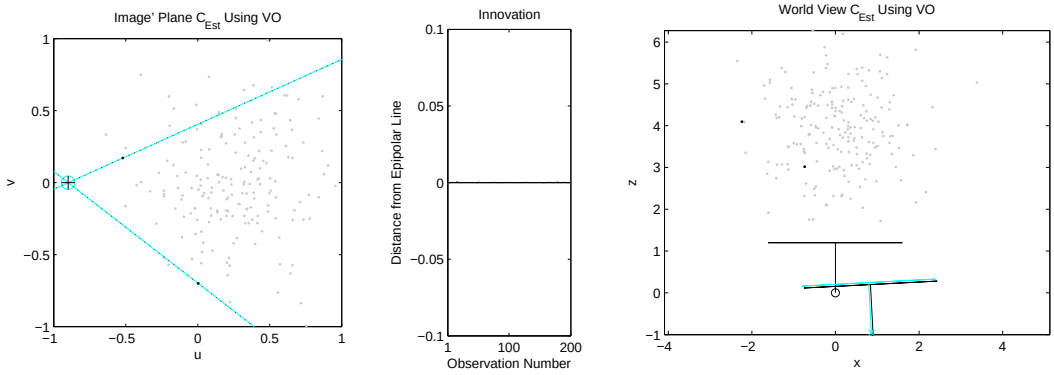
where J_{homog} and J_{ideal} are the Jacobians for the conversion into normalised homogeneous coordinates and the ideal camera frame respectively. Uncertainty exists only for the measurement of the patch in the second frame since the position where the patch was extracted in the first frame is known perfectly. This measurement noise can also be inflated to give higher weight to the observation of map landmarks if desired.

The visual odometry update is illustrated in Figure 3.3 using a simple simulation for clarity. The camera begins at the origin looking down the z-axis at 200 points in the world. The camera is then moved backwards and to the right while rotating about the y-axis. The view of the points from this new camera position is shown in the left column. The epipolar geometry is indicated by the epipole and the epipolar lines for two selected points. Unfortunately, the predicted camera motion is unlikely to precisely match the true motion. Here, the predicted motion was slightly incorrect in translation and had no rotation. This leads to an incorrect epipolar geometry prediction. The true point positions are not directly on the predicted epipolar lines. The distance from the epipolar line for each point is indicated by the bar graph in the middle column.

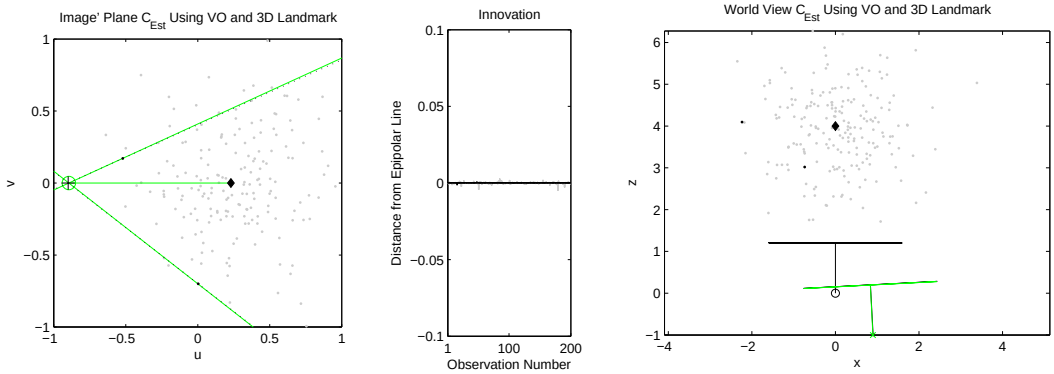
Using the distance from the predicted epipolar line as an observation, the motion prediction can be updated using an EKF. In Figure 3.3b, the resultant estimate



(a) **Before EKF Update:** The predicted motion is incorrect in both translation and rotation.



(b) **Updated Using Visual Odometry:** The estimated pose is corrected up to a projective ambiguity. With planar motion, the orientation and the direction of the translation are determined but the scale of the translation is not.



(c) **Updated Using Landmark and Visual Odometry:** The single measurement of a 3D landmark ⧫ removes the projective ambiguity allowing the pose determined.

Figure 3.3: **EKF Update Process With Visual Odometry:** *Left Column:* The image plane after motion showing the position of 200 visual odometry features (•), the true (◯) and estimated (+) epipole, and the true (···) and estimated (—) epipolar lines for two selected features (•). *Middle Column:* Distance to the epipolar line for each of the visual odometry features given the estimated camera motion. *Right Column:* The 3D pose of camera (⊤) relative to the features (•). The camera starts at the origin and then translates and rotates about the y-axis to the true pose shown in black. The estimate for this pose (×) is in colour.

is shown. The motion estimate correctly matches the true epipolar geometry as can be seen in the image view (left column). The distance from the epipolar line for each of the visual odometry features is greatly reduced (middle column). By iterating the EKF and increasing the prior uncertainty, the estimate would converge to the true epipolar geometry. However, in the world view (right column), it is clear that the estimated camera motion does not precisely match the ground truth. With only observations of visual odometry features, the estimate converges to a relative pose which has the correct epipolar geometry but the wrong 3D position. In fact, the estimated motion is correct up to scale. The orientation is correct and the translation is in the direction of the true translation, but the scale of that translation is wrong.

Figure 3.3c shows how a single 3D landmark observation can provide the scale information needed to get an estimate with both the correct epipolar geometry and 3D pose. Since the 3D position of this landmark is known, this provides the constraint on the scale of the translation missing when only visual odometry features are observed. By iterating the EKF and increasing the prior uncertainty, this estimate would converge to the true camera motion. It is important to note that the observation of this single landmark without visual odometry would be unable to achieve this without the observation of other landmarks.

By combining the two observation types, the system gains the benefit of high accuracy inter-frame motion estimates without adding new landmarks to the state vector which would greatly increase the update time. In the next section, results are presented showing the performance of our monocular SLAM system with visual odometry and the robocentric framework.

3.4 Results

To test the performance of the robocentric monocular SLAM system with visual odometry we have run experiments on both real and simulated data. First, we test the system on a simple sequence of the camera moving in a small environment. After demonstrating that the system can build maps of simple scenes, we test the accuracy of the system by evaluating the performance in simulation. Simulations provide a good test since perfect ground truth is known, and enable us to verify the improved accuracy and consistency claims we make. However, a simulation cannot perfectly replicate realistic operating conditions so a further test of the estimation accuracy is performed using an aerial photo to provide ground truth of the camera position.

3.4.1 Monocular SLAM in a Simple Scene

The monocular SLAM system was used in a bedroom to test the performance of the system in a small, simple environment. Figure 3.2 on page 60 shows a frame from this sequence with the visible map landmarks indicated. The full state estimate at the timestep corresponding to this frame is represented in 3D in Figure 3.4. This shows the position of each landmark relative to the camera along with the corresponding uncertainty in this estimate. The scale on the axes is arbitrary. If a known scale is required, the system can be initialised with a set of known landmarks as Davison did. This can be achieved by placing a known target in front of the camera at the start of a sequence.

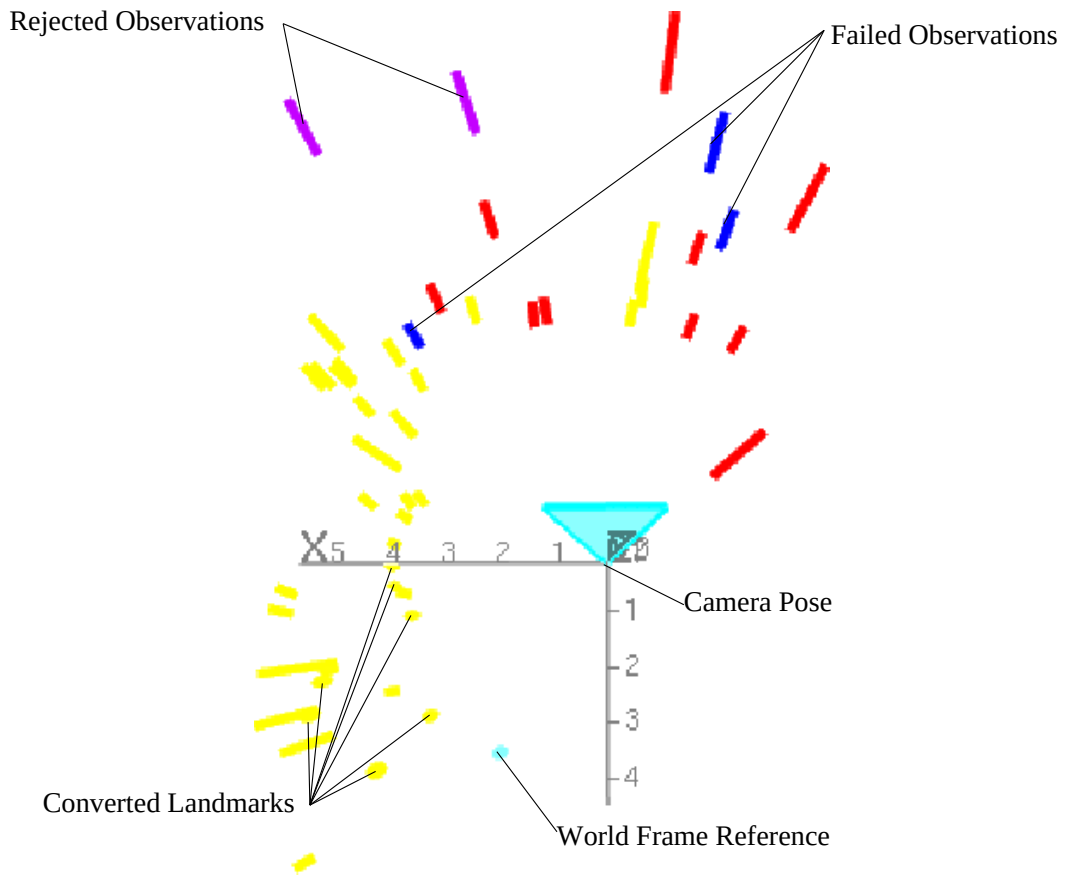
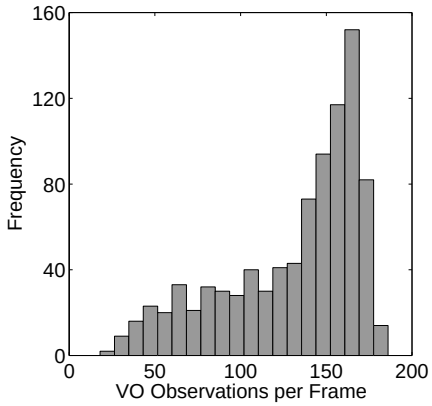
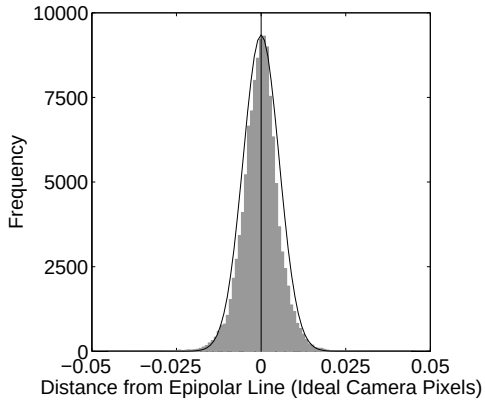


Figure 3.4: **3D State Estimate:** The system tracked the pose of the camera and built a map of landmarks as the camera moved around a small bedroom. A top down view of the 3D state estimate is shown here. The camera view showing the observations for this timestep is shown in Figure 3.2. Along with the visual odometry features, fifteen landmarks were selected for observation in this frame. Three failed to find a match (blue) within the active search ellipse. The matches found for two of the landmarks were rejected by JCBB (magenta). The remaining ten selected landmarks were successfully observed (red). Landmarks not selected for observation in this frame are shown in yellow. Some of the landmarks have already been converted from the inverse depth parameterisation to the 3D representation and are shown as an uncertainty ellipsoid rather than a line. The line indicates the three standard deviation uncertainty in depth along the estimated ray from which the landmark was first observed. The pose of the base reference frame (cyan) is also estimated to allow conversion back to the world coordinate frame if needed.



(a) The number of VO observations varies depending on appearance of each frame. A 200 feature maximum is imposed by the user.



(b) The distribution in the innovation for each VO observation conforms to the zero-mean Gaussian assumption in the measurement model.

Figure 3.5: **Analysis of Visual Odometry:** The bedroom sequence was used to analyse the behaviour of the visual odometry (VO) part of the monocular SLAM system.

Since the robocentric framework is used, the camera pose is centred at the origin with zero uncertainty. If the worldcentric representation is required then the full state can be converted using the base reference frame estimate. At the beginning of the sequence, the world reference frame corresponding to the initial pose of the camera is placed in the state vector as an unobservable entry. The position and uncertainty of this coordinate frame is shown in Figure 3.4 as a cyan ellipsoid.

The uncertainty in the position of the landmarks is indicated either by an ellipsoid or, if the landmark is still parameterised in inverse depth, as a ray whose endpoints indicate the three standard deviation depth uncertainty. These landmarks have been colour coded to indicate the result of attempted observations in this frame. The same coding is used in Figure 3.2 which also shows the active search region used to make observations of these landmarks.

As well as 3D landmarks, the system also observes visual odometry features at each frame. These are indicated in Figure 3.2 by lines showing the motion since the previous frame. A maximum of two hundred visual odometry observations are made at each frame but in some frames, fewer suitable interest points exist. Figure 3.5a shows the distribution in the number of visual odometry observations made per frame in this sequence. This is highly dependent on the visual appearance of the scene however. More features are available in scenes with richer texture and images with less motion blur. Also, some features at each timestep are not used if a match is not found in the second frame or the match is rejected as an outlier.

This sequence was also used to analyse the visual odometry observation model. By using the EKF framework to incorporate these observations, the system assumes that the innovation for these observations follows a zero-mean Gaussian distribution. The distribution for innovation for all the visual odometry observations in the sequence is shown in Figure 3.5b. A zero-mean Gaussian distribution has been fit to the data showing that the assumption used in our model is valid.

The performance of the system for this sequence was very good. It was able to track the camera pose as it moved through the scene by building a sparse map of point landmarks. Simple sequences like this are ideally suited to monocular SLAM since the environment is small and contains sufficient texture for landmark selection. The camera does not move far and periodically returns to observe all the landmarks in the map, improving the map estimate. The motion is smooth and the camera is never occluded so landmark observations are always available.

However, for a monocular SLAM system to be truly useful, it must be able to work in more challenging circumstances. In later chapters, we will demonstrate our system's robustness to difficult tracking conditions, and its ability to map much larger environments. First though, the quality of the estimation is analysed using a simulation and then in a real-world sequence.

3.4.2 Estimation Quality in Simulation

A simulator was developed to test our monocular SLAM system. Simulations provide a way to investigate the accuracy of the trajectory and map estimate since ground truth is known perfectly. Monte Carlo runs can also be performed to examine statistically the quality of the estimated mean and uncertainty produced using different SLAM algorithms. This allows us to compare the worldcentric and the robocentric framework as well as examine the effect of including visual odometry measurements alongside landmark observations.

The simulation consists of a 100×20 metre courtyard that the camera moves around while facing the wall. The top down view of the map of landmarks and the camera trajectory is shown in Figure 3.6 where the camera moves in the clockwise direction starting at the origin. Along with this general motion around the courtyard, the camera also rocks back and forth about the optical axis and follows a periodic motion up and down in the y direction. These accelerations provide a more challenging motion for the monocular system to track. All accelerations were within the noise estimates of the motion model though the overall trajectory does not conform to the constant velocity motion model used by the SLAM

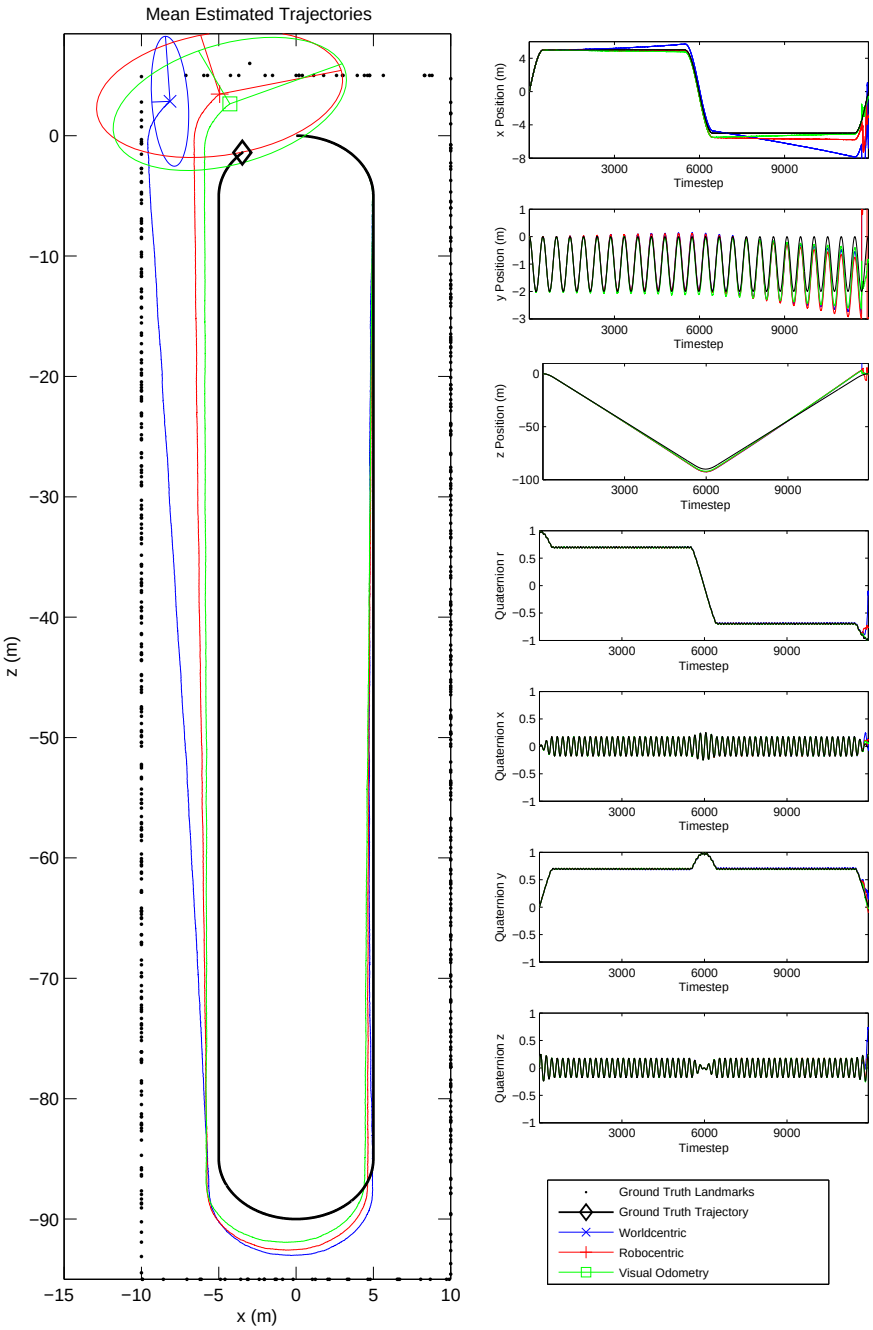


Figure 3.6: **Simulation – Mean Estimated Trajectory:** The trajectory used in the simulation is shown component wise (right column) and in a top down view with the landmarks (right column). Along with the ground truth, the mean estimate of 20 Monte Carlo runs for each algorithm is shown. The top down view also shows the three standard deviation uncertainty ellipse for the mean estimate for each algorithm just before the loop is closed. Using the robocentric approach produces more consistent estimates with the true pose closer to the 3 sigma uncertainty ellipse. It also improves the mean through the reduction of linearization errors. The use of visual odometry observations further improves the mean of the estimate. Similar improvements are found in the landmark position estimates. The error in the estimate and uncertainty is further explored in Figure 3.7.

system. The value for each component of the camera pose over the trajectory is shown in the right column of Figure 3.6.

Twenty Monte Carlo runs were performed using this simulated trajectory. For each run, the monocular SLAM system automatically selected, initialised, and observed landmarks from the simulated environment. Observations of these landmarks were perturbed with random Gaussian noise with a standard deviation of 0.25 pixels. However, when testing each of the three algorithms on a particular run, the same noisy observations of landmarks were used. Correct data association for each observation was given to the SLAM system.

The simulation begins with the camera at the origin with a correct estimate of the initial linear and angular velocities in the state vector. The initial map also contains four known landmarks to fix the scale of the map created. One of these four lies off the plane to ensure that the camera pose estimate does not suffer from the ambiguity caused by observing only landmarks lying on a single plane. After the estimate has converged to the correct motion, this ambiguity is no longer a problem.

For visual odometry observations, the simulator randomly selects 200 features in the image plane to track the motion between each timestep. The depth of these features is initialised to be on the wall of the courtyard plus a random offset of up to 2 metres. This offset is used to avoid all of the features lying on a single plane.

The results of these simulated runs can be seen in Figure 3.6. Each of the three

monocular SLAM techniques is able to track the true pose of the camera throughout the sequence with different degrees of accuracy. This figure shows the mean estimate for each of the techniques alongside the ground truth for the trajectory. The largest part of the error in the estimate for all three techniques is due to scale drift. This can be most readily seen in the z-component of the trajectory. The scale of the world begins to grow as the camera gets further from the initial known features. This is due to the differences between the assumed motion given by the constant velocity motion model and the true trajectory of the camera. Scale drift is also seen when working with real world data. The scale error is corrected when the camera comes around the loop and reobserves the initial features again, 'closing the loop'. In a simulation with known data association, this is straightforward, in practice however this poses challenges. Loop closure detection will be discussed in Chapter 5.

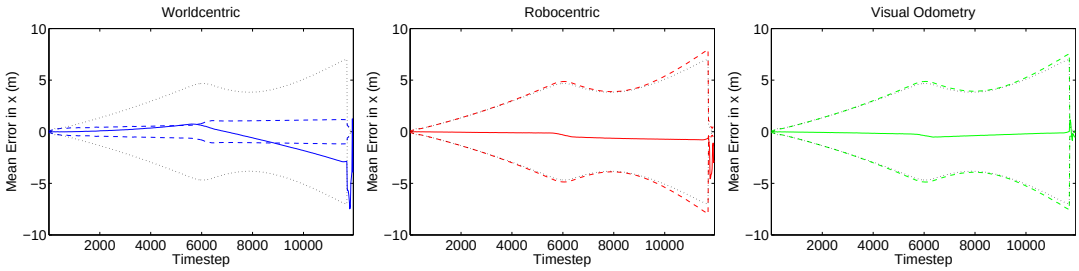
With measurement noise and an imperfect motion model, errors in the estimate are inevitable. However, a good estimation algorithm should keep errors to a minimum and correctly estimate its uncertainty in the answer given. As can be seen in Figure 3.6 the robocentric framework provides both a better estimate and a more realistic estimate of the uncertainty. In Figure 3.7, two components of the camera pose estimate are examined in more detail. The mean error and the mean uncertainty at three standard deviations are shown for the x position error and a component of the error in orientation. The most straightforward orientation to examine for this trajectory is the rotation about the y axis since the optical axis of the camera stays perpendicular to the camera motion in the x-z plane shown in Figure 3.6. The quaternion in the camera pose estimate was converted

to suitable Euler angles and the corresponding angular error was plotted with its uncertainty.

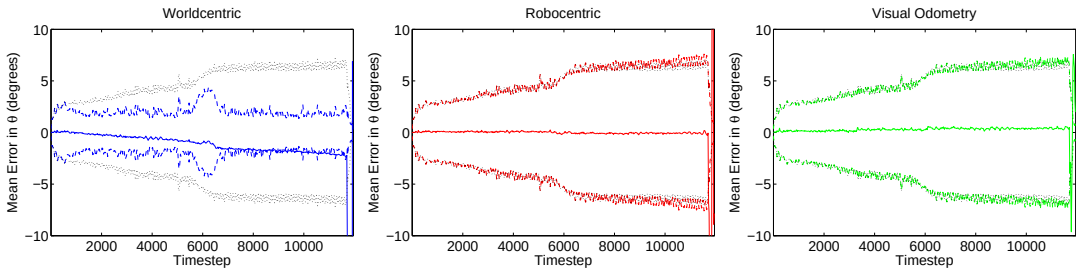
The ideal uncertainty for each simulation run can also be determined. By running the simulation with the same observations but with zero measurement noise, the estimated uncertainty calculated by the EKF more closely reflects the true uncertainty. This is possible because a noise-free simulation produces an estimate with no linearization errors. The uncertainty estimate produced by a good SLAM algorithm should match this ideal uncertainty. In Figure 3.7, the ideal uncertainty ($\cdots$) is shown alongside the uncertainty estimated for each algorithm (- - -). As expected, the ideal uncertainty tends to grow as the camera moves further from the initial position. The uncertainty calculated using the robocentric framework closely follows this growth in uncertainty with a small overestimation. In contrast, the uncertainty estimate produced using worldcentric mapping stops growing very early in the test and from then on greatly underestimates the uncertainty in the pose estimate.

The underestimation of the uncertainty when using the worldcentric approach is due to linearization errors. These errors become significant when the orientation uncertainty grows above 2 degrees. Once this occurs, the uncertainty estimate becomes corrupted by linearization errors and is lower that the ideal uncertainty. This result was also found by Bailey *et al.* [3]. The robocentric approach is able to maintain a better estimate of the uncertainty because in the reference frame of the camera, the angular uncertainty of observed landmarks is much lower.

An incorrect uncertainty estimate demonstrates the inconsistency of the world-



(a) **Position Error (x Coordinate):** The estimate calculated using the worldcentric framework diverges from the ground truth and becomes inconsistent due to linearization errors. The estimates produced in the robocentric framework stays close to the ground truth and the uncertainty estimate more closely matches the ideal uncertainty.



(b) **Angular Error:** θ represents the rotation about the y-axis in Figure 3.6. Once the true angular uncertainty grows beyond a couple degrees, linearzisation errors cause the uncertainty to be underestimated when the worldcentric framework is used.

Figure 3.7: **Estimation Accuracy:** The mean difference (—) between the estimated value and the ground truth for two components of the camera pose was found for 20 Monte Carlo runs. The mean three standard deviation uncertainty (- - -) for the estimate is also shown relative to the ideal uncertainty ($\cdots$) determined using noise-free observations. Without using the robocentric framework, the uncertainty tends to be underestimated once the angular uncertainty grows beyond a couple degrees. Estimates made using the robocentric framework have uncertainty closer to the ideal because linearization errors are reduced.

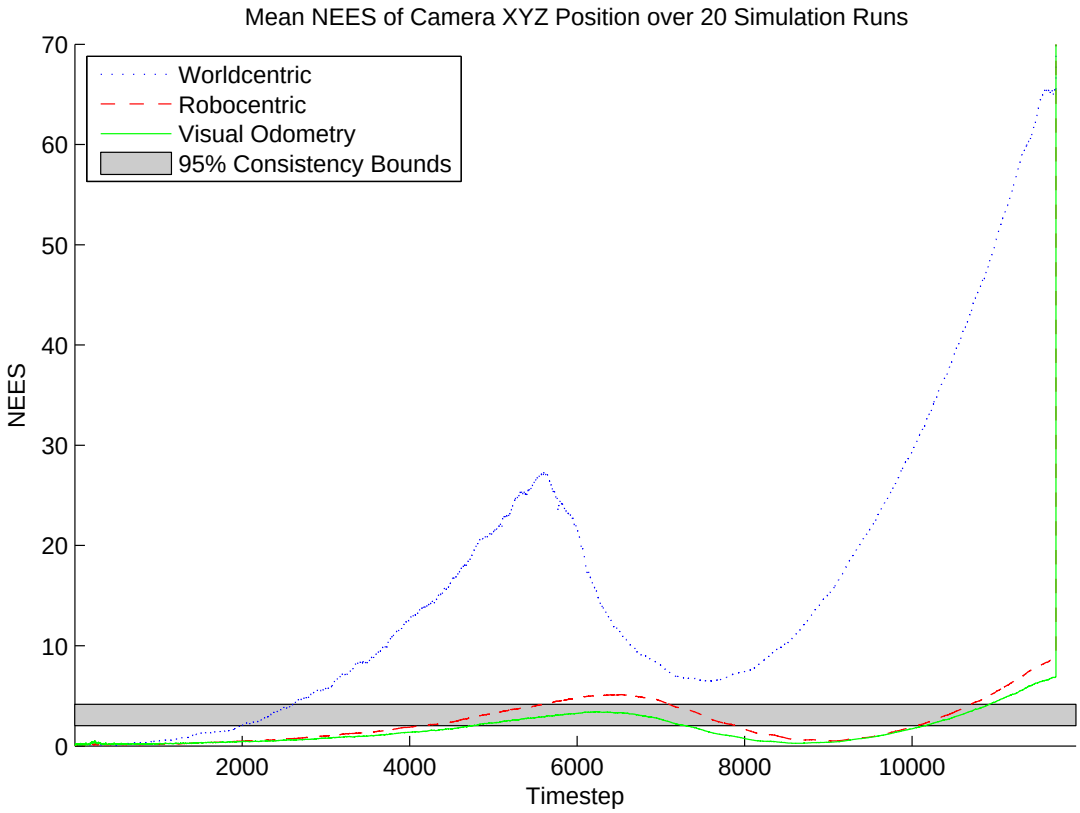


Figure 3.8: **Consistency Check:** The consistency of the three algorithms is tested by examining the mean normalised estimated error squared (NEES) over twenty Monte Carlo runs in simulation. To remain consistent the NEES should remain within the 95% consistency bounds shown here in grey most of the time. A higher NEES indicates an overconfident estimate and a lower NEES indicates a conservative estimate. The worldcentric estimate soon becomes very overconfident. Using the robocentric estimation keeps the estimate more consistent. The spike near the end of the run for all three algorithms is due to the original landmarks coming back into view during the loop closure event.

centric EKF. An estimator is said to be consistent if its state estimation error is unbiased and the actual Mean Square Error matches the calculated covariances.

The consistency of an estimation algorithm can be investigated by examining the normalised estimation error squared (NEES), ϵ.

$$\epsilon = (\mathbf{x} - \hat{\mathbf{x}})^{\top} (\mathtt{P})^{-1} (\mathbf{x} - \hat{\mathbf{x}}) \tag{3.31}$$

If the filter is consistent and linear-Gaussian, ϵ is χ^2 distributed with dimension equal to the size of $\mathbf{x}$. However, multiple Monte Carlo runs must be performed and the average NEES calculated because the errors in a single run are correlated.

If the filter is consistent it will pass a χ^2 acceptance test. We have performed 20 runs and have calculated the NEES using the camera position estimate. This corresponds to a 95% acceptance region between 2.02 and 4.16. If the average NEES is below the lower bound the estimation is conservative. If it is higher than the upper bound then it is optimistic. More detail on this standard consistency check can be found in [3] and [6].

The results of this test for the three algorithms are shown in Figure 3.8. By working in the robocentric framework, the estimate is more consistent than when using the worldcentric framework.

The simulation also provides a way to test the benefit of visual odometry. The simulation was rerun with different numbers of visual odometry observations but identical noisy landmark observations. Figure 3.9 shows that the estimation error

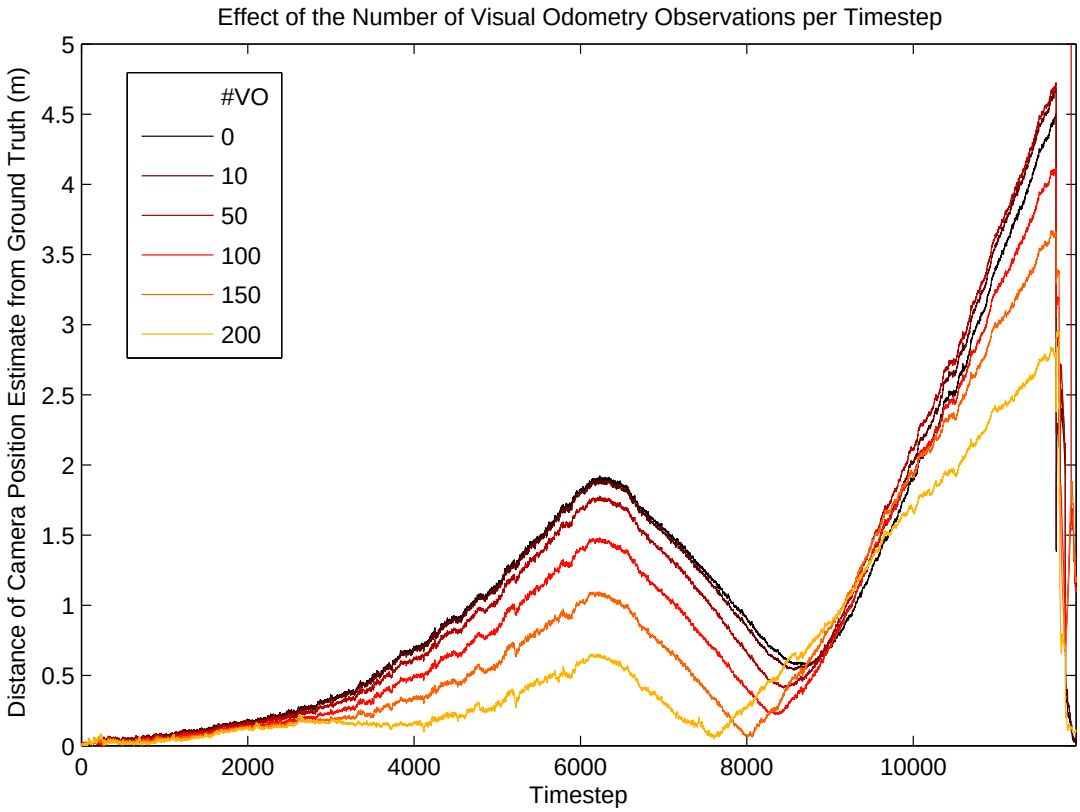


Figure 3.9: **Number of Visual Odometry Observations:** In this experiment, the number of visual odometry features observed per frame was changed to see the improvement in the estimated camera trajectory compared to the simulation ground truth. Identical noisy landmark observations were used in each run.

generally decreases as the number of visual odometry observations per timestep is increased. In all other results in this section, 200 visual odometry observations are performed per timestep.

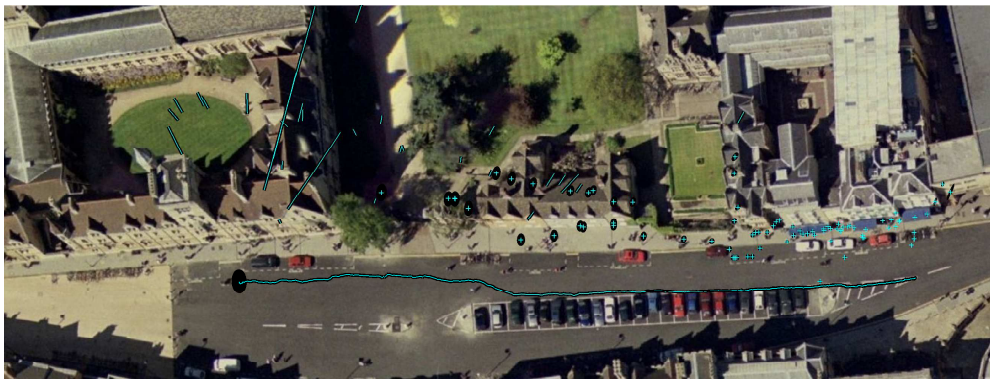
3.4.3 Estimation Quality in the Real World

To test the accuracy of our system on real world data, a sequence was recorded on Broad Street in Oxford using a trajectory which can be aligned to an aerial photo.

The hand-held camera was pointed at a row of buildings while the experimenter walked down the white line painted in the road. This sequence was then used to test the benefit of the robocentric framework and visual odometry measurements. The same landmark observations at each frame were used in each test. The results are shown in Figure 3.10. Alignment was performed manually using the trajectory and building facades visible near the start of the trajectory (on the right). This makes any scale drift during the sequence more apparent.

When the traditional worldcentric framework is used, the scale increases over the sequence and the trajectory and map begin to curve towards the top of the image. With the same observations, the robocentric approach gives a very similar final estimate, but with larger estimated uncertainty. However, when visual odometry measurements are used alongside landmark observations, the motion estimate is far more accurate which in turn leads to a more accurate map estimate. The trajectory can be aligned to match the true trajectory shown by the white line above the parked cars in the photo. We hypothesize that the main benefit here accrues in a few key frames in which only a few map features were observed, and which poorly constrained the motion in the absence of additional visual odometry features.

In another experiment, the accuracy was tested by moving the camera around a loop and then returning to exactly the same position. If the estimated trajectory is correct, the final camera pose estimate will have the same position as the camera position at the start of the sequence. Though, the camera only travelled a relatively small distance in this sequence (3 metres) compared to the outdoor sequence (85 metres), it stayed much closer to the landmarks making this trajec-



(a) Worldcentric

(b) Robocentric

(c) Robocentric with Visual Odometry

Figure 3.10: **Broad Street:** The camera was moved along the white line just above the parked cars in this aerial photo while facing the buildings. The trajectory estimated using both visual odometry and landmark observations closely matches the true trajectory when aligned with this aerial photo. Without these extra constraints on the motion, the estimate accuracy is worse.

tory *effectively* twice as long. This effective distance is determined by noting how many sets of covisible landmarks pass out of view as the camera moves along (9 vs. 5 for the outdoor). All three algorithms were tested using the same observations of landmarks at each frame. At the end of this trajectory, the error in the estimated position for the camera is 21cm for the worldcentric algorithm, 16cm for the robocentric, and 13cm for the robocentric with visual odometry. Though our approach produced a more accurate estimate, a separate loop closure detection system would still be needed as the uncertainty at this point is too large to reliably use active search to reobserve the initial landmarks when they come back into view. Our own loop closure detection system will be presented in Chapter 5.

3.4.4 Timing

We analyse the computation time required by our system using the Broad Street sequence. The scene viewed in this sequence is larger than what can be handled in real-time without using submapping techniques which will be discussed in Chapter 5. However, it does allow us to investigate the average time of some steps and the computational complexity of others.

During these experiments, the system observed up to 200 visual odometry features and 20 landmarks in each frame. New map landmarks were initialised when the number of visible landmarks dropped below this threshold. The set of successful observations in each frame was used by the system to update the state estimate and then the robocentric composition was performed to transform the map estimate to keep the camera pose at the origin.

Since the system attempts approximately the same number of observations per frame, many of the steps take a fairly constant amount of time. The time taken for these steps is given in Table 3.4.4. The remaining time per frame can then be used to perform the EKF update and the robocentric composition.

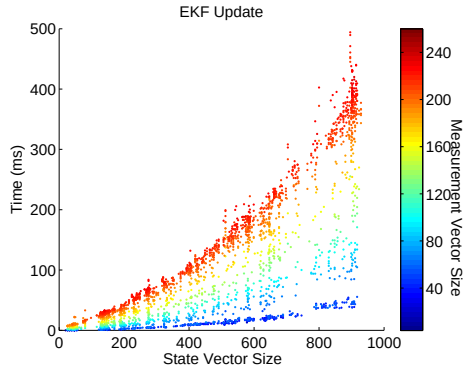
Corner Detection	2 ms
Visual Odometry - Measurement	3 ms
Visual Odometry - Outlier Rejection	3 ms
Landmark - Measurement	2.5 ms
Landmark - Outlier Rejection	0.7 ms
New Landmark Initialisation	1 ms

Table 3.1: **Observation Time:** The average time taken for observations and landmark management when 20 landmarks and 200 visual odometry features are observed per frame.

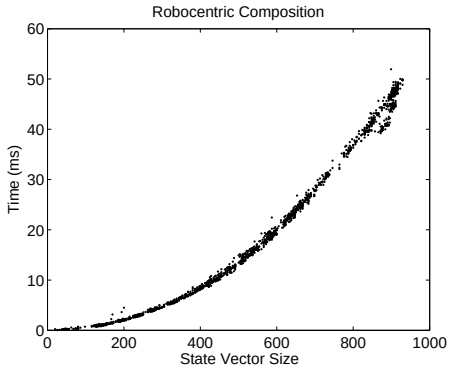
The computation time required for the EKF update at each timestep depends on both the number of successful observations, r, and the current size of the state, n. As the system explores and adds new landmarks to the map, the update time quickly begins to dominate all other steps. Figure 3.11a shows the time taken to perform the update over all of the frames for the Broad Street sequence. The computational complexity of the steps in the EKF update explains this growth in update time.

The EKF update is $O(r^3)$ due to the inversion of the innovation covariance matrix, $\mathtt{S}$, required in Equation 3.18. However, since this matrix is positive definite, this inversion can be performed using the Cholesky decomposition and actually adds only a small fraction to the update time (7ms for the largest number of measurements allowed in this sequence).

By far the most costly step in the update is the calculation of the new state



(a) **Update:** The time taken to perform the update depends on both the number of successful observations and the size of the state vector.



(b) **Composition:** The time taken to perform the robocentric composition increases quadratically with the size of the state vector.

Figure 3.11: **Computation Time:** The time taken to perform the EKF update (a) and the robocentric composition (b) varies over the timesteps in the Broad Street sequence.

covariance matrix, $\mathtt{P}^C_{k|k}$, shown in Equation 3.17. This step requires $r^2n + rn^2$ multiplications and $r^2n + rn^2 + n^2$ sums and so is $O(nr^2)$ or $O(rn^2)$ depending on which is more dominant. Each visual odometry observation increases r by 1, while each landmark increases n by 3 (or 6 if it has not been fully initialised) and increases r by two if it is observed. We impose an upper bound on the size of r by limiting the number of landmarks and visual odometry features which can be observed at each timestep. However, with the number of observations considered here (r can be up to 240), the update time quickly grows beyond framerate as the state size begins to increase.

With limited processing time available in a real-time system, the right balance between the number of visual odometry measurements and the number of landmark observations must be selected. While maintaining observations of a sparse map,

the remaining time available can be used to observe as many visual odometry features as possible. When the state size grows too large, submapping techniques can be used to allow the system to continue mapping in real-time by breaking the map into manageable pieces.

The time required for the robocentric composition is shown in Figure 3.11b. This step depends on the size of the state estimate and is $O(n^2)$. This composition step is the computational price paid for the gain in consistency in the robocentric approach. In these experiments, this step took an average of 10% of the total computation time.

3.5 Conclusion

We have presented a monocular SLAM system which provides a high quality estimate of the camera pose both in accuracy and consistency. The increase in accuracy is achieved through a novel method of including many more observations per frame without the need for increasing the size of the state estimated. As well as observing the map landmarks at each frame to prevent drift, our system also observes the inter-frame motion of every other corner feature in a visual odometry style method. These extra observations are used to constrain the estimate of the inter-frame motion of the camera leading to a less noisy pose estimate.

The consistency of the estimate is improved through the use of the robocentric mapping framework. We have adapted this technique for use with a hand-held camera and have shown that it provides more consistent estimates in monocular

SLAM than the traditional worldcentric EKF algorithm. The robocentric framework provides a natural method for handling the visual odometry observations since estimating the inter-frame camera motion at each frame is a key part of the robocentric approach.

The system described in this chapter shows good performance in tracking the pose of the camera under most conditions. However, tracking can still fail when the camera is occluded or the motion is too fast. The next chapter describes a relocalisation system which is able to detect and recover from these failures.

4

Relocalisation

The results in this chapter have been published in the proceedings of the International Conference on Robotics and Automation (ICRA), Rome, Italy, 2007 [90], the proceedings of the International Conference on Computer Vision (ICCV), Rio de Janeiro, Brazil, 2007 [88], and has been submitted to the journal Pattern Analysis and Machine Intelligence.

4.1 Introduction

In this chapter, we present our relocalisation module for the monocular SLAM system. When this module is combined with the SLAM system, it allows the

detection and recovery from tracking failures. Since these failures are inevitable in real world scenarios, the combined system has far greater robustness. This dramatically increases the usability of the system and pushes hand-held monocular SLAM closer to use outside of laboratory conditions.

4.1.1 Why is relocalisation important?

The monocular SLAM system described in Chapter 2 is able to build a map and estimate the pose of the camera provided it is able to make reliable observations of the environment. Unfortunately, when using a single hand-held camera as the only sensor, situations that prevent these observations are all too common. The three main causes of these failures are:

- camera occlusion
- motion blur
- unmodeled sudden motions.

When all of the attempted landmark observations fail, the system relies completely on the constant velocity motion model to predict the motion of the camera. This model is very poor at predicting camera motion for anything more than a couple of frames. For longer term predictions, the true motion is typically significantly different.

However, the motion model also tries to estimate the increase in camera pose uncertainty during these blind frames. Normally, this uncertainty is reduced by making observations of map landmarks but, in these situations, observations are not possible. The uncertainty growth indicates low confidence in the motion estimated during this sensor blackout. Ideally, when conditions improve and observations are again possible, the system can make measurements of map landmarks to correct the camera pose estimate and reduce the uncertainty.

In reality though, making correct observations when the camera pose is so uncertain is quite unlikely. The system uses only simple normalised cross correlation to identify and measure the position of map landmarks in the image. This strategy is normally adequate during normal tracking because active search is used to limit the search region. Using the estimated position and uncertainty of the camera relative to each map landmark, a 3σ search region in the image is calculated. This is done to decrease the time taken for the search but, importantly, it also lowers the probability of incorrect data association due to visually similar regions of the world. Active search is no longer useful after a sensor blackout. The uncertainty in the camera pose, and hence the size of the search region, quickly grows during the blackout. Making observations in the normal way with very large search regions is very unreliable and slow. When incorrect data association occurs, both camera pose and the map estimate become corrupted irreversibly causing the whole system to fail. This problem is highlighted in Figure 4.1.

Instead of relying on the system making correct observations after sensor blackout, a relocalisation module supervises the system to detect these situations and prevent map corruption. When the sensor information becomes reliable again this

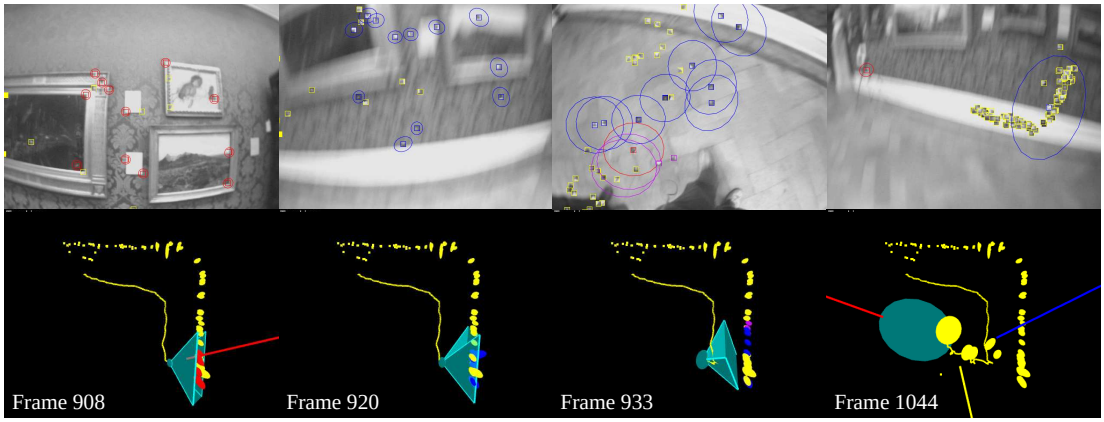


Figure 4.1: The camera view (top) and the map view (bottom) are shown for four different timesteps during a tracking failure. Frame 908 shows the system tracking reliably before the failure. A sudden motion then occurs as the camera is pointed at the floor. This causes all of the attempted observations to fail as shown in frame 920. With no observations, the camera pose uncertainty and hence the active search ellipses begin to grow. At frame 933, the system makes incorrect observations of two of the landmarks while using the large search ellipses. With no truly correct observations, the joint compatibility test is only able to rule out one of these false observations. Incorporating this false measurement of a landmark into the EKF update irreversibly corrupts both the map and the camera pose estimate. The map is further corrupted in frame 1044 as the system initialises new landmarks in the map using the incorrect camera pose estimate. Instead of trying to fine tune the underlying SLAM system to prevent map corruption, our approach is to have a separate relocalisation module supervise the SLAM system and recover from tracking failures.

module then relocalises the camera. This technique prevents the map estimate from becoming corrupted and allows the system to recover after tracking failures.

4.1.2 Related Work

Recovery from tracking failure is important not only for SLAM systems, and a number of different attempts at improving robustness have been presented. These range from very local methods which attempt to bridge tracking failure over a few frames to methods which attempt a global relocalisation.

Chekhlov *et al.* have built two systems which can bridge short tracking failures in a monocular SLAM context. This was first done by exploring multiple hypotheses with a particle filter [63] and later by using a rich feature descriptor to provide robust data association [14]. Both of these methods are local in that they rely on the point of recovery being close to the point of failure.

We desire a more global approach to relocalisation which does not require a prior estimate for the pose of the camera. This global relocalisation problem has been explored in robotics where it is called the kidnapped robot problem. The goal is to determine the location of the robot relative to the map after it has been placed within the mapped environment at an unknown location. Dellaert *et al.* [25] demonstrate global localisation using a particle filter based approach for a robot with sonar sensors. Neira *et al.* [53] also demonstrate localisation in 2D but their system uses a stochastic map built using SLAM. Though their system uses a laser scanner rather than a camera, the problem is quite similar to ours. They

determine the pose of the robot relative to a map of point landmarks using the observations from a single timestep. Like our system, they use a combination of geometric compatibility and random sampling. However, their method is simpler since the laser scanner provides range as well as bearing measurements and the robot and landmarks are all assumed to lie on a plane.

A method to relocalise in vision based systems is to find correspondences between the latest image and a set of previously seen keyframes. Reitmayr & Drummond [65] use edge based tracking to track a hand-held camera relative to a known model. When tracking fails, point feature correspondences are established between the latest image and a set of recent keyframes. From these matches, they solve for the relative rotational component of the pose (taking the translational component to be zero). Provided a keyframe exists in the vicinity, this usually gives a sufficiently accurate estimate of the pose to re-initiate tracking, which can then converge to the correct full pose. Klein & Murray [41] also relocalise their monocular SLAM system using a keyframe based approach. Rather than searching for point correspondences though, the entire images are matched using sum-squared-difference after subsampling and applying a Gaussian blur. Keyframe based approaches are only suitable for SLAM systems where the relative pose of these keyframes is included in the estimation. We require a relocalisation system which can determine the camera pose relative to map landmarks instead.

Se *et al.* [70] achieve global relocalisation for a vision based SLAM system on a mobile robot. Like our system, a stochastic map is built of natural 3D point landmarks. However, the use of multiple cameras allows the depth of the land-

marks to be easily determined and the robot is restricted to move in a plane. To perform relocalisation they determine correspondences to landmarks in the image and use either a Hough transform or RANSAC to determine the robot pose relative to the landmarks. Landmark matching is reliably performed using the rich 128D SIFT descriptor. However, this descriptor is slow to calculate and so their system does not run in real-time.

The monocular SLAM system of Eade & Drummond [30] is able to achieve real-time relocalisation. Their technique is to first do a global search to determine which local region of the map the camera is currently viewing. This global search uses a bag of words approach similar to the appearance based mapping method of Cummins & Newman [21]. During tracking, the system records which features from a visual vocabulary are visible in each local map node. The first stage of relocalisation determines which local node the camera is viewing by searching for visual words in the current image. Once the current map node is determined, the system relocalises relative to the local landmarks. For this second stage, they also search for landmarks in the image using SIFT. However, to achieve real-time performance, they use of an abbreviated 16D SIFT descriptor. They are still able to achieve acceptable performance with this weaker descriptor by storing multiple descriptors for each landmark to capture the appearance from different viewpoints.

A fast landmark matching method is crucial for a real-time relocalisation system. In this chapter we present our randomised lists classifier and compare its performance to both the sub-realtime 128D descriptor used by Se *et al.* and the fast 16D descriptor used by Eade & Drummond.

4.1.3 Contributions

This chapter describes a module capable of detection and recovery from tracking failures in monocular SLAM. Methods for determining the camera location relative to a set of point landmarks have been shown before. The contribution here is a method fast enough to run in real-time which is well integrated with the monocular SLAM system. This tight integration leads to improved performance in landmark recognition and allows the system to build a more visually distinctive map. The resulting monocular SLAM system with relocalisation is far more robust and thus more useful outside of the laboratory environment.

The main contributions presented in this chapter are as follows:

- *Relocalisation Module.* When tracking failure does occur, this module is able to relocalise the camera relative to the map and then resume the monocular SLAM system using the map geometry and the randomised lists classifier.
- *Monitoring of tracking performance.* The monocular SLAM system is monitored to detect when tracking failure has occurred. This prevents map corruption from spurious observations which are likely in such situations.
- *Randomised Lists Classifier.* This novel classifier is used to quickly recognise map landmarks in an image based on their visual appearance. This work was done jointly with Georg Klein.
- *Online learning of map landmark appearance.* This is achieved through the integration of the randomised lists classifier with the monocular SLAM

system described in Chapter 2. By acquiring real training examples online in a process called *harvesting*, greater recall can be achieved by the classifier.

4.1.4 Outline

We will begin by presenting our relocalisation module for the monocular SLAM system. We will explain how this module detects tracking failure and how it is then able to relocalise the camera after tracking conditions improve. We will then present our randomised lists classifier which is a key component of the relocalisation module. It allows the module to quickly recognise map landmarks in the latest image from the camera. Finally, we will present results showing that the relocalisation module greatly increases the robustness of the monocular SLAM system.

4.2 The Relocalisation Module

The relocalisation module consists of several different components, each of which will be discussed in turn. First, the method of detecting tracking failure is presented. This monitors the SLAM system and, when it deems tracking to have failed, it halts normal SLAM operation and the relocalisation process begins. The method for determining the camera pose relative to map landmarks is then discussed followed by how those map landmarks are detected in the image. Finally, the procedure to restart the SLAM system after a successful relocalisation

is given.

4.2.1 Detecting Tracking Failure

The tracking quality of the monocular SLAM system is assessed at each frame by the relocalisation module. This is done by monitoring the result of attempted observations of map landmarks using active search. These observations can fail for several reasons; either

- the landmark is outside of the search region in the image,
- the landmark's current appearance is too different from the stored image patch, or
- the landmark is currently occluded by other objects in the scene.

The first situation occurs when the true camera motion is quite different from the prediction given by the motion model. High accelerations can cause this since the motion model used assumes constant velocity with noise on the acceleration.

An incorrect camera motion prediction can also lead to the second cause of observation failure. Appearance changes of the landmark due to viewpoint change are handled by warping the image patch to match the predicted camera position. If this prediction is wrong the warping can be sufficiently different to cause normalised cross correlation matching to fail.

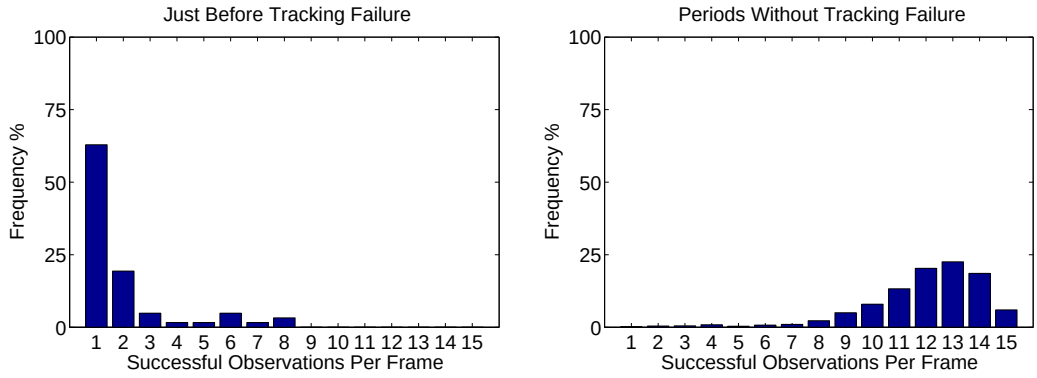


Figure 4.2: **Threshold for Tracking Failure:** We use a threshold of zero successful observations in a frame to detect tracking failure. To investigate this, an experiment was performed using a sequence with a smooth motion of 1600 frames followed by a series of 31 tracking failures. The system attempted observations of 15 landmarks in each frame. The graph on the right shows the percentage of timesteps during the smooth motion where each number of successful observations occurred. This is shown for the two frames preceding each tracking failure in the graph on the left. If the system observes fewer than three landmarks in a frame, the camera pose estimate is poorly constrained and so tracking could be deemed to have failed. However, during the long period with no tracking failures, the system observed fewer than three landmarks in 0.6% of frames but was able to continue tracking successfully using the prior and the motion model to constrain the pose. The threshold of zero successful observations for one frame is chosen to balance the need for detecting true failures and the desire for uninterrupted tracking when the system would be able to cope.

The landmark appearance in the image is also affected by motion blur or occlusion which will also lead to failed observations.

The relocalisation module monitors the success of attempted observations made by the SLAM system. A threshold was chosen where if the number of successful observations is zero for one frame, the system is deemed to be lost and the relocalisation process begins. Really, once fewer than three observations occur, the camera motion is poorly constrained and tracking could be said to have failed.

However, this threshold was found to be too sensitive. The relocalisation module would be activated even during relatively smooth motions when the number of successful observations would sometimes fall below three for a single frame (Figure 4.2). By lowering the threshold to zero and allowing the motion model to aid the tracking during these rare poorly constrained frames the system is less sensitive and begins relocalising when it is truly lost. The system is also deemed to be lost if the uncertainty of the camera pose grows too large or if no landmarks are predicted to be in view at the current estimated location.

4.2.2 Determining the Camera Pose

The aim of the relocalisation module is to determine the camera pose relative to the metric map of landmarks using the images from the camera. If the image location of the map landmarks is determined, then several methods exist which can determine the pose of the camera relative to those landmarks. Lepetit *et al.* [45] provide a good review of recent methods to solve this 'pose from n points' problem while providing their own algorithm.

The minimum number of map landmark correspondences needed to fully constrain the camera pose is four. With just three correspondences however, the pose is constrained to just four discreet positions. This ambiguity is illustrated in Figure 4.3. A fourth correspondence can then be used to disambiguate between the solutions. A survey of three–point–pose algorithms can be found in [34]. We use the method of Fischler and Bolles which was originally presented along with their RANSAC algorithm in [32]. It involves solving a fourth order polynomial

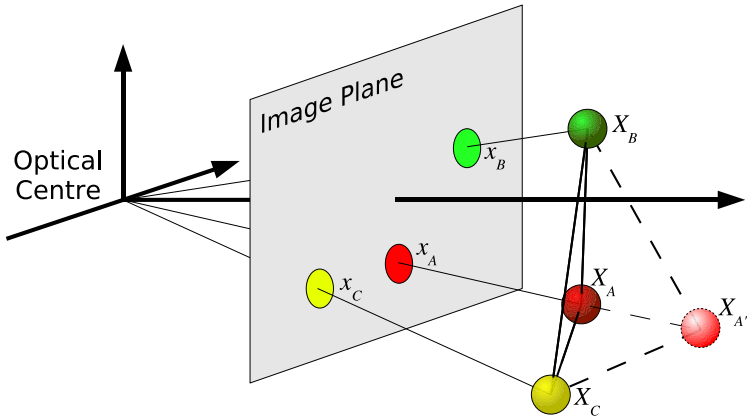
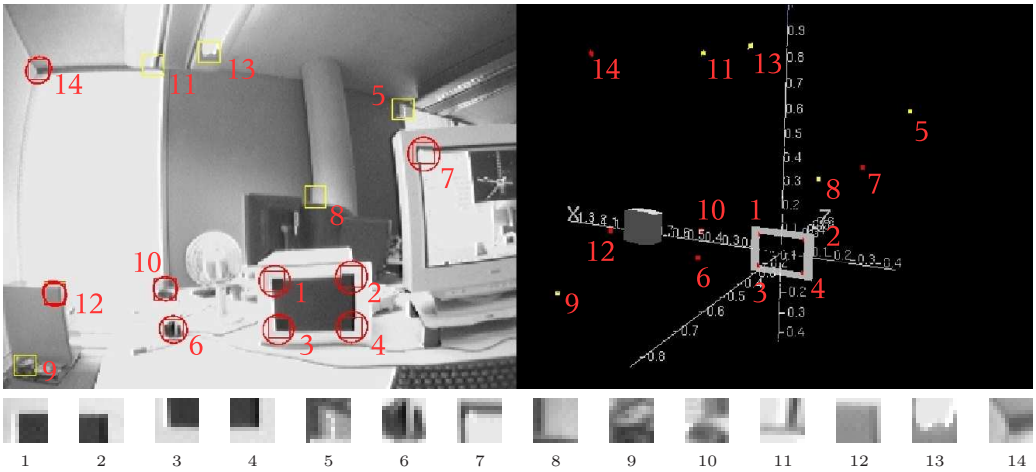


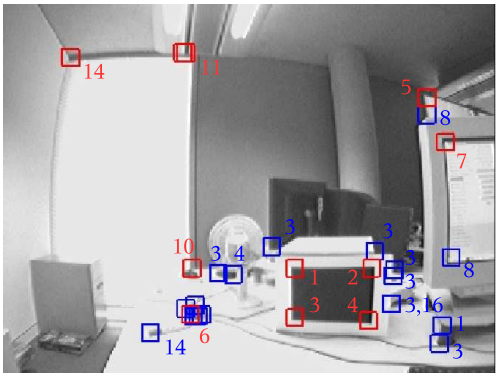
Figure 4.3: **Three–Point–Pose:** A discreet set of up to four valid camera poses can be determined from the image projection, $\mathbf{x}$, of three points, A, B, and C if their 3D position, $\mathbf{X}$, is known. There is an inherent ambiguity illustrated here, where the same image projection and relative landmark geometry results from the point A being at position $\mathbf{X}_{A'}$ instead. This ambiguity can be resolved if a fourth point is observed.

and was shown in [34] to be more numerically stable than other methods. As will be discussed in Section 4.2.3, not all of the correspondences to map landmarks will be correct. The use of the RANSAC algorithm can determine the true pose despite these false matches.

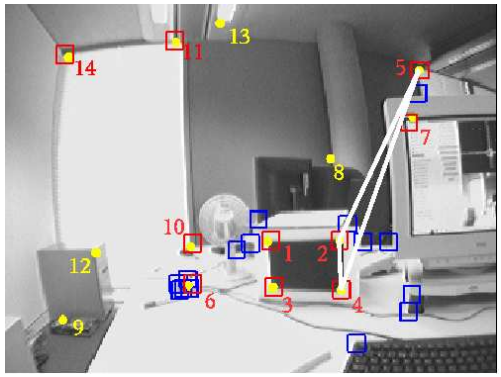
Applying the three–point–pose algorithm to a set of three landmark correspondences provides up to four hypotheses for the camera pose. We use the RANSAC method and evaluate each pose hypotheses from each randomly (with weighting and restrictions described below) chosen set of three landmark matches by testing it for consensus. Consensus for a pose is determined by checking how well that camera pose predicts the location of the image projection for the other map landmarks. A pose which correctly predicts the image location for many other map landmarks is likely to be the true pose of the camera. Sets of three corre-



(a) A small map has been built by the SLAM system to demonstrate the steps in the relocalisation process. Numbered labels are given for the map landmarks in the metric map (right), their true locations in the image (left), and their patch identifiers (bottom).



(b) Potential matches to map landmarks are found in the image. The simple NCC matching approach is shown here leading to correct matches (red), false matches (blue), failed detections, and several multiple matches. In practice, we use a more sophisticated randomised lists classifier (Section 4.2.3).



(c) A set of three potential matches (joined by white triangle) is selected and the three–point–pose algorithm is used to give a camera pose hypothesis. The other map landmarks are then projected into the image (yellow circles) using this pose and the metric map. The high consensus of seven matched projections shows this pose is correct.

Figure 4.4: **The Relocalisation Process:** The camera is relocalised relative to a small map (a). The system first detects the landmarks in the image (b) and then uses these correspondences to determine the camera pose (c).

spondences are randomly chosen and evaluated until a pose with high consensus is found or the time is exhausted.

The steps in the relocalisation process are illustrated in Figure 4.4 using a simple landmark recognition scheme. Landmarks are recognised in the image using normalised cross correlation (NCC) of each Shi-Tomasi corner point [71] with the patch stored for each landmark. This exhaustive search is costly and not robust to large changes viewpoint but it is still able to find sufficient matches in this simple case (Figure 4.4b). The more sophisticated landmark recognition algorithm we use will be presented in Section 4.3. After the set of potential matches is found, sets of three can be randomly chosen and a set of pose hypotheses is calculated. The evaluation of one such pose is shown in Figure 4.4c. A consensus of seven landmarks whose projection lies close to a match for that landmark indicates that this pose is probably the correct one.

The fraction of correct poses amongst the many hypotheses given by every combination of three landmark correspondences is tiny. However, when the correct pose is tested, it is clearly identified by its high consensus. False pose hypotheses rarely make even one correct prediction for the projection of another map landmark whereas the true pose typically gets consensus from every visible landmark. Therefore, the RANSAC search is stopped when a pose is found with a consensus of at least two correctly predicted projections.

To speed up the search for a correct pose amongst the many outliers we make some effort to select sets of three matches in a way which increases the chances obtaining three inliers. This is done by assigning a weight to each landmark

correspondence returned by the classifier: these weights are determined by a combination of a motion model and a set of match scoring heuristics.

The motion model considered when the camera is lost is different from that used when tracking: instead of a constant velocity model with Gaussian noise on the acceleration, we consider a random walk with a maximal linear velocity of 1.5 m/s (walking speed) and an arbitrarily high rotational velocity. This predicts that the camera is within a steadily expanding sphere centred about its last known position and that it could be oriented in any direction. Feature correspondences returned by the classifier are filtered by potential visibility: landmarks which could not be visible from any point in the sphere are given zero weight. Thus, our system switches seamlessly from an efficient local relocalisation immediately after tracking is lost, to a global kidnapped-robot relocalisation after a few seconds have elapsed.

We further apply a set of match scoring heuristics: matches with low classification scores are penalised, as are matches to the same landmark which occur many times in the same image. We further reject match triplets which are collinear or very close together in the image as these produce poor pose estimates. We also check match triplets against a co-visibility database maintained by the SLAM system: sets of matches which have never been observed together in a single image are culled. This check prevents attempting to calculate a pose using three landmarks from distant parts of the map which are unlikely all be correct. Together, these measures help to reduce the number of RANSAC trials required for successful re-initialisation.

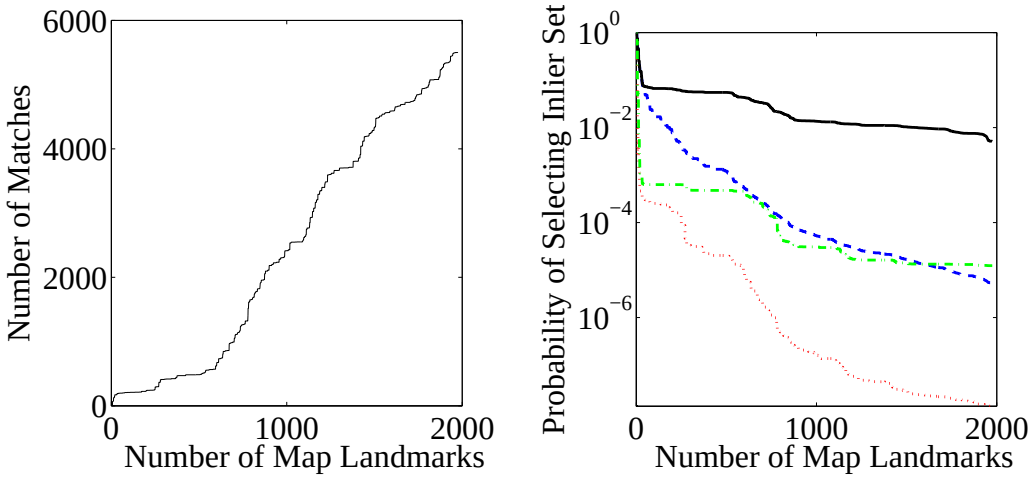


Figure 4.5: **Scalability of the Relocalisation System:** As the number of landmarks in a map grows, so too does the number of potential matches to these landmarks in an image. Here, the probability of selecting a set of three correct matches is investigated as the number of landmarks considered is increased. When landmarks which are not present in the chosen frame are considered, any matches found are false and will decrease the chances of selecting an inlier set. Without match scoring heuristics, the probability of selecting a set of three true correspondences in a large map would be very low. The graph on the right shows the probability of choosing a set of three inliers randomly without guidance (· · ·), if multiple matches to the same map landmark are penalised (- - -), if covisibility information is used (· — ·—), and if both of these checks are used (—).

Figure 4.5 shows the benefit of these match scoring heuristics for improving the probability of a successful relocalisation as the number of landmarks grows. In this experiment, a large map of 1975 landmarks was created. Then, the ability to relocalise was tested using a single image showing part of the map. Landmarks were found in the image using the randomised lists classifier which will be discussed in Section 4.3. Initially, only three truly visible landmarks are considered. For each of these landmarks, only the true correspondence to each is found in the image. With three correct matches found and no false positives, the probability of choosing an inlier set is of course 100%. As the set of landmarks considered

grows to include all of the truly visible landmarks, some false correspondences are also found so the probability of choosing an inlier set decreases slightly. The set of landmarks considered is then incrementally increased to include the landmarks that are not visible in the image. Any correspondences to these landmarks are false positives and will decrease the chances of an inlier set being chosen. When the entire map of landmarks is considered, the large number of false matches leads to a tiny probability of randomly selecting a set of three correct matches needed for relocalisation. However, by taking into account the simple heuristics outlined here, many of the incorrect combinations are unlikely to be chosen.

These heuristics allow the system to relocalise in maps of several thousand landmarks. At this point, the limiting factor then becomes the memory required to store the randomised lists classifier to recognise the landmarks. This classifier, which will be discussed in Section 4.3, trades memory for speed. At 1.25MB per class, this limits the number of landmarks that can be stored in RAM. This makes the randomised lists classifier unsuitable for arbitrarily large numbers of landmarks.

4.2.3 Recognising Map Landmarks

To determine the pose of the camera, the relocalisation module must first find correspondences to the map landmarks in the current image. The simplest approach is to use the same normalised cross correlation (NCC) matching technique that the underlying SLAM system uses for making observations. This method was used in an early implementation of our relocalisation system [90] and is il-

lustrated in Figure 4.4 on page 99.

Rather than exhaustively search the entire image, we restrict the search to corner points since all map landmarks in the SLAM system are initialised at corner points. The FAST corner detector [67] is used to quickly find corner points across the entire image. Then, every corner point is checked against every map landmark using NCC between the patch around the corner and the patch stored for each map landmark. Every match above a threshold is kept as a potential match and used to determine the camera pose via RANSAC and the three-point-pose algorithm described in Section 4.2.2.

This simple correlation based method for recognising map landmarks works well when the camera viewpoint remains sufficiently similar to allow NCC to work. However, this limits the SLAM system to recovery from only brief tracking failures where the viewpoint will not have changed much. Exhaustive correlation between every map landmark and every corner feature is also computationally quite costly typically taking several frames to complete even for very small maps. By the time the camera pose has been determined, it can already be out of date.

Other feature matching methods exist which would outperform NCC in invariance to viewpoint [7, 46, 48, 49] some of which would even be fast enough for the purposes of relocalisation. Most feature matching algorithms however, are designed to find a match in later images using only a single example of a feature whereas the SLAM system provides up to 30 examples per second of the landmarks being observed. We have developed a new feature matching method called the randomised lists classifier which takes advantage of this data available

during tracking to train a classifier. This trained classifier is then used to identify map landmarks in the image when attempting relocalisation. Furthermore, this algorithm is fast enough in both training and classification that it can be used with the SLAM system to allow relocalisation at framerate. This randomised lists algorithm is presented in the next section.

4.3 Landmark Recognition as Classification

This section describes the algorithm for map landmark recognition that is key to achieving rapid correspondence between the current image and the map landmarks. This work is based on the randomised trees classifier of Lepetit & Fua [44] and was developed jointly with Georg Klein. First, the method of Lepetit & Fua is described before outlining our modifications that yield a classifier better suited to relocalisation in monocular SLAM.

4.3.1 The Implementation of Lepetit & Fua

Lepetit and Fua [44] employ randomised trees of simple image tests to rapidly detect re-occurrences of previously trained image patches in a new input image. The system can detect re-occurrences even in the presence of image noise, changes in scale, rotations, aspect ratio and illumination changes. The novelty of the system is to treat real-time feature recognition as a classification problem, where each class is trained with (ideally) all the different appearances a single

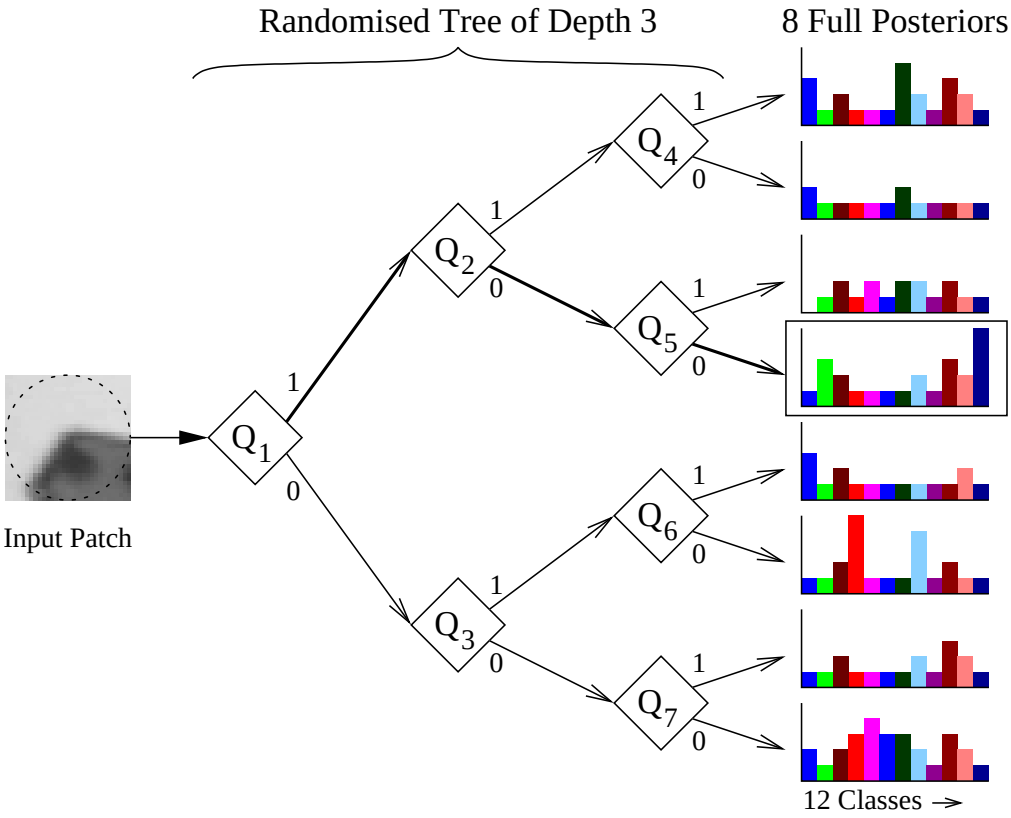


Figure 4.6: A Randomised Tree as described in [44].

feature can take due to aforementioned changes. This is in marked contrast to approaches like SIFT where a descriptor is engineered to be invariant to all these expected variations. The classification approach allows the use of a simple classifier which can be evaluated quickly; a disadvantage is the requirement to train many instances of every class.

At the centre of the system is a forest of N decision trees. A single simplified tree of depth D=3 is illustrated in Figure 4.6. Each tree maps an input image patch to one of 2^D posterior distributions (the leaves of the tree) which describe the probability of the patch belonging to one of C=12 previously trained feature

classes.[1] The mapping is the result of a series of simple binary tests Q which make the tree's nodes: each test Q simply compares the patch's Gaussian-smoothed intensity values $\mathrm{I}_\sigma(\cdot)$ at two pixel locations $\mathbf{a}$ and $\mathbf{b}$, such that

$$Q_i = \begin{cases} 0 & \mathrm{I}_\sigma(\mathbf{a}_i) - \mathrm{I}_\sigma(\mathbf{b}_i) >= 0 \\ 1 & \text{otherwise} \end{cases} . \tag{4.1}$$

Once the patch's posterior probability for every tree has been estimated, the class whose sum of posteriors is greatest forms the classification result (subject to a threshold.) While each individual tree may itself be a rather poor classifier, the combination of N trees' posteriors yields good recognition results. The posterior distributions themselves are built during a training phase; hundreds or thousands of patches belonging to each known class are simply dropped down each tree and the associated posterior is accumulated. Training patches are either synthetically generated by warping [44] or extracted from images in a carefully controlled training video sequence (called harvesting) [57] with the aim of teaching the trees all possible views of a class.

One of the key results in [44] concerns the selection of the locations $\{\mathbf{a}, \mathbf{b}\}$. During a training phase, these may be chosen from a small random set so as to maximise each test's information gain (in accordance with [1]); alternatively they can be chosen entirely randomly, which results in a small drop in classification performance but massively reduces the time required for training.

[1] In an actual implementation, reasonable values for these parameters might be N=40, D=15 and C=100-1000.

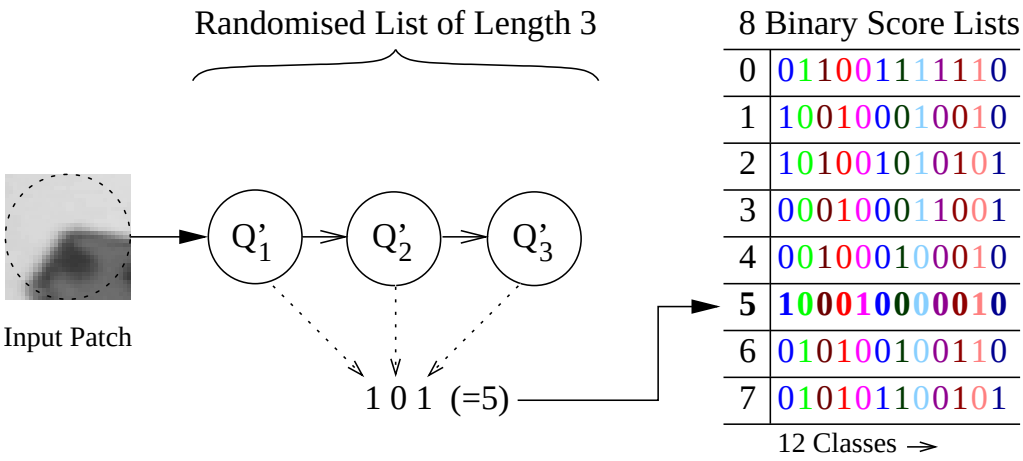


Figure 4.7: The simplified classifier structure used here.

4.3.2 Randomised Lists for SLAM Relocalisation

Lepetit and Fua originally applied the classifier described above to the recognition and localisation of known textured objects (e.g., a book) for which the classifier could be trained once with hundreds of optimally chosen classes and then re-used many times. In our monocular SLAM setting, in which the goal to the build a map and localise simultaneously, a separate training phase would be a self-defeating exercise. This means that the binary tests Q_i must be pre-selected at random.

If the tests are random, it is not necessary for the tests at any one level of a tree to be different. Instead of a tree with $2^D - 1$ potential tests, a list of D sequential tests is sufficient, and produces identical classification performance (Indeed, Ozuysal *et al.* have independently developed this approach and further improvements in [56]). The results of these tests form a binary string which indexes into the array of 2^D posteriors. Using the list structure allows for significant im-

provements in run-time speed (for example, we can replace patch rotation with a look-up-table of pre-rotated tests.)

Besides this structural change, we also make changes to the operating characteristics of the classifier; whereas the systems of Lepetit and Fua operate with many dozens or hundreds of keypoints visible per frame, our SLAM implementation uses a far sparser set of landmarks: At any given frame, only $10 - 20$ might be visible. For this reason, it is important to tune the classifier towards recall rather than precision, and this motivates further changes. We also propose some tweaks to boost efficiency and classification rates. The modified classifier is illustrated in Figure 4.7 and the changes are described below. The performance of the classifier is then described in Section 4.3.4.

Multiple Hypotheses and Class Independence - For an extended exploration, it is unrealistic to assume that all map landmarks will appear completely unique. Due to perspective and lighting changes - and self-similarities common in man-made environments - some map landmarks will resemble one another. The method of Lepetit and Fua penalises such cases: only the strongest match is returned, and since posteriors are normalised, the presence of multiple similar landmarks penalises the score of all of them, sometimes to the extent that none is recognised. Here, we consider each class' score independently and the classifier returns all classes scoring higher than a threshold. The independent treatment of classes has the further benefit that the classification rate of any one class is not affected by the later addition of other classes, which is very important for a system which is continually trained on-line.

Binary Leaf Scores - Instead of storing full normalised posterior distributions over the classes at each leaf, we store a binary score string of one bit per class. A class' score at a leaf is either zero (if no training example from that class ever found the leaf) or one (if at least one training example did.) The benefit of this approach is a reduction in storage requirements ($2^D \times C \times N$ bits for all scores) which allows operation with higher values of D.

Intensity Offset Tests - In man-made environments, image areas of uniform intensity are common. For regions of uniform brightness, tests of the form "is this pixel brighter than the other" only measure image noise. We replace tests Q_i with modified tests Q'_i:

$$Q'_i = \begin{cases} 0 & \mathrm{I}_\sigma(\mathbf{a}_i) - \mathrm{I}_\sigma(\mathbf{b}_i) >= z_i \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

where z_i is a randomly chosen intensity offset in the range 0-20. These tests increase repeatability in areas of the patch with uniform intensity. The slight reduction in illumination invariance is in practice mitigated by feature harvesting.

Explicit noise handling during training - In [44], random noise is added to training examples to make the classifier noise-resistant. This is not done here; instead, during training, we explicitly check if each training test Q'_i is close to a noise threshold. If this is the case, the ith bit of the binary string being formed is set to a "don't care" state. When the full string is formed, the scores for *all* possible values of the string are updated for the class (this is made possible by using lists instead of trees.) This reduces the number of training patches required

to achieve noise tolerance, which is important for real-time operation.

4.3.3 Training the Classifier

The classifier must be trained to recognise each new landmark that is added to the map during "steady-state" SLAM operation. When the landmark is first observed we train the classifier with 400 synthetically warped versions of the landmark appearance so that it can immediately be recognised under a variety of views. We use graphics card to perform the warps quickly. Even so this initial training is costly at around 30ms per new landmark; so that this does not adversely affect the real-time performance of the SLAM system, it is performed in a background thread. Further, we can in some cases reduce the number of warps performed: with our binary posterior representation, there is no benefit in incrementing a class posterior more than once, so we can stop training once a large fraction (95%) of warped examples re-visit previously seen posterior nodes.

This relatively crude initial training stage is subsequently reinforced using real data harvested [57] during normal SLAM operation: when a map landmark is successfully observed by the SLAM system, the classifier is also trained with this new view of the landmark.

To improve the distinctiveness of the landmarks in the map, the classifier aids in the selection of new map landmarks. When the system is initialising a new map landmark it is usually faced with a choice of potential landmarks in the image. Rather than simply take the one with the strongest corner response as in [24],

each candidate corner is instead first classified with the current randomised lists classifier. The system then chooses to initialise the landmark that scores lowest against all other landmarks already in the map. This leads to a map of landmarks which are more easily discriminated.

There is a potential problem with continually training the classifier online if only a binary posterior is used. Ideally, each new example of the appearance of a landmark harvested would only fill the posterior bins associated with the true appearance of the landmark. With noisy observations however, there is the potential that over time, every posterior bin will be filled for every landmark making the classifier unusable. Before this point, the performance would begin to degrade and the classification threshold would need to be raised. In practice though, we have not found this to be an issue. Perhaps if a system were used continuously, this problem would become apparent.

4.3.4 Classifier Performance

This section describes the performance of our randomised lists classifier. First, we analyse the effects of the classifier design choices described in Section 4.3.2. Then, the benefit of harvesting for training the classifier is shown. The classifier is also compared to the popular SIFT descriptor and the short SIFT descriptor used by Eade & Drummond for their real-time relocalisation system.

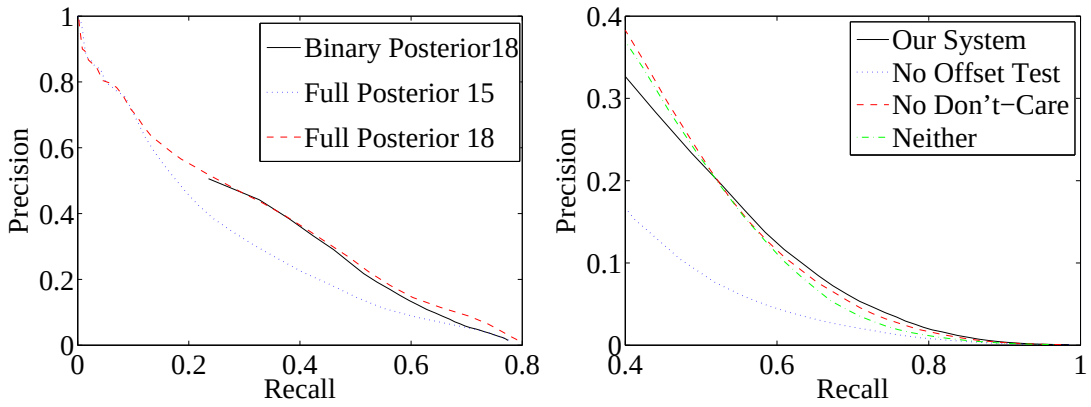


Figure 4.8: Performance of the Randomised Tree Classifier.

Effects of Classifier Design Choices

To test the performance impact of the classifier design choices described in Section 4.3.2, two tests were performed on a controlled 2300-frame SLAM run during which a map of 70 landmarks was built without tracking failure. The tracked landmark positions are considered ground truth for classifier evaluations at every frame. Classes were trained using synthetic warps only; no harvesting was performed since this would inflate the classifiers' performance under the slow-moving conditions tested. We set the operating point of the classifier to 0.65 recall (65% of the known landmarks visible in the image are detected) and test how our changes affect the classifier's precision (the fraction of returned matches which are actually inliers.) Full precision-recall curves are given in Figure 4.8.

Table 4.1: Binary vs. full posterior classifier performance

Leaf node type	Depth	Memory Use	Precision
Full Posterior	18	1321 MB	.12
Full Posterior	15	164 MB	.07
Binary Score	18	164 MB	.09

Table 4.1 demonstrates the effect of using a binary leaf score list instead of a full posterior (with 8 bits per class in our implementation) at the operating point of 0.65 recall. As may be expected, simplifying the posterior to a binary score reduces classification performance for equal tree depths; however the binary score outperforms the full posterior when using the same memory footprint, since a larger depth can be used. Using the binary score further reduces classification times and eliminates an otherwise problematic tuning constant (the uniform prior distribution density.)

Table 4.2: Performance impact of classifier modifications

Intensity Offset Test	Don't-care-bit	Precision
No	No	.07
No	Yes	.03
Yes	No	.08
Yes	Yes	.09

Table 4.2 illustrates the effects of the other classifier adaptations on classification rates. The use of intensity offsets together with the explicit training of all image noise permutations produce a noticeable increase in performance at a negligible increase in classification time. It is interesting to note that the noise-aware training by itself (without the intensity-offset tests) *reduces* classification performance: this is due to the frequent occurrence of areas of uniform intensity in many of the patches used; the use of don't-care bits in these areas causes them to match *any* input image patch.

It should be noted that the low precision scores in above tables are obtained without any landmark appearance harvesting. If this is enabled, precision increases from 0.09 to 0.2, and the actual value encountered during real-world relocalisa-

tion likely lies between these two. Finally, the heuristic guided sampling criteria described in Section 4.2.2 help boost inlier rates to a level normally manageable by RANSAC.

Benefit of Harvesting

In many applications, keypoint recognition is forced to find correspondences using only a single example view of the keypoint. This has led to the development of rich feature descriptors like SIFT [46] with inbuilt invariance to many transformations. However, rather than attempting to build some invariance into the descriptor, the ideal solution would be to learn the appearance of the keypoint over the complete range of conditions and viewpoints from real world examples. In practice, this is typically not possible since labelled training examples for each keypoint are not available. However, in our application, the SLAM system can provide reliable real world training examples at a rate of 30Hz while a landmark is visible. Harvesting provides a way to take advantage of this wealth of appearance information.

An experiment was performed to assess the benefit of harvesting on landmark recognition rates. The randomised list classifier was also compared to the popular SIFT descriptor. SIFT is typically used with no harvesting i.e. a single descriptor is calculated for each landmark to be recognised. For the experiment, fifty landmarks were initialised and tracked by the SLAM system during a three minute sequence. Harvesting was performed during the first two minutes of the sequence. The classifier was then frozen and the final minute of the sequence was used to test the recognition rate. Keypoints were detected in each frame of the

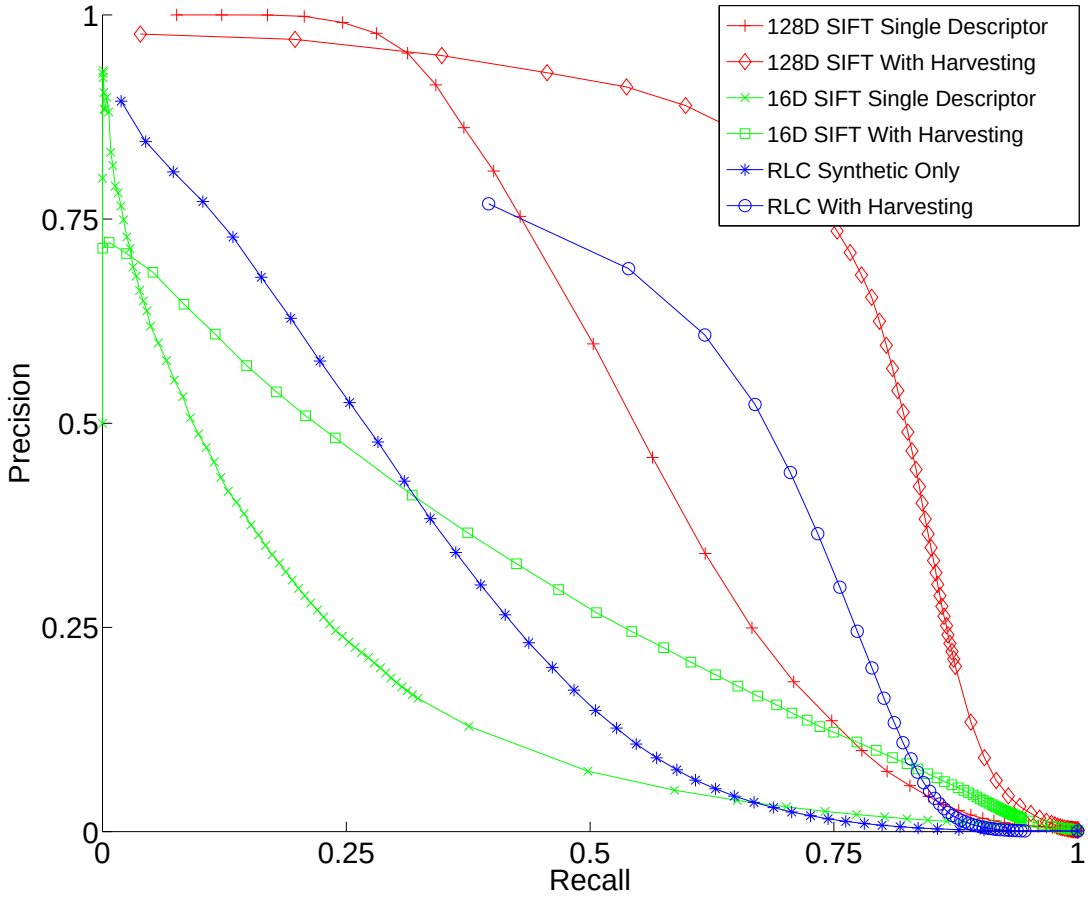


Figure 4.9: **The Benefit of Harvesting:** Classification performance was tested for 50 landmarks during the final minute of a three minute sequence. The first two minutes of the sequence were available for harvesting. A randomised list classifier (RLC) trained with only synthetic warps of a single patch performs poorly compared to a 128D SIFT descriptor made from a single patch. However, when the RLC is also trained with two minutes of harvesting from a range of viewpoints the recall is higher than for 128D SIFT. To achieve real-time performance using SIFT, Eade & Drummond used a small 16D descriptor but boosted the recall with harvesting when the appearance changed significantly. Here, harvesting was performed on every frame of the training sequence. The computational cost required to compare the huge number of SIFT descriptors is prohibitive in a real-time system whereas the classification time for the RLC does not grow.

test sequence using the FAST corner detector [68]. Each of these corners was then classified. Ground truth for landmark locations was given by the SLAM system. Multiple hypotheses were allowed since the goal is recall rather than precision. To achieve scale invariance for SIFT matching, a scale pyramid was built. Descriptors were then created for the FAST corners for all scales since FAST corners do not have an intrinsic scale unlike the DOG extrema typically used in SIFT.

The results of this experiment can be seen in Figure 4.9. With only synthetic training performed using the first observation of each landmark, the randomised list classifier performs poorly compared to a 128D SIFT descriptor created using only the first observation. However, once the randomised list classifier is trained using harvesting, its recall greatly increases.

Though not typically done, harvesting can also be performed using the SIFT descriptor. In this test, a SIFT descriptor is stored for every observation in the initial two minute training sequence used for the randomised lists classifier. Matching is then performed during the one minute test sequence against this much larger descriptor set. This greatly improves the recall. However, this method would be not be feasible in practice due to the prohibitive cost in matching against the huge (and growing) number of descriptors. In contrast, the recognition time for the randomised list classifier does not increase as more examples are trained.

Eade & Drummond [30] were able to use SIFT with harvesting and achieve real-time performance by using short 16D SIFT descriptors. Also, to reduce the number of descriptors to compare, they only harvested when the appearance of the

landmark changed enough to significantly alter the SIFT descriptor. In Figure 4.9 results are also given for this shorter 16D SIFT descriptor. Without harvesting, the 16D SIFT achieves quite poor performance but if harvesting is performed at every frame, the precision and recall is significantly improved. However, these results overestimate the performance of their system since they harvested new descriptors less frequently. The computational cost required to compare the huge number of SIFT descriptors obtained from framerate harvesting is prohibitive in a real-time system whereas the classification time for the RLC does not grow.

Timing alone also makes a good case for the randomised lists classifier for real-time systems. The classifier takes an average of 0.12ms to train each new appearance of a landmark and then 0.04ms to classify each detected corner. SIFT takes 0.29ms to create each new descriptor, however, more pre-processing is also required. A Gaussian pyramid is needed for scale invariance, and gradient images are required for descriptor creation. Combined, these take 440ms, already beyond the frame rate of the system. Faster implementations of SIFT do exist, but the computation of this rich descriptor cannot be made faster than the simple pixel intensity comparisons required for classification using randomised lists.

Lepetit has proposed a classifier-based method that does not involve harvesting [10]. Instead, a randomised tree classifier is created off-line and trained to recognise a few hundred distinct generic features. During on-line operation, features are then recognised using their 'signature'. A signature is a descriptor for a patch generated from the response of the classifier over all the generic classes. These signatures achieve performance similar to SIFT descriptors but can be calculated much faster. It also does not suffer from the growing memory footprint

required to train new landmarks when using a non-generic classifier. The price paid is the gain in recall which can be achieved through harvesting in situations where many examples are available for training.

4.4 Restarting After Recovery

After the map landmarks have been recognised in the image and the camera pose determined via the three-point-pose algorithm and RANSAC, the SLAM system can be resumed using this recovered pose. The camera pose must be put into the EKF state vector and properly correlated with the map to allow tracking to continue. To achieve this, it is initially placed in the state estimate with no correlations with the map and an artificially inflated uncertainty. Then, while keeping the map estimate frozen, the detection of each landmark in the consensus set is used as an observation for an EKF update. If the pose estimate is indeed close enough to the true estimate then one or two iterations of the EKF are sufficient to refine the pose and to reduce the uncertainty dramatically.

During this time the SLAM system is closely monitored. The recovered camera pose is only trusted after the system successfully tracks for several frames. A high proportion of observations (we use 75%) must be successful in each of these frames, otherwise the pose is deemed to be incorrect and the relocalisation process restarts. This final check makes the relocalisation system very conservative. It is better to perform relocalisation again rather than risk corrupting the map with poor data association from an incorrect pose. Only after passing this final

test is the map unfrozen and the SLAM system is allowed to continue tracking. Subsequent observations reintroduce the correlations between the camera pose and the map.

4.5 Algorithm for Our Relocalisation Method

Before presenting results for the relocalisation system described in this chapter, we provide a summary of the procedure in Algorithm 1.

Algorithm 1 Our relocalisation method: relocalise($image, 3D\_map, covisibilty$)

```
relocalisation_needed ⇐ true
best_score ⇐ 0
while relocalisation_needed do
  corners ⇐ detect_FAST_corners(image)
  potential_matches ⇐ randomised_list_classifier(corners, image)
  repeat
    valid_set ⇐ randomly_select_3_valid_matches(potential_matches, covisibilty)
    camera_poses ⇐ three_point_pose(valid_set, 3D_map)
    for i = 1 to 4 do
      score ⇐ test_for_concensus(camera_poses[i], potential_matches, 3D_map)
      if score > best_score then
        best_score ⇐ score
        best_pose ⇐ camera_poses[i]
      end if
    end for
  until time_exhausted
  if best_score > minimum_score then
    relocalisation_needed ⇐ false
    restart_SLAM_system(best_pose)
  else
    image ⇐ wait_for_next_frame()
  end if
end while
```

4.6 Results

Now, we will present results on the general performance of our SLAM system when used with the relocalisation module. Then, a single relocalisation will be analysed in more detail to show the effect of guiding RANSAC and the choice of the feature matching algorithm. Results will then be presented on the time taken for the relocalisation process. Finally, we will show the range of viewpoint change over which relocalisation can be still be achieved.

By recovering after tracking failure, our monocular SLAM system is able to complete sequences which would be very challenging (or probably even impossible) for other similar systems. The system is easily able to detect when tracking fails and then stop the SLAM system to preserve map integrity. Tracking and mapping are only resumed when the system relocalises again using previously mapped landmarks.

4.6.1 Monocular SLAM with Recovery

Given the ability to re-initialise rapidly and often, we can afford to tune our system to be much more cautious than previous systems. Thus there are times when tracking is stopped due to failed observations when dead reckoning according to the motion model might otherwise actually carry it through. However, the predicted camera poses in such cases are not likely to be accurate and as likely error grows, so do linearization errors and the possibility of poor data association. Our system simply restarts tracking and avoids these potential sources of map

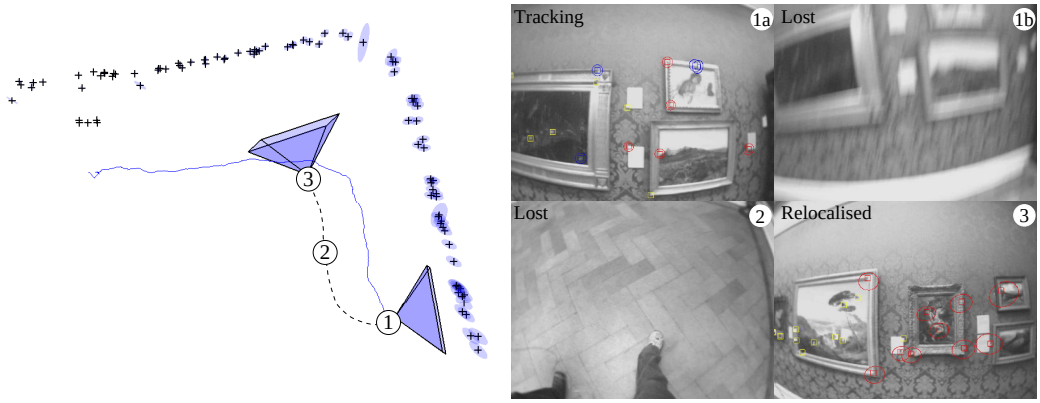


Figure 4.10: The system relocalises when it is moved to another region of the map. On the left is an overhead view of the final map (green 3σ ellipses) and trajectory estimate. The first tracking failure (1b) and subsequent relocalisation (3) are numbered in the map and in the camera frames (right).

corruption.

To demonstrate our system's mapping and relocalisation operation we track a sequence recorded in an art gallery. The camera moves along two walls of paintings. Several times during this run, tracking is lost when the camera is suddenly pointed at the floor. The corrupted map which results when this sequence is tracked without using the relocalisation module was already shown as the motivation for relocalisation in Figure 4.1 on page 89. By using the relocalisation module, the system is able to detect all of these occasions, correctly stop mapping, and does not resume mapping until the camera once again points at one of the walls which were previously mapped, whereupon relocalisation occurs within 1-2 frames. Figure 4.10 shows the map of 80 landmarks and trajectory for the run. The first relocalisation is annotated in the map, numbered to match the images on the right.

To evaluate the reliability of the system, it was run for over an hour (>100,000 frames) in an open-plan office environment. The map generated contained 70 landmarks. Despite erratic camera motion and hundreds of tracking failures, the map was not corrupted. Although the volume of space traversed by the camera was small compared to the size of the environment, this is purely due to limitations in the underlying SLAM system (no support for occlusions and N^2 complexity in map size for the EKF). Relocalisation was consistently successful until the arrival of co-workers changed the environment (the map management policies do not scale well to sequences of this length).

Table 4.3: Typical relocalisation timings (map size 54)

Corner detection (found 145 corners)	2 ms
Classification (found 42 matches)	12 ms
Selection of Match Triplets (chose 335)	0.3 ms
Evaluation of Poses (tried 14)	0.7 ms
Final Pose Optimisation	4 ms
Total	**19 ms**

Table 4.3 shows the time taken (on a Core 2 Duo 2.7GHz processor) by the individual stages of a typical relocalisation with 54 landmarks in the map. Here the classifier has returned 42 matches of which 7 are inliers: RANSAC must therefore find one of the 35 inlier triplets out of 11480 possible match triplets. In this run, 335 triplets were hypothesised but most of these could be rejected before even attempting three-point pose by testing for co-visibility and image geometry; the three-point-pose algorithm was only run on 14 hypotheses. In this case, a correct pose was produced in a total of 19ms.

Further system timings are shown in Figure 4.11; this plots the processing time required per frame for another run of the art gallery sequence. Each frame (in-

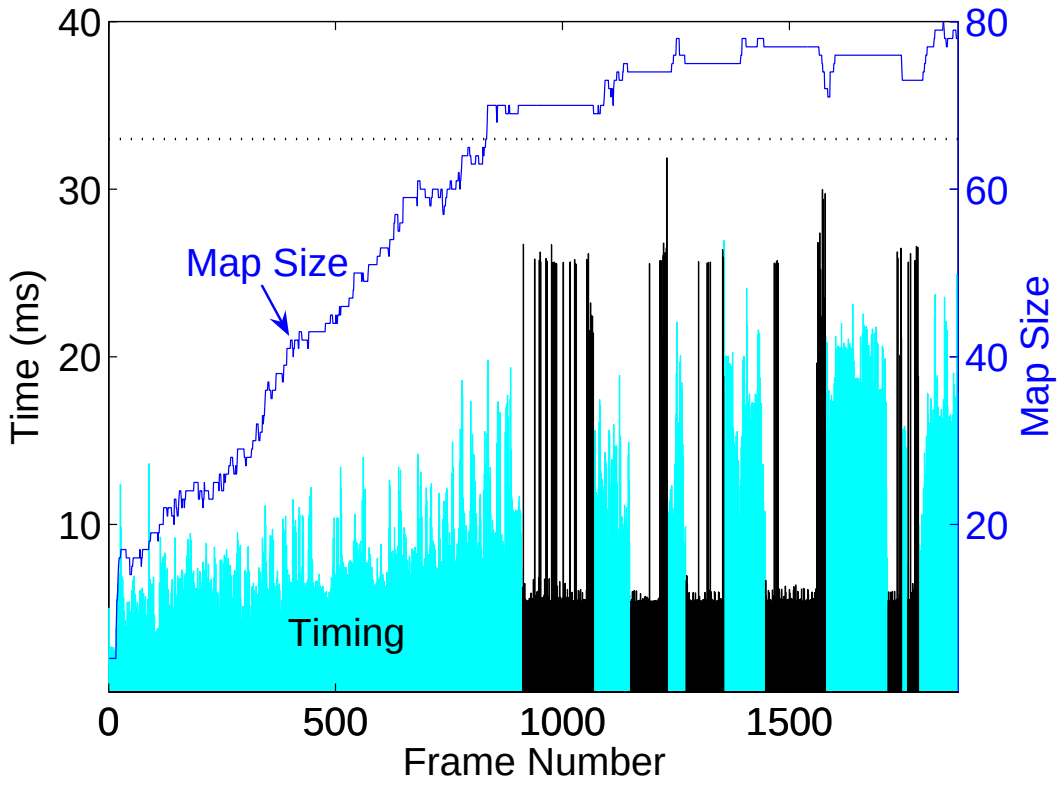


Figure 4.11: **Processing Time:** While performing SLAM on the Art Gallery sequence shown in Figure 4.10 processing is performed in real time. Frames where the system is lost and relocalisation is attempted are shown in black.

cluding those during which recovery is attempted) is processed in under 33ms; this is possible for maps up to $80 - 90$ landmarks, beyond this the N^2 EKF updates cause skipped frames. Black timing values indicated frames during which the system was lost; of these, those frames with low run-times are those in which insufficient landmarks were detected to attempt relocalisation.

The volume of space in which relocalisation is possible was also investigated. A small map was generated and harvesting performed while moving the camera

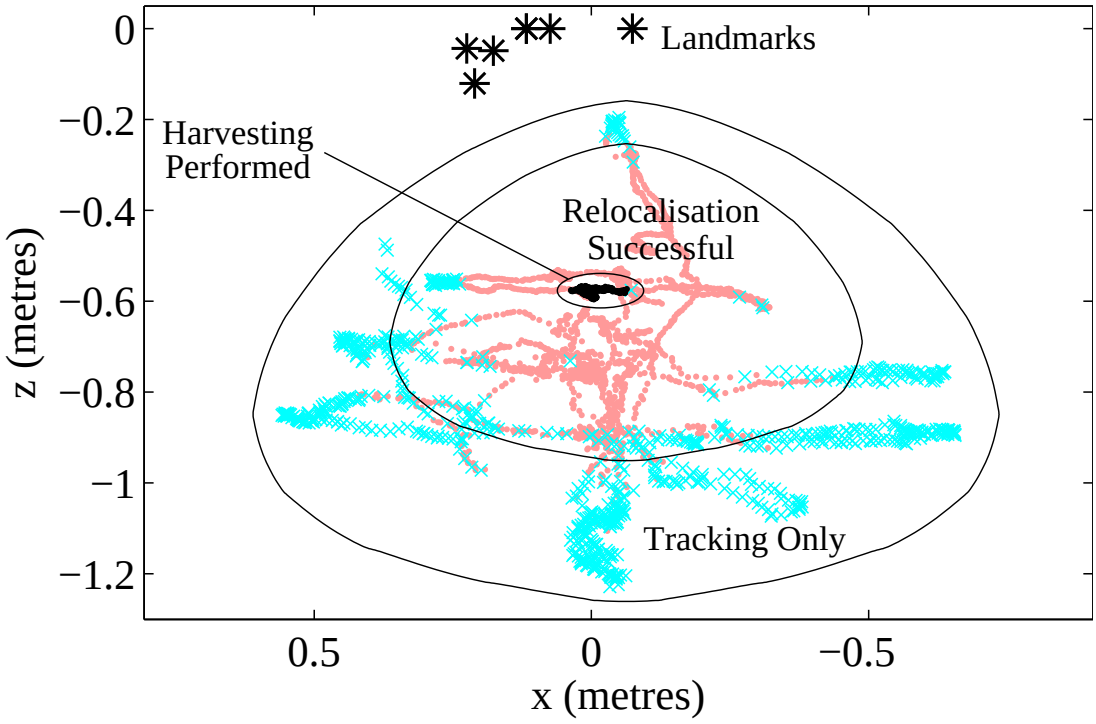


Figure 4.12: **Relocalisation Volume:** This figure shows the typical volume over which the system can relocalise after having harvested in only the small central volume. The pose was tracked and the system attempted relocalisation at every frame outside of this (• for success, × for fail). Some frames from this experiment are shown in Figure 4.13.

across a small range of viewpoints indicated by the metric map in Figure 4.12 and the top row of images in Figure 4.13. Mapping and harvesting were then disabled, and the system attempted relocalisation and tracking from a much larger set of viewpoints. Relocalisation was successful from a range of novel views that have not been trained during harvesting. Some examples are given in the bottom row of Figure 4.13.

Even with a significant range over which recovery should be possible, there are still cases in which it can fail in practice. In order for the system to relocalise

Figure 4.13: **Relocalisation from Novel Views:** Here tracking (and thus harvesting) was only performed along a short linear motion illustrated by the top row of images. This is sufficient to allow relocalisation from the views shown in the bottom row despite the difference in viewpoints. A metric representation of this experiment is shown in Figure 4.12.

several conditions must be met: Mapped landmarks must be visible in the image; the relative landmark positions in 3D must be accurate; the corner detector must detect a corner at the landmark location; finally, at least five correct matches to landmarks must be found by the classifier (three for the pose determination and two for consensus). In practice, when recovery from a particular pose is not possible, moving the camera a small distance in any direction will often allow the system to succeed.

4.6.2 Detection vs. Tracking

Since the relocalisation algorithm is able to recover the camera pose at frame-rate with no prior information, it is natural to ask why one would bother with

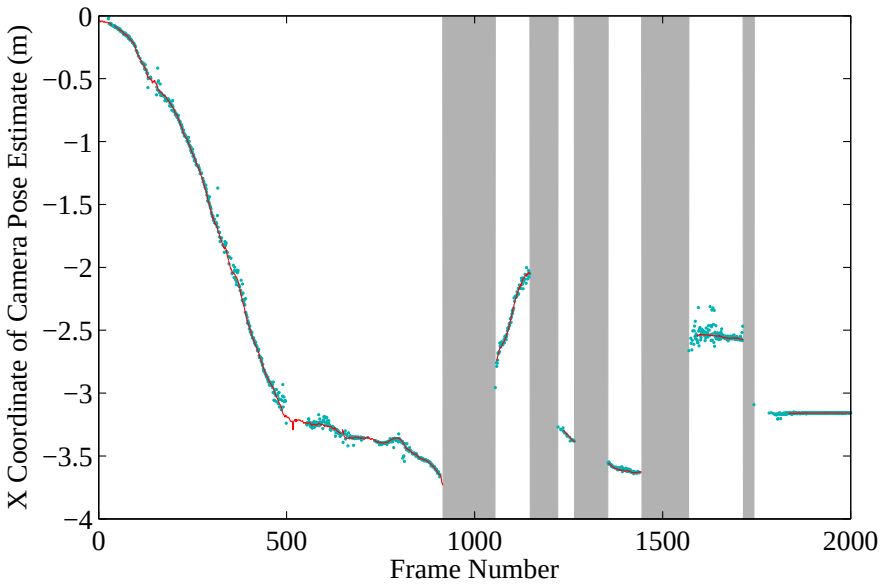


Figure 4.14: Relocalisation vs. Tracking for the art gallery sequence. The x-coordinate of the pose estimated by frame-to-frame tracking (solid line) is compared to the output of the relocaliser if it is run at every frame of the sequence. The grey regions indicate un-trackable sections when the camera is pointed at the floor or the images are heavily blurred.

tracking. Timing alone makes a persuasive case: active landmark search typically takes only 3ms to establish correspondences, and this is independent of the map size. This contrasts with 19ms or so required for a map of about 50 landmarks for full relocalisation. Accuracy is a second reason: in experiments comparing the position estimates of localisation and tracking for every frame of a sequence, we have consistently found SLAM poses to be smoother and more accurate. To make the comparison fairer, some form of pose refinement and filtering could be introduced to smooth the results of the relocalisation algorithm: but this is pretty much what the SLAM system does anyway.

There are also regions where tracking is possible but relocalisation is difficult.

At frame 500 in Figure 4.14, the camera is moving around a corner between two walls. Many of the landmarks used by the system at this point lie on a statue sitting in the corner. These landmarks do not fit the planar patch assumption we use so they are only successfully tracked for a small part of the motion. These short-lived landmarks are sufficient for tracking but the relocalisation algorithm cannot cope with the small viewing angle for these landmarks and their large depth uncertainty.

Finally, we continually update the classifier with new images of each patch being tracked; using this classifier to track the patch in the next frame is the classic recipe for drift! Here, the image patches used for tracking are never modified and so drift is avoided, even with continual training of the classifier.

4.7 Conclusion

In this chapter, we presented a relocalisation module for a monocular SLAM system to help it recover from tracking failures which are inevitable in real-world operation. Instead of trying to avoid tracking failure altogether, the module automatically detects the failure, halts the SLAM system, and begins relocalising instead. Mapping is only resumed when the camera pose has been redetermined, thus preserving map integrity.

Relocalisation is performed by first using a randomised lists classifier to establish landmark correspondences in the image and then RANSAC to determine the pose robustly from these correspondences. We have shown this method to be reliable

and fast for the large maps which can be built using the submapping techniques which will be described in the next chapter. We have shown results from the system running reliably in various scenarios, including one sequence which lasted for over an hour. During this time it created a good map despite frequent camera shake, sudden motion and occlusion.

While we would not claim that real-time monocular SLAM is now ready for use outside the lab, the ability to carry on mapping after a tracking glitch has dramatically increased the usability of our experimental system: tracking failures for the most part just do not matter anymore and the consequent ability to track long live sequences routinely will greatly facilitate investigations into a variety of other SLAM problems.

In the next chapter, it is shown how the ability of the relocalisation system to recognise regions of the map can greatly help the system when exploring larger environments. When building larger maps it is important for the system to recognise when the camera has returned to a previously mapped region so that the errors in the map which inevitably accumulate due to noisy measurements can be corrected. The relocalisation system described here is adapted for this purpose to detect loop closure and allow map joining.

5

Loop Closing and Map Joining

The results in this chapter have been published in the "Inside Data Association Workshop" at Robotics Science and Systems (RSS), Zurich, Switzerland, 2008 [86], the proceedings of the International Conference on Intelligent Robots and Systems (IROS), Nice, France, 2008 [87] and the journal Robotics and Autonomous Systems, 2009 [85].

5.1 Introduction

In this chapter we show how the relocalisation algorithm developed in the previous chapter can be applied to two other important problems in mapping – loop closure

detection and map joining.

5.1.1 Why is loop closing important?

As a robot explores its environment and the initial landmarks pass out of view, it becomes more uncertain of its position relative to these landmarks. If the robot traverses a loop in the environment and recognises and reobserves these landmarks then this growing uncertainty is quickly reduced. If however, the robot does not recognise the original landmarks then it will believe it is still exploring new territory and the uncertainty in the map will continue to increase. It is therefore very important for any SLAM system to recognise when it is revisiting a previously mapped area. This process is called loop closure detection.

Ideally, using the map built during SLAM, the robot would expect to see the old landmarks as they come back into view and attempt to observe them using active search. Unfortunately, the relative position uncertainty between the robot and landmarks increases as the robot traverses the loop. With such large uncertainty, active search is infeasible as the search regions can easily exceed the size of the image. Secondly, though the EKF algorithm is good at mapping the local geometry of landmarks, errors can accumulate along large trajectories causing large errors in the global structure of the map. Recognising the loop closure and so correcting the map is crucial for building maps of large areas.

In a loop closure situation, the position of the two ends of the loop will often be in different places in the map estimate despite the fact that they actually both

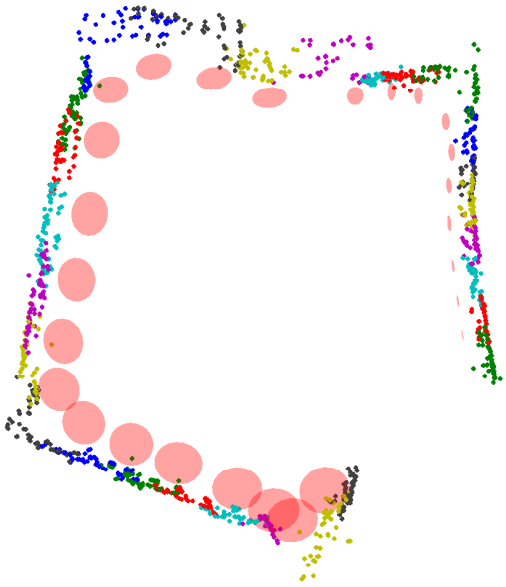
represent the same region of the world. Ideally, the estimation algorithm used provides a consistent estimate and so the two ends of the loop will be within the uncertainty level given by the estimator. If this is the case, the system will recognise that it may have returned to a previously mapped region and can attempt to reobserve the previously mapped landmarks. These observations then provide the constraints needed to close the loop.

However, after traversing a large loop, the true uncertainty is often so large that standard data association techniques will no longer work. With such a large uncertainty, it can be difficult to determine from the map estimate if the system is reobserving an old region of the world or observing a new one for the first time. Both would have reasonable probability. Rather than use standard data association techniques a separate algorithm should be used in loop closure situations.

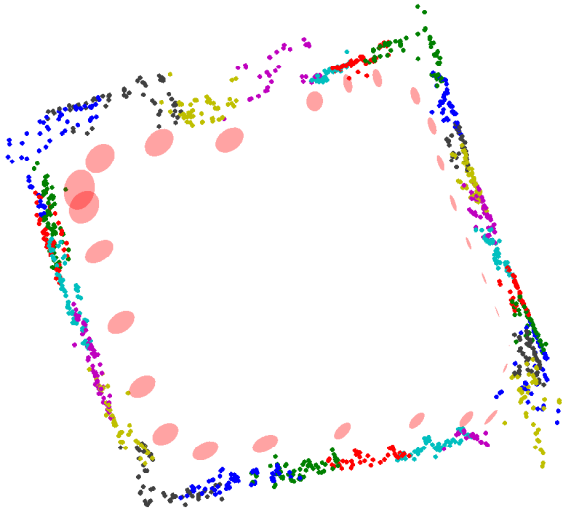
An independent loop closure detection system is also useful if the estimate becomes inconsistent. Most SLAM algorithms become overconfident while traversing a large loop due to linearization errors and inaccuracies in the models used. If this is the case, the two ends of the loop in the map built are outside of the estimated uncertainty. A loop closure detection system which recognises when the robot has returned to a previously mapped region regardless of the uncertainty in the map would still be able to close the loop and improve the map.

This situation can be seen in Figure 5.1. The monocular SLAM system has built a map of a large university courtyard using the submapping techniques described later in this chapter. Even though the camera has returned to the same place in

(a) Aerial view of the courtyard.



(b) Before loop closure.



(c) After loop closure.

Figure 5.1: **Keble College Sequence:** Twenty eight submaps with a total of 1983 landmarks were made during the 230m trajectory around the courtyard. Ellipses indicate 1σ uncertainty bounds for the origin of the base reference frame for each submap.

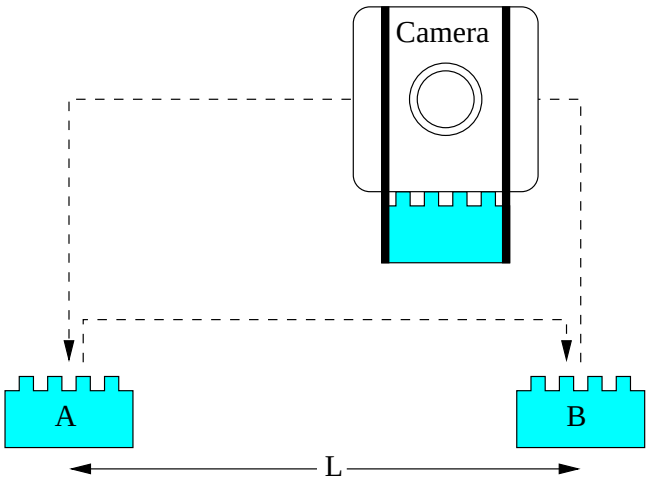


Figure 5.2: **Scale Drift Investigation:** Since absolute scale is unobservable, monocular SLAM systems are susceptible to scale drift during exploration. In this experiment, the camera was moved in a loop (- - -) while facing a textured wall approximately 15cm away. At two points in this loop approximately 1 metre apart (labeled A and B) the camera was clipped into position using Lego bricks attached firmly to a tabletop and the camera itself. At these points, the camera pose was left in the state vector so that the position of A and B could be estimated. After three circuits around the loop the three estimates of the distance L were compared to show that the local scale had drifted during the exploration (Table 5.1. Loop closure detection allows the scale drift to be corrected when a loop is closed.

the world at the end of the sequence, this is not reflected in the state estimate due to noisy observations and imperfections in the motion and measurement models. Using the loop closure detection model presented in this chapter, the system recognises the previously mapped area when it comes back into view and is able to correct the state estimate.

Another issue which can be alleviated through loop closure detection is scale drift. The absolute scale of the world in monocular SLAM is unobservable because all of the observations are bearing only. The scale of the map is arbitrary but is

	Estimate of Length L	Uncertainty in Estimate (1 Standard Deviation)
Loop 1	4.08	0.042
Loop 2	3.41	0.042
Loop 3	3.84	0.045

Table 5.1: **Scale Drift Investigation:** The camera performed three loops of the scene shown in Figure 5.2. The length, L, was estimated in each loop but due to scale drift three different values for L were found. The length given is unitless since absolute scale is arbitrary.

should ideally remain constant over the entire map. In practice however, as the system explores the scene and moves away from the initial landmarks, the local scale can begin to drift from its initial value since it cannot be observed.

We have conducted an experiment to demonstrate the phenomenon of scale drift using the setup shown in Figure 5.2. The camera was moved around a loop while facing a textured wall. At two positions, A and B, in this trajectory, the camera was clipped into a fixed position using a Lego bricks firmly attached to the table and the camera. At these two positions, the camera pose was kept in the state vector so the system could estimate the position of A and B relative to the map. After finishing the loop as described, the process was repeated a further two times. In these later loops, the system was prevented from reobserving old landmarks and was forced to believe it was continuing exploration. The final map contains correlated estimates of the position of landmarks initialised in all three loops of the trajectory. If the scale remains constant throughout the entire map then the three estimates of the distance, L, between A and B will be the same. This however was not the case.

Table 5.1 shows the three estimates given for the distance L. The estimates were found using the correlated estimate of the position of A and B in the portion of the map corresponding to each loop. During the sequence, the scale of the map drifts leading to three different estimates of the same length. This problem of scale drift is analogous to the drift in orientation encountered in laser-based SLAM systems. Since a laser provides absolute range measurements the scale is observable and so does not drift. However, if no absolute orientation sensor (e.g. a compass) is used then with only local measurements of orientation are available. This causes drift in the global orientation estimate as the robot explores. Loop closure allows this drift to be corrected each time a SLAM system returns to and correctly recognises a previously mapped region of the world.

5.1.2 Why is map joining important?

Map joining is the process of joining two independently built maps of the same environment by recognising where an overlap in the maps occurs. These are where a different region in each map actually represents the same region of the world. This process is closely related to that of loop closure.

Traditionally, map joining has been used for multi-robot mapping systems where two or more robots are used to map the same environment. Each robot begins building its own map and the system attempts to join these maps into a single large map as overlaps between the maps are found.

In hand-held single-camera systems though, map joining is even more important.

With a limited viewing angle, a quick rotation is enough to suddenly point the camera at an unmapped region of the world. In this situation, the system could wait until the user points the camera back at the mapped landmarks and relocalise. Alternatively, we propose that the system should begin a new map at this point and join the two maps when the system later returns to a previously mapped region of the world. In this way, the system makes better use of the data and is therefore able to make larger and more accurate maps. This approach is also used by Eade & Drummond [30].

The process of recognising overlap between two independent maps for map joining is related to that of detecting loop closure. The loop closure detection module described in this chapter is used for both purposes.

5.1.3 Approaches to loop closure detection

A loop closure detection system must first be able to reliably recognise when a robot has returned to a previously mapped location. Then, to allow the metric map to be corrected, the metric alignment of the two map regions must also be determined. When working with bearing only sensors like a single camera, a further difficulty is encountered. The scale of this alignment as well as the rotation and translation must be determined.

If an incorrect loop closure detection is incorporated into a map, the map will be corrupted and become unusable. More sophisticated SLAM algorithms can track multiple hypotheses [64, 82] or even undo decisions but these techniques are

computationally very costly as the number of hypothesis increases. Therefore, a good loop closure detection system should be tuned to give very few (ideally zero) false positives, while still correctly identifying many of the true positives.

Many methods for loop closure detection have been proposed. We divide the approaches into three categories:

- **Map–to–Map** – where correspondences are sought between landmarks in two parts of the map. These techniques can use both the landmark appearance and their relative geometry to determine the correspondences.
- **Image–to–Image**[1] – where correspondence is sought between the latest image from the camera and the previously seen images.
- **Image–to–Map** – where correspondences are sought between the latest image from the camera and the landmarks in the map.

Map–to–map matching is a common strategy for 2D robotic systems which use laser or sonar sensors because single scan from these sensors typically does not contain rich enough information to reliably find matches. One strategy is to accumulate a series of scans over time using scan matching [47]. This is well suited to aligning the most recent scan to the map given a good prediction of the robots position. However, if there is a large error in this prediction, the scan matching algorithm has difficulty converging. This makes it difficult to use in loop closure situations where the prediction is typically quite uncertain. Instead,

[1] In non vision based SLAM systems 'image' refers to the sensor space for whichever sensor is being used – for instance the latest scan from a 2D laser.

Gutmann and Konolige [33] propose to use scan matching locally but to perform loop closure detection using correlation between different regions of the map. They have some success with this approach but in order to limit false positives, large map patches must be used for the correlation. Naturally, since the matching uses only metric information, the patches must also have distinctive geometry to improve confidence that a match is unique.

The Atlas framework by Bosse *et al.* [9] also uses map–to–map matching to detect loop closure. However, instead of comparing the map patches directly, it constructs a 'signature' for each submap for comparison. This signature is created by extracting descriptors of geometric features such as the angle and distance between two lines or the distance between two points. The submaps are first matched using their signature and then the relative transformation between the maps is found using the landmark positions. To remove ambiguity due to repetitive texture, each submap is first compared to itself and any feature corresponding to other features in the same submap are removed from the signature.

Loop closure detection using laser range finders or sonar sensors is difficult because of the indistinctive information these sensors provide. Using cameras for detecting loop closure provides much richer information. Map–to–map based loop closure detection has been demonstrated using monocular vision system by Clemente *et al.* [20]. The appearance of each map landmark as well as its position is stored in the map and used for map–to–map matching. This particular method will be discussed further in Section 5.6.1 when it is used for comparison against our method.

The richer information offered by cameras allows sensor based image–to–image and image–to–map matching techniques to be used. Both Košecká *et al.* [42] and Ulrich and Nourbakhsh [83] record a series of keyframes in an environment. Then, as the robot is driven back around the environment, it is able to localise against these keyframes using the new images recorded. Both methods use simple voting schemes but they differ on the image information used for the matching. Košecká *et al.* use SIFT features in the images where as Ulrich and Nourbakhsh use colour histograms. These techniques are both able to successfully localise the robot, but the number of keyframes is quite small and the goal of simultaneous mapping is not achieved.

Ho and Newman [37] perform SLAM on a mobile robot using a using a laser scanner and scan matching to build a map. However, loop closure detection is performed using a camera which records a series of key frames. Matches are found between the images using the bag of words technique [72] where features from a visual vocabulary are detected in each frame. Loop closure is detected as a series of consecutive images which match well. This work was then built upon by Cummins & Newman [21] who formulate the problem in a Bayesian framework. Instead of using the vision system only to detect loop closures, they abandon the laser scanner and perform SLAM in appearance space rather than metric space. They too use the bag of words technique but they also learn a first order generative model for the co-occurrence of the visual words. This makes their system more robust to perceptual aliasing by learning that the visual words related to patterns like brickwork or bushes tend to co-occur in the world. It also allows the system to determine the probability that the latest image is of a new

place which is crucial to allow it to act as a SLAM system. This method will also be used for comparison against our method and so will be described in more detail in Section 5.6.2.

A similar image–to–image based technique is that of Angeli *et al.* [2]. Rather than creating a vocabulary offline like [21, 37], it is built incrementally as the system explores. This gives the advantage that the system can be run on different environments without training. However, it cannot take advantage of the learned prior information that systems with an offline vocabulary can.

Image–to–image based techniques can take advantage of the rich appearance information offered by cameras, but the map built by the SLAM system also contains useful information. In the case of monocular SLAM, the map contains the geometric structure of the environment which is unobservable from a single image. Map–to–map based techniques take advantage of the information contained in the map. However, they have the disadvantage that the loop closure detection is limited to using information that has already been included in the map. The map must contain salient, stable and reliably recognised landmarks.

The method presented in this thesis falls under the image–to–map category. We have adapted the relocalisation module presented in Chapter 4 for the purposes of detecting loop closure. Comparisons are made between this loop closure detection method and a map–to–map based technique (Clemente *et al.* [20]) and an image–to–image based one (Cummins & Newman [21]).

Recently, Eade & Drummond [30] have presented a loop closure detection module

with similarities to both our own system and that of Cummins & Newman [21]. The visual words search is used to determine regions of the world with similar appearance. Then, map landmarks within that region are matched in the image before using three–point–pose and RANSAC to determine the camera's relative pose. The global-to-local approach of this recent system offers efficiency advantages over the method presented in this chapter when used in large environments.

5.1.4 Contributions

This chapter describes a new solution to loop closure detection for a monocular SLAM system. The solution to this problem requires several contributions:

- *Image–to–map loop closure detection:* This system finds correspondences between the current image and landmarks in the map using appearance and metric information.
- *Trajectory based map alignment:* A method for the determination of relative map alignment including scale by tracking a common trajectory relative to the two map regions.
- *Map joining:* A method for combining two independently built monocular SLAM maps.
- *Topological distance metric:* A method for determining topological distance of landmarks for monocular SLAM systems.

5.1.5 Outline

First, we will present our method for place recognition which relies on image–to–map matching followed by trajectory based alignment. Then, we will show how this technique can be used to merge two independently built maps into a single correlated map by recognising the overlap between them. Then, we will outline the submapping framework used in our system to build larger maps. Our place recognition technique is then used to detect loop closure within these large maps which can then be corrected using the loop closure constraints. We will compare our own loop closure detection algorithm with two other quite different techniques suitable for monocular SLAM.

5.2 Image–to–Map Matching: Our Method

In this section we outline our method of recognising when the camera has returned to a previously mapped region of the world. This ability is crucial for both independent map merging and loop closing. The method we use is based on the relocalisation module presented in Chapter 4. This process has two stages starting with the initial detection of the overlap event. Then, the relative alignment of the two map regions is determined. These stages will be discussed in the following sections.

5.2.1 Initial Detection

The relocalisation module discussed in Chapter 4 has been adapted for use as a loop closure detection system. Alongside tracking, each new image from the camera can be used to attempt relocalisation in other parts of the same map or in previously built maps. This relocalisation attempt will fail when the camera is exploring new regions of the world. However, if the camera moves so that a previously mapped region of the world is again visible, this relocalisation attempt should be successful. This successful relocalisation indicates that the current map being built by the system overlaps at this point with the previously mapped region.

Note that this relocalisation technique does not require the same landmarks to exist in both map regions. The monocular SLAM system creates only a sparse map of the world and there is no guarantee that it will choose to initialise the same landmarks when it remaps an old region. It can be building a map of one set of landmarks when it relocalises using a completely different set of landmarks which it mapped when it previously mapped the currently visible region of the world.

5.2.2 Determining the Alignment

When a relocalisation is successful, it gives a correspondence between the current camera pose being tracked, and the pose given by the relocalisation in the other map region. The correspondence between these two poses gives the translation

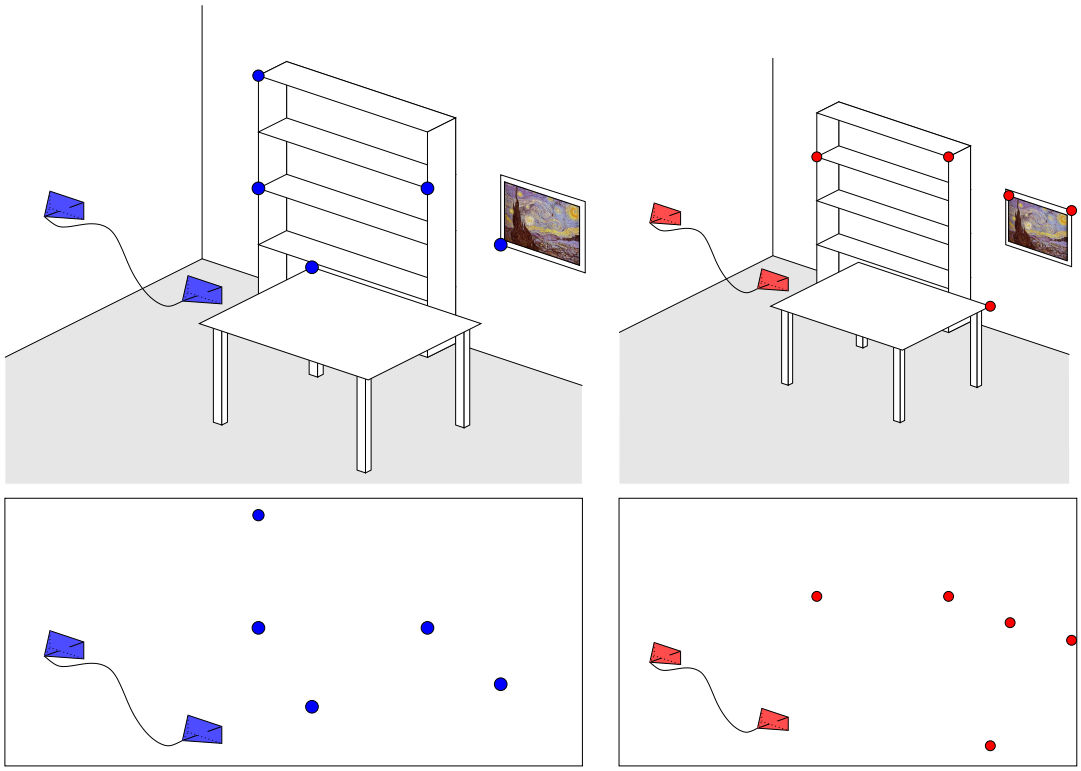


Figure 5.3: **Determining the Alignment:** After the initial detection of a loop closure, the system determines the relative transformation needed to align the map regions and so correct the global map. This is done by tracking the camera relative to each set of landmarks and then aligning this common trajectory in scale, rotation and translation. This is shown here for two maps (red and blue) of the same world region using different landmarks, •. For clarity, the maps here differ only in their overall scale. The trajectory used to determine the alignment must be correlated with the landmarks to properly correct the map. Adding this many camera poses to the state would be too costly, so for efficiency, we use only two camera poses from the trajectory which is sufficient to fully determine the transformation needed for alignment.

and rotation needed to align the two sets of map landmarks. However, a single pose is not enough to determine the scale difference between the map regions. To achieve this, the camera is tracked for some time relative to each set of landmarks. The transformation required to align the trajectory relative to each map is also the transformation we require to align the two map regions themselves. This idea is illustrated in Figure 5.3.

If the trajectory relative to each set of landmarks were estimated perfectly, the closed form solution proposed by Horn [39] could be used to determine the relative rotation, translation and scale needed to align the two trajectories. In practice though, noise in the observations and inaccuracies in the map will lead to noisy trajectory estimates. To properly account for these effects, the trajectory used to determine the alignment should be placed in the state vector so that the uncertainty and the correlation to the map are also estimated. Including all these camera poses in the state vector would be too costly since the tracking must be done at framerate. Therefore, we include just the first and last poses from this common trajectory. Two poses separated by a distance are sufficient to fully determine the relative map alignment.

The relative position and orientation of the first pose in the trajectory is used to align the maps in rotation and translation. The relative scale difference is then determined using the distance between the first and the last pose in the two trajectories. This is sufficient for the submapping strategy we use where the relative transformation between each submap is required. If desired though, once an overlap is found between two map regions in this way, they can be combined into one fully correlated estimate. This will be discussed in Section 5.3.

During this period of tracking relative to both map regions, the map estimate for the map where the system relocalised is kept frozen. Observations of these landmarks during the overlap are not permitted to update the landmark position estimate. They are only used to track the pose of the camera during the common trajectory. This is done for two reasons.

Firstly, if the relocalisation was a false positive, this is usually detected while attempting to track the common trajectory. Tracking will fail as the system attempts to track relative to a part of the map which does not truly reflect what is visible in the camera. Freezing the map prevents false observations corrupting the map estimate if the relocalisation turns out to be incorrect.

Secondly, observing the two sets of landmarks in the same image from an uncertain camera position causes correlations between the landmark position estimates. Therefore, when tracking the camera relative to both map regions, correlations would develop between the map estimates. Freezing one set of landmarks prevents these observations from affecting the map estimate and so the map regions remain uncorrelated. Avoiding this correlation is very important if the initial detection proves to be false. Moreover, the submapping strategy we use assumes submaps to be statistically independent (Section 5.4). If submaps are kept not independent but their correlation is known from their construction, then other submapping techniques such as [59] could be used.

5.3 Independent Map Merging

Two maps depicting overlapping regions of the world can be merged into a single map using the image–to-map matching method described here. Map merging allows a single large map to be built using several different sequences taken in the same environment. It is also very useful after a tracking failure. If, after the failure, landmarks are no longer visible in the frame, relocalisation is not possible. Instead of just waiting for the old landmarks to come back into view, the system can begin a new, independent map. Later, when the camera returns to the previously mapped region, these two independent maps can be merged.

To merge two independent maps, the relative rotation, translation, and scale between the maps must be determined. The trajectory based image–to-map alignment method described in Section 5.2 can provide this information.

At each new frame, while tracking in map A, the system attempts relocalisation in the map with potential overlap, map B. If this relocalisation is successful, then a possible overlap between map A and map B is indicated. While the SLAM process continues in map A, the pose is tracked relative to the frozen map B starting from the initial relocalised position.

If the camera pose is tracked successfully for several seconds in map B then a true overlap between the two maps is deemed to exist. To align the two maps, a single pose correspondence, $\hat{\mathbf{x}}^A_{cam} \rightarrow \hat{\mathbf{x}}^B_{cam}$, from the common trajectory is sufficient to give the relative rotation and translation but not the scale difference. To determine the scale difference, the distance between this first pose, $\hat{\mathbf{x}}_{cam}$, and

a second pose, $\hat{\mathbf{x}}_{cam^*}$, from the end of the trajectory is used. These two pose estimates are maintained in the EKF state vector, $\hat{\mathbf{x}}$, so they become properly correlated with the landmarks through the covariance matrix, $\mathtt{P}$.

After transforming the maps into a common coordinate frame using the transformation determined from the common poses, the result is two aligned but uncorrelated maps.

$$\hat{\mathbf{x}}_{aligned} = \begin{pmatrix} \hat{\mathbf{x}}^A_{cam^*} \\ \hat{\mathbf{x}}^A_{cam} \\ \hat{\mathbf{x}}^A_{map} \\ \hat{\mathbf{x}}^B_{cam^*} \\ \hat{\mathbf{x}}^B_{cam} \\ \hat{\mathbf{x}}^B_{map} \end{pmatrix} \tag{5.1}$$

$$\mathtt{P}_{aligned} = \begin{pmatrix} \mathtt{P}_A & 0 \\ 0 & \mathtt{P}_B \end{pmatrix} \tag{5.2}$$

To calculate the correlation between the two maps, the constraint that the corresponding poses are identical is imposed. This can be done using an EKF update where the 'observation', z, is the difference between the corresponding poses.

$$z = \begin{pmatrix} \hat{\mathbf{x}}^A_{cam} - \hat{\mathbf{x}}^B_{cam} \\ \hat{\mathbf{x}}^A_{cam^*} - \hat{\mathbf{x}}^B_{cam^*} \end{pmatrix} \tag{5.3}$$

An EKF update is performed using the prediction that this observation should equal zero and the uncertainty of this measurement is also equal to zero. Afterwards, the combined map is correlated and the extra camera poses can be

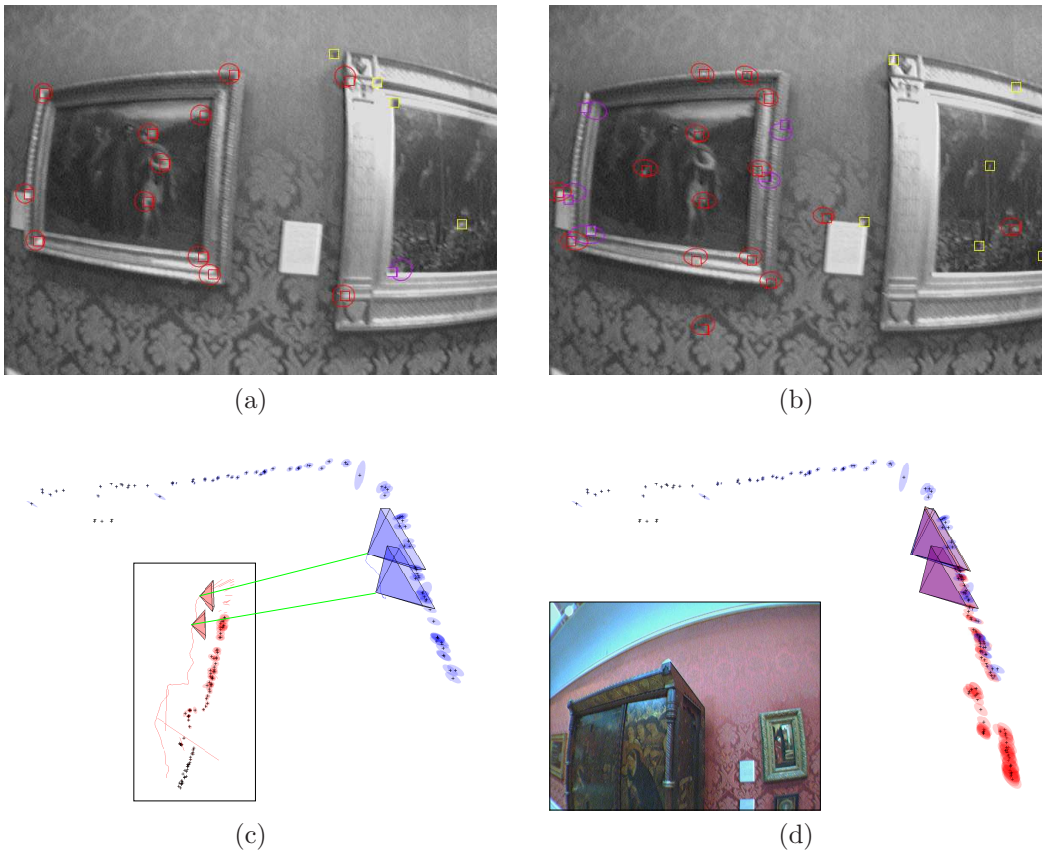


Figure 5.4: **Independent Map Merging:** The system tracks the camera during a second sequence in the same art gallery used in Figure 4.10. During this second sequence, the camera eventually moves into a region already mapped in the first sequence. The system relocalises in the first map (blue), and tracks a common trajectory relative to each map (c). The landmarks being tracked in the same image frame for the first and second map are shown in (a) and (b) respectively. Note that though the maps overlap in the world, they do not necessarily share landmarks. For such sparse maps, map joining based on common landmarks is not suitable. Instead, two poses from the common trajectory are used to determine the alignment and the two maps can then be merged into one as shown in (d). The inset in (d) shows the wardrobe which causes the bulge in the second map (red).

removed from the state.

This map merging process is shown in Figure 5.4. A new map was begun in a different part of the art gallery used in Figure 4.10. When landmarks from the old map come into view, relocalisation occurs in the old map, and a common trajectory is estimated as shown in Figure 5.4(c). Two poses from this common trajectory are then used to merge the two maps into the single correlated map shown in Figure 5.4(d). This method of joining independent maps was introduced by Tardós *et al.* [77] where the position of common landmarks in independent maps are equated in this manner to produce the combined map.

5.4 Building Larger Maps

Loop closure detection becomes important as larger environments are mapped so a method to extend the mapping ability of the monocular SLAM system is needed. Due to the update time for the EKF algorithm, the monocular SLAM algorithm described in Chapter 3 is limited to building small maps of less than 100 landmarks if real-time operation is required. A common approach to allow EKF based systems to map larger environments in real-time is to break the environment down into a set of submaps, each of which is small enough to allow fast filter updates. The resulting estimate is typically globally suboptimal during construction but the local structure is usually good enough for most applications. At any time, an estimate as optimal as the normal EKF (up to linearisation errors) can be constructed by performing map joining between all neighbouring

submaps but this process is computationally expensive. Also, this results in a fully correlated map which is expensive to update further and so is usually delayed until it is needed. Submapping also improves the consistency of the estimate by capping the absolute angular uncertainty of the robot [3]. Several submapping techniques have been demonstrated for monocular SLAM.

Eade & Drummond [30] maintain a graph of local maps. Each set of observations made is assigned to the map which gives the least linearization error. Edges in the graph, specifying a similarity transformation, exist when there is a set of common landmarks between the two local maps. The graph edges are updated iteratively using all of the measurements of these common landmarks using standard Levenberg-Marquardt bundle adjustment [80]. Further edges are added to the graph when detected by the loop closure detection system.

Clemente *et al.* [20] extend Davison's system to work in larger environments using the Hierarchical SLAM algorithm [31]. A series of statistically independent submaps is created, each small enough to be maintained by an independent EKF. When the number of landmarks in a submap reaches a limit, a new submap is begun. The relative rotation and translation of the coordinate frame of the new submap is given by the final camera pose in the parent submap. However, the relative scale change between the maps must also be estimated while ensuring independence between the maps. To achieve this, new landmarks are initialised in the child submap at the image location of the landmarks in the parent submap. Once, the depth of these landmarks is estimated, the geometry of these common landmarks can be used to determine the relative scale. The common landmarks must be reinitialised with unknown depth in the child submap because submaps

must remain independent when using the Hierarchical SLAM algorithm. When loops in the environment are detected, the Hierarchical SLAM algorithm allows the global map to be improved by optimising the transformations between the submaps given these loop constraints. Clemente *et al.* present results for a single loop around a university courtyard.

Piniés & Tardós [59] present a similar submapping method to [20] but their algorithm relaxes the strict probabilistic independence required between submaps in Hierarchical SLAM. Their algorithm allows information such as landmark position estimates to be shared between submaps by keeping the submaps conditionally independent. This leads to better initialisation of new submaps with a small increase in computation to propagate information between submaps.

An interesting new submapping proposal for monocular SLAM is that of Chli & Davison [17]. Rather than partitioning submaps based on the number of landmarks like [20, 59] or by deviation from linearity like [30], they propose to group landmarks according to the mutual information in the predicted observations. The result is a hierarchical partitioning of the world according to the correlation between landmark estimates. This technique is well suited to visual SLAM since it can naturally handle the landmarks over a range of scales from very close range detail up to distant landmarks visible from many locations. An efficient estimation framework to take advantage of the proposed submap partitioning remains for future work.

We have used the method of Clemente *et al.* to map larger environments. However, unlike [20], we also give results for challenging multi-loop environments and

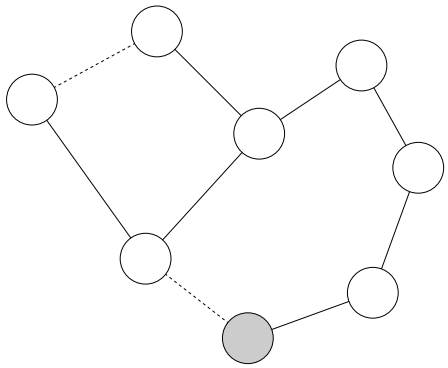


Figure 5.5: **Submapping Graph Structure:** The global map is represented in our system as a graph of submaps. Observations are only used to update landmarks in one submap at a time (•) to retain the statistical independence between the submaps. When the number of landmarks in the current submap reaches a threshold, a new submap is initialised. The edge linking these two submaps (—) represents the estimated transformation between the coordinate frames of the two maps. The relative translation and rotation is specified by the camera pose in the previous submap when the new submap is created and the relative scale is determined by common landmarks initialised in the new submap at their last observed position in the old submap. Edges are also added to this graph when loop closure is detected (- - -). The transformation for these edges is determined via our trajectory based technique. The system only allows transitions between submaps linked by an edge. Transition is performed using the relocalisation module to preserve statistical independence.

submap retraversal. As the camera moves through the environment, a series of submaps is created. The next submap is initialised using the final pose in the previous submap to represent the relative translation and rotation and the scale is determined from independently estimated common landmarks. A detected loop closure also provides a link between two submaps with their relative translation rotation and scale. The framework is illustrated in Figure 5.5.

This submapping process was used to produce a map of a multi-loop trajectory in a library. Figure 5.6 shows a diagram of the trajectory and layout of the library.

The map created by our system from this sequence is shown in Figure 5.7.

In this sequence, the camera first traverses a small loop. When it returns to the beginning, the loop closure event is detected using the method described in Section 5.5. This introduces a new link between the two submaps (shown in green in Figure 5.7a). As the camera re-enters each of the old submaps it transitions into each of them. The system predicts how many of the landmarks from a neighbouring submap should be visible at each frame. A transition is triggered when a larger number of landmarks would be visible if the camera pose were tracked in that submap. To retain statistical independence between the two submaps, the relocalisation module is used to determine the camera pose in the neighbouring submap independent of the current estimate. Later, when the camera begins to move into an unexplored region, a new series of submaps is created. Eventually, the camera completes this second loop and the loop closure event is again recognised and the new link is added between the submaps.

This final global map can be represented in two ways. The first, is to draw each submap in a relative framework. Here, the submap graph is traversed in a breadth first search using the transformations between each submap to determine its relative position. This method is used in the Atlas submapping algorithm [9]. It is sufficient for most applications involving exploration or retraversal since new links can always be added by a loop closure detection system. Alternatively, using the Hierarchical SLAM algorithm, the map can be globally improved using the constraints supplied by the detected loop closures. The composition of the relative transformations around each loop in the submap graph should equal the identity. By enforcing this constraint using non-linear optimisation techniques (we use

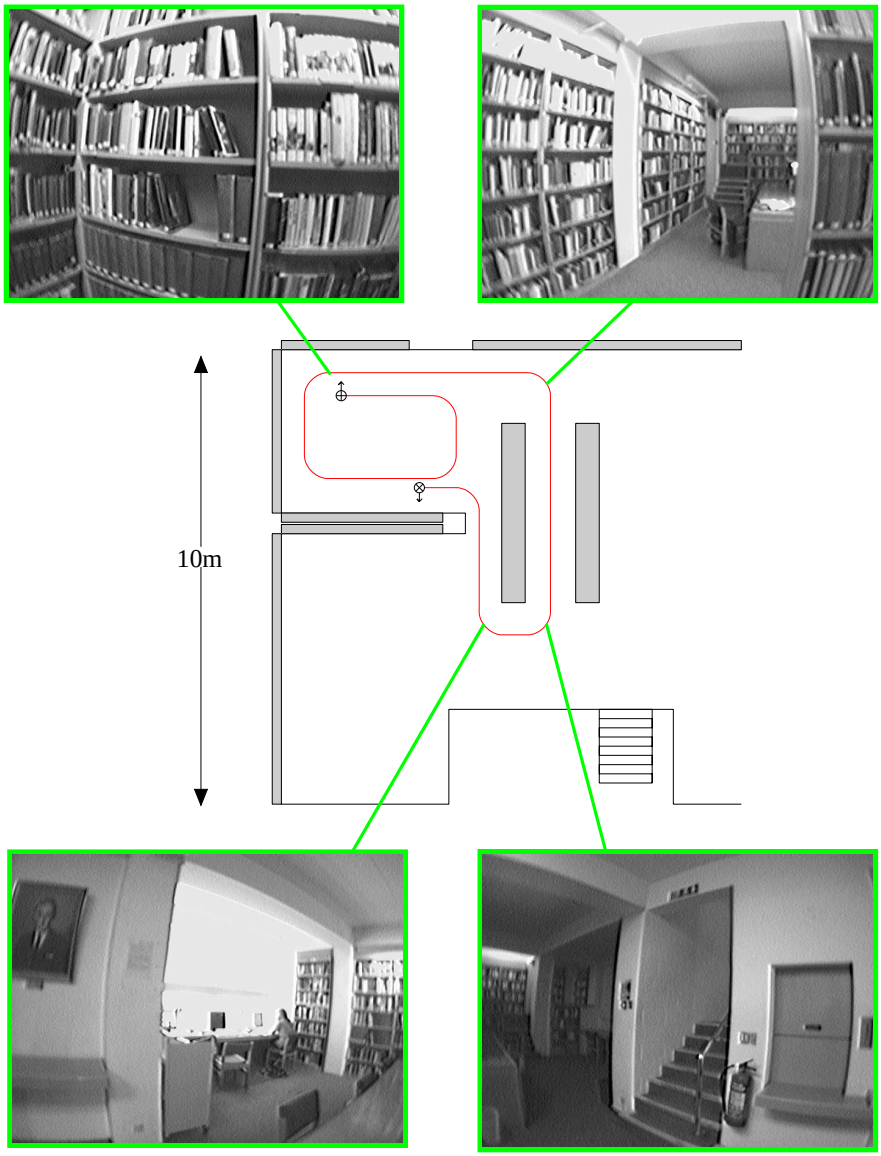
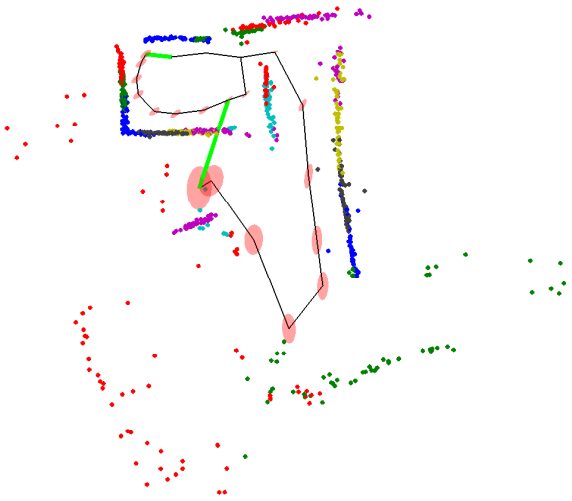
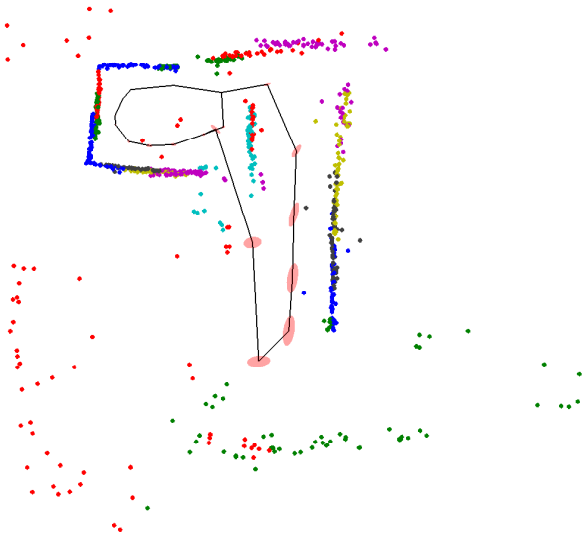


Figure 5.6: **Library Sequence Schematic:** This diagram shows the room used for the library sequence. The camera starts at $\oplus$ and follows the trajectory around a multi-loop path through the library to $\otimes$. The arrows indicate the direction the camera faces which is roughly perpendicular to the direction of motion. Four example frames are given for the sequence indicating the range of appearance of the scene. This sequence demonstrates the systems ability to retraverse previous submaps and close multiple loops. The results for this sequence are shown in Figure 5.7.



(a) Before loop closure optimisation.



(b) After loop closure optimisation.

Figure 5.7: **Library Sequence results:** Twenty submaps were created as the system built a map of the library shown in Figure 5.6. Ellipses indicate 1σ uncertainty bounds for the origin of the base reference frame for each submap. The graph connecting submaps is shown in black with thick green edges indicating links made using loop closure detection. The scale of the two submaps in the lower part of the figure was not well estimated due to the camera undergoing mostly rotation with little translation at this part of the sequence.

the iterated EKF as proposed in [31]), a globally more accurate map can be created. This technique was used to produce the map shown in Figure 5.7b. This optimisation alters the relative transformations between submaps and removes the statistical independence. The local changes this would induce in each submap are ignored. However, a globally optimal map is rarely needed in practice. It is useful for displaying the final map created for human interpretation or for globally efficient path planning. For our purposes, the global map is kept in the relative framework until a globally optimal map is required.

5.5 Loop Closure

Once maps of larger environments can be built, loop closure detection becomes very important to limit the uncertainty and correct the errors which accumulate in the map estimate. Like the map merging algorithm, we use our image–to-map matching method described in Section 5.2 to detect the loop closure events. However, unlike map merging, in loop closing, we are attempting to relocalise against landmarks within the same map. We first discuss a method of delaying premature loop closure where the system would attempt to observe these topologically distant landmarks using just active search. The loop closing module must supervise the SLAM system and delay loop closure until sufficient evidence is available.

5.5.1 Delaying Loop Closure

As the system traverses a large loop in the environment, the uncertainty in the camera position relative to the map landmarks initialised at the start of the loop grows. If an observation of these landmarks is attempted using active search the search region would be very large (typically larger than the image itself), making it unlikely the correct landmark will be found within the region. An observation with incorrect data association must be avoided since it corrupts the map estimate and causes the tracking system to fail.

Our approach is to prevent the system from attempting these observations and delay the loop closure until a separate module has detected it (Section 5.2.1). During normal tracking, the system should only attempt the observation of landmarks which are in the current local neighbourhood in the map. However, this neighbourhood cannot simply be defined as a metric distance within the map. It must capture the topological information of the landmarks.

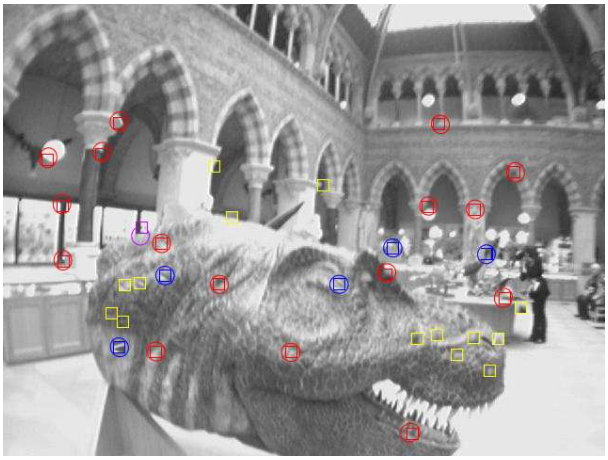
Topological distance for monocular systems can be illustrated using the following example. If the camera approaches a wall from far away, the system ceases to attempt to observe some landmarks as their scale changes beyond a limit. As it approaches, it adds new landmarks attempting to keep a constant number of observable landmarks in view. The resultant map will consist of many landmarks added at different scales. Though they may be metrically close, they can be topologically distant. If the camera then translates along the wall close in for a while and then moves away from the wall again, the large scale landmarks will come back into view. However, the system should not attempt to observe

them as usual. Though metrically the camera has not moved a great distance, these landmarks are now topologically very distant. This effect can be seen in Figure 5.8 where the camera moves along the head of a dinosaur in a museum. It then moves backwards and then back to the front of the head again.

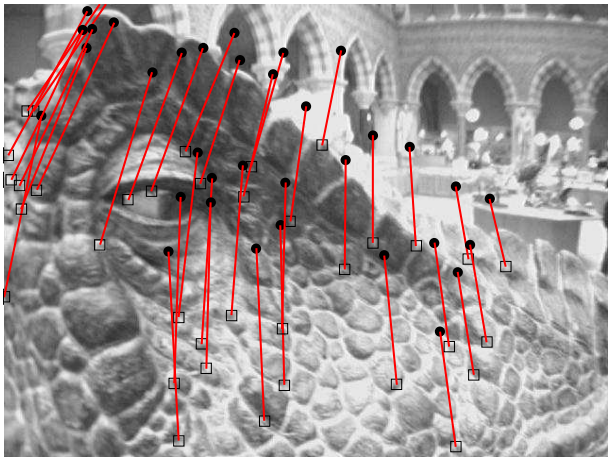
To determine which observations to attempt, we make use of the covisibility data from all the landmarks in the map. With every set of observations, a tally is updated indicating which landmarks have been successfully observed together. Using this information, a simple graph is constructed where a vertex corresponds to each landmark, and the edges indicate those that have been observed together. This graph provides an easy way of determining which landmarks are in the local neighbourhood and which are not. Those which are distant in the graph are not eligible for observation since their relative position to the local landmarks is very uncertain and attempting their observation would likely lead to incorrect data association. Only a loop closure detection will result in the successful co-observation of landmarks which were initially distant in this graph thus adding a new edge.

This effect of preventing observations is also achieved when performing submapping by only attempting observations within the current submap. Transitions to other submaps are only allowed if the other submap is a neighbour of that submap, or if a loop closure module has detected an overlap and thus added a new edge to the graph.

Another way of determining landmark covisibility in a stochastic map is to compute the inverse covariance, the information matrix. Landmarks that have been

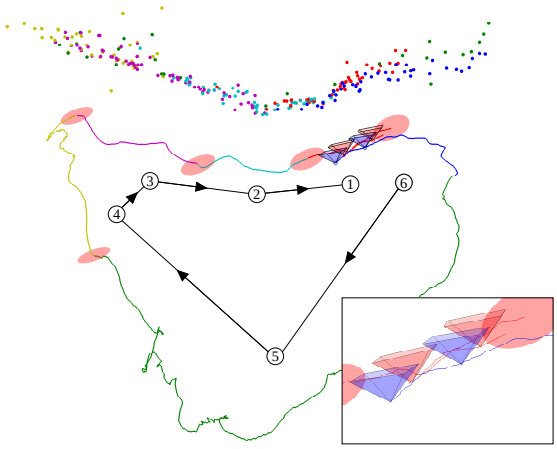


(a) The system tracking in a frame halfway through the sequence. Ellipses (mostly circles) indicate the 3σ 'active search' regions for successful (red) and unsuccessful (blue) observations. An incorrect observation of an occluded landmark has been rejected using joint compatibility (purple). The remaining landmarks in this submap (yellow) were not measured in this frame. The system does not attempt observation of landmarks initialised when the camera was very close to the dinosaur head due to their topological distance.

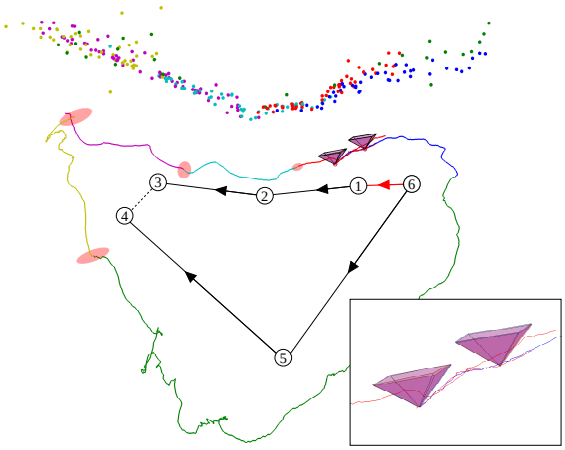


(b) When the camera has traversed the loop, there is a discrepancy between the predicted (•) and true image location (□) of the landmarks in the first submap at the moment of loop closure. This discrepancy is due to the accumulation of errors as the camera travelled around the loop causing the incorrect pose estimate.

Figure 5.8: **Delaying Loop Closure:** This sequence demonstrates the necessity of delaying loop closure as described in Section 5.5.1. The camera moves along the dinosaur head before moving backwards. At this point, the initial landmarks are visible in the image again but at a reduced scale. (Figure continued on next page.)



(c) An overhead view of the map estimated by the SLAM system before the loop closure has been completed. More distant landmarks than those on the dinosaur head are outside the borders of this figure. The submaps are labelled 1-6 in the graph-based representation shown alongside the global metric map. The submaps are drawn relative to the 6th submap and the ellipses indicate the 3σ uncertainty in the position of the origin of each submap. The two sets of poses from the common trajectory would overlap if the state estimate were perfect. This can be seen more clearly in the zoomed in inset.



(d) The global map after the loop closure has been taken into account. Using the two sets of poses from the common trajectory, the correct relative rotation translation and scale change between the 1st and 6th submaps is determined. This adds a new edge to the graph and closes the loop. The submaps are still drawn relative to the 6th submap but the new edge allows a different traversal.

Figure 5.8: **Delaying Loop Closure Cont'd.:** The system must be prevented from attempting observation as normal since active search is unreliable with such large uncertainty. A separate loop closure detection system must be used to recognise when the camera revisits a topologically distant location.

covisible at some point will have a high value of co-information. However, this inversion is expensive to compute. A tally of covisibility is more efficient and is sufficient for our purposes.

5.5.2 Loop Closure Detection

To detect loop closure events, our image–to–map matching method is used. Relocalisation is attempted in topologically distant regions of the map or in the case of submapping, in submaps which are more than one edge away in the graphical representation. Each submap is kept small enough that there is sufficient time remaining after processing each frame to attempt relocalisation in one other submap. The system cycles through the submaps until a relocalisation is successful.

If the common trajectory can be tracked relative to the other submap then the loop closure detection is deemed to be valid. The two common camera poses in the trajectory relative to each submap can then be used to determine the relative transformation between the submaps.

5.5.3 Correcting the Map

If submapping is not used or the loop closure occurs within a submap then the map can be corrected using the method that was used for map merging (Section 5.3). The constraint that the two poses from the common trajectory cor-

respond is imposed. This corrects accumulated errors in the map, reduces the uncertainty, and closes the loop. SLAM can then continue as before.

If the loop closure is detected with a different submap, then a new edge can be added between these submaps in the global map representation. The relative transformation for this edge is given by the trajectory based alignment described in Section 5.2.2. The new edge indicates that these two submaps are actually neighbours. Now, when representing the global map relative to the current submap, this new edge can be used in the breadth first search. This is shown in Figure 5.8.

The new edge in the global map also adds a new loop constraint. In Hierarchical SLAM, the global map can be optimised using the loop constraints as described in Section 5.4. A globally optimal map is useful for global path planning or visual inspection of the map by a human. For autonomous navigation or augmented reality purposes though the locally optimal relative submap transformations are sufficient. Figure 5.7 on page 157 showed this global optimisation process in a multiple loop trajectory.

5.6 Comparison to Other Methods

In this section we compare the performance of our loop closure technique to two other methods which have been used successfully in monocular SLAM systems. We have chosen the map–to–map method of Clemente *et al.* [20] and the image–to–image method of Cummins & Newman [21]. First, these two methods are

described in more detail. Then, we describe our method for comparing the three techniques. Finally, the results of the comparison are given and discussed.

5.6.1 Map–to–Map Matching: Clemente *et al.*

Clemente *et al.* [20] presented a method to close loops in monocular SLAM maps based on finding correspondences between common landmarks in different submaps. The algorithm used is a variable scale version of the original geometric compatibility branch and bound algorithm (GCBB)[53].

First, the visual appearance of the landmarks is used. The image patch stored for each of the landmarks in submap A is compared to the patches for the landmarks in submap B. This is done using normalised cross correlation and relies on landmarks being initialised from a similar location so that their patch will be similar. The threshold is kept quite low to give greater invariance to viewpoint and insure a large number of true positive matches. The appearance based correspondences between landmarks in submap A and submap B are known as the unary constraints.

Next, the geometry of the submaps is used. The relative distances between all the landmarks in each submap are calculated along with the uncertainty. A set of landmark correspondences is compatible if the distances between landmarks for the set in submap A and in submap B are compatible in terms of Mahalanobis distance. These binary constraints are calculated for a range of scales to account for any scale difference between the submaps.

Finally, the maximum clique which is compatible in both geometry and visual appearance is found. This set indicates which landmarks in submap A correspond to landmarks in submap B. The relative scale between the submaps is given along with the correspondence set and the translation and rotation can easily be determined from the relative positions of the corresponding landmarks.

This loop closure detection system was shown to work in [20] where it found a set of five common landmarks between the first and last submaps in a large loop around Keble College. The results for our own system on this sequence were shown at the start of this chapter in Figure 5.1 on page 133.

5.6.2 Image–to–Image Matching: Cummins & Newman

Cummins & Newman [21] have developed a method to detect loop closures based on recognising the visual appearance of previously seen places using a bag of words based method. The matching is performed by detecting in each image the presence or absence of features from a visual vocabulary [72] based on SURF features [7]. This vocabulary is learned off-line from training data of generic images not collected in the environment where loop closure detection is performed.

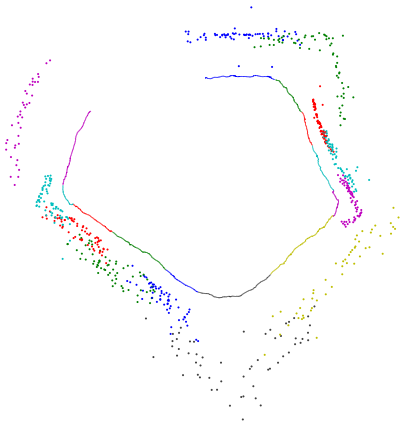
Cummins & Newman's system is made particularly robust to visual aliasing by modelling the joint distribution of word appearance. This is learned using the training data and is approximated using a Chow-Liu tree [18]. It allows the system to dismiss false loop closures that would have been hypothesised when observing things like brick walls. Though two different brick walls will tend to

give a large number of detections of the same visual words, the system knows that these visual words tended to co-occur in the training data so low probability would be given for a loop closure. For this trick to work, the training data must contain a representative set of images.

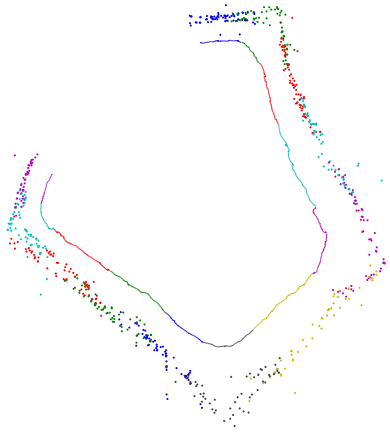
This method does not use the map created by the SLAM system to recognise loop closure events. It only determines the probability that two images show the same region of the world. This makes this method more suitable to pose based SLAM algorithms (rather than landmark based like we use here) since it indicates a correspondence between two camera poses without reference to the landmarks. Still, this loop closure detection system could be used to correct a metric map if the relative pose of the camera for corresponding images can be determined. This could be done using point correspondences between the images. However, in order to determine the relative scale difference, at least two pairs of images would be required. More information on the problem of determining the relative pose of the camera for two images can be found in [36]. Here though, we test the performance of Cummins & Newman's system by only testing the algorithm's ability to find the image–to–image loop closure hypotheses. We do not determine the relative camera poses needed to correct the map.

5.6.3 Comparison Results

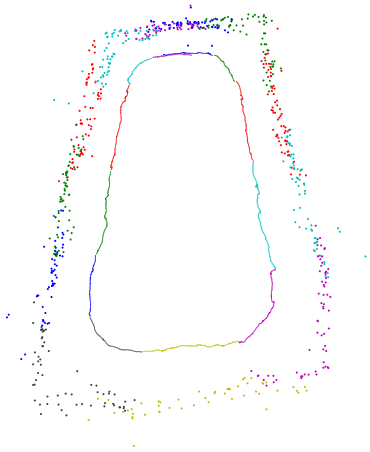
We have evaluated the performance of the three loop closure detection algorithms by checking their susceptibility to false positives and the processing time. To do this, we recorded two laps around Pembroke College courtyard. We first check



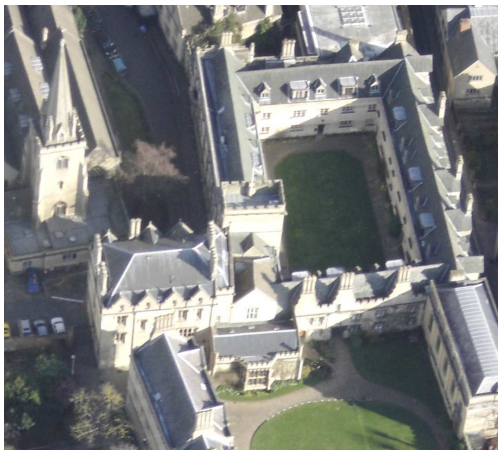
(a) Local maps obtained with pure monocular SLAM.



(b) Local maps auto-scaled.



(c) After loop closure.



(d) Aerial view of the courtyard.

Figure 5.9: **Pembroke College Sequence:** Twelve submaps with a total of 848 landmarks were made during the 70m trajectory around the courtyard. This sequence is used to compare our loop closure detection system with other existing methods.

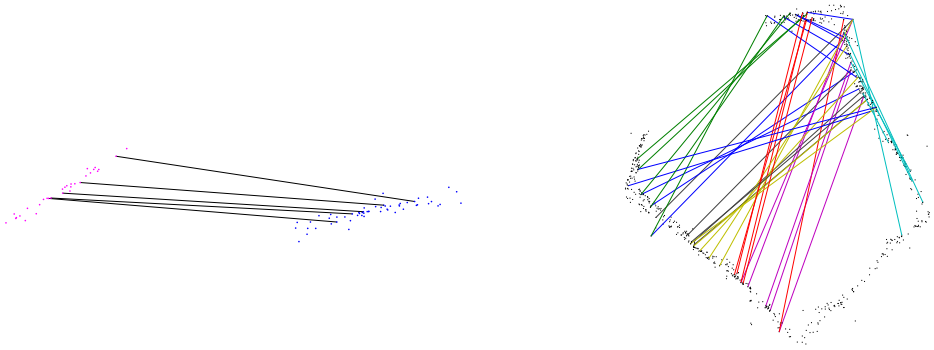
second lap.

We will first discuss the performance of the map–to–map and image–to–image method before moving on to discuss our image–to–map method. Figure 5.9 shows the map built by the system for this sequence.

Map–to–Map Matching: Clemente *et al.*

The map–to–map matching method checks each pair of submaps to determine if they are of the same region of the world. A match would indicate a loop closure. When the system compares the first and last submaps built, it is able to successfully find common landmarks between the two maps as shown in Figure 5.10a. Unfortunately, during the loop closure, there is no guarantee that the system will have initialised landmarks in exactly the same place in the two overlapping submaps. In fact, in our experiments to date, we have found submaps with sufficient common landmarks to detect the loop closure in this way to be rare. In a different run on this sequence, there were not enough common landmarks initialised to allow the system to detect the loop closure. Figure 5.11 shows the same frame being tracked in two different submaps. Despite the large number of landmarks visible, only two landmarks are common to both maps. This is not enough to determine the transformation between the submaps and so the loop closure is missed.

Even getting a corresponding set of landmarks does not guarantee a true correspondence between the two submaps. Figure 5.10b shows that the GCBB algo-



(a) **Map–To–Map Matching:** Loop closure detected using the method of Clemente *et al.* [20]. The system finds a set of five landmarks consistent in both geometry and appearance between the first and last submaps. It is only successful if the SLAM system has initialised common landmarks in the two submaps. In another run of the same sequence, the loop closure was missed (Figure 5.11)

(b) **Map–to–Map Reliability:** Matching was attempted between every pair of non-consecutive submaps. Shown here are the eight false positives sets with five correspondences. We were unable to find a threshold able to reliably distinguish between true positives and false positives.

Figure 5.10: The performance of the map–to–map method in the Pembroke College sequence.

rithm also found sets of five "common" landmarks between eight other pairs of submaps. We were unable to find a threshold able to reliably distinguish between true positives and false positives for the maps created by our SLAM system.

To test the reliability, every submap in the first lap was compared to every submap in the second lap. The precision–recall curve in Figure 5.15 on page 179 shows the result of changing the correspondence set size threshold. Note that the sample size for this method is small compared to the others tested. With only twelve submaps built for each lap, there are potentially only twelve true positive correspondences.

The map-to-map matching technique is quite fast. During our tests, the variable

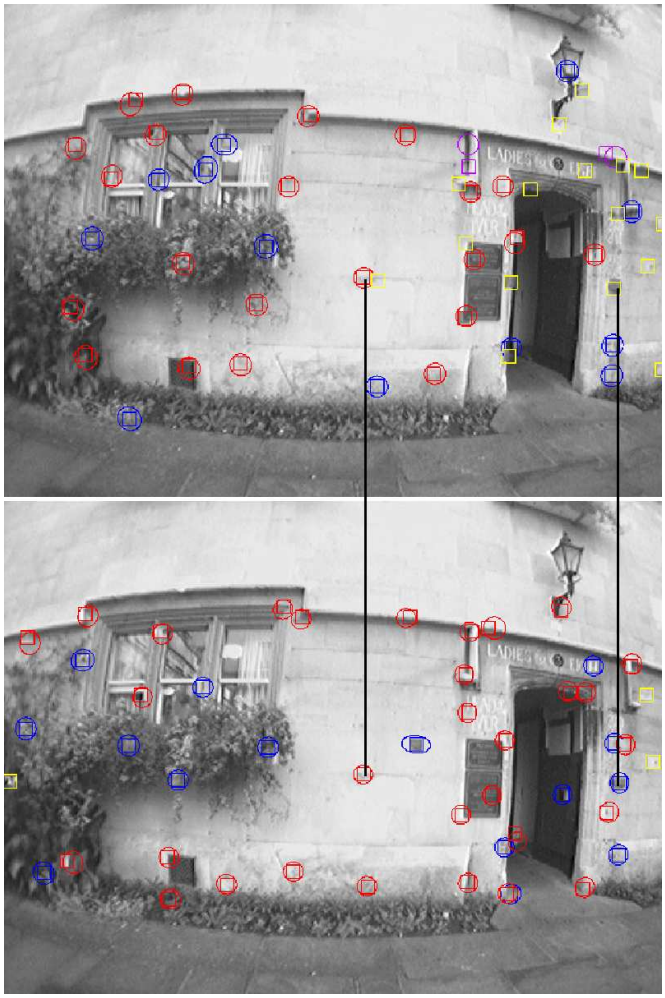


Figure 5.11: **Map-to-map Failure Mode:** False negative in the Pembroke College sequence. Shown are landmarks in first and last submap for the same frame. Only two of the landmarks are actually common to both submaps making it impossible for the map–to–map method to detect the loop close. Two maps made of the same part of the world do not necessarily have any common landmarks.

scale GCBB algorithm took around 100ms to compare two submaps[2]. When the SLAM system finishes one submap, there is easily time to compare this submap to all previous submaps before the next one is completed.

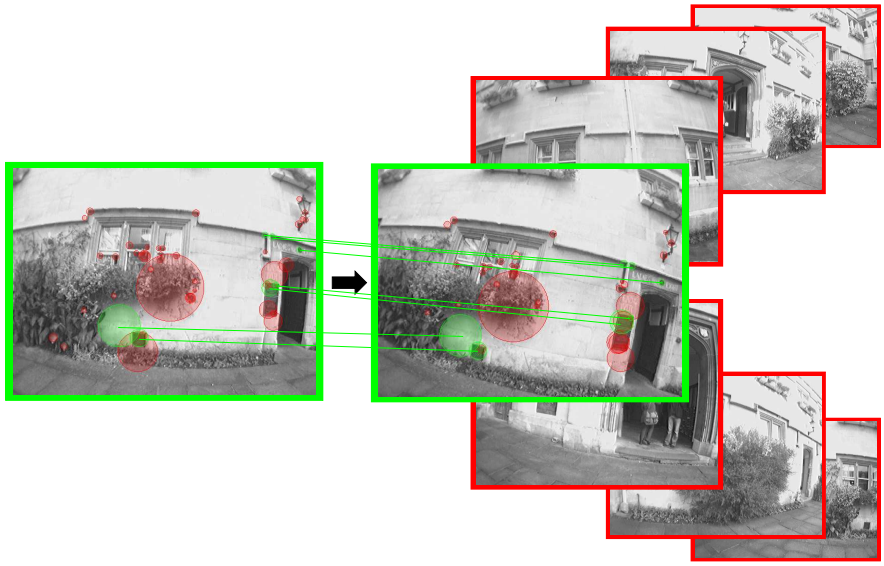
Image–to–Image Matching: Cummins & Newman

The image–to–image matching method of Cummins & Newman is designed to work with non-overlapping key frames. When run on a robot, the odometry is used to trigger key frame capture. Without odometry, we simply used every 40th frame of the video to test the system. Ideally though, some form of automatic key frame selection should be used.

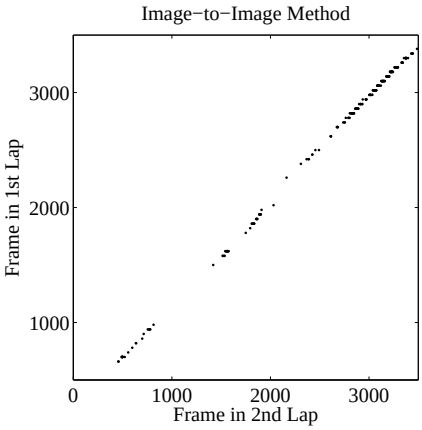
The loop closure detection system inspects each of these input images and determines if it is of a new place or is of a previously seen place. On the Pembroke College sequence, the algorithm correctly gave high probability that each image was a new place until the camera had traversed the loop and returned to the start of the courtyard. At this point, the system gave high probability (99.9%) that the most recent image corresponded to a frame at the start of the sequence (Figure 5.12a).

To further test the reliability of Cummins & Newman's loop closure detection algorithm, the images from the second lap of the courtyard were used. We computed the match probability for every frame of the second lap, against the set of keyframes from the first lap. This simulates the 'kidnapped robot situation', a

[2]Tests were done on a Dual Core 3GHz machine.



(a) **Image–To–Image Matching:** Loop closure detected using the method of Cummins *et al.* [21] in the Pembroke College sequence. The system detects visual words in each image and the co-occurrence of these words is used to calculate the probability of loop closure. The system finds a high probability that the most recent image matches one seen earlier in the sequence. Visual words are detected in the two images are indicated in green if they match in the other image. Note that interest point geometry is not considered.



(b) **Image–To–Image Reliability:** A second lap of the Pembroke College sequence was used to test the reliability of the image–to–image method. Correspondences were found between every frame in a second lap and every 40th frame in the first lap. With a threshold chosen to remove all false positives the system was successful in 8% of attempts. Gaps are in regions of the world with lots of foliage (where the image–to–map method also struggles).

Figure 5.12: The performance of the image–to–image method in the Pembroke College sequence.

sudden transition from the end of the first loop to a random part of the courtyard. This tests if the algorithm would be able to detect a loop closure at each position in the trajectory. The results are shown in Figure 5.12b where frames that matched an image in the previous loop are marked. A threshold was chosen that removes all false positives to allow comparison with our image–to–map method. The system found matches that met this probability threshold in 8% of attempts indicating that the system would be able to close the loop at these positions. The precision–recall curve in Figure 5.15 on page 179 shows the effect of the probability threshold on the reliability of the system.

On each image, the algorithm takes on average 283ms to run. Much of this time (73ms) is taken up by SURF feature detection. This method relies on this descriptor which is richer yet slower than the randomised lists classifier. The overall speed is slower than the framerate. However, the loop closing algorithm does not need to be run on every frame.

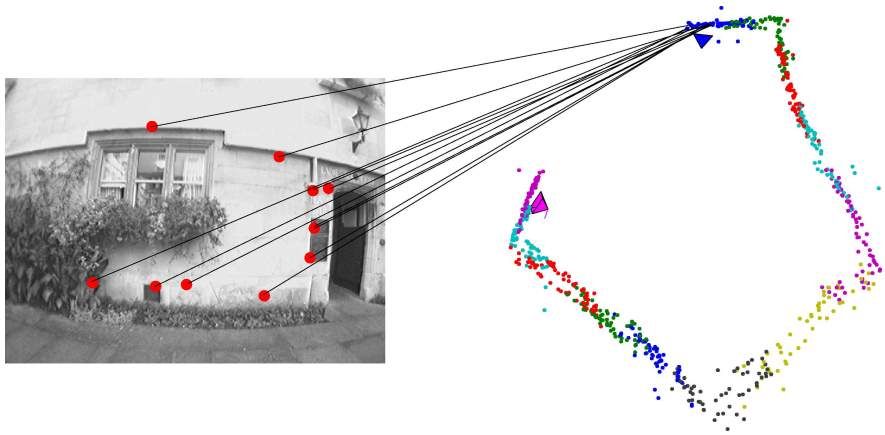
Since these tests were completed, Cummins & Newman have improved their algorithm. The latest version, called FAB-MAP 2.0 [22], includes an essential matrix based geometric test to eliminate false positives like those shown in Figure 5.12b. This new algorithm has been shown to perform very well in trajectories up to 1000km in length. On the Pembroke College sequence used here, it would achieve as good a performance as our own method.

Image–to–Map Matching: Our Method

For each frame, there is usually enough remaining time after tracking to attempt relocalisation in one other submap. The system cycles through submaps until a relocalisation is successful, indicating a loop closure. For the Pembroke College sequence, the system successfully detected the loop closure as the landmarks in the original map came back into view (Figure 5.13a). The alignment between the submaps can then be determined using the common camera trajectory relative to each submap. Note that for this method, no common landmarks are needed between submaps as they are for the map–to–map method.

The reliability of our loop closure detection method was tested using the same 'kidnapped robot' situation we used to test the image-to-image method. The system was allowed to continue searching for loop closures as the camera continued around the courtyard for a second lap. For the test, the system attempts relocalisation in every submap for every frame. The results of this test can be seen in Figure 5.13b. With all parameters set to their default values, the system recognised 20% of the true positive loop closures with no false positives. The precision-recall curve produced by varying the successful match fraction required after relocalisation can be seen in Figure 5.15. One of the false negatives is analysed in Figure 5.14

The method takes 10-15ms to find potential matches to map landmarks in each image. The remaining time is used to run RANSAC on the matches to determine the pose. This is usually found within a few milliseconds if a valid pose exists for those matches. This is fast enough to allow the algorithm to run on a single



(a) **Image–To–Map Matching:** Loop closure detected using the method presented in this chapter . While tracking in the last submap, the system finds a camera pose consistent with the landmarks in the first submap. The common trajectory is used to determine the relative rotation translation and scale needed to align the submaps.

(b) **Image–to–Map Reliability:** Relocalisation was attempted on every frame of a second lap. The light dots show the camera poses recovered relative to the map and trajectory created on the first lap (black). This indicates that loop close would be successful for these frames. Successful in 20% of frames. No false positives.

Figure 5.13: The performance of the image–to–map method in the Pembroke College sequence.

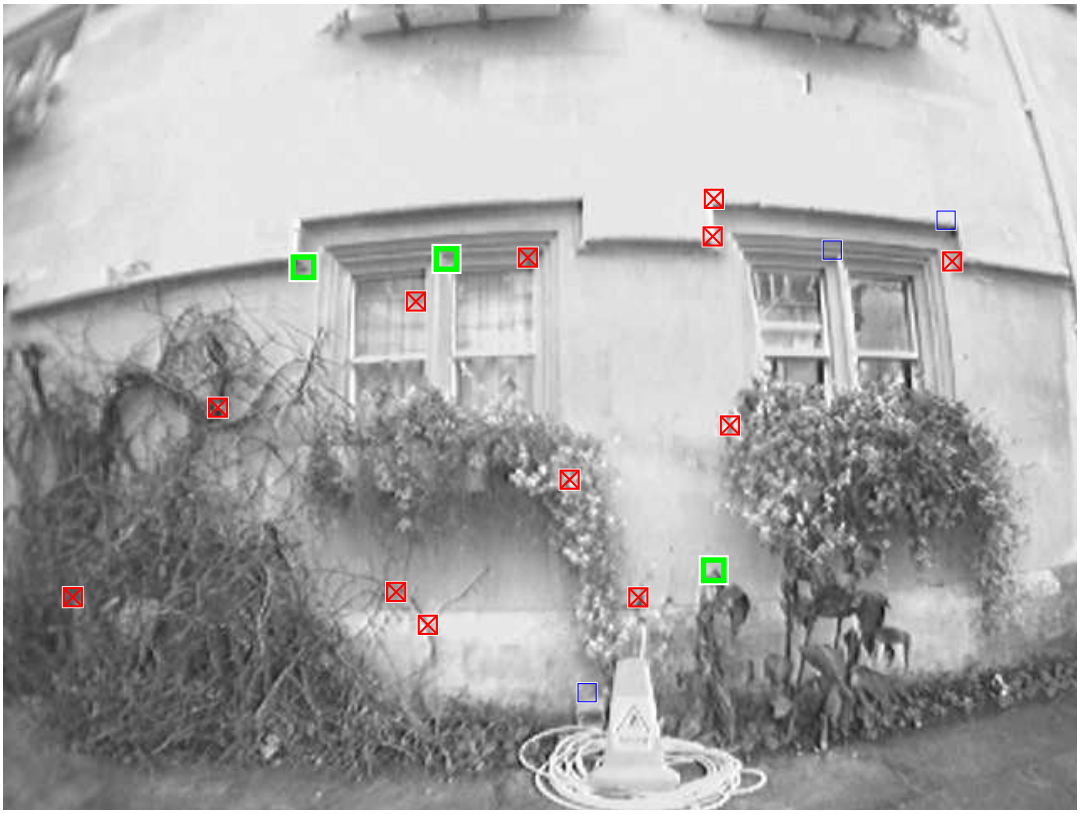


Figure 5.14: **Image-to-map Failure Modes:** Shown here is a false negative loop closure detection by our image-to-map method. Loop closure detection failed because only three of the visible landmarks were correctly matched (thick □). Three matches are insufficient to relocalise the camera. Of the remaining landmarks, some were not found because an interest point was not detected near true image location (□). For others, the interest point was not classified correctly (⊠). The image-to-map method will also fail if the map geometry is incorrect or if tracking fails before trajectory alignment is complete.

submap after the system has finished tracking in each frame.

5.6.4 Discussion

The results of our quantitative testing of the three methods using the second lap of the Pembroke College sequence are shown in Figure 5.15. The salient points that should be taken from this are:

- To create the curve for the image–to–map method, we varied the threshold on the fraction of successful landmark observations after a potential relocalisation. All other parameters were left at their default values. In practice, we require 50% of observations to succeed giving the 20% recall at 100% precision quoted above. Other possible threshold choices could include the corner detection threshold, the randomised lists score, and the number of consensus landmarks.

- The curve for the image–to–image method is more straightforward since a single threshold (the match probability) can be varied to achieve a larger range of recall. The performance is similar to the image-to-map method with only a few extra false positives detected with high probability.

- The curve for the map–to–map method has fewer points. The threshold varied here is the number of landmarks in the compatible set. The twelve submaps built in the second lap were matched to the twelve from the first lap. In this run, a single true positive was found with a seven common

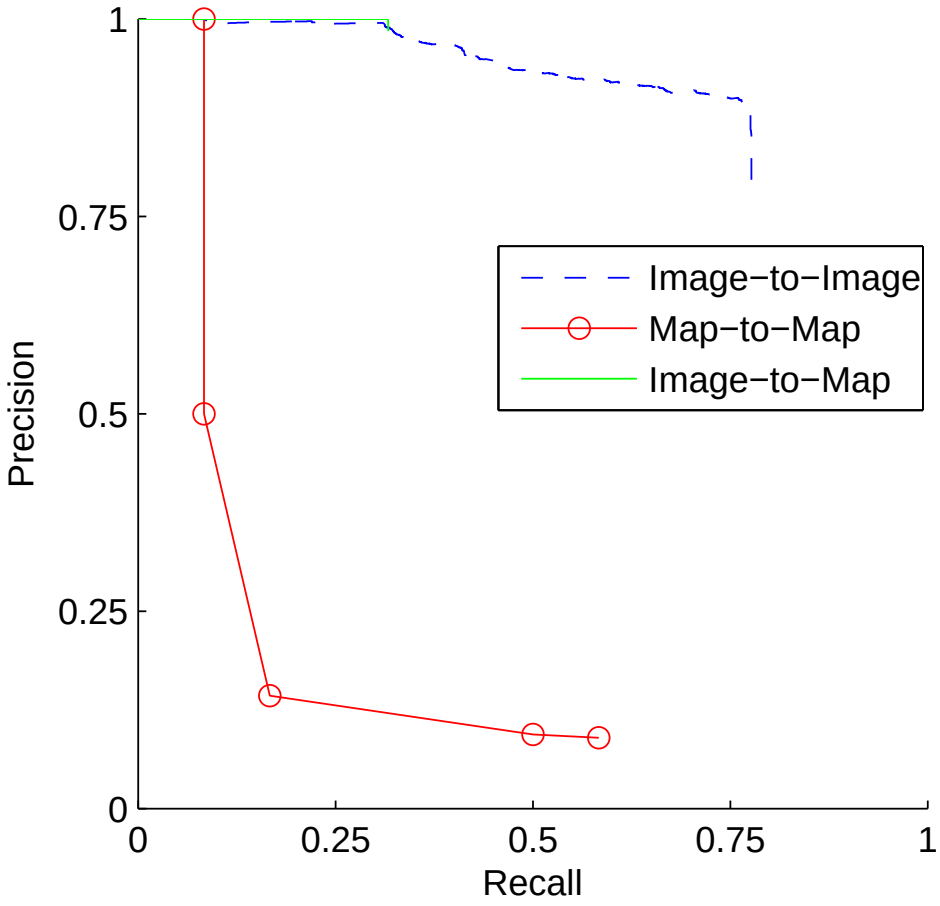


Figure 5.15: The performance of the three loop closure detection methods was assessed by testing the ability to recognise loop closure at each position of the second lap of the Pembroke College sequence. The map-to-map method suffers due to the lack of common landmarks in submaps built of the same region of the courtyard. The image-to-map and image-to-image method both perform very well but the image-to-image method has slightly more false positives. Many of these could be ruled out with geometric constraints in the visual word correspondences.

Figure 5.16: Despite these two regions of the world having both similar appearance and structure, they belong to different parts of Keble College courtyard. A loop closure detection system should indicate a match and a higher level system should then be used to mark it as false using either the uncertainties in the global metric map or more observations before and after the archway.

landmarks. For all lower thresholds many false positives were also found leading to the steep drop in precision.

We would not want the reader to infer too much from these numerical results. The actual numbers found would depend on the precise sequence used and the characteristics of the environment, but the general trends in the graph are representative of our overall findings. Of most importance are the relative performance, benefits, and failure modes of each algorithm. These aspects are now discussed more qualitatively.

We found the map–to–map matching technique of Clemente *et al.* is successful when a sufficient number of common landmarks exist between two submaps. For instance, it reliably detects the overlap between consecutive submaps where common landmarks are intentionally added to determine the relative scale. However,

in general, it is unsuitable for the sparse maps created by the monocular SLAM system. These maps are designed to be good enough to track the camera but otherwise as sparse as possible to allow faster updates. Perhaps a map–to–map based method would be more suitable if the maps contained higher level information or there were more consistency on which potential landmarks are added to the map.

The image–to–image matching technique of Cummins *et al.* works well since it can be tuned to remove all false positive while still detecting 8% of true positives for the Pembroke College sequence. As the probability threshold is lowered, the first few false positives could easily be removed with a simple essential matrix test for geometric compatibility. This can be seen for the false match with highest probability in Figure 5.17. With this test, the performance of the image–to–image method would have equivalent performance to the image–to–map method. On the other hand, the fact that the image–to–image method does not rely on a good metric map existing can also be an advantage. This makes the system more flexible and allows it to work in a greater variety of applications. Finally, the method scales well to very large environments and has been shown to work on datasets of 1000 kilometres [22].

The best performance here was found using our image–to–map matching technique. This method is able to prune more false positives than the image–to–image method by making use of the geometry information of the features detected in the image since these features have already been mapped by the SLAM system. It is fast enough to detect loop closure while tracking but it requires a large amount of memory for the randomised fern classifier (1.25 MB per map landmark). This

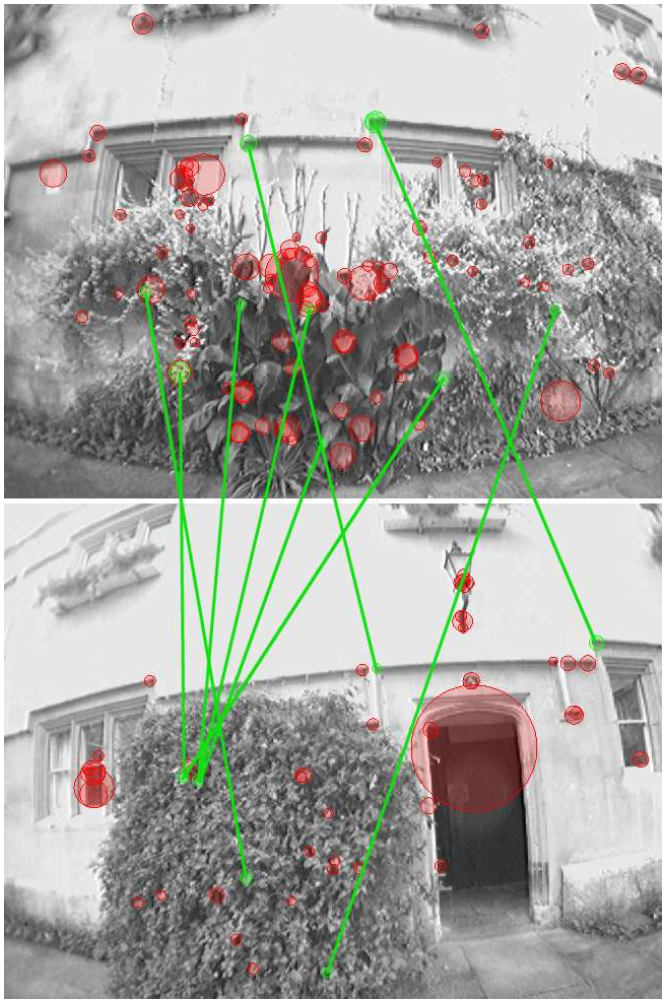


Figure 5.17: **Image-to-image Failure Mode:** False positive with matching probability of 99.9935% in the Pembroke College sequence. The detected visual words are indicated in each image in green if they match the other image. This false positive could easily be discarded if the geometric information were used for the detected visual words.

limits its use to environments not much larger than those considered here. It is this classifier though that the system relies on for fast training and recognition of the map landmarks in each image.

Perhaps even better performance could be achieved through a hybrid system combining the benefits of the image-to-image and the image-to-map methods. The loop closure detection system developed by Eade & Drummond [30] does just this. They first use a bag of visual words approach to establish which submap is in view. This stage is similar to the image-to-image method tested in this chapter. Then, local landmarks are identified in the image and the camera pose relative to the landmarks is determined in a similar way to the image-to-map method. This global to local approach harnesses the strengths of each method.

There also exist some truly ambiguous regions of the world that pose difficulty for a loop closure detection system. Figure 5.16 shows one such situation in the Keble College sequence. The two frames from the sequence show two different regions of the courtyard which are very similar in both appearance and geometry. The subtle differences could possibly indicate that this is not a loop closure situation, but in man made environments, two different regions of the world can be completely identical locally. In these situations, a higher level system must determine the validity of the loop closure hypothesis based on the uncertainty of the metric map or by waiting for further observations.

Method	Pros	Cons
Map–to–Map	Finds alignment when common landmarks exist.	Sufficient common landmarks are unlikely. False positives have similar number of 'common' landmarks.
Image–to–Image	Detects true loop closures throughout the environment when tuned for 100% precision. Does not require metric map. Scales well to large environments.	Offline learning of good vocabulary required. Does not make use of geometric information. Does not give metric transformation directly.
Image–to–Map	Detects true loop closures throughout the environment when tuned for 100% precision. Online training for map landmark appearance. Relative transformation between submaps with scale computed from trajectory.	Requires good metric map. Very memory intensive.

Table 5.2: A summary of the pros and cons of each of the three loop closure detection methods.

5.7 Conclusion

In this chapter an image–to–map loop closure detection system was presented. The system derives from the relocalisation system presented in Chapter 4. It detects loop closure by finding correspondences between the current image and the map landmarks in topologically distant parts of the map. This topological distance is determined using covisibility information for the map landmarks. The submapping system was shown to work in environments with multiple loops where a final global map was built taking into account all of the loop closures detected.

This system can also be used to detect an overlap between two independent maps. This problem is particularly important when using hand-held cameras where a fast rotation can cause the camera to be pointing at an unmapped region of the world where relocalisation would be impossible. The system instead starts an independent map which is later joined to the previous map when the overlap is determined. The map joining process creates a single correlated map covering both regions of the environment.

Our loop closure detection system was compared to two other techniques — the map–to–map method of Clemente *et al.* and the image–to–image method of Cummins & Newman. We found the map–to–map method to be unsuitable for the sparse maps used in monocular SLAM where there is no guarantee of common landmarks existing in two submaps. The image–to–image matching method worked well but was unable to reject some obvious false positives since geometry information was not used. By using both geometry and appearance information

while not relying on common landmarks, our loop closure system achieved the best performance. It was able to reliably detect loop closures throughout the test environment and was then able to correct the metric map using a common trajectory in the two coordinate frames.

6

Conclusion

To conclude, we will first give a summary of the material covered in this thesis. Then, the specific contributions made in the development of our system will be highlighted. Finally, we will outline possible directions for future work in this area.

6.1 Summary of the Thesis

In this thesis, we have presented a new monocular SLAM system which can track the pose of a hand-held camera in real-time while building a sparse map of the structure of the scene the camera is moving through. Our system addresses some

of the problems encountered in previous systems. We specifically address the issue of estimation consistency and accuracy, robustness to tracking failure, and the ability to recognise previously seen places when building large maps. Each of these contributions makes the final system more useful outside the lab for applications such as augmented reality and mobile robot navigation.

In Chapter 2, we reviewed the previous work done in this field. Specifically, we reviewed the different design choices made in the development of other monocular SLAM systems. These included different methods for representing the world seen by the system, how to recognise and track features from the world in the images recorded by the camera, and the different estimation algorithms available for jointly estimating the scene structure and camera trajectory. We also highlighted the algorithms used in our system which have been developed by others in the field.

In Chapter 3, we presented our own monocular SLAM system which aims to provide a more accurate and consistent estimate of the camera trajectory than previous approaches. This is achieved through the application of two new methods for monocular SLAM. The first is to utilise all potential point matches in each frame in a visual odometry style approach rather than just the observations of the landmarks in a sparse map. Since the depth of these extra features is not estimated, the observations only provide constraints on the inter-frame epipolar geometry. Crucially though, they do not increase the state estimate size which would be too computationally costly. The motion model used in the EKF estimation framework provides a prediction for the inter-frame motion of the camera and hence a prediction of the inter-frame epipolar geometry. The system measures the

distance of each match from the predicted epipolar line and these measurements are used as observations in the EKF to refine the predicted inter-frame motion estimate. Unfortunately, the extra computation required for these visual odometry style observations proved to be too large to maintain framerate performance in reasonably sized environments.

The second technique used to improve the quality of the estimate produced by our system is robocentric mapping. We have adapted this framework for use with a hand-held camera in monocular SLAM. The robocentric framework produces a more consistent estimate than the standard EKF algorithm which is known to underestimate the uncertainty of its estimate. It achieves this by representing all landmarks relative to the coordinate frame centred on the camera at each timestep. In this representation, the linearization errors which cause inconsistency in the EKF algorithm are reduced. The robocentric framework is also ideal for including the proposed visual odometry style observations of features not maintained in the state estimate. These observations constrain the inter-frame relative camera motion which is a key aspect of tracking in the robocentric framework. We show that the addition of these two techniques improves the accuracy and consistency of our system over traditional worldcentric systems observing only sparse map landmarks.

We then presented a method for detecting and recovering from tracking failures in Chapter 4. Tracking failures are inevitable when using a hand-held camera where a sudden motion or occlusion can cause all of the observations to fail. Our system detects when observations fail and halts normal tracking to prevent the map estimate from being corrupted by incorrect observations. The system then

searches for the map landmarks in each image using a fast landmark recognition method we have developed called randomised lists. This algorithm is a novel adaptation of the randomised trees technique well suited to monocular SLAM. This classifier is trained to recognise landmarks using real examples from different viewpoints harvested using the observations of the landmarks made as the camera moves around the scene. The result is a fast recognition system with very high recall rate which is key to relocalising against the sparse maps used in monocular SLAM. Once the landmarks have been detected in an image, RANSAC and a three-point-pose algorithm are used to relocalise the camera so the system can continue tracking. This ability to relocalise makes the overall system very robust. For the user, tracking failures are no longer a problem as the system can quickly recover after each one.

This relocalisation method was then applied to the problem of loop closure detection in Chapter 5. When mapping larger environments, it is important to recognise when the camera has returned to a previously seen region of the world. Our system can recognise when the landmarks from another region of the map are visible from their appearance and geometry. The system relocalises relative to this other part of the map and begins to track the camera relative to those landmarks. If the camera tracks successfully for long enough, the loop closure hypothesis is accepted. The map is then corrected after determining the relative transformation between the two map regions using the common trajectory. This method is used to build accurate maps of larger environments and to join two independent maps that overlap in the world. We show that our method performs very well compared to other loop closure detection methods developed for

monocular SLAM.

6.2 Contributions

The main contributions which have been presented in this thesis are as follows:

- A novel method for utilising a large number of observations to improve camera motion estimation accuracy while retaining just a sparse map to facilitate real-time drift-free operation.
- The reformulation of the robocentric mapping framework for use in monocular SLAM to achieve a more consistent estimate.
- A new landmark recognition algorithm developed specifically for monocular SLAM which learns the appearance of landmarks online so they can be quickly found again after tracking failure.
- A relocalisation system which can detect tracking failure, prevent map corruption, and then restart tracking after determining the camera pose using our landmark recognition system and the map geometry.
- A method for recognising loop closure events well suited to monocular SLAM which establishes correspondences between the image and the appearance and geometry of the map built by the system.
- A trajectory based map alignment technique which is able to cope with the scale ambiguity inherent in the monocular SLAM problem. The alignment

is used to correct the map estimate in large maps after a loop closure has been detected.

6.3 Future Work

Though the techniques presented in this thesis improve the capability of our monocular SLAM system compared to earlier systems, much work still remains to make these systems suitable for use outside the lab. In this section, we will discuss improvements that could be made to our own techniques as well as considering more general areas for future research.

Our relocalisation system relies on finding matches to the map landmarks in the image, but there is much more useful information within the image that is currently ignored in this search. Each landmark is identified based only on its local appearance. Since each landmark is initialised at a corner point, there is actually not huge scope for variation amongst the landmark appearance. Our current system succeeds by using the map covisibility information which indicates which sets of landmarks have been seen together. This provides information about the context in which the match was found and can be used to disambiguate multiple matches resulting from using local appearance alone. The rest of the image too contains valuable context information which could be used. In fact, Eade & Drummond [30] use such an approach. They first determine which local map region the camera is observing with a bag–of–words technique and then search only for the landmarks in that region to relocalise relative to the map. The

image–to–image matching system of Cummins & Newman offers highly scalable appearance based matching which if combined with metric relocalisation would lead to a very formidable relocalisation and loop closing system. There is also no need to ignore map-to-map matching information once higher level information is included in the maps.

The trajectory alignment technique we have presented also has room for improvement. We have demonstrated that using two pose correspondences relative to each map region can be used to determine the rotation, translation and scale needed to align these regions. However, the two poses used for determining the alignment should be selected with more care. Currently, our system chooses two camera poses separated in time. A better solution would be to select a set of two well localised poses for which the relative alignment transformation is well constrained. Also, a method to incorporate more of the trajectory so that the transformation can be determined with more accuracy without too large a computational cost would be beneficial.

The submapping method used in our system is unsatisfactory for vision based mapping. Partitioning the world into submaps simply by starting a new one when the current submap grows too large is not suitable for sensors like cameras which can make observations of both very distant and very near landmarks simultaneously. A more intelligent partitioning like the mutual information approach presented by Chli & Davison [17] should be used. Devising an efficient estimation framework to use such a partitioning remains for future work. Ideally, this framework would be able to share some landmarks between submaps to provide greater constraints on the global map structure as is done by Piniés & Tardós [59]

and Eade & Drummond [30].

Another important ability missing from our system is the ability to change decisions made about loop closure hypotheses. Currently, care is taken to ensure that there is a very low probability of false positives. It would be far better however if a false loop closure incorporated into the map could be detected and removed when it becomes apparent from new observations. This ability to change decisions on data association would also be useful for ordinary observations as well. Despite all our checks, bad measurements can still go undetected by the system. Sequential methods continually bake in each new set of observations making decisions virtually impossible to undo. However, using techniques like a sliding window, data association decisions can be flexible in even in sequential systems as demonstrated by Bibby & Reid [8]. For flexibility in this way though, batch algorithms seem more promising. Systems like Klein & Murray [40] and Eade & Drummond [30] continually optimise the global map using bundle adjustment. Changes to the global map structure caused by events like loop closure are handled in a way which would make removing such a constraint much more straightforward.

Batch methods like the system of Klein & Murray [40] also allow a very large number of landmark observations to be made per frame which produces very accurate tracking. Unlike our system, all of these features are fully estimated in 3D leading to much denser maps than we can produce. Unfortunately, this system is limited to mapping small environments due to a limit on the number of keyframes which can be included. A promising direction for future work is the extension of this system through some sort of submapping approach.

Finally, one of the most obvious areas for improvement, is the representation of the world used in our system. Point landmarks are excellent for tracking the camera pose but they provide little information about the scene structure. Though lines [27, 41, 74] and planes [13] have been used in other monocular SLAM systems, the implementations produced so far have failed to produce a significantly better representation of the world. Dense visual reconstruction techniques provide detailed representations but are usually not performed online from a freely moving camera. For robotics, it is essential to understand the full 3D extent of objects in the world so obstacles can be avoided. In augmented reality, a world representation beyond simple points is required so that the virtual objects added to a scene can appear to be sitting on real surfaces and be occluded by real objects.

Though point-based systems do not provide this dense map representation themselves, they can provide the crucial camera trajectory estimate from which to build the higher level world representation. Such a dense mapping system must still be able to cope with the challenges posed by a freely moving hand-held camera. These conditions are very different from the carefully calibrated conditions typically used in visual reconstruction. Also, for a real-time system, the framework used should allow the representation to be incrementally improved as the camera views the scene at different scales and from different viewpoints.

If a monocular SLAM system is to be used over an extended period of time, then a more sophisticated map management scheme than the one used in our system should be used. Landmarks are only removed from our map when they have been unsuccessfully measured in 50% of attempts. An object moved in the world can

take some time before it is corrected in the map. Our system also does not handle occlusions of landmarks well. If a landmark is often occluded then it is likely to simply be removed from the map. If the world were represented more densely, then these occlusions would be expected and the system would not attempt to observe an occluded landmark. Moving objects too are not modelled by the system. Currently, all landmarks are assumed to be static so any landmarks placed on a moving object will be marked as outliers by JCBB. This prevents the estimate being corrupted by these observations, but it would be better if the system understood that the object is moving and that it will occlude parts of the scene. Some systems have already demonstrated the ability to incorporate observations of moving objects within a SLAM context [8].

In general, a higher level understanding of the scene would improve many aspects of the system. This higher level understanding can be achieved by combining a visual SLAM system with other systems being developed in the field of computer vision. For example, the ability to recognise certain objects like a parked car will help such a system explain the car's disappearance from the map at a later time. Recognising people in an image is a good way to identify at least one particular class of dynamic object. Also, systems in the field of semantic mapping are able to classify and label different regions in the world such as grass or buildings [61]. This ability to understand the world seen by the camera will continue to improve.

The ultimate aim is a system which can understand what it sees as it moves around its environment. Such a system, though interesting for its own sake, will also be useful in so many applications such as robotics and augmented reality. There is still much work to be done before we develop intelligent systems which

can truly understand the world they see but each piece of research brings us closer to this exciting goal.

Bibliography

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7), 1997.

[2] A. Angeli, D. Filliat, and S. Doncieux. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics, Special issue on Visual SLAM*, 24, 2008.

[3] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM algorithm. In *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2006.

[4] T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In *Proc. International Conference on Robotics and Automation*, 2006.

[5] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3), 2004.

[6] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley and Sons, 2001.

[7] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proc. European Conference on Computer Vision*, 2006.

[8] C. Bibby and I. Reid. Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. In *Proceedings of Robotics Science and Systems*, 2007.

[9] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *Proc. International Conference on Robotics and Automation*, 2003.

[10] M. Calonder, V. Lepetit, and P. Fua. Keypoint signatures for fast learning and recognition. In *Proc. European Conference on Computer Vision*, 2008.

[11] J. A. Castellanos, R. Martínez-Cantín, J. D. Tardós, and J. Neira. Robocentric map joining: Improving the consistency of EKF-SLAM. *Robotics and Autonomous Systems*, 55(1), 2007.

[12] R. Castle, G. Klein, and D. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *Proc. 12th IEEE Int. Symp. on Wearable Computers*, 2008.

[13] D. Chekhlov, A. Gee, A. Calway, and W. Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual SLAM. 2007.

[14] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and A. Calway. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *Proc. 2nd International Symposium on Visual Computing*, 2006.

[15] M. Chli. *Applying Information Theory to Efficient SLAM*. PhD thesis, Imperial College, 2009.

[16] M. Chli and A. J. Davison. Active Matching for visual tracking. *Robotics and Autonomous Systems*, 57(12), 2009. Inside Data Association.

[17] M. Chli and A. J. Davison. Automatically and efficiently inferring the hierarchical structure of visual maps. In *Proc. International Conference on Robotics and Automation*, 2009.

[18] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3), 1968.

[19] J. Civera, O. Grasa, A. J. Davison, and J. Montiel. 1-point RANSAC for EKF-based structure from motion. In *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2009.

[20] L. Clemente, A. Davison, I. Reid, J. Neira, and J. D. Tardós. Mapping large loops with a single hand-held camera. In *Proc. Robotics Science and Systems*, 2007.

[21] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6), 2008.

[22] M. Cummins and P. Newman. Highly scalable appearance-only SLAM – FAB-MAP 2.0. In *Proc. Robotics Science and Systems*, 2009.

[23] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. IEEE International Conference on Computer Vision*, 2003.

[24] A. J. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2007.

[25] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. International Conference on Robotics and Automation*, 1999.

[26] C. Douse. Moving reality. *Newsline*, 31, 2004.

[27] E. Eade and T. Drummond. Edge landmarks in monocular SLAM. In *Proc. British Machine Vision Conference*, 2006.

[28] E. Eade and T. Drummond. Scalable monocular SLAM. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[29] E. Eade and T. Drummond. Monocular SLAM as a graph of coalesced observations. In *Proc. IEEE International Conference on Computer Vision*, 2007.

[30] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular SLAM. In *Proc. British Machine Vision Conference*, 2008.

[31] C. Estrada, J. Neira, and J. D. Tardós. Hierarchical SLAM: Real-time accurate mapping of large environments. *Transactions on Robotics*, 1(4), 2005.

[32] M. A. Fischler and R. C. Bolles. RANdom SAmple Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981.

[33] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1999.

[34] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Review and analysis of solutions of the three point perspective problem. *International Journal of Computer Vision*, 13(3), 1994.

[35] C. J. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conference*, 1988.

[36] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[37] K. L. Ho and P. Newman. Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74(3), 2007.

[38] S. Holmes, G. Klein, and D. Murray. A square root unscented kalman filter for visual monoSLAM. In *Proc. International Conference on Robotics and Automation*, 2008.

[39] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society*, 4(4), 1987.

[40] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. IEEE and ACM International Symp. Mixed and Augmented Reality*, 2007.

[41] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *Proc. European Conference on Computer Vision*, 2008.

[42] J. Košecká, F. Li, and X. Yang. Global localization and relative positioning based on scale-invariant keypoints. *Robotics and Autonomous Systems*, 52(1), 2005.

[43] K. Kraus. *Photogrammetry*. Dümmler, Bonn, Germany, 1997. Vol.1: Fundamentals and Standard Processes. Vol.2: Advanced Methods and Applications.

[44] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9), 2006.

[45] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: an accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2), 2008.

[46] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[47] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3), 2004.

[48] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. European Conference on Computer Vision*, 2002.

[49] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. European Conference on Computer Vision*, 2002.

[50] N. Molton, A. J. Davison, and I. Reid. Locally planar patch features for real-time structure from motion. In *Proc. 15th British Machine Vision Conference*, 2004.

[51] J. M. M. Montiel, J. Civera, and A. J. Davison. Unified inverse depth parametrization for monocular SLAM. In *Proc. Robotics Science and Systems*, 2006.

[52] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. In *IEEE Transactions on Robotics and Automation*, 2001.

[53] J. Neira, J. D. Tardós, and J. A. Castellanos. Linear time vehicle relocation in SLAM. In *Proc. International Conference on Robotics and Automation*, 2003.

[54] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(6), 2004.

[55] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1), 2006.

[56] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proc. IEEE Conference on Computing Vision and Pattern Recognition*, 2007.

[57] M. Ozuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. In *Proc. European Conference on Computer Vision*, 2006.

[58] L. M. Paz, J. Guivant, J. D. Tardós, and J. Neira. Data association in O(n) divide and conquer SLAM. In *Robotics Science and Systems*, 2007.

[59] P. Piniés and J. D. Tardós. Large scale SLAM building conditionally independent local maps: Application to monocular vision. *IEEE Transactions on Robotics*, 24(5), 2008.

[60] M. Pollefeys, R. Koch, and L. J. van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. IEEE International Conference on Computer Vision*, 1998.

[61] I. Posner, D. Schröter, and P. Newman. Online generation of scene descriptions in urban environments. *Robotics and Autonomous Systems*, 56(11), 2008.

[62] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[63] M. Pupilli and A. Calway. Real-time camera tracking using a particle filter. In *Proc. British Machine Vision Conference*, 2005.

[64] E. Ranganathan, E. Menegatti, and F. Deleart. Bayesian inference in the space of topological maps. *IEEE Transactions on Robotics and Automation*, 22(1), 2006.

[65] G. Reitmayr and T. Drummond. Going out: Robust model-based tracking for outdoor augmented reality. In *Proc. IEEE International Symposium on Mixed and Augmented Reality*, 2006.

[66] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proc. IEEE International Conference on Computer Vision*, 2005.

[67] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, 2006.

[68] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. European Conference on Computer Vision*, 2006.

[69] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 2001.

[70] S. Se, D. Lowe, and J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3), 2005.

[71] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[72] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proc. IEEE International Conference on Computer Vision*, 2003.

[73] G. Slabaugh, B. Culbertson, T. Malzbender, M. Stevens, and R. Schafer. A survey of methods for volumetric scene reconstruction. *International Journal of Computer Vision*, 2003.

[74] P. Smith, I. Reid, and A. J. Davison. Real-time monocular SLAM with straight lines. In *Proc. 17th British Machine Vision Conference*, 2006.

[75] R. C. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *4th International Symposium on Robotics Research*, 1987.

[76] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60, 2006.

[77] J. D. Tardós, J. Neira, P. Newman, and J. Leonard. Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research*, 21(4), 2002.

[78] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*, 2004.

[79] P. Torr, A. W. Fitzgibbon, and A. Zisserman. Maintaining multiple motion hypotheses over many views to recover matching and structure. In *Proc. IEEE International Conference on Computer Vision*, 1998.

[80] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In *Vision Algorithms: Theory and Practice*, LNCS. Springer Verlag, 2000.

[81] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4), 1987.

[82] S. Tully, G. Kantor, H. Choset, and F. Werner. A multi-hypothesis topological SLAM approach for loop closing on edge-ordered graphs. In *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2009.

[83] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Proc. International Conference on Robotics and Automation*, 2000.

[84] R. Van der Merwe and E. Wan. The square-root unscented kalman filter for state and parameter estimation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2001.

[85] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12), 2009.

[86] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. D. Tardós. A comparison of loop closing techniques in monocular SLAM. In *Inside Data Association Workshop, Robotics Science and Systems*, 2008.

[87] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. D. Tardós. An image–to–map loop closing method for monocular SLAM. In *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2008.

[88] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In *Proc. International Conference on Computer Vision*, 2007.

[89] B. Williams and I. Reid. On combining visual SLAM and visual odometry. In *Proc. International Conference on Robotics and Automation*, 2010.

[90] B. Williams, P. Smith, and I. Reid. Automatic relocalisation for a single-camera simultaneous localisation and mapping system. In *Proc. International Conference on Robotics and Automation*, 2007.