UNIVERSITY OF
CAMBRIDGE
Department of Engineering

Monocular Simultaneous Localisation and Mapping

Ethan Eade
Trinity College

A thesis submitted for the degree of
Doctor of Philosophy
September 2008

Declaration

This dissertation is submitted to the University of Cambridge in partial fulfilment for the degree of Doctor of Philosophy. It is an account of work undertaken at the Department of Engineering between October 2004 and September 2008 under the supervision of Dr T.W. Drummond. It is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text. This dissertation is approximately 50,000 words in length and contains 38 figures.

Ethan Eade

Acknowledgements

I thank my supervisor, Dr. Tom Drummond, for his guidance, and all of my colleagues for their good company and intellectual generosity. I am especially indebted to Dr. Gerhard Reitmayr for thought-provoking and insightful discussions about this work.

The Marshall Commission, the National Science Foundation, and the Boeing Company have funded my research, and I am grateful for their support.

Lee and Danielle Bassett, Nikhil Vellodi, Hugh Warrington, and Helena Bushman have not only tolerated me, but have made my life in Cambridge far richer than it could otherwise be. For that they have my continued gratitude.

Most of all, I thank my parents for their constant support and encouragement.

Abstract

Simultaneous localisation and mapping is the task of estimating from sensor observations both motion and structure in an unknown environment. Performing SLAM with a single video camera, while an attractive prospect, adds its own particular difficulties to the already considerable general challenges of the problem. This thesis advances the state of the art in monocular SLAM in terms of efficiency, richness of scene description, statistical correctness, and robustness.

First, a SLAM algorithm from the robotics literature, designed to permit efficient operation with complex maps, is adapted to the monocular setting. A method for efficiently and correctly adding landmarks to the map is presented. The implemented SLAM system accurately maps thousands of landmarks in real time, giving an order-of-magnitude performance improvement over previous methods.

Next, the system is extended to allow incorporation of edge landmarks as well as points. Edgelet landmarks and their representation are defined, and a method is described for reliably tracking edgelets, even in the presence of measurement ambiguity. An efficient selection algorithm for acquiring new edgelets from video allows the system to quickly extend the map. The working system produces geometrically accurate and meaningful edge maps at frame rate.

With a focus on preserving statistical consistency during estimation, a novel monocular SLAM algorithm is presented. Estimation proceeds on a graph of local maps, partitioning and coalescing the observations taken from video. Careful parameterisation keeps local maps consistent, while optimisation of the connecting graph structure aids global convergence. The system can handle thousands of landmarks at frame rate, while delivering statistical performance superior to existing methods.

Finally, this thesis mitigates the problems of tracking failure and large-scale localisation with a unified framework for loop closing and recovery. A hierarchical method is presented for finding correspondences between new video images and the existing map, using local and global appearance models and structure estimates. The framework is instantiated within the graph-based monocular SLAM system. The extended implementation continues mapping despite repeated tracking failures, successfully joining maps and closing loops in real time.

Contents

List of Figures

1

Introduction

1.1 Simultaneous Localisation and Mapping

Simultaneous localisation and mapping (SLAM) is the process by which a mobile entity answers two primary questions: "Where am I?" and "What is the structure of my environment?". The entity, which might be a robot, a vehicle, or a human, requires the answers continuously, as navigation or other decisions depend upon them. The entity's sensors provide the information from which a SLAM algorithm produces these on-the-fly results. The nature of the sensors and results varies with the entity and the algorithm.

Humans have been successfully practising SLAM for some time now. We observe the environment through vision, sound, balance, touch, smell, and taste. Our brain combines these observations, and – exploiting significant assumptions and prior knowledge – reports our location and the nature of the surroundings, both quantitatively

and qualitatively. For instance, the localisation question might be answered "at home" or "riding downhill on a bicycle at about 20 mph". The mapping result might be an abstract conception of a floor plan, or a set of topological relationships between places of interest. The process is usually unconscious. Occasionally, however, fusing all of the evidence into a best guess, while accommodating uncertainty, requires conscious effort.

For synthetic systems, the questions and the algorithm of SLAM must be specified explicitly. For the task of robotic or vehicular navigation, the most useful localisation and mapping output is geometric in nature. Pose and structure estimates are represented in specific coordinate systems and parameterisations. The uncertainty in these estimates must be explicit and readily available to the system, to aid decision making.

Accurate, reliable, and efficient SLAM is crucial for meaningful autonomy in unknown surroundings. Even if the SLAM platform is passive, merely following along with a human or robot, it gives the active entity a richer knowledge of its motion and the environment. The accrued map is also useful in itself, as a persistent description of the explored territory.

1.2 Monocular SLAM

A variety of sensors have been employed for SLAM with robots and vehicles. Laser range finders or sonar give both range and bearing measurements for a field of points in the environment. Much of the work in robotic SLAM assumes that such 3D geometric measurements are available, in addition to mechanical odometry readings from the robot. However, the equipment is costly, bulky, and resource-hungry. Furthermore, *data association* of one set of measurements from the sensor with another can be difficult, as only the perceived geometry can be used to distinguish parts of the scene.

A video camera addresses these concerns, but introduces some of its own. Cameras are lightweight, low-power devices that provide rich data at a relatively high sampling

rate. Image information is highly distinctive, and the extensive research on extracting it in useful forms can be brought to bear. However, the images captured from a camera lack the 3D information provided by laser range finders. Each pixel is a measure of light intensity or frequency along a ray, but the distance along the ray to an object in the world is unknown to the camera. This can be somewhat mitigated with a stereo camera rig, but then the advantages of low-power and compactness are also lessened. This thesis presents contributions in the realm of SLAM with a single video camera as the only sensor – *monocular* SLAM. A review of important work in structure from motion and SLAM with vision is given in Section 2.2.

1.3 Efficient SLAM

In one popular and well-tested framework for representing structure, a SLAM system selects sparse, distinct *landmarks* in the environment, and estimates their parameters using sensor data. Landmarks might be geometric features (when using a range finder) or visual features (when using a camera). As the number of landmarks estimated by the system increases, so does the computational effort required in order to fuse new sensor measurements into the state. In the case of the commonly used Extended Kalman Filter framework for SLAM, the cost grows quadratically with the number of landmarks.

Extensive research, especially in the robotics literature, has focused on making the estimation more efficient when dealing with complex maps with many landmarks. Section 2.3 reviews some of the salient approaches. In particular, the FastSLAM particle filtering method allows state updates with linear or sublinear computational cost.

In Chapter 4 of this thesis, I show how FastSLAM can be applied to monocular SLAM, allowing for frame-rate mapping of thousands of 3D point landmarks. The application is nontrivial because of the difficulties associated with a bearing only sensor. In particular, the partial observation model requires special care to be taken when adding

new landmarks to the map. Chapter 4 also shows that top-down active search for locally planar textured patches can be fruitfully employed in the particle filter setting, making image processing efficient. The resulting system gives accurate trajectory and structure estimates for small scenes with many landmarks, all while running at frame rate.

1.4 Edge Landmarks in Monocular SLAM

Existing work in vision SLAM has been motivated both by the robotics SLAM literature and the computer vision research in structure from motion (see Section 2.2.1). Thus 3D points, which are tracked as small regions in images, are a natural choice for landmarks. However, much more information can be gleaned from the video.

In particular, the edge information in an image reflects edge structures in the world. Straight edges are useful, well-studied ways of representing certain classes of structure, often encountered in indoor environments. Edge models have been variously and successfully employed in model-based tracking, and representing and estimating edges within monocular SLAM would yield a richer map with higher-level geometric primitives.

Chapter 5 describes how edge landmarks can be incorporated into monocular SLAM. I introduce the *edgelet*, with a landmark representation and appearance model that are the edge analogues to 3D points and locally planar textured patches. Edgelets are short, straight pieces of 1D structure that appear as intensity discontinuities in images. Chapter 5 shows how they can be efficiently acquired from video and tracked using top-down search. The method is implemented within the framework of Chapter 4, and the resulting maps model structure not captured by point landmarks.

1.5 Graph SLAM

Though the particle filter SLAM system described in Chapter 4 achieves accurate localisation and efficient operation with many landmarks, it is unable to produce coherent maps over large trajectories. The cause of the problem is a lack of *consistency* in the estimate of the filter. The uncertainty in the state is underestimated, preventing the filter from converging to the true result.

In Chapter 6, I discuss the causes of this inconsistency and present a new SLAM framework, Graph SLAM, which maintains independent local maps connected in a graph structure. The design of the framework is motivated by consistency, but efficient operation with many landmarks is an important secondary benefit. The estimation process is factored into updates to local maps and refinements of the inter-map graph structure.

The method is inspired by earlier submapping strategies (cf. Section 2.3.3), but pays special attention to the nonlinear observation model of monocular SLAM. The transformations between local coordinate frames are determined from shared landmarks' estimates in the nodes, so edge estimates improve with nearby local maps. Further, cycle constraints in the graph are accommodated by iterative global graph optimisation.

The method is implemented, and its consistency is tested using Monte-Carlo simulation. The results show that it is qualitatively more consistent than FastSLAM or Extended Kalman Filter SLAM, while maintaining frame-rate operation with thousands of landmarks.

1.6 Loop Closing and Recovery

Tracking-based vision SLAM systems, such as those described in Chapter 4 and Chapter 6, suffer from a crippling fragility. A sudden bump or temporary occlusion can

cause tracking to fail, at which point the system can either stop mapping or corrupt the map estimate. This sensitivity makes visual SLAM systems unusable in any but the most controlled scenarios. Even when the tracking assumption isn't violated, the localisation estimate around sufficiently large loops is innaccurate enough to prevent top-down search from finding landmarks when they reappear.

The problem of tracking failure can be addressed with recovery methods, which attempt to detect failed tracking, and stop updating the structure estimate. Then, upon identifying previously-mapped landmarks, the system reinitialises the pose estimate. While this approach prevents corruption of the map, it also precludes additional mapping before existing structure is reencountered. If tracking fails at the beginning of a run, and the camera doesn't return to the pre-failure area until the end of the run, then no mapping is performed in the middle of the sequence.

In Chapter 7, I present a unified framework for recovering from tracking failure and closing loops. The framework uses a hierarchical appearance and structure model, first identifying regions visually similar to the current view, and then matching local landmarks and structure. Instantiated within the graph-based setting of Chapter 6, both recovery and loop closing reduce to adding an edge to graph, according to the appearance and structure correspondence. Global graph optimisation then accommodates new cycle constraints induced by loop closure.

The visual similarity indexing relies on a bag-of-words appearance model, adapted from the image-retrieval application for which it has recently received much attention in computer vision. Instead of identifying image matches in a database, the system selects graph nodes with visual appearance that resembles the latest video frame. Then a local dictionary of landmark appearance is queried for image-to-map correspondences, allowing relative pose recovery. Efficient invariant descriptors are used for both global and local appearance models.

The implemented system successfully recovers from tracking failure due to jerky camera motion and total occlusion, and closes loops even when active search fails. Further,

failed tracking does not halt the mapping process. Instead, the system continues mapping in a new graph, which is later reconnected to the existing graph upon recovery. The appearance models are built online, without any offline training phase. All computation is performed at frame rate. Results are shown for challenging indoor and outdoor sequences.

1.7 Layout

The primary contributions of this thesis have been sketched by the above introduction; each corresponding chapter declares its contributions in more detail. Chapter 4 describes monocular SLAM with the FastSLAM algorithm, and Chapter 5 shows how edge landmarks can be added to the system. Chapter 6 presents the Graph SLAM formulation, and Chapter 7 extends Graph SLAM with unified loop closing and recovery.

Chapter 2 reviews important previous work in SLAM in general, structure from motion, and visual SLAM. It also discusses the various approaches to getting around the scaling problems of SLAM. Each chapter in the body of the thesis also reviews more specifically related work.

Chapter 3 presents the mathematical framework and notation employed by the rest of the thesis. This includes a brief review of rigid transformations and their Lie group properties and representations. The Kalman filter and its extension are described from a general perspective, with a localisation example. Then the well-established Extended Kalman Filter SLAM framework is presented.

Chapter 8 concludes the thesis by summarising the contributions presented and discussing directions for further research.

1.8 Publications

The kernels of the main chapters of this thesis have been peer-reviewed and presented at conferences. The following publications are the result:

Eade & Drummond (2006b): Scalable Monocular SLAM. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, New York, 2006.

Eade & Drummond (2006a, 2008a): Edge Landmarks in Monocular SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*, Edinburgh, 2006. Also in press for *Image and Vision Computing (IVC)*, 2008.

Eade & Drummond (2007): Monocular SLAM as a Graph of Coalesced Observations. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Rio de Janeiro, 2007.

Eade & Drummond (2008b): Unified Loop Closing and Recovery for Monocular SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*, Leeds, 2008.

2

Background

This chapter describes important work of the last two decades relevant to the simultaneous localisation and mapping (SLAM) and structure from motion (SFM) problems. First, the probabilistic framework for SLAM is reviewed, with an emphasis on the common filtering techniques employed by applications. Next, the development of approaches to the now well-understood structure from motion problem is described. Recursive approaches to SFM are closely related to the SLAM problem, so initial and more recent efforts in recursive SFM are presented. A review of the work on SLAM using vision as the primary input follows, which provides a baseline for the contributions of this thesis. Finally, the limitations of the basic approach to SLAM in general are described, and a survey of the various approaches to mitigating or circumventing these limitations is presented.

2.1 Simultaneous Localisation and Mapping

If the structure of the environment is known, determining the pose of the sensor platform (localisation) is straightforward. If, instead, the trajectory of the sensor platform is known, building a representation of the environment (mapping) is equally approachable. However, when the robot or camera is exploring unknown territory with an unknown or uncertain trajectory, the problem becomes considerably more difficult. Both the map and the motion must be estimated from noisy sensor input, and an up-to-date estimate of both must be available in a useful form. This is the domain of simultaneous localisation and mapping (SLAM).

In this section I review the SLAM literature, starting with the first presentation of SLAM as a well-defined statistical estimation problem. The important initial results come almost exclusively from the robotics community. The terms robot, vehicle, and sensor platform are used interchangeably to denote a mobile entity that observes the environment.

2.1.1 Probabilistic Framework

The seminal paper of **Smith & Cheeseman** (1988) gives a general treatment of uncertainty in geometric relationships and parameterisations. The key insight is that uncertain relationships should be represented by a probability distribution over spatial parameters. In particular, the first and second moments of the actual distribution are sufficient for modelling relationships and quantities in most domains. Thus, the natural choice of representation is a mean vector and covariance with dimension equal to the total number of uncertain parameters. The system state mean and covariance can be "built" incrementally by adding a new element to the mean and a new row and column to the covariance matrix for each modelled relation. The resulting state, as a whole, is called the *stochastic map*. In general, the problem of SLAM as pursued in this thesis boils down to representing and maintaining the stochastic map, and fusing sensor observations into it.

Conveniently, representing uncertain quantities by the first and second moments of their distributions renders transformation and propagation straightforward. **Smith & Cheeseman** (1988) and **Durrant-Whyte** (1988) show that a truncated Taylor expansion of a smooth function sends the mean through the function and transforms the covariance by the Jacobian of the function at the mean. The information (inverse covariance) transforms by the transpose of the inverse of the transformation, as shown in **Durrant-Whyte** (1988). For linear transformations and Gaussian distributions, this process yields the exact resulting distribution. In other cases, the result is a linear-Gaussian approximation of the true distribution.

Given this representation, the Kalman Filter (**Kalman** (1960)) is the natural machinery for maintaining a stochastic map (**Smith & Cheeseman** (1988), **Moutarlier & Chatila** (1990)). It is the optimal estimator for linear transformations and Gaussian distributions, with the Extended Kalman Filter (EKF) providing a sub-optimal estimator in the non-linear or non-Gaussian case (**Maybeck** (1979)). An observation of one part of the state actually causes updates to the whole state, due to correlations between the estimates. These correlations are encoded by the off-diagonal elements of the covariance.

Moutarlier & Chatila (1990) describe SLAM in the EKF framework: All objects, called *landmarks*, are represented by coordinates in a common, global frame, including the vehicle state. The parameters of the vehicle follow a *dynamic model*, which predicts the motion of the vehicle over time, incorporating any information given by odometry. Measurements from sensors follow an *observation model*, which, given the current state estimate, predicts the measurement. The process and observation models might be linear or linearised, depending on the nature of the sensor. Noisy observations of landmarks are incorporated into the state estimate, yielding a posterior. Because each individual observation (assumed independent of other observations) depends only on the state of the robot and a fixed portion of the state vector corresponding to the modelled object, the Jacobian of the observation is sparse. Thus, the filter update for one observation takes quadratic time in the dimension of the state vector, in contrast to the cubic time required by the EKF in general. An experimental implementation is described in which a mobile robot successfully builds a sparse, two-dimensional line-segment map of its environment.

2.1.2 The Importance of Correlations

Unlike earlier formulations, the stochastic map of **Smith & Cheeseman** (1988) and **Moutarlier & Chatila** (1990) explicitly model the correlations between different parts of the state. These correlations have been shown to be crucial to the accuracy and statistical consistency of SLAM. **Castellanos et al** (1997) perform an experimental comparison of SLAM using an EKF with either a full covariance (all correlations) or a block-diagonal covariance (where landmarks are assumed independent of each other). In the latter case, the landmark estimates quickly become overconfident – that is, the covariance shrinks below the actual uncertainty in the landmark coordinates, and convergence to the true solution becomes impossible. In contrast, the full-covariance EKF avoids over-optimistic uncertainty estimates, and, via the cross-covariances, changes the estimates of unobserved landmarks when others are observed. Though the author refers to these updates explicitly as backward estimations, they are an implicit result of a full-covariance EKF formulation of SLAM.

The same conclusions are drawn by **Csorba** (1997), by examining the mathematical machinery of the EKF instead of testing experimentally. Landmark estimates become correlated through the uncertain vehicle state. Estimating the vehicle and the landmark states from observations necessarily correlates their estimate errors with each other.

The importance of maintaining correlations is further examined by **Dissanayake et al** (2001), including a theoretical treatment of the questions of correlation and convergence within the Kalman filter SLAM framework. As observations from sensors are repeatedly fused in the filter, the error and uncertainty of landmark estimates decrease monotonically to a lower bound given by the initial uncertainty of the vehicle. Further, the *relative* landmark location estimates tend to certainty, and they become fully correlated. Intuitively, the map estimate becomes rigid – certain knowledge of a small subset of landmarks determines all other landmark locations with certainty. Omitting the cross correlations destroys the structure of the problem and the convergence of the solution.

The EKF SLAM framework has been employed with success in many robotic platforms. **Betge-Brezetz et al** (1996) describe a robot equipped with a laser range-finder that extracts natural landmarks from the environment and performs recursive estimation with the EKF. The robot pose is represented in the filter with four parameters, for 3D position and yaw, while landmarks are parameterised only by their 3D positions. Observations are temporally sparse; results show decent mapping of 12 landmarks over 12 prediction-observation steps. Each time step includes observation of multiple landmarks. The estimated trajectory is more accurate than that of odometry alone when compared to ground-truth.

Csorba (1997) implements SLAM with a laser range-finder and the EKF. The system delivers consistent and accurate performance in an environment with 20 natural landmarks, extracted from the laser scans. Relative observations between landmarks are used to update the filter, described in further detail below.

Davison & Murray (1998) rely on the EKF SLAM algorithm for a mobile robot using active stereo vision as the primary sensor, reporting consistent results. Work on more sophisticated SLAM algorithms often treats the EKF as the optimal filter and uses its results as a reference (**Newman** (1999),**Williams** (2001), **Guivant** (2002), **Bailey** (2002)).

2.2 Localisation and Mapping with Vision

I first review batch structure from motion techniques, then the adaptation of these techniques to recursive frameworks, and finally true SLAM with vision as the primary sensor.

2.2.1 Structure from Motion

In general, structure from motion (SFM) refers to the problem of estimating camera poses and scene structure from multiple images. It might be more aptly called struc-

ture and motion. I consider here techniques for estimating structure and motion from an image sequence using batch methods: Given keypoint correspondences or other image observations from the whole sequence, these algorithms yield an optimal estimate of motion and scene parameters, and sometimes intrinsic camera parameters. Early SFM work depends on strictly controlled or known camera motion or highly constrained scene structure. The review starts with the advent of more mature methods. See **Hartley & Zisserman** (2004) for a bibliography including earlier literature.

Tomasi & Kanade (1991) present an efficient factorisation of structure and motion estimation when working with orthographic projection cameras. Interest point features are tracked through a sequence of images, and then all feature tracks are processed in parallel. Three dimensional points, corresponding to the interest points, are described by world-centred (instead of camera-centred) coordinates, avoiding ill-conditioned estimates when scene depth is small relative to viewing distance. Crucially, by considering "shape" instead of depth, the structure and motion estimates become independent of each other. A singular value decomposition (SVD) is used to split the motion from the structure given all of the data, and the result is robust to noise in the measurements. Note that this factorisation holds only for orthographic projection cameras (and not for perspective projection, where the projection function is nonlinear in depth).

Taylor et al (1991) consider the problem of estimating structure and motion from multiple images where the structure and camera motion is constrained to a 2D plane. This scenario is equivalent to a camera moving in a fixed plane parallel to the ground, observing vertical edges. The presented solution, like most subsequent approaches, attempts to minimise the reprojection error of the observations. This is approached in an iterative least-squares framework which is common to most batch SFM methods. At each iteration, the current estimates of motion and structure provide a linearisation point for a non-linear optimisation algorithm to improve the objective function. As the objective function is a sum of reprojection error over all observations, each iteration takes into account all of the data in order to adjust the parameters. In this work, the authors advocate optimising the camera and scene parameters independently at each iteration (keeping the others fixed). This means much lower dimensional linear

equations are solved at each iteration, making each update much more efficient. The authors claim that this approach does not require significantly more iterations before convergence. The iterative algorithm is shown to deliver better results than an EKF, as it relinearises at each iteration around a better estimate.

Szeliski & Kang (1993) describe a general framework for metric SFM in three dimensions with a full-freedom calibrated perspective-projection camera. The authors argue for a reformulation of perspective projection with object-centred coordinate system, as used by **Tomasi & Kanade** (1991), so that estimation is well-conditioned even as the scene depth shrinks relative to viewing distance. As described by **Taylor et al** (1991), an iterative non-linear least squares optimisation is applied to all of the observations (point tracks or line correspondences) in order to minimise the reprojection error. The Levenberg-Marquardt algorithm is used for the optimisation, so each iteration requires solving the full dimensional linear equation. The method shows rapid convergence even when the initialisation is poor or trivial. A robust fitting with outlier detection is achieved by gating and reweighting the observations after convergence, and then iterating again.

McLauchlan & Murray (1995) show that this common Levenberg-Marquardt global optimisation framework, a particular case of *bundle adjustment*, can be manipulated so that each iteration takes cubic time in the *smaller* of number of landmarks or number of poses. Either motion or structure can be factored out of the main linear equation using Gaussian elimination, so that the dimensionality is reduced to only the structure or motion parameters, respectively. Then the factored-out parameters can be readily recovered after solving the linear equation for the other parameters. The comprehensive review by **Triggs et al** (2000) explains this process in detail. Thus the per-iteration run-time of bundle adjustment using Levenberg-Marquardt with N landmarks and M poses is $O(\min\{N^3 + M, N + M^3\})$.

Pollefeys et al (1998) extend this bundle adjustment framework to allow the intrinsic camera parameters to vary over the course of the sequence. First a projective recovery is performed on the sequence, which does not depend on knowing the intrinsic

parameters, and then the camera parameters are recovered from the projective reconstruction using the absolute conic. **Faugeras** (1995) describes the absolute conic, a projective "circle" invariant to affine transformations that permits recovery of intrinsic parameters. Using these intrinsic parameters (possibly different for each pose), **Pollefeys et al** (1998) upgrade the projective reconstruction to a metric one, aiding a search for dense correspondences between pairs of images. The dense points are triangulated to yield a surface approximation of the scene, with textured faces. The stability and accuracy of the calibration step is greatly improved by fixing some intrinsic parameters to be static throughout the sequence. In typical cases the aspect ratio is a known constant, and there is zero skew. In contrast, focal length changes with zoom and focus during a sequence.

Fitzgibbon & Zisserman (1998) present a robust and efficient method for performing calibration and reconstruction from possibly closed image sequences (where the poses form a loop). The approach is hierarchical, in that subsequences of increasing size are processed and registered together until the whole sequence is recovered. First, the *trifocal tensor* (**Hartley & Zisserman** (2004)) is recovered for each contiguous image triplet. Combining these triplets into subsequences yields a projective piecewise reconstruction, and then the subsequences are all registered into the same frame. Camera parameters and Euclidean structure are recovered by bundle adjustment in the global frame.

Taking the outlier detection strategy of **Szeliski & Kang** (1993) one step further, **Torr et al** (1998) fold multiple frame-to-frame correspondence hypotheses into the optimisation, allowing the rigidity constraint to select the best matches. The paper also presents a robust method for detecting degenerate motions in the case of uncalibrated cameras. As in **Tomasi & Kanade** (1991), the availability of all observations in the optimisation permits local motion degeneracy that is resolved by global rigidity constraints.

An excellent review and exposition of bundle adjustment methods, trade-offs, and implementation details is found in **Triggs et al** (2000).

2.2.2 Recursive SFM

While the bundle adjustment methods described above aim for optimal recovery of structure and motion from a sequence, they also require that the entire sequence be available for processing. This precludes online recovery, and the computational expense of the iterative optimisation is often undesirable or infeasible for applications where sequences do not have predetermined extent. Thus causal and recursive methods, which efficiently produce structure and motion estimates given data only up to the current time, have been developed in tandem with the batch optimisation approaches. As online structure and motion recovery algorithms, these are natural precursors to SLAM using vision.

A crucial property common to all of the recursive SFM methods described in this section is that world features, once they have been occluded or out of view for some fixed number of frames, are not re-observed at any later time. Thus returning to the same actual pose will not necessarily yield the same estimated pose, because differential pose error accumulates over the course of the sequence. It is this odometric property that distinguishes these algorithms from the vision-based SLAM algorithms discussed later..

The seminal work of **Broida et al** (1990) presents and evaluates a recursive system for estimating the structure and motion of a rigid object. The object is assumed to move with constant rotational and translational velocity throughout the available sequence, but all other parameters are estimated from feature-matching observations. The estimation is boot-strapped by an initial batch process over the beginning of the sequence, to give a non-degenerate structure estimate. Then the estimate is maintained and updated recursively with an iterated EKF (an EKF that performs multiple iterations on each update to find the optimal point of linearisation). The system generally shows reasonable and convergent results for simulated and real input. Though the output consists of the state of a moving object with respect to a fixed camera, the information could be trivially manipulated to yield pose estimates for a moving camera and the structure parameters of a fixed object. The authors note that an observation model

with large noise – even when the uncertainty of the observations is correctly modeled – will lead to a divergent filter due to linearisation error.

Azarbayejani & Pentland (1995) provide an excellent treatment of structure and motion recovery in a recursive framework, also taking into account unknown (but static) focal length. An iterated EKF serves as the filter. In contrast to most other representations of 3D points, this work argues for a single parameter estimated in the filter per point, which encodes the depth of the point along a fixed ray direction in the first frame. The justification for this reduced parameterisation is that the uncertainty of point location is so small within the image plane (on the pixel level) that it need not be modeled by the filter. The authors show that explicitly including these two additional state dimensions for each point is equivalent to tracking the bias of the observation model. Experiments with low-error matches support the claim that the bias is a second-order effect, and thus need not be estimated by the filter. Importantly, this representation is well-defined only when the reference pose for the ray direction is known. If features are acquired during the course of the sequence, the parameters of the acquiring camera must be estimated for all subsequent time in the filter. Using the reduced representation yields a factor of three compression of the filter state, which gives a factor 27 reduction in runtime (as each update is cubic the state dimension). Instead of modelling focal length, the system estimates an analogue, β of inverse focal length. When β tends to zero, the represented focal length increases, until in the limit, $\beta = 0$ perfectly encodes the case of orthographic projection. The relationship between focal length and translation along the depth axis is strong, so translation in this direction is estimated indirectly as a function of focal length.

The one-parameter representation of **Azarbayejani & Pentland** (1995), however, is reexamined by **McLauchlan & Murray** (1995) and **Chiuso et al** (2002). Both papers find that privileging the first observation by ignoring its angular uncertainty leads to bias that, at best, keeps the map from converging, and at worst, causes catastrophic error.

Another example of recursive SFM using the EKF and point feature tracks is given by **Bouguet & Perona** (1995). This is essentially a sliding-window method, where the

EKF maintains a representation of the state older than the back of the window. In contrast to the batch approaches, all poses and observations older than the window are not relinearised around updated estimates, so linearisation errors affect the result, which is necessarily suboptimal. Motion estimates for the camera are integrated from frame to frame using optical flow.

McLauchlan & Murray (1995) present a generalised recursive formulation of SFM motivated by the typical batch approach. The variable state dimension filter (VSDF) allows features to be added and removed from the state (maintained by an EKF) in a principled way as the sequence progresses. This process corresponds to marginalisation of subsets of the estimated state, and in the EKF framework, it happily reduces to deletion and insertion of state elements, rows, and columns from the mean vector and covariance matrix. Further, heterogeneous state is straightforward to represent; the paper shows results for points and lines. The system is boostrapped from a batch recovery over a starting subsequence. The authors emphasise that the resulting filter is suboptimal for nonlinear models (relative to an iterative batch approach), as is true for all of the recursive SFM methods.

Taking the work of **Azarbayejani & Pentland** (1995) as a starting point, **Chiuso et al** (2002) formalise the mathematical and geometrical framework of recursive structure from motion. The work includes an analysis of the minimality and observability of typical models for the process. In particular, the single-parameter representation of landmarks used by **Azarbayejani & Pentland** (1995) is shown to be sub-minimal, in that it unfairly weights initial observations more than subsequent ones. Though not the focus of this paper, the use of the exponential map to perform optimal estimate updates on the manifold of rigid motions is included in the formulation. As with earlier approaches, the operation of the system is split into transient and steady-state stages, the first of which involves initialisation of features/landmarks. However, departing from the earlier work, this initialisation does not rely on batch methods, but instead uses a separate "sub-filter" EKF per new feature. After a probationary period, the estimate of the sub-filter is added to the main EKF. This method ignores the correlations induced between existing structure estimates and the new landmark through the camera pose. New features are chosen in order to uniformly populate the image plane, and

selected from images using an SSD brightness-gradient measure. No details are given of the feature-tracking method used to obtain subsequent observations. The results for favourable scenes with sufficient features visible and slow, smooth motion show accurate reconstruction for short sequences and decent motion estimates. However, when features are occluded they are removed from the filter, so drift will accumulate over time.

Jin et al (2003) takes the formalisation of **Chiuso et al** (2002) further, extending the representation of the scene to planar patches with orientation. By estimating in the EKF the surface normals associated with each patch, the appearance model for the patches is richer. Affine lighting variation is accounted for by normalising the variance of the patches. An important innovation of the work concerns its approach to landmarks which were occluded but have since reappeared. They are reobserved, and their structure estimate is updated to reflect the newer estimate. However, because old landmarks are removed from the filter, other landmarks in the map are not updated. Feature tracking proceeds by projecting estimates of the patches into the image and searching for the patch in a fixed radius of the predicted location. This is a crude example of active search, discussed in more detail below.

In contrast to the EKF used by the above systems, **Nistér et al** (2004) perform sliding-window bundle adjustment over a video sequence to recover camera motion, with a calibrated monocular or stereo rig. Each portion of the algorithm is carefully optimised for robustness and efficiency. Harris features (**Harris & Stephen** (1988)) are matched across frames with a fixed maximum disparity limit, and outliers are rejected using the five-point algorithm and preemptive RANSAC (**Nistér** (2003)). Point positions are triangulated from the resulting feature tracks, using the earliest and latest observations as an approximation of the widest available baseline. Then scale across triples of frames is robustly recovered using the three-point algorithm (**Fischler & Bolles** (1981)). Finally, bundle adjustment over the structure and motion parameters of the most recent three frames yields a locally optimal estimate. In order to avoid propagating structure errors through the differential motion estimation, "firewalls" are placed at fixed time intervals, isolating all feature observations before the firewalls

from those after. This serves to reset the structure estimation. Results from real sequences are impressive, yielding accurate motion recovery through long sequences compared to onboard GPS taken as ground truth. The system runs in real time at a frame rate of $\sim$15 Hz. Despite the high differential accuracy, however, drift still accumulates, and the system does not correctly estimate the return of the vehicle to the beginning of a sequence. Again, unless old landmarks are remembered and reobserved later, drift in SFM will be inevitable with noisy sensors.

2.2.3 SLAM with Vision

In contrast to the recursive SFM methods described above, the work in this section aims to maintain a more permanent representation of structure, and remembers "old" structure upon a return to a previously visited portion of the environment. These approaches thus either implicitly resemble, or explicitly attempt, solutions to the SLAM problem with vision as the primary sensor.

Harris & Pike (1988) show an early attempt at recursive SFM without drift, lacking the insights about correlation and uncertainty espoused by **Smith & Cheeseman** (1988) and **Moutarlier & Chatila** (1990). Landmarks, chosen from feature points (from an unspecified detector), are represented in separate EKFs, as though they are statistically independent of each other. The uncertainty in the camera pose is not maintained by the system. Because correlations are ignored, the system cannot converge to the correct map (though the results show decent performance on short simple sequences). However, the implementation exhibits two important properties. Firstly, the parameterisation of the landmarks is chosen to allow better representation within the filter, by employing "disparity" coordinates. These are coordinates defined with respect to the location of the feature in an image, along with the inverse depth of the landmark along the viewing direction of the camera. Because the parameters are simply related to image properties, they make use of the EKF feasible for representing highly uncertain landmark estimates. Secondly, this paper presents one of the first clear explanations of *active search* (albeit without taking into account camera uncertainty). Active search comprises the use of current state estimates to direct a search for further observations

of the state, either within the current image or by actively moving the sensor. In this case, the state, including the current pose estimate, permits the prediction, through the observation model, of where each landmarks will appear in the current image. The uncertain estimate for each landmark is projected into the image using the perspective projection model and camera parameters, and gated to yield an ellipse inside which the landmark should appear with a chosen confidence. The image region inside the ellipse is then searched for the landmark using its appearance model (an image patch).

Though **Rahimi et al** (2001) attack the problem of drift within a differential tracking setting, the approach highlights the crucial difference between recursive SFM and vision SLAM. In a differential tracking system, the pose of a tracked object (static camera) or camera (static world) is estimated by making only differential measurements from frame to frame. Integrating these changes over time yields a pose estimate, but the estimate suffers from unbounded drift. This paper presents a method for using past frames to get temporally non-local differential measurements. When the pose or state is sufficiently close to a previously saved state (a subset of all frames is preserved), tracking occurs not only between the most recent two frames, but also between the newest frame and the nearby past frame. By combining these differential measurements, the estimate of current pose or state is modified to satisfy both old and new observations, and drift is bounded. Intuitively, when seeing the same image, the system will report the same state. Examples and results, showing bounded error over time, are presented for head tracking and camera egomotion estimation.

Neira et al (1997) present one of the first direct applications of vision within the canonical stochastic map regime of **Smith & Cheeseman** (1988) and **Moutarlier & Chatila** (1990). A robot with odometric sensors is also equipped with a camera. From the video, vertical edges are extracted and tracked, and represented as 2D points in the plane of the robot's motion. A variant of the EKF filters the whole state with full covariance, so observations of the vertical edges greatly reduce the orientational uncertainty of the robot. Thus closed trajectories are navigated without drift. The paper brings up the important issue of data association – the procedure by which sensor observations are associated with landmarks in the state. The authors note that a different approach than that used for range-finder SLAM is appropriate when using vision,

and that "tracking" the edges by predicting their location from the filter greatly eases the problem. This common strategy – first predicting landmark locations in the image, then associating observations with the most likely landmark – is similar to active search, but in this case does not focus the image search using the prediction. Edges are extracted from the image in a bottom-up fashion before the data association process begins. The system gives reasonable results with $\sim$20 vertical edge landmarks combined with the robot's odometry in a closed trajectory.

The system of **Se et al** (2001), implemented on a robot with a trinocular rig, takes a different approach to data association with vision. SIFT features (**Lowe** (2004)) extracted at each time-step are matched to landmarks whose estimated projection is nearby in the image. Matches are confirmed using the orientation and scale associated with each SIFT feature. The estimation framework is quite similar to that of **Harris & Pike** (1988), in that each landmark's estimate is treated as independent from all others, and maintained in a separate EKF, ignoring the correlations required for consistency. In this system, however, camera uncertainty is estimated and propagated to newly initialised landmarks. The combination of trinocular rig and good odometry results in usable maps, though no quantitative results are given for structure or motion recovery. The distinctive nature of SIFT descriptors permits global localisation within the system, addressing the kidnapped robot problem.

The robot SLAM system described by **Davison & Murray** (1998), **Davison & Kita** (2001), and **Davison & Murray** (2002), and presented in detail in **Davison** (1998), successfully employs vision as the primary sensor, and provides an excellent reference baseline for the implementation of stochastic-map EKF SLAM using active search. A stereo rig is mounted on a pan-tilt head, and the only other input to the estimation is from odometry. Three-dimensional point landmarks are maintained in a full-covariance EKF, along with the robot pose. At each time step, the estimates of landmarks that lie within the current viewing volume are projected into the image plane, along with a gated search ellipse, taking into account landmark and pose uncertainty. Then the landmark whose image uncertainty (innovation covariance) is largest by volume is sought within its region, as observing it will (roughly) yield the most information to the filter. This notion is formalised and improved by **Davison** (2005). The

ellipse is searched using normalised sum-of-squared-differences for a small patch representing the landmark, providing invariance to affine lighting changes. Each of the two stereo frames is searched separately, and the results are discarded if they disagree with the known epipolar geometry of the rig.

The map built by the system is purposely sparse: it is used primarily for localisation, so it need not represent dense detail, and the quadratic time and storage cost of the EKF makes maps with many landmarks computationally prohibitive. Because the map is carefully constructed, despite its sparseness it allows accurate and repeatable localisation over small environments. Further, it can be harnessed for basic navigation and obstacle avoidance tasks, as shown in the results. The system also exhibits an important and perhaps surprisingly successful approach to map management. Even though the constraints on landmarks (that they be locally planar, static, and well-represented by their image patch) cannot be confirmed immediately, landmarks that pass the most basic tests are added to the filter, and discarded when they fail to be observed sufficiently often. The model is assumed to hold, and the landmarks are rejected when the assumption proves spurious; this technique is employed by many subsequent SLAM systems. Landmark acquisition, with sparseness in mind, is triggered only when necessary to ensure a minimum density of landmarks in the vicinity. The detector of **Shi & Tomasi** (1994) is employed to select new features, and the stereo rig provides a non-degenerate initial estimate of position for adding the new landmark to the stochastic map.

Jung & Lacroix (2003) describe a SLAM implementation with low altitude stereo imagery as the sole sensor. The visual processing relies on robust Harris feature point matching, both within a stereo pair and across images captured at different times. The resulting matches are partitioned, some to be used as landmark observations, and some to be used for frame-to-frame egomotion estimation. The egomotion estimation is used in the prediction step of the EKF just as other odometry would be. By splitting up the observations, the system avoids using information more than once, and thus keeps the filter prediction step independent of the update step. When old landmarks come into view, they are reobserved, decreasing the error of both the map and the

localisation. In the results shown, the visual motion estimation by itself is quite accurate over the length of the sequences; adding SLAM on top of it slightly improves the results. Note that the scenario does not require frame-rate operation (as the aerial platform is stable and slow-moving), so filter updates occur less than once a second, and the visual motion estimation step can be more computationally intensive than would be possible in a more time-constrained system.

Using **Davison & Murray** (2002) as a basis, **Davison** (2003) develops a real-time SLAM algorithm for a single hand-held camera. While much of the machinery and approach carries over, SLAM with a monocular sensor without odometry is much more challenging than with a robot equipped with a stereo head. This work presents the first real-time implementation of SLAM with only a single camera, and provides a good point of reference for the work of this thesis. As with the methods discussed above, the EKF provides the statistical machinery for estimation. Davison represents the current camera pose with a quaternion and a translation vector, along with velocity terms in each. Imposing a constant velocity model on the pose dynamics greatly increases the stability of the system, and make active search feasible.

Run-time operation begins by observing a fiducial configuration with known geometry, thus fixing the coordinate frame and scale of the map, and providing good enough pose estimates to begin adding new landmarks. Landmarks are occasionally chosen from small windows of the image where no existing landmark estimates project, in order to maintain a minimum workable density of landmarks, well distributed in the environment. As in **Davison & Murray** (2002), Shi-Tomasi features are detected in these windows for choosing new landmarks. One of the particular difficulties of monocular SLAM now rears its head: from just one observation of a feature with a single camera, the landmark estimate is unconstrained in depth. The problem of how this degenerate estimate should be represented and improved in the filter is called *initialisation*, and several approaches have been described above for recursive structure from motion algorithms. Davison first adds a ray estimate to the EKF corresponding to the single observation of the landmark. As mentioned above, the depth distribution is not initially Gaussian, so a different representation must be chosen if depth is to be estimated incrementally. Davison maintains a separate particle filter along the ray for

estimating the depth of each new landmark. Particles begin uniformly spaced over a fixed depth interval (0.5m to 5m in this work). As a new landmark is repeatedly observed, its depth particles are reweighted and resampled, until they converge enough that they can be accurately represented as a Gaussian. Then the appropriate Gaussian transformed to world coordinates and added to the filter, replacing the ray, and the landmark is fully initialised. We discuss other approaches to initialisation later in this thesis.

For the same reasons espoused by **Davison & Murray** (2002), Davison argues that the map should be as sparse as possible while still supporting localisation, as the pose estimate is the real output of the system. On the test hardware, the system can operate at 30 Hz with up to 100 landmarks before the EKF update cost is too high. Again, innovation covariance is used to prioritise which landmarks to observe. The results show that the reconstruction and localisation is reasonably accurate for small scenes. Importantly, the active search method allows the system to close small loops successfully.

Molton et al (2004) extend the appearance model used in the system of **Davison** (2003) to include patch normals for landmarks. The primary goal is not to enrich the map representation, but rather to increase the range of motion over which active search for a given landmark is successful. Normals are maintained outside of the main EKF, and updated using inverse compositional techniques and gradient descent, every time landmarks are observed. With an estimate of patch normals, full projective warping can be applied to patches before active search, better approximating the actual appearance of the patch in the image. For scenes with many textured planar areas, the patch-normal estimation process works reliably, and significantly broadens the view change over which a given landmark can be observed. The authors are careful to emphasise that a poorly-constrained normal estimate reflects a patch with insufficient texture to distinguish within a range of normal directions, so that a poor estimate of the normal will not affect the ability of the system to observe the landmark.

2.3 Limitations of EKF SLAM

While the EKF machinery offers a straightforward method of maintaining a stochastic map, it has significant limitations. The quadratic cost of updates and storage is the most immediate problem, harshly constraining the number of landmarks in the map if efficient operation is required. However, even with small maps, nonlinear process and observation models can lead to divergence of the filter. There has been extensive work in the SLAM literature addressing these problems; we review some of the key results here.

2.3.1 Computational Scaling

Because the EKF maintains a full covariance matrix for the state estimate, which has dimension $O(N)$ for N landmarks, the storage cost is $O(N^2)$. Further, due to the complete correlation to which the stochastic map tends, an update to any part of the state will decrease uncertainty in all other parts of the state. Each update thus modifies every element of the covariance, with a computation cost of $O(N^2)$. Mapping with hundreds or thousands of landmarks then becomes infeasible for realistic hardware or time constraints.

2.3.1.1 Map Minimisation

The problem can be rather directly avoided by limiting the size of the map. **Dissanayake et al** (2000) show that landmarks can be deleted from the state vector without compromising the consistency of the estimate. By enforcing a maximum spatial density of landmarks, discarding all but one representative within a region of the map, the cost of updates is greatly reduced while the accuracy of localisation is not significantly degraded. Further, by selecting which landmarks in a region to keep or discard according to their information content, the degradation in accuracy can be minimised.

Davison, both in his work with active stereo vision (**Davison & Murray** (2002)) and with a hand-held camera (**Davison** (2003)) emphasises the importance of maintaining a sparse map when using the EKF for localisation. By careful selection of landmarks and observation scheduling, maximal localisation performance can be squeezed out of a relatively small set of landmarks in an area.

2.3.1.2 Relative Representations

Csorba (1997) presents a relative landmark formulation, where instead of estimating landmark locations in a global frame, the filter estimates the relative transformations between pairs of landmarks. Information about the transformation between landmarks A and B can be gleaned from an sensor observation that includes both A and B. The representation requires $O(N^2)$ state elements, but with the advantage that estimates for distinct relations are independent of each other. Thus a block-diagonal covariance is not an approximation, and the update cost is $O(1)$ per observation. The downside of the representation is that there is no simple access to location and uncertainty of landmarks or robot in a global frame, so that localisation is expressed relative to landmarks. Further, the relative transformations need not be coherent, so that one transformation into a common frame (by composing landmarks relationships) can differ from another. The thesis emphasises that robot pose uncertainty in the global frame increases with distance from the global origin, as do landmark location uncertainties. However, due to the covariance between the two, the uncertainty in robot-landmark relationships is independent of the global frame. This property is encoded explicitly by the relative location filter.

The Geometric Projection Filter (GPF) (**Newman** (1999), **Newman & Durrant-Whyte** (2001)) combines a relative map with a constraint application algorithm in order to arrive at a coherent estimate of landmark locations. Each cycle in the network of relative transformations should compose to the identity transformation, so each cycle provides a constraint on the relations involved. Application of a set of N_C constraints requires $O(N_C^3)$ computation, so special attention is paid to minimising the number of

constraints involved. Results are shown for synthetic landmarks observed by a submersible vehicle equipped with sonar. The GPF delivers relative landmark distances that are similar to those provided by the standard EKF formulation, while decoupling the map and vehicle estimates.

2.3.1.3 Update Amortisation

In common SLAM scenarios, landmarks from a small, local subset of the whole map are observed often in a short period of time, before the robot moves on to other parts of the environment. In these cases, the quadratic cost of updating the whole state with each observation is mostly wasted, as only a small part of the state will be needed anytime soon. Two algorithms, mostly equivalent but developed in parallel, address this burden by reformulating the EKF machinery without affecting the mathematical result.

The postponement algorithm of **Knight et al** (2001) maintains a working set of landmarks, which are fully updated with respect to one another upon observation. The earlier incarnation of the work by **Davison** (1998) assumes the working set contains exactly one landmark, which will be repeatedly and exclusively observed, but the method generalises to arbitrary subsets. Observing landmarks in the working set always updates their inter-covariances and state fully (along with the vehicle state), while updates to the rest of the map state are delayed. Landmarks can be incrementally added to the working set, whether they be existing in the map or newly initialised, until the working set grows to a bounded size. Then the full map state is updated using data maintained during the processing of the working set. This global update has full quadratic cost in the size of the map, but it also yields the same result as if the whole state had been updated at each step. Further, this expensive step can be computed in parallel while building and observing a new working set. The update for the working set is quadratic only in the size of the working set.

The Compressed EKF (**Guivant & Nebot** (2001), **Guivant** (2002)) is a reformulation of the EKF similar to that of postponement. The difference is that the working set is

determined not adaptively by what the robot is observing, but instead *a priori* by a geometric partitioning of the environment. A simple rectangular grid is suggested, with hysteresis at the boundaries so that nearby landmarks, which might in reality belong to the local set, are still observed. The runtime costs are the same as for postponement: local update cost quadratic in the size of the working set and global update cost quadratic in the size of the whole map.

2.3.2 Consistency

Even when the map is small enough that the EKF is not limited by computation demands, the estimation process can still yield inconsistent results. In general, a consistent filter is one that, given unlimited observations, converges to the correct estimate of state. More intuitively, though a consistent filter might, during operation, have an inaccurate estimate of the mean of the state distribution, given many more observations, the estimate should converge to the truth. This convergence requires a correct handling of uncertainty by the filter. In particular, the estimation process must not be overconfident in its knowledge of the state. The estimated uncertainty must never drop below the true uncertainty bound imposed by fusing uncertain observations. With non-linear observation and dynamics model, the EKF is prone to exactly this sort of overconfidence.

Julier & Uhlmann (2001) give a clear mathematical and empirical example of the inconsistency and subsequent divergence of the EKF even in the most basic scenarios. The paper considers a filter estimating the state of a vehicle and exactly one landmark in the two-dimensional plane, observed with a typical range-bearing model. The estimate of vehicle state, theoretically, should never be more certain than it is at the first time step, as no observation of the landmark can reduce the global registration uncertainty of the map or vehicle. However, in the example, the orientation uncertainty of the static vehicle state quickly drops below its original level after only a few observations. The decreased vehicle uncertainty feeds back into estimation of the landmark, resulting in an inconsistent map and vehicle estimate. The culprit is shown to be the

dependence of the observation Jacobian on both the noisy observations and the relative configuration of the vehicle and the landmark. Because the model is nonlinear, even if the true state is used as the linearisation point, the Jacobians change as the vehicle moves, and the mathematical conditions necessary for a consistent EKF are violated. The result implies that the EKF is not capable of providing long-term consistent estimation in the general SLAM case.

Castellanos et al (2004) recognise the same problem with the EKF, confirming it by simulated experiments. The paper suggests always keeping the map estimate in the local frame of the robot, which greatly reduces the linearisation discrepancies, resulting in extended consistent operation. However, even with this modification the filter eventually becomes inconsistent. More details of using a robot-local coordinate frame in the EKF framework, as well as incorporating it with a postponement/compressed EKF local-to-global update strategy, are given by **Castellanos et al** (2007).

Bailey et al (2006a) examine the causes of the inconsistency in more detail, showing that the source of the problem in the case of two-dimensional SLAM is vehicle heading uncertainty. As soon as the heading becomes sufficiently uncertain, the landmark estimates show spurious information gain. Filtering with Jacobians taken around the current estimate quickly leads to inconsistency, while using Jacobians taken around the ground-truth state generally avoids the problem. Of course, ground-truth state is not available to a SLAM algorithm. The authors conclude that in order to minimise inconsistency with the EKF, the system must take periodic absolute measurements of orientation (to bound heading uncertainty), or always represent estimates in the local frame of the robot, so that heading uncertainty is minimised. This second technique is a natural consequence of the submapping formulations described below.

Another important contribution of **Bailey et al** (2006a) is the realisation that consistency of a filter can only be evaluated over many runs with observations sampled from their noise distributions independently each time. The average *normal estimation error squared* (NEES) over these runs then provides good input for a chi-squared confidence test of consistency for all or part of the state. A single run determines nothing,

as an inconsistent filter can appear consistent for certain sampled observations and vice versa. We employ the NEES method for testing consistency later in this thesis.

The more recent work of **Julier & Uhlmann** (2007) argues that consistent estimation is essentially impossible in the EKF SLAM framework, and proposes a different information fusion algorithm that is guaranteed to be conservative. The algorithm employs *covariance intersection* (**Julier & Uhlmann** (1997b)) as the primary statistical machinery. Covariance intersection conservatively combines two Gaussian observations of the same state with unknown correlation, always producing a result that is consistent regardless of what the correlation might be. A simple application of covariance intersection to SLAM ignores all correlations between vehicle and landmark states, and fuses all observations into the state estimate conservatively. A benefit of this formulation is that the storage and update costs of the filter are vastly reduced, allowing SLAM with an arbitrarily large map in bounded update time. However, the map is highly over-conservative, and does not necessarily converge to the true solution. The authors instead suggest a hybrid approach that leverages some certain knowledge of correlation, such as the independence of process noise from one time step to the next, and the independence of observations from each other. Covariance intersection is used for all other operations. This framework still yields very conservative estimates, but better than those of covariance intersection alone. Further, covariance intersection can be used to combine the output of multiple heterogenous SLAM algorithms in a consistent manner, potentially yielding highly robust operation.

2.3.3 Submapping Strategies

The computational demands and consistency concerns of EKF SLAM lead naturally to the development of divide-and-conquer techniques for estimation on large maps. These approaches, which generally decompose the global map into many local maps of bounded complexity. For this reason we refer to them here as submapping strategies.

Chong & Kleeman (1999) build a collection of local maps to avoid the computational limit of the EKF. When the uncertainty of robot pose grows beyond a bound, indicating that the correlation to landmarks near the origin of the current map is weak, the robot saves the current map and starts creating a new map, recording the topological connection of the two. Pose coordinates are always represented in the global frame, and a list of poses that robot has assumed is stored with each map. In order to detect when the robot re-enters an existing map, the current pose estimate is compared to all poses associated with topologically nearby maps. When a sufficiently close pose match is found, the hypothetical map is loaded and a map matching algorithm is performed. Map matching attempts to associate the latest observations with elements of an existing map, and when successful, yields the pose of the robot relative to the map. Thus the system can detect map re-entry and initialise the pose appropriately within the existing map. Because the size of local maps is bounded, computation is also bounded. Only one map is ever updated in a time step.

Leonard & Feder (2001) introduce a submapping strategy called decoupled stochastic mapping (DSM). Submap regions have pre-determined geometric bounds in the global coordinate space. In this paper, the space is split into a cartesian grid with some overlap.If the density of landmarks in space is bounded, then bounded region volume implies a bounded number of landmarks in each region, and thus bounded computation for region-local maps. All local maps are globally registered. When the robot leaves a region, it creates a new map if none exists in the new region, or it must transition into an existing map for the destination region. Two transition algorithms are described: cross-map relocation and cross-map updating. In the relocation case, the uncertainty of the vehicle pose estimate in the source submap is added to that in the destination submap, and the current pose estimate from the source submap is taken as the mean for the destination. This is consistent only if the cross-correlation between vehicle and landmarks has not changed in the destination submap since the last time it was updated. For cross-map updating, the destination covariance for vehicle and landmarks is artificially inflated, the destination pose mean is “randomised”, and the pose estimate from the source submap is fused with the destination as an observation of pose, using the EKF. In this case, the resulting vehicle pose uncertainty in the destination is reduced, but the landmark uncertainty is increased. No mathematical proof

of consistency can be given for these approaches, especially with nonlinear models. The results shows that the consistency of the method depends on the trajectory of the robot in an unexplained manner.

Leonard & Newman (2003) improve the DSM formulation by more rigorously defining the inter-map transition algorithm, and by using local coordinate frames within each local map. Each submap's frame is tied to a root landmark in the region, whose position is defined as the origin of the local map. The global location for the root is found by composing the transformations between map roots of any local maps between the map in question and the first map, which defines the global origin. Landmarks in a local map can be globally registered with reference to their root. The global submap location estimates can be improved whenever the robot moves from one submap to another. Landmarks shared between the adjacent submaps are examined, and if the relation between any shared landmarks better constrains the transition than the existing global root position estimate, the map is "re-rooted" around the shared landmark, and the global position estimate is updated. When a local map becomes fully internally correlated, no re-rooting will improve the global position estimate. Only simulation results are shown in the paper; data association is assumed to be known and the observation and dynamic models are linear. For the simulated data, the technique appears to be consistent.

Williams (2001) also describes a mapping algorithm – the "constrained relative submap filter" (CRSF) – that builds local maps, each with its own coordinate frame. However, the map boundaries are not determined *a priori*, but a new map is created when the robot pose uncertainty becomes large relative to the current coordinate system. The map-to-map transformation estimates form a tree structure, through which local estimates can be propagated to give global estimates. Revisitation of old maps is detected during the data association process, through a feature matching algorithm which attempts to find the best joint compatibility between sets of features extracted from laser scans. Few details are given for how this matching operates against many independent local submaps. Assuming revisitation is detected, the robot's pose can be represented relative to the existing coordinate frame, and mapping continues there. Though

the tree of map-to-map transformations is sufficient to allow traversal between the coordinate systems, it does not guarantee global convergence of the map. Least-squares optimisation can be applied to the whole set of submaps, effectively enforcing global constraints, but few details of this optimisation are shown.

Bailey (2002) presents a flexible formulation of local maps joined by coupling estimates, called "network coupled feature maps" (NCFM). NCFM consists of local maps with local coordinate systems, estimated with the EKF machinery, joined by uncertain coordinate transformations, yielding a graph of local frames. The coupling estimates between the submaps are constrained both by the initial motion estimates, when one submap is created at the boundary of an existing one, and by common landmarks estimated in the two incident maps. This latter constraint allows the coupling uncertainty to shrink as the local maps converge, so that the whole configuration of submaps converges globally over time. Traversal of the map during estimation is driven by the data association algorithm, which uses either joint compatibility tests or subgraph matching of feature observations. When the robot is near the boundary of the current map, an attempt is made to associate each observation set with nearby maps. When this association is successful, and the robot uncertainty relative to the current map is high, a transition occurs, and the robot is represented within the other local map.

The NCFM permits efficient loop closure detection with a two-pass algorithm. In the first pass, potentially visible local maps which are not adjacent to the current map are identified by composing the coupling estimates through the graph. This registers distant submaps with respect to the current map. Sufficiently proximate submaps then pass to the second stage, where a map matching algorithm, or data association with the latest observation set, yields correspondences between portions of the two maps, or the observations and the distant map, respectively. Then the coupling estimate between the now-joined local maps is updated using any common landmark estimates. Because common landmarks are used to constrain inter-map coupling estimates, the estimates are statistically correlated with other coupling estimates incident to the endpoint local maps. However, the author argues intuitively and empirically that ignoring these correlations actually results in conservative (but still consistent) coupling estimates rather than optimistic ones.

Bosse et al (2004) describe a framework for graph-based scalable SLAM called Atlas, developed independently from NCFM but nearly simultaneously. The Atlas framework also maintains a set of local map estimates with local coordinate frames, joined by transformation estimates, as in NCFM. However, the local map estimates of Atlas can be processed with heterogeneous filtering approaches, not only the EKF. At each time step, only constant computation is required beyond the local map update. Because local maps have bounded size, operation in general thus requires only constant time. The map-to-map transformation estimates in Atlas are constrained only by the vehicle pose estimates at the boundaries between two maps. Map traversal with Atlas is more sophisticated than in NCFM: when the estimated pose near other connected local maps, a new hypothesis for pose relative to the other map is spawned. This hypothesis is updated along with the current mature hypothesis, which keeps the robot in the current map. Multiple new hypotheses (called juvenile hypotheses) might exist simultaneously with the current mature hypothesis. Juvenile hypotheses do not update their local maps, but instead maintain a quality estimate reflecting the success of data association with each new set of observations. When the quality of a hypothesis exceeds all others including the current mature hypothesis, it becomes the dominant hypothesis. This multiple-hypothesis traversal mechanism makes Atlas robust to environments with repeated or ambiguous structure.

In order to transform pose or landmark estimates from one local map to another, or to register all map estimates with a common global frame, a path must be chosen through the graph of submaps. The Dijkstra shortest-path tree from the current map is maintained incrementally for this purpose. The metric by which the tree is built reflects the uncertainty of each map-to-map transformation, such that more uncertain transformations correspond to longer edges in the Dijkstra formulation. For Gaussian coupling estimates, the edge-weight for the Dijsktra algorithm can be a function of either the trace or the determinant of the transformation covariance. Thus the path taken through the graph from the current map to any other local map is the one that minimises the uncertainty of the resulting total transformation. If all local estimates are registered in a common frame using this tree (e.g. for rendering purposes), the inter-submap transformation estimates outside the tree are ignored. These can be taken into account using an iterative global least-squares method, which, although it converges

quickly, cannot be performed during real-time operation for graphs with many local maps.

Estrada et al (2005) present a submapping framework called Hierarchical SLAM that, unlike Atlas, imposes loop constraints on the graph during operation. Transformations between maps are represented in a relative stochastic map. These transformation estimates, which are purely a function of vehicle pose estimates upon submap creation, are statistically independent of each other, so that the covariance is block diagonal. However, the relative stochastic map of submap transformations might not be coherent (just as different paths between two local maps in the Atlas graph give different transformations). When an overlap between existing maps is detected, a cycle is created in the transformation graph, yielding a new constraint on all transformations in the cycle. This constraint is imposed using sequential quadratic programming (SQP), which requires computation linear in the number of the maps in the cycle. The cost of SQP grows cubically with the number of cycles in the graph, so that a graph with many cycles is expensive to correct, but cycle constraints are composed only when the cycles are created. Results show that imposing these constraints results in more accurate and precise global map estimates than CRSF, NCFM, or Atlas (without offline global optimisation) would yield.

2.3.4 Decorrelation

In contrast to the submapping strategies discussed above, the following approaches to SLAM take advantage of the weak correlation between spatially or temporally distant state estimates in order to make the estimation process more efficient.

Guivant & Nebot (2002) note that, when using a relative landmark representation, the correlations between distant constellations of landmarks in the stochastic map are weak. This is because they are correlated only indirectly, through other landmarks and the motion of the robot over significant distances. By decorrelating these constellations, the storage and update costs of EKF SLAM can be reduced from quadratic to

linear. For each constellation, a small subset of the landmarks are represented in absolute coordinates, and all others are represented in coordinates relative to these. The relative landmarks in one constellation are only weakly correlated to relative landmarks of other constellations. Instead of simply ignoring these weak correlations, which would lead immediately to an inconsistent estimate, the algorithm subsumes the correlations by inflating the diagonal covariance blocks of the constellations. Weaker cross-constellation correlations means less the uncertainty need be added to each constellation in order to keep the filter consistent. Because the covariance is inflated as a function of the cross-correlation, the resulting filter is always conservative. Because the compressed EKF delays modification of the whole map while continuously updating a subset of it, the decorrelation process occurs only intermittently. This results in less conservative (but still consistent) estimation.

Thrun et al (2002) and **Thrun et al** (2004) detail a SLAM filter called the sparse extended information filter (SEIF) which explicitly ignores weak correlations between landmarks. In contrast to the various EKF SLAM algorithms discussed above, SEIF builds on the extended information filter (EIF), which is identical to the EKF except in terms of representation. The EIF maintains the canonical or *information* form of the state estimate, given by the information matrix (inverse covariance) and the information vector (mean multiplied by inverse covariance):

$$\mathbf{H} \equiv \mathbf{\Sigma}^{-1} \tag{2.1}$$

$$\mathbf{b} \equiv \mathbf{\Sigma}^{-1}\boldsymbol{\mu} \tag{2.2}$$

In the EIF, the measurement update for k simultaneously observed landmarks takes $O(k)$ time, as the update consists of simply adding information to $O(k)$ elements of the information matrix and adding to $O(k)$ elements of the information vector. Further, the information matrix is naturally sparse, because non-zero off-diagonal terms exist only between landmarks observed at the same time. Unfortunately, the EIF motion update takes quadratic time, as the old pose estimate must be marginalised out of the distribution. However, if the information matrix is exactly sparse, so that each row or column has $O(1)$ non-zero elements, then the motion update can be accomplished in constant time if the mean vector is known. In order to achieve exact sparsity the SEIF periodically removes small non-zero off-diagonal elements from the information

matrix, maintaining a constant number of *links* (off-diagonal information elements) per landmark. The mean vector can be efficiently recovered at each time step using gradient descent, with the predicted mean as the starting point. This generally takes a small number of iterations. Results show that the SEIF yields maps and trajectories very similar to those given by full EKF SLAM, but with constant runtime complexity. However, because correlations are ignored by sparsifying the information matrix, the filter is necessarily optimistic, and might eventually become inconsistent. Also, all landmarks are stored in absolute coordinates, which might lead to significant linearisation errors at large scales, though local coordinate frames could be used instead. Because of the information form of representation, covariance estimates for landmarks cannot be easily recovered, but instead must be approximated. This can impact the performance of data-association techniques.

Eustice et al (2005) observe that the sparsification procedure of SEIF, while maintaining consistency between local landmark groups, creates globally inconsistent maps. Reinterpretation of the process as a Bayes net, in a similar spirit to **Paskin** (2003), shows that accounting for conditional independence properties in the landmarks yields much better consistency in the global frame, while keeping a sparse representation. Unfortunately, the modified sparsification can no longer be completed in constant time, so that the cost grows cubically with the size of the map. This makes its application infeasible.

Paskin (2003) analyses EKF SLAM as a Bayesian graphical model in general, and an instance of a Gaussian Markov random field in particular. The junction tree algorithm for inference by message passing also provides a useful representation of state in this framework. Paskin shows how the prediction and update steps of EKF SLAM can be accomplished by message passing on a junction tree. Further, by enforcing a maximal *thickness* on the tree, the message passing can be performed in linear or constant time in the number of landmarks in the map. Thus the filter is called the thin junction tree filter (TJTF). The thickness of the tree corresponds to the size of the largest cluster in the tree. Within a cluster, all cross correlations are stored explicitly in the information form. Using variable elimination, clusters are adaptively resized and split, in order

to maintain the size bound. Each basic message passing event requires cubic computation in the size of the biggest cluster, so bounding the cluster sizes yields constant time operation per message. A straightforward implementation yields constant time operation when exploring new areas and linear time updates when closing loops. By further adaptively amortising message propagation over multiple time steps, operation can be always constant time. The resulting filter very closely approximates the EKF, while remaining at least as conservative as the EKF. Simulated results show mapping results equivalent to the EKF with greatly reduced computation.

The TJTF can be viewed either as an adaptive submapping scheme, or as a more rigorously-defined adaptive sparsification scheme similar to SEIF. In both cases, it has a more probabilistically sound motivation and approach, and guarantees at least as much consistency as the EKF. Unfortunately, as with Bayesian graphical models in general, application of the TJTF in loopy scenarios is unclear, and the algorithm cannot be trivially applied to large maps with many cycles.

Wang et al (2005) exploit the relative landmark representation to achieve a sparse information matrix. A small subset of landmarks (the "base") is represented with absolute coordinates, and all the other landmarks are represented in polar coordinates relative to the base. Then observations at each time step are partitioned, with a small set being used to update the pose estimates, and the rest used solely to update the relative landmark map. This decouples the relative landmark estimates from the vehicle estimate, and thus keeps the information matrix sparse. Validation on simulated data shows mapping results similar to EKF SLAM, with superior efficiency. However, some information is lost in the decoupling process. This effect is directly related to the ratio of dynamic noise to sensor noise, so for very precise sensor measurements, the loss is not significant.

Eustice et al (2006a) show that for delayed-state (also called view-based) SLAM, the information matrix is exactly sparse, allowing consistent constant-time operation. In delayed-state SLAM, there is no explicit map of the environment. Instead, past pose estimates (delayed states) are stored and updated. Observations constrain the transformations between these delayed states. In a temporal sequence of poses, each pose

is only directly dependent on the immediately previous and subsequent poses, and conditionally independent of all other pose estimates. The information form encodes this conditional independence property exactly, yielding a band-diagonal covariance matrix. Revisiting a previous pose corresponds to loop closure, and results in off-diagonal entries in the information matrix. As long as there are no more than a constant number of loop closures to any delayed state, the information matrix remains exactly sparse. As with the SEIF, the mean vector is not directly available, but can be recovered iteratively and approximately in constant time. A similar approach permits conservative recovery of the diagonal covariance block for the robot pose. **Eustice et al** (2006b) further develop the covariance recovery algorithm to allow access to covariance estimates for past states, yielding improved uncertainty bounds and aiding data association and recovery.

In order for this delayed-state algorithm to be well-conditioned, the pose change from one sensor observation to the next must be fully observable, as only these relative pose observations serve to constrain the state estimate. The authors show results for an underwater vehicle that uses both vision and odometry to perform mapping. While most of the sensor information comes from keypoint matching between pairs of images, the odometry is required to provide an estimate of the translation magnitude between frames. Without this, the observations do not constrain the scale of the trajectory from one observation to another, and the incremental estimation procedure is under-constrained.

2.3.5 Particle Filtering for SLAM

All of the SLAM algorithms discussed above choose a parametrised model to represent pose and landmark state – typically a high-dimensional Gaussian. Instead, all or part of the state can be estimates in a parameterless framework using a particle filter. Here we review important applications of particle filtering to the SLAM problem.

Dellaert et al (1999) describe a direct application of the CONDENSATION particle-filtering algorithm (**Isard & Blake** (1998)) to the localisation algorithm. An image

map of the ceiling of the robot's environment is built offline, and made available to the online system. The robot's only sensor is a brightness pointing toward the ceiling, giving a scalar observation of the brightness of the ceiling directly above the robot at each time step. The robot's pose estimate is represented by a collection of particles. Initially tens of thousands of pose particles are distributed uniformly over the environment. Each brightness observation, compared to the image map, yields a likelihood for each particle. The particles are reweighted according to this likelihood and their prior weight, and then resampled to yield the new pose distribution. The only odometry used by the system is a noisy estimate of total distance traveled over the last time step. Thus a closely packed group of particles, under the dynamic model prediction, becomes an annulus, which is then further constrained by the observation. While this is not a solution to the SLAM problem, as the map is known ahead of time, it is a good example of how a particle filter can naturally represent multi-modal, highly non-Gaussian distributions for localisation.

Kwok & Dissanayake (2003) present a straightforward application of particle filtering to SLAM estimation. Landmarks are vertical edges, with the robot moving in the 2D plane. Both the pose estimate and the landmark estimates are encoded by a large collection of particles (on the order of 10,000). A genetic algorithm is used to avoid sample impoverishment, wherein particles are not sufficiently densely distributed around a peaked likelihood function to allow an accurate representation of the posterior (described below in more detail). As the dimension of the state grows, so must the number of particles. Results are shown for simulated data using only 15 landmarks (each in two dimensions), so that the state has at most 33 dimensions. For the circular trajectories used, the map is recovered accurately, but the technique cannot scale well to larger maps.

Montemerlo et al (2002) make the useful observation that landmark estimates in SLAM are conditionally independent given the pose trajectory. If the trajectory is known with certainty, then SLAM is reduced to localisation, and landmarks are actually independent. The authors exploit this result using a modified particle filter, calling the resulting algorithm FastSLAM. Instead of parameterising the whole state (as in EKF

SLAM) or representing all of it by particles (as in **Kwok & Dissanayake** (2003)), FastSLAM samples the pose trajectory and maintains a distinct Gaussian map conditioned on each trajectory sample. This mixing of particles and parameterised random variables is called Rao-Blackwellisation, so FastSLAM is a Rao-Blackwellised particle filter. Each particle represents a certain sample of the trajectory from the beginning of the sequence, as well as a landmark map conditioned on the sample. At each time step, the particles are first moved according to the dynamic model. Because of the conditional independence property, the landmarks in a given map, parameterised as Gaussians, can be maintained in independent EKFs, making updates within each particle constant time. Landmarks are updated by their EKFs, particles are reweighted according to the likelihood of the observations, and finally resampled to give the prior for the next update. The resampling stage requires copying of landmark estimates, so a naive implementation requires linear time in the number of landmarks. However, a clever copy-on-write scheme reduces this to $O(\log N)$ time per particle. As all other operations are constant-time per particle, total computation per time step is $O(M \log N)$ for M particles and N landmarks. This makes FastSLAM unique among general SLAM algorithms. Convincing mapping results are shown for maps with tens of thousands of landmarks.

In general, particle filtering depends on the particle cloud sufficiently representing the distribution to be estimated, especially near the likelihood function. When this requirement is not satisfied, the resampled posterior is at best a poor approximation to the actual posterior, and at worst a completely degenerate representation where only one particle has survived. This problem is especially acute when the likelihood function is peaked relatively to the prior given by the dynamic model. Then the particles in the proposal distribution are spread out thinly, so that few are near the areas of high likelihood, and sample impoverishment results. FastSLAM 2.0 (**Montemerlo et al** (2003)) addresses this problem by explicitly moving the particles nearer to the peaked likelihood before resampling, by taking into account the latest observations. This results in better convergence and higher SLAM performance with fewer particles.

The particle filter representation of FastSLAM has the additional benefit of more flexible data association than traditional EKF SLAM, as shown by **Montemerlo & Thrun**

(2003). A different data association for each observation can be made by each particle, and incorrect associations will be automatically penalised by the resampling process. However, for this additional ambiguity to be statistically well-represented, the number of particles used must be significantly increased.

In fact, because FastSLAM is effectively compressing the covariance information of the map using a particle cloud, it is not clear that a fixed finite number of particles is sufficient to represent an arbitrarily growing map. **Bailey et al** (2006a) show that FastSLAM quickly becomes inconsistent as the size of the map increases, and suggests that actually the number of particles might need to grow exponentially with time. Nonetheless, the FastSLAM algorithm gives useful and accurate, if not consistent, maps for a variety of domains.

3

Mathematical Framework

This chapter presents the mathematical notation and framework used throughout the thesis.

3.1 Points and Vectors

Vectors are written as lower case boldface characters: $\mathbf{x}, \boldsymbol{\mu}$. A point in N-dimensional Euclidean space ($\mathbb{E}^N$) is represented either as a vector in $\mathbb{R}^N$ or as a homogeneous vector in $\mathbb{R}^{N+1}$, where vectors differing only by a scalar factor are equivalent:

$$\mathbf{x} \cong \lambda \cdot \mathbf{x}, \ \forall \lambda \in \mathbb{R} \tag{3.1}$$

In particular, a point in 3D space is represented as a Euclidean 3-vector or a homogeneous 4-vector, often with last coordinate normalised to 1:

$$\begin{pmatrix} x & y & z \end{pmatrix}^T \simeq \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T \tag{3.2}$$

Homogeneous coordinates permit a natural representation of direction vectors (as opposed to points) with last coordinate set to zero:

$$\begin{pmatrix} a & b & c & 0 \end{pmatrix}^T$$

3.2 Rigid Transformations

3.2.1 Basic Representation

Points and vectors are always specified with respect to a designated coordinate frame. A rigid transformation between two 3D frames, which preserves distances and angles between points, is represented as a 4×4 matrix operating on homogeneous vectors:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \tag{3.3}$$

In this matrix representation, $\mathbf{R}$ is a 3×3 rotation matrix, with $|\mathbf{R}| = 1$ and $\mathbf{R}^T\mathbf{R} = \mathbf{I}$, and $\mathbf{t}$ is a translation vector. The homogeneous linear transformation is a convenient representation of the equivalent rigid affine transformation in $\mathbb{R}^3$:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathbf{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \mathbf{t} \tag{3.4}$$

In this thesis, a transformation with rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ will occasionally be written $(\mathbf{R}, \mathbf{t})$ for compactness.

3.2.2 Lie Group Properties

Rigid transformations are members of the Lie group $\mathrm{SE}(3)$, with identity element given by $\mathbf{I}$, and group multiplication is naturally given by 4×4 matrix multiplication:

$$\left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right) = \left(\begin{array}{c|c} \mathbf{R}_1 & \mathbf{t}_1 \\ \hline \mathbf{0} & 1 \end{array}\right) \cdot \left(\begin{array}{c|c} \mathbf{R}_0 & \mathbf{t}_0 \\ \hline \mathbf{0} & 1 \end{array}\right) = \left(\begin{array}{c|c} \mathbf{R}_1\mathbf{R}_0 & \mathbf{R}_1\mathbf{t}_0 + \mathbf{t}_1 \\ \hline \mathbf{0} & 1 \end{array}\right) \tag{3.5}$$

Left multiplication of group elements is equivalent to composition of transformations, so that for $A, B \in \mathrm{SE}(3)$, the transformation $B \cdot A$ first takes a point through A, and then through B. The inverse of a transformation composes with itself to the identity:

$$\left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right)^{-1} = \left(\begin{array}{c|c} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right) \tag{3.6}$$

The group action of $\mathrm{SE}(3)$ on $\mathbb{R}^4$ is matrix-vector multiplication as shown above, and we will also use multiplication to denote the equivalent action on $\mathbb{R}^3$, where the vector is implicitly augmented with a 1 before multiplication and truncated afterwards.

The tangent space of an element of $\mathrm{SE}(3)$ is the 6D Lie algebra $\mathrm{se}(3)$, so any rigid transformation is minimally parameterised as a 6-vector in the tangent space of the identity element. The tangent vector $\begin{pmatrix} \mathbf{u}^T & \boldsymbol{\omega}^T \end{pmatrix}^T$, $\mathbf{u}, \boldsymbol{\omega} \in \mathbb{R}^3$, is mapped into $\mathrm{SE}(3)$ via the exponential map $\exp : \mathrm{se}(3) \to \mathrm{SE}(3)$, which has a closed-form expression. First define the 3×3 skew-symmetric matrix $[\boldsymbol{\omega}]_\times$ for $\boldsymbol{\omega} \in \mathbb{R}^3$:

$$[\boldsymbol{\omega}]_\times \equiv \begin{pmatrix} 0 & -\boldsymbol{\omega}_3 & \boldsymbol{\omega}_2 \\ \boldsymbol{\omega}_3 & 0 & -\boldsymbol{\omega}_1 \\ -\boldsymbol{\omega}_2 & \boldsymbol{\omega}_1 & 0 \end{pmatrix} \tag{3.7}$$

Then, defining terms and using the Rodrigues formula (**Gallier** (2001)):

$$\begin{aligned} A &\equiv \frac{\sin\theta}{\theta} \\ B &\equiv \frac{1-\cos\theta}{\theta^2} \\ \theta &\equiv \sqrt{\boldsymbol{\omega}^T\boldsymbol{\omega}} \\ \mathbf{R} &\equiv \mathbf{I} + A\,[\boldsymbol{\omega}]_\times + B\,[\boldsymbol{\omega}]_\times\,[\boldsymbol{\omega}]_\times \\ \mathbf{V} &\equiv \mathbf{I} + B\,[\boldsymbol{\omega}]_\times + \left(\frac{1-A}{\theta^2}\right)[\boldsymbol{\omega}]_\times\,[\boldsymbol{\omega}]_\times \\ \exp\begin{pmatrix}\mathbf{u}\\ \boldsymbol{\omega}\end{pmatrix} &= \left(\begin{array}{c|c} \mathbf{R} & \mathbf{V}\mathbf{u} \\ \hline \mathbf{0} & 1 \end{array}\right) \end{aligned} \tag{3.8}$$

The exponential map can be implemented efficiently for the case of small θ using truncated Taylor expansions of A and B.

There is similarly a closed form expression of the logarithm function $\ln : \mathrm{SE}(3) \to \mathrm{se}(3)$, which takes a group element to a tangent vector. More details about $\mathrm{SE}(3)$ and Lie groups and algebras can be found in **Varadarajan** (1974) and **Gallier** (2001).

Vectors in the tangent space can be mapped linearly from one coordinate frame to another using the adjoint. Consider $C = (\mathbf{R}, \mathbf{t}) \in \mathrm{SE}(3)$ and $\boldsymbol{\epsilon} \in \mathrm{se}(3)$, with

$$\mathrm{Adj}(C) \equiv \begin{pmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{pmatrix} \tag{3.9}$$

$$\boldsymbol{\delta} \equiv \mathrm{Adj}(C)\boldsymbol{\epsilon} \tag{3.10}$$

we have

$$\exp(\boldsymbol{\delta}) \cdot C = C \cdot \exp(\boldsymbol{\epsilon}) \tag{3.11}$$

3.2.3 Linearisation

Differentiating the exponential map about the origin ($\mathbf{u} = \mathbf{0}$, $\boldsymbol{\omega} = \mathbf{0}$) gives Jacobians for the left-multiplication scenario used throughout this thesis. Let

$$C \equiv \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right) \in \mathrm{SE}(3)$$

$$\mathbf{x} \equiv \begin{pmatrix} x & y & z & w \end{pmatrix}^T$$

Consider the transformation of $\mathbf{x}$ through C modified by left-multiplication:

$$\mathbf{y}(\mathbf{u}, \boldsymbol{\omega}, \mathbf{x}) \equiv \exp\begin{pmatrix} \mathbf{u} \\ \boldsymbol{\omega} \end{pmatrix} \cdot C \cdot \mathbf{x} \tag{3.12}$$

Then, differentiating by the parameters of the tangent space at C,

$$\frac{\partial \mathbf{y}}{\partial \mathbf{u}} = \left(\begin{array}{c} w \cdot \mathbf{I}_3 \\ \hline 0 \quad 0 \quad 0 \end{array}\right) \tag{3.13}$$

$$\frac{\partial \mathbf{y}}{\partial \boldsymbol{\omega}} = \left(\begin{array}{c} -\left(\mathbf{R}\begin{pmatrix} x & y & z \end{pmatrix}^T + w\mathbf{t}\right)_\times \\ \hline 0 \quad 0 \quad 0 \end{array}\right) \tag{3.14}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = C \tag{3.15}$$

3.2.4 Uncertainty

Often transformations are estimated from noisy data, and thus have uncertain parameters. This uncertainty can be modelled by a zero-mean Gaussian error distribution in

the tangent space around the mean transformation $\hat{C}$:

$$C = \exp(\boldsymbol{\epsilon})\,\hat{C} \tag{3.16}$$

$$\boldsymbol{\epsilon} \in \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \tag{3.17}$$

Here $\boldsymbol{\Sigma}$ is a 6×6 covariance matrix over the parameters of $\mathrm{se}(3)$. Note that the mean $\hat{C}$ need not be represented as an element of $\mathrm{se}(3)$ relative to the identity, but instead can be maintained in $(\mathbf{R}, \mathbf{t})$ form as a point on the manifold $\mathrm{SE}(3)$. Then the error distribution is represented in the tangent space, as a covariance matrix.

Because covariance is a first-order measure of error, it can be mapped from one coordinate frame to another using the adjoint. Given $B, C \in \mathrm{SE}(3)$, and zero-mean noise on C with covariance $\boldsymbol{\Sigma}_C$, the covariance matrix $\boldsymbol{\Sigma}_B$ in the frame given by B is computed using the adjoint of the transformation from C to B:

$$\boldsymbol{\Sigma}_B = \mathrm{Adj}\left(BC^{-1}\right) \cdot \boldsymbol{\Sigma}_C \cdot \mathrm{Adj}\left(BC^{-1}\right)^T \tag{3.18}$$

3.3 The Kalman Filter

As the Extended Kalman Filter (EKF) is a key component of many SLAM algorithms, and is employed throughout this thesis, we give a brief treatment of Kalman filtering here, with a special focus on the stochastic map of SLAM.

The seminal paper of **Kalman** (1960) continues to be a good reference for the basic linear case. There is a vast body of literature on Kalman filtering in general and its extensions; the introduction by **Welch & Bishop** (1995) provides an excellent starting point and informs the presentation and notation of this section.

3.3.1 Parameters

The Kalman filter estimates the state $\mathbf{x} \in \mathbb{R}^N$ of a process as it changes in discrete time steps, representing the state estimate by its mean $\hat{\mathbf{x}}$ and covariance $\mathbf{P}$. The pro-

cess is described by two linear stochastic equations for dynamics and observations, respectively:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{3.19}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{3.20}$$

Here $\mathbf{x}_{k-1}$ is the previous state, $\mathbf{x}_k$ is the current state, $\mathbf{u}_{k-1}$ is the control input (e.g. sent to motors) from the previous time step, and $\mathbf{z}_k$ is the latest observation vector (from the sensors). The matrices $\mathbf{A}$ (dynamic model), $\mathbf{B}$ (control model), and $\mathbf{H}$ (observation model) might change at each time step, or, for $\mathbf{H}$, with each observation. The noise in the dynamics and control process, called the *process noise*, is described by the random variable $\mathbf{w}_k$, while the noise in the observations, called the *measurement noise*, is described by the random variable $\mathbf{r}_k$. Both are assumed to be independent of each other (over time and observations), and to be drawn from white, zero-mean normal probability distributions:

$$\mathbf{w} \in \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{3.21}$$

$$\mathbf{r} \in \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{3.22}$$

The covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ parameterise the process and measurement noise respectively, and can vary with $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{H}$.

In Bayesian probabilistic terms, at each time step the filter calculates a prior distribution from the state and the dynamic model, multiplies this by a likelihood function given by sensors and the observation model, and stores the resulting posterior back into the state. Thus, at the end of time step k, the state parameters $(\hat{\mathbf{x}}_k, \mathbf{P}_k)$ represent the posterior estimate given all state estimates and observations up to the current time:

$$p(\mathbf{x}_k | \mathbf{x}_1 \dots \mathbf{x}_{k-1}, \mathbf{z}_1 \dots \mathbf{z}_k) \sim \mathcal{N}(\hat{\mathbf{x}}_k, \mathbf{P}_k) \tag{3.23}$$

Because this property holds recursively, this expression can be simplified to depend only on the previous state and current observations:

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k) \sim \mathcal{N}(\hat{\mathbf{x}}_k, \mathbf{P}_k) \tag{3.24}$$

3.3.2 Computation

The recursive computation splits naturally into two stages per time step. The *prediction* stage applies the dynamic model, predicting the current state based on the previous state and any control information, such as odometry. Thus prediction yields a prior state distribution for the current time step. Then the *update* stage fuses noisy measurements from sensors into the estimate, bringing the current estimate up-to-date given all observations. This corresponds to multiplying the prior by the likelihood of the observations, yielding a posterior. Because the state estimate, dynamics, and measurements are represented by Gaussian random variables, the prediction and update stages can be computed in closed form.

Using the dynamic model parameters described above, the prediction stage takes the estimate from last time step, $(\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1})$, to the current prior $(\tilde{\mathbf{x}}_k, \tilde{\mathbf{P}}_k)$:

$$\tilde{\mathbf{x}}_k = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} \tag{3.25}$$

$$\tilde{\mathbf{P}}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{3.26}$$

The update stage is more easily understood in terms of information (inverse covariance), where the information matrix quantifies how well the state estimate is constrained:

$$\mathbf{\Lambda} \equiv \mathbf{P}^{-1} \tag{3.27}$$

The updated information is the sum of the prior information and the information from the observation, transformed into state space:

$$\mathbf{\Lambda}_k = \tilde{\mathbf{\Lambda}}_k + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} \tag{3.28}$$

The updated mean is simply the information-weighted mean of the prior $\tilde{\mathbf{x}}_k$ and the measurement $\mathbf{z}_k$, transformed into state space:

$$\begin{aligned}\hat{\mathbf{x}}_k &= \frac{\tilde{\mathbf{\Lambda}}_k\tilde{\mathbf{x}}_k + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}_k}{\mathbf{\Lambda}_k} \\ &= \mathbf{P}_k\left(\tilde{\mathbf{\Lambda}}_k\tilde{\mathbf{x}}_k + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}_k\right)\end{aligned} \tag{3.29}$$

Transforming equation (3.28) into covariance form using the Woodbury matrix identity yields the standard Kalman covariance update equation:

$$\begin{aligned}\mathbf{P}_k &= \left(\hat{\mathbf{\Lambda}}_k + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1} \\ &= \tilde{\mathbf{P}}_k - \tilde{\mathbf{P}}_k\mathbf{H}^T\left(\mathbf{H}\tilde{\mathbf{P}}_k\mathbf{H}^T + \mathbf{R}\right)^{-1}\mathbf{H}\tilde{\mathbf{P}}_k \end{aligned} \tag{3.30}$$

Substituting (3.30) into (3.29), expanding, and simplifying yields the standard Kalman mean update equation:

$$\mathbf{S} \equiv \mathbf{H}\tilde{\mathbf{P}}_k\mathbf{H}^T + \mathbf{R} \tag{3.31}$$

$$\begin{aligned}\hat{\mathbf{x}}_k &= \left(\tilde{\mathbf{P}}_k - \tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{S}^{-1}\mathbf{H}\tilde{\mathbf{P}}_k\right)\left(\tilde{\mathbf{\Lambda}}_k\tilde{\mathbf{x}}_k + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}_k\right) \\ &= \tilde{\mathbf{x}}_k + \tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}_k - \tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{S}^{-1}\mathbf{H}\tilde{\mathbf{x}}_k - \tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{S}^{-1}\mathbf{H}\tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}_k \\ &= \tilde{\mathbf{x}}_k + \tilde{\mathbf{P}}_k\mathbf{H}^T\left(\mathbf{R}^{-1} - \mathbf{S}^{-1}\mathbf{H}\tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{R}^{-1}\right)\mathbf{z}_k - \tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{S}^{-1}\mathbf{H}\tilde{\mathbf{x}}_k \\ &= \tilde{\mathbf{x}}_k + \tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{S}^{-1}\left(\mathbf{z}_k - \mathbf{H}\tilde{\mathbf{x}}_k\right) \end{aligned} \tag{3.32}$$

Usually these update equations are written in the following form:

$$\mathbf{v} \equiv \mathbf{z}_k - \mathbf{H}\tilde{\mathbf{x}}_k \tag{3.33}$$

$$\mathbf{K} \equiv \tilde{\mathbf{P}}_k\mathbf{H}^T\mathbf{S}^{-1} \tag{3.34}$$

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}\mathbf{v} \tag{3.35}$$

$$\mathbf{P}_k = \tilde{\mathbf{P}}_k - \mathbf{K}\mathbf{H}\tilde{\mathbf{P}}_k \tag{3.36}$$

The matrix $\mathbf{K}$ is called the *Kalman gain*. The vector $\mathbf{v}$ and the matrix $\mathbf{S}$, known as the *innovation* and *innovation covariance* respectively, quantify how "surprising" the measurements are given the prior. That is, the likelihood of the observation is a function of the the two:

$$p(\mathbf{z}_k|\tilde{\mathbf{x}}_k) = \mathcal{N}\left(\mathbf{v}; \mathbf{0}, \mathbf{S}\right) \tag{3.37}$$

3.3.3 Extension to Nonlinear Models

The Kalman filter as defined above assumes linear dynamic and observation models, described by matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{H}$. When the dynamic or observation models are

nonlinear, the Kalman filter can be trivially extended by linearising the models around the current mean. The filter is then called the Extended Kalman Filter (EKF). Let the dynamic and observation models be defined by possibly nonlinear (but differentiable) functions $\mathbf{f}$ and $\mathbf{h}$ respectively:

$$\mathbf{x}_k = \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}\right) \tag{3.38}$$

$$\mathbf{z}_k = \mathbf{h}\left(\mathbf{x}_k, \mathbf{r}_k\right) \tag{3.39}$$

Then the state and measurements are predicted from their means, ignoring noise:

$$\tilde{\mathbf{x}}_k = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}\right) \tag{3.40}$$

$$\tilde{\mathbf{z}}_k = \mathbf{h}\left(\tilde{\mathbf{x}}_k, \mathbf{0}\right) \tag{3.41}$$

Control vector $\mathbf{u}$ and zero-mean noise vectors $\mathbf{w}$ and $\mathbf{r}$ are the same as defined above. The model matrices are defined as the appropriate differentials of the dynamic and observation models evaluated at the state mean:

$$\mathbf{A}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}\right) \tag{3.42}$$

$$\mathbf{H}_k = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\left(\tilde{\mathbf{x}}_k, \mathbf{0}\right) \tag{3.43}$$

The resulting EKF equations transform means through the nonlinear models and covariances through the linearisations. Hence the EKF prediction equations:

$$\tilde{\mathbf{x}}_k = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}\right) \tag{3.44}$$

$$\tilde{\mathbf{P}}_k = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{Q}_k \tag{3.45}$$

And the the update equations:

$$\mathbf{S} \equiv \mathbf{H}_k \tilde{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R}_k \tag{3.46}$$

$$\mathbf{v} \equiv \mathbf{z}_k - \tilde{\mathbf{z}}_k \tag{3.47}$$

$$\mathbf{K} \equiv \tilde{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{S}^{-1} \tag{3.48}$$

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k \oplus \mathbf{K}\mathbf{v} \tag{3.49}$$

$$\mathbf{P}_k = \tilde{\mathbf{P}}_k - \mathbf{K}\mathbf{H}_k \tilde{\mathbf{P}}_k \tag{3.50}$$

The $\oplus$ operator in (3.49) denotes the mean update operation, which need not be simple addition of vectors. For instance, the state might contain an element $C \in \mathrm{SE}(3)$ representing a pose in 3D, in which case the update corresponds to left-multiplication of C by the exponential of the update vector:

$$\hat{C}_k = \exp(\mathbf{K}\mathbf{v}) \cdot \tilde{C}_k \tag{3.51}$$

3.3.4 Example: Localisation

A simple example of localisation using the EKF is instructive before describing SLAM with the EKF. Consider a mobile robot constrained to move along the X-axis of the XY plane. At each time step, the robot makes a noisy measurement of the bearing to a fixed beacon located on the Y-axis at (0,1). Assume the robot motion is well-modeled by constant velocity with white-noise acceleration at each time step. Then the state of the system, which describes the robot's position and velocity along the X-axis, is two dimensional:

$$\mathbf{x}_k = \begin{pmatrix} x_k \\ v_k \end{pmatrix} \tag{3.52}$$

Each measurement is an offset in radians from the vertical, with positive offsets pointing to the right:

$$z_k = h(\mathbf{x}_k) = \arctan(-x_k) \tag{3.53}$$

For simplicity, assume no control input for the robot. Let the delay between time step $k-1$ and step k be $\triangle t_k$. Then, according to a constant velocity model, with acceleration of variance σ^2 distance units per time unit squared, we have prediction equations

$$\tilde{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{0}) = \begin{pmatrix} x_{k-1} + \triangle t_k v_{k-1} \\ v_{k-1} \end{pmatrix} \tag{3.54}$$

$$\mathbf{A}_k \equiv \begin{pmatrix} 1 & \triangle t_k \\ 0 & 1 \end{pmatrix} \tag{3.55}$$

$$\mathbf{Q}_k \equiv \sigma^2 \cdot \begin{pmatrix} \frac{1}{3}\triangle t_k^3 & \frac{1}{2}\triangle t_k^2 \\ \frac{1}{2}\triangle t_k^2 & \triangle t_k \end{pmatrix} \tag{3.56}$$

$$\tilde{\mathbf{P}}_k = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{Q}_k \tag{3.57}$$

The process noise $\mathbf{Q}_k$ is the covariance over position and velocity corresponding to a random walk in velocity for $\triangle t_k$ time, so it can change with each time step.

Differentiating the observation model,

$$\mathbf{H}_k = \begin{pmatrix} \frac{-1}{1+\tilde{x}_k^2} & 0 \end{pmatrix} \tag{3.58}$$

The observation matrix $\mathbf{H}$ is 1×2 because the measurements are 1D and the state is 2D. The second element is always zero because the measurements do not depend on the velocity of the robot.

Let the zero-mean noise associated with each bearing measurement of the beacon have standard deviation of r radians. Then we have the update equations:

$$\mathbf{S} \equiv \mathbf{H}_k \tilde{\mathbf{P}}_k \mathbf{H}_k^T + \left(r^2\right) \tag{3.59}$$

$$\mathbf{v} \equiv \left(z_k - \arctan\left(-\tilde{x}_k\right)\right) \tag{3.60}$$

$$\mathbf{K} \equiv \tilde{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{S}^{-1} \tag{3.61}$$

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}\mathbf{v} \tag{3.62}$$

$$\mathbf{P}_k = \tilde{\mathbf{P}}_k - \mathbf{K}\mathbf{H}_k \tilde{\mathbf{P}}_k \tag{3.63}$$

It is useful to investigate the actual output of the filter in operation. Set $\triangle t_k = 1$, $\sigma = 1.0$, and $r = 10^{-2}$, and let the actual evolution of x_k be given by

$$x_k = \frac{k^2}{1+k} - 4 \tag{3.64}$$

Set $\hat{\mathbf{x}}_0$ to the true starting state with zero uncertainty. Running the filter for ten time steps yields the output in Figure 3.1, where the true state (x_k and v_k) is shown next to the estimated state ($\hat{x}_k$ and $\hat{v}_k$), along with the standard deviations of the estimate, and finally the actual error in the measurement induced by noise. Note that the uncertainty in the position estimate shrinks as the robot moves closer to the origin, where the bearing measurements better constrain the state. As the robot moves away again, its position uncertainty starts to grow again. Also, even though the robot never makes direct measurements of its velocity, the filter gives a reasonable estimate of v_k.

k	x_k	$\hat{x}_k$	$\sqrt{\mathbf{P}_{1,1}}$	v_k	$\hat{v}_k$	$\sqrt{\mathbf{P}_{2,2}}$	$z_k - h(x_k)$
1	-4.000	-3.962	0.163	0.000	0.057	0.557	-0.002
2	-3.500	-3.356	0.160	0.750	0.679	0.616	-0.007
3	-2.667	-2.594	0.081	0.889	0.775	0.593	-0.009
4	-1.750	-1.747	0.043	0.938	0.862	0.557	-0.000
5	-0.800	-0.823	0.018	0.960	0.939	0.543	0.015
6	0.167	0.193	0.010	0.972	1.036	0.539	-0.026
7	1.143	1.152	0.025	0.980	0.939	0.539	-0.005
8	2.125	2.091	0.054	0.984	0.939	0.543	0.006
9	3.111	3.035	0.101	0.988	0.946	0.559	0.007
10	4.100	4.074	0.165	0.990	1.058	0.594	0.001

Table 3.1: Evolution of the real and estimated state over ten time steps

3.4 EKF SLAM

Having described the Kalman filter and EKF in general, and having walked through a simple concrete example, it is now worthwhile to consider the case of EKF SLAM. The presentation here does not assume specific models or parameters, but describes the general layout of the state and nature of the stochastic map updates. To reduce clutter, we omit the k time-step subscripts, as these will be clear from context.

3.4.1 State

In SLAM, the estimated state is a combination of the vehicle (or robot, or pose) state $\hat{\mathbf{x}}_v$ and the landmark states $\{\hat{\mathbf{x}}_i\}$. The mean vector consists of these sub-states stacked on top of each other, and the covariance matrix describes all of the correlations between vehicle and landmark states:

$$\hat{\mathbf{x}} = \left(\begin{array}{c} \hat{\mathbf{x}}_v \\ \hline \hat{\mathbf{x}}_1 \\ \vdots \\ \hat{\mathbf{x}}_N \end{array}\right) \qquad \mathbf{P} = \left(\begin{array}{c|ccc} \mathbf{P}_{v,v} & \mathbf{P}_{v,1} & \dots & \mathbf{P}_{v,N} \\ \hline \mathbf{P}_{1,v} & \mathbf{P}_{1,1} & \dots & \mathbf{P}_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{N,v} & \mathbf{P}_{N,1} & \dots & \mathbf{P}_{N,N} \end{array}\right) \tag{3.65}$$

Note that $\hat{\mathbf{x}}_v$ need not be represented in vector form. Often it will include quantities represented as in (3.51). Because $\mathbf{P}$ is symmetric, usually only its lower- or upper-

triangular half is stored.

3.4.2 Prediction

For all of the SLAM algorithms reviewed and presented in this thesis, the landmarks are considered static. Thus the motion model does not affect their state. Odometry and control parameters are omitted here, though they can be easily folded into the vehicle motion model. Under these assumptions, the EKF mean prediction is simple:

$$\tilde{\mathbf{x}}_v = \mathbf{f}\left(\hat{\mathbf{x}}, \mathbf{u}\right) \tag{3.66}$$

$$\tilde{\mathbf{x}}_i = \hat{\mathbf{x}}_i \qquad i \in \{1, \ldots, N\} \tag{3.67}$$

Again, because of the static model for landmark state, their uncertainty does not increase with time, so the covariance prediction stage leaves them alone, affecting only the vehicle covariances:

$$\tilde{\mathbf{P}}_{v,v} = \mathbf{A}\mathbf{P}_{v,v}\mathbf{A}^T + \mathbf{Q} \tag{3.68}$$

$$\tilde{\mathbf{P}}_{v,i} = \mathbf{A}\mathbf{P}_{v,i} \qquad i \in \{1, \ldots, N\} \tag{3.69}$$

$$\tilde{\mathbf{P}}_{i,j} = \mathbf{P}_{i,j} \qquad i, j \in \{1, \ldots, N\} \tag{3.70}$$

3.4.3 Update

In most SLAM scenarios, a bounded subset of the landmarks is observed at each time step. Observations can be fused into the filter either in serial or as a batch. Both approaches give the same result, up to linearisation. However, updating with one observation at a time is usually more efficient, so we present that approach here.

For an observation of a single landmark i, the observation model $\mathbf{h}(\mathbf{x})$ is a function only of the landmark and the vehicle state. Thus the observation Jacobian $\mathbf{H}$ is highly

sparse:

$$\mathbf{h}(\mathbf{x}) \equiv \mathbf{h}(\mathbf{x}_v, \mathbf{x}_i) \tag{3.71}$$

$$\mathbf{J}_v \equiv \frac{\partial \mathbf{h}}{\partial \mathbf{x}_v} \tag{3.72}$$

$$\mathbf{J}_i \equiv \frac{\partial \mathbf{h}}{\partial \mathbf{x}_i} \tag{3.73}$$

$$\mathbf{H}_i = \left(\begin{array}{c|ccccccc} \mathbf{J}_v & \mathbf{0} & \dots & \mathbf{0} & \mathbf{J}_i & \mathbf{0} & \dots \end{array} \right) \tag{3.74}$$

Expanding the innovation covariance $\mathbf{S}_i$ for landmark i shows that it involves only the state for the vehicle and the observed landmark:

$$\begin{aligned} \mathbf{S}_i &\equiv \mathbf{H}_i \tilde{\mathbf{P}} \mathbf{H}_i^T + \mathbf{R} \\ &= \mathbf{J}_v \mathbf{P}_{v,v} \mathbf{J}_v^T + \mathbf{J}_v \mathbf{P}_{v,i} \mathbf{J}_i^T + \mathbf{J}_i \mathbf{P}_{i,v} \mathbf{J}_v^T + \mathbf{J}_i \mathbf{P}_{i,i} \mathbf{J}_i^T \end{aligned} \tag{3.75}$$

Thus $\mathbf{S}_i$ is the projection into the measurement space of the covariance over the vehicle and landmark i.

The sparseness of $\mathbf{H}_i$ has important computational consequences for the update stage. Writing out equations (3.49) and (3.50),

$$\begin{aligned} \hat{\mathbf{x}}_v &= \tilde{\mathbf{x}}_v \oplus \left(\begin{array}{c|ccc} \tilde{\mathbf{P}}_{v,v} & \tilde{\mathbf{P}}_{v,1} & \dots & \tilde{\mathbf{P}}_{v,N} \end{array} \right) \mathbf{H}_i^T \mathbf{S}_i^{-1} \mathbf{v}_i \\ &= \tilde{\mathbf{x}}_v \oplus \left(\tilde{\mathbf{P}}_{v,v} \mathbf{J}_v^T + \tilde{\mathbf{P}}_{v,i} \mathbf{J}_i^T \right) \mathbf{S}_i^{-1} \mathbf{v}_i \end{aligned} \tag{3.76}$$

$$\begin{aligned} \hat{\mathbf{x}}_j &= \tilde{\mathbf{x}}_j \oplus \left(\begin{array}{c|ccc} \tilde{\mathbf{P}}_{j,v} & \tilde{\mathbf{P}}_{j,1} & \dots & \tilde{\mathbf{P}}_{j,N} \end{array} \right) \mathbf{H}_i^T \mathbf{S}_i^{-1} \mathbf{v}_i \\ &= \tilde{\mathbf{x}}_j \oplus \left(\tilde{\mathbf{P}}_{j,v} \mathbf{J}_v^T + \tilde{\mathbf{P}}_{j,i} \mathbf{J}_i^T \right) \mathbf{S}_i^{-1} \mathbf{v}_i \qquad j \in \{1, \dots, N\} \end{aligned} \tag{3.77}$$

Observing one landmark updates the means of all other landmarks through their covariance with the vehicle and the observed landmark, but the whole mean update is computed in $O(N)$ time due to the sparse Jacobian (instead of $O(N^2)$ time in the general case).

Expanding the covariance update:

$$\mathbf{P}_{v,v} = \tilde{\mathbf{P}}_{v,v} - \left(\tilde{\mathbf{P}}_{v,v} \mathbf{J}_v^T + \tilde{\mathbf{P}}_{v,i} \mathbf{J}_i^T \right) \mathbf{S}_i^{-1} \left(\mathbf{J}_v \tilde{\mathbf{P}}_{v,v} + \mathbf{J}_i \tilde{\mathbf{P}}_{i,v} \right) \tag{3.78}$$

$$\mathbf{P}_{v,j} = \tilde{\mathbf{P}}_{v,j} - \left(\tilde{\mathbf{P}}_{v,v} \mathbf{J}_v^T + \tilde{\mathbf{P}}_{v,i} \mathbf{J}_i^T \right) \mathbf{S}_i^{-1} \left(\mathbf{J}_v \tilde{\mathbf{P}}_{v,j} + \mathbf{J}_i \tilde{\mathbf{P}}_{i,j} \right) \qquad j \in \{1, \dots, N\} \tag{3.79}$$

$$\mathbf{P}_{j,k} = \tilde{\mathbf{P}}_{j,k} - \left(\tilde{\mathbf{P}}_{j,v} \mathbf{J}_v^T + \tilde{\mathbf{P}}_{j,i} \mathbf{J}_i^T \right) \mathbf{S}_i^{-1} \left(\mathbf{J}_v \tilde{\mathbf{P}}_{v,k} + \mathbf{J}_i \tilde{\mathbf{P}}_{i,k} \right) \qquad j,k \in \{1, \dots, N\} \tag{3.80}$$

Again, observing one landmark changes the covariance of the whole state, but because of the sparse Jacobian, each covariance block can be computed in $O(1)$ time, so the whole covariance update takes $O(N^2)$ instead of $O(N^3)$ time.

4

Monocular SLAM with a Rao-Blackwellised Particle Filter

4.1 Introduction

As discussed in 2.3, the EKF SLAM framework does not permit efficient mapping of many landmarks, because its storage and computation costs grow quadratically with the number of landmarks in the map. This quadratic cost is due to the full correlation of landmarks with each other as the map converges. Various approaches to reducing this computational complexity by partitioning the state into submaps (2.3.3) or exploiting approximate independence of distant landmarks (2.3.4) are reviewed above. One particle-filtering SLAM algorithm – FastSLAM (**Montemerlo et al** (2002), **Montemerlo et al** (2003)) – is unique with respect to both its exploitation of the probabilistic structure of SLAM, and its resulting computational efficiency.

This chapter describes the application of the FastSLAM algorithm to frame-rate SLAM with a single camera. The FastSLAM approach is attractive because it allows efficient filter operation with maps containing thousands of landmarks. However, the domain of monocular SLAM introduces significant challenges. Some of these challenges are tackled by **Davison** (2003), which describes single-camera SLAM with the EKF. The system described here adopts and adapts some of those techniques, while mapping, at frame rate, significantly more landmarks than the $\sim$100 possible with the EKF SLAM framework.

Most of the robotics SLAM literature, including that on FastSLAM, assumes odometry or control estimates which are reliable over short time periods, so that sensor observations are effectively used to correct drift in the odometry. If a single camera, moved by unknown external forces, is the only sensor, the system lacks any direct odometry, and the images captured at each time step must provide all of the information used for SLAM.

If FastSLAM is to be applied in this demanding setting, its particle filtering approach must be reconciled with the top-down active search framework that allows efficient image processing and reliable data association in a visual SLAM system. Additionally, initialisation of new landmarks is nontrivial due to the non-linear partial observation model of a single camera. Earlier solutions (**Davison** (2003), **Lemaire et al** (2005)) do not harness observations of such partially-initialised landmarks to help constrain the camera localisation estimate.

4.1.1 Contributions

The contributions presented in this chapter address the difficulties and shortcomings described above:

- Active search with the FastSLAM framework

- Inverse-depth linear-Gaussian estimation of partially initialised landmarks, allowing accurate initialisation and the use of such landmarks to constrain pose
- Frame-rate SLAM with hundreds to thousands of point landmarks

First, Section 4.2 describes the details of the general FastSLAM algorithm. Then, Section 4.3 explains the state representation and the dynamic and observation models of the system. Section 4.4 describes the recursive estimation process, including the active search framework and the update stage. Section 4.5 introduces the inverse depth parameterisation and presents the partial initialisation algorithm. Evaluation results and discussion are given in Section 4.6.

4.1.2 Previous Work

FastSLAM has been previously applied to vision-based SLAM by **Sim et al** (2005). The system takes a bottom-up approach to observation data association, building a large database of SIFT (**Lowe** (2004)) descriptors into which descriptors from novel views are matched. This approach precludes real-time operation of the system, which has an $O(N)$ processing time per frame of roughly 10s. Furthermore, Sim's system uses a stereo camera rig, which simplifies the observation model but does not match the flexibility, efficiency, and small footprint of a monocular system. In later work (**Elinas et al** (2006)), the data association algorithm is refined, but each frame still requires multiple seconds of processing time (350 ms for SIFT computation alone).

Particle filters that do not exploit the FastSLAM factorisation have also been applied to vision-based SLAM. The system of **Kwok & Dissanayake** (2003) (see 2.3.5) uses a particle filter to perform SLAM in a planar world by observing vertical edges with a camera. Odometry is available to the robot, and results are not shown for more than 15 landmarks.

Pupilli & Calway (2005) represent camera pose hypotheses with a particle cloud, while landmarks are represented communally. The correlations between landmark

and pose estimates and between pairs of landmarks are not maintained. The focus of the work is on robust camera localisation, so results with many landmarks are not shown. With 500 pose particles, the system operates at frame rate while observing four fiducial landmarks, but drops to below frame rate while observing eight landmarks.

Inverse depth coordinates have been suggested before; the early work of **Harris & Pike** (1988) employs a "disparity space" parameterisation, as it yields more Gaussian 3D estimates. The structure-from-motion algorithm of **Azarbayejani & Pentland** (1995) represents inverse focal length and inverse depth jointly in the optimisation.

Davison (2003) initialises the depth of each new landmark with a 1D particle filter, while the landmark's directional ray is estimated by the main EKF. The particles are distributed uniformly over a fixed depth range. Correlations between the landmark's depth estimate and other state variables are not maintained during the partial initialisation phase.

The work of **Lemaire et al** (2005) and **Sola et al** (2005) describes a multiple-hypothesis partial initialisation algorithm, where Gaussian estimates are distributed uniformly along inverse depth, in an attempt to approximate a uniform likelihood function in image disparity. See Section 4.5 for a comparison of these approaches to the method presented in this chapter.

4.2 FastSLAM

This section describes the FastSLAM algorithm in detail. Readers familiar with FastSLAM should skip to the next section. The original descriptions of **Montemerlo et al** (2002) and **Montemerlo et al** (2003) provide the basis for the notation and structure used here. Data association is assumed to be known by the filter, as the active-search observation framework described below provides it.

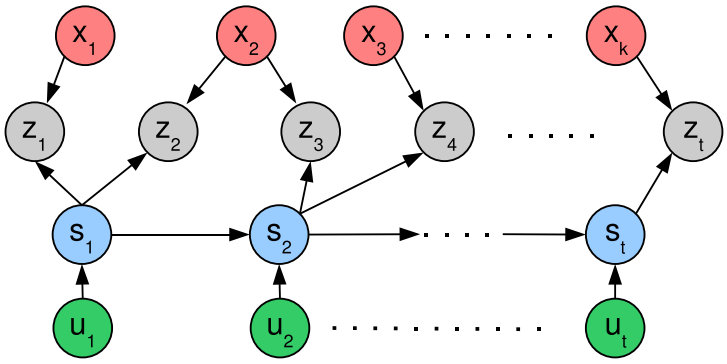

Figure 4.1: The Bayesian generative probabilistic model for SLAM. Poses $\{\mathbf{s}_1, \ldots, \mathbf{s}_t\}$ depend on the dynamic model $\{u_1, \ldots, u_t\}$, and observations $\{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$ depend on the poses and the landmarks $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$.

4.2.1 Conditional Independence in SLAM

A Bayesian analysis of the SLAM problem yields the generative probabilistic model shown in Fig. 4.1. The evolution of poses $\{\mathbf{s}_1, \ldots, \mathbf{s}_t\}$ over time depends on the motion model $\{u_1, \ldots, u_t\}$ (subsuming any odometry or control input). Landmark measurements $\{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$ are a function of the observed landmark $\mathbf{x}_k$ and the pose at the time of measurement. A key insight to be drawn from the model is that estimates of distinct landmarks are coupled only through the poses. If the poses are known with certainty, the landmarks can be estimated independently. Of course, perfect knowledge of the poses reduces SLAM to simple mapping, so this independence is not a surprise. By factoring the estimation process around this conditional independence, FastSLAM turns a single SLAM problem into a multiple hypothesis mapping problem, where each hypothesis can be estimated efficiently.

In probabilistic terms, as discussed in 2.1.1, SLAM is the problem of determining the posterior over pose and landmark parameters given the motion model and all observations up to the current time. Letting $\boldsymbol{\alpha}^t$ denote all instances of variable $\boldsymbol{\alpha}$ up to time t, the posterior can be written as a conditional distribution:

$$p\left(\mathbf{s}^t, \mathbf{x}^t | \mathbf{u}^t, \mathbf{z}^t\right) \tag{4.1}$$

The conditional independence of landmarks given poses implies that this expression factors exactly:

$$p\left(\mathbf{s}^t, \mathbf{x}^t | \mathbf{u}^t, \mathbf{z}^t\right) = p\left(\mathbf{s}^t | \mathbf{u}^t, \mathbf{z}^t\right) \prod_k p\left(\mathbf{x}_k | \mathbf{s}^t, \mathbf{u}^t, \mathbf{z}^t\right) \tag{4.2}$$

The factored form makes explicit the division of the estimation problem into the task of estimating the posterior over pose trajectory, and the task of estimating the landmark posteriors conditioned on the trajectory estimate. FastSLAM tackles the trajectory estimation using a modified particle filter, and estimates the landmarks, conditioned on the trajectory particles, with independent EKFs.

4.2.2 Trajectory Particle Filter

FastSLAM maintains a set S^t of M particles, where each $\mathbf{s}_m^t \in S^t$ is a sample from the pose trajectory distribution. Each particle represents a full trajectory hypothesis up to the current time (though only the latest pose needs to be stored). The trajectories are computed incrementally by sampling from the posterior over poses at each time step. The true posterior $p\left(\mathbf{s}_m^t | \mathbf{u}^t, \mathbf{z}^t\right)$, however, is not directly available, and is not representable in closed form.

Instead, the posterior is first approximated by a distribution q, called the *proposal distribution*. Then *importance sampling* is used to sample from the posterior by sampling $\{\mathbf{s}_m^t\}$ from q according to *importance weights* $\{w_m\}$. This process is an instance of sequential Monte Carlo sampling, explained in detail by **van der Merwe et al** (2000). The basic definitions given here are sufficient to understand the essential operation of FastSLAM.

The closer the proposal distribution q is to the true posterior, the better the results of importance sampling will be. Generating a good proposal for importance sampling in general is the subject of much research (**van der Merwe et al** (2000)). A simple proposal generation algorithm using the dynamic model, as used by **Montemerlo et al** (2002), and an improved algorithm, as recommended by **Montemerlo et al** (2003), are explained below. The latter is a natural extension of the former.

4.2.2.1 Simple Proposal

A simple proposal distribution is given by applying the dynamic model:

$$q_m \equiv p\left(\mathbf{s}_m^t | \mathbf{u}^t, \mathbf{s}_m^{t-1}\right) \tag{4.3}$$

The typical case is that the dynamic model, though perhaps nonlinear, has noise described by a zero-mean Gaussian distribution, as in the EKF. Then the proposal distribution $\{\tilde{\mathbf{s}}_m\}$ is represented by a mixture of Gaussians, where each component is generated by propagating the last pose $\mathbf{s}_{m,t-1}$ through the dynamic model $\mathbf{f}$, and setting the covariance of the result to the process noise $\mathbf{Q}$:

$$\tilde{\mathbf{s}}_m \in q_m \sim \mathcal{N}\left(\mathbf{f}\left(\mathbf{s}_{m,t-1}\right), \mathbf{Q}\right) \tag{4.4}$$

Each component of the mixture is weighted by the importance weight w_m. If M samples drawn from this mixture are to well-approximate samples from the true posterior, the $\{w_m\}$ must be calculated as the ratio of true posterior to the proposal distribution for each component:

$$w_m = \frac{p\left(\mathbf{s}_m^t | \mathbf{u}^t, \mathbf{z}^t\right)}{q_m} \tag{4.5}$$

An expansion using Bayes rule, given in full by **Montemerlo et al** (2002), shows that the importance weights, thus defined, are in fact proportional to the likelihood of the latest observations under the proposal distribution:

$$w_m \propto p\left(\mathbf{z}_t | \tilde{\mathbf{s}}_m, \mathbf{u}^t\right) \tag{4.6}$$

This result is intuitive: those trajectory samples that best accommodate the latest observations are the most likely samples of the true posterior. Samples are efficiently drawn from the mixture by assigning a number of "children" c_m for each $\tilde{\mathbf{s}}_m$ proportional to w_m, such that the total number of children is M. Then c_m samples are drawn from each mixture component $\tilde{\mathbf{s}}_m$, yielding the updated set of trajectory samples $\{\mathbf{s}_m^t\}$.

4.2.2.2 Improved Proposal

Using only the dynamic model to generate the proposal distribution, as in Eq. (4.3), gives a poor approximation of the posterior. This is especially true when the process noise significantly exceeds the measurement noise. Then the proposal distribution, for a fixed number of particles, is spread out by the dynamic model, while the observation likelihood is acutely peaked around the posterior. In these conditions, the importance weights for most proposal components are tiny, and with a finite number of particles, children will be sampled from only a very small set of the components – often just a single component.

In general, a superior proposal distribution is generated by taking into account the latest observations (**van der Merwe et al** (2000), **Montemerlo et al** (2003)). This is simply expressed by modifying Eq. (4.3) to include $\mathbf{z}_t$:

$$q_m \equiv p\left(\mathbf{s}_m^t | \mathbf{u}^t, \mathbf{s}_m^{t-1}, \mathbf{z}^t\right) \tag{4.7}$$

Representing the improved q_m requires incorporating $\mathbf{z}_t$ into the Gaussian mixture $\tilde{\mathbf{s}}_m$. The natural machinery for this task is again the EKF, which linearises the observation model $\mathbf{h}(\mathbf{x}, \mathbf{s})$ to fuse the measurements (with noise $\mathbf{R}$) into each component. Note that there is a separate EKF for each mixture component (corresponding to a particle), and that the state of each EKF consists only of the pose state of the proposal component. While the simple proposal distribution of Eq. (4.4) performs the prediction step of the EKF, the improved proposal also includes the measurement update step:

$$(\hat{\mathbf{s}}_m, \mathbf{\Sigma}_m) = \mathrm{EKF}\left(\mathbf{s}_{m,t-1}, \mathbf{f}, \mathbf{Q}; \mathbf{z}_t, \mathbf{h}, \mathbf{R}\right) \tag{4.8}$$

$$\tilde{\mathbf{s}}_m \in q_m \sim \mathcal{N}\left(\hat{\mathbf{s}}_m, \mathbf{\Sigma}_m\right) \tag{4.9}$$

By updating the mixture components using the latest observations, the proposal distribution is "moved" closer to the peaked measurement likelihood, and the component-wise uncertainties $\{\mathbf{\Sigma}_m\}$ are reduced.

Note that the only difference between the simple and improved proposal-generation algorithms is the use of the latest observations. For an accurate posterior, the importance weights w_m (which determine c_m), should still be proportional to the likelihood

of the observations under the *unimproved* proposal. That is, the importance weights are identical for the simple and improved proposals.

4.2.3 Landmark Update

Once the trajectory particles have been resampled, yielding S^t, the landmark estimates conditioned on each $\mathbf{s}_m^t \in S^t$ can be updated based on the latest observations. Note that because S^t contains trajectory samples, instead of trajectory estimates, there is no uncertainty associated with the pose of each $\mathbf{s}_m^t$. Thus the landmark estimates conditioned on each particle are independent. Also, even though the latest observations may have been used to obtain a good sample set S^t (as in the improved proposal algorithm), the same observations may be used again to update the landmark estimates associated with each sample, without violating the stochastic correctness of the filter. This follows from the factorisation of Eq. (4.2).

Let the estimate of landmark i associated with trajectory particle m be $\mathbf{x}_{i,m}$. Each landmark for each particle is estimated by its own EKF, so its state is given by mean and covariance in the landmark parameters:

$$\mathbf{x}_{i,m} \sim \mathcal{N}\left(\hat{\mathbf{x}}_{i,m}, \mathbf{P}_{i,m}\right) \tag{4.10}$$

For every particle, for each observed landmark, the landmark estimate is updated by the EKF (without dynamics) using the observation model $\mathbf{h}_{i,m}\left(\mathbf{x}_{i,m}^{t-1}, \mathbf{s}_m^t\right)$:

$$\left(\hat{\mathbf{x}}_{i,m}^t, \mathbf{P}_{i,m}^t\right) = \mathrm{EKF}\left(\hat{\mathbf{x}}_{i,m}^{t-1}, \mathbf{P}_{i,m}^{t-1}, \mathbf{s}_m^t, \mathbf{z}_{i,t}, \mathbf{h}_{i,m}, \mathbf{R}_{i,t}\right) \tag{4.11}$$

4.2.4 Computational Cost

Consider a FastSLAM system with M particles and N landmarks, with $O(1)$ landmarks observed at each time step. The trajectory resampling step of FastSLAM involves first applying the dynamic model to each particle, then improving the resulting

proposal components using the latest observations, and finally importance sampling from the improved proposal distribution.

Application of the dynamic model requires $O(1)$ time per particle, so $O(M)$ time in total. Improving the proposal components using the EKF update also requires constant time per particle, as there are constantly many landmark observations, and the state dimension of each particle's pose EKF is constant. So this stage also requires $O(M)$ time in total.

The importance sampling stage involves the computation of the importance weights $\{c_m\}$ and the sampling of children particles. The $\{c_m\}$ can be efficiently computed by factoring the observation likelihood. Consider simultaneous observations $\mathbf{z}$ of k landmarks:

$$\mathbf{z} \equiv \begin{pmatrix} z_1 & \dots & z_k \end{pmatrix}^T \tag{4.12}$$

The overall observation likelihood is factored by conditioning on each landmark measurement in turn:

$$\begin{aligned} \alpha_m &\equiv \{\tilde{\mathbf{s}}_m, \mathbf{u}^t\} \\ p\left(\mathbf{z}|\alpha_m\right) &= p\left(z_1, \dots, z_k|\alpha_m\right) \\ &= p\left(z_1, \dots, z_{k-1}|\alpha_m, z_k\right) p\left(z_k|\alpha_m\right) \\ &\quad \vdots \qquad\qquad\qquad\qquad \ddots \\ &= p\left(z_1|\alpha_m, z_2, \dots, z_k\right) p\left(z_2|\alpha_m, z_3, \dots, z_k\right) \dots p\left(z_k|\alpha_m\right) \end{aligned} \tag{4.13}$$

For each particle, each factor in the expression can be computed in constant time just after the corresponding landmark observation is fused into the proposal by the EKF. Thus the importance weight computation for M particles and k simultaneous landmark observations takes $O(Mk)$ time.

The resampling step, where new particles are drawn from the proposal mixture, is the most expensive part of the trajectory update. The main cost is copying the conditional landmark estimates associated with each particle. In a naive implementation, all N estimates need to be copied for each of the M sampled particles, requiring $O(MN)$ computation.

However, only the estimates of the $O(1)$ observed landmarks will actually be modified in the landmark update step. This can be exploited by storing the landmark estimates for each particle as leaves in a balanced binary tree, with landmark IDs providing the keys. Only those nodes on the path from the root of the tree to the observed landmark need to be copied, while pointers to the other nodes and leaves are unchanged. Using this copy-on-write scheme, only $O(\log N)$ nodes need to be copied for each sampled particle, resulting in an overall resampling cost of $O(M \log N)$.

The landmark update stage modifies constantly many landmark estimates for each particle, requiring $O(M)$ time in total. Thus the dominating cost of FastSLAM is the resampling stage, so the algorithm runs in $O(M \log N)$ time while requiring $O(MN)$ storage. For reasonable M, FastSLAM is qualitatively more efficient than EKF SLAM, which requires $O(N^2)$ update computation and storage.

4.3 System Model

This section describes the estimated state of the monocular SLAM system, and the dynamic and observation models employed by the filter.

4.3.1 State

The SLAM system described here models a single moving camera capturing 640×480 greyscale images of a static environment at 30Hz. The camera has full six-degree-of-freedom motion, so its configuration at any time instant is represented as a pose $C \in \mathrm{SE}(3)$. The environment is assumed to be sufficiently populated with locally-planar patches of texture, which supply the observations. The filter estimates the 3D positions $\{\mathbf{x}_i\}$ of the centre points of these patches, in a common (but arbitrary) Euclidean coordinate frame.

Thus the estimated state describes a distribution over the pose C of the camera and

the positions of landmarks $\{\mathbf{x}_i\}$. The uncertainty in pose is represented directly by the poses of M particles $\{C_m\}$, while the uncertainty over landmark parameters $\{\mathbf{P}_{i,m}\}$ is given with respect to each particle's trajectory hypothesis. The overall landmark mean and covariance could be recovered by taking expectations over all particles:

$$\begin{aligned}\hat{\mathbf{x}} &= \mathrm{E}\left[\mathbf{x}\right] \\ &= \frac{1}{M}\sum_m \mathbf{x}_m \end{aligned} \tag{4.14}$$

$$\begin{aligned}\mathbf{P} &= \mathrm{E}\left[\mathbf{x}\mathbf{x}^T\right] - \mathrm{E}\left[\mathbf{x}\right]\mathrm{E}\left[\mathbf{x}\right]^T \\ &= \frac{1}{M}\sum_m \left[\begin{pmatrix}\mathbf{P}_{1,m} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{P}_{N,m}\end{pmatrix} + \mathbf{x}_m\mathbf{x}_m^T\right] - \hat{\mathbf{x}}\hat{\mathbf{x}}^T \end{aligned} \tag{4.15}$$

The number of landmarks changes during the course of mapping; such variations are accommodated by adding or removing landmarks' Gaussian estimates from all of the particles.

4.3.2 Dynamic Model

The camera is assumed to move under a constant velocity model with white-noise acceleration. Because each particle represents a *certain* trajectory, the current velocity of a particle's pose can be directly computed at the end of each time step, by assuming that the camera will continue to move with the average velocity displayed over the last time step $\triangle_t$.

$$\mathbf{v}_m^t = \frac{1}{\triangle_t}\ln\left(C_m^t \cdot C_m^{t-1}\right) \tag{4.16}$$

This gives a straightforward dynamic model for each particle, taking the sampled pose from the end of the last time step to a Gaussian pose estimate $(\hat{C}, \Sigma)$ at the current time:

$$\hat{C}_m^{t+1} = f(C_m^t, \mathbf{v}_m^t) = \exp\left(\triangle_{t+1}\mathbf{v}_m^t\right) \cdot C_m^t \tag{4.17}$$

$$\Sigma_m^{t+1} = \frac{1}{3}\triangle_t^3 \begin{pmatrix}\sigma_1^2 & & \\ & \ddots & \\ & & \sigma_6^2\end{pmatrix} \tag{4.18}$$

The process noise, identical for all particles at a given time step, is characterised by $\boldsymbol{\sigma} \equiv \begin{pmatrix}\sigma_1 & \dots & \sigma_6\end{pmatrix}^T$, the standard deviation of acceleration in the six coordinates of the tangent space.

4.3.3 Observation Model

The intrinsic camera parameters, including those which account for nonlinear effects such as radial distortion, are assumed to be known from offline calibration, and fixed during online operation. The function taking a point $\begin{pmatrix} u & v & 1 \end{pmatrix}^T$ in the metric camera plane to image pixel coordinates is denoted cam. Though cam might not have a closed form inverse, if it is differentiable, its inverse cam^{-1} can be efficiently approximated using the Newton-Rapheson algorithm with a small constant number of iterations.

In order to simplify all of the observation mathematics, noisy 2D measurements from the image are first transformed through cam^{-1} into the camera plane. Such measurements are characterised by a normal distribution in pixel coordinates; the mean and covariance can be transformed using either a Taylor expansion of cam^{-1} around the mean, or the unscented transform of **Julier & Uhlmann** (1997a). Note that, with radial distortion or any other nonlinear effects, isotropic noise in pixel coordinates might transform to anisotropic noise in the camera plane.

In this calibrated, camera plane framework, the observation model maps Euclidean landmark coordinates $\mathbf{x}_i \in \mathbb{R}^3$ into a local camera frame (represented by $C = (\mathbf{R}_C, \mathbf{t}_C) \in \mathrm{SE}(3)$), and then perspective-projects the result onto the camera plane in that frame:

$$\begin{aligned} \mathbf{h}(\mathbf{x}_i, C) &= \mathrm{project}\left(C \cdot \mathbf{x}_i\right) \\ &= \mathrm{project}\left(\mathbf{R}_C \mathbf{x}_i + \mathbf{t}_C\right) \end{aligned} \tag{4.19}$$

$$\mathrm{project}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} \tag{4.20}$$

The Jacobians of the Euclidean point observation model, computed using the chain rule, are made explicit in Section A.1.1.

4.4 Recursive Estimation

At each time step, after a frame is captured from the camera, three stages of computation take place: prediction, observation, and updates. In the prediction stage, the set of certain pose samples from the previous time step is transformed into a Gaussian mixture of poses according to the dynamic model of Section 4.3.2. Using this predicted pose distribution, and the landmark estimates conditioned on each particle, landmark observations are extracted from the new frame through active search. The observations are then used both to refine the proposal distribution, from which new particles are sampled, and also to update the landmark estimates conditioned on each particle. Thus, at the end of processing for each frame, the state is again represented by pose samples with associated independent Gaussian landmark estimates.

4.4.1 Active Search for Observations

In a general SLAM scenario, observations come from an abstract sensor, and the influx of observations does not depend on the estimation machinery. When the sensor is a camera, and the observations are correspondences between landmarks and image locations, this becomes a bottom-up framework, as employed by **Sim et al** (2005). A bottom-up approach requires landmarks with highly distinct and reliable appearances, and makes the task of data association difficult and inefficient.

In contrast, a top-down approach to observing landmarks takes advantage of the existing information about the camera and the landmarks throughout the observation process. Observations are made by actively searching new frames for specificlandmarks. The search regions are determined by the current estimates of camera pose and landmark locations, and by the uncertainty in these estimates. In an EKF SLAM system like **Davison** (2003), the search region for a landmark is simply the landmark's uncertain Gaussian estimate projected into the image, and gated by likelihood. This active search approach has two important benefits: efficiency and improved data association.

Because only a small region of the image is examined for a particular landmark, the search computation is reduced, and furthermore, the region (and thus computation) shrinks with increasing frame rate. Moreover, the system knows something about how the landmark should "look": from the current estimate of camera pose and landmark parameters, the landmark's appearance can be approximated to make the search more accurate. Instead of a match needing to be unique across the whole image, it needs to be unique only within the constrained search region. And because a specific landmark is sought, a successful search is also a successful data association.

The active search method used by **Davison** (2003) is straightforward in the setting of EKF SLAM. However, with a more complex pose distribution, and distinct landmark estimates for each pose hypothesis, a slightly different strategy for searching the image must be adopted. There are multiple possible approaches for taking into account the whole distribution of poses and landmark parameters for the active search. A simple but effective one is described here.

First, the mean of the pose distribution is computed. Because the poses are distributed over the manifold $\mathrm{SE}(3)$, the mean is defined as the pose which has minimal sum of squared distances (in the tangent space) to the particle poses:

$$\hat{C} \equiv \arg\min_{C} \left\{ \sum_{m} \| \ln(C \cdot C_m^{-1}) \|^2 \right\} \tag{4.21}$$

Using $\mathrm{Adj}(C)$, the minimum can be efficiently computed from most pose distributions with a small number of Gauss-Newton iterations.

Landmarks can be tested for possible visibility by checking if their mean estimated positions fall into the viewing frustum of the mean pose. A simple scan through all landmarks takes $O(N)$ time (though with small constant); for large N, a visibility data structure over landmark positions can help constrain the visibility test.

Landmarks that might be visible in the current frame are candidates for observations. For each such candidate, the M Gaussian estimates of the landmark under all particles are projected into the image, and the mean and covariance of this set of projected

distributions is computed. This yields a single Gaussian estimate of landmark location in the image. The landmark should appear within the corresponding 3σ ellipse in the image with high likelihood.

A landmark's appearance is described by a small image patch, grabbed from the image where the landmark is first observed. This patch will transform roughly according to an affine homography, $\mathbf{A}$, a function of the pose displacement from the first observation. Without knowledge of the patch normal, the skew and shear of $\mathbf{A}$ cannot be estimated, so the homography is constrained to rotation and scaling. The stored patch is warped according to $\mathbf{A}$, which is computed as a function of the mean pose and the landmark's mean estimate. Predictive warping allows landmarks to be observed over significant viewpoint changes.

The location inside the search ellipse yielding maximal zero-normalised cross correlation (ZNCC) with the warped patch is taken as an observation of the landmark if the ZNCC score is above a threshold. Use of ZNCC gives invariance to affine lighting changes. A quadratic form is fitted to the local ZNCC surface around a match in order to approximate the measurement noise. If the measurement noise is high, the match location is poorly constrained, possibly due to repeated texture or an edge-like appearance. Thus if the square root of largest eigenvalue of the measurement noise covariance exceeds a threshold (default is 4 pixels), the observation is discarded as a failure.

4.4.2 Pose and Landmark Updates

Given a set of observations from the latest image, the Gaussian mixture of poses can be improved, as described above (4.2.2.2). The Jacobians (see Section A.1) are used to perform repeated EKF updates of each particle's Gaussian pose estimate, using each landmark observation in turn. Then M new particles are sampled from the resulting mixture.

The system evaluated here does not implement the FastSLAM tree data structure for

achieving $O(M \log N)$ computation in the resampling stage. Instead it uses reference-counted shared pointers to manage landmark estimates, so that each new particle involves $O(N)$ copying, but only of the pointers (and not the landmark state). The copying cost, for $M{\sim}100$ and $N{\sim}1000$, is not significant compared to the rest of the update process, though the more efficient data structure should be used for larger N.

Once the new particles are sampled, the same observations are used to update the landmark estimates conditioned on each particle's certain pose. These EKF updates, each in three dimensions, require differentiating the observation model only by the landmark parameters.

4.4.3 Map Management

During operation, landmarks are acquired from images on demand, using the procedure described below (4.5), so as to maintain a minimum number of visible landmarks at all times. Because the filter is capable of efficiently handling maps with many landmarks, more than 40 landmarks can be observed in each frame without exceeding computation bounds. Thus, when the camera is exploring unmapped territory, landmarks will be acquired regularly and rapidly. If the camera remains in known regions with a sufficiently dense map, no landmarks will be acquired. This is a much less careful and more aggressive policy than that of **Davison** (2003), as there is minimal performance penalty for mapping more landmarks, while observing more landmarks each frame helps pose tracking.

During the motion of the camera, landmarks may be occluded, or their appearance may not match the static, locally-planar model assumed by the active search framework. Attempts to observe such landmarks will fail. When the ratio of failures to successes exceeds a threshold ($\frac{2}{3}$ by default), the landmark is dropped from the map by removing its estimates from all particles and freeing all resources related to its representation.

4.5 Landmark Initialisation

In the above framework, landmark estimates for each particle are represented as 3D Gaussians. With a single camera, the depth of a landmark observed only once is known only up to sign (it must lie in front of the camera). Even when observed over many time steps, the distribution of the landmark's position in Euclidean coordinates is poorly approximated by a Gaussian. Such a landmark is said to be *partially initialised*, and it must be carefully represented in the map until its estimate can be represented as a Gaussian in Euclidean space.

In his monocular EKF SLAM system, **Davison** (2003) maintains a set of depth hypotheses uniformly distributed in a fixed depth range – a particle filter in one dimension. Every observation of the partially initialised landmark is used to update the depth filter, until the distribution of depth particles is roughly Gaussian. Then the estimate is added to the map as a 3D Gaussian. Until this full initialisation occurs, the EKF maintains an estimate of the landmark's ray, a 5D Gaussian in global coordinates computed from the first observation. Subsequent observations of the partially initialised landmark affect neither the ray nor the camera pose estimate.

Sola et al (2005) and **Lemaire et al** (2005) distribute 3D Gaussian landmark hypotheses uniformly in inverse depth along the ray, as this arrangement corresponds to constant density of hypotheses in the camera plane. As new measurements are made, unlikely hypotheses are pruned, until only one remains. Then the new landmark is fully initialised (in a central EKF) using the sole survivor hypothesis as a starting point. Until full initialisation, the landmark observations do not factor into the camera pose estimate, and the strong correlations between the partially initialised landmark's estimate and other landmarks are not maintained.

One of the primary contributions of this chapter is a solution to the problem of partial initialisation within the FastSLAM framework that also allows observations of a partially initialised landmark to help constrain the camera pose during the estimation

process. The key to the solution is the use of inverse depth coordinates to parameterise partially initialised landmarks. After the initial work of this chapter was published (**Eade & Drummond** (2006b)), an extensive study of the use of inverse depth parameterisations in vision SLAM was published by **Montiel et al** (2006). A 6D parameterisation, which subsumes coordinate frame parameters, is incorporated into the latest version of Davison's MonoSLAM system (**Davison et al** (2007)).

4.5.1 Inverse Depth Coordinates

The perspective projection function of Eq.(4.20) takes Euclidean points in the camera frame to the camera plane by dividing their x and y coordinates by the z coordinate. This division step makes the function highly nonlinear, so that a Gaussian distribution in the camera frame does not transform to a Gaussian in the camera plane. Thus measurements with Gaussian uncertainty in the camera plane will not combine to give a Gaussian estimate in the camera frame. However, modifying the parameterisation of points in the camera frame makes the projection function linear.

Let $\begin{pmatrix} x & y & z \end{pmatrix}^T$ represent a 3D point in Euclidean coordinates. The same point can be described in inverse depth coordinates:

$$\begin{pmatrix} u \\ v \\ q \end{pmatrix} \equiv \begin{pmatrix} x/z \\ y/z \\ 1/z \end{pmatrix} = \frac{1}{z} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4.22}$$

This representation can also be derived from a point in homogeneous coordinates by normalising the third coordinate to 1, instead of the fourth coordinate as in the Euclidean case:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \sim \begin{pmatrix} x/z \\ y/z \\ 1 \\ 1/z \end{pmatrix} = \begin{pmatrix} u \\ v \\ 1 \\ q \end{pmatrix} \tag{4.23}$$

In these coordinates, the projection function from camera frame to camera plane is

exactly linear (and independent of q):

$$\text{inverse\_depth\_project} \begin{pmatrix} u \\ v \\ q \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} \tag{4.24}$$

This linearity implies that a Gaussian measurement in the 2D camera plane can be fused with a Gaussian estimate in the 3D camera frame using the Kalman filter to give a Gaussian posterior in the camera frame.

However, the camera is moving, so the observation model must account for transformation between coordinate frames according to the camera motion. In the case of Euclidean coordinates, this transformation has a constant Jacobian by the point parameters (namely, the rotation matrix). But with inverse depth coordinates, the transformation is not so simple. Consider a camera motion given by $C = (\mathbf{R}, \mathbf{t}) \in \mathrm{SE}(3)$. Using the homogeneous representation (4.23),

$$C \cdot \begin{pmatrix} u \\ v \\ q \end{pmatrix} = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right) \begin{pmatrix} u \\ v \\ 1 \\ q \end{pmatrix} = \begin{pmatrix} \mathbf{R} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + q\mathbf{t} \\ q \end{pmatrix} \tag{4.25}$$

This can be converted to Euclidean coordinates in the camera frame by dividing by the fourth coordinate, q, though this (or any other) scaling has no effect on the projection onto the camera plane. The total inverse depth observation model $\mathbf{h}^*$, with landmark coordinates $\mathbf{x}^* \equiv \begin{pmatrix} u & v & q \end{pmatrix}^T$ observed under camera displacement C, is the composition of camera frame transformation and perspective projection:

$$\begin{aligned} \mathbf{h}^*(\mathbf{x}^*, C) &= \text{project}(C \cdot \mathbf{x}^*) \\ &= \text{project}\left(\frac{1}{q}\left(\mathbf{R}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + q\mathbf{t}\right)\right) \\ &= \text{project}\left(\mathbf{R}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + q\mathbf{t}\right) \end{aligned}$$

Carrying through the projection, with $\mathbf{D} \equiv \mathbf{R}\begin{pmatrix} u & v & 1 \end{pmatrix}^T$,

$$\mathbf{h}^*(\mathbf{x}^*, C) = \frac{1}{\mathbf{D}_3 + q\mathbf{t}_3}\begin{pmatrix} \mathbf{D}_1 + q\mathbf{t}_1 \\ \mathbf{D}_2 + q\mathbf{t}_2 \end{pmatrix} \tag{4.26}$$

From this expression it is clear that $\mathbf{h}^*$ is nearly linear in the coordinates u, v, and q as long as the camera rotation remains mostly around the optical axis (so $\mathbf{D}_3 \approx 1$) and the displacement along the optical axis is small relative to the depth of the point (so $q\mathbf{t}_3 \approx 0$).

The camera displacement C is expressed relative to a fixed coordinate frame C_0 in which the landmark estimate is represented. The natural choice for C_0 is the pose from which the landmark is first observed, as subsequent camera poses will be nearby, making the observation model nearly linear. Within the FastSLAM framework, the camera pose at the end of a time step is represented independently in each particle with certainty. The partial initialisation algorithm saves this first pose as the coordinate frame for the inverse depth parameterisation. The Jacobians of the inverse depth observation model are shown in Section A.1.2.

The near-linearity of the inverse depth observation model implies that the EKF, or its inverse form the EIF, can be employed to estimate $\mathbf{x}_*$ in the local pose neighbourhood of the first sighting of the landmark, as the distribution of the estimate is nearly gaussian. In contrast, the distribution of Euclidean coordinates is a cone with apex at the camera centre and altitude along the optical axis, and a single Gaussian approximation of this distribution is poor. Figure 4.2 shows the evolution of the posterior over depth and over inverse depth, as a new landmark is repeatedly observed. Both distributions become more peaked and Gaussian as observations are made. However, only the inverse depth estimate is well-approximated by a Gaussian at all stages.

4.5.2 Initialisation Process

The landmark initialisation process can now be summarised:

1. **Landmark selection**: A new landmark is created by selecting an interest point (u_0, v_0) from the current image.

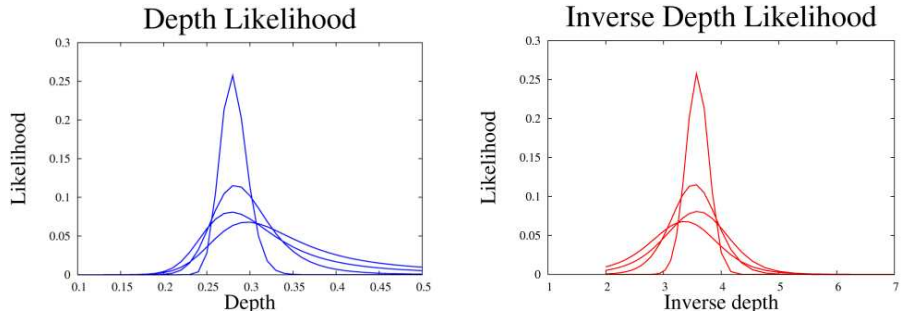

Figure 4.2: Depth and inverse depth estimates for successive observations of a new landmark. The depth estimate is not Gaussian when the depth is uncertain. The inverse depth estimate is always Gaussian

2. **Map augmentation**: The landmark's inverse depth estimate, $\begin{pmatrix} u_0 & v_0 & 1 \end{pmatrix}^T$, is appended to every particle, with covariance given by the localisation noise of the keypoint (and infinite uncertainty in the third coordinate). Each particle's current camera pose is stored with the landmark estimate as C_0.

3. **Re-observation**: Through active search, the partially initialised landmark is observed in subsequent images. Just as for fully initialised landmarks, such observations are used to improve the proposal distribution, and then to update the landmark's per-particle inverse depth estimates with the EKF.

4. **Full initialisation**: When the landmark's inverse depth landmark estimate for a given particle becomes sufficiently Gaussian in Euclidean coordinates, the estimate is transformed into global Euclidean coordinates. The landmark is henceforth fully initialised with respect to that particle.

4.5.2.1 Landmark Selection

The appearance model for landmarks is an image patch that is assumed to be locally planar and will be localised according to the algorithm described above (4.4.1). New landmarks should be chosen with this task in mind (**Shi & Tomasi** (1994)). In practice, most interest point detectors will suffice, as the appearance model with warping and

normalised cross correlation is invariant enough to viewpoint changes that tracking succeeds over reasonable motions.

However, efficiency is a primary concern, as new landmarks might be acquired in any frame. The interest point detection cannot consume too much of the per-frame computation budget, which for a 30Hz camera is 33 ms for all processing. This requirement, on current hardware, makes a direct application of the SIFT algorithm (**Lowe** (2004)), for example, undesirable.

The system of **Davison** (2003) applies the detector described by **Shi & Tomasi** (1994) to rectangular subregions of the image into which no existing landmarks project. If a sufficiently strong interest point is detected in the candidate region, it is chosen as a new landmark. The work of **Lemaire et al** (2005) uses Harris corners (**Harris & Stephen** (1988)) detected from the whole image, and attempts to choose corners in the parts of the image that have most recently come into view.

The vision SLAM system described here applies the FAST corner detector (**Rosten & Drummond** (2005), **Rosten & Drummond** (2006)) to the whole image downsampled to 320×240, then selects corners that are maximally distant from the locations of existing landmarks. The FAST corner detector runs on this quarter-size image in ~ 1 ms on current desktop hardware, so it uses a minimal fraction of the computing budget.

4.5.2.2 Map Augmentation

Upon selecting the image location of a new landmark, a corresponding inverse depth estimate is added to each particle. The interest point location is assumed to have Gaussian noise with 1 pixel isotropic variance. The image location and uncertainty are unprojected through cam^{-1} into the camera plane, and this 2D Gaussian forms the initial estimate of parameters u_0 and v_0 for the landmark. The q_0 parameter mean is set to 1. Using an EKF for inverse depth landmark estimation, q_0 is assigned very large variance to reflect the lack of prior knowledge. If the EIF is used instead, the lack of knowledge can be represented exactly by zero information for q_0. The current pose

estimate of each particle is recorded with the landmark estimates to establish the fixed inverse depth coordinate frame.

4.5.2.3 Re-observation

Observation of partially initialised landmarks follows the general active search framework. However, the process can be made more efficient by aggressively discarding new landmarks that are difficult to track.

The FAST detector is highly reliable over small viewpoint changes, detecting mostly the same corners. This property is exploited by further constraining the active search for partially initialised landmarks. For these landmarks, ZNCC is performed only in small windows surrounding the FAST corners inside the search area. If the new landmark's appearance changes over the first few views to the extent that the FAST corner is not re-detected, the landmark is unlikely to track well by the standard search method, and is dropped from the map early.

A common failure mode of SLAM systems is the inability to deal with a slowly-moving or static camera. In these scenarios, noise dominates the narrow baseline between frames, and the depth estimates of new landmarks spuriously converge. This tendency can be avoided by requiring the first several observations of partially initialised landmarks to decrease the uncertainty of q by a non-negligible amount. The threshold is imposed at landmark update time. If q is highly uncertain (e.g. on a landmark's second observation), and the latest observation does not adequately reduce the variance of q, the estimate is left unmodified. This mitigates early depth convergence when the camera is not translating.

Observations of partially initialised landmarks pass through the common filter update process described by 4.4.2. Thus, in contrast to the out-of-filter depth estimation of **Davison** (2003) and **Sola et al** (2005), such observations automatically constrain the camera pose as well as the landmark estimate. In the case of fully initialised landmarks with Gaussian estimates in Euclidean coordinates, the constraint on the pose estimate

is obvious: a 2D image observation constrains two abstract dimensions of a pose estimate. Though the update mathematics is identical for observations of landmarks with highly uncertain or unknown depth, it is worthwhile to consider intuitively how a 2D observation effectively provides a 1D constraint on the pose.

Consider the observed location (u, v) of a new landmark in the image in terms of the landmark's epipolar line, which is determined by the pose displacement from the first pose in which the landmark was observed (C_0) to the current pose. The vector, in the camera plane, from the epipole (projection of C_0) to (u, v) has components along the epipolar line and also perpendicular to the epipolar line. The first component, along the epipolar line, yields information about the landmark's depth (or inverse depth). The second component, reflecting perpendicular distance from the epipolar line, should be zero for perfect estimates of camera pose.

Thus, observations of new landmarks are a measure of epipolar reprojection error. Fusing these observations into the filter effectively applies the epipolar constraint over multiple frames and a variety of frame pairs. As with standard epipolar geometry estimation, this gives the filter information about both rotation and translation, up to scale. Thus, even when viewing mostly partially initialised points (such as at the beginning of SLAM) the camera pose is well-constrained (Figure 4.3).

4.5.2.4 Full Initialisation

After each update of a partially initialised landmark's estimate in each particle, the variance of q is examined. If the variance is small relative to the mean, then the depth is well-constrained, and the estimate is roughly Gaussian in Euclidean coordinates. The 3D estimate is transformed from inverse depth to Euclidean coordinates using the unscented transform (**Julier & Uhlmann** (1997a)), and the landmark is marked as fully initialised within that particle. The unscented transform is used to avoid the systematic bias that would result from directly transforming the inverse depth mean to the Euclidean mean: the depth would be consistently underestimated due to the nonlinearity of the inversion.

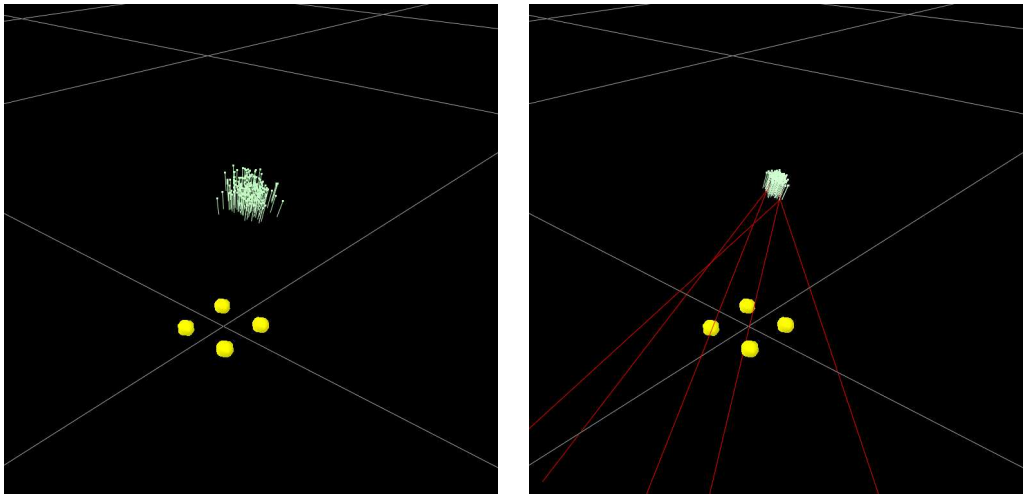

Figure 4.3: The pose estimate, shown as a particle cloud, is poorly constrained by the four fully initialised landmarks, shown as yellow blobs (left). Incorporating five partially initialised landmarks, shown as red rays, helps narrow the pose estimate (right)

Note that the event of full initialisation can occur at different times for the same landmark with respect to different particles. For those particles whose trajectory does not well-constrain the inverse depth estimate, the landmark will remain partially initialised for longer. If an estimate has spuriously converged in a particle, then subsequent observations should have lower likelihood under that particle, and it will yield fewer or no children in the resampling process.

4.6 Results

4.6.1 Running the System

The implementation can be initialised with or without a fiducial grid of known geometry. When the grid is used, it establishes the coordinate system of mapping, and four landmarks known with certainty. When mapping starts without the grid, the pose at the first time step is assumed to be the identity.

Input video can be recorded or live; the calibration of the camera is known to the system and assumed fixed throughout a run. The camera providing the live and recorded video used for evaluation captures 640×480 8-bit greyscale frames at 30Hz. The 6-parameter lens model includes two parameters for barrel distortion. The field of view of the lens is roughly 60°.

4.6.2 Visualisation

The estimated camera pose and map are rendered in a 3D view at the end of each time step. During resampling, the first particle sampled from the mixture component with the largest weight is marked as the "mode" particle. The landmark of and pose estimates of the mode particle are used to render the view. Fully initialised landmarks are depicted by yellow three-standard-deviation uncertainty ellipsoids, while the ellipsoids of partially initialised landmarks are pink. Partially initialised landmarks with highly uncertain depths are depicted as red rays emanating from the camera position of their first observation. The camera pose estimate is shown as a small frustum. Alternatively, the pose estimates from all particles are rendered as small vectors corresponding to the optical axis of the pose.

The video image is rendered in a separate view, with image search ellipses of fully initialised landmarks shown in yellow and those of partially initialised landmarks shown in pink. The search lines for partially initialised landmarks with highly uncertain depth are red. Successful observations are shown by a green box where the patch correlates best with the search region.

Figure 4.4 shows the 3D view and corresponding video image when the system is observing only the four fiducial landmarks. Figure 4.5 shows the pair for a mostly planar desktop scene.

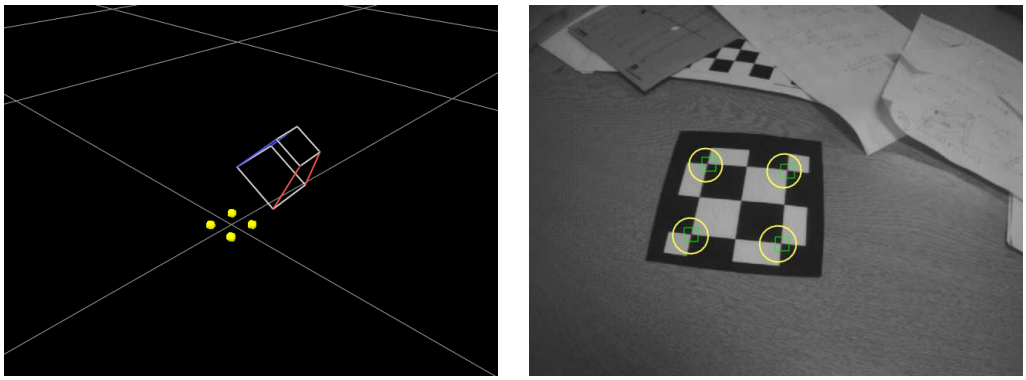

Figure 4.4: Rendered view and video image of a fiducial grid. The grid is used for initialisation

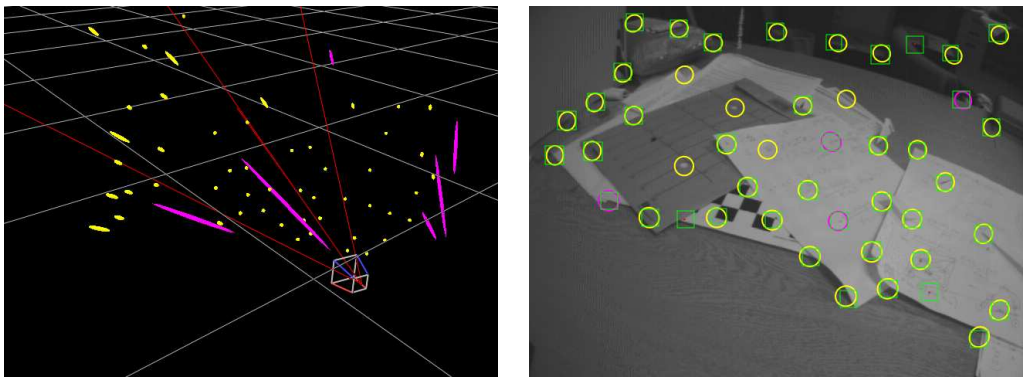

Figure 4.5: Rendered view and annotated video image of a mostly planar scene

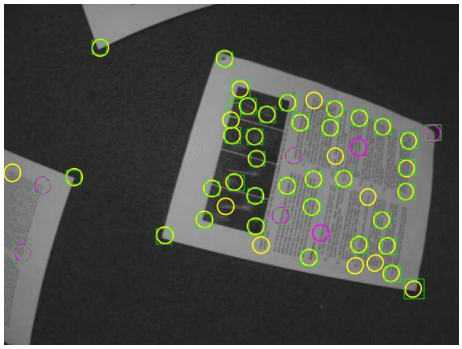

Figure 4.6: Annotated video image of real planar scene

4.6.3 Accuracy

The accuracy of the mapping and localisation of the SLAM system are evaluated using both real and synthetic sequences.

4.6.3.1 Mapping Accuracy

Figure 4.6 shows a video image from a real planar scene created by placing paper sheets on a 1m by 2m section of floor. The camera is panned, by hand, back and forth over the scene for 30 seconds, facing the floor from an altitude of ∼30cm. $M = 100$ particles are used. The generated map contains 243 landmarks. The map, viewed from an angle that highlights its planarity, is shown in Figure 4.7. A maximum-likelihood plane is fit to the landmark estimates; all fully initialised landmark estimates have means within 0.3 cm of the plane, so the error is less than 1% of the viewing distance.

The same experiment is performed using a synthetically rendered sequence. The scene consists of a single textured plane, and the virtual camera twice pans across the plane and back to the starting position. The rendering is performed a linear projection model and a 50° field of view. The resulting map contains 312 landmarks, and The same plane fitting is performed. All landmarks lie on the plane within 0.2% of the viewing distance. The superior mapping accuracy relative to the real sequence could be due to

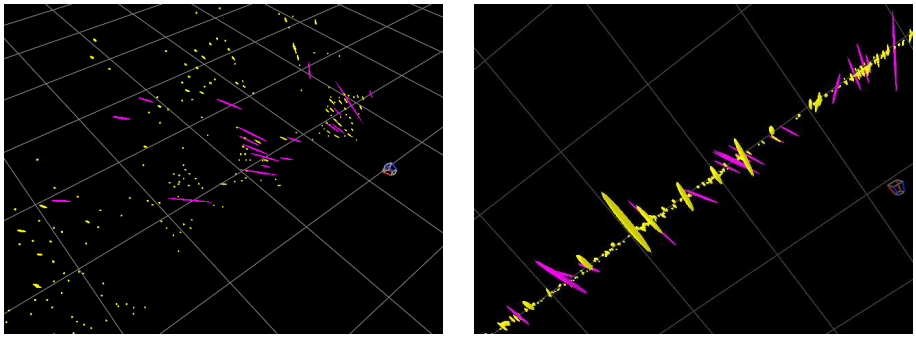

Figure 4.7: Perspective and side-on view of the map of a real planar scene

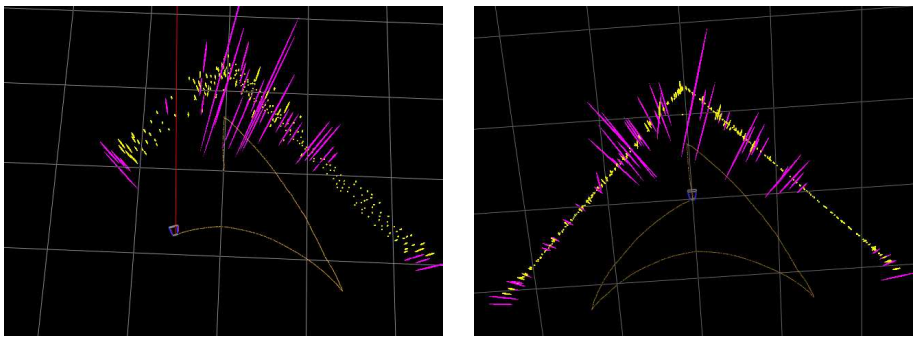

Figure 4.8: Map of a synthetically rendered corner structure, at the middle and end of the sequence

a higher density of texture in the rendered scene or an imperfect camera calibration of the real camera.

To confirm that the system correctly maps 3D structure, a synthetic sequence of two planes meeting in a right angle is rendered and run through the system. The virtual camera starts facing the corner, moves forward then pans right and left, eventually returning to the starting point. The resulting map, shown with camera trajectory from overhead in Figure 4.8, shows that the reconstruction is accurate. Two planes separately recovered by least-squares fitting to the appropriate landmark estimates have a meeting angle of within 1° of a right angle.

Plane 1 740 frames		Plane 2 782 frames		Corner 755 frames	
M	RMS Err (m)	M	RMS Err (m)	M	RMS Err (m)
10	0.015	10	0.021	10	0.046
20	0.015	20	0.015	20	0.061
40	0.017	40	0.014	40	0.031
80	0.016	80	0.014	80	0.036
160	0.014	160	0.014	160	0.032
320	0.015	320	0.014	320	0.030
640	0.014	640	0.013	640	0.022

Table 4.1: Localisation error relative to ground truth trajectory for synthetic sequences, using M particles

4.6.3.2 Localisation Accuracy

The pose estimates produced by the system are compared to ground-truth for synthetic sequences. The localisation error is expressed in terms of absolute positional discrepancies between the estimated and real camera poses. Because the sequences are processed without fiducials, the output trajectory is first transformed to the same coordinate system as the reference trajectory using least squares. The results for synthetic sequences are shown in Figure 4.1. In the first plane sequence, the virtual camera stays on a straight line without rotating. In the second plane sequence, the camera makes small motions and rotations off of the straight trajectory. Roughly 40 landmarks are observed each time step.

The trajectory estimate is also compared to the output of global iterative bundle adjustment for a real sequence. All landmark observations made while the system is running are recorded, and provide the input to bundle adjustment, using the map and trajectory estimated by SLAM as the starting point for optimisation. Thus the posterior of global optimisation is the best the filter can hope do with the observations it is given.

The test sequence, of 1835 frames, is a view of several desktops and walls in an indoor area, covering 12 square meters. With the default settings ($\sim$30 landmarks observable

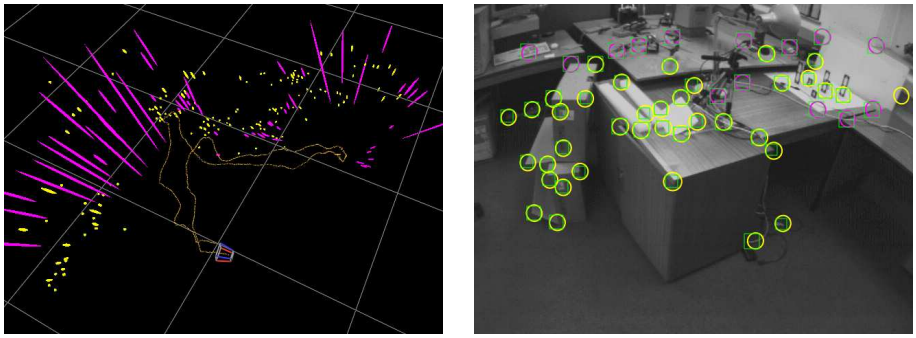

Figure 4.9: Map and camera view at the end of an indoor sequence

Desktop and Wall Sequence, 1835 frames

M	RMS Err (m)
10	0.065
20	0.058
40	0.043
80	0.038
160	0.040
320	0.038
640	0.036

Table 4.2: Localisation error relative to bundle adjustment trajectory for an indoor sequence, using M particles

each frame), the resulting map has 270 landmarks, well beyond the map complexity feasible with EKF SLAM. The map view and corresponding camera image at the end of the sequence is shown in Figure 4.9. The localisation error, larger than for synthetic sequences, generally decreases with increasing number of particles, as shown in Figure 4.2.

4.6.4 Efficiency

For all of the sequences shown above, when $M \leq 160$, the system spends no more than 33 ms processing each frame, on a 3.0 GHz Pentium Core 2 Duo. With small number of particles ($M = 20$), the processing time is dominated by image processing, while

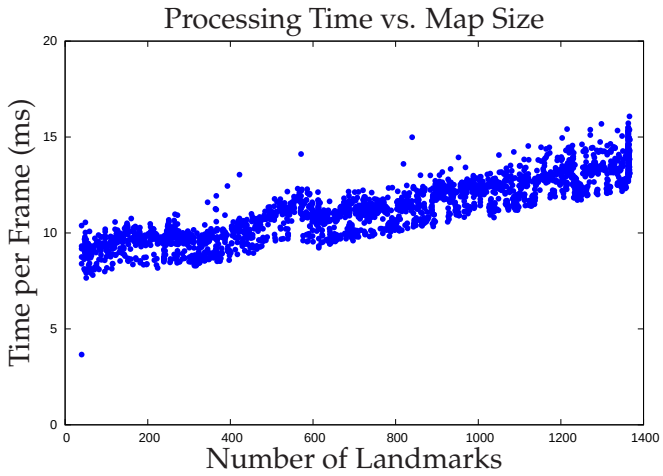

Figure 4.10: Processing time with respect to number of landmarks, using 80 particles. The linear growth is due to the copying in the particle resampling step

for large M, the per-particle updates and resampling require an increased chunk of computation.

In order to test the performance of the system when mapping many landmarks, a synthetic sequence is generated that contains a large, textured plane. The virtual camera moves across the plane in a scanning pattern. There are $N = 1363$ landmarks in the final map.

Figure 4.10 shows the timing results (not including visualisation) for a run with $M = 80$ particles. The processing time never exceeds 16 ms. The implementation does not include the copy-on-write landmark tree that allows resampling in $O(M \log N)$ time, so the cost of resampling grows linearly with N. However, the constant factor is small: the slope of the least-squares line fit to the data is 0.00365, or a penalty of 3.65 ms per frame for an additional 1000 landmarks. A simple scan through the landmark set is used to determine what landmarks should be observed, so some of the linear penalty could also be mitigated by storing landmarks in a spatial data structure.

Loop Closing Success

r	$M = 20$	40	80	160	320	640	1280
1.50	✓	✓	✓	✓	✓	✓	✓
1.75		✓	✓	✓	✓	✓	✓
2.00			✓	✓	✓	✓	✓
2.25			✓	✓	✓	✓	✓
2.50					✓	✓	✓
2.75						✓	✓
3.00							✓
3.25							✓
3.50							

Table 4.3: Loop closing results vs. radius of trajectory (r) and number of particles (M)

4.6.5 Loop Closing

When the trajectory of the camera is a loop, where the starting scene is not again visible until the camera returns to the same area, reobserving the initial landmarks by active search requires accuracy and precision in the state estimate. Inevitably, the pose (and map) estimate around a trajectory of this sort will accrue error. Then, upon returning, the landmark searches succeed only if the uncertainty in the estimate is well-captured in addition to the mean.

To evaluate the loop closing ability of the system, a planar scene is rendered, with the camera facing the plane and moving in a circular trajectory parallel to the plane, at distance 1 unit. The radius of the circle determines the size of the "loop". For a given set of parameters, the estimate may or may not be accurate enough to close the loop at the end of the trajectory. This binary condition is used to establish the number of particles necessary to allow at least simple loop closing at different scales.

The radius of the circle is varied from 1.5 to 3.5 units. The field of view of the virtual camera is roughly one square unit. The results, shown in Figure 4.3, show that a larger number of particles are required to close bigger loops. More specifically, the number of required landmarks appears to grow exponentially with the size of the loop.

4.6.6 Discussion

The performance results show that adopting FastSLAM as the estimation filter for monocular SLAM is efficient and feasible. For small environments, the system is capable of accurately mapping hundreds or thousands of landmarks at frame-rate, permitting a far denser representation of the scene than can be achieved with EKF SLAM. The partial initialisation parameterisation, using inverse depth coordinates, accommodates the problem of unknown depth in the monocular setting without resorting to out-of-filter methods. Active search is key to the efficient operation of the algorithm, as only a small fraction of the image is considered when looking for a given landmark.

However, the approach has considerable limitations. As with any purely active search framework, localisation depends on the tracking assumption – that landmarks will be found near where they are expected, according to their estimates and the dynamic model. When the dynamic model is violated, for instance by dropping or jerking the camera, tracking fails.

The particle cloud provides a sampled representation of the state distribution. This must be a faithful representation for estimation to succeed. As the state dimension grows, due to complex maps or long trajectories, the number of particles must grow with it. The loop closing results suggest that this growth might be exponential. This corroborates the suggestion of **Bailey et al** (2006a). The causes of this problem in general are discussed further in Chapter 6.

Nonetheless, for small domains and smooth trajectories, the system allows mapping of an order-of-magnitude more landmarks in real time than previously feasible. The output is accurate enough, at least locally, to use it as the starting point for further batch optimisation, or perhaps as the input for more sophisticated online estimation.

5

Edge Landmarks

5.1 Introduction

Point landmarks have desirable properties in the context of visual SLAM: point feature selection and description are well studied, the resulting feature descriptors are well-localisable in images, and they are highly distinctive, easing the task of data association. Many environments, however, have abundant edges and edge-like features. By tracking edges in the world, a SLAM system can build richer maps containing higher-level geometric information, and need not rely on an abundance of good point features.

However, in contrast to point features, edges can be well-localised in only one image dimension, and often have non-local extent in the other image dimension. Though highly invariant to lighting, edges are also difficult to distinguish from each other

locally. Such characteristics make the use of edge landmarks in visual SLAM challenging.

5.1.1 Contributions

This chapter describes how edge landmarks can be incorporated into the SLAM system of Chatper 4. A solution to this overall task requires several contributions:

- Definition and representation of edge landmarks as short, straight pieces of 1D structure
- An active search method for localising edge landmarks in an image
- An efficient algorithm for acquiring suitable new edge landmarks from an image
- A parameterisation of new edge landmarks permitting partial initialisation
- A robust data association method for accommodating ambiguous edge observations

The rest of this section reviews related work. Section 5.2 defines *edgelet* landmarks and their representation. Section 5.3 describes how edgelets can be observed within an active-search framework. Section 5.4 presents an efficient method for selecting new edgelets from an image. Section 5.5 describes how new edgelets can be initialised reliably. Section 5.6 explains a simple robust data association algorithm for dealing with inevitably ambiguous observations of edge landmarks. Section 5.7 presents and discusses performance results on real video sequences.

5.1.2 Related Work

Edges have been recognised as critical features in image processing since the beginning of computer vision. While edge detection methods abound, the algorithm of

Canny (1986) for choosing *edgels* in an image has emerged as the standard technique, and consistently ranks well in comparisons (**Heath et al** (1996); **Shin et al** (1999)). It provides a starting point for the edge feature selection algorithm presented in this chapter.

The invariance of edges to lighting, orientation, and scale makes them good candidates for tracking. For instance, model-based trackers use edge models to permit highly efficient tracking of moving objects or cameras. The RAPID tracker (**Harris** (1992)) is an early example of this approach, with the work of **Drummond & Cipolla** (2002) refining and improving the method. At each time step, a wireframe edge model of the structure is rendered into image space, and the camera pose is optimised to make the model align with the edgels in the image. This predict-update loop resembles the active-search model of SLAM; the difference is that in the tracking case, the structure is known. Model-based tracking with edges is a solution to a particular case of the more general structure-from-motion problem.

Reitmayr & Drummond (2006) use a richer, textured model of the environment to improve edge tracking. Instead of using only the wireframe edges, the tracker renders the textured model into the current scene, detects natural edges from the rendered image, and then tracks them in the live image. Thus edges that are weak or occluded from the current viewpoint are not detected and not tracked. This gives good results in the outdoor environments tested, but depends on the availability of a rich model.

The work of **Taylor et al** (1991) (see 2.2.1) estimates planar motion and the location of vertical edges in the plane. This basic algorithm is extended by **Taylor & Kriegman** (1995), which operates solely on edge segments detected in frames of a video sequence, and represents lines of arbitrary orientation in 3D. A global cost function is optimised to yield camera trajectory and line parameters. Each line segment is parameterised as part of an infinite line by its closest point to the origin and its direction, in a common global coordinate frame. Segments are extracted from video with the Canny edge detector. The cost function measures reprojection error as the total area between projected lines and edge segments in the image space. To initialise the optimisation, camera orientations are randomly selected from a constrained set of possibilities. Then

the local minimum of the cost function over all parameters is found by gradient descent. This is performed with multiple starting points, and the best local minimum is returned as the result. The algorithm yields reasonable reconstructions for sequences of 10-30 well-separated images. Additional results of the algorithm are shown in **Shin et al** (1999).

The previously discussed SLAM systems of **Neira et al** (1997) and **Kwok & Dissanayake** (2003) both treat vertical edges as landmarks, with the camera constrained to the 2D plane. The vision SLAM system of **Folkesson et al** (2005) is designed to support heterogeneous landmark types in a common framework. While lines are used as features, they are assumed to be confined to planes of known orientation.

Since the initial publication of the work of this chapter (**Eade & Drummond** (2006a)), others have investigated the problem of SLAM with edges of arbitrary orientation. The system presented by **Gee & Mayol-Cuevas** (2006) takes a model-based tracking approach, first estimating edge segments with an unscented Kalman filter framework, then adding them to an edge model used for tracking. New edge landmarks do not contribute to the localisation process until they have been set as part of the model. Once part of the model, the edge parameters do not change. The system described in **Lemaire & Lacroix** (2007) uses a Gaussian sum representation, related to that of **Lemaire et al** (2005), to initialise new edge landmarks, and a constrained EKF to maintain the state estimate. Line segments are represented with Plücker coordinates and extent bounds, and the observation process uses set-to-set matching of segments to perform data association.

Most similar to the work of this chapter is the system by **Smith et al** (2006), which estimates line segment landmarks in the standard EKF framework. The selection and observation process for landmarks focuses on long edge segments in the image, detected between interest points. The endpoints provide the parameterisation for the segments. The partial initialisation process ignores correlations between new and existing landmarks until the new landmarks are fully initialised. In the EKF SLAM framework, the map size is limited by computational bounds, so line segment landmarks must be conservatively acquired, and more than a few dozen cannot be mapped at frame rate.

5.2 Edgelet Landmarks

Point landmarks fit well in a SLAM system because they have a well-defined representation, both in image space and world space. In the image, a point landmark is represented as a locally planar patch with a distinct, but view-dependent, appearance. In the world, it is estimated as a three-dimensional point with Gaussian uncertainty. In order to use edge features, their image and world representations must also be defined.

5.2.1 Definition

The edge features that the SLAM system will estimate, dubbed *edgelets*, are defined with locality properties analogous to those of points. An edgelet is a local portion of an edge, with an edge being a strong, one-dimensional intensity change. Thus, given an edge, which may have significant extent in an image, any small segment on the edge can be taken as an edgelet observation. Furthermore, the edge need only be locally straight: a slow curve has many locally line-line pieces, all of which can be considered edgelets. Tracking only local edge segments avoids several problems inherent to estimating full edges in the world. Full edges, because they are not local quantities in an image, may be partially occluded, or broken into pieces in the image. They might never be wholly visible, so determining their full extent and actual endpoints could be impossible. The locality of edgelets means that assumptions made about an edgelet as a whole (for instance, that it is straight) is much more likely to be satisfied than the same assumption made about a long edge.

5.2.2 Representation

An edgelet in the environment can be represented as a 3D point $\mathbf{x}$ corresponding to the centre of the edgelet, and a unit vector $\hat{\mathbf{d}} \in \mathrm{S}^2$ denoting the direction of the edgelet. If $\hat{\mathbf{d}}$ is a 3D vector with unit length, this representation is not minimal, as $\hat{\mathbf{d}}$ has only

two degrees of freedom. However, the 3D representation is often more convenient in calculations. The uncertainty in the parameters is represented as a Gaussian with covariance $\mathbf{P}$. Given a camera pose $C = (\mathbf{R}, \mathbf{T}) \in \mathrm{SE}(3)$, the observation model $\mathbf{h}_1$ sending $\mathbf{x}$ to a point in the camera plane is identical to the observation model for 3D points:

$$\begin{aligned} \mathbf{X} &\equiv \mathbf{R}\mathbf{x} + \mathbf{T} \\ \mathbf{h}_1(\mathbf{x}) &= \mathit{project}\,(\mathbf{X}) \end{aligned} \tag{5.1}$$

The direction, $\hat{\mathbf{d}}$, is projected to a unit vector in the image plane:

$$\mathbf{D} \equiv \mathbf{R}\hat{\mathbf{d}} \tag{5.2}$$

$$\mathbf{h}_2(\hat{\mathbf{d}}) = \begin{pmatrix} \mathbf{X}_3\mathbf{D}_1 - \mathbf{X}_1\mathbf{D}_3 \\ \mathbf{X}_3\mathbf{D}_2 - \mathbf{X}_2\mathbf{D}_3 \end{pmatrix} \Big/ \left| \begin{pmatrix} \mathbf{X}_3\mathbf{D}_1 - \mathbf{X}_1\mathbf{D}_3 \\ \mathbf{X}_3\mathbf{D}_2 - \mathbf{X}_2\mathbf{D}_3 \end{pmatrix} \right| \tag{5.3}$$

The Jacobians of the edgelet observation model are shown in Section A.2.1.

Due to the aperture problem along the edge, observations can not decrease the uncertainty of $\mathbf{x}$ along the direction $\hat{\mathbf{d}}$. However, the location of the edgelet along the edge to pixel accuracy is determined from the first observation (Section 5.5), which is sufficiently precise to allow subsequent observation by active search.

5.3 Observing Edgelets

5.3.1 Prediction

Active search is employed to observe landmarks in video frames. Given a Gaussian estimate of an edgelet $(\mathbf{x}, \hat{\mathbf{d}})$, the landmark is observed by predicting its location in an image, with gated uncertainty region, and searching for the edgelet in the region. The edgelet's parameters project into the image plane according to 5.1 and 5.2, and its covariance projects through a linearisation of the observation functions $\mathbf{h}_1$ and $\mathbf{h}_2$. Then the image plane quantities project into pixel coordinates according to the calibrated

camera model cam. The result is a prediction of the edgelet in the image: an image location $\mathbf{x}_p$ and a 2D image direction $\mathbf{d}_p$, with associated covariance.

The prediction implies that the system expects to find a short edge segment centred at $\mathbf{x}_p$ with normal $\mathbf{n}_p$ perpendicular to $\mathbf{d}_p$. In the case of active search for point landmarks, the appearance of the landmark (a warped patch) is assumed roughly constant over the search region. The appearance model for edges is much simpler (a directed intensity discontinuity), but the same assumption is applied. The normal direction is approximated as constant over the search region, which is given by a three-standard-deviation variation of $\mathbf{x}_p$ in the direction of $\mathbf{n}_p$, with a predetermined, fixed "local" width (e.g. 15 pixels). The actual extent of edgelets along the edge is undefined and unestimated, as they are local features, so this fixed search region is adequate for reasonable changes in viewpoint.

5.3.2 Search

The set of edgels in the prediction region with gradient direction similar to $\mathbf{n}_p$ is recovered from the image. Straight segments within this local set of edgels are taken as possible observations of the edgelet. A variant of the Hough transform finds such segments (Figure 5.1).

First, the edgels are binned according to their quantised gradient angle θ, relative to the direction $\mathbf{n}_p$. Angle bins of width $\pi/100$ radians are sufficient, as any edgels with orientation more $\pi/10$ radians away from $\mathbf{n}_p$ are discarded. Peaks in total edgel count are computed by convolving the bins with a low-pass filter. Such peaks reflect many edgels with a common gradient direction.

For each peak group of edgels, a histogram of edgel location component along the direction of the bins' average gradient angle is constructed. These displacement histograms have bins of width ~ 1 pixel. The two kinds of histogram – angle and displacement – correspond to the two dimensions of the angle-radius representation of

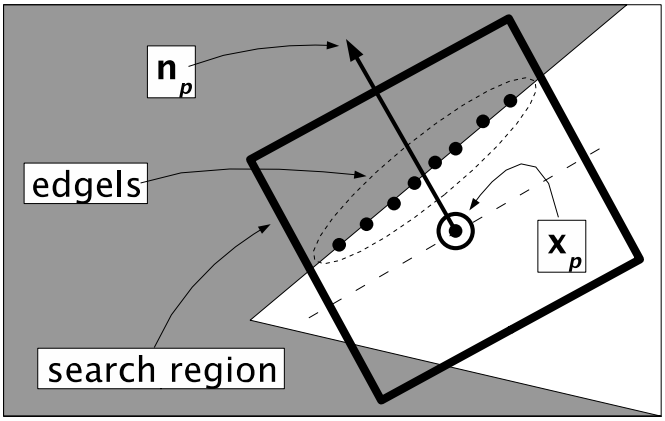

Figure 5.1: Edgelet observation: The coordinate frame of the search region, given by $\mathbf{x}_p$ and $\mathbf{n}_p$, is computed from the current edgelet estimate. Edgels are detected in the region and grouped into straight segments, to which slope-intercept lines are fitted

the Hough line transform. Thresholding the peaks in the radius histograms yields subsets of edgels that form straight segments in the region.

5.3.3 Observation Representation

For each straight-segment edgel set found in the search region, the edgel locations $\{\mathbf{e}^i\}$ are mapped into the camera plane through cam^{-1}, giving $\{\mathbf{e}_c^i\}$. The image coordinate frame described by $\mathbf{x}_p$ and $\mathbf{n}_p$ is also mapped into the camera plane, giving $\mathbf{x}_c$ and $\mathbf{n}_c$ respectively.

Each edgels offset in the direction of $\mathbf{n}_c$ from the line given by $\mathbf{x}_c$ is computed as $(\mathbf{e}_c^i - \mathbf{x}_c)^T \mathbf{n}_c$. A least-squares intercept-slope line, $y = b + mx$, is fitted to the resulting offsets. The parameters (b, m) of these lines are the 2D observations used to update the pose and landmark estimates of the filter. The position uncertainties in $\{\mathbf{e}_i\}$ are mapped through these transformations, yielding covariance in the line parameters (b, m) for each fit. The innovation of an observation hypothesis (for use in updates) is then the difference between the observed intercept-slope and the predicted intercept-slope, computed from $\mathbf{x}_c$ and $\mathbf{n}_c$.

In the simplest case, only one edgelet observation hypothesis is found by active search. When multiple hypotheses are found, the observation closest to the predicted edgelet location could be taken as the single hypothesis. However, clutter around the edgelet in the image can lead to incorrect data association. To avoid spurious edgelet observations, all of the observation hypotheses in each frame are considered together, deciding maximum likelihood data association as described in Section 5.6.

5.4 Finding new Edgelets

The map of the environment is initially empty, except for fiducial landmarks used to bootstrap the system. New edge landmarks must be acquired to populate the map as the camera moves. As with interest point detectors used for selecting point landmarks, an edgelet selection algorithm should yield edgelets that can be easily localised in subsequent frames. It must also be efficient, so as not to burden the real-time operation of the system. This section describes a simple, effective, and efficient method for choosing edgelets to track. The method yields the locations of short, straight edge segments that are well-separated from nearby edges of similar direction.

First, all the edgels in the image with a gradient magnitude that is maximal along the direction of the gradient are identified. The output of the first steps of the Canny algorithm**Canny** (1986) is sufficient for this stage. The edgels are considered in subsets determined by placing a grid of a fixed size over the image. A grid with boxes of size 16×16 pixels works well for a 320×240 image. All subsequent processing happens within each grid subset.

For a subset of edgels $\{\mathbf{e}_i\}$, the average second moment, $\mathbf{M}$, of the gradients $\mathbf{g}_i$ at the edgels is computed:

$$\mathbf{M} \;=\; \left\langle \mathbf{g}_i \mathbf{g}_i^T \right\rangle \tag{5.4}$$

The eigenvectors of $\mathbf{M}$ describe the dominant directions of intensity change in the image patch. For a patch containing a single edgelet, the eigenvector corresponding to

the larger eigenvalue should be normal to the edgelet. Let this dominant eigenvector be $\hat{\mathbf{n}}$. For each edgel, the angle θ between $\hat{\mathbf{n}}$ and the edgel's gradient satisfies

$$\cos\theta \quad = \quad \mathbf{g}_i^T \hat{\mathbf{n}} / \left|\mathbf{g}_i\right| \tag{5.5}$$

Those edgels with gradients in agreement with $\hat{\mathbf{n}}$ are selected by choosing a minimal $\cos\theta$ and thresholding:

$$\left(\mathbf{g}_i^T \mathbf{g}_i\right) \cos^2\theta \quad > \quad \left(\mathbf{g}_i^T \mathbf{n}\right)^2 \tag{5.6}$$

For all edgel locations $\mathbf{e}_i$ with gradient in agreement with $\hat{\mathbf{n}}$, consider the distribution of location in the direction of $\hat{\mathbf{n}}$, given by

$$b_i = \mathbf{e}_i^T \hat{\mathbf{n}} \tag{5.7}$$

The mean and variance of $\{b_i\}$ describe the location and agreement of edgels along the dominant direction. For a grid element with one clear, single straight edge, the variance will be on the order of a single pixel. Thresholding on this variance identifies grid elements containing candidate edgelets. Note that edgels with gradient directions not similar to the dominant gradient direction do not affect the edge-normal variance, as they are culled from the calculations early. Thus, a grid patch can contain two orthogonal segments and the stronger one will be chosen as a candidate edgelet. Each grid element contributes either one or zero candidate edgelets, with associated location, direction, and strength.

The algorithm accepts well-separated short straight segments, but rejects grid elements that might contain easily confused edge portions. Figure 5.2 shows a natural and a constructed example of this criterion in action. The rendered indicators are normal to each detected edgelet in the direction dark-to-light, and their length is proportional to the strength of the edgelet. The double lines are rejected by the edgelet selector because they are too close, and might be confused in a search.

On a Pentium IV 2.8 GHz workstation, using grid elements of size 16×16, the entire algorithm, including non-max-suppressed edgel detection, processes a typical 320×240 grayscale image in 2-3 ms, yielding up to 300 edgelets. Candidates in the image sufficiently distant from all recently observed landmarks are chosen as edgelets.

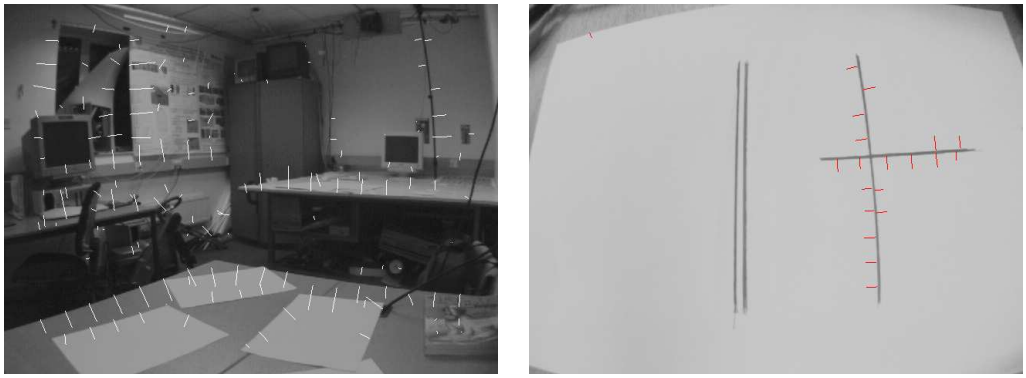

Figure 5.2: Output of the candidate edgelet detection algorithm

Edgelet selection is further guided by choosing edgelets with normal direction orthogonal to the image motion due to camera translation. This property can be computed using the current estimated camera translational velocity. The depth of such edgelets is likely to be recovered more rapidly, because the aperture problem is avoided.

5.5 Initialising Edgelets

A new edgelet cannot be added to the map as a fully-initialised landmark described in Section 5.2.2 until enough is known about its location and direction to make its estimate Gaussian. While its location and direction in the image plane is well-determined from one observation, the components along the viewing ray are unknown. Thus, the landmark must remain partially initialised until all of its dimensions are well-represented by a Gaussian. The situation is analogous to that of partially initialised points (4.5), and the linearity of an inverse depth observation model applies.

A partially initialised edgelet is parameterised in its initial observation frame with inverse depth. That is, instead of world coordinates $\mathbf{x} = (x, y, z)$, the edgelet position is given as $\mathbf{x}^* = (u, v, q)$, where (u, v) is the position of the edgelet in the camera plane and $q = z^{-1}$ (inverse depth), all with respect to the first camera pose from which the landmark was observed. Just as $\hat{\mathbf{d}}$ is the unit differential of $\mathbf{x}$ along the edge, the unit

differential $\hat{\mathbf{d}}^*$ of $\mathbf{x}^*$ along the edge parameterises the direction of partially initialised edgelets. The Jacobians of the inverse depth edgelet observation model are shown in Section A.2.2.

However, even the inverse-depth representation is not adequate when a new edgelet has been observed only once. After only one observation, there is no information about inverse depth or its derivative, so the representation is not Gaussian. The unscented transform (**Julier & Uhlmann** (1997a)) is used in this very initial phase to combine the first observations of an edgelet until its inverse-depth representation is Gaussian. This almost always happens in two frames.

The rest of the partial initialisation proceeds just as it does for point landmarks. As the edge landmark is repeatedly observed in subsequent images, the estimates of $\mathbf{x}^*$ and $\hat{\mathbf{d}}^*$ are updated, in each particle, using the EKF framework. The conditional independence of the estimates in each particle means that the EKF update is computed independently on each (low-dimension) estimate.

When the estimate converges so that it is well-represented as Gaussian in Euclidean world coordinates, a change-of-variables is performed using the unscented transform, and the landmark's mean and covariance is thereafter expressed in world coordinates $\mathbf{x}$, $\hat{\mathbf{d}}$, and $\mathbf{P}$. Given non-degenerate camera motion, new edgelets are usually fully initialised in fewer than ten frames. Each particle maintains a separate estimate of each landmark, so the change-of-variables occurs independently, sometimes at different times and with different results for different particles.

Just as with point landmarks, partially-initialised edgelets help to constrain the camera pose in the pose update stage of the filter update. Although observations of new edgelets cannot constrain the camera pose until the third observation (because of the degrees of freedom of the edgelet), once the inverse-depth representation is valid, the edgelet participates fully in the pose constraint process described above. Indeed, the system can operate well without ever changing variables to Euclidean world coordinates; the change-of-variables simply permits more efficient operation.

5.6 Robust Data Association

While point features have a highly distinct appearance model given by a textured patch, edges are characterised only by the direction of their intensity change from low to high. If an edgelet's prediction uncertainty is large, there may be several possible matching edges in the image search region, making data association between image observation hypotheses and landmarks ambiguous. The ambiguity can be resolved by choosing the set of associations that has the maximum likelihood given the current estimates of poses and landmarks.

However, for m observations, there are 2^m possible association combinations to consider. This number grows when multiple hypotheses exist for some observations. Random sampling consensus (RANSAC, **Fischler & Bolles** (1981)) can be used to sample from the subsets. The observations are sampled, with replacement, k at a time ($k = 4$ for this chapter), yielding subsets $\{S_i\}$.

The likelihood of a given edgelet observation is computed in measurement space. Each observation is a 2D Gaussian distribution $(\mathbf{z}_i, \mathbf{R}_i)$ of intercept-slope parameters relative to the predicted edgelet under an uncertain camera pose. The innovation covariance is computed by projecting the edgelet and pose uncertainty into this measurement space and adding $\mathbf{R}_i$. Evaluating the resulting zero-mean Gaussian density function at the observation's mean yields the observation's likelihood under the hypothesis.

For a given S_i, the posterior estimate P_i of a chosen particle's pose is computed. Each P_i is tested by evaluating the combined likelihood of the whole set of observations under it. For each observed landmark with more than one observation hypothesis in the frame, the most likely hypothesis under P_i is used for the test. If the likelihood of the best observation for a given landmark is below a threshold, the observation is considered an outlier, and the threshold likelihood is used in place of the observation's likelihood. Combining the likelihood over all observed landmarks for a given P_i yields a score, and a set of associations, for S_i.

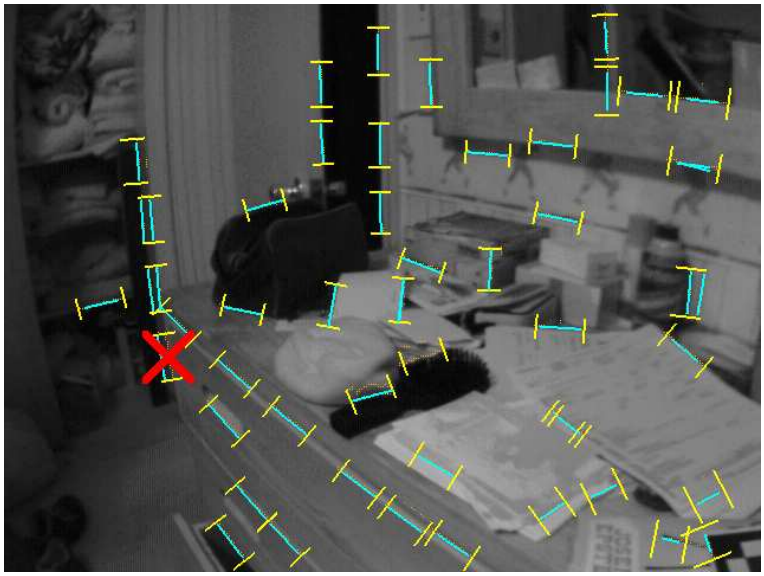

Figure 5.3: An example of outlier detection performed by the robust data association algorithm

The scoring process is repeated with many randomly sampled subsets S_i of k observations. After a fixed number of tries, the maximum-likelihood set of inliers is taken as the data association, and that set is used to update all pose and landmark estimates. Figure 5.3 shows an example with many multiple-hypothesis observations as well as a detected outlier. . The thick red '$\times$' indicates an observation determined to be an outlier: the active search algorithm has found the wrong edge in the image. Note that many observations have multiple hypotheses (shown as multiple blue segments); robust data association has chosen the most likely hypothesis in each case.

The number of random subsets tried is limited by computation time; 30 tries gives good experimental results without requiring excessive time. Even when only one observation hypothesis is found in the image for each observation, there are in fact two possibilities for each observation: inlier or outlier (the null hypothesis). This data association framework greatly improves the reliability of the system when viewing cluttered scenes, when partial initialisation gives spurious estimates, or when the static-world assumption is violated. Furthermore, the formulation is independent of landmark type, so it can be applied to points, edgelets, or both simultaneously.

5.7 Results

An implementation of monocular SLAM with edgelets is evaluated with the same hardware used for the results of Chapter 4. The system runs at frame rate (30Hz) while observing in excess of 40 edgelets each frame, and choosing among 30 data association RANSAC hypotheses. The average search time for locating an edgelet in a video frames is 0.1 ms. This cost of finding the landmark is exceeded by the cost of incorporating the observations into the filter estimates, which currently requires $\sim$0.2 ms per landmark for 80 particles. The tracking is noticeably more reliable when using the RANSAC-based maximum-likelihood data association described in Section 5.6.

Figure 5.4 shows the result of mapping a real planar scene. The system accurately captures the scene geometry: The mean displacement of the edgelet centres from the ground plane is $8.58 \cdot 10^{-5}$m. The standard deviation is 2.5mm. The standard deviation of edge angles out of the plane is 0.0331rad, or 1.9°.

Mapping simple geometric structure gives a qualitative check of the edgelet estimation process. Figure 5.6 shows a small, artificial desktop scene and a portion of the map, reflecting the orthogonal edges of the items. Figure 5.5 shows additional qualitative mapping results from a scene with more complex structure. The system correctly captures the structure of the cabinet top and face. The locality property of edgelets, along with the detection algorithm, also allows the system to track curved edges, as shown in Figure 5.7.

Figure 5.8 shows the results of a 195 second run in a dining room scene (Figure 5.9). The sequence was captured live from the camera, not pre-recorded and processed from disk. The fiducial grid is used for initialisation. The constructed map contains 196 edgelet landmarks. Between 30 and 50 observations are made each frame, all while running at frame rate (including graphic display and data output). The objects on the table are clearly represented, including the curved hot plates. The estimated trajectory of the camera during the run is shown in Figure 5.10. The trajectory is not post-processed; it is the concatenated output of the filter at every time step. Even

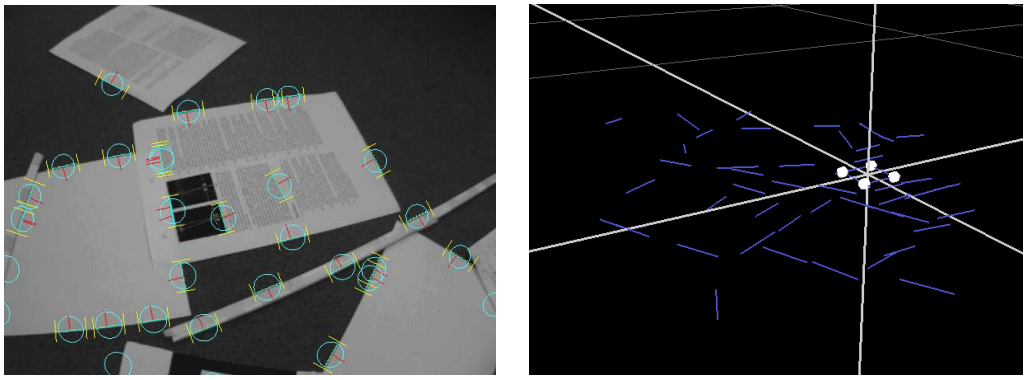

Figure 5.4: A planar scene with 51 edgelets

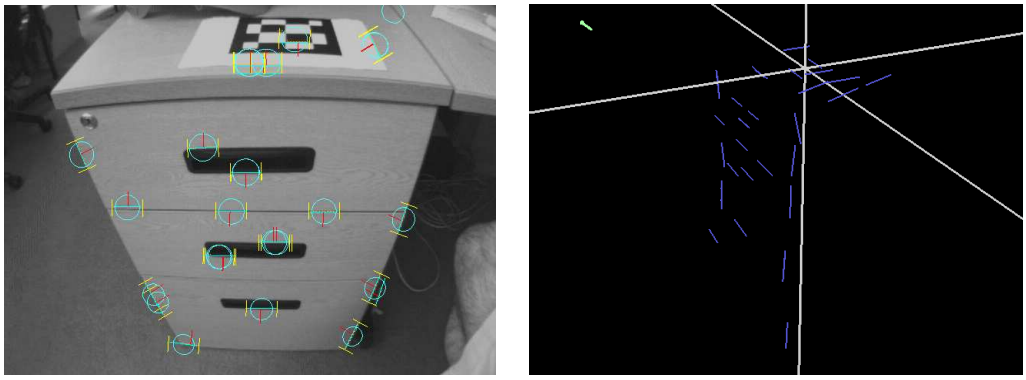

Figure 5.5: Edgelets of varying orientations

though the trajectory covers only one portion of the table, the positions and orientations of distant edgelets (such as those on the vertical walls) are correctly estimated.

5.7.1 Discussion

This chapter has defined edgelets and established how to select, observe, initialise, and estimate them in the context of particle-filter SLAM. The selection algorithm has minimal computational cost and delivers short, straight segments in the image, well-separated from edges of similar orientation. The partial initialisation approach, using an inverse-depth representation, allows landmark estimates to be maintained in EKFs even when the distribution is not yet Gaussian in world coordinates. Partially

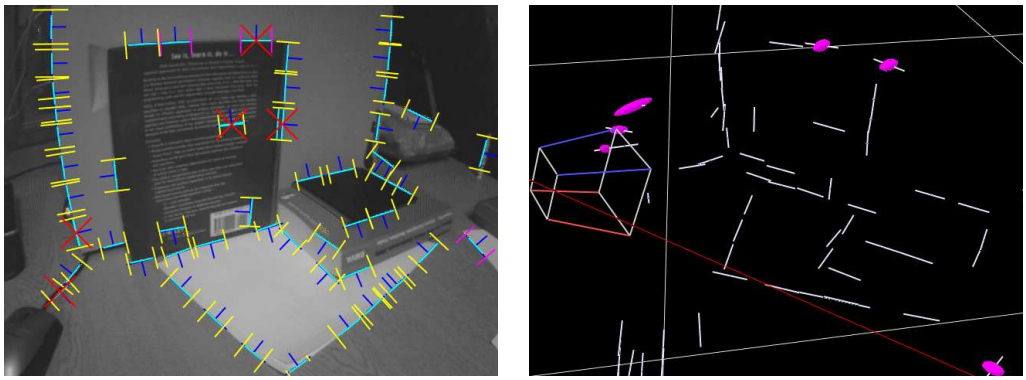

Figure 5.6: A scene with 3D structure, and the resulting map

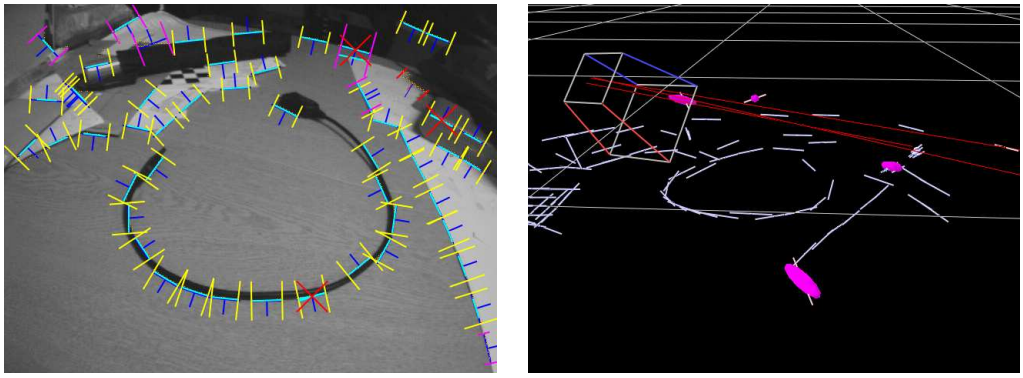

Figure 5.7: Edgelets on curved surfaces

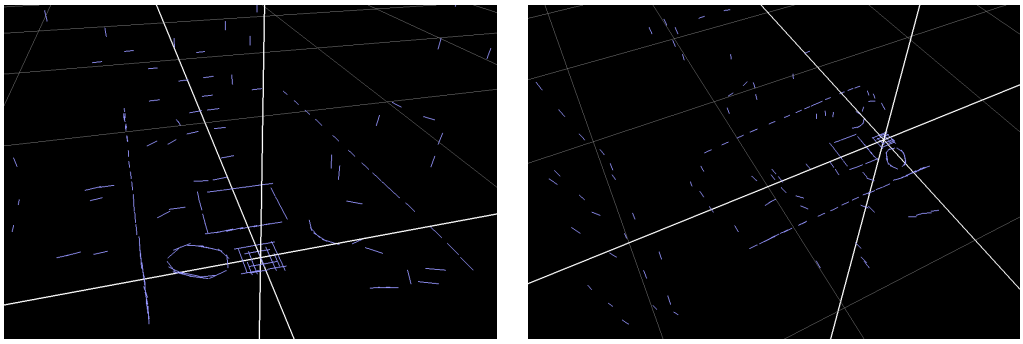

Figure 5.8: Map constructed from a 195s live run in a dining room environment, with 196 edgelets

Figure 5.9: A dining room scene

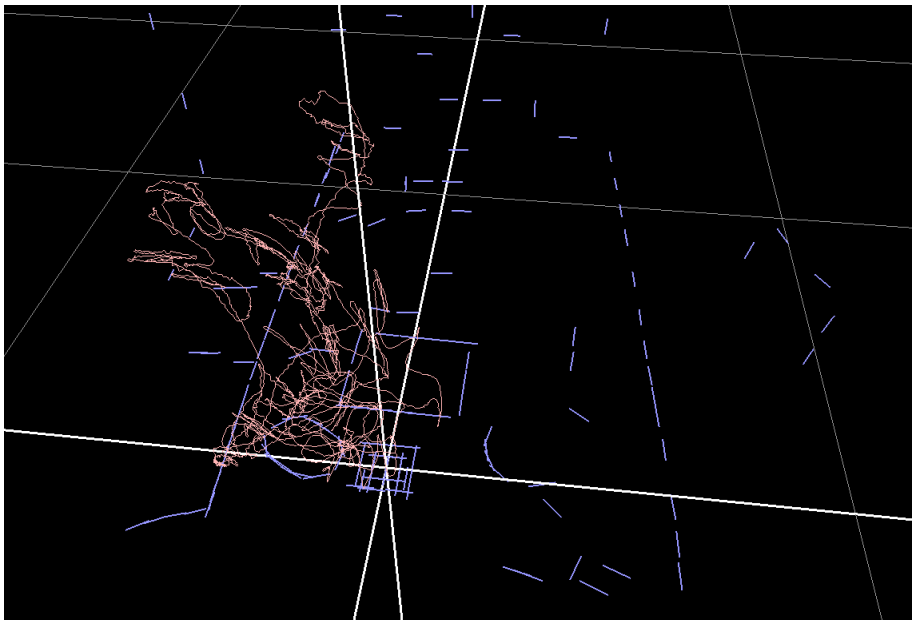

Figure 5.10: Estimated camera trajectory for the dining room sequence

initialised edgelets participate fully in the pose refinement process crucial to the operation of the particle filter.

The indistinct nature of edge image features makes data association more difficult than with point landmarks. This problem is addressed by a robust data association method that evaluates all observations in a frame simultaneously. Using MLESAC and existing landmark estimates, unlikely hypotheses are discarded.

The presented system shows that edges can be successfully tracked and mapped in an active-search monocular SLAM setting. The SLAM operation is accurate and efficient, capturing the edge geometry of the environment while running at frame rate with hundreds of landmarks. Furthermore, the use of only local portions of possibly extended edges yields a framework flexible enough to map curved intensity changes.

While edge landmarks are useful for capturing higher-level geometry in scenes and when point features are scarce, they need not be used in isolation. Future work should examine how points and edgelets can be used in tandem. The current implementation already permits this, but the difficulty lies in determining when points rather than edges should be selected, and vice versa. A SLAM system might also wish group together individual landmarks (points and edgelets) that are structurally linked into composite entities, to be represented as a unit.

6

SLAM as a Graph of Local Maps

6.1 Introduction

The monocular SLAM system presented in Chapter 4 and extended in Chapter 5 permits frame-rate mapping of hundreds or thousands of landmarks, and yields accurate localisation in small environments. Nonetheless, the map and trajectory estimates do not always converge to their true values, leading to lost tracking and incorrect maps. The culprit is the same inconsistency that plagues EKF SLAM, discussed in Section 2.3.2.

Fundamentally, the problem is not that the pose or structure estimate of the filter is inaccurate. After all, observations are noisy and the motion model is approximate, so temporary inaccuracy is impossible to avoid during SLAM. Rather, the problem is that the uncertainty in the estimate is underestimated. The filter is too confident in its inaccurate trajectory or map to allow the state estimate to converge to the truth.

This chapter further discusses the nature of the consistency problem in SLAM in general, and in monocular SLAM in particular. The causes of inconsistency motivate a SLAM algorithm that maintains a graph of local, statistically independent maps. The local maps are updated with a focus on preserving consistent estimation, factoring out significantly nonlinear constraints into the graph edges. Similarly to other submapping strategies, bounding the complexity of local maps also permits efficient operation with many landmarks. The system effectively factors the estimation problem into an efficient recursive component and an iterative optimisation component.

6.1.1 Contributions

The monocular SLAM method presented in this chapter is called Graph-SLAM in the sequel. [Note that, despite the name, this is not directly related to the work by **Thrun & Montemerlo** (2006), which is discussed below.] This chapter documents the following contributions:

- Efficient monocular SLAM with local maps represented in a graph
- Compact local mapping parameterisation with a nearly-linear observation model
- Nonlinearity measure to guide the use of image observations in updates
- Iterative graph optimisation, accounting for local and topological constraints

The remainder of this section reviews the consistency problem in SLAM, and discusses related work. Section 6.2 describes the graph-based state representation of Graph-SLAM. Section 6.3 details the local mapping procedure, Section 6.4 shows how the graph is maintained during operation, and Section 6.5 describes how the graph edges are iteratively optimised to satisfy structure and cycle constraints. Section 6.6 presents performance and consistency results for simulated and real data, comparing to EKF SLAM, FastSLAM, and bundle adjustment.

6.1.2 Consistency

As described in Chapter 4, the FastSLAM algorithm factors the SLAM problem by conditioning landmark estimates on samples drawn from the trajectory distribution. These samples are maintained by a modified particle filter. In order for the factorisation to consistently represent the underlying distribution of structure and motion, the particle set must encode the full uncertainty in the trajectory. However, as time passes, and pose uncertainty relative to a fixed coordinate frame grows, the bounded number of particles inevitably fails to fill out the tails of the distribution.

This failure is shown empirically by **Bailey et al** (2006b). For any fixed number of particles, the filter acts as a non-optimal local search in the state space. After a short period of time passes, the search cannot adequately explore the space, so the estimate is no longer Bayesian and the filter becomes inconsistent. Increasing the number of particles increases the accuracy of the result, and slightly delays the inconsistency, but without an exponential growth in the number of particles over time, the algorithm cannot be consistent.

As discussed above in section 2.3.2, the EKF SLAM algorithm becomes inconsistent due to linearisation errors. Such errors are mitigated by using a coordinate frame local to the vehicle, iterating the update step of the EKF to achieve more accurate linearisation, or using the Unscented Kalman Filter (**Julier & Uhlmann** (1997a)) to avoid explicit linearisation. But these techniques only delay the onset of inconsistency. **Bailey et al** (2006a) and **Castellanos et al** (2007) argue that the only way to limit inconsistency in SLAM is to maintain independent submaps, so that any local inconsistency in one submap does not escape to other submaps and cause global inconsistency.

6.1.3 Related Work

Motivated by the avoidance of inconsistency, Graph-SLAM is closely related to previous submapping methods. Several SLAM systems in the robotics literature represent

state using multiple local coordinate frames (**Bailey** (2002); **Bosse et al** (2004); **Castellanos et al** (2007); **Leonard & Newman** (2003); **Lisien et al** (2003)).

In particular, the graph of local maps described below is similar to the formulation of Network Coupled Feature Maps (NCFM, **Bailey** (2002)) and the Atlas framework (**Bosse et al** (2004)). In both of these systems, submaps are statistically independent of each other, and in the case of Atlas, the local mapping technique can vary from one submap to another. Submaps are connected in a graph structure, with map-to-map coordinate transformations associated with graph edges. Landmark and pose estimates can be propagated through the graph from one map to another.

NCFM uses the common landmarks estimated by two local maps help constrain the transformation estimate between the maps. Atlas, on the other hand, requires no common structure between submaps, so depends on motion estimates when moving between submaps to decrease the uncertainty in the edges. When cycles are present, different paths through the graph between two submaps might compose to different transformations. Neither NCFM nor Atlas apply graph-wide constraints online to enforce that edge cycles compose to the identity transformation, though Atlas results are often shown with a post-optimised graph.

The Graph-SLAM approach can also be viewed as a hierarchical bundle adjustment structure-from-motion method. Multiple observations sharing a nearly-linear observation model are coalesced into nodes containing high-dimensional, rich observations, and the relations between these high-dimensional observations are optimised at the global level. Thus optimisation of the linear parts of the parameter space proceeds recursively, permitting much more efficient global optimisation than standard bundle adjustment.

Thrun & Montemerlo (2006) present a bundle adjustment formulation confusingly called "GraphSLAM". Cast in terms of robot localisation and mapping, the paper actually describes the standard bundle adjustment algorithm, where structure parameters are eliminated and solved as a function of pose parameters. Iterative nonlinear optimisation yields a minimum on the graph energy, just as the nonlinear optimisation

of bundle adjustment minimises reprojection error. Sharing the limits of global batch optimisation in general, the method cannot run on large graphs efficiently enough to allow online operation.

Hierarchical bundle adjustment has been previously explored. The method of **Fitzgibbon & Zisserman** (1998), discussed in Section 2.2.1, first estimates trifocal tensors over view triplets, then performs a full bundle adjustment over subsets of increasing size. The system described by **Mouragnon et al** (2006b) and **Mouragnon et al** (2006a) performs bundle adjustment over a constant number of recent key frames, which are chosen sparsely from the set of all views using simple heuristics. While performing such temporally local bundle adjustment improves the accuracy of the result, for sufficiently large sequences it will not converge to the solution given by a full optimisation.

Recent work by **Klein & Murray** (2007) (published after the work of this chapter) selects key frames from video, performing full Levenberg-Marquardt bundle adjustment while tracking the camera pose relative to the current map. The state consists of the poses of all of the key frames and the parameters of all of the 3D points used as landmarks. Thousands of interest point correspondences are established between nearby key frames, providing the observations for the optimisation. Small local subsets of the parameters are optimised alone before the whole state is optimised jointly. The consistency of the algorithm is not tested, though it successfully closes medium-size loops, thanks to its highly accurate mapping. The brute-force nature of the optimisation limits its useful operation to about 150 key frames, with which small indoor areas can be adequately represented. Such areas are the target domain of the system, in which it yields excellent accuracy. However, the pose estimation does not take into account landmark uncertainty, so a proper measure of uncertainty for the pose estimate is not available at each time step.

The key frame mapping method of **Konolige & Agrawal** (2007) (published concurrently with the work of this chapter), like the work of **Klein & Murray** (2007) also employs iterative global optimisation over selected poses, but factors out the landmark state earlier in the process. Only relative pose measurements between key frames are maintained, in graph form. A sparse bundle adjustment optimises the inter-frame

transformations. This approach is similar to the one below, except that the interframe measurements are not updated with new observations of the joining features. However, multiple adjacent frame-to-frame pose measurements are collapsed together, which can be seen as a similar refinement. The system operates well in large outdoor environments, especially when using a stereo camera, but does not quite reach 30Hz processing speeds.

6.2 Graph-based State Representation

Graph-SLAM represents the map state in a graph. Each node of the graph contains a local map, built from a subset of all observations, with its own local coordinate frame. Each edge of the graph represents an uncertain transformations between the coordinate frames of its endpoints, constrained by estimates of common landmarks in the endpoint maps. Because each local map has its own scale, independent from the maps of other nodes, the transformations represented by edges must include scale change.

6.2.1 Local Maps

Each local map estimates landmark parameters using a subset of all observations made from video images. These subsets are disjoint across all submaps, so a local map's estimate is statistically independent from all other maps. Camera pose is not estimated as part of the local state, but is always represented with respect to a local coordinate frame.

A local map estimate is represented by a high-dimensional Gaussian, encoding the mean and covariance over all landmark parameters in the map. In order for this representation to be a consistent approximation to the actual state distribution, the observation model should be as linear as possible, as discussed in Section 4.5.1. Thus the inverse depth parameterisation is a natural choice for landmarks in a local map.

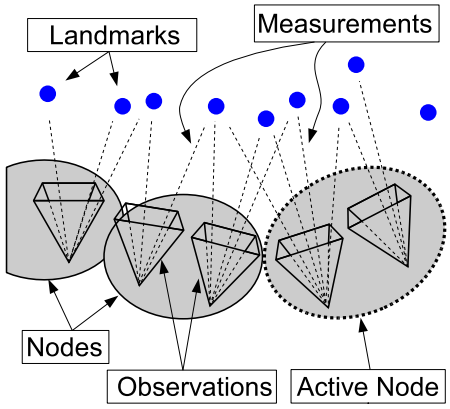

Figure 6.1: Observations are coalesced into nodes, each with a local coordinate frames.

Further, in order to accurately represent the lack of depth information when initialising landmarks, the inverse covariance (information) matrix is stored instead of the covariance. Zeros in the diagonal of this matrix encode total lack of information about the corresponding parameter (whereas zeros in the covariance diagonal imply perfect knowledge). The mean vector is stored directly, in inverse depth coordinates.

The landmark estimates are highly correlated within a local map, in contrast to their conditional independence within a FastSLAM particle. If the EIF or similar filter machinery is to be used to maintain the estimates, the computational cost grows quickly with the dimension of the state. In the application of inverse depth parameterisation to landmark initialisation in FastSLAM, the pose relative to which the inverse depth coordinates are expressed is fixed. In that case, the trajectory is sampled, so no uncertainty or correlation in the pose parameters needs to be represented in the state. In an EKF or EIF, however, the inverse depth coordinate frame parameters and their uncertainties and correlations need to be folded into the estimation. Using the representation described in **Montiel et al** (2006), this requires six parameters per landmark.

In fact, this uncertainty in the coordinate frames for the each landmark's inverse depth parameters need not be represented. To motivate this conclusion, consider the case of Euclidean landmark parameters. In a simple scenario, all coordinates are represented in the same global coordinate frame. However, a different arbitrary, fixed coordinate

system could be used for each landmark, without any loss of correctness. Such coordinate systems can be chosen randomly, and as long as they are fixed, estimation proceeds as normal, though the observation model is distinct for each landmark.

Likewise, distinct fixed coordinate systems can be used for landmarks in the inverse depth parameterisation. Instead of generating them arbitrarily, they are chosen so that the observation model is nearly linear. Using the camera pose estimate C_0 at the first observation of the landmark to define the landmark's coordinate frame (relative to the local map) confers this benefit. The pose estimate is copied and stored with the landmark's mean. It is not part of the state vector, so it does not change with subsequent filter updates. If C_0 is sufficiently accurate, the observation model will remain linear for small camera displacements. If not, then the observation model will become more nonlinear, but the estimation is otherwise correct. Fixing these per-landmark coordinate frames halves the state dimension in the local map from $6N$ to $3N$ for N landmarks.

From the viewpoint of bundle adjustment for structure-from-motion, each local map can be seen as a high-dimensional observation, constructed from 2D image observations coalesced by the local filter (see Figure 6.1). Because all image observations are statistically independent of each other, and each local map is built from image observations used by no other maps, the "big" observations are also statistically independent. The goal of the partitioning of observations into multiple local maps is to keep the observation models as linear as possible, so that intra-node mapping is as consistent as possible. This is described in more detail in Section 6.3.2.

6.2.2 Camera Pose

The camera pose estimate is maintained independently from the landmark estimates. A constant velocity model (see 4.3.2) is used to approximate camera motion, so the pose estimate is 12 dimensional: 6 dimensions for the pose and 6 for the velocity in the tangent space. All observations made from each video image are used to refine the camera pose by nonlinear optimisation of the pose parameters (Section 6.3.3). Unlike

in EKF SLAM, the map update process (Section 6.3.5) does not incorporate the pose information given by the dynamic model, to avoid spurious information gain when the model is incorrect. In fact, all of the information in the pose estimate is ignored in the update process, which uses the mean pose only as a starting point for nonlinear optimisation.

The dynamic model serves primarily to help tracking succeed by improving active search (Section 6.3.2). It could be replaced by a more complex model, or it could incorporate visual odometry (**Nistér et al** (2004), **Klein & Drummond** (2005)) or inertial measurements (**Klein & Drummond** (2002)) for better predictions and more robust observation. However, the simple model performs well enough for smooth motions.

As the graph is traversed during the course of SLAM, the pose estimate is transformed from one local coordinate frame to another through the edges. The state mean and uncertainty are modified according to the equations given in Section 6.2.3.

6.2.3 Map-to-Map Transformations

The edges of the graph explicitly represent the transformations between the coordinate frames of nodes, induced by estimates of landmarks common to the two endpoints of the edge. Because the scale within each node is a free parameter, the transformations must represent scale changes as well as rigid transformations. Thus the edge transformations are scaled Euclidean transformations – i.e., similarity transformations. Edges are bidirectional; the transformation and its inverse, along with respective uncertainties, are both computed and stored whenever the edge estimate changes.

A similarity transformation $S = [(\mathbf{R}, \mathbf{t}), s]$ is given by a rotation and translation $(\mathbf{R}, \mathbf{t}) \in \mathrm{SE}(3)$ and a scale $s \in \mathbb{R}$. A Cartesian point $\mathbf{x}$ is mapped via S as

$$S(\mathbf{x}) \equiv s\,(R\mathbf{x} + T) \tag{6.1}$$

This has a matrix representation similar to that of $\mathrm{SE}(3)$:

$$S = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & \frac{1}{s} \end{pmatrix} \tag{6.2}$$

In this representation, the formula for the inverse is obvious:

$$S^{-1} = \begin{pmatrix} \mathbf{R}^T & -s\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & s \end{pmatrix} \tag{6.3}$$

$$= \left[\left(\mathbf{R}^T, -s\mathbf{R}^T\mathbf{t} \right), s^{-1} \right] \tag{6.4}$$

As with rigid transformations, differentials and covariances of similarity transforms are represented in the tangent space around the origin, and mapped onto the manifold via the exponential map:

$$\exp \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\omega} \\ \sigma \end{pmatrix} = \left[\exp \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\omega} \end{pmatrix}, e^{\sigma} \right] \tag{6.5}$$

The differentials of this mapping are easily computed by extending the Jacobians of the exponential map in $\mathrm{SE}(3)$.

The edge transformations are used to propagate landmark and pose estimates through the graph. Propagation of landmark estimates is straightforward; they can either be pushed through the unscented transform or by mapping the mean through the transformation and transforming the covariance with the Jacobian.

Transforming a camera pose through an edge means something different. The process is easily defined if the edge transformation S has unit scale ($s = 1$, so $S \in \mathrm{SE}(3)$): A camera pose $C = (\mathbf{R}_C, \mathbf{t}_C) \in \mathrm{SE}(3)$ should map a point $\mathbf{y} \in \mathbb{R}^3$ to the same thing to which the transformed camera pose maps the transformed point:

$$T[C] \cdot (T \cdot \mathbf{y}) = C \cdot \mathbf{y}$$

$$T[C] = C \cdot T^{-1} \tag{6.6}$$

But $T[C] \in \mathrm{SE}(3)$ only if $T \in \mathrm{SE}(3)$. When $s \neq 1$, the transformed pose should map $T \cdot \mathbf{y}$ to a scaled version of $C \cdot \mathbf{y}$ by renormalising the composite transformation to unit

scale:

$$\begin{aligned} T[C] &\equiv [(\mathbf{I}, \mathbf{0}), s] \cdot C \cdot T^{-1} \\ &= \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \frac{1}{s} \end{pmatrix} \cdot \left(\begin{array}{c|c} \mathbf{R}_C & \mathbf{t}_C \\ \hline \mathbf{0} & 1 \end{array}\right) \cdot \begin{pmatrix} \mathbf{R}^T & -s\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & s \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{R}_C\mathbf{R}^T & s\left(\mathbf{t}_C - \mathbf{R}_C\mathbf{R}^T\mathbf{t}\right) \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \end{aligned} \tag{6.7}$$

From this formula, it is clear that vectors in the tangent space of C are transformed through T by simply multiplying their translation components by s.

6.3 Local SLAM

6.3.1 Overview

This section describes the SLAM algorithm within a local map. At the beginning of each time step, camera pose is represented relative to the local coordinate frame of the *active node*. Observations of landmarks are gleaned from active search, and the pose estimate is updated using the observations. Then a new active node is chosen, into which some or all of the observations will be fused. The choice of active node is driven by a measure of the nonlinearity of the observation model.

During the local mapping process, the graph structure is held constant. Graph maintenance occurs only at the end of the time step, after the observations have been used to constrain the camera pose and update a local map. The extra-local machinery is explained in Section 6.4.

6.3.2 Active Search

When an image is retrieved from the camera, the time step begins. The Gaussian pose estimate in the active node's coordinate frame is modified according the noisy

Figure 6.2: Active search regions for a frame from a desktop sequence. The search ellipses are yellow, and successful observations are shown with green boxes. The observations are of different scales in the image because of their distinct warps for the current pose

dynamic model, yielding a prior estimate of pose at the current time. This prior gives a starting point for finding landmark patches in the image.

Each landmark has an associated simple appearance model in each node where it is estimated, based on a small image patch acquired when the landmark is first observed in the node. As in Section 4.4.1, the appearance model is made somewhat invariant to viewpoint change by warping the patch before searching the image. Using the prior pose estimate, the displacement from the pose where the landmark was first observed dictates the rotation and scaling of the patch, while the distortion due to the nonlinear camera model is approximated by small affine warping. For efficiency reasons, the warp is computed from the mean landmark parameters and mean pose parameters, and held constant over the search region in the image.

The region is searched for the patch using ZNCC, and a quadratic form is fitted to the correlation score surface around the peak, giving an estimate of registration uncertainty. The result is a 2D observation of the landmark. An example video image with search ellipses and successful observations is shown in Figure 6.2.

The predictive search is not based solely on information in the active node; information about the landmark from nearby nodes in the graph is also incorporated. A distance-limited breadth-first search of the graph, starting from the active node, yields a tree connecting nearby nodes. A landmark's estimates are propagated through this tree into the active node (the root). This propagation progresses in a depth-first manner, with each node first collecting estimates from its children in the tree before transforming the combination through the edge to its parent. Thus the uncertainty of each coordinate transformation and landmark estimate is reflected properly in the result.

When the estimates reach the the active node, any local information about the landmark is added to the prediction. This yields a posterior estimate of the landmark in the local coordinate frame, conditioned on the information in all nodes of the tree. This posterior is projected through the observation model, accounting for the uncertainty in the current camera pose relative to the active node. The result is a more constrained search region in the image than would be given by using only the local landmark estimate.

Thus, even if a node does not have a direct estimate of a landmark, or even if its estimate is very uncertain, by combining the information from nearby nodes, a feasible prediction can be made for the landmark. Note that the internal state of all nodes in the tree remains unchanged by this process; it serves only to aid the active search. This mechanism also allows the active node to increase the number of landmarks it shares with nodes to which it is connected. Landmark estimates from connected nodes are transformed through the connecting edges and into the image, and successful observation allows them to be added to the local map's filter. As edge transformations are constrained by estimates of common landmarks, this process helps improve the local edge constraints.

6.3.3 Pose Update

Once a set of 2D observations have been extracted from the image, they are used to update the pose estimate. Because the pose estimate is maintained independently

from the local structure, no correlation is maintained between pose and landmarks. The pose update is formulated as a nonlinear optimisation of the pose parameters that minimises the reprojection error of the latest observations given the current estimates in a local map.

A subset O of the landmarks in a map is observed in a given timestep. An iterative Gauss-Newton optimisation adjusts the pose C to jointly accommodate the observations $\mathbf{z}_O$ given the map's estimate of landmark parameters. In each iteration, the dense covariance block matrix $\mathbf{P}_O$ corresponding to the observed landmarks $\mathbf{x}_O$ is projected through the observation model $\mathbf{h}$ into the camera plane, and block-diagonal measurement noise $\mathbf{R}_O$ is added. This gives the innovation covariance $\mathbf{S}$ in the camera plane:

$$\mathbf{J}_\mathbf{x} \equiv \frac{\partial \mathbf{h}(\mathbf{x}_O, C)}{\partial \mathbf{x}_O} \tag{6.8}$$

$$\mathbf{J}_C \equiv \frac{\partial \mathbf{h}(\mathbf{x}_O, \exp(\boldsymbol{\epsilon}) \cdot C)}{\partial \boldsymbol{\epsilon}} \tag{6.9}$$

$$\mathbf{S} = \mathbf{J}_\mathbf{x} \mathbf{P}_O \mathbf{J}_\mathbf{x}^T + \mathbf{R}_O \tag{6.10}$$

The innovation information (inverse covariance) is projected into pose parameter space, along with the innovation $\mathbf{v}_O$, weighted by the information. Then a pose update increment $\boldsymbol{\delta}$ in the tangent space is computed by solving the linear least squares equation:

$$\mathbf{v}_O \equiv \mathbf{z}_O - \mathbf{h}(\vec{x}_O, C) \tag{6.11}$$

$$\mathbf{J}_C^T \mathbf{S}^{-1} \mathbf{J}_C \cdot \boldsymbol{\delta} = \mathbf{J}_C^T \mathbf{S}^{-1} \mathbf{v}_O \tag{6.12}$$

The equation can be efficiently solved using the Cholesky decomposition (for computing both $\mathbf{S}^{-1}$ and $\boldsymbol{\delta}$). The residual error r under the pose C is given by the innovation and its covariance:

$$r = \mathbf{v}_O^T \mathbf{S}^{-1} \mathbf{v}_O \tag{6.13}$$

At each iteration the pose is updated using the exponential map and left multiplication:

$$C' = \exp \boldsymbol{\delta} \cdot C \tag{6.14}$$

Convergence is confirmed by checking the decrease rate of the residual r after each iteration. The covariance of the resulting estimate is given by the inverse information:

$$\left[\mathbf{J}_C^T \mathbf{S}^{-1} \mathbf{J}_C\right]^{-1} \tag{6.15}$$

Because some landmarks might have estimates (also or only) in nearby nodes besides the active node, this optimisation can be done independently in each local map with at least three observed landmarks. The resulting pose estimates are transformed back through the graph and combined with the motion model in the current active node. This combination (by multiplication of Gaussians) gives a posterior distribution over pose that accounts for all nearby estimates of observed landmarks.

6.3.4 Choosing the Active Node

Already used to update the pose, the observations should also be used to update a local map. In order to avoid inconsistency, the landmark's observation models should be as linear as possible for the map update. This is accomplished by quantifying the nonlinearity of the model in the landmark parameters and choosing a local map where the largest subset of the observations have sufficiently linear models.

A nonlinearity metric of the observation model for an uncertain landmark estimate $\mathbf{x} \in (\hat{\mathbf{x}}, \mathbf{P})$ should reflect the difference between how the linearised model and the full model project the landmark distribution into the measurement space. Let the camera pose be C. The linearised model $\mathbf{H}$ is given by the first-order Taylor expansion of the actual model $\mathbf{h}$ around the landmark parameter mean $\hat{\mathbf{x}}$:

$$\mathbf{H}_{\hat{\mathbf{x}}}(\mathbf{x}, C) = \mathbf{h}(\hat{\mathbf{x}}, C) + \mathbf{J}_{\mathbf{x}}(\mathbf{x} - \hat{\mathbf{x}}) \tag{6.16}$$

The nonlinearity nl of $\mathbf{h}$ is reflected by the expected squared difference between projection by $\mathbf{h}$ and by $\mathbf{H}$:

$$\mathrm{nl}(\mathbf{h}; \hat{\mathbf{x}}, C) = \mathrm{E}\left[\|\mathbf{H}_{\hat{\mathbf{x}}}(\mathbf{x}, C) - \mathbf{h}(\mathbf{x}, C)\|^2\right] \tag{6.17}$$

This expectation can be approximated using the unscented transform on $\mathbf{x} \in \mathcal{N}(\hat{\mathbf{x}}, \mathbf{P})$.

In order to evaluate the nonlinearity of the observation model in the presence of uncertain camera pose, another unscented transform should be layered on top, sampling

sigma points $\{C_i\}$ from the pose distribution $C \in \mathcal{N}(\hat{C}, \mathbf{\Sigma})$ to compute expectation:

$$\mathrm{nl}(\mathbf{h}; \hat{\mathbf{x}}, C) \approx \mathrm{E}[\mathrm{nl}(\mathbf{h}; \hat{\mathbf{x}}, C_i)] \tag{6.18}$$

Note that the nonlinearity is being measured only with respect to the landmark parameters, not the camera parameters. The second unscented transform serves only to compute the expected landmark-wise nonlinearity over the distribution of poses.

In this formulation, landmarks with very small uncertainty will have nearly-linear observation models, because the displacements of samples from the mean are all small in measurement space. This is intuitively correct, because a first-order approximation Taylor expansion of a function about a point will always perfectly match the function at that point.

The new active node is chosen from the current active node and all nodes connected to it. Each node's suitability is evaluated by considering the nonlinearity of the observation model for each of the observations of landmarks in each node. This requires first transforming the (updated) pose estimate to each nearby node in turn, before evaluating nl for each observation. Comparing the value of nl to a fixed threshold determines the admissibility of each observation. The node that admits the most observations is chosen to become active. Observations not admissible in this new active node are not used in the map update process, to avoid inconsistency.

If the new active node is different from the old one, the pose estimate is transformed into the node's coordinate frame before the local map is updated.

6.3.5 Local Map Update

Let O be a set of observations of landmarks L estimated in the local map, collectively parameterised by mean vector $\hat{\mathbf{x}}$ and information matrix $\mathbf{\Lambda}$. The local map update adjusts the pose and landmark parameters to satisfy both the prior and the observations. Because the the observations have already been used to update the camera pose, the information in the pose estimate must not be used in the local map update. Instead,

the mean of the pose estimate is used only as a starting guess for nonlinear optimisation.

Each observation $o_i \in O$ is given by a 2D position and covariance in the camera plane, $(\mathbf{z}_i, \mathbf{R}_i)$. The landmark index associated with observation o is $l(o)$. Each landmark $l_j \in L$ has a slice of the mean vector, $\mathbf{x}_j$, shortened to $\mathbf{x}_{o_i}$ for the landmark associated with observation o_i. The residual to be minimised is the sum of landmark error relative to the prior and observation projection error:

$$\mathbf{v}_i \equiv \mathbf{z}_i - \mathbf{h}(\mathbf{x}_{o_i}, C) \tag{6.19}$$

$$r(\mathbf{x}, C) = (\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{\Lambda} (\mathbf{x} - \hat{\mathbf{x}}) + \sum_{o_i \in O} \mathbf{v}_i^T \mathbf{R}_i^{-1} \mathbf{v}_i \tag{6.20}$$

This residual can be iteratively minimised using the Levenberg-Marquardt algorithm, relinearising the observation model around the landmark means and current pose at each iteration. Let the error vector $\mathbf{e}$ be the landmark differences from the prior stacked on top of the observation innovations:

$$\mathbf{e} = \begin{pmatrix} \mathbf{x} - \hat{\mathbf{x}} \\ \hline \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_m \end{pmatrix} \tag{6.21}$$

The residual can be rewritten as a quadratic error function:

$$r(\mathbf{x}, C) = \mathbf{e}^T \left(\begin{array}{c|ccc} \mathbf{\Lambda} & \multicolumn{3}{c}{\mathbf{0}} \\ \hline & \mathbf{R}_1^{-1} & & \\ \mathbf{0} & & \ddots & \\ & & & \mathbf{R}_m^{-1} \end{array}\right) \mathbf{e} \tag{6.22}$$

Minimising this residual, for instance by conjugate gradients, results in optimised parameters $\mathbf{x}$ and C. A quadratic form is then fitted to the residual surface around the minimum, by linearising the observation model at the optimised parameters. The camera pose is marginalised out of this representation, yielding an updated landmark information matrix $\mathbf{\Lambda}'$ to go with the updated mean $\mathbf{x}$.

Because the observation model is nearly linear, one iteration of Levenberg-Marquardt or conjugate gradients optimisation is usually sufficient to reach the local minimum.

Each iteration has cost quadratic or cubic in the number of landmarks. Marginalising out the camera from the information matrix, using the Cholesky decomposition, requires cubic time in the number of landmarks in the local map. This cubic scaling factor limits the maximum number of landmarks per node to 80, if efficient operation is to be maintained.

Despite the cubic cost, this iterated information filter is chosen over the EKF (which has $O(N^2m)$ update cost for m landmarks) because it permits a more faithful representation of the information in the local map. New landmarks can be added to the map before the update step with zero information in the inverse depth coordinates, and their co-information with the other landmarks observed in the same frame is correctly computed and linearised. In the EKF, the lack of information about depth has to be approximated by a huge variance, which also makes the update calculations numerically ill-conditioned.

After the update, the same Cholesky decomposition is also used to compute the covariance matrix by inverting the information matrix. The gauge freedom in local scale is fixed by adding a (temporary) prior in the scaling direction before inverting, and then subtracting any remaining scale uncertainty using the Sherman-Morrison formula (**Press et al** (1992)).

6.3.6 Landmark Management

If active search for a landmark fails more often than it succeeds, the landmark is removed from the local map by marginalising out its estimate from the information matrix and removing it from the mean vector. It is not removed from other maps where it might be estimated. However, its identifier is added to a local list of "failed" landmarks, so that the system does not continue attempt to observe it from within the current node (by transforming it through an edge). This acts as a crude local occlusion model, as the landmark is dropped from those nodes from which it cannot be seen, and remains in those from which it can. After a fixed time period of $\sim$5 seconds,

an identifier is removed from the list of failed landmarks, as the occlusion may be temporary.

New landmarks are acquired by the system so that the number of landmarks observable in the current pose remains at a target level (default 30). Landmarks are selected using an interest point detector such as FAST (**Rosten & Drummond** (2005)) or Harris (**Harris & Stephen** (1988)). Points maximally distant from existing landmarks in the image are taken until the target quota is filled. A patch around each landmark is stored as the landmark's appearance model, and the landmark's parameters (with initial measurement noise) are added to the mean vector and information matrix. The state is augmented *before* the update, so that the correlations between the new landmarks and other landmarks observed in the same frame are correctly computed.

If few points are observable, and insufficiently many can be added to the current local map due to its size bound, a new node will be created, as described in Section 6.4.2.

6.4 Graph Maintenance

After the pose estimate and active node are updated from image observations, graph edges are updated and created. When exploring new territory, new nodes are created as well.

6.4.1 Edge Updates

A graph edge encodes an estimated transformation between the local coordinate frames of the nodes that are its endpoints. As shown in Section 6.2.3, these uncertain transformations are represented as Gaussian distributions in the space of similarity transforms. Each edge maintains two transformation estimates (and their inverses). Estimate S_L, with covariance $\mathbf{\Sigma}_L$, encodes the local constraints imposed by common land-

marks in the endpoint nodes, and S_G represents the result of the global optimisation algorithm described in Section 6.5.

After the local map of the active node is updated, all locally-constrained estimates of edges incident to the active node are refined. The local estimate is computed from the estimates of landmarks L_c common to the endpoints of an edge (Figure 6.3). One direction of the edge is modified by the refinement process, and the other direction is then recovered by inversion (and appropriate transformation of uncertainty). The node at the tail of the chosen direction is called the source, and the other endpoint is called the target.

Iterated Gauss-Newton optimisation modifies the parameters of S_L to maximise the likelihood that estimates of landmarks L_c in the source node, projected through S_L into the target node, match with the target's corresponding estimates. The more precise the local estimates of L_c in the source and target, the more precise the resulting transformation estimate (the larger the eigenvalues of $\mathbf{Sigma}_L^{-1}$). Because the landmarks are highly correlated in each of the source and target node maps, the transformation must be applied to all landmarks L_c simultaneously, mapping their full covariance through the linearised transformation. The state block in the source and target corresponding to L_c are effectively treated as high-dimensional observations of structure. Straightforward application of Levenberg-Marquardt optimisation then brings these observations into alignment by refining the edge transformation.

The optimisation is well-defined only when at least three landmarks are common between the nodes, but more shared landmarks provide more constraints and yield a better transformation estimate. Because each iteration requires the decomposition of a matrix with dimension $3|L_c|$, the edge update becomes too expensive if there are many common landmarks. This is mitigated by limiting the cardinality L_c. Selecting only those shared landmarks that have well-constrained estimates in both the source and target works well when $|L_c|$ is large.

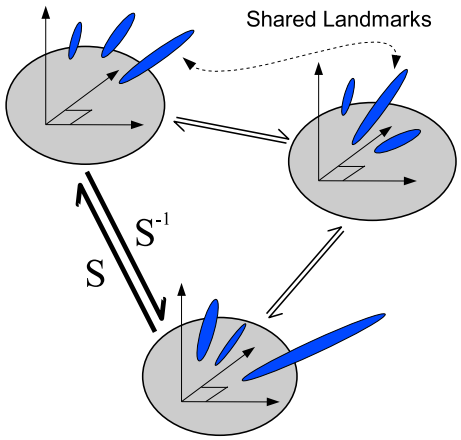

Figure 6.3: Landmarks commonly estimated in two nodes determine a similarity transformation between the nodes.

6.4.2 Node Creation

When not enough landmarks are observable, and there is not enough room in the active node's map to reach the minimum quota (default is 10), a new node is created. Any observations made in the current video image are used to bootstrap the new node's filter, with additional landmarks acquired in order to reach the desired density. An edge between the previous active node and the new node is created, with its transformation estimate initialised to the last estimated camera pose in the previous active node (with no scale change).

As landmarks are observed in the new node in subsequent time steps, the new local map converges, and the edge transformation computed solely from shared landmarks becomes well-conditioned. When this occurs, the estimate from the edge update algorithm replaces the approximation from the camera pose when the edge was created.

New nodes could also be created to ensure more thorough use of observations. Requiring a minimum number of admissible observations (in terms of Section 6.3.4) can cause more frequent creation of nodes, resulting in a denser mapping of the environment. The cost is increased storage and computation, as the graph is more complex.

6.4.3 Edge Creation

Edges explicitly represent the transformations between nodes induced by shared landmarks. Thus, their creation need not be limited to coincide with node creation. Whenever two unconnected nodes share a sufficient number of landmarks, they can be connected with an edge in a well-defined manner.

At each time step, the nodes up to $d > 1$ hops from the active node in the graph are checked for common landmarks with the active node. If a sufficiently large number of landmarks is shared between a distant node and the active node (8 by default), an edge is created between them. A starting estimate for the edge transformation is given by composing the hops in the existing graph. The edge is immediately optimised according the common landmark estimates.

By default, $d = 2$, so any edges added by this procedure create cycles of length 3 in the graph. Setting d higher creates longer cycles, but requires many more nodes to be examined. Cycles provide additional constraints on the edges of the graph, and improve the results of the graph optimisation algorithm of Section 6.5. Also, adding edges decreases the graph distance between the active node and other nodes in the graph, allowing more landmarks from other nodes to be observed by active search in the active node.

6.5 Graph Optimisation

When the graph is a tree, the most likely configuration of the edges is given by their locally-optimised transformations. The only constraints on each edge are those imposed by the common landmarks in its two endpoints' maps. When cycles are created in the graph, further constraints can be expressed. The graph optimisation algorithm computes, by nonlinear optimisation, the most likely edge transformations that satisfy these constraints.

6.5.1 Constraints

There are two type of constraints imposed on edges: local probabilistic constraints induced by shared landmark estimates, and global topological constraints imposed by graph cycles. The first type of constraint is discussed in Section 6.4.1; probabilistic estimates of the same landmarks in the endpoint nodes determine a probabilistic estimate $(S_L, \mathbf{\Sigma}_L)$ of the edge transformation that satisfies them. These locally-constrained transformation estimates can be seen as springs between nodes.

The second type of constraint is topological: composing the transformations around a cycle must give the identity transformation upon returning to the starting point. Each cycle imposes such a constraint, and the constraint affects every edge in the cycle.

The constraints – probabilistic and topological – can be expressed together by writing a probabilistic cost function over graphs that satisfy the topological constraint. First, the difference vector between any two similarity transforms is expressed in the tangent space, and the local error of an edge's global transformation S_G with respect to its local estimate $(S_L, \mathbf{\Sigma}_L)$ can be given in terms of this difference:

$$\mathrm{diff}(A, B) \equiv \ln\left(A \cdot B^{-1}\right) \tag{6.23}$$

$$\mathrm{local\_error}(t) = \mathrm{diff}(S_G[t], S_L[t])^T \mathbf{\Sigma}_L^{-1} \mathrm{diff}(S_G[t], S_L[t]) \tag{6.24}$$

Cycles are most easily expressed in terms of a spanning tree on the graph, a set of edges that is a tree and connects all nodes. Let E be the set of all edges. For any spanning tree $T \subseteq E$, each edge $e \in E \setminus T$ induces a cycle $c(e)$ in the graph. The other edges of the cycle, the ordered set $\mathrm{path}(e) = \{c_1(e), \ldots, c_n(e)\}$, point forward along the path in the tree from the source node to the target node of e (Figure 6.4). The cost associated with each cycle is the error between the path in the tree and the local spring of the edge out of the tree:

$$\mathrm{tree\_path}(e) = c_n(e) \cdot c_{n-1}(e) \ldots c_1(e) \tag{6.25}$$

$$\mathrm{cycle\_error}(e) = \mathrm{diff}(\mathrm{tree\_path}(e), S_L[e])^T \mathbf{\Sigma}_L^{-1} \mathrm{diff}(\mathrm{tree\_path}(e), S_L[e]) \tag{6.26}$$

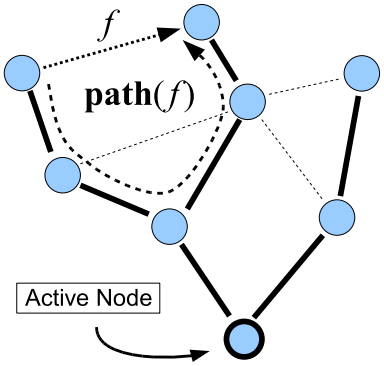

Figure 6.4: Each cycle in the graph imposes a constraint on the edge transformations. The transformations of edges around a cycle should compose to the identity

Putting these two kinds of error together, the total cost function g is then a sum of the error induced by all edges in the tree with respect to their local springs, and all edges out of the tree with respect to their cycles:

$$g(E,T) = \sum_{t \in T} \text{local\_error}(t) + \sum_{e \in E \setminus T} \text{cycle\_error}(e) \tag{6.27}$$

The cost is proportional to the negative log-likelihood of the graph with the non-tree transformations set so that cycles compose to the identity transformation.

6.5.2 Constraint Satisfaction

Satisfying the constraints requires first identifying cycles in the graph. A breadth first search starting at the active node builds a spanning tree T with active node as root, and each time a non-tree edge e is encountered, it is added to the (initially empty) list of cycles.

With a complete list of cycles, the cost function g can be differentiated with respect to the transformation parameters of edges in T, permitting iterative nonlinear optimisation. Setting the out-of-tree edge transformations to the composition of their in-tree counterpart paths after the in-tree edges are modified guarantees that the cycle constraints are always satisfied.

Using Levenberg-Marquardt or conjugate gradients, the computational expense of each iteration of cost function optimisation grows super-linearly with the number of graph edges, so preconditioned gradient descent (PCGD) is used instead. PGCD is equivalent to Levenberg-Marquardt where the data matrix outer product is approximated by its block diagonal. This block diagonal matrix can be inverted in time linear in the number of edges. The process is iterated, adjusting the diagonal regularising term of Levenberg-Marquardt according to whether the cost function decreases after each update. When the cost function stops decreasing and the regularising term is large, the optimisation has converged.

This convergence might take hundreds or thousands of iterations with PGCD. However, each iteration has a low cost, and the optimisation can be easily amortised over multiple time steps. A fixed number of iterations (default 30) is performed each frame after all other updates, so convergence takes place over several video frames. When there are no cycles in the graph, the optimisation process simply moves S_G quadratically towards S_L for every edge.

6.5.3 Caveats

While the local maps in the graph are statistically independent, the edge transformations are correlated. This is because the transformation estimates of distinct edges depend on landmark estimates in common endpoint nodes. Thus, the cost function above is not strictly probabilistically correct. However, as discussed in **Bailey** (2002) (and Section 2.3.3), the correlations between edges may be of the sort such that ignoring them gives conservative rather than overconfident estimates.

Nonetheless, the edge configuration that minimises the cost function g is not necessarily the coherent configuration with maximum likelihood, due to these correlations. In practice, though, the optimisation seems to yield good results, and considerably improves the accuracy of the global map in the presence of cycles.

6.6 Results

6.6.1 Consistency Evaluation

To evaluate the consistency of the proposed SLAM filter, the Monte-Carlo approach of **Bailey et al** (2006a) is adopted. Given a known camera trajectory and known landmark positions, observations can be sampled for each time step of the trajectory. The observations are sampled by projecting the true landmark positions through the observation model, and adding samples from the measurement noise distribution. Then the SLAM system is run with these observations as input. At each time step k, the mean and covariance $(\hat{C}_k, \mathbf{\Sigma}_k)$ of the estimated pose are saved.

Each pose estimate $(\hat{C}_k, \mathbf{\Sigma}_k)$ can then be compared to the true pose C_k, yielding the normalised estimation error squared (NEES) for each time step k:

$$\epsilon_k = \ln(C_k * \hat{C}_k^{-1})^T \mathbf{\Sigma}_k^{-1} \ln(C_k * \hat{C}_k^{-1}) \tag{6.28}$$

The NEES is related to the negative log-likelihood of the true pose given the estimated pose. The filter consistency is measured by computing the average NEES over M Monte-Carlo runs of the filter, each with a different set of samples from the observation model. If the filter is consistent and linear-Gaussian, then ϵ_k is χ^2 distributed with 6 degrees of freedom, and the average $\bar{\epsilon}_k$ over the M runs tends to the dimension 6. This hypothesis can be subjected to a χ^2 acceptance test.

For $M = 25$ runs, if the filter is consistent and linear-Gaussian, $M\hat{\epsilon}_k$ has a χ^2 density with $6M$ degrees of freedom. The 95% probability region for $\hat{\epsilon}_k$ is then given by $[4.719, 7.432]$. If $\hat{\epsilon}_k$ rises much above the interval, the filter is optimistic – it overestimates its information. If $\hat{\epsilon}_k$ sits below the interval, the filter is conservative.

The consistency of graph-based SLAM, FastSLAM, and EKF SLAM are compared using this approach. The FastSLAM implementation is the same as in Chapter 4, except that landmarks are never converted to Euclidean parameterisation from inverse

depth. The EKF SLAM implementation also stores all landmarks in the inverse depth parameterisation, relative to the pose at their first observation. Using the Euclidean parameterisation in the EKF was found empirically to give poor results. The same constant velocity model is used in all three filters.

In order to stabilise the three filters, the measurement noise level reported to the filter is 2σ, while the noise with which observations are actually sampled is σ. Further, the measurement noise is drawn from a uniform distribution on a disc of radius 2σ, instead of from a Gaussian with standard deviation σ. The variance of both distributions is σ^2. Using this noise distribution prevents the occasional addition of very large measurement error to the observations, which is impossible in video-based runs due to gated search. When the actual measurement error is too large, the map and pose estimates diverge. The inflated measurement noise implies that the NEES should reflect somewhat conservative pose estimates if the filter is consistent.

In the first simulated sequence, the virtual camera views a fronto-parallel plane at distance 1 unit with evenly spaced landmarks. The camera strafes 0.2 units left and 0.4 units right, then returns to the origin, where it sits still for 240 time steps. Figure 6.5 shows the landmarks (yellow) and trajectory (blue). Figure 6.6 plots the average NEES over 25 Monte-Carlo runs of the pose estimates given by the three filters. The 95% bounds are drawn as horizontal lines. Note the scale of the FastSLAM results; the filter is highly optimistic. The pose estimate EKF SLAM is conservative at first, but then becomes inconsistent when the camera motion changes direction, and is optimistic from then on. The estimate of Graph SLAM is temporarily inconsistent at the acceleration point, but then returns to conservative behaviour.

The landmark configuration of the second simulated sequence is identical to the first, but the camera motion is more complex. The camera strafes 3 units to the left and right, while smoothly rotating about the horizontal and vertical axes, and moving along the optical axis. Figure 6.7 shows the landmarks (yellow) and trajectory (blue). Figure 6.8 shows the results. EKF SLAM starts conservatively, but becomes optimistic when the camera reaches the left end of the trajectory. It recovers consistency when passing

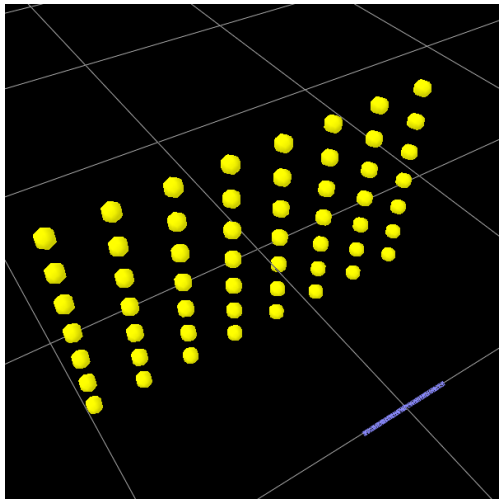

Figure 6.5: Map and camera trajectory for a simulated planar scene with simple motion

through the origin, but then becomes even more optimistic as the camera moves to the right. Graph SLAM is conservative for almost the entire run.

The final simulated scene is four vertical sides of a square box, with the camera performing two outward-facing circuits inside the box. The camera starts at 12 o'clock, and moves clockwise twice around the inside of the box, always facing away from the centre. Figure 6.9 shows the landmarks (yellow) and trajectory (blue). The average pose NEES plots are shown in Figure 6.10. The EKF SLAM pose estimate remains conservative until halfway around the first loop, when it becomes highly optimistic. Upon closing the loop, the estimate is again conservative. However, the second time around the loop, the filter is even more optimistic than the first, showing that the inconsistency is growing. FastSLAM remains conservative for a shorter time before giving highly optimistic estimates, and never recovers consistency. Graph SLAM is inconsistent only at the loop closing point, and the inconsistency is smaller the second time around. This reflects the refinement of the graph, bringing the map (and thus the pose) closer to the truth.

6.6.2 Performance on Real Video

While the consistency evaluation confirms that the Graph SLAM filter is considerably better at avoiding inconsistency than FastSLAM or EKF SLAM, it requires knowledge of the true camera motion and landmark positions. These quantities are difficult to know for real video sequences. However, following the approach of Chapter 4, the pose estimates can be directly compared to those given by bundle adjustment over the same observations. The globally optimised trajectory is the best the filter can hope to achieve given the observations it acquires.

A desktop sequence of 920 frames is mapped with varying parameters, and the pose estimate for each parameter set is compared to the output of bundle adjustment using the same observations (which vary with the parameters). The total diameter of motion of the camera is 2 units. Characteristic Graph SLAM visualisation output for the scene is shown in Figure 6.11.

The error is quantified by the root-mean-square (RMS) camera position displacement, relative to the bundle adjustment output, over all the frames in the sequence. Table 6.1 shows the error over the same sequence with varying parameters. The maximum number of landmarks estimated in a node's local map is N_L. This bounds the local computational cost of updates. When it gets very large, the filter approaches an iterated EIF. The parameter m_{min} is the minimum number of observations required each time step. If the number of observed landmarks drops below this, and there isn't enough room in the current node's map to make up the deficit by acquiring new landmarks, a new node is created (see Section 6.4.2). The target number of landmarks observable each time step is given by m_t. New landmarks are acquired, up to the limit in the local map, to meet this quota. As m_{min} and m_t increase, the number of nodes in the graph (and thus total number of landmarks) also increases.

A few conclusions can be drawn from the results. Generally, the localisation accuracy is reasonable relative to the total scale of the trajectory (less than 1% error). The localisation error generally decreases with more observations made each frame. How-

N_L	m_{min}	m_t	$\|V\|$	RMS Err.
50	10	20	3	0.0451
		30	3	0.0310
		40	4	0.0270
		50	3	0.0230
	20	20	3	0.0541
		30	7	0.0375
		40	10	0.0181
		50	6	0.0160
60	10	20	2	0.0297
		30	2	0.0277
		40	2	0.0187
		50	2	0.0472
	20	20	3	0.0255
		30	6	0.0298
		40	5	0.0159
		50	5	0.0184
70	10	20	3	0.0330
		30	3	0.0235
		40	4	0.0189
		50	3	0.0181
	20	20	3	0.0421
		30	7	0.0403
		40	10	0.0187
		50	6	0.0196
80	10	20	2	0.0309
		30	2	0.0296
		40	2	0.0242
		50	2	0.0295
	20	20	3	0.0304
		30	6	0.0286
		40	5	0.0171
		50	5	0.0187

Table 6.1: Localisation error relative to the output of global bundle adjustment for a desktop sequence, as the parameters are varied. N_L is the maximum number of landmarks permitted in a node, m_{min} is the minimum number of observations to make each frame, m_t is the target number of observable landmarks each frame, and $|V|$ is the resulting number of nodes in the final graph. The last column is root-mean-square position error over all time steps

ever, the ratio of target to minimum number of observations is also important. This is due to the crude heuristic for acquiring landmarks – when the landmark density m_t is high, but the minimum threshold m_{min} is low, the relatively few observations made while exploring new territory with a full node are not well-distributed in the view, so the localisation error is larger. This could be improved with a more sophisticated incremental acquisition strategy. However, increasing m_{min} gives a noticeable improvement, at the cost of a more complex graph. Values of $N_L = 60$, $m_{min} = 20$, and $m_t = 40$ seen to give good results over a range of sequences, while limiting the computational cost of local map updates (cubic in N_L).

6.6.3 Loop Closing

To test loop closing ability in a repeatable way, the system is given as input the same synthetically rendered planar loop sequences used in Section 4.6.5. In contrast to the FastSLAM system, Graph SLAM successfully closes all of the loops. This reflects the improved consistency of the filter, as the active search ellipses encompass the actual landmark in the image. Figure 6.12 shows the large image search ellipses corresponding to landmarks from the node furthest in the graph. The search regions are found by transforming the landmark estimates through the graph into the current node, and then into the image.

Figure 6.13 shows the graph being built during the course of a run on a rendered sequence of a loop with radius three, using default parameters ($N_L = 60$, $m_{min} = 20$, $m_t = 40$). When the loop is almost complete, landmark estimates furthest away in the graph have the largest uncertainty in the local coordinate frame (even though they are geometrically nearby). When landmarks from the first node of the loop are successfully re-observed and added to the local map, an edge is created between the active node and the first node, closing the loop. Immediately, the uncertainty of landmark estimates decreases, as the path in the graph is now much shorter and better constrained. Figure 6.14 shows the rendered map just before adding the edge, and 20 frames later at the end of the sequence. There are 1369 unique landmarks in the final map. The system processes every frame of the video in under $30ms$.

6.6.4 Limitations

Graph SLAM is considerably more consistent than EKF SLAM, but it also has its own failure modes. When a new node is created, the estimation process is overly sensitive to camera motion. Each node creation event is just like the system start; if the camera sits still or undergoes pure rotation, the new local map will converge incorrectly. This might be addressed by detecting a lack of camera motion and not updating the local graph. Initial investigation shows that this helps in some, but not all scenarios.

When the graph becomes large, with many edges and cycles, the optimisation process is computationally expensive. The cost might be mitigated by only optimising over a fixed radius of the graph, but then long cycles would never be rectified. Further, the edge optimisation assumes that each edge is statistically independent on all others, when in fact they are all correlated through the local map estimates. This isn't a problem on simple graphs, perhaps due to the reasons cited by **Bailey** (2002). However, when there are many cycles, the graph does not always converge to the correct local minimum.

Finally, the system does not by default produce a global map suitable for rendering. This can be easily produced by mapping all landmarks through the graph into a common frame, but the cost of this operation increases with the complexity of the graph. Of course,the rendering operation will always require linear time in the number of landmarks, but the total cost might be reduced by an amortised rendering scheme.

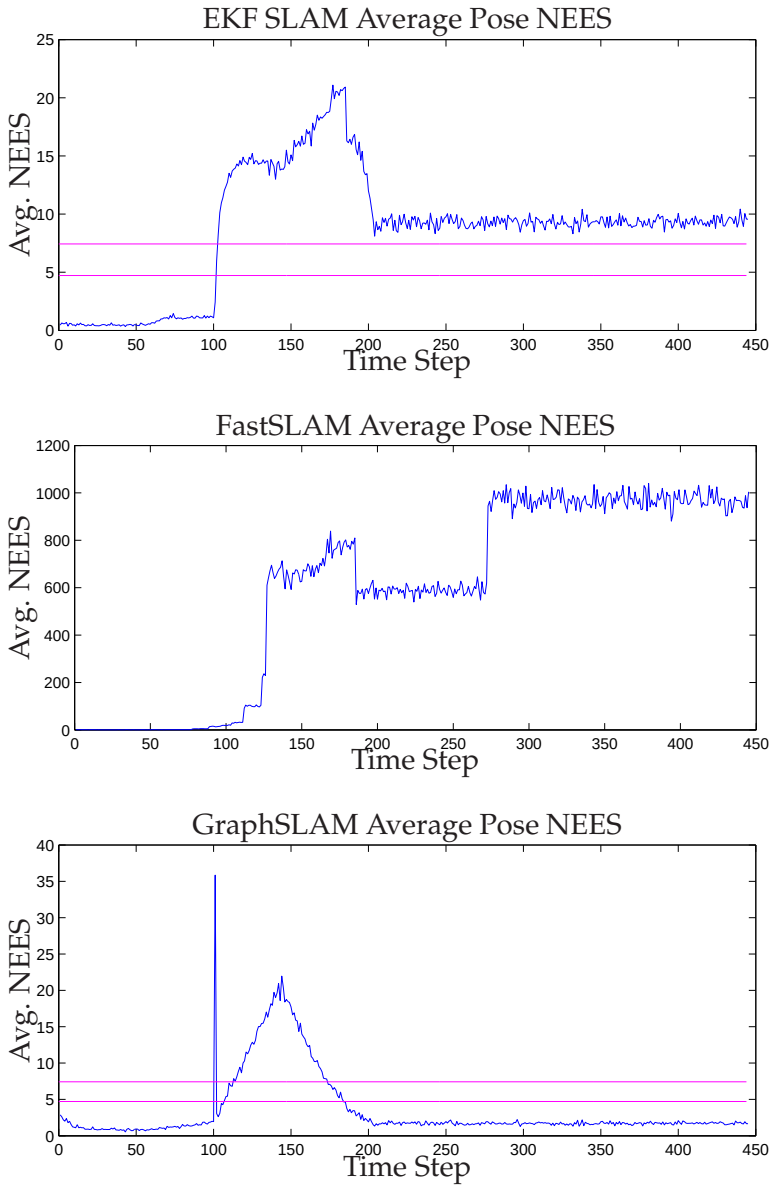

Figure 6.6: Average NEES of pose over 25 runs in a planar scene with simple motion

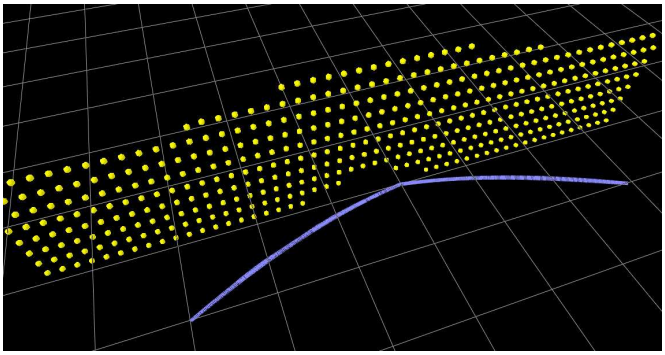

Figure 6.7: Map and camera trajectory for a simulated planar scene with complex motion

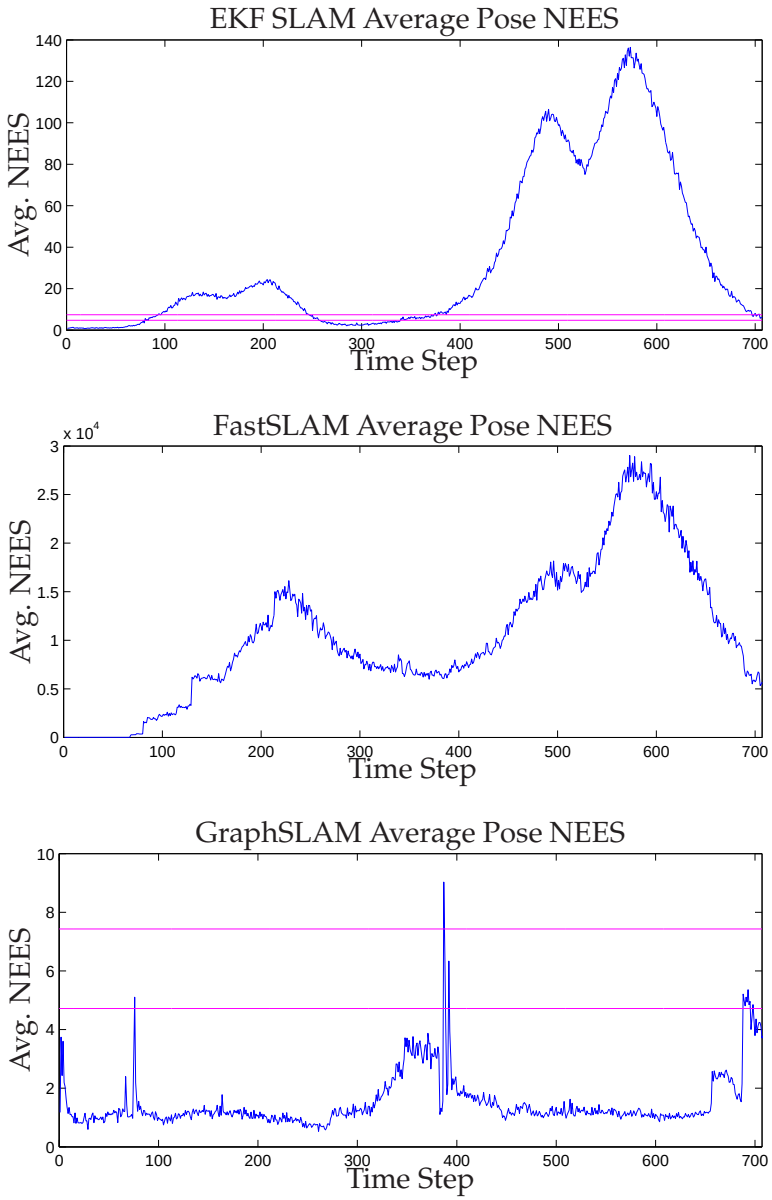

Figure 6.8: Average NEES of pose over 25 runs in a planar scene with complex motion

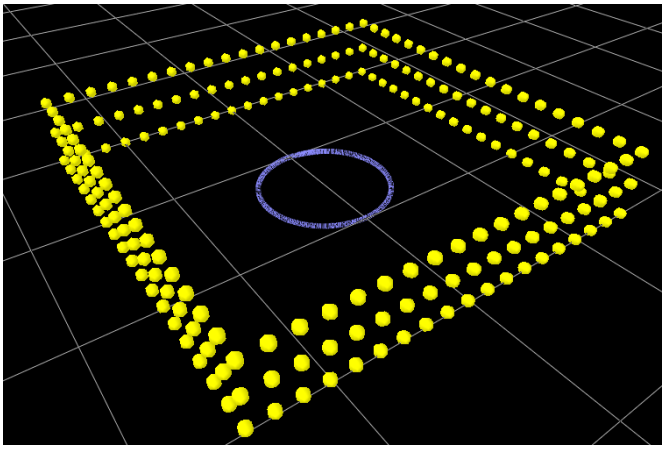

Figure 6.9: Map and camera trajectory for a simulated box scene with twice-looping motion

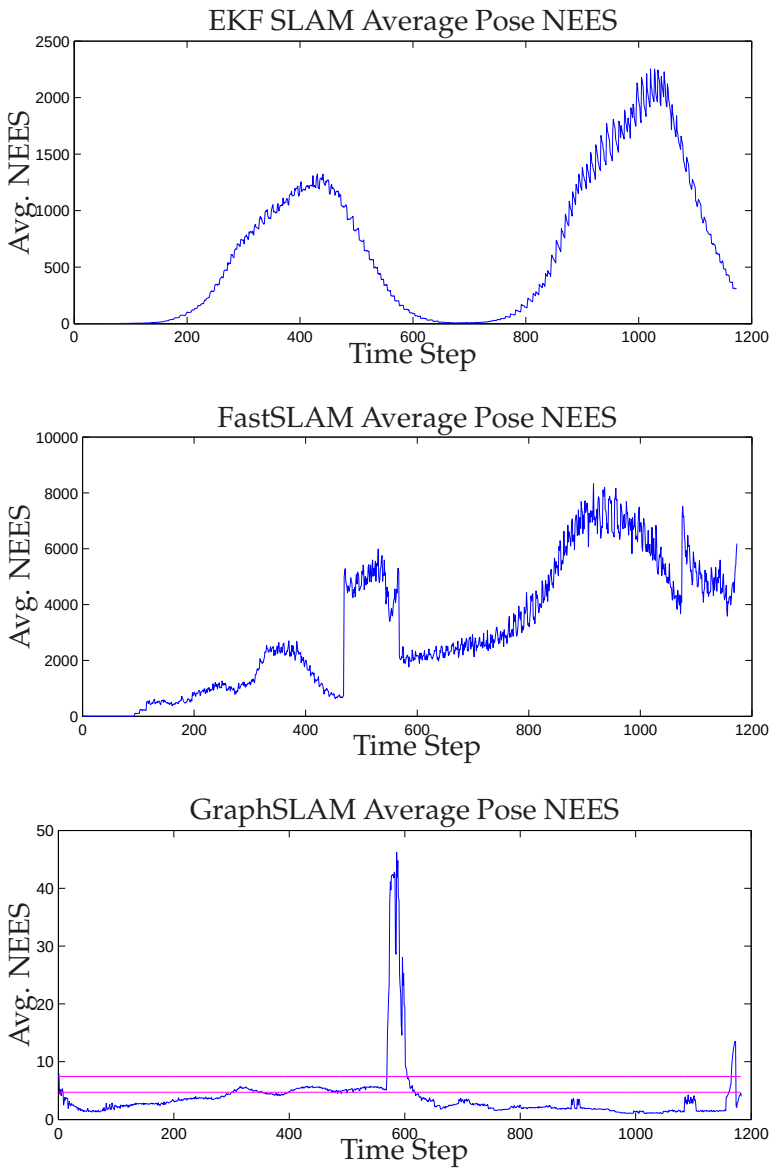

Figure 6.10: Average NEES of pose over 25 runs in a piecewise-planar scene with complex motion

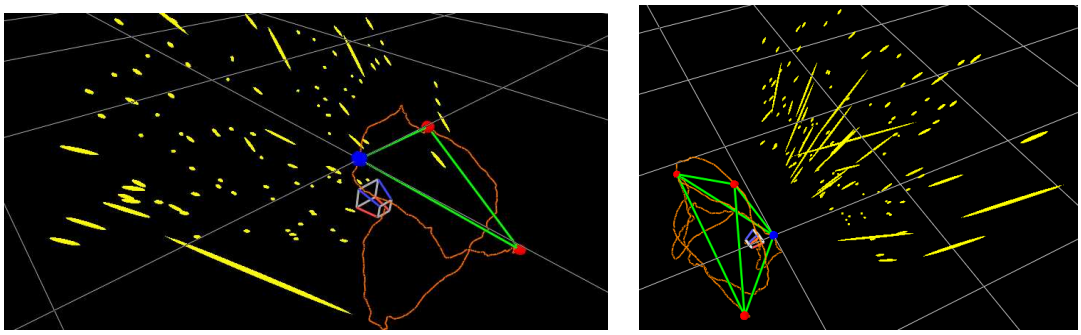

Figure 6.11: Graph and map at two points during a desktop sequence. The active node is blue; other nodes are red. Graph edges are green. The landmark 3σ ellipsoids are yellow. The camera trajectory is orange.

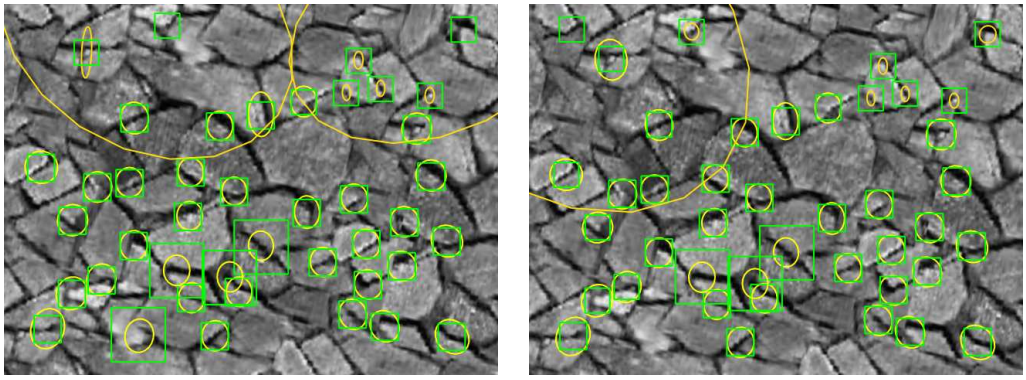

Figure 6.12: When nearing the starting point of the loop, the large search ellipses reflect the uncertainty in the location of the landmarks in the image. Once the landmarks are added to the current node, the uncertainty collapses

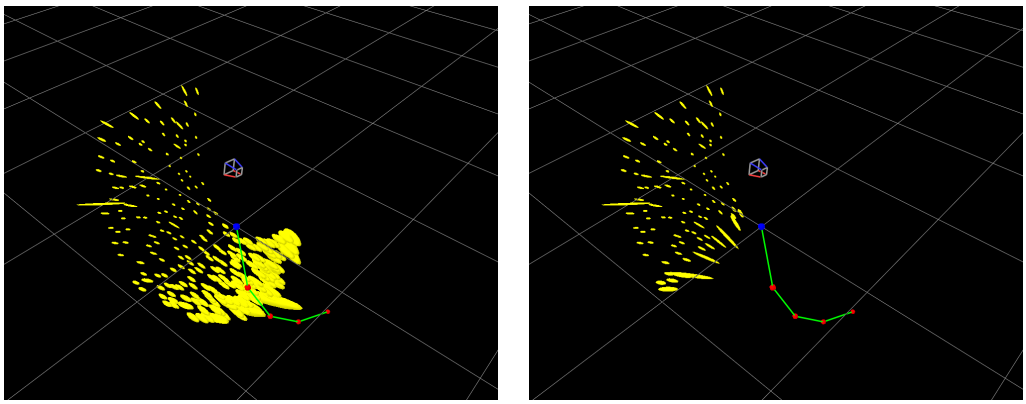

Figure 6.13: The map and graph during a synthetic planar sequence, rendering using only landmark estimates in nearby nodes (left) and using all estimates (right). The current active node (blue) is at the origin of the rendering

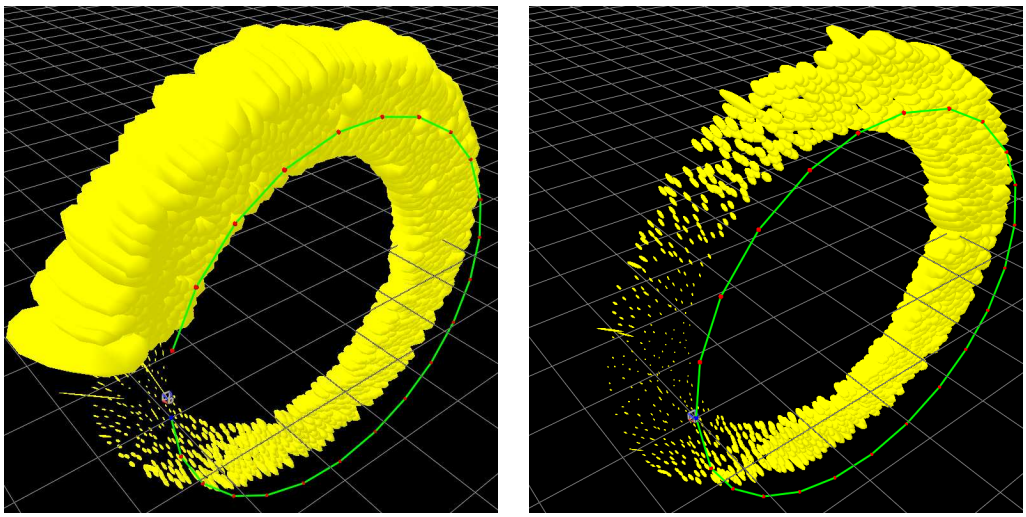

Figure 6.14: The map before (left) and after (right) closing the large circular loop in a synthetically rendered planar sequence. There are 1369 landmarks in total in the final map. The reduced uncertainty of landmarks in other nodes, mapped into the current coordinate frame, is reflected on the right

7

Unified Loop Closing and Recovery

7.1 Introduction

Most existing real time monocular SLAM systems rely primarily on tracking to perform localisation at each time step, using a motion model and active search to constrain the camera trajectory and update the map. Certainly, the EKF SLAM system of **Davison** (2003) and the approaches described in chapters 4 and 6 fall into this category. The efficient use of sensor data in these systems, formulated as active search, relies on the tracking assumption: The camera motion is assumed to be decently approximated by the dynamic model, and sufficient observations can be made at each time step to maintain steady tracking.

However, the tracking assumptions are easily violated by unmodeled motion or failure to find landmarks in the video due to blur, occlusion, or an insufficient appearance

model. Tracking failure causes incorrect data association and motion estimation, leading to catastrophic corruption of the structure estimate.

Even when the camera motion and environment are favourable, the statistical filtering often delivers inconsistent results on a large scale. This problem is not confined to monocular SLAM, but plagues metric SLAM in general. Resulting maps are locally correct, but globally incoherent, and nontrivial loops are difficult to close using the standard active search techniques.

Such fragility and inaccuracy make real-time monocular SLAM unusable for all but the most carefully constructed sequences, and motivate the development of out-of-band *recovery* and *loop closing* algorithms. Recovery refers to relocalisation of the camera pose following a tracking failure. Loop closing refers to data association between two logically distinct parts of the map that correspond to the same part of the environment, even when tracking is proceeding smoothly.

7.1.1 Contributions

This chapter proposes a unified framework for loop closing and recovery in monocular SLAM, and presents an instantiation of this framework building on the SLAM system of Chapter 6. The contributions are thus both general and particular:

- Formulation of recovery and loop closing as instances of the same association process in graph-based or submap SLAM
- General hierarchical method for performing this association in visual SLAM based on global and local appearance models and structure matching
- Implementation consisting of:
  - Visual bag-of-words model constructed online
  - Local appearance dictionaries built from active search
  - Structure matching using MLESAC and the three-point-pose algorithm

First, related work is reviewed in Section 7.2. The proposed framework and general hierarchical approach is described in Section 7.3. Section 7.4 discusses the image features used for the appearance models. Section 7.5 describes the global appearance model, while Section 7.6 describes the local appearance model. Then Section 7.7 explains the operation of the unified loop closing and recovery algorithm. Section 7.8 presents performance results and suggests further avenues of investigation.

7.2 Related Work

7.2.1 Recovery

The SLAM algorithm of **Pupilli & Calway** (2005) (see also Section 4.1.2) uses a particle filter to model camera pose, which makes the tracking robust to erratic motion, but fails to account for dependence between the camera and landmark estimates, and cannot make coherent maps for many landmarks. The tracking recovery mechanism relies on the camera continuing to observe existing structure immediately after failure.

Williams et al (2007) present a robust relocalisation method built into the EKF SLAM system of **Davison** (2003). Classification with randomised trees**Lepetit et al** (2005) yields image-to-landmark matches, from which pose is recovered when tracking has failed. However, classification using randomised trees breaks down in the domain of thousands of classes, and the online class training and storage costs (30ms and 1.25MB per landmark) are prohibitive when dealing with many landmarks each time step.

Chekhlov et al (2007) employ invariant descriptors based on SIFT (**Lowe** (2004)) to increase the robustness of SLAM tracking. Active search by correlation of patches is replaced with matching of these descriptors in gated regions. Multiple exemplar descriptors are used to identify landmarks under significant viewpoint changes. This technique allows robust tracking under camera shake, and limited relocalisation when tracking fails. However, if the camera suddenly leaves the region, then the constrained

matching procedure cannot accommodate the complete pose uncertainty when the camera returns at a later time.

7.2.2 Loop Closing

Clemente et al (2007) have extended Davison's system to allow loop closing when two sub-maps have independently chosen the same landmarks from similar viewpoints. However, the loop closing is not efficient (taking more than 1 minute), and the loop detection conditions are rarely satisfied in practice (**Williams et al** (2008)).

Loop closing using visual appearance is not a novel idea; the richness of camera data makes it particularly suited to the task of recognising similarity. **Dudek & Jegessur** (2002) use descriptors derived from principal component analysis over Fourier transformed image patches to describe and match frames, and then use a vote over descriptors to choose a database image. **Newman et al** (2006) build a similarity matrix to evaluate the statistical significance of matching images when laser range data also matches.

Sivic & Zisserman (2003) apply the bag-of-words model, previously used in text retrieval, to perform content-based retrieval in video sequences. Affine-invariant descriptors extracted from the videos are clustered at training time, and then quantised to the cluster centres at run time to yield visual word histograms in the images. Potential matches are ranked using the term-frequency inverse-document-frequency metric.

Nistér & Stewénius (2006) extend the framework of **Sivic & Zisserman** (2003) to very large vocabularies (in the millions) by using a hierarchical clustering scheme. Each level of the hierarchy splits the word set into $\sim$10 clusters, which are then recursively split in the branches. A query image is matched to the database by propagating its words through the tree toward the leaves, computing a query vector according to the weights assigned to nodes. The query vector is then a descriptor for the image, matched against the database images using an inverted index. While this method

yields excellent matching results into large databases of words and images, the training and storage costs for such large vocabularies is very high (not only with this clustering method).

The appearance-based SLAM work of **Cummins & Newman** (2007) applies the bag-of-words method within a probabilistic framework to detect loop closures. A generative model of word expression yields a likelihood of observed words over stored places, permitting maximum-likelihood data association and update of the place's appearance model parameters. The generative model establishes conditional dependencies in word expression by computing a Chow-Liu tree in training. While the system delivers high accuracy visual matching, the generative model must be computed offline and the model update cost at each time step is high.

Very recent work by **Williams et al** (2008) uses the randomised-trees relocalisation method described above to close loops in submap-based SLAM. This loop-closing approach is very similar to the one presented here. However, it shares the drawbacks listed above for randomised-trees classification. Further, relocalisation is tried against each submap in turn, in a brute-force manner. The hierarchical visual appearance model described by this chapter focuses the search for correspondences.

7.3 Method Overview

7.3.1 Motivation

In a SLAM algorithm operating on a set local maps, such as the one described in Chapter 6, the maps encode local structure estimates, and the connections between the maps (e.g. edges) describe the transformations between local coordinate frames. When camera motion is smooth and tracking is successful, the local maps are updated as the pose estimate moves through the set via the connections.

When something goes wrong, such as unmodeled camera motion or dynamic occlusion from moving objects, not enough is known about the motion of the camera to localise it within the set of existing maps. The system can detect this failure and avoid corrupting the map by ceasing filter updates. Then it can continue to look for known scenes and try to relocalise when they are detected, recovering tracking. This is the approach of **Williams et al** (2007).

However, this policy is satisfactory only when known structure is likely to be encountered again soon after tracking failure. Consider a trajectory in a large loop, with a tracking failure near the beginning of the loop. If mapping stops at the failure, and the relocalisation algorithm doesn't find the known structure until the camera returns to the beginning of the loop, all of the potential information from the middle of the sequence is lost. Instead, mapping should continue in the middle of the sequence, and the new map should be connected to the old one upon recognition of the beginning of the loop.

Further, the recognition of known scenes and structure yields data association in general, not only when tracking has failed. An algorithm that solves the relocalisation problem should also be able to aid the closing of loops among a set of local maps.

7.3.2 Unified Method

Given a graph of local maps connected by coordinate transformation estimates, recovery and loop closing both reduce to the task of recognising already-mapped structure in the environment, thereby creating new connections in the graph. This can be accomplished robustly and efficiently by using a hierarchical appearance model and taking advantage of existing structure estimates.

At the coarse level, a global appearance model encodes the rough appearance characteristics of different local maps. Then every image captured by the camera can be represented in a similar fashion and compared to these appearance signatures, in order to identify local maps that include scenes visually similar to the query image.

The appearance of landmarks within each map is better modeled at the finer, local level. Associated with each local map is a local appearance model, which matches interest points of a query image with landmarks in the map. The model for each landmark is rich enough to represent the landmark's varying appearance within the local map.

Loop closing or recovery proceeds as follows.

1. The global appearance database is queried to find candidate local maps that include scenes visually similar to the current video image.
2. Interest points in the image are matched against the local appearance models of the resulting maps, yielding hypothetical correspondences between locations in the image and landmarks in the map.
3. These correspondences are checked for coherent structure using the candidate map's estimate, possibly giving as output a pose estimate relative to the map.
4. The relative pose estimate is treated as a candidate edge over multiple time steps, during which landmarks from the candidate map are observed by transforming their estimates through the candidate edge.
5. Once the candidate edge is confirmed by sufficiently many observations, it as promoted to a graph edge.

When tracking fails, a new graph component is created and mapping continues. If the procedure above succeeds and creates an edge between nodes in two different connected graph components, a recovery or reconnection has occurred. If the two nodes connected by the new edge are in the same component, a loop has been closed, and a cycle has been added to the graph.

7.4 Invariant Features

The appearance models used in loop closing and recovery should be robust to moderate viewpoint changes, so that loops can be closed or recovery performed without requiring the camera to exactly revisit a previous pose. To this end, the models are built from distinctive feature descriptors of interest points robust to viewpoint changes. Detectors and descriptors for these sorts of features abound, and have been the subject of much research.

Examples of detectors that have some degree of reliability over scale and affine image deformation include FAST (**Rosten & Drummond** (2005)), difference-of-Gaussians (**Lowe** (2004)), Harris-affine and Hessian-affine (**Mikolajczyk & Schmid** (2004)), Fast-Hessian (**Bay et al** (2008)), and MSER (**Matas et al** (2002)). Descriptors computed around detected interest points include SIFT (**Lowe** (2004)), PCA-SIFT (**Ke & Sukthankar** (2004)), MOPS (**Brown et al** (2005)), GLOH (**Mikolajczyk & Schmid** (2005)), and SURF (**Bay et al** (2008)).

In combination with the difference-of-Gaussians (DoG) scale space detector, the SIFT descriptor has been widely tested in the literature, and fares well in a performance evaluation comparison with other descriptors (**Mikolajczyk & Schmid** (2005)). The SURF descriptor is similar, but sacrifices the isotropic detection and description of SIFT. PCA-SIFT and GLOH are also inspired by SIFT, but both require offline PCA from large datasets. Most importantly, **Sivic & Zisserman** (2003) report that the SIFT descriptor gives good performance in their bag-of-words image retrieval application.

The DoG detector and the SIFT descriptor are attractive options for constructing the global and local appearance models. The problem with using DoG+SIFT in this setting is the cost of computation. In order to achieve reasonably efficient online operation, the detector is highly optimised and the descriptor computation is reduced.

The DoG detector, as described by **Lowe** (2004), approximates scale-space with a discrete pyramid of Gaussian-blurred images. The Gaussian convolution can be very

expensive if implemented naively. However, downsampling the image (to 320×240) and omitting the recommended initial upsampling removes two levels of the pyramid, while eliminating only the smallest image features, which are most sensitive to noise. Further, using the recursive Gaussian filter of **Young & van Vliet** (1995), with the improvements suggested by **Triggs & Sdika** (2006), the pyramid construction time is significantly reduced.

The descriptor computation proceeds similarly to the standard routine of **Lowe** (2004). The orientation of the patch around a detected corner, at the scale of detection, is estimated, and the patch is mapped into a canonical 13×13 representation using bilinear interpolation. If multiple orientations are detected, then multiple corresponding patches are generated.

For each such canonical patch, the polar gradient is computed and multiplied by a 2D Gaussian envelope, decreasing the influence of values on the edges. Separate histograms of gradient angle, weighted by gradient magnitude, are computed over a grid on the patch, with trilinear interpolation in the grid and angle bins. In the original formulation, there are 4×4 grid elements, each with 8 angle bins, resulting in a 128D vector. To reduce the description, storage and lookup cost, this is reduced to a 2×2 grid with 4 angle bins in each element, giving a 16D vector. As in **Lowe** (2004), the vector is normalised to unit magnitude, clamped so that no value exceeds 0.2, and then renormalised.

These modifications allow efficient detection and description of interest points. On a 3 GHZ Intel Core 2 Duo, the detection algorithm takes approximately 6 ms, and the descriptor computation takes 7 ms for the 250 strongest interest points. Detection and description for each video image are performed on the second processing core, in parallel to the SLAM algorithm, so their output is ready for use at 6 ms and 13 ms into the time step, respectively. The resulting 16D descriptors, while likely sub-optimal, perform more than well enough to show that the system works.

7.5 Global Appearance Model

The global appearance model is used to select nodes visually similar to the current video image, based on a visual bag-of-words. The visual words are created and clustered online. Each node builds a word expression signature to describe its appearance, against which video images are compared.

7.5.1 Bag-of-Words

The bag-of-words model, originally used for text processing, treats a document as an unordered collection ('bag') of words, each of which is listed in a central dictionary. The document can then be described by a histogram over word expression, counting how many times each word appears. This description allows similar documents to be retrieved from a database, using any of a variety of similarity scores.

Sivic & Zisserman (2003) extend this model to image description by treating each image as a document, and quantising feature descriptors in the image to *visual words* in a fixed vocabulary. SIFT descriptors of interest point tracks are harvested from a subset of the frames in a video, and the resulting 200,000 descriptors are clustered using k-means into 10,000 cluster centres. The cluster centres are the visual words. The descriptors for interest points in a query image are quantised to these words, giving a histogram over word expression. Then frames from the database are ranked using the *term-frequency inverse- document-frequency* (TF-IDF) metric.

TF-IDF weights words proportionally to their expression in the query image and inversely to their expression in the database. Let the set of all visual words be V, and let $e_i(q)$ be the expression count of word $v_i \in V$ in the image q. Then the term frequency $\mathrm{tf}_i(q)$ of word v_i is given as the fraction of total word expression for which v_i is responsible in the image:

$$\mathrm{tf}_i(q) = \frac{e_i(q)}{\sum_{v_j \in V} q_j} \tag{7.1}$$

The inverse document frequency idf_i for word v_i increases with the rarity of the word in the database. Let F represent all frames in the database. Then

$$\mathrm{idf}_i = \log \frac{|F|}{\sum_{f \in F} e_i(f)} \tag{7.2}$$

The TF-IDF vector $\mathrm{tfidf}(q)$ for an image q is then specified component-wise for each word:

$$\mathrm{tfidf}(q)_i = \mathrm{tf}_i(q) \cdot \mathrm{idf}_i(q) \tag{7.3}$$

Given a query image q, images $f \in F$ are ranked according to the cosine of the angle between the query and database TF-IDF vectors:

$$\mathrm{score}(q, f) = \frac{\mathrm{tfidf}(q)^T \mathrm{tfidf}(f)}{\|\mathrm{tfidf}(q)\| \|\mathrm{tfidf}(f)\|} \tag{7.4}$$

An equivalent ranking results from taking the Euclidean distance between normalised vectors as the difference metric. Note that the inner product need only be computed on words existing in both q and f. If no words expressed in q are expressed in f, it need not be considered at all.

7.5.2 Online Clustering

Sivic & Zisserman (2003) compute the vocabulary offline by extracting descriptors from a similar set of images that will be used as queries, clustering with k-means. This offline training can be expensive, as the clustering requires many iterations for convergence given thousands of clusters and descriptors.

In the interest of avoiding any offline training requirements, and in order to tailor the vocabulary to the run-time environment, online vocabulary construction is used for the global appearance model. The clustering method employed is crude but effective. It relies on a fixed cluster radius, r_g, which is chosen by analysing the relation between cluster radius and vocabulary size in offline clustering. For this work, $r_g = 0.3$. Since the descriptors are of unit length, the largest possible distance between any two

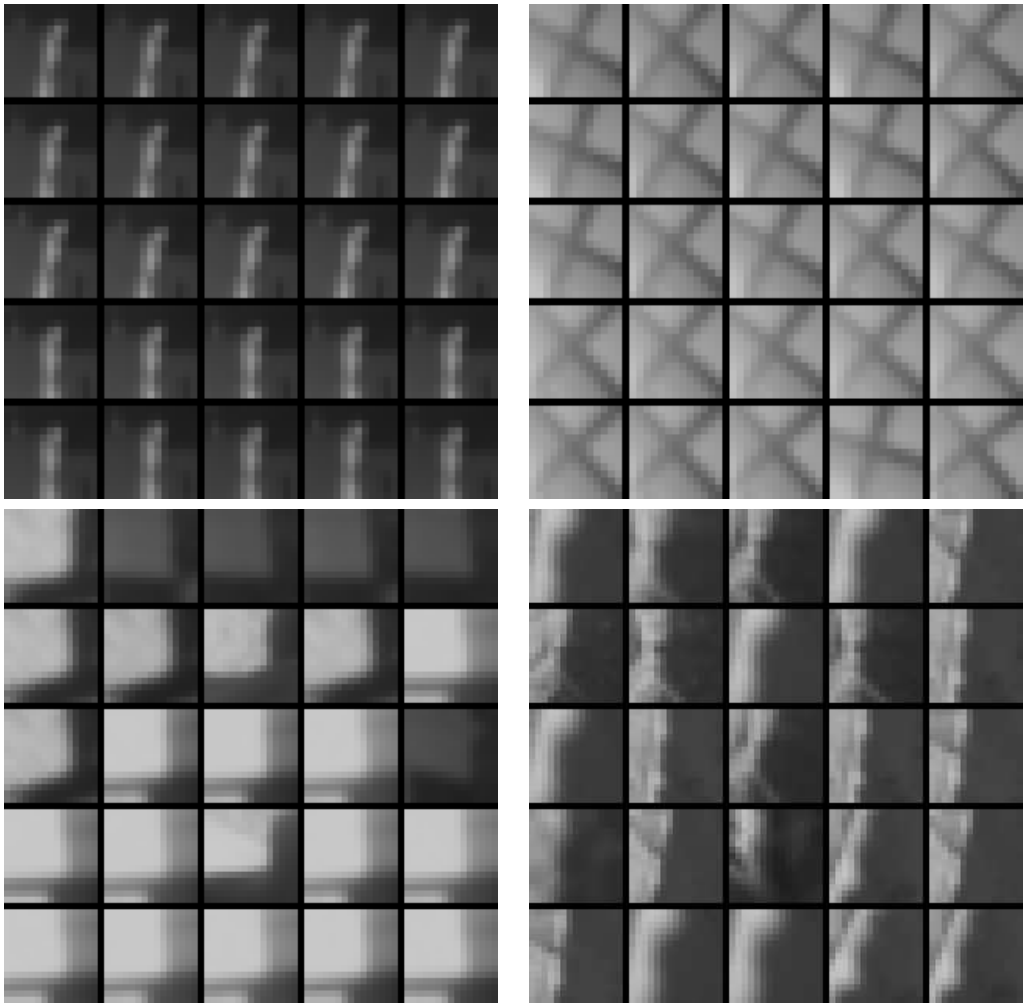

Figure 7.1: Example image patches that quantise to each of four visual words

descriptors is 2, if they are antipodes on S^15. Example word clusters are shown in Figure 7.1.

Spill trees are used to perform descriptor quantisation and online clustering. The spill tree data structure is presented by **Liu et al** (2004) as an extension of metric trees. In metric trees, a set of points is recursively partitioned by choosing directions of high variance, projecting the points on to the 1D subspace, and partitioning around the mean or median in that subspace. The nearest neighbour of a query point x is found in a metric tree using depth first search. At each node in the tree, the branch corre-

sponding to the side of the splitting hyperplane in which x lies is followed first, and the current nearest neighbour hypothesis h is maintained. The other branch is taken (recursively) only if the lower bound on distance to points in that branch (imposed by the splitting plane) is smaller than the distance to h. Most of the cost of queries comes in this backtracking phase. A spill tree allows overlap in the splitting, so that the need to backtrack is reduced. Using a greedy algorithm, only the favourable branch is taken all the way to a leaf, yielding an approximate nearest neighbour. This defeatist search in a spill tree is much faster than the exact search in a metric tree, and usually returns the correct result.

The vocabulary, V, is empty when the system starts. In order to avoid adding spurious or ephemeral words to V during online clustering, a second vocabulary, Y, the young word database is also maintained, and also is empty to start. The SIFT-like descriptors of Section 7.4 D are computed for all DoG interest points detected in the video image.

A spill tree on V is maintained, and the nearest neighbour w of each descriptor $d \in D$ is queried in the tree using defeatist search. In the simple case, $\|w - d\| \leq r_g$, so the descriptor is quantised to w, and the expression count of w for the video image is incremented.

If no existing word is found within distance r of d, the tree is queried again using exact (metric tree) search. If the exact nearest neighbour in V is still farther than r_g from d, then d could be an as-yet unseen word.

Then the nearest neighbour $y \in Y$ to d is found. If $\|y - d\| > r_g$, then d_i is added to Y as a new young word. Additionally, two counters are associated with d: $\mathrm{ttl}(d)$ and $\mathrm{occur}(d)$, corresponding to time-to-live and occurrence count respectively. The time-to-live is initialised to the constant ttl_0, and the occurrence count is initialised to 0, as the word has not been reobserved yet.

If instead $\|y - d\| \leq r_g$, then d quantises to y in the young word database. The time-to-live $\mathrm{ttl}(y)$ is reset to ttl_0, and the value of $\mathrm{occur}(y)$ is incremented. If $\mathrm{occur}(y)$ reaches

a threshold $\mathrm{min\_occur}$, y is removed from Y and added to the main vocabulary V as a word. By default, $\mathrm{min\_occur} = 3$.

At each time step, the value of $\mathrm{ttl}(y)$ is decremented for each $y \in Y$. If $\mathrm{ttl}(y)$ reaches 0 before y is promoted to V, y is removed from Y and discarded. Thus words are discarded that are not seen at least once every ttl_0 time steps, for $\mathrm{min\_occur}$ times in a row.

7.5.3 Per-Node Expression Histogram

A word expression histogram is associated with each node in the graph. Thus, the database of images used by **Sivic & Zisserman** (2003) is replaced by a database of local maps. Instead of a database histogram describing what visual words are expressed in one image, it describes what words are expressed in all the video images associated with a node.

At the end of each time step, the word expression signature of the current active node is updated using all of the words expressed by the current video image. The expression histogram for the current image has already been computed and used to update the global vocabulary. It is added to the expression histogram for the active node. This cumulative per-node histogram constitutes the nodes' general appearance model in terms of the global vocabulary.

Whenever a word w is expressed at least $\mathrm{min\_expr}$ times in a node, the node is added to a global inverted index mapping words to those nodes that express them. So until w has been expressed at least $\mathrm{min\_expr}$ times in a node, that node is ignored in the ranking process with respect to w. By default, $\mathrm{min\_expr} = 3$.

7.5.4 Visual Similarity Ranking

Given the word expression histogram for the current video image, visually similar nodes in the graph are identified using the TF-IDF ranking. Because the similarity search is used for loop closing and recovery, the current active node and any nodes connected to it are omitted from the ranking.

The ranking is efficiently computed by iterating through each word w expressed by the image, and computing TF-IDF components for the nodes listed in the global index as having expressed w. The scores for nodes expressing words expressed by the image are thus incrementally computed, and the best k are returned as the candidate nodes most likely to be visually similar to the current image. By default, $k = 2$.

7.6 Local Appearance Model

7.6.1 Local Landmark Descriptor Dictionary

Once a candidate node α is selected using the global appearance model, points in the current image must be associated with landmarks in α. These correspondences are established using the local appearance model L_α associated with α, which encodes the varied appearances of the landmarks observed in α. L_α is constructed similarly to the global appearance model V, except that only observations of landmarks made while α is active are used as input. Each entry in $L\alpha$ is a pair (w, l_w), where w is a descriptor vector and l_w is the landmark associated with the descriptor.

Assume node α is active. Every time an image search for landmark l_i is successful, a descriptor d_i is computed at the position and scale of the observation mean in the image. The nearest neighbour $(w, l_w) \in L$ to d_i is found (using a metric tree on L). This local nearest neighbour search is bounded by the local cluster radius r_l. If $\|w - d_i\| \leq r_l$, and $l_w = l_i$, then the appearance of l_i is already represented (to within r_l in descriptor space) by L.

If $\|w - d_i\| > r_l$ or $l_w \neq l_i$, then this appearance of the landmark is not well-represented by the local appearance model. The pair (d_i, l_i) is inserted into L. The local cluster radius r_l is motivated by the observed distance between descriptor values for observations of the same landmark relative to those of distinct landmarks. By default, $r = 0.15$. Usually, the number of entries in L for any one landmark is between one and ten. Often, the entries correspond to the three or four descriptors of right-angle rotations of the landmark patch, which are the possible local minima of the orientation detection stage of the descriptor computation.

7.6.2 Modified Landmark Acquisition and Search

In order to increase the reliability of matches between interest points in a video image and the appropriate local appearance model, new landmarks are acquired from the same DoG interest points used for the appearance models, instead of from FAST corner locations as described in Chapter 6.

Because each DoG point has an associated scale, a patch for the landmark is acquired at that scale from the image of first observation. The scale can vary over several octaves (in powers of two) of the image, so patches are grabbed from the image of appropriate resolution. These scaled images are available from the scale pyramid already computed for DoG detection.

When observing landmarks using active search and ZNCC, the predictively-warped patches have a similarly wide range of scales. Again, the appropriate level of the scale pyramid is chosen, wherein the patch has a reasonable size (neither too big nor too small). The varied resolution of such searches is accounted for by multiplying the measurement noise by the appropriate scaling factor.

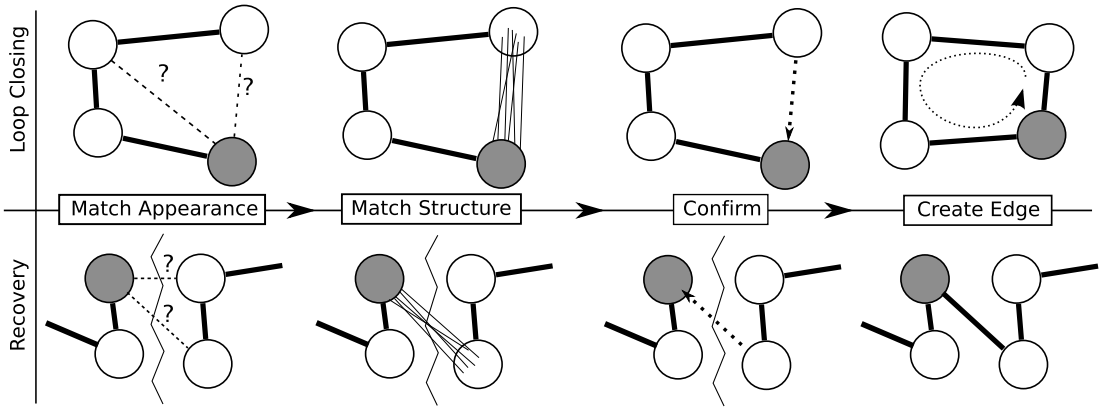

Figure 7.2: Loop closing and recovery: Candidate matching nodes are chosen by visual appearance similarity. Then structure is matched using landmark appearance models, and a candidate edge is created. Observations are made via the candidate edge until it is promoted, at which point a cycle is created (top) or two components are connected (bottom). The active node is shaded

7.7 Loop Closing and Recovery

Given the appearance models established by Section 7.5 and Section 7.6, the machinery for the loop closing and recovery method is prepared, and the procedure is described here. Both the loop closing and recovery circumstances are depicted in Figure 7.2.

7.7.1 Successful and Failed Tracking

When motion is smooth and fits the dynamic model, and active search is working as designed, tracking is successful. Mapping then proceeds according to the algorithm described by Chapter 6. However, when very few observations are made, tracking is considered failed. Instead of corrupting the maps in the current graph, or waiting until recovery is detected, a new graph component is created.

A global list of disjoint graph components is maintained by the system. Upon the creation of a new graph component, mapping begins afresh just as when the system started. In the absence of recovery, these components remain disconnected. The pose

estimate is always given relative to the active node in the current connected component.

7.7.2 Global and Local Appearance Matching

The first step in loop closing or recovery is identifying those nodes which are visually similar to the current video image. Section 7.5 explains the appearance model used for this step, and in particular, Section 7.5.4 describes how the nodes are ranked by visual similarity. Each of the top k ranked nodes is a *candidate node* for matching. Each candidate node is considered in turn.

The descriptors of the current image are matched against the local appearance model L_α of candidate node α to get correspondences between interest points in the image and landmarks in the local map of α. This descriptor matching in L_α is made more robust by finding the two nearest neighbours (n_1, l_1) and (n_2, l_2) to each descriptor d from the image. If $\|n_1 - d\| > r_l$, the match is rejected outright. Otherwise, if $l_1 \neq l_2$, the distance ratio test of **Lowe** (2004) is applied: If $\|n_2 - d\| < 1.2\|n_1 - d\|$ the match is rejected.

The result is a set of hypothetical correspondences $\{(p_i, l_i)\}$ between interest points in the image and landmarks in the local map of α. Some or all of these correspondences may be spurious, due to repeated texture and noise in the interest points and descriptors. Additionally, no uniqueness constraint is applied to the landmark correspondences; multiple interest points might match to the same landmark in the local map. Outliers and incorrect hypotheses are eliminated by structure matching.

7.7.3 Structure Matching

The structure matching algorithm is similar to the one used by **Williams et al** (2007) for relocalisation, though the input correspondences are produced by a different process.

MLESAC (**Torr & Zisserman** (2000)) and the three-point-pose algorithm are used (**Fischler & Bolles** (1981)) to identify the inliers in the set of hypothetical correspondences. Any three matches from descriptors to distinct landmarks in the candidate node determine a camera pose relative to the local map of candidate node α. For many such poses, the maximum-likelihood set of inlier correspondences are computed, with a fixed log-likelihood threshold of -5.0. Thus, under a pose hypothesis, if the log-likelihood of a correspondence, taking account measurement noise, is less than -5.0, the correspondence is considered an outlier for that pose. For each pose, the correspondence per landmark of highest likelihood is chosen if multiple correspondences with the same landmark exist.

Up to 200 random three-point-pose hypotheses are tested in this manner. If the highest-likelihood pose M has at least 7 inliers, the structure matching is considered successful. The estimate of M is refined using all of the inliers, and a directed *candidate edge* is created, pointing from α to the current active node. The transformation estimate $S \sim \mathcal{N}(\hat{S}, \mathbf{\Sigma}_S)$ associated with the candidate edge is initialised using both $M \sim (\hat{M}, \mathbf{\Sigma}_M)$ and the current pose estimate $C \sim \mathcal{N}(\hat{C})$ relative to the active node. In particular, the mean of the transformation is given by

$$\hat{S} = \hat{C}^{-1}\hat{M} \tag{7.5}$$

The covariance $\mathbf{\Sigma}_S$ is computed analogously. Note that the scale of this transformation is unknown, but is not required for the confirmation stage.

7.7.4 Confirmation

A candidate edge e created from structure matching is not necessarily a valid correspondence between the current scene and the local map of the candidate node α. It must be vetted over multiple time steps and motion of the camera, to ensure that the match was not a fluke. This confirmation is achieved by checking that the estimates of landmarks in α, transformed through e, are observable in the video over multiple frames.

A list of candidate edges pointing into the current active node is maintained. At each time step, after the pose estimate has been updated with the latest observations, landmark estimates from the source nodes of these candidate edges are projected into the image, according to the candidate edge transformation and the current pose estimate. Observations of these landmarks are attempted using the standard active search procedure. Because the pose has been well-constrained by local observations, the search ellipses reflect mostly the uncertainty in the candidate edge transformation, which is usually small. Thus the search ellipses are small, and the active search is very efficient.

Any observations made of such landmarks are used only to update the confidence measure of the candidate edge. A counter keeps track of the number of successful and failed observations. When both the number of successes and the ratio of successes to failures are sufficiently high, the edge is confirmed. If either the number of failures is too high or the success rate is too low, or if the candidate edge has been in the confirmation stage for many time steps, it is discarded.

7.7.5 Edge Creation

A confirmed candidate edge is promoted to a normal graph edge, with its rigid transformation unchanged, its scale set to unity, and its uncertainty in scale effectively infinite. The latest observations used to confirm the edge are also promoted to normal observations, so that landmarks from the now-connected node can be added to the local map of the active node. Recall that the new edge's transformation will be constrained only by estimates of such shared landmarks.

If the candidate node is in a different graph component than the active node, then the components are merged in a *reconnection* event. However, if the younger of the two components is very young (less than 2 seconds old by default), then it can be considered to contain little useful mapping information, and it is discarded. In this case, the pose estimate is transformed into the candidate node by the edge transformation, and the candidate node becomes the active node. This is a *recovery* event.

Otherwise, the new edge is directly connecting two nodes in the same graph component, in a *loop closing* event. This introduces a new cycle in the component. Recall that cycles induce constraints on the graph, which are satisfied and optimised iteratively (see Section 6.5). A loop closing event often causes rapid modification of the graph edges in the optimisation, as the residual is re-minimised under the additional constraints.

7.8 Results

The described method is implemented on top of the system of Chapter 6 to run in real time on a dual-core computer. Image searches and filter updates proceed in parallel with interest point detection, descriptor computation, and global appearance maintenance. On a 2.2 GHz Pentium Core 2 Duo, per-frame processing never exceeds 33 ms. The system successfully closes loops and recovers from tracking failure in both indoor and outdoor sequences, while operating in real time and mapping thousands of landmarks.

A completely planar real scene is used as a basic test of reconstruction accuracy. The camera hovers at typical viewing distance h above one part of the scene, before being kidnapped to the other half (the lens is covered during the motion). The system continues mapping by creating a new component. When the camera again views the original portion of the scene, the two components are matched and reconnected. The final map contains 251 landmarks. All 226 landmarks with depth uncertainty $\sigma < h/50$ are no farther than $h/100$ from the maximum likelihood plane.

In an outdoor sequence, the camera moves in an elliptical loop, with the camera facing outwards. Rough camera motion causes tracking failure, but the system immediately recovers. Extended failure occurs when the camera is suddenly rotated toward the ground. Mapping of novel views then continues in a new component. As the camera returns to near the starting point, a node in the first connected component is recognised and matched, and the components are merged. As the trajectory continues

around the loop a second time, the loop itself is closed. The resulting map contains 1043 landmarks.

In an indoor scene, a complex external loop is traversed and closed. Then the camera is repeatedly kidnapped from one part of the environment to another, with new viewpoints significantly different from the originals. In all cases, recovery occurs within 15 frames. The final map contains 1402 landmarks.

7.8.1 Future Work

The method presented in this chapter greatly improves the robustness of real time monocular SLAM, but is not flawless. The worst failure mode of the system is spurious loop closure given extensive repeated structure, causing massively incorrect graphs. In testing, this occurs only in synthetic sequences with large, exactly repeating textures. The problem is particularly difficult to solve in general, as repeated structure at arbitrary scales could always be encountered, theoretically. A probabilistic model for appearance-based loop closure, such as the method of **Cummins & Newman** (2007, 2008), could mitigate the issue.

Another problem is that the cycle optimisation treats the edge transformation estimates as independent, though they are in fact correlated through the node estimates. The assumption is not harmful when the graph contains few cycles, or mostly short cycles, but results in over-confident (and thus inconsistent) global maps when many loops are optimised. The optimisation strategy could be replaced by conservative local graph adjustments, or by global bundle adjustment over all edge transformations.

While the bag-of-words global appearance model with TF-IDF ranking is sufficiently distinctive for the test environments, its performance and discrimination needs to be evaluated in much larger environments, with larger node sets. In particular, the relation between vocabulary size and discrimination should be established. This chapter has merely proposed a unified framework and a proof-of-concept implementation; in

Figure 7.3: Left: video frames of loop closure or recovery events. Right: the most similar previous view of the scene. Normal observations are green, while observations via candidate edges are magenta

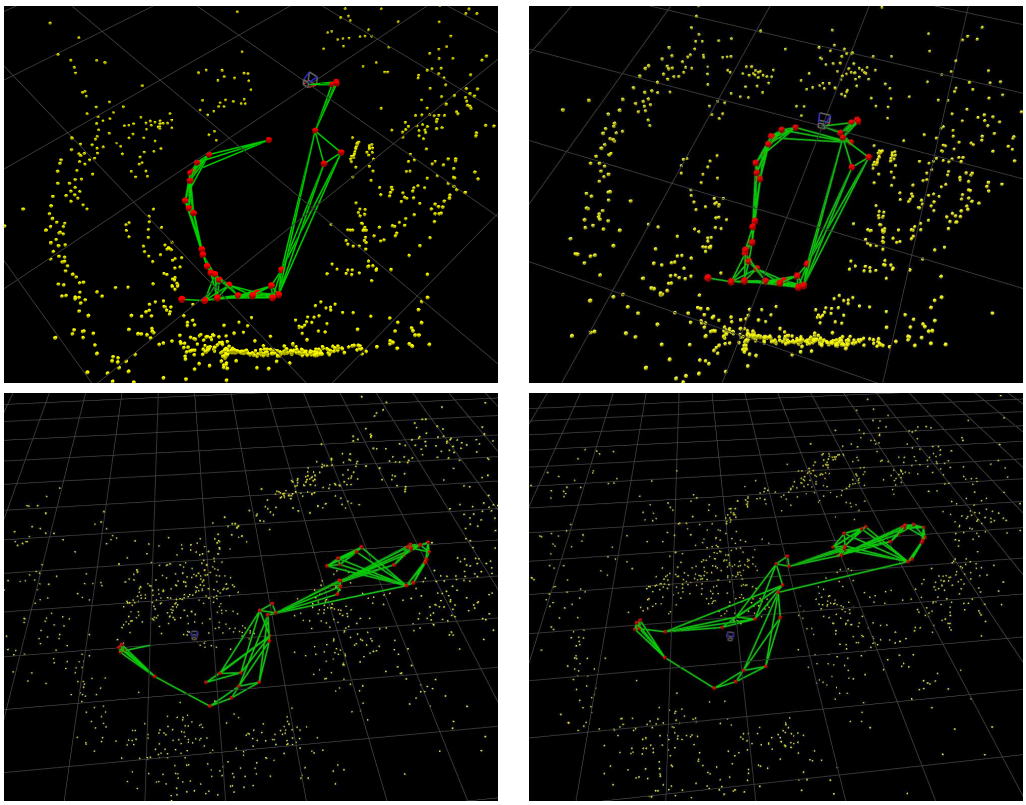

Figure 7.4: Before and after loop closure in two sequences: Landmarks are yellow, graph edges are green, nodes are red, and the camera is a small frustum. Each pair is from consecutive time steps (33 ms apart), before further incremental refinement by the graph optimiser

the larger scale, a more hierarchical or altogether different appearance model might be necessary.

8

Conclusion

8.1 Summary

This thesis has explored the manifold of frame-rate SLAM with a single camera, considering scaling efficiency, landmark types and representations, consistency, robustness, and loop closing. Two monocular SLAM systems have provided the context for these investigations, with one focusing on efficiency and the other on consistency.

The first system adapts the Rao-Blackwellised particle filtering framework of FastSLAM to a monocular SLAM setting, allowing the efficient mapping of hundreds to thousands of landmarks. The top-down search method that makes data association and image processing efficient is adapted to the particle filter framework. An inverse depth parameterisation is employed for the partial initialisation problem. The system delivers accurate localisation over small areas.

This system is extended to include edge landmarks, permitting a different geometric representation of the environment. The definition and representation of the new landmarks, called edgelets, are motivated by the locality properties of point landmarks. The edge appearance model makes image operations efficient, but data association becomes ambiguous. A random sampling method is used for robust data association in the presence of multiple hypotheses.

The second system, called Graph SLAM, is motivated by the consistency problem in SLAM. The graph formulation is designed to split the estimation problem into local and global components, so that local maps are consistently estimated and the nonlinear relations between them are globally optimised. Pose is estimated outside of the local map state, and the motion model is not used to update the local maps. Pose and landmark estimates are propagated through the graph from one coordinate frame to another. The estimation is qualitatively more consistent than FastSLAM or EKF SLAM, and efficiently maps more than a thousand landmarks.

Graph SLAM is then extended with a unified framework for loop closing and recovery. The framework comprises a hierarchical appearance model, using a bag-of-words representation on the global scale and a dictionary of landmark exemplars on the per-node scale. Both models are built on invariant descriptors similar to SIFT. Appearance and structure are matched between video images and local maps, and confirmed over multiple time steps. Creation of new graph edges by this method addresses both loop closing and recovery. The resulting extended system successfully recovers from multiple tracking failures, stitching together graphs created at different times, and closing loops when active search is insufficient. Though presented within the Graph SLAM system, the framework can be applied in any submapping SLAM scenario.

8.2 Contributions

The body of this thesis makes the following contributions:

- Real time monocular SLAM with thousands of landmarks using FastSLAM. The application and adaptation of the FastSLAM filter increases by an order of magnitude the number of landmarks feasibly mapped at frame-rate.
- An inverse depth parameterisation for landmarks. The parameterisation makes the observation model nearly linear under small camera displacements, allowing standard linear-Gaussian filters to be used for landmark initialisation.
- Edge landmarks suitable for monocular SLAM. The landmarks have the convenient locality properties of point landmarks, while modelling different image and world features. The resulting maps give a pleasing wireframe representation of edges in the environment, including slowly curving edges.
- An efficient selection method for edgelet landmarks. The method requires minimal computation and finds short edge segments unlikely to be confused with others during tracking.
- The Graph SLAM system for consistent SLAM. By partitioning observations between local maps and estimating coordinate transformations using shared landmarks, consistent estimation is maintained where EKF SLAM and FastSLAM become inconsistent. A global optimisation over graph edges aids map convergence.
- A unified framework for loop closing and recovery. The framework uses global and local appearance models and structure matching to find correspondences between the video and parts of the map. Mapping continues after failure and the map is reconnected upon recovery. Loop closing succeeds even when the filter's localisation estimate is inaccurate.
- An instance of the loop closing and recovery framework in the context of Graph SLAM. Invariant feature descriptors and bag-of-words provide the appearance models, and structure matching proceeds by the three-point-pose algorithm and MLESAC. The system successfully maps difficult sequences with several tracking failures and loop closures.

8.3 Future Work

The investigation of monocular SLAM in this thesis highlights a number of avenues for further research:

- Particle filtering methods such as FastSLAM rapidly become inconsistent. Methods for maintaining or recreating particle diversity might extend the period of consistent operation. Alternatively, FastSLAM might be viewed as an efficient front-end, the output of which could be fed to a more statistically correct filter.
- Higher-order geometric elements in the world are difficult to track with a moving camera. Small pieces can be more reliably estimated, but the map representation is then fragmented. If low-order primitives could be agglomerated in the filter or at a higher stage, then a more coherent model could be built, also improving tracking.
- A better heuristic or more formal method for node creation would immediately improve the accuracy and reliability of Graph SLAM. Currently, creation of new nodes is the most vulnerable time for the system.
- The global graph optimisation algorithm of Graph SLAM is not statistically correct, because it assumes that edge estimates are independent. It also requires, per iteration, time linear in the number of graph edges and cycle edges. The graph optimisation might be replaced with a message-passing algorithm or a background bundle adjustment over all state variables.
- Monocular SLAM is fragile. The tracking assumption often fails, and no recovery mechanism is foolproof. A more thorough use of image information might make SLAM more robust under unpredictable motions. Further, data association should be reversible so that incorrect decisions can be undone.

Appendix A

Projection and Transformation Jacobians

A.1 Point Projection Jacobians

This section describes the derivatives of the perspective-projection observation model that takes 3D points to 2D points in the camera plane. These Jacobians are used for pose and landmark updates in Chapter 4.

A.1.1 Euclidean Points

The observation model $\mathbf{h}_C(\mathbf{x})$ first transforms a 3D point $\mathbf{x}$ in global Euclidean coordinates into the local frame of $C = (\mathbf{R}, \mathbf{t}) \in \mathrm{SE}(3)$, and then projects the result, $\mathbf{y}$, onto

the 2D camera plane by dividing by depth.

$$\begin{aligned}\begin{pmatrix} x \\ y \\ z \end{pmatrix} &= \mathbf{y} \equiv C \cdot \mathbf{x} \\ &= \mathbf{R}\mathbf{x} + \mathbf{t} \qquad \text{(A.1)} \\ \begin{pmatrix} u \\ v \end{pmatrix} &= \text{project}(\mathbf{y}) \\ &= \begin{pmatrix} \frac{x}{z} & \frac{y}{z} \end{pmatrix}^T \qquad \text{(A.2)} \\ \mathbf{h}_C(\mathbf{x}) &= \begin{pmatrix} u \\ v \end{pmatrix} \qquad \text{(A.3)}\end{aligned}$$

Applying the chain rule, the derivative by an arbitrary parameter α splits into two pieces:

$$\frac{\partial \mathbf{h}_C(\mathbf{x})}{\partial \alpha} = \frac{\partial \text{project}(\mathbf{y})}{\partial \mathbf{y}} \cdot \frac{\partial (C \cdot \mathbf{x})}{\partial \alpha} \qquad \text{(A.4)}$$

The projection derivative is a common first factor to both pose and landmark observation model derivatives:

$$\frac{\partial \text{project}(\mathbf{y})}{\partial \mathbf{y}} = \frac{1}{z} \cdot \begin{pmatrix} 1 & 0 & -\frac{x}{z} \\ 0 & 1 & -\frac{y}{z} \end{pmatrix}$$

The derivative of the global-to-local transformation by landmark coordinates is simple, as the mapping is affine:

$$\begin{aligned}\frac{\partial (C \cdot \mathbf{x})}{\partial \mathbf{x}} &= \frac{\partial \left(\mathbf{R}\mathbf{x} + \mathbf{t}\right)}{\partial \mathbf{x}} \\ &= \mathbf{R} \qquad \text{(A.5)}\end{aligned}$$

As in Section 3.2.3, the derivative of the transformation by the camera parameters is expressed with respect to left multiplication by an exponentiated element of the

tangent space, $\boldsymbol{\epsilon} \in \mathbb{R}^6$:

$$\frac{\partial(C \cdot \mathbf{x})}{\partial C} \approx \frac{\partial(\exp(\boldsymbol{\epsilon}) \cdot C \cdot \mathbf{x})}{\partial \boldsymbol{\epsilon}} \tag{A.6}$$

$$= \frac{\partial(\exp(\boldsymbol{\epsilon}) \cdot \mathbf{y})}{\partial \boldsymbol{\epsilon}} \tag{A.7}$$

$$= \left(\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & 1 & 0 & \text{-}[\mathbf{y}]_\times \\ 0 & 0 & 1 & \end{array}\right) \tag{A.8}$$

$$= \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{array}\right) \tag{A.9}$$

Recall that the first three tangent space coordinates correspond to translation parameters and the last three to rotation (see Section 3.2).

Combining the projection and transformation derivatives gives observation model Jacobians by landmark parameters ($\mathbf{J}_\mathbf{x}$) and by pose parameters ($\mathbf{J}_C$):

$$\mathbf{J}_\mathbf{x} \equiv \frac{\partial \mathbf{h}_C(\mathbf{x})}{\partial \mathbf{x}} \tag{A.10}$$

$$= \frac{1}{z} \cdot \begin{pmatrix} 1 & 0 & -\frac{x}{z} \\ 0 & 1 & -\frac{y}{z} \end{pmatrix} \cdot \mathbf{R} \tag{A.11}$$

$$\mathbf{J}_C \equiv \frac{\partial \mathbf{h}_C(\mathbf{x})}{\partial C} \tag{A.12}$$

$$= \frac{1}{z} \cdot \begin{pmatrix} 1 & 0 & -\frac{x}{z} \\ 0 & 1 & -\frac{y}{z} \end{pmatrix} \cdot \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{array}\right) \tag{A.13}$$

$$= \begin{pmatrix} \frac{1}{z} & 0 & -\frac{x}{z^2} & -\frac{xy}{z^2} & 1+\frac{x^2}{z^2} & -\frac{y}{z} \\ 0 & \frac{1}{z} & -\frac{y}{z^2} & -(1+\frac{y^2}{z^2}) & \frac{xy}{z^2} & \frac{x}{z} \end{pmatrix} \tag{A.14}$$

A.1.2 Inverse Depth Points

The observation model $\mathbf{h}_C^*$ for inverse depth point $\mathbf{x}^* = \begin{pmatrix} u' & v' & q \end{pmatrix}^T$ is most easily expressed by first converting to Euclidean coordinates:

$$\mathbf{x} = \frac{1}{q} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \tag{A.15}$$

Substituting this into (A.1) and (A.3) and simplifying gives the model for inverse depth points:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{y} \equiv \frac{1}{q}\mathbf{R} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} + \mathbf{t} \tag{A.16}$$

$$\begin{aligned} \mathbf{h}_C^*(\mathbf{x}^*) &= \text{project}(\mathbf{y}) \\ &= \text{project}(q\mathbf{y}) \\ &= \text{project}\left(\mathbf{R} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} + q\mathbf{t} \right) \end{aligned} \tag{A.17}$$

The pre-projection transformation is again linear in the landmark coordinates:

$$\begin{aligned} \frac{\partial (C \cdot \mathbf{x}^*)}{\partial \mathbf{x}^*} &= \frac{\partial \left(\mathbf{R} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} + q\mathbf{t} \right)}{\partial \mathbf{x}^*} \\ &= \begin{pmatrix} \mathbf{R}_{1,1} & \mathbf{R}_{1,2} & \mathbf{t}_1 \\ \mathbf{R}_{2,1} & \mathbf{R}_{2,2} & \mathbf{t}_2 \\ \mathbf{R}_{3,1} & \mathbf{R}_{3,2} & \mathbf{t}_3 \end{pmatrix} \end{aligned} \tag{A.18}$$

Then computation of Jacobians by point and pose parameters is then straightforward. Starting from (A.11),

$$\mathbf{J}_{\mathbf{x}^*}^* = \frac{1}{qz} \cdot \begin{pmatrix} 1 & 0 & -\frac{x}{z} \\ 0 & 1 & -\frac{y}{z} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{R}_{1,1} & \mathbf{R}_{1,2} & \mathbf{t}_1 \\ \mathbf{R}_{2,1} & \mathbf{R}_{2,2} & \mathbf{t}_2 \\ \mathbf{R}_{3,1} & \mathbf{R}_{3,2} & \mathbf{t}_3 \end{pmatrix} \tag{A.19}$$

$$\tag{A.20}$$

The pose Jacobian is unchanged from (A.14):

$$\mathbf{J}^*_C = \frac{\partial \mathbf{h}_C(\mathbf{x})}{\partial C} = \mathbf{J}_C \tag{A.21}$$

A.2 Edgelet Jacobians

This section describes the observation model for the edgelets described in Chapter 5. There are two representations of edgelets; Euclidean coordinates and inverse depth.

A.2.1 Euclidean Edgelets

The edgelet representation is a 6-vector $\begin{pmatrix}\mathbf{x}^T & \mathbf{d}^T\end{pmatrix}^T$, where the edgelet centre is $\mathbf{x} \in \mathbb{R}^3$ and the edgelet direction is $\mathbf{d} \in \mathbb{R}^3$. The observation model first transforms the edgelet into the camera frame of pose C, and then projects the result to a centre point $\mathbf{x}_p$ and a direction $\mathbf{d}_p$ in the camera plane:

$$\mathbf{X} \equiv C \cdot \mathbf{x} = \mathbf{R}\mathbf{x} + \mathbf{t} \tag{A.22}$$

$$\mathbf{D} \equiv \mathbf{R}\mathbf{d} \tag{A.23}$$

$$\mathbf{x}_p = \mathrm{project}(\mathbf{X}) \tag{A.24}$$

$$\mathbf{d}_p = \begin{pmatrix}\mathbf{X}_3\mathbf{D}_1 - \mathbf{X}_1\mathbf{D}_3 \\ \mathbf{X}_3\mathbf{D}_2 - \mathbf{X}_2\mathbf{D}_3\end{pmatrix} \tag{A.25}$$

Note that rigid transformations change the direction vector $\mathbf{d}$ purely by rotation. The Jacobians by edgelet parameters are computed using the chain rule and the point

derivatives of Section A.1.1:

$$\frac{\partial \mathbf{x}_p}{\partial \mathbf{X}} = \frac{\partial \mathrm{project}(\mathbf{X})}{\partial \mathbf{X}} \tag{A.26}$$

$$\frac{\partial \mathbf{x}_p}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{x}_p}{\partial \mathbf{X}}\right)\mathbf{R} \tag{A.27}$$

$$\frac{\partial \mathbf{d}_p}{\partial \mathbf{X}} = \begin{pmatrix} -\mathbf{D}_3 & 0 & \mathbf{D}_1 \\ 0 & -\mathbf{D}_3 & \mathbf{D}_2 \end{pmatrix} \tag{A.28}$$

$$\frac{\partial \mathbf{d}_p}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{d}_p}{\partial \mathbf{X}}\right)\mathbf{R} \tag{A.29}$$

$$\frac{\partial \mathbf{d}_p}{\partial \mathbf{D}} = \begin{pmatrix} \mathbf{X}_3 & 0 & -\mathbf{X}_1 \\ 0 & \mathbf{X}_3 & -\mathbf{X}_2 \end{pmatrix} \tag{A.30}$$

$$\frac{\partial \mathbf{d}_p}{\partial \mathbf{d}} = \left(\frac{\partial \mathbf{d}_p}{\partial \mathbf{D}}\right)\mathbf{R} \tag{A.31}$$

$$\tag{A.32}$$

The derivatives by pose parameters, again assuming left multiplication of C by $\exp \boldsymbol{\epsilon}$, are similarly related to the pose Jacobians for Euclidean points:

$$\frac{\partial \mathbf{x}_p}{\partial \boldsymbol{\epsilon}} = \left(\frac{\partial \mathbf{x}_p}{\partial \mathbf{X}}\right) \cdot \left(\mathbf{I}_3 \,\middle|\, -[\mathbf{X}]_\times \right) \tag{A.33}$$

$$\frac{\partial \mathbf{d}_p}{\partial \boldsymbol{\epsilon}} = \left(\frac{\partial \mathbf{d}_p}{\partial \mathbf{X}}\right) \cdot \left(\mathbf{I}_3 \,\middle|\, -[\mathbf{X}]_\times \right) + \left(\frac{\partial \mathbf{d}_p}{\partial \mathbf{D}}\right) \cdot \left(\mathbf{0} \,\middle|\, -[\mathbf{D}]_\times \right) \tag{A.34}$$

A.2.2 Inverse Depth Edgelets

The partially initialised edgelet representation is a 6-vector $\begin{pmatrix} \mathbf{x}^{*T} & \mathbf{d}^{*T} \end{pmatrix}^T$, along with a fixed pose $C_0 = (\mathbf{R}_0, \mathbf{t}_0) \in \mathrm{SE}(3)$ corresponding to the initial view of the landmark. The vectors components are individually named:

$$\mathbf{x}^* \equiv \begin{pmatrix} u & v & q \end{pmatrix}^T \tag{A.35}$$

$$\mathbf{d}^* \equiv \begin{pmatrix} d_u & d_v & d_q \end{pmatrix}^T \tag{A.36}$$

Components (u, v) and (d_u, d_v) correspond to the location and direction, respectively, of the edgelet in the camera plane of C_0. Component q describes the inverse depth

of the edgelet in C_0, and d_q is its differential moving along the edge. For a pose $C = (\mathbf{R}, \mathbf{t})$, the transformation from the initial frame to the new frame is the result of first transforming to world coordinates and then into the new pose. This yields a displacement transformation:

$$\mathbf{R}' \equiv \mathbf{R} \cdot \mathbf{R}_0^T \tag{A.37}$$

$$\mathbf{t}' \equiv \mathbf{t} - \mathbf{R}'\mathbf{t}_0 \tag{A.38}$$

The observation model again maps the state to a centre point and direction in the camera plane:

$$\mathbf{P} \equiv \mathbf{R}' \begin{pmatrix} u & v & 1 \end{pmatrix}^T + q\mathbf{t}' \tag{A.39}$$

$$\mathbf{V} \equiv \mathbf{R}' \begin{pmatrix} d_u & d_v & 0 \end{pmatrix}^T + d_q\mathbf{t}' \tag{A.40}$$

$$\mathbf{x}_p = \text{project}(\mathbf{P}) \tag{A.41}$$

$$\mathbf{d}_p = \begin{pmatrix} \mathbf{P}_3\mathbf{V}_1 - \mathbf{P}_1\mathbf{V}_3 \\ \mathbf{P}_3\mathbf{V}_2 - \mathbf{P}_2\mathbf{V}_3 \end{pmatrix} \tag{A.42}$$

The Jacobians by the inverse depth edgelet parameters then follow a similar form as for inverse depth points:

$$\frac{\partial \mathbf{x}_p}{\partial \mathbf{P}} = \frac{\partial \text{project}(\mathbf{P})}{\partial \mathbf{P}} \tag{A.43}$$

$$\tag{A.44}$$

$$\frac{\partial \mathbf{x}_p}{\partial \mathbf{x}^*} = \left(\frac{\partial \mathbf{x}_p}{\partial \mathbf{P}}\right) \cdot \begin{pmatrix} \mathbf{R}'_{1,1} & \mathbf{R}'_{1,2} & \mathbf{t}'_1 \\ \mathbf{R}'_{2,1} & \mathbf{R}'_{2,2} & \mathbf{t}'_2 \\ \mathbf{R}'_{3,1} & \mathbf{R}'_{3,2} & \mathbf{t}'_3 \end{pmatrix} \tag{A.45}$$

$$\frac{\partial \mathbf{d}_p}{\partial \mathbf{P}} = \begin{pmatrix} -\mathbf{V}_3 & 0 & \mathbf{V}_1 \\ 0 & -\mathbf{V}_3 & \mathbf{V}_2 \end{pmatrix} \tag{A.46}$$

$$\frac{\partial \mathbf{d}_p}{\partial \mathbf{x}^*} = \left(\frac{\partial \mathbf{d}_p}{\partial \mathbf{P}}\right) \cdot \begin{pmatrix} \mathbf{R}'_{1,1} & \mathbf{R}'_{1,2} & \mathbf{t}'_1 \\ \mathbf{R}'_{2,1} & \mathbf{R}'_{2,2} & \mathbf{t}'_2 \\ \mathbf{R}'_{3,1} & \mathbf{R}'_{3,2} & \mathbf{t}'_3 \end{pmatrix} \tag{A.47}$$

$$\frac{\partial \mathbf{d}_p}{\partial \mathbf{V}} = \begin{pmatrix} \mathbf{P}_3 & 0 & -\mathbf{P}_1 \\ 0 & \mathbf{P}_3 & -\mathbf{P}_2 \end{pmatrix} \tag{A.48}$$

$$\frac{\partial \mathbf{d}_p}{\partial \mathbf{d}^*} = \left(\frac{\partial \mathbf{d}_p}{\partial \mathbf{D}}\right) \cdot \begin{pmatrix} \mathbf{R}'_{1,1} & \mathbf{R}'_{1,2} & \mathbf{t}'_1 \\ \mathbf{R}'_{2,1} & \mathbf{R}'_{2,2} & \mathbf{t}'_2 \\ \mathbf{R}'_{3,1} & \mathbf{R}'_{3,2} & \mathbf{t}'_3 \end{pmatrix} \tag{A.49}$$

$$\tag{A.50}$$

The Jacobians by pose tangent space parameters $\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = \begin{pmatrix}\boldsymbol{\epsilon}_t^T & \boldsymbol{\epsilon}_\omega^T\end{pmatrix}^T$:

$$\frac{\partial \mathbf{x}_p}{\partial \boldsymbol{\epsilon}} = \left(\frac{\partial \mathbf{x}_p}{\partial \mathbf{P}}\right) \cdot \left(\begin{array}{c|c} q\mathbf{I}_3 & -[\mathbf{P}]_\times \end{array}\right) \tag{A.51}$$

$$\frac{\partial \mathbf{d}_p}{\partial \boldsymbol{\epsilon}_t} = q\frac{\partial \mathbf{d}_p}{\partial \mathbf{P}} + d_q\frac{\partial \mathbf{d}_p}{\partial \mathbf{V}} \tag{A.52}$$

$$\frac{\partial \mathbf{d}_p}{\partial \boldsymbol{\epsilon}_\omega} = \begin{pmatrix}(\mathbf{V}_1\mathbf{P}_2 - \mathbf{V}_2\mathbf{P}_1) & 0 & (\mathbf{P}_2\mathbf{V}_3 - \mathbf{P}_3\mathbf{V}_2) \\ 0 & (\mathbf{V}_1\mathbf{P}_2 - \mathbf{V}_2\mathbf{P}_1) & (\mathbf{P}_3\mathbf{V}_1 - \mathbf{P}_1\mathbf{V}_3)\end{pmatrix} \tag{A.53}$$

$$\frac{\partial \mathbf{d}_p}{\partial \boldsymbol{\epsilon}} = \left(\begin{array}{c|c} \frac{\partial \mathbf{d}_p}{\partial \boldsymbol{\epsilon}_t} & \frac{\partial \mathbf{d}_p}{\partial \boldsymbol{\epsilon}_\omega} \end{array}\right) \tag{A.54}$$

A.2.3 Intercept-Slope Form

Observations of edgelets are given in intercept-slope form relative to a fixed two-dimensional frame, determined by an origin $\mathbf{x}_0$ and a unit normal direction $\hat{\mathbf{n}}$ in the camera plane (see Section 5.3.1). The vector $\hat{\mathbf{n}}$ acts as the y-axis in the intercept-slope equation $y = b + mx$. The vectors $\mathbf{x}_0$ and $\hat{\mathbf{n}}$ are computed in the landmark prediction phase from one particle's estimate of the edgelet. The two components of the observation are then the parameters b and m, relative to $\mathbf{x}_0$ and $\hat{\mathbf{n}}$. The observation model that takes $\mathbf{x}_p$ and $\mathbf{d}_p$ to intercept-slope form, and the corresponding Jacobians, are as follows:

$$\hat{\mathbf{h}} \equiv \begin{pmatrix}\hat{\mathbf{n}}_1 & -\mathbf{n}_0\end{pmatrix}^T \tag{A.55}$$

$$m = \frac{\mathbf{d}_p^T\hat{\mathbf{n}}}{\mathbf{d}_p^T\hat{\mathbf{h}}} \tag{A.56}$$

$$b = (\mathbf{x}_p - \mathbf{x}_0)^T\hat{\mathbf{n}} - m \cdot (\mathbf{x}_p - \mathbf{x}_0)^T\hat{\mathbf{h}} \tag{A.57}$$

$$\frac{\partial b}{\partial \mathbf{x}_p} = \hat{\mathbf{n}} - m \cdot \hat{\mathbf{h}} \tag{A.58}$$

$$\frac{\partial d}{\partial \mathbf{d}_p} = \frac{\hat{\mathbf{n}} - m \cdot \hat{\mathbf{h}}}{\mathbf{d}_p^T\hat{\mathbf{h}}} \tag{A.59}$$

$$\frac{\partial b}{\partial \mathbf{d}_p} = (\mathbf{x}_0 - \mathbf{x}_p)^T\hat{\mathbf{h}}\frac{\partial d}{\partial \mathbf{d}_p} \tag{A.60}$$

Bibliography

Azarbayejani, A. & Pentland, A.P. (1995). Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **17**, pp 562–575. 2.2.2, 4.1.2

Bailey, T. (2002). *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. Ph.D. thesis, University of Sydney, Australia. 2.1.2, 2.3.3, 6.1.3, 6.5.3, 6.6.4

Bailey, T., Nieto, J., Guivant, J., Stevens, M. & Nebot, E. (2006a). Consistency of the ekf-slam algorithm. In *Proc. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*, pp 3562–3568, Beijing, China. 2.3.2, 2.3.5, 4.6.6, 6.1.2, 6.6.1

Bailey, T., Nieto, J. & Nebot, E. (2006b). Consistency of the fastslam algorithm. In *Proc. 2006 IEEE International Conference on Robotics and Automation (ICRA'06)*, pp 424–429, Orlando, FL, USA. 6.1.2

Bay, H., Ess, A., Tuytelaars, T. & Gool, L.V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, **110**, pp 346–359. 7.4

Betge-Brezetz, S., Hebert, P., Chatila, R. & Devy, M. (1996). Uncertain map making in natural environments. In *Proc. 1996 IEEE International Conference on Robotics and Automation (ICRA'96)*, vol. 2, pp 1048–1053, Minneapolis, MN, USA. 2.1.2

Bosse, M., Newman, P., Leonard, J. & Teller, S. (2004). Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *The International Journal of Robotics Research*, **23**, pp 1113–1139. 2.3.3, 6.1.3

Bouguet, J.Y. & Perona, P. (1995). Visual navigation using a single camera. In *Proc. 5th IEEE International Conference on Computer Vision (ICCV'95)*, pp 645–652, Cambridge, MA, USA. 2.2.2

Broida, T.J., Chandraskekhar, S. & Chellappa, R. (1990). Recursive 3-d motion estimation from a monocular image sequence. *IEEE Transactions on Aerospace and Electronic S*, **26**, pp 639–656. 2.2.2

Brown, M., Szeliski, R. & Winder, S. (2005). Multi-image matching using multi-scale oriented patches. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR '05)*, pp 510–517, IEEE Computer Society, San Diego, CA, USA. 7.4

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **8**, pp 679–698. 5.1.2, 5.4

Castellanos, J., Tardos, J. & Schmidt, G. (1997). Building a global map of the environment of a mobile robot: the importance of correlations. In *Proc. 1997 IEEE International Conference on Robotics and Automation (ICRA'97)*, vol. 2, pp 1053–1059, Albuquerque, NM, USA. 2.1.2

Castellanos, J., Neira, J. & Tardos, J. (2004). Limits to the consistency of ekf-based slam. In *IFAC Symposium on Intelligent Autonomous Vehicles (IAV 2004)*, Lisbon, Portugal. 2.3.2

Castellanos, J., Martinez-Cantin, R., Tardos, J. & Neira, J. (2007). Robocentric map joining: Improving the consistency of ekf-slam. *Robotics and Autonomous Systems*, **55**, pp 21–29. 2.3.2, 6.1.2, 6.1.3

Chekhlov, D., Pupilli, M., Mayol-Cuevas, W. & Calway, A. (2007). Robust real-time visual slam using scale prediction and exemplar based feature description. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pp 1–7, IEEE Computer Society, Minneapolis, MN, USA. 7.2.1

Chiuso, Favaro, Jin & Soatto (2002). Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **24**, pp 523–535. 2.2.2

Chong, K. & Kleeman, L. (1999). Feature-based mapping in real, large scale environments using an ultrasonic array. *International Journal of Robotics Research*, **18**, pp 3–19. 2.3.3

Clemente, L.A., Davison, A.J., Reid, I., Neira, J. & Tards, J.D. (2007). Mapping large loops with a single hand-held camera. In *Proc. Robotics Science and Systems III*. 7.2.2

Csorba, M. (1997). *Simultaneous Localisation and Map Building*. Ph.D. thesis, University of Oxford, UK. 2.1.2, 2.3.1.2

Cummins, M. & Newman, P. (2007). Probabilistic appearance based navigation and loop closing. In *Proc. 2007 IEEE International Conference on Robotics and Automation (ICRA'07)*, Rome, Italy. 7.2.2, 7.8.1

Cummins, M. & Newman, P. (2008). Accelerated appearance-only SLAM. In *Proc. 2008 IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena,California. 7.8.1

Davison, A., Reid, I., Molton, N. & Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, **29**, pp 1052–1067. 4.5

Davison, A.J. (1998). *Mobile Robot Navigation Using Active Vision*. Ph.D. thesis, University of Oxford. 2.2.3, 2.3.1.3

Davison, A.J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proc. 9th IEEE International Conference on Computer Vision (ICCV'03)*, pp 1403–1410, Nice, France. 2.2.3, 2.3.1.1, 4.1, 4.1.2, 4.4.1, 4.4.3, 4.5, 4.5.2.1, 4.5.2.3, 7.1, 7.2.1

Davison, A.J. (2005). Active search for real time vision. In *Proc. 10th IEEE International Conference on Computer Vision (ICCV'05)*, pp 66–73, Beijing, China. 2.2.3

Davison, A.J. & Kita, N. (2001). Simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In *Proc. 8th IEEE International Conference on Computer Vision (ICCV'01)*, pp 384–391, Vancouver, BC, Canada. 2.2.3

Davison, A.J. & Murray, D.W. (1998). Mobile robot localization using active vision. In *Proc. 5th European Conference on Computer Vision (ECCV'98)*, pp 809–825, Freiburg, Germany. 2.1.2, 2.2.3

Davison, A.J. & Murray, D.W. (2002). Simultaneous localisation and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **24**, pp 865–880. 2.2.3, 2.3.1.1

Dellaert, F., Burgard, W., Fox, D. & Thrun, S. (1999). Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR '99)*, pp 594–600, Colorado, USA. 2.3.5

Dissanayake, G., Durrant-Whyte, H. & Bailey, T. (2000). A computationally efficient solution to the simultaneouslocalisation and map building (SLAM) problem. In *Proc. 2000 IEEE International Conference on Robotics and Automation (ICRA'00)*, vol. 2, pp 1009–1014, San Francisco, CA, USA. 2.3.1.1

Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H. & Csorba, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, **17**, pp 229–241. 2.1.2

Drummond, T. & Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **24**, pp 932 – 946. 5.1.2

Dudek, G. & Jegessur, D. (2002). Robust place recognition using local appearance based methods. In *Proc. 2002 IEEE International Conference on Robotics and Automation (ICRA'02)*, pp 466–474, Washington, DC, USA. 7.2.2

Durrant-Whyte, H.F. (1988). Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, **4**, pp 23–31. 2.1.1

Eade, E. & Drummond, T. (2006a). Edge landmarks in monocular slam. In *Proc. British Machine Vision Conference (BMVC'06)*, pp 1–8, BMVA, Edinburgh. 1.8, 5.1.2

Eade, E. & Drummond, T. (2006b). Scalable monocular slam. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp 469–476, IEEE Computer Society, New York, NY, USA. 1.8, 4.5

Eade, E. & Drummond, T. (2007). Monocular slam as a graph of coalesced observations. In *Proc. 11th IEEE International Conference on Computer Vision (ICCV'07)*, Rio de Janeiro, Brazil. 1.8

Eade, E. & Drummond, T. (2008a). Edge landmarks in monocular slam. *Image and Vision Computing*, **Special Issue for BMVC'06**, in Press, Corrected Proof. 1.8

Eade, E. & Drummond, T. (2008b). Unified loop closing and recovery for real time monocular slam. In *Proc. British Machine Vision Conference (BMVC'08)*, pp 53–62, BMVA, Leeds. 1.8

Elinas, P., Sim, R. & Little, J.J. (2006). σslam: Stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *Proc. 2006 IEEE International Conference on Robotics and Automation (ICRA'06)*, pp 1564–1570, Orlando, FL, USA. 4.1.2

Estrada, C., Neira, J. & Tardos, J. (2005). Hierarchical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, **21**, pp 588–596. 2.3.3

Eustice, R., Walter, M. & Leonard, J. (2005). Sparse extended information filters: Insights into sparsification. In *Proc. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pp 3281–3288, Edmonton, Alberta, Canada. 2.3.4

Eustice, R., Singh, H. & Leonard, J. (2006a). Exactly sparse delayed-state filters for view-based slam. *IEEE Transactions on Robotics*, **22**, pp 1100–1114. 2.3.4

Eustice, R.M., Singh, H., Leonard, J.J. & Walter, M.R. (2006b). Visually mapping the rms titanic: Conservative covariance estimates for slam information filters. *The International Journal of Robotics Research*, **25**, pp 1223–1242. 2.3.4

Faugeras, O. (1995). Stratification of 3-d vision: projective, affine, and metric representations. *Journal of the Optical Society of America A*, **12**, pp 465–484. 2.2.1

Fischler, M. & Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communcations of the ACM*, **24**, pp 381–395. 2.2.2, 5.6, 7.7.3

Fitzgibbon, A.W. & Zisserman, A. (1998). Automatic camera recovery for closed or open image sequences. In *Proc. 5th European Conference on Computer Vision (ECCV'98)*, pp 311–326, Freiburg, Germany. 2.2.1, 6.1.3

Folkesson, J., Jensfelt, P. & Christensen, H. (2005). Vision slam in the measurement subspace. In *Proc. 2005 IEEE International Conference on Robotics and Automation (ICRA'05)*, pp 30–35, Barcelona, Spain. 5.1.2

Gallier, J. (2001). *Geometric Methods and Applications for Computer Science and Engineering*. No. 38 in Texts in Applied Mathematics, Springer-Verlag. 3.2.2, 3.2.2

Gee, A.P. & Mayol-Cuevas, W. (2006). Real-time model-based slam using line segments. In *2nd International Symposium on Visual Computing*. 5.1.2

Guivant, J. (2002). *Efficient Simultaneous Localization and Mapping in Large Environments*. Ph.D. thesis, University of Sydney, Australia. 2.1.2, 2.3.1.3

Guivant, J. & Nebot, E. (2001). Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, **17**, pp 242–257. 2.3.1.3

Guivant, J. & Nebot, E. (2002). Improving computational and memory requirements of simultaneouslocalization and map building algorithms. In *Proc. 2002 IEEE International Conference on Robotics and Automation (ICRA'02)*, vol. 3, pp 2731–2736, Washington, DC, USA. 2.3.4

Harris, C. (1992). Tracking with rigid models. In A. Blake, ed., *Active Vision*, chap. 4, pp 59–73, MIT Press. 5.1.2

Harris, C. & Stephen, M. (1988). A combined corner and edge detection. In M. Matthews, ed., *Proc. 4th ALVEY Vision Conference*, pp 147–151, University of Manchester, England. 2.2.2, 4.5.2.1, 6.3.6

Harris, C.G. & Pike, J.M. (1988). 3d positional integration from image sequences. *Image and Vision Computing*, **6**, pp 87–90. 2.2.3, 4.1.2

Hartley, R.I. & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2nd edn. 2.2.1

Heath, M., Sarkar, S., Sanocki, T. & Bowyer, K. (1996). Comparison of edge detectors: A methodology and initial study. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp 143–148, San Francisco, CA, USA. 5.1.2

Isard, M. & Blake, A. (1998). CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, **29**, pp 5–28. 2.3.5

Jin, H., Favaro, P. & Soatto, S. (2003). A semi-direct approach to structure from motion. *The Visual Computer*, **19**, pp 377–394. 2.2.2

Julier, S. & Uhlmann, J. (2001). A counter example to the theory of simultaneous localization and map building. In *Proc. 2001 IEEE International Conference on Robotics and Automation (ICRA'01)*, vol. 4, pp 4238–4243, Seoul, Korea. 2.3.2

Julier, S.J. & Uhlmann, J.K. (1997a). New extension of the kalman filter to nonlinear systems. *Signal Processing, Sensor Fusion, and Target Recognition VI*, **3068**, pp 182–193. 4.3.3, 4.5.2.4, 5.5, 6.1.2

Julier, S.J. & Uhlmann, J.K. (1997b). A non-divergent estimation algorithm in the presence of unknown correlations. In *Proc. American Control Conference*, pp 2369–2373, IEEE. 2.3.2

Julier, S.J. & Uhlmann, J.K. (2007). Using covariance intersection for slam. *Robots and Autonomous Systems*, **55**, pp 3–20. 2.3.2

Jung, I.K. & Lacroix, S. (2003). High resolution terrain mapping using low altitude aerial stereo imagery. In *Proc. 9th IEEE International Conference on Computer Vision (ICCV'03)*, pp 946–951, Nice, France. 2.2.3

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, **82**, pp 35–45. 2.1.1, 3.3

Ke, Y. & Sukthankar, R. (2004). Pca-sift: A more distinctive representation for local image descriptors. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR'04)*, vol. 2, pp 506–513, Washington, DC, USA. 7.4

Klein, G. & Drummond, T. (2002). Tightly integrated sensor fusion for robust visual tracking. In *Proc. British Machine Vision Conference (BMVC'02)*, vol. 2, pp 787 –796, BMVA, Cardiff. 6.2.2

Klein, G. & Drummond, T. (2005). A single-frame visual gyroscope. In *Proc. British Machine Vision Conference (BMVC'05)*, vol. 2, pp 529–538, BMVA, Oxford. 6.2.2

Klein, G. & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM Int'l Symp. Mixed and Augmented Reality*. 6.1.3

Knight, J., Davison, A. & Reid, I. (2001). Toward constant time slam using postponement. In *Proc. 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, vol. 1, pp 405–413, Maui, HI, USA. 2.3.1.3

Konolige, K. & Agrawal, M. (2007). Frame-frame matching for realtime consistent visual mapping. In *Proc. 2007 IEEE International Conference on Robotics and Automation (ICRA'07)*, pp 2803–2810, Rome, Italy. 6.1.3

Kwok, N. & Dissanayake, G. (2003). Bearing-only slam in indoor environments using a modified particle filter. In *Proc. 2003 Australasian Conference on Robotics and Automation*, Brisbane, Australia. 2.3.5, 4.1.2, 5.1.2

Lemaire, T. & Lacroix, S. (2007). Monocular-vision based slam using line segments. In *Proc. 2007 IEEE International Conference on Robotics and Automation (ICRA'07)*, pp 2791–2796, Rome, Italy. 5.1.2

Lemaire, T., Lacroix, S. & Sola, J. (2005). A practical 3d bearing-only slam algorithm. In *Proc. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pp 2449–2454, Edmonton, Alberta, Canada. 4.1, 4.1.2, 4.5, 4.5.2.1, 5.1.2

Leonard, J. & Feder, H. (2001). Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering*, **26**, pp 561–571. 2.3.3

Leonard, J.J. & Newman, P.M. (2003). Consistent, convergent, and constant-time slam. In G. Gottlob & T. Walsh, eds., *Proc. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, pp 1143–1150, Morgan Kaufmann, Acapulco, Mexico. 2.3.3, 6.1.3

Lepetit, V., Lagger, P. & Fua, P. (2005). Randomized trees for real-time keypoint recognition. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR '05)*, IEEE Computer Society, San Diego, CA, USA. 7.2.1

Lisien, B., Morales, D., Silver, D., Kantor, G., Rekleitis, I. & Choset, H. (2003). Hierarchical simultaneous localization and mapping. In *Proc. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, vol. 1, pp 448–453, Las Vegas, NV, USA. 6.1.3

Liu, T., Moore, A.W., Gray, E. & Yang, K. (2004). An investigation of practical approximate nearest neighbor algorithms. In *Advances in Neural Information Processing Systems (NIPS 2004)*, pp 825–832, MIT Press. 7.5.2

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**, pp 91–100. 2.2.3, 4.1.2, 4.5.2.1, 7.2.1, 7.4, 7.7.2

Matas, J., Chum, O., Urban, M. & Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conference (BMVC'02)*, pp 384–393, BMVA, Cardiff. 7.4

Maybeck, P. (1979). *Stochastic models, estimation and control*, vol. 1, chap. 1. Academic Press. 2.1.1

McLauchlan, P.F. & Murray, D.W. (1995). A unifying framework for structure and motion recovery from image sequences. In *Proc. 5th IEEE International Conference on Computer Vision (ICCV'95)*, pp 314–320, Cambridge, MA, USA. 2.2.1, 2.2.2

Mikolajczyk, K. & Schmid, C. (2004). Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, **60**, pp 63–86. 7.4

Mikolajczyk, K. & Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **27**, pp 1615–1630. 7.4

Molton, N.D., Davison, A.J. & Reid, I.D. (2004). Locally planar patch features for real-time structure from motion. In *Proc. British Machine Vision Conference (BMVC'04)*, BMVA, London, UK. 2.2.3

Montemerlo, M. & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using fastslam. In *Proc. 2003 IEEE International Conference on Robotics and Automation (ICRA'03)*, vol. 2, pp 1985–1991, Taipei, Taiwan. 2.3.5

Montemerlo, M., Thrun, S., Koller, D. & Wegbreit, B. (2002). Fastslam: a factored solution to the simultaneous localization and mapping problem. In *Proc. Eighteenth National Conference on Artifical Intelligence*, pp 593–598, Edmonton, Alberta, Canada. 2.3.5, 4.1, 4.2, 4.2.2, 4.2.2.1

Montemerlo, M., Thrun, S., Koller, D. & Wegbreit, B. (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In G. Gottlob & T. Walsh, eds., *Proc. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, Morgan Kaufmann, Acapulco, Mexico. 2.3.5, 4.1, 4.2, 4.2.2, 4.2.2.2

Montiel, J., Civera, J. & Davison, A. (2006). Unified inverse depth parametrization for monocular slam. In *Proc. Robotics: Science and Systems*, Philadelphia, USA. 4.5, 6.2.1

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F. & Sayd, P. (2006a). 3d reconstruction of complex structures with bundle adjustment: an incremental approach. In *ICRA 2006*, pp 3055–3061, IEEE Computer Society, Orlando, USA. 6.1.3

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F. & Sayd, P. (2006b). Monocular vision based slam for mobile robots. In *Proc. 18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, pp 1027–1031, Hong Kong, China. 6.1.3

Moutarlier, P. & Chatila, R. (1990). Stochastic multisensory data fusion for mobile robot location and environmental modelling. In H. Miura, ed., *Fifth International Symposium of Robotics Research*, pp 85–94, MIT Press, Cambridge, MA, USA. 2.1.1, 2.1.2, 2.2.3

Neira, J., Ribiero, M. & Tardos, J. (1997). Mobile robot localization and map building using monocular vision. In *Proc. 5th International Symposium on Intelligent Robotic Systems (SIRS'97)*, Stockholm, Sweden. 2.2.3, 5.1.2

Newman, P., Cole, D. & Ho, K. (2006). Outdoor slam using visual appearance and laser ranging. In *Proc. 2006 IEEE International Conference on Robotics and Automation (ICRA'06)*, pp 1180–1187, Orlando, FL, USA. 7.2.2

Newman, P.M. (1999). *On the structure and solution of the simultaneous localization and mapping problem*. Ph.D. thesis, University of Sydney, Australia. 2.1.2, 2.3.1.2

Newman, P.M. & Durrant-Whyte, H. (2001). Geometric projection filter: an efficient solution to the slam problem. In *Proc. International Society for Optical Engineering*, vol. 4571, Boston, MA, USA. 2.3.1.2

Nistér, D. (2003). Preemptive ransac for live structure and motion estimation. In *Proc. 9th IEEE International Conference on Computer Vision (ICCV'03)*, pp 199–206, Nice, France. 2.2.2

Nistér, D. & Stewénius, H. (2006). Scalable recognition with a vocabulary tree. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp 2161–2168, IEEE Computer Society, New York, NY, USA. 7.2.2

Nistér, D., Naroditsky, O. & Bergen, J. (2004). Visual odometry. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR'04)*, pp 652–659, Washington, DC, USA. 2.2.2, 6.2.2

Paskin, M. (2003). Thin junction tree filters for simultaneous localization and mapping. In G. Gottlob & T. Walsh, eds., *Proc. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, Morgan Kaufmann, Acapulco, Mexico. 2.3.4

Pollefeys, M., Koch, R. & van Gool, L.J. (1998). Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. 6th IEEE International Conference on Computer Vision (ICCV'98)*, pp 90–95, Bombay, India. 2.2.1

Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (1992). *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA. 6.3.5

Pupilli, M. & Calway, A. (2005). Real-time camera tracking using a particle filter. In *Proc. British Machine Vision Conference (BMVC'05)*, pp 519–528, BMVA, Oxford. 4.1.2, 7.2.1

Rahimi, A., Morency, L.P. & Darrell, T. (2001). Reducing drift in parametric motion tracking. In *Proc. 8th IEEE International Conference on Computer Vision (ICCV'01)*, vol. 1, pp 315–322, Vancouver, BC, Canada. 2.2.3

Reitmayr, G. & Drummond, T. (2006). Going out: Robust model-based tracking for outdoor augmented reality. In *Proc. 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'06)*, Santa Barbara, CA, USA. 5.1.2

Rosten, E. & Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Proc. 10th IEEE International Conference on Computer Vision (ICCV'05)*, vol. 2, pp 1508–1515, Beijing, China. 4.5.2.1, 6.3.6, 7.4

Rosten, E. & Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proc. 9th European Conference on Computer Vision (ECCV'06)*, vol. 1, pp 430–443, Graz, Austria. 4.5.2.1

Se, S., Lowe, D. & Little, J. (2001). Local and global localization for mobile robots using visual landmarks. In *Proc. 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, pp 414–420, Maui, HI, USA. 2.2.3

Shi, J. & Tomasi, C. (1994). Good features to track. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp 593–600, Seattle, WA, USA. 2.2.3, 4.5.2.1

Shin, M., Goldgof, D. & Bowyer, K. (1999). An objective comparison methodology of edge detection algorithms using a structure from motion task. In *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR '98)*, pp 190–197, Santa Barbara, CA, USA. 5.1.2

Sim, R., Elinas, P., Griffin, M. & Little, J.J. (2005). Vision-based slam using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*, pp 9–16, Edinburgh, Scotland. 4.1.2, 4.4.1

Sivic, J. & Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *Proc. 9th IEEE International Conference on Computer Vision (ICCV'03)*, pp 1470–1477, Nice, France. 7.2.2, 7.4, 7.5.1, 7.5.2, 7.5.3

Smith, P., Reid, I. & Davison, A. (2006). Real-time monocular SLAM with straight lines. In *Proc. British Machine Vision Conference (BMVC'06)*, pp 17–26, BMVA, Edinburgh. 5.1.2

Smith, R.C. & Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships. In *Proc. Fourth International Symposium on Robotics Research*, pp 467–474, Santa Clara, CA, USA. 2.1.1, 2.1.2, 2.2.3

Sola, J., Monin, A., Devy, M. & Lemaire, T. (2005). Undelayed initialization in bearing only slam. In *Proc. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pp 2499–2504, Edmonton, Alberta, Canada. 4.1.2, 4.5, 4.5.2.3

Szeliski, R. & Kang, S.B. (1993). Recovering 3d shape and motion from image streams using non-linear least squares. Tech. Rep. 3, Digital Equipment Coorporation, Cambridge Research Lab. 2.2.1

Taylor, C., Kriegman, D. & Anandan, P. (1991). Structure and motion in two dimensions from multiple images: a least squares approach. In *Proc. IEEE Workshop on Visual Motion*, pp 242–248, Princeton, NJ, USA. 2.2.1, 5.1.2

Taylor, C.J. & Kriegman, D. (1995). Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **17**, pp 1021–1033. 5.1.2

Thrun, S. & Montemerlo, M. (2006). The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, **25**, pp 403–429. 6.1.1, 6.1.3

Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H. & Ng, A. (2002). Simultaneous mapping and localization with sparse extended information filters. In *Proc. of The Eighth International Workshop on the Algorithmic Foundations of Robotics*, Nice, France. 2.3.4

Thrun, S., Liu, Y., Koller, D., Ng., A.Y., Ghahramani, Z. & Durrant-Whyte, H. (2004). Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, **23**, pp 693–716. 2.3.4

Tomasi, C. & Kanade, T. (1991). Factoring image sequences into shape and motion. In *Proc. IEEE Workshop on Visual Motion*, pp 21–28, Princeton, NJ, USA. 2.2.1

Torr, P., Fitzgibbon, A.W. & Zisserman, A. (1998). Maintaining multiple motion hypotheses over many views to recover matching and structure. In *Proc. 6th IEEE International Conference on Computer Vision (ICCV'98)*, pp 485–491, Bombay, India. 2.2.1

Torr, P.H.S. & Zisserman, A. (2000). Mlesac: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, **78**, pp 138–156. 7.7.3

Triggs, B. & Sdika, M. (2006). Boundary conditions for young - van vliet recursive filtering, late 2005, to appear in. *IEEE Transactions on Signal Processing*, **54**, pp 2365–2367. 7.4

Triggs, B., McLauchlan, P., Hartley, R. & Fitzgibbon, A. (2000). Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman & R. Szeliski, eds., *Vision Algorithms: Theory and Practice*, vol. 1883 of *Lecture Notes in Computer Science*, pp 298–372, Springer-Verlag. 2.2.1

van der Merwe, R., Doucet, A., de Freitas, N. & Wan, E. (2000). The unscented particle filter. Tech. Rep. 380, Cambridge University Engineering Department. 4.2.2, 4.2.2.2

Varadarajan, V. (1974). *Lie Groups, Lie Algebras and Their Representations*. No. 102 in Graduate Texts in Mathematics, Springer-Verlag. 3.2.2

Wang, Z., Huang, S. & Dissanayake, G. (2005). Decoupling localization and mapping using compact relative maps. In *Proc. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pp 3336–3341, Edmonton, Alberta, Canada. 2.3.4

Welch, G. & Bishop, G. (1995). An introduction to the Kalman filter. Tech. Rep. TR 95-041, University of North Carolina at Chapel Hill, updated 2002. 3.3

Williams, B., Klein, G. & Reid, I. (2007). Real-time SLAM relocalisation. In *Proc. 11th IEEE Int'l Conf. Computer Vision*. 7.2.1, 7.3.1, 7.7.3

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I. & Tardos, J. (2008). An image-to-map loop closing method for monocular slam. In *Proc. Int. Conf. Intelligent Robots and and Systems*, to appear. 7.2.2

Williams, S. (2001). *Efficient Solutions to Autonomous Mapping and Navigation Problems*. Ph.D. thesis, University of Sydney, Australia. 2.1.2, 2.3.3

Young, I.T. & van Vliet, L.J. (1995). Recursive implementation of the gaussian filter. *Signal Process.*, **44**, pp 139–151. 7.4