In theory, there is no difference between theory and practice. But, in practice, there is.

– Jan L. A. van de Snepscheut

University of Alberta

PERFORMANCE EVALUATION OF FEATURE EXTRACTORS FOR VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING

by

Jonathan Stephen Peters Klippenstein

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 2008

Your file Votre référence
ISBN: 978-0-494-45835-8
Our file Notre référence
ISBN: 978-0-494-45835-8

Abstract

Visual Simultaneous Localization And Mapping (SLAM) implementations must use a feature extraction method to reduce the dimensionality of image input, yet no comparison of feature extractors in this context exists. This thesis presents a framework for the comparison of performance using several different extractors and the first experimental study of feature extractor performance for visual SLAM. Evaluation is performed prior to SLAM processing using the recall and precision metrics and after SLAM processing using the novel accumulated uncertainty metric. Three feature extractors commonly used for visual SLAM are examined: the Harris corner detector, the Kanade-Lucas-Tomasi tracker, and the Scale-Invariant Feature Transform. All perform similarly in an indoor test environment, close to or within the limits of measurement error, and all feature extractors are capable of handling a modest scale change. This leads to the conclusion that feature extractor choice is not important with respect to SLAM performance.

To my parents, for a lifetime of support and encouragement.
To Claudia, for helping me through it all.

Acknowledgements

I would like to thank Dr. Hong Zhang for his support and guidance and for encouraging me in my education, even when tangential to the main goal. I have learned much during my time here under his supervision. I would also like to thank my colleagues in the department and robotics lab. Chris, John, Cathy, Linda, Dave, Jay, to name a few, thanks for the discussions, friendship, and badminton that we have shared over the years.

Funding for this research from the Natural Sciences and Engineering Research Council of Canada and the Alberta Informatics Circle of Research Excellence is gratefully acknowledged.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Localization and mapping are two key abilities required for mobile robot navigation. Without localization–the ability to track a robot's position in an environment–it is difficult to navigate between two points. Without a map or some sort of representation of the environment, it is difficult to perform tasks in that environment. These problems are complementary: with accurate localization we can construct a map, and with an accurate map we can determine a robot position. This leads to simultaneous estimation of both a map and robot position. Estimation is hindered since perfect data is never available in the real world. Data from wheel or inertial sensors used for localization is always corrupted by noise and subject to drift, while all sensors used to examine the environment have intrinsic noise. This motivates the field of *probabilistic robotics*, which treats state estimates and sensor measurements as probability densities instead of perfectly known values, and allows the robot to account for uncertainty in the system.

1.1 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) fuses the two problems to allow a robot to navigate without any prior information. As the robot travels through the world, pose uncertainty increases. New landmarks are added to the map as they are observed and re-observation acts to decrease state uncertainty. Total state uncertainty is kept from growing indefinitely and the robot is able to perform useful mapping and localization.

The following notation and derivation are based on Thrun *et al.* [46]. SLAM systems estimate a state $\mathbf{x} = (\mathbf{r}, \mathbf{m})$ consisting of a robot pose $\mathbf{r}$ and map of the environment $\mathbf{m}$. Robot pose is typically two- or three-dimensional robot position and orientation and a map is made up of sparse landmarks in space (typically points or lines) or an occupancy grid. The SLAM problem is to estimate the posterior state probability given a series of sensor observations $\mathbf{z}_{1:t} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_t\}$ and control inputs $\mathbf{u}_{1:t} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_t\}$. This posterior is often described as the *belief* as it represents a robot's internal knowledge of the world, and written as

$$bel(\mathbf{x}_t) = P(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \,. \tag{1.1}$$

Sensor observations measure a part of the environment, while control inputs represent motion commands or measurement.

The Bayes filter is a recursive state estimator that is used to incorporate new information into the system. Current SLAM systems are specific instances of the general Bayes filter. Two steps are required to generate a new estimate $bel(\mathbf{x}_t)$ from $bel(\mathbf{x}_{t-1})$: a prediction step that incorporates new a new control $\mathbf{u}_t$, followed by an observation step that incorporates a new sensor observation $\mathbf{z}_t$. For a complete derivation, see [45,46].

- *Prediction Step:* $P(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$ is calculated from $bel(\mathbf{x}_{t-1})$ and $\mathbf{u}_t$ as

$$\begin{aligned} P(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) &= \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)\ P(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})\ d\mathbf{x}_{t-1} \\ &= \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)\ bel(\mathbf{x}_{t-1})\ d\mathbf{x}_{t-1}. \end{aligned} \tag{1.2}$$

  The term $P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$ is known as the *motion model* and describes how the state evolves at each timestep given a control input.

- *Observation Step:* The posterior belief $bel(\mathbf{x}_t)$ is calculated from $P(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$ and $\mathbf{z}_t$ using Bayes' rule as

$$\begin{aligned} bel(\mathbf{x}_t) &= P(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \\ &= \frac{P(\mathbf{z}_t|\mathbf{x}_t)\ P(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{P(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} \\ &= \eta\ P(\mathbf{z}_t|\mathbf{x}_t)\ P(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \end{aligned} \tag{1.3}$$

  where the denominator has no dependence on $\mathbf{x}$ and is replaced by a normalizing constant η. The term $P(\mathbf{z}_t|\mathbf{x}_t)$ is known as the *observation model* or *sensor model* and describes how the current landmarks and robot pose map to sensor readings.

This formulation uses the Markov assumption, that is, the current state contains all information about the world, and past and future state are independent. Dynamic environments, model inaccuracies, and errors in the representing probabilities all violate this assumption, although in practice Bayesian filtering is quite robust.

Using the Bayes filter requires choosing a representation for the probabilities, and specifying the initial condition $P(\mathbf{x}_0)$, motion model, and observation model. Integration over a continuous space in the prediction step is generally intractable on digital computers, so how to implement this operation is a key consideration in choosing a probability representation.

The Kalman filter is an instantiation of the Bayes filter that is often used for SLAM [35,48]. It uses the assumption that the state is represented as a Gaussian random variable, of the form

$$P(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \Sigma; \mathbf{x}) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{1.4}$$

where N is the dimension of $\mathbf{x}$, and $\boldsymbol{\mu}$ and Σ are the mean and covariance matrix that specify the distribution. Additionally, motion and sensor models must be linear systems with noise modeled as

zero-mean Gaussian distributions. With these assumptions, the Bayes filter equations have a closed form applicable to digital computation. The extended Kalman filter (EKF) handles the case of systems that cannot be expressed as linear equations by linearizing around the current state estimate using a first-order Taylor expansion. Although this allows any system to be used within the Kalman framework, it introduces error that can cause filter failure.

The EKF has a long history of SLAM usage, dating back to the seminal work of Smith, Self, and Cheeseman [44], and has been proven to converge under certain conditions [14]. EKF-SLAM is known to have certain failings however, particularly that erroneous observations can introduce unrecoverable error into the state estimate, along with the aforementioned linearization error. Given N total map landmarks and L landmarks observed at one time, the update has $O(LN^2)$ complexity, which can be prohibitive for large maps. In spite of this, many successful implementations are based on EKF-SLAM. Variants of the EKF such as the unscented Kalman filter, the extended information filter, and the Gaussian sum filter have also been successfully applied to the SLAM problem.

The other major family of SLAM algorithms is based on particle filtering. Instead of a Gaussian distribution, probabilities are represented by a set of weighted particles, each of which contains a separate robot pose and map estimate. FastSLAM is the *de facto* standard and factorizes the posterior belief to achieve an update complexity of $O(M \log N)$ where M is the number of particles [39]. However, the optimal number of particles for a certain number of landmarks is currently unknown and interpreting the set of particles as a single result may be difficult. Although there are advantages to the particle filter approach, the system built in this thesis is based on the EKF-SLAM approach since it is a well established, simple foundation.

A number of SLAM sub-problems must be solved to construct a working system, as well as the fundamental state estimation problem. Of interest to this thesis are:

- *Feature extraction:* Sensors may return more information than can be feasibly processed. In this case the data is examined and interesting "features" are extracted, effectively reducing the dimensionality of the measurement.
- *Data association:* Features identified in sensor readings must be matched with existing map landmarks. Association varies greatly in complexity as some sensors are able to provide a unique description of each feature, while others simply report the presence of an object.
- *Feature initialization:* Previously unseen features must be classified as either a new landmark, an erroneous measurement, or an artifact from a previous error. Valid new landmarks are added to the map.

SLAM systems can be classified in terms of the category of the sensor used: range-only (RO), bearing-only (BO), or range-and-bearing (RB). Range-and-bearing sensors are able to detect both the distance and direction to an object in the environment, and as such there is a one-to-one mapping

between physical space and observation space. Laser range-finders, RADAR, and stereo cameras are included in this category. Range-only sensors such as acoustic SONARs have very poor or no bearing sensing, and bearing-only sensors such as monocular cameras cannot determine range. These sensors only return partial information about an object, but often use simpler or less expensive hardware at the cost of increased software complexity. Multiple observations are necessary to recover the missing information about a landmark.

1.2 Visual SLAM

Recently, cameras have been gaining attention as a SLAM sensor as hardware and computing power has become readily available. Their use is motivated by the fact that they are information-rich, compact, and fast. They give easy access to the third dimension as opposed to traditional laser range-finder systems which typically sweep out a plane perpendicular to the ground. A typical camera image contains on the order of a few hundred thousand pixels. Although the large amount of information is helpful when distinguishing different landmarks from each other, it comes at the cost of increased processing. Feature extraction must be performed to reduce the dimensionality of the input in order to use the visual data.

For visual SLAM, the feature extractor has two distinct parts: a *detector* that finds points according to some interest metric, and a *descriptor*, which is a representation of the immediate neighborhood around an interesting point. Different approaches to feature extraction vary in the metric used to define interesting points and the method used to describe the neighborhood around the points. Desirable properties of extractors include invariance to geometric changes in viewpoint such as scaling and rotation, and also to lighting changes that affect pixel intensities.

1.3 Thesis Definition

All visual SLAM implementations must consider the feature extraction process and choose a particular method, yet there are no comparisons of feature extractors in this context. The number of features successfully found and matched at each frame affects the overall system uncertainty, and extractors that are prone to false matches can cause the system to fail catastrophically. Feature extractor performance may be measured either before or after SLAM processing, as described below.

Measuring performance immediately after feature extraction (the "feature stage") before any further SLAM processing means that results are not conditioned on the particular SLAM implementation, so there are less assumptions about the system. However, SLAM performance is not directly measurable at this stage. Instead, human-derived metrics based on the *recall* and *precision* ratios [1, 37] are used. In brief, the ratio of matches made to all matches possible, and the ratio of correct matches to total matches made for different feature extractors are used to approximate SLAM performance.

Evaluation after SLAM processing (the "SLAM stage") has the advantage of directly measuring the desired quantity, but becomes dependent on the particular SLAM system implemented. SLAM performance is measured in this case by comparing the SLAM estimate to a ground truth to check for filter consistency, and then examining the amount of uncertainty present in the SLAM system with each feature extractor.

This thesis develops a framework for evaluating the effect of feature extractor choice on SLAM performance at both stages and then presents a comparison of three feature extractors commonly used for visual SLAM in order to determine the relative performance of each.

1.4 Thesis Organization

The remainder of this thesis is organized as follows.

- Chapter 2 examines previous work in the fields of feature extraction and matching, visual SLAM implementations, and performance evaluation.
- Chapter 3 presents the implementation of a visual SLAM system.
- Chapter 4 discusses how to overcome limitations of previous work and describes the methodology and results obtained from experiments conducted for performance evaluation at the feature stage.
- Chapter 5 discusses a new metric that allows for comparison of SLAM system results and describes the methodology and results obtained from experiments conducted for performance evaluation at the SLAM stage.
- Chapter 6 draws conclusions based on performance evaluation at both the feature and SLAM stages and describes future work.

Chapter 2

Related Work

A few main research areas are relevant to this project: feature extraction and matching methods, existing visual SLAM research, and performance evaluation of both feature extraction and SLAM results. Feature extraction and matching methods have been investigated in the context of image matching and object recognition for some time by the computer vision community, while visual SLAM has been addressed separately by robotics researchers. Performance evaluation has been addressed by both, in terms of measuring matching performance between pairs of images, and to a limited extent measuring SLAM performance.

2.1 Feature Extractors

While it is possible to mix and match detectors and descriptors, for simplicity this work only considers configurations commonly found in the literature. A brief overview is presented here, with references to detailed explanation.

2.1.1 Detectors

The Harris corner detector (more completely the Harris-Stephens detector, also known as the Plessey detector) is one of the most established and successful algorithms [18]. It is based on the detector of Moravec and attempts to rectify a number of limitations in the earlier work. For each pixel in an image, a matrix is formed that is related to the autocorrelation function. The matrix captures the principal curvatures of the image intensity, that is, how quickly the intensity changes in response to a small change in position. The eigenvalues of the matrix are proportional to the curvatures and are used to decide if a point is a corner, part of an edge, or a "flat" region of the image. A response function involving the trace and determinant of the matrix is used to avoid calculating the eigenvalues explicitly and local maxima points with response above a threshold are taken to be corners. The autocorrelation idea is also used in other detectors [38,40] but the Harris approach has proven the most popular. Two implementations are used in this thesis, one from Kovesi [31] and one from the OpenCV vision library [23].

The work of Shi and Tomasi [42], which follows from the pioneering work of Lucas and Kanade [6], approaches feature detection from a tracking perspective rather than a human-derived metric for "interestingness". Features are chosen for their suitability for the method used to track features from frame to frame, so in some sense it is "optimal by construction." The Lucas and Kanade method uses a Newton-Gauss style gradient based search to track an image patch through consecutive frames, estimating a displacement that minimizes the sum of squared differences. Interestingly, this leads to a similar metric as the Harris detector, where a point is chosen if both eigenvalues of a matrix similar to the autocorrelation matrix are above a threshold. Shi and Tomasi extend the work to estimate an affine warp between the current and original image patches, and use a measure of dissimilarity to detect tracking failure (which could occur due to occlusion, a feature leaving the camera field of view, or tracking a depth discontinuity that appears as a point to the camera but is actually a virtual point). An implementation by Birchfield [10] is used for this work and is referred to as the KLT.

Since the KLT and Harris methods are based on a very similar metric similar results may be expected. However, the method of tracking points between frames is very different. While the KLT actively tracks points, the Harris method must perform matching of feature descriptors across the entire image to find the new point. Therefore, some variation of results is expected.

The theory of *scale-space* shows that in addition to a two-dimensional image position, a third dimension, scale, can be constructed using successive Gaussian convolution, which calculates the appearance of the image as if seen from further away [32]. It can be shown that points invariant to scale can be found by generating a pyramid of scaled-down images and then searching for extrema of the second derivative of the Gaussian convolution (or Laplacian of Gaussian). This is utilized by Lowe in the scale-invariant feature transform (SIFT) [34]. SIFT approximates the Laplacian of Gaussian function using a difference of Gaussians and extrema are found in the images formed by subtracting adjacent levels in the Gaussian pyramid. The method is engineered to be robust, is invariant to scaling and rotation, and partially invariant to affine transformation. Lowe's reference implementation is used in this SLAM system [33].

While this is not an exhaustive list of all feature detectors, these three are representative of those commonly used for visual SLAM, and are thus the focus of this work. An overview of the full state of the art with respect to feature detectors is given by Mikolajczyk [38].

2.1.2 Descriptors

The most basic method of describing the local neighborhood of a point is by directly storing the raw image intensity values from a small square window around the point. This has the advantage of simplicity of computation but is not invariant to lighting changes, rotation, or viewpoint changes which may warp the image in an affine or projective manner. To overcome the susceptibility to lighting change, the descriptor can be normalized by subtracting the mean and scaling the values to cover a certain range (for example, the maximum range of the data type used for representation).

This can also be accounted for in the matching step instead. This style of descriptor is used with the Harris detector and the KLT tracker.

Studies have been performed on more sophisticated descriptors [37]. They have shown that the SIFT descriptor is one of the top performers. SIFT first computes gradient magnitude and orientation for every pixel in a small region around the point. The region is divided into 4×4 subregions, and an orientation histogram is formed for the subregion, with the contribution of each pixel orientation the histogram bins weighted by gradient magnitude. Scale and rotational invariance comes at the cost of additional computation and may not be usable in real-time systems.

2.2 Feature Matching

Features are tracked from frame to frame by using a matching process on their descriptors. Given two sets of features extracted from two images, the correct feature matches between the images should have descriptors that are similar by some metric.

The simplest matching method is the nearest neighbor algorithm where a descriptor's match is the closest descriptor in the other image in the sense of Euclidean distance under a maximum distance threshold. Lowe improves on this by also considering the distance to the second-nearest neighbor [34]. Matches are only accepted if the ratio of distances to the first- and second-nearest neighbors is less than a threshold. The reasoning behind this test is that incorrect matches will tend to match equally poorly to multiple features, so the ratio will approach unity. Gating based on this ratio ensures that accepted matches are in some sense unique. The concept of cross-correlation between two descriptors is also considered. The correlation coefficient between two vectors lies in the range $[-1, 1]$ and describes their similarity. Normalized cross-correlation accounts for a constant offset, or DC bias, in each descriptor, caused for example by lighting variation [17, p. 870]. The correlation coefficient is calculated as

$$\gamma(\mathbf{a}, \mathbf{b}) = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2 \sum_i (b_i - \bar{b})^2}} \tag{2.1}$$

where a_i and b_i represent elements of descriptor vectors $\mathbf{a}$ and $\mathbf{b}$, and $\bar{a}$ and $\bar{b}$ are the means. Matches are accepted if correlation is above a threshold value. Finally, the Lucas and Kanade gradient-based search algorithm is used in the KLT for tracking features through a sequence of images [6].

In addition to the matching described above, features are also filtered based on their position with respect to the camera. As described later, the estimated position of each landmark in the SLAM map is projected into the image, and only matched with those features detected within a certain distance of the projection. The distance is calculated from the uncertainty in robot pose, landmark position, and camera model. This gating greatly reduces false matches and enforces temporal continuity of landmark position.

2.3 Visual SLAM Implementations

Since this thesis is focused on evaluation of feature extractors, current visual SLAM implementations are examined to determine the extractors are in use. A few representative papers are detailed.

Davison has performed pioneering work in the field of bearing-only SLAM with a single camera [12] and uses the Shi and Tomasi (KLT without tracking) feature extractor along with an EKF-based system. Eade and Drummond [15] present a similar implementation, substituting a FastSLAM-based particle filter for the EKF. Lemaire *et al.* use Harris corners and the EKF, as do Kim *et al.* [28].

The visual range-bearing SLAM case uses cameras in a stereo configuration to recover approximate feature depth from a single measurement. Examples using SIFT and variations of the FastSLAM particle filtering algorithm include Sim *et al.* [43] and Goncalves *et al.* [16]. Barfoot has success with a similar algorithm outdoors, albeit in a somewhat manufactured setting [8].

Although other extractors have been utilized for SLAM, these three, Harris, KLT, and SIFT, are used in the majority of implementations and are therefore examined here.

2.4 Performance Evaluation

Performance evaluation has been treated at both the feature stage as a vision problem, and the SLAM stage as a robotics problem.

2.4.1 Feature Stage Evaluation

Evaluating feature extractor performance before SLAM processing begins with verifying matches made between descriptors in different images. Mikolajczyk and Schmid find a homography between an image pair, which allows points to be mapped between the images [37]. However, since a homography is a mapping between two planes it is unable to capture the inter-frame geometry of a camera moving in an arbitrarily structured, non-planar environment. Shaw and Barnes overcome this limitation by generating fundamental matrices between frames when evaluating matching performance [41]. While the fundamental matrix can describe the geometry between two arbitrary scenes, a point in one image maps to an *epipolar line* in the second, along which the corresponding point must lie [20]. The mapping is no longer unique, but this is a necessary tradeoff in the case of non-trivial environments.

Once matches are determined to be correct or incorrect, it remains to quantify performance. Agarwal and Roth apply the *recall* and *precision* metrics from the data mining and information retrieval field to a visual classification task [1]. Ke and Sukthankar adapt this method when examining feature descriptors [27], and Mikolajczyk and Schmid adopt it for their survey of descriptors [37]. Curves describing feature extraction and matching performance are generated by varying system parameters and used to compare extractors. This is described in the discussion on feature stage performance evaluation in Chapter 4. Shaw and Barnes only consider the number of matches between

frames, which is a less useful metric as it does not include information quantifying the rate of false matches, which has important ramifications in terms of performance (both speed and the possibility of using a false match to update a SLAM system).

2.4.2 SLAM Result Evaluation

Most work in SLAM performance evaluation has simply been to show that the presented systems "work." This is often shown by displaying a final map and robot trajectory with quantification of error (for example [12,28,43]). In many cases it is difficult to obtain a ground truth for verification, so this is understandable. When a ground truth is available, such as in simulation or with an accurate GPS system outdoors, performance is given in terms of pose error or error plots with 2σ or 3σ limits derived from the estimated covariance (for example [8, 13, 26, 39]). Alternatively, the state error is normalized according to the estimated covariance. If the average normalized error over multiple Monte Carlo runs stays within error bounds it is judged to be consistent and offer a correct estimate of the environment. Consistency testing is described for the general case by Bar-Shalom and Fortmann [7], and rigorously applied to EKF-SLAM and FastSLAM by Bailey *et al.* [4,5].

These techniques are only able to judge if a system provides correct estimates, not compare performance between different systems or configurations. The novel concept of accumulated uncertainty as a metric for SLAM is introduced in Chapter 5 and used to compare performance of a SLAM system with different feature extractors.

Chapter 3

Bearing-Only Visual SLAM System

This chapter describes the implementation of a bearing-only visual SLAM system.[1] The system requires a camera and a mobile robot that is capable of returning odometry information. Two major components comprise the system: a SLAM system based on the extended Kalman filter (EKF) and a visual front-end that handles transforming image information into a form usable by the SLAM system. The venerable EKF-SLAM formulation serves as a base for this system [2, 44, 46]. A graphical overview of the system is shown in Figure 3.1, along with references to sections containing detailed description.

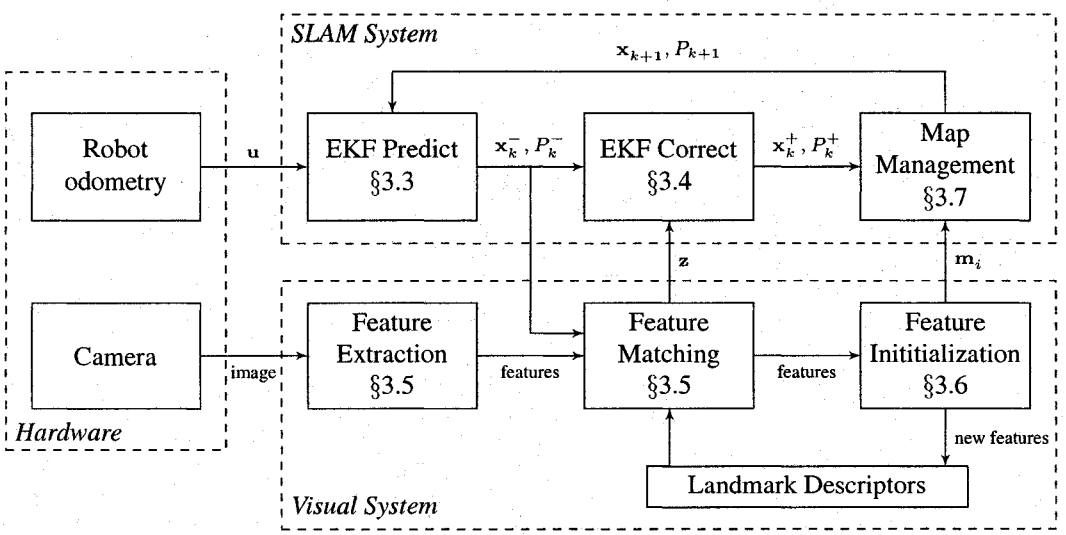


Figure 3.1: An overview of the visual SLAM system and guide to this chapter.

A single timestep proceeds as follows. First, an odometry reading is taken from the robot, and used by the EKF to predict forward the previously estimated state. Next, an image is captured from the camera, and interesting features are identified using a feature extractor. These are then matched

[1] Portions of this chapter have been published. J. Klippenstein, H. Zhang, and X. Wang. Feature Initialization for Bearing-Only Visual SLAM Using Triangulation and the Unscented Transform. In *Proc. of IEEE International Conference on Mechatronics and Automation*. Harbin, China, August 2007 [30].

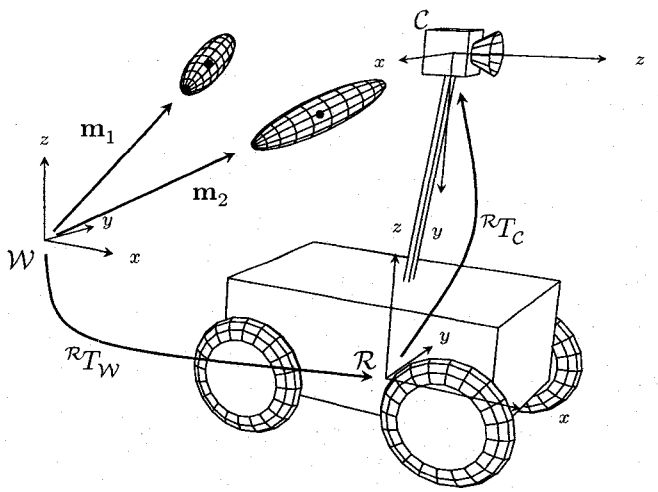


Figure 3.2: Physical setup and reference frames for the visual SLAM system. Robot and camera are related by an arbitrary transform. Robot and world frame are related through the robot position $\mathbf{r}$ and corresponding homogeneous transform ${}^{\mathcal{R}}T_{\mathcal{W}}$. Landmarks $\mathbf{m}_i$ are represented as Gaussians in the world frame.

to a list of features already contained in the SLAM map. Successful matches are used by the EKF to correct the predicted state and covariance estimates. Features not matched to the SLAM map are passed along to the feature initialization module and matched to stored features currently undergoing initialization. For these matches, the image points are triangulated to provide a three-dimensional landmark position estimate. Finally, a map management process adds valid newly initialized features to the map as well as deleting erroneous landmarks. This final state becomes the input value for the next SLAM iteration, depicted in the figure as a feedback loop. The remainder of this chapter is laid out in order of the processing steps.

3.1 Reference Frames

The physical setup, along with reference frames is illustrated in Figure 3.2. We have the following frames in the system:

- *World $\mathcal{W}$:* Base reference frame, coincides with the initial robot position.
- *Robot $\mathcal{R}$:* Coincides with the centre of the physical robot, with x-axis pointing toward the front of the robot, y-axis pointing left, and z-axis pointing up.
- *Camera $\mathcal{C}$:*. Origin at the optical centre of the camera, with z-axis pointing forward into the image, x-axis pointing to the right of the image, and y-axis pointing down.

Transformations between these frames are represented by 4×4 homogeneous transforms, using the notation ${}^{\mathcal{D}}T_{\mathcal{S}}$ to represent a matrix that transforms points from frame $\mathcal{S}$ to frame $\mathcal{D}$. That is,

$$\mathbf{X}^{\mathcal{D}} = {}^{\mathcal{D}}T_{\mathcal{S}}\mathbf{X}^{\mathcal{S}}$$

where $\mathbf{X}$ is expressed as a homogeneous 4-vector. These transforms have the form

$$ {}^{\mathcal{D}}T_{\mathcal{S}} = \begin{bmatrix} R_{3\times 3} & \mathbf{t}_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix} $$

where R is a rotation matrix and $\mathbf{t}$ is a translation vector. As usual, the inverse ${}^{\mathcal{S}}T_{\mathcal{D}}$ transforms points from $\mathcal{D}$ to $\mathcal{S}$.

The robot and camera are related by an arbitrary homogeneous transform ${}^{\mathcal{C}}T_{\mathcal{R}}$ that is found during an external calibration step. Since it is assumed the robot is moving in a two-dimensional plane, the robot to world transform for a robot at position (x, y) with heading θ is

$$ {}^{\mathcal{W}}T_{\mathcal{R}} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & x \\ \sin\theta & \cos\theta & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{\mathcal{R}}T_{\mathcal{W}} = \left({}^{\mathcal{W}}T_{\mathcal{R}}\right)^{-1} . \tag{3.1} $$

3.2 State Definition

The system tracks robot pose along with landmark positions. Robot movement is assumed to occur in the two-dimensional plane corresponding to $z = 0$, so the state tracked includes a two-dimensional position and a one-dimensional orientation, or heading angle. Landmarks are represented as a point in three-dimensional space without orientation or size. The state vector $\mathbf{x}$ is

$$ \mathbf{x} = \left(\mathbf{r}^T, \mathbf{m}_1^T, \ldots, \mathbf{m}_N^T\right)^T $$

where $\mathbf{r} = (x, y, \theta)^T$ is the robot position and heading and $\mathbf{m}_i = (m_x, m_y, m_z)^T$ is the position of the ith map landmark.

Additionally, the system maintains the full covariance matrix Σ corresponding to $\mathbf{x}$ that contains the variance of every variable and the cross-correlation between each pair, which in the SLAM formulation can be decomposed as

$$ \Sigma = \begin{bmatrix} \Sigma_r & \Sigma_{rm} \\ \Sigma_{rm}^T & \Sigma_m \end{bmatrix} \tag{3.2} $$

where Σ_r is the covariance of the robot pose $\mathbf{r}$, Σ_m is the covariance of the map landmarks, and Σ_{rm} represents the cross-correlation between the two.

3.3 Prediction Step

At the beginning of each timestep k, a new predicted state estimate $\mathbf{x}_k^-$ is generated by a motion model $\mathbf{f}$ that describes how the state evolves over time, given the previous estimate $\mathbf{x}_{k-1}^+$ and a control input $\mathbf{u}_k$, which is assumed to be a Gaussian random variable with covariance matrix Q

$$ \begin{aligned} \mathbf{x}_k^- &= \mathbf{f}\left(\mathbf{x}_{k-1}^+, \mathbf{u}_k\right) \\ \Sigma_k^- &= F_x \Sigma_{k-1}^+ F_x^T + F_u Q F_u^T \end{aligned} \tag{3.3} $$

where $F_x = \partial\mathbf{f}/\partial\mathbf{x}|_{\mathbf{x}_k^-}$ and $F_u = \partial\mathbf{f}/\partial\mathbf{u}|_{\mathbf{x}_k^-}$ are Jacobian matrices of $\mathbf{f}$ evaluated at the new state estimate $\mathbf{x}_k^-$.

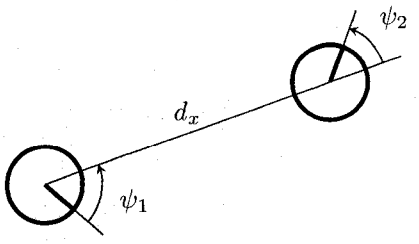


Figure 3.3: Robot motion parametrization. Motion from one point to a second is represented by the noisy control input $\mathbf{u} = (\psi_1, d_x, \psi_2)$: a rotation ψ_1, translation d_x, and second rotation ψ_2. After Thrun *et al.* [46, Figure 5.7].

3.3.1 Motion Model

Odometry information from the robot is prone to drift and accumulated error, but is useful as a control input under the assumption that it is locally correct. In other words, it is assumed that the relative motion between the two odometry points corresponds to the actual relative motion, even though the odometry reference frame does not coincide with the SLAM reference frame.

The odometry motion model from Thrun *et al.* [46, §5.4] is used since odometry is available and the model can represent arbitrary motions. Motion is parametrized as a noisy control vector $\mathbf{u}_k = (\psi_1, d_x, \psi_2)$ comprised of a rotation ψ_1, followed by a translation d_x along the x-axis, followed by a second rotation ψ_2, as shown in Figure 3.3.

Odometry from the robot is obtained as a pair of two-dimensional pose estimates $\mathbf{r}_k = (x_k, y_k, \theta_k)$ and $\mathbf{r}_{k-1} = (x_{k-1}, y_{k-1}, \theta_{k-1})$. The control parameters $\mathbf{u}$ can then be calculated as

$$\begin{aligned} \psi_1 &= \arctan\left(\frac{y_k - y_{k-1}}{x_k - x_{k-1}}\right) - \theta_{k-1} \\ d_x &= \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \\ \psi_2 &= \theta_k - \theta_{k-1} - \psi_1 \end{aligned} \tag{3.4}$$

where the rotation angles ψ_1, ψ_2 are mapped to $[-\pi, \pi]$.

The EKF state update function predicts the SLAM state vector forward in time using the motion model defined above, with landmarks assumed to be stationary. The update is defined as

$$\begin{pmatrix} \mathbf{r}_k^- \\ \mathbf{m}_{1,k}^- \\ \vdots \\ \mathbf{m}_{N,k}^- \end{pmatrix} = \mathbf{f}\left(\mathbf{x}_{k-1}^+, \mathbf{u}_k\right) = \begin{pmatrix} \mathbf{f}_r(\mathbf{r}_{k-1}^+, \mathbf{u}_k) \\ \mathbf{m}_{1,k-1}^+ \\ \vdots \\ \mathbf{m}_{N,k-1}^+ \end{pmatrix} \tag{3.5}$$

where the function $\mathbf{r}_k^- = \mathbf{f}_r\left(\mathbf{r}_{k-1}^+, \mathbf{u}_k\right)$ that updates robot pose is

$$\begin{pmatrix} x_k^- \\ y_k^- \\ \theta_k^- \end{pmatrix} = \mathbf{f}_r\left(\mathbf{r}_{k-1}^+, \mathbf{u}_k\right) = \begin{pmatrix} x_{k-1}^- + d_x \cos\left(\theta_{k-1}^- + \psi_1\right) \\ y_{k-1}^- + d_x \sin\left(\theta_{k-1}^- + \psi_1\right) \\ \theta_{k-1}^- + \psi_1 + \psi_2 \end{pmatrix}. \tag{3.6}$$

Noise is assumed to follow a Gaussian distribution with mean equal to the values in Equa-

tion (3.4), and variances proportional to the squares of the magnitudes of the variables

$$\begin{aligned}\sigma_{\psi_1}^2 &= \alpha_1\psi_1^2 + \alpha_2 d_x^2 \\ \sigma_{d_x}^2 &= \alpha_3 d_x^2 + \alpha_4(\psi_1^2 + \psi_2^2)\,. \\ \sigma_{\psi_2}^2 &= \alpha_1\psi_2^2 + \alpha_2 d_x^2\end{aligned} \tag{3.7}$$

Process noise covariance matrix Q is represented as a diagonal covariance matrix consisting of these variances

$$Q = \begin{bmatrix}\sigma_{\psi_1}^2 & 0 & 0 \\ 0 & \sigma_{d_x}^2 & 0 \\ 0 & 0 & \sigma_{\psi_2}^2\end{bmatrix}\,. \tag{3.8}$$

This matrix is recalculated from $\mathbf{u}$ at each timestep using Equation (3.7).

Jacobian matrices F_x and F_u required to linearize the system are derived in Appendix A.1.

3.4 Correction Step

For the correction step, the EKF uses the difference between a measurement and a prediction based on the current state $\mathbf{x}_k^-$ to correct the state estimate. Without loss of generality, it is assumed that a single landmark is observed in a timestep. Multiple observations are dealt with in Section 3.4.2. For visual SLAM this prediction is generated by projecting a three-dimensional landmark position to a two-dimensional image point, given the current robot position. Feature extraction and matching on a current camera image generates an observation that can be used in the EKF. The mechanics of the observation update are discussed in this section, before describing the implementation of feature extraction and matching in Section 3.5.

After an observation $\mathbf{z}$ is obtained from a landmark in the environment a corrected estimate $\mathbf{x}_k^+$ is generated from the predicated state estimate $\mathbf{x}_k^-$. An observation model $\mathbf{h}$ calculates a predicted observation $\hat{\mathbf{z}}$ given the currently estimated robot pose $\mathbf{r}^-$ and ith landmark position $\mathbf{m}_i^-$. Observation $\mathbf{z}$ is assumed to be a Gaussian random variable with covariance R. The difference between the measured observation and the prediction of the observation model is known as the innovation $\boldsymbol{\nu}$, which has a corresponding innovation covariance S that combines robot pose uncertainty and measurement uncertainty

$$\begin{aligned}\boldsymbol{\nu} &= \mathbf{z} - \hat{\mathbf{z}} = \mathbf{z} - \mathbf{h}(\mathbf{r}^-, \mathbf{m}_i^-) \\ S &= H_x \Sigma_k^- H_x^T + R\end{aligned} \tag{3.9}$$

where $H_x = \partial\mathbf{h}/\partial\mathbf{x}|_{\mathbf{x}_k^-}$ is the Jacobian of $\mathbf{h}$ with respect to the SLAM state, evaluated at the current estimate $\mathbf{x}_k^-$, and has the form

$$H_x = \begin{bmatrix}H_r & 0 & \ldots & 0 & H_m & 0 & \ldots\end{bmatrix} \tag{3.10}$$

where H_r is the Jacobian with respect to the robot pose and H_m is the Jacobian with respect to the observed map landmark. Finally, the Kalman gain W is calculated and a corrected estimate of state

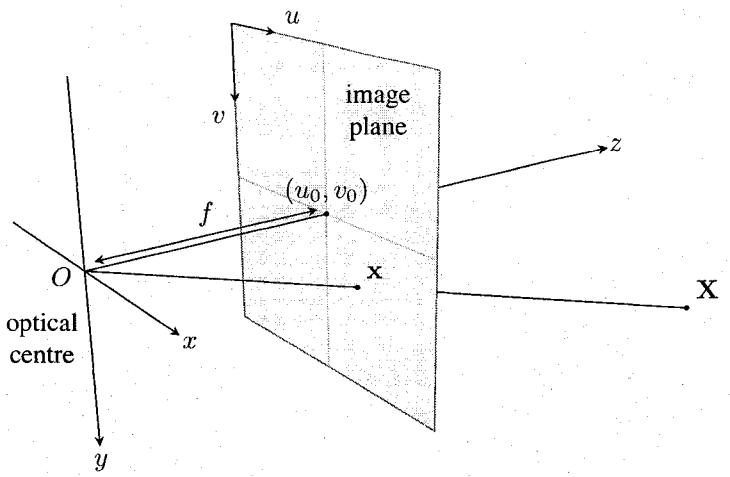


Figure 3.4: Pinhole camera model. A 3D point $\mathbf{X}$ is mapped to a 2D image point $\mathbf{x}$ using Equation (3.12). The effect of radial distortion is not depicted. In a real camera images are formed on the image sensor behind the optical centre, but the image plane is shown here in front of the centre for convenience.

$\mathbf{x}_k^+$ and covariance Σ_k^+ is generated

$$\begin{aligned} W &= H_x \Sigma_k^- S^{-1} \\ \mathbf{x}_k^+ &= \mathbf{x}_k^- + W\boldsymbol{\nu} \\ \Sigma_k^+ &= \Sigma_k^- - WSW^T . \end{aligned} \tag{3.11}$$

This new estimate is now used as the starting point for the next iteration. The EKF assumes the Markov property, so all information about the system is contained in the most recent estimate.

3.4.1 Observation Model

A standard pinhole camera model [20,21,49] as shown in Figure 3.4 is used. Second-order radial distortion is accounted for, although not shown in the figure. This is a six-parameter model with parameters $\pi = (f_x, f_y, u_0, v_0, k_1, k_2)$, consisting of focal lengths (f_x, f_y), a principal point (u_0, v_0), and radial distortion parameters (k_1, k_2). The measurement $\mathbf{z}$ is a point in the image plane calculated from the observation function $\mathbf{h}\left(\mathbf{r}^-, \mathbf{m}_i^-\right)$ using the currently estimated robot position $\mathbf{r}^-$ and ith map landmark position $\mathbf{m}_i^-$

$$\mathbf{h}\left(\mathbf{r}, \left(\mathbf{m}_i^-\right)^{\mathcal{C}}\right) = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \left(1 + k_1 r^2 + k_2 r^4\right) \frac{1}{m_z} \begin{pmatrix} f_x m_x \\ f_y m_y \end{pmatrix} \tag{3.12}$$

where $r^2 = (m_x^2 + m_y^2)/m_z^2$, and $\left(\mathbf{m}_i^-\right)^{\mathcal{C}} = (m_x, m_y, m_z)$ is the position of the landmark in the image coordinate system $\mathcal{C}$. The landmark position in image frame is obtained from the world frame through a series of homogeneous transformations

$$\mathbf{M}^{\mathcal{C}} = {}^{\mathcal{C}}T_{\mathcal{R}}\, {}^{\mathcal{R}}T_{\mathcal{W}} \mathbf{M}^{\mathcal{W}} . \tag{3.13}$$

Uncertainty in the observation model is assumed to come from error in intrinsic camera calibration. The observation noise covariance matrix R_π contains the variances of the camera parameters along the diagonal

$$R_\pi = diag\left(\sigma_{f_x}^2, \sigma_{f_y}^2, \sigma_{u_0}^2, \sigma_{v_0}^2, \sigma_{k_1}^2, \sigma_{k_2}^2\right) . \tag{3.14}$$

Mapping the camera parameter variance into the image-space covariance R requires a Jacobian transformation using the Jacobian H_π of $\mathbf{h}$ with respect to the camera parameters

$$R = H_\pi R_\pi H_\pi^T . \tag{3.15}$$

An additional small diagonal covariance is added to the resulting covariance matrix for u and v to account for possible location error in the feature extractor and is typically set to half a pixel.

Jacobians H_x and H_π are derived in Appendix A.2.

3.4.2 Batch Updating

When multiple landmarks are observed in a single image it is possible to call the EKF observation step multiple times sequentially, or combine the observations and perform a single batch update. Batch updating is performed by simply concatenating the observation vectors and Jacobian matrices. For n simultaneous observations, the combined observation and Jacobian are

$$\mathbf{z}_b = \begin{pmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_n \end{pmatrix} \qquad H_{x,b} = \begin{bmatrix} H_{x,1} \\ \vdots \\ H_{x,n} \end{bmatrix} \tag{3.16}$$

and combined observation noise covariance matrix and Jacobian are

$$R_{\pi,b} = \begin{bmatrix} R_{\pi,1} & 0 & \cdots & 0 \\ 0 & R_{\pi,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & R_{\pi,n} \end{bmatrix} \qquad H_{\pi,b} = \begin{bmatrix} H_{\pi,1} & 0 & \cdots & 0 \\ 0 & H_{\pi,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & H_{\pi,n} \end{bmatrix} . \tag{3.17}$$

It can be shown that both methods have the same asymptotic complexity, although the batch update has a larger constant factor. Batch updating is used in this system since the sequential update will produce different results depending on the ordering of observations, which is undesirable.

3.5 Feature Extraction and Matching

Observations for the SLAM filter are generated by applying the feature extraction techniques from Section 2.1 to new camera images. The feature extractor processes the entire image and returns a set of feature points with associated descriptors.

Matching is accomplished by first determining the projections of the current SLAM map landmarks in the current image and then matching each map landmark with only those extracted features that lie within a threshold of the landmark position. Equations (3.12) and (3.13) are used to project the map landmark from world-space to the image plane, resulting in a predicted observation $\hat{\mathbf{z}}$. A

landmark is within the perceptual range of the camera if its position in the camera frame lies within the view frustum defined by the field of view (FOV) angles FOV_x and FOV_y defined in Appendix B. The angles θ_x and θ_y between the x and y axes and the map landmark $\mathbf{m}^C$ in camera-space are

$$\begin{aligned} \theta_x &= \arctan\left(\frac{m_x^C}{m_z^C}\right) \\ \theta_y &= \arctan\left(\frac{m_y^C}{m_z^C}\right) . \end{aligned} \tag{3.18}$$

Map landmarks that lie in the current FOV satisfy the following properties

$$\begin{aligned} m_z^C &> 0 \\ \theta_x &< \frac{FOV_x}{2} \\ \theta_y &< \frac{FOV_y}{2} . \end{aligned} \tag{3.19}$$

Once the ith landmark is determined to lie in the camera FOV the innovation covariance S_i is calculated according to Equation (3.9). This is the combination of uncertainty in robot pose, landmark position, and camera parameters, and describes the region in which the correct match should lie. The Mahalanobis distance is then used to measure the distance between the ith projected map landmark $\hat{\mathbf{z}}_i$ and the jth extracted feature $\mathbf{z}_j$

$$r_{ij}^2 = \left(\hat{\mathbf{z}}_i - \mathbf{z}_j\right)^T S_i^{-1} \left(\hat{\mathbf{z}}_i - \mathbf{z}_j\right) . \tag{3.20}$$

This metric weights distance based on the shape of the uncertainty region and is used as a gate to reject features outside the 95% or 99% confidence limit derived from the innovation covariance.

After all extracted features that lie within a map landmarks uncertainty region have been found, the nearest-neighbor with distance ratio (NNDR) feature matching algorithm is used to find the correct match. Any matches found are used in the SLAM correction step described above.

3.6 Feature Initialization

New landmarks in the EKF-SLAM framework must have a fully specified position and covariance matrix, which is difficult or impossible in the bearing-only case since the many-to-one mapping inherent in bearing-only sensors cannot be inverted. A landmark could conceivably be initialized with a Gaussian uncertainty that extends out to infinity. However, a Gaussian cannot adequately capture this uncertainty, as very large uncertainties cause linearization issues, and "heavy-tailed" distributions cannot be correctly processed by the EKF.

To recover a full position in the bearing-only case, observations of the same landmark are accumulated from multiple robot positions, and triangulation is used to determine the intersection of the bearings. This process is illustrated in Figure 3.5. The unscented transform is used to estimate the landmark covariance by combining the robot pose covariances and camera model uncertainties.

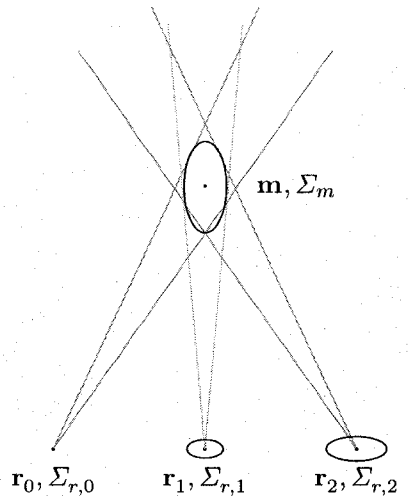


Figure 3.5: Feature initialization process visualized. Landmark position $\mathbf{m}$ and covariance Σ_m are recovered from a series of image points $\mathbf{z}_k$ that represent bearings to the landmark from robot poses $\mathbf{r}_k$ with covariance $\Sigma_{r,k}$. The cones represent uncertainty in bearing measurement from each pose. The triangulation process is carried out in the $\mathcal{R}_o$ initial robot pose reference frame.

After each new observation, the estimate is recalculated and checked to determine if it is valid, and valid landmarks are initialized into the SLAM map. This process runs separately from the main SLAM filter until a landmark is ready for promotion to the SLAM map. The algorithm is depicted in flowchart form in Figure 3.6 and described below.

In the remainder of this section three reference frames are used. Robot pose and SLAM landmarks are described in the world frame $\mathcal{W}$. Triangulation is performed in the frame defined by the robot pose corresponding to the initial observation of the the landmark, denoted $\mathcal{R}_o$. The camera frame $\mathcal{C}$ differs from the robot frame by a transformation ${}^{\mathcal{C}}T_{\mathcal{R}}$ as described in Section 3.1.

3.6.1 Local Robot Pose Estimation

During feature initialization, robot pose is tracked with respect to the reference frame representing the pose from which the landmark was initially observed. Triangulation and landmark covariance estimation is also performed in this local frame so that the result is independent of the uncertainty of the initial robot pose. When a landmark is ready for insertion into the SLAM map it is transformed to the world frame, a process described later in Section 3.6.5.

When a new feature is first observed the current robot pose $\mathbf{r}^{\mathcal{W}}$, covariance $\Sigma_r^{\mathcal{W}}$, and observation $\mathbf{z}_0$ are stored. The reference frame corresponding to this pose is denoted $\mathcal{R}_o$ and the current robot pose $\mathbf{r}_0^{\mathcal{R}_o}$ and covariance $\Sigma^{\mathcal{R}_o}$ are set to zero.

Local pose and covariance estimates are predicted forward in time using the same prediction step as the main SLAM system, given in Equation (3.3). The same motion model $\mathbf{f}$ and control vector $\mathbf{u}$ as before are used, but the state contains only the locally estimated robot pose, without landmarks.

After obtaining a new observation and correcting the main SLAM filter, the observation infor-

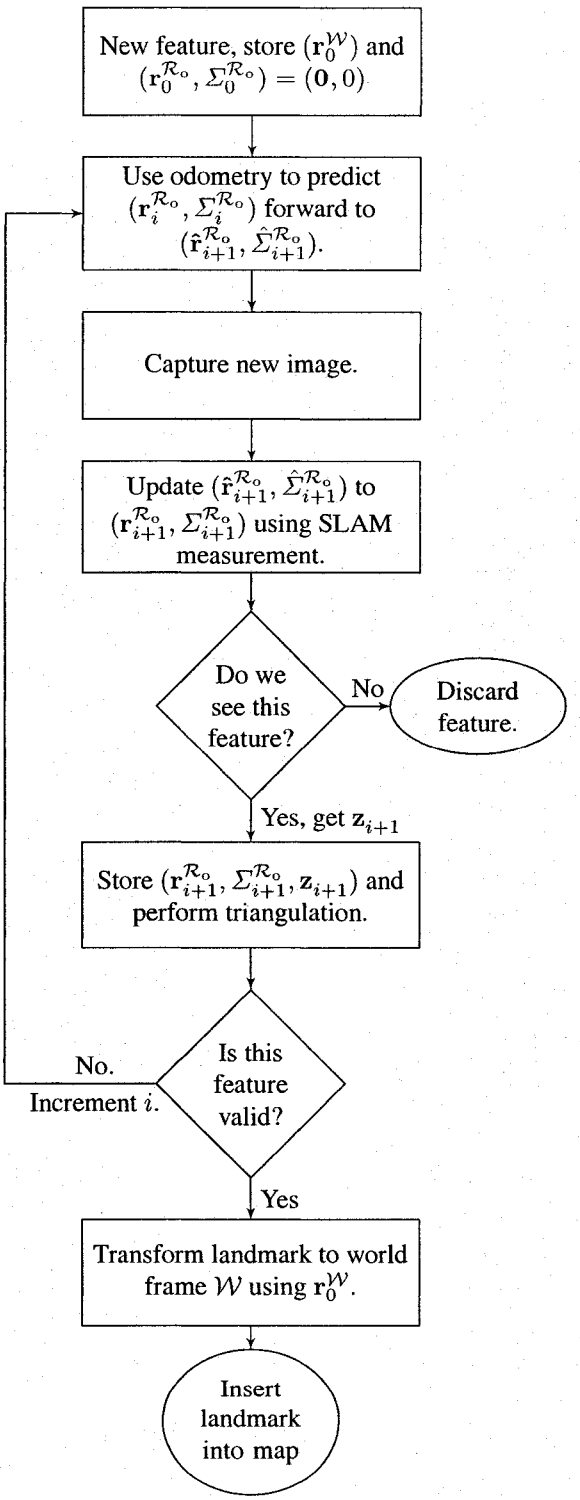


Figure 3.6: Feature initialization procedure using triangulation and the unscented transform. See text for details.

mation is used to update the local pose estimate and covariance. This corrects the local estimate and results in more accurate triangulation than using the prediction step alone, thus improving the final landmark estimates. The local correction step is applied in the same manner as Equation (3.11) using the innovation ν and innovation covariance S directly from the SLAM filter correction step, and the Jacobian H_x in Equation (3.10) consists of only H_r.

3.6.2 Triangulation

Once the robot pose estimate is corrected, the direct linear transformation (DLT) triangulation method [19, 20] is used to recover the landmark position. Although the technique is not projective-invariant and therefore not optimal, it is simple, provides acceptable results, and is applicable to multiple views.

Given n images, the kth image contains a two-dimensional measurement in image space, $\mathbf{z}_k = (u_k, v_k)$ related to the three-dimensional landmark position $\mathbf{m}_k^{\mathcal{R}_o} = (m_x, m_y, m_z)$ that is to be recovered. Points are then represented as homogeneous coordinates such that $\mathbf{Z}_k = (wu_k, wv_k, w)$ is the measurement, where w is an unknown scale factor, and $\mathbf{M}^{\mathcal{R}_o} = (m_x, m_y, m_z, 1)$ is the landmark. This relationship can be written in terms of a 3×4 projection matrix P_k

$$\mathbf{Z}_k = P_k \mathbf{M}^{\mathcal{R}_o} . \tag{3.21}$$

In this case, the projection matrices involve the intrinsic camera calibration matrix K, the camera to robot transform ${}^{\mathcal{R}}T_{\mathcal{C}}$, and a transform ${}^{\mathcal{R}_k}T_{\mathcal{R}_o}$ that has the same form as ${}^{\mathcal{R}}T_{\mathcal{W}}$ from Equation (3.1), replacing $\mathcal{W}$ with $\mathcal{R}_o$ and $\mathcal{R}$ with $\mathcal{R}_k$ as

$$\begin{aligned} K &= \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \\ P_k &= K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} {}^{\mathcal{C}}T_{\mathcal{R}} \, {}^{\mathcal{R}_k}T_{\mathcal{R}_o} . \end{aligned} \tag{3.22}$$

It can be shown that each image yields two equations that can be expressed in the form

$$\begin{aligned} \left(u_k (\mathbf{p}_k)_3^T - (\mathbf{p}_k)_1^T\right) \mathbf{M}^{\mathcal{R}_o} &= 0 \\ \left(v_k (\mathbf{p}_k)_3^T - (\mathbf{p}_k)_2^T\right) \mathbf{M}^{\mathcal{R}_o} &= 0 \end{aligned} \tag{3.23}$$

where $(\mathbf{p}_k)_j^T$ is the jth row of P_k. Since landmark position is three-dimensional, this implies that at least two views are required. Given n images a $2n \times 4$ matrix A is formed such that $A\mathbf{M}^{\mathcal{R}_o} = \mathbf{0}$,

$$A = \begin{bmatrix} u_0 (\mathbf{p}_0)_3^T - (\mathbf{p}_0)_1^T \\ v_0 (\mathbf{p}_0)_3^T - (\mathbf{p}_0)_2^T \\ \vdots \\ u_n (\mathbf{p}_n)_3^T - (\mathbf{p}_n)_1^T \\ v_n (\mathbf{p}_n)_3^T - (\mathbf{p}_n)_2^T \end{bmatrix} . \tag{3.24}$$

Since there are always more equations than unknowns the system is over-constrained, with very little likelihood of an exact solution existing since there will always be some error in image measurements and pose estimates. It can be shown that a solution to this problem that minimizes residual error is the unit singular vector corresponding to the smallest singular value of A. This is obtained using the singular value decomposition (SVD)

$$A = UDV^T \tag{3.25}$$

where the columns of U and V are the left- and right-singular vectors of A and D is a diagonal matrix of size equal to A containing the singular values. The three-vector result $\mathbf{m}^{\mathcal{R}_o}$ is then found by normalizing the homogeneous result $\mathbf{M}^{\mathcal{R}_o}$.

3.6.3 Landmark Estimation

As mentioned, applying triangulation techniques to probabilistic robotics requires accounting for the uncertainty in triangulation parameters. In this work, the *unscented transform* [24] is used to perform this uncertainty propagation. A Gaussian distribution representing all the parameters is formed and deterministically sampled, with each sample containing the information required to perform a triangulation. The landmark estimates resulting from performing triangulation on the samples are then recombined into final landmark position and covariance estimates. For convenience, the $\mathcal{R}_o$ reference frame designator will be dropped from symbols in the rest of this section. It should be understood that $\mathbf{r}$ and Σ_r will now refer to estimates in the local reference frame.

The mean vector $\boldsymbol{\mu}$ is formed by concatenating the n estimated robot poses $\mathbf{r}_k$ in $\mathcal{R}_o$ and the intrinsic camera parameters $\boldsymbol{\pi}$

$$\boldsymbol{\mu} = \left((\mathbf{r}_1)^T, \ldots, (\mathbf{r}_n)^T, \boldsymbol{\pi}^T \right)^T \tag{3.26}$$

where $\mathbf{r}_0$ is not included since it and its covariance $\Sigma_{r,0}$ are identically zero, and its omission decreases computation as less samples are required for uncertainty propagation. The covariance Σ_μ of this mean vector is formed as a block-diagonal matrix containing pose estimate covariance matrices $\Sigma_{r,k}$ and a diagonal block with the variances of the intrinsic camera parameters

$$\Sigma_\mu = \begin{bmatrix} \Sigma_{r,1} & 0 & \cdots & 0 & 0 \\ 0 & \Sigma_{r,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & \cdots & \Sigma_{r,n} & 0 \\ 0 & 0 & \cdots & 0 & \Sigma_\pi \end{bmatrix}. \tag{3.27}$$

The set $\{\boldsymbol{\mathcal{X}}\}$ of $2N+1$ *sigma points* are generated by sampling the covariance matrix Σ_μ

$$\begin{aligned} \boldsymbol{\mathcal{X}}_0 &= \boldsymbol{\mu} \\ \boldsymbol{\mathcal{X}}_i &= \boldsymbol{\mu} + \left(\sqrt{(N+\lambda)\Sigma_\mu} \right)_i \qquad i = 1, \ldots, N \\ \boldsymbol{\mathcal{X}}_i &= \boldsymbol{\mu} + \left(\sqrt{(N+\lambda)\Sigma_\mu} \right)_{i-N} \qquad i = N+1, \ldots, 2N \end{aligned} \tag{3.28}$$

where N is the dimension of $\boldsymbol{\mu}$ and $\lambda = \alpha^2(N+\kappa) - N$ is a scaling parameter. The notation $\left(\sqrt{(N+\lambda)\Sigma_\mu}\right)_i$ refers to the ith row of the square root of the covariance matrix Σ_μ scaled by $(N+\lambda)$, where the square root of the matrix can be found as, for example, the Cholesky decomposition. The α parameter determines how close the sample points are to the mean $\boldsymbol{\mu}$ and is usually set to a small positive number, and κ is an additional scaling parameter usually set to zero unless the state size is very small.

Once the sigma points have been determined the samples are transformed into a set of result points $\{\boldsymbol{\mathcal{Y}}\}$ according to the triangulation function described in the previous section. In this case the triangulation is performed using the sampled robot poses and camera parameters contained in $\{\boldsymbol{\mathcal{X}}\}$

$$\boldsymbol{\mathcal{Y}}_i = \text{triangulate}(\boldsymbol{\mathcal{X}}_i) \quad i = 0, \ldots, 2N \tag{3.29}$$

Transformed points are recombined to form the new landmark position and covariance estimates

$$\begin{aligned} \mathbf{m}^{\mathcal{R}_o} &= \sum_{i=0}^{2N} W_i^{(m)} \boldsymbol{\mathcal{Y}}_i \\ \Sigma_m^{\mathcal{R}_o} &= \sum_{i=0}^{2N} W_i^{(c)} \left(\boldsymbol{\mathcal{Y}}_i - \mathbf{m}^{\mathcal{R}_o}\right)\left(\boldsymbol{\mathcal{Y}}_i - \mathbf{m}^{\mathcal{R}_o}\right)^T \end{aligned} \tag{3.30}$$

using the weights W_i given by

$$\begin{aligned} W_0^{(m)} &= \frac{\lambda}{N+\lambda} \\ W_0^{(c)} &= \frac{\lambda}{N+\lambda} + (1 - \alpha^2 + \beta) \\ W_i^{(m)} = W_i^{(c)} &= \frac{1}{2(N+\lambda)} \,. \end{aligned} \tag{3.31}$$

Since this technique does not depend on a particular triangulation algorithm, other algorithms can easily be substituted for the above linear triangulation.

3.6.4 Landmark Validity

Large uncertainty causes problems with the EKF linearization process, so introducing landmarks with large covariance into the map must be avoided. The distance along the optic axis to the landmark, or depth, has the largest uncertainty due to the geometry of the system. New landmarks are accepted as valid only if the ratio of the magnitude of depth to the standard deviation in depth is less than 30%. This heuristic has been shown to be effective in rejecting poor landmark estimates [12,47].

3.6.5 Map Augmentation

Once a new landmark has been acceptably validated, it remains to transform it from the initial robot frame $\mathcal{R}_o$ to the world frame $\mathcal{W}$ and add it to the SLAM map. This is a straightforward transformation using the robot to world transformation of Equation (3.1), which is expressed in non-homogeneous coordinates as

$$\mathbf{m}^{\mathcal{W}} = \mathbf{g}(\mathbf{r}^{\mathcal{W}}, \mathbf{m}^{\mathcal{R}_o}) = {}^{\mathcal{W}}R_{\mathcal{R}_o}\mathbf{m}^{\mathcal{R}_o} + \mathbf{t}^{\mathcal{W}} \,. \tag{3.32}$$

Transforming the covariance requires the Jacobian matrix of this equation with respect to $\mathbf{m}^{\mathcal{R}_o}$

$$\Sigma_m^{\mathcal{W}} = G_m \Sigma_m^{\mathcal{R}_o} G_m^T \tag{3.33}$$

where $G_m = \partial \mathbf{g} / \partial \mathbf{m}^{\mathcal{R}_o}$.

Finally, augmented versions of the SLAM state vector and covariance matrix are formed with the new landmark pose $\mathbf{m}_{n+1}^{\mathcal{W}}$ and covariance $\Sigma_m^{\mathcal{W}}$ from Equations (3.32) and (3.33)

$$\begin{aligned} \mathbf{x}_a &= \left(\mathbf{r}^T, \mathbf{m}_1^T, \ldots, \mathbf{m}_m^T, \left(\mathbf{m}_{n+1}^{\mathcal{W}}\right)^T\right)^T \\ \Sigma_a &= \begin{bmatrix} \Sigma_r & \Sigma_{rm} & \Sigma_r G_r^T \\ \Sigma_{rm}^T & \Sigma_m & \Sigma_{rm}^T G_r^T \\ G_r \Sigma_r & G_r \Sigma_{rm} & \Sigma_m^{\mathcal{W}} + G_r \Sigma_{r,0}^{\mathcal{W}} G_r^T \end{bmatrix} \end{aligned} \tag{3.34}$$

where $G_r = \partial \mathbf{g} / \partial \mathbf{r}^{\mathcal{W}}|_{\mathbf{r}_0^{\mathcal{W}}}$ is the Jacobian of $\mathbf{g}$ with respect to initial robot pose, and is used to incorporate uncertainty of the initial robot pose reference frame $\mathcal{R}_o$ into the new landmark covariance estimate. This result is derived in full in [2, §2.2.4]. These augmented versions are then taken as the new state and covariance estimates for the next iteration.

Jacobian matrices G_m and G_r are derived in Appendix A.3.

3.7 Map Management

Due to the quadratic scaling of the EKF, it is not computationally feasible to use every valid landmark. Therefore, a criterion for selecting a subset of landmarks to use in the SLAM map is required. The approach taken uses simple heuristics to determine which landmarks to add to the map, and when to remove landmarks that are no longer useful.

3.7.1 Landmark Addition

The assumption is made that there is little benefit to having two landmarks very close together. Enforcing a maximum local density is a more efficient placement, as more space can be covered with fewer landmarks. Also, feature matching data association as described in Section 3.5 is more successful if landmarks are spaced further apart as there is less chance of overlap and confusion. Based on this assumption, in order for a new landmark $\mathbf{m}_{N+1}$ to be added to the map, it must be further than a minimum threshold d_{min} from all existing landmarks in three-space

$$\left\|\mathbf{m}_i - \mathbf{m}_{N+1}\right\| > d_{min} \qquad i = 1, \ldots, N \tag{3.35}$$

where $\|\cdot\|$ is the vector 2-norm.

For an additional sparsification measure, a maximum local density in image-space is enforced. Each image is divided into a number of axis-aligned cells, and new landmarks are only added to the map if there are less than a threshold number of features projected into a particular cell. This also reduces possibly landmark confusion by ensuring landmarks to not get too close together in image-space.

Once a new landmark passes these tests, the SLAM state and covariance are augmented according to Section 3.6.5.

3.7.2 Landmark Deletion

Eventually features may need to be deleted when they are no longer useful to the SLAM process. This occurs as new features sometimes quickly become occluded, or simply fail to match in subsequent frames. To decide when to delete a landmark, a method is required to determine if a particular feature should be seen given the current robot position. This is accomplished using the method described above in the discussion on feature matching, Section 3.5. A successful observation occurs when the landmark is calculated to be visible (in the camera FOV) and is observed. An observation failure occurs when the landmark is calculated to be visible and is not observed.

The posterior probability of the existence of a landmark is tracked using a binary Bayes filter [46, §4.2], [39]. The probability of an indicator variable, $exist$, is estimated and represents the probability of the landmark existing. Estimation is performed using the log-odds formulation, since it is convenient and numerically stable. Log-odds of $exist$ is calculated as

$$l(exist) = \frac{P(exist)}{1 - P(exist)} \,. \tag{3.36}$$

The rule to update the log-odds given an observation consisting of map landmark $\mathbf{m}$ and robot position $\mathbf{r}$ is

$$l_t = l_{t-1} + \log \frac{P(exist|\mathbf{m},\mathbf{r})}{1 - P(exist|\mathbf{m},\mathbf{r})} - \log \frac{P(exist)}{1 - P(exist)} \tag{3.37}$$

where $P(exist|\mathbf{m},\mathbf{r})$ is the conditional probability that a landmark exists given an observation, and $P(exist)$ is the prior probability of a landmark existing.

Conditional probability $P(exist|\mathbf{m},\mathbf{r})$ is calculated using the perceptual range test discussed above. Since there is a non-zero probability a landmark is within range but not observed, this probability is not $1/0$:

$$P(exist|\mathbf{m},\mathbf{r}) = \begin{cases} p & \text{if landmark } \mathbf{m} \text{ is in camera FOV when robot is at } \mathbf{r}, \\ 1-p & \text{if landmark } \mathbf{m} \text{ is out of camera FOV.} \end{cases} \tag{3.38}$$

where p is typically a large value.

The prior probability $P(exist)$ biases the posterior calculation since in general a landmark is more likely to exist than not exist. Since there is a higher occurrence of actual valid landmarks than false results, this prior is typically set to bias the filter towards accepting new landmarks as valid

Finally, the log-odds value is used as a threshold for landmark acceptance. The value is updated every time the landmark is calculated to lie within the perceptual range of the robot. If it becomes negative (corresponding to a probability of existence of less than 50%), we eliminate the corresponding landmark from the map. Conversely, landmarks undergoing initialization must have a positive log-odds ratio of existence before being added to the map.

3.8 Summary

This chapter described a complete bearing-only visual SLAM system based on the well-established EKF-SLAM framework with a visual front-end. Two-dimensional robot pose is tracked along with three-dimensional point features, and a full covariance matrix describes the state uncertainty. Prediction in the EKF uses a model based on odometry, where motion is parametrized as a rotation, translation, and rotation. This parametrization is capable of describing arbitrary motion of a differential drive robot. Observations are modeled using a pinhole camera model with radial distortion. Feature initialization is performed with a novel technique using triangulation and the unscented transform. Map management is accomplished by estimating the log-odds ratio of existence for a landmark to keep computational requirements within reasonable limits. This system has been implemented from scratch and forms the basis of experiments in the following chapters.

Chapter 4

Performance Evaluation at the Feature Stage

This chapter examines performance at the feature extraction stage before SLAM processing.[1] Dependence of the results on the choice of SLAM algorithm is removed, but a metric must be derived that approximates SLAM performance based only on feature extraction and matching. The approach taken is to first extract features from a sequence of images, match features between all images, then validate the matches. The recall and precision metrics are used to compare the relative performance of different extractors.

4.1 Methodology

A sequence of images is captured using a digital camera mounted to a mobile robot. Images are stored for later processing due to computational cost and the need to perform multiple experiments on the same data.

Fundamental matrices are used to represent the geometric relationship between each pair of images in sequence, incorporating both the intrinsic camera parameters and the transformation between image viewpoints. A fundamental matrix F satisfies the relationship

$$\mathbf{x}_1^T F \mathbf{x}_2 = 0 \tag{4.1}$$

where $\mathbf{x}_1$ is a homogeneous image coordinate in an image and $\mathbf{x}_2$ is the point in a second image that represents the same physical point. Since finding the fundamental matrix for each image pair manually is infeasible due to the large number of images, an implementation of Hartley and Zisserman's automated method is used, which is based on robust estimation using RANSAC [20,31].

Features are extracted from images using the feature extractors described in Section 2.1. SIFT uses a descriptor length of 128, while the Harris and KLT detectors use an 11×11 image patch descriptor, yielding a descriptor of dimension 121. KLT is also used purely as a detector, without

[1] Portions of this chapter have been published. J. Klippenstein and H. Zhang. Quantitative Evaluation of Feature Extractors for Visual SLAM. In *Proc. of Fourth Canadian Conference on Computer and Robot Vision*. Montreal, Canada, May 2007 [29].

tracking. Matching of features is performed using the methods described earlier in Section 2.2: nearest-neighbor with distance ratio (NNDR), normalized cross-correlation (NCC). KLT tracking is used only for the KLT tracker. Feature matches are verified using the generated fundamental matrices. Given a point in an image, the corresponding point in a second image must lie along the epipolar line, which is calculated from the fundamental matrix. For a match to be correct the point in each image must lie within a small distance of the epipolar line corresponding to the matching point in the other image.

4.1.1 Performance Metrics: Recall and Precision

The ideal feature extractor produces a set of features that maximizes the rate of correct matches while minimizing the rate of incorrect matches. The *recall* and *precision* metrics are used to quantify these attributes. Recall is the percentage of correct matches out of all possible matches; it measures how well a particular extractor is able to find features that are matchable between images. A low recall may indicate that while the extractor can find a large number of features, different features are found in each frame, so no tracking or matching is possible. Precision is the percentage of correct matches out of all the matches actually made.[2] Low precision indicates that incorrect matches are made, which can lead to catastrophic failure of the SLAM system.

To calculate these values over a sequence of images, the numbers of correct matches, total matches, and possible matches between each pair of images in sequence are summed. Given a sequence of N images $\{\mathcal{I}\}$,

$$\begin{aligned} \text{recall} &= \frac{\sum_{i=1}^{N-1} \text{correct}(\mathcal{I}_i, \mathcal{I}_{i+1})}{\sum_{i=1}^{N-1} \text{possible}(\mathcal{I}, \mathcal{I}_{i+1})} \\ \text{precision} &= \frac{\sum_{i=1}^{N-1} \text{correct}(\mathcal{I}_i, \mathcal{I}_{i+1})}{\sum_{i=1}^{N-1} \left(\text{correct}(\mathcal{I}_i, \mathcal{I}_{i+1}) + \text{incorrect}(\mathcal{I}_i, \mathcal{I}_{i+1})\right)} \end{aligned} \tag{4.2}$$

where $\text{correct}(\mathcal{I}_i, \mathcal{I}_{i+1})$, $\text{incorrect}(\mathcal{I}_i, \mathcal{I}_{i+1})$, and $\text{possible}(\mathcal{I}_i, \mathcal{I}_{i+1})$ represent the numbers of correct, incorrect, and possible matches between the $i-$th and $(i+1)-$th images. It is worth noting the distinction between the concept of total matches made and matches possible. Extractors may not choose the same set of interest points in a pair of images, so some features in one image may not have a corresponding feature in the other image. Such features should count as a possible match, but will not be included in the total matches (correct plus incorrect matches). Since it is impossible to tell exactly which features should remain in the field of view of an image based on another image, the number of possible matches is approximated as the minimum of the number of features detected in either image, as this is the maximum assuming one-to-one matching.

[2] This definition of precision comes from the field of information retrieval and differs slightly from the usage in other fields of science. We adopt this definition to match the relevant literature [1, 37].

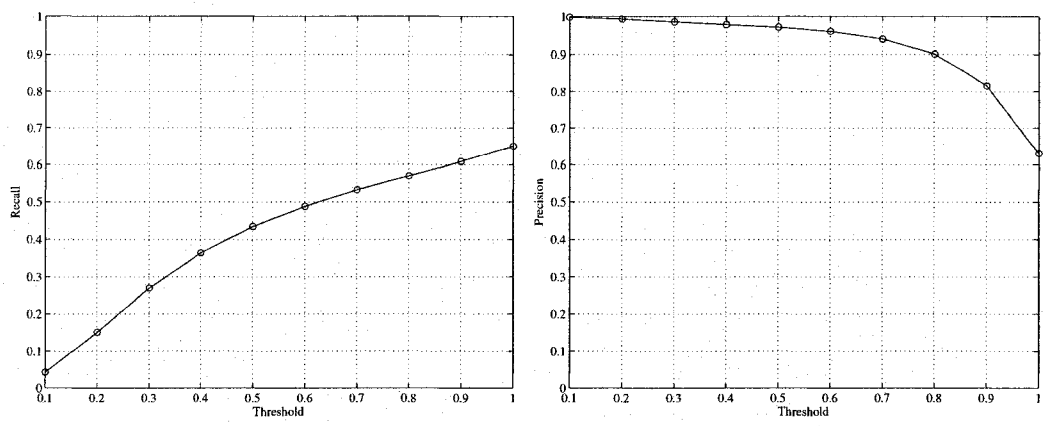


Figure 4.1: Example plots of recall and precision against a matching algorithm threshold parameter. Interpreting the plots separately like this is difficult as recall must be minimized below a certain value as precision is simultaneously maximized above a certain value. This motivates the use of two-dimensional recall vs. precision plots.

4.1.2 Interpreting Recall and Precision Curves

Performance in terms of recall and precision is generated by varying parameters of the matching algorithm. This allows for objective comparison between feature extractors since the effect of the matching algorithm is eliminated. Performance corresponding to any point along the curve can be obtained by choosing the proper parameter value.

While recall and precision can be displayed separately as shown in Figure 4.1, it is difficult to compare extractors since this requires minimizing a value on one graph while simultaneously maximizing a value on the other. A more useful and informative technique is to display a 2-D plot of recall versus precision, as in Figure 4.2.

It should be noted that 1−precision (that is, one minus precision) is plotted instead of precision, to match the style of existing literature [1, 37]. Recall, which is to be maximized, runs along the vertical axis while 1−precision, which is to be minimized, runs along the horizontal axis.

Plotting recall against 1−precision allows for the comparison of feature extractors. Distance between curves gives an indication of the relative performance of feature extractors in different regions. Optimal performance corresponds to the upper-left corner of a recall/1−precision plot, with perfect recall and zero 1−precision (equivalent to perfect precision). Correctness of matches if of primary importance when evaluating performance. However, the recall rate at which correct matches are produced must be considered, since perfect matching is not useful if too few matches are made as matches are required to reduce the overall uncertainty in the SLAM system. A lower bound on recall and precision defines a rectangular region in the recall/1−precision graphs that is considered acceptable performance. A lower bound on recall is approximated by the ratio of the minimum desired correct matches to the average number of possible matches for a particular feature

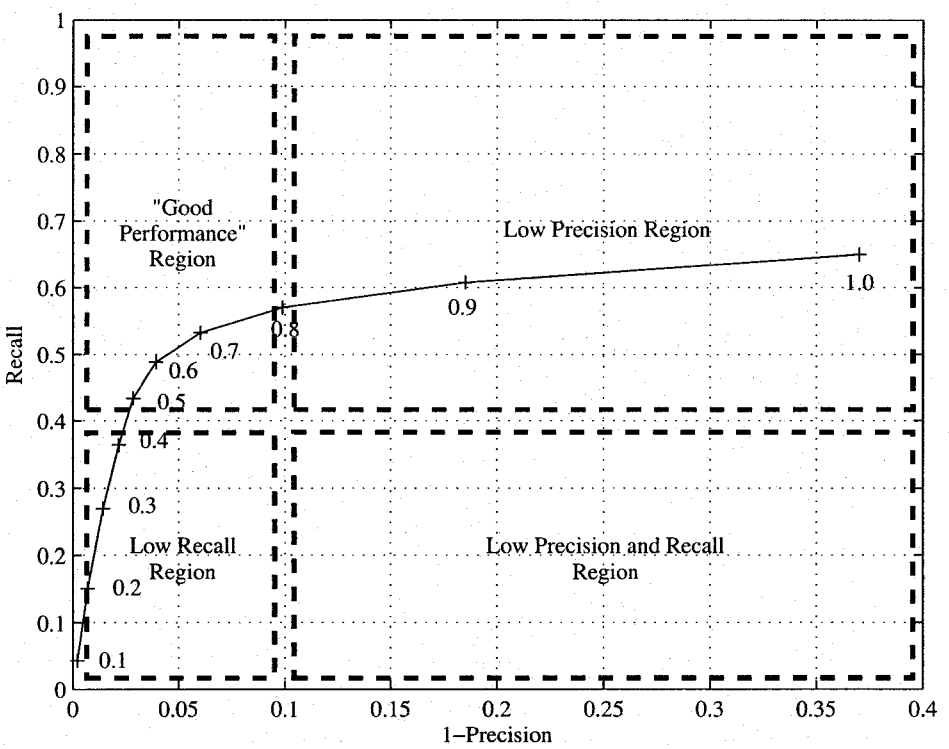


Figure 4.2: Example recall/1-precision (one minus precision) plot. Numbers along the curve represent matching algorithm threshold parameters corresponding to the recall and precision attained at that point. Performance regions are marked with dashed lines. The region of acceptable performance is a rectangular area since this is a two-dimensional display of recall and precision and feature extractor performance must enter this region to be considered useful for SLAM.

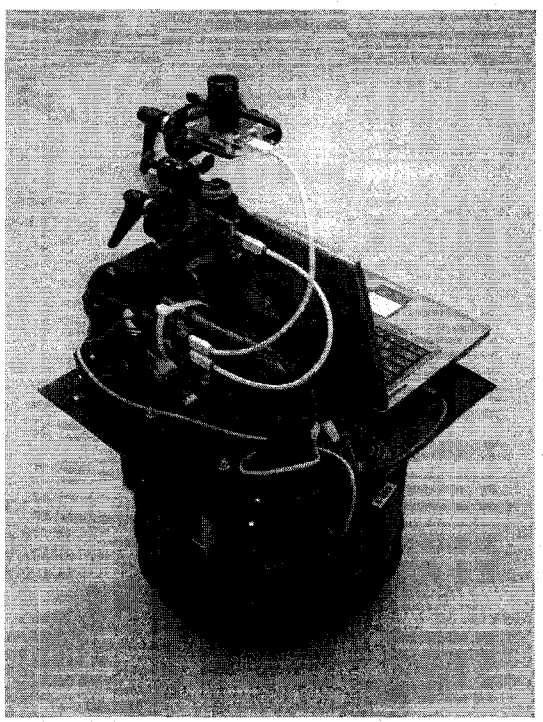


Figure 4.3: The iRobot Magellan Pro with Point Grey Research Dragonfly IEEE-1394 camera used for feature stage experiments.

extractor

$$\text{min. recall} = \frac{\text{min. desired correct matches}}{\frac{1}{N}\sum_{i=1}^{N-1} \text{possible}(\mathcal{I}, \mathcal{I}_{i+1})} . \tag{4.3}$$

4.2 Experimental Setup

An iRobot Magellan Pro robot was used to collect a set of images with a Dragonfly IEEE-1394 digital camera from Point Grey Research. Bayer-encoded images were captured at a resolution of 640×480 pixels and converted to grayscale. The camera was calibrated using an implementation of Zhang's calibration method [11, 49], and images are undistorted using a function in the same package. Two sets of images taken in environments of different sizes were created by driving the robot manually while capturing images. The hardware setup is pictured in Figure 4.3.

The *hallway* dataset was created on the third floor of the University of Alberta Computing Science Centre, which is fairly typical of an office building environment. An image was taken after the robot performed a 150 mm translation or 5° rotation, whichever was encountered first. A sample image is shown in Figure 4.4. The total distance traveled is around thirty metres and observed features are anywhere from one to thirty metres from the camera. The dataset is available online at the Robotics Data Set Repository (Radish) [22] as `ualberta-csc-flr3-vision`.

A second dataset, *lab*, was taken in the robotics laboratory by driving the robot manually and capturing an image every 100 mm or 5° rotation, with a total length of around ten metres. A sample

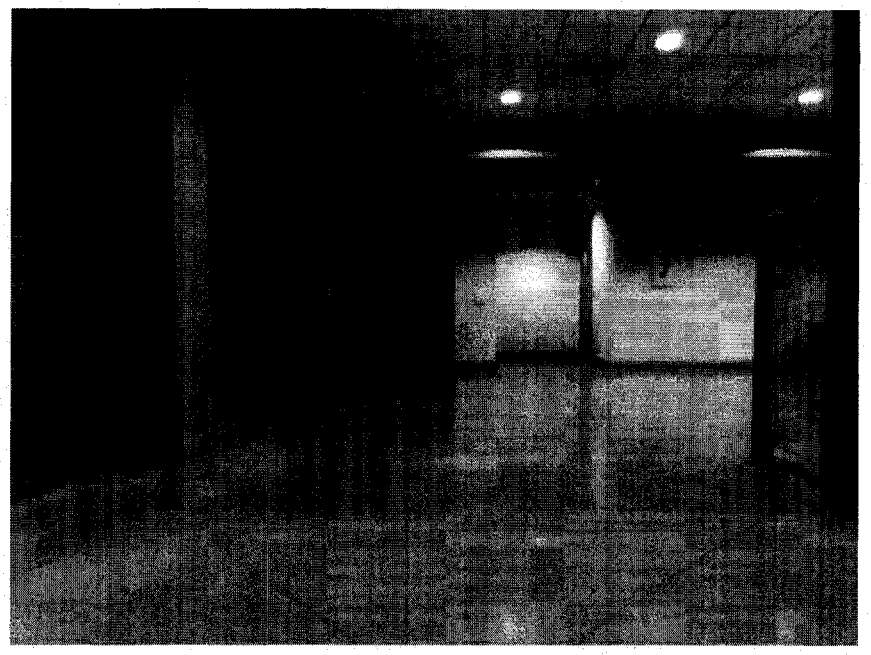


Figure 4.4: Sample image for feature-stage performance evaluation from the *hallway* dataset.

image is shown in Figure 4.5.

Since the lab environment is a more confined space, observed features are generally no more than five metres from the camera. While the open space and corridor of *hallway* is more regular and structured, the *lab* images contain more dissimilar objects and clutter. This allows for testing extraction and matching at different feature depths, and thus arriving at general conclusions.

4.3 Results and Discussion

The datasets described above were processed using combinations of the described feature extractors and matchers. Results are displayed using the recall/1−precision graphs discussed above. Further analysis was performed by stepping over frames, effectively subsampling the image sequence, in order to determine how the performance is affected when distance between images increases. Performance is considered "good" if an extractor can achieve high precision with recall greater than a small threshold, a heuristic discussed further in the next section.

4.3.1 Characterizing "Good" Performance

In order to determine what constitutes "good" performance, a minimum acceptable recall rate is calculated using Equation (4.3). As a simple heuristic, it is desirable to have at least on the order of ten correct matches between images. The average number of possible matches was calculated for each dataset and are shown in Table 4.1. The Harris extractor has the smallest average of 113 possible matches, which results in a recall of 9% required to meet the heuristic using Equation (4.3). This is doubled to allow for some uncertainty, so the assumption made is that the feature extractors perform

Figure 4.5: Sample image for feature-stage performance evaluation from the *lab* dataset.

Extractor	*hallway*	*lab*
SIFT	283	467
Harris	113	156
KLT	395	219
KLT + tracking	426	220

Table 4.1: Average possible matches per frame for feature extractors on the *hallway* and *lab* datasets. The smallest number is returned by the Harris extractor on the *hallway* dataset.

acceptably when recall is greater than 20%. Additionally, precision must be greater than 95%, or equivalently 1−precision must be less than 5%, as false matches can cause SLAM algorithms to fail catastrophically.

4.3.2 Comparison of Matching Techniques

Performance of the NNDR and NCC matching techniques was very similar, especially in the lower precision (larger 1−precision) region of the curve. This trend can be seen for the *hallway* dataset in Figure 4.6. NNDR and NCC curves appear to converge as 1−precision decreases. This is likely because the two techniques are similar in nature, as both are based on a distance measure between normalized descriptors.

Only NNDR results are considered for the remainder of this chapter for simplicity of presentation. Performance of the two feature matchers is similar, with results for NCC matching being no better than NNDR results. Additionally, NNDR has a higher recall value for a given 1−precision in the low 1−precision region, which is the desired behavior.

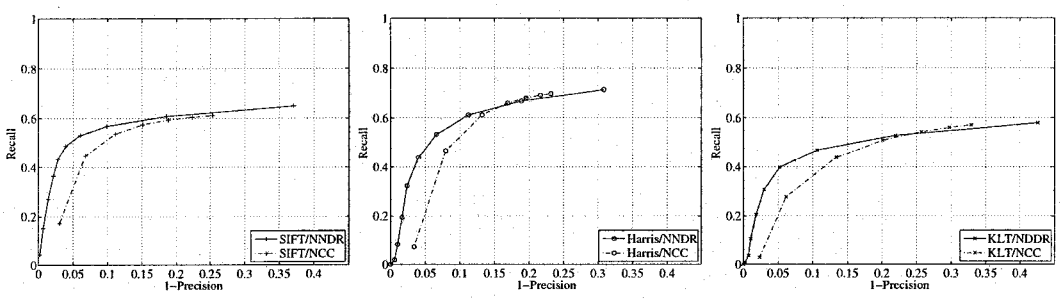


Figure 4.6: Comparison of nearest-neighbor with distance ratio (NNDR) and normalized cross-correlation (NCC) matching techniques on the *hallway* dataset. Solid lines represent NNDR matching, while dashed lines represent NCC matching. NNDR outperforms NCC by achieving a higher recall value for the same precision.

4.3.3 Comparison of Feature Extractors

Comparing the performance of the different feature extractors is the key consideration. Performance of SIFT, Harris, and KLT with and without tracking on the *hallway* dataset is shown in Figure 4.7. Performance data for the *lab* dataset is shown in Figure 4.8.

The results show that in general any of the feature extractors can be made to perform well with a suitable choice of parameters. For every extractor in Figure 4.7 and Figure 4.8 it is possible to find a point on the curve with precision above 95% and recall of about 40% or better. KLT with tracking outperforms the other extractors, as it achieves the highest recall in the high precision region. Tracking leads to better performance than simply matching interest points detected independently in two images. Additionally, the KLT curves with tracking are nearly vertical, which suggests that the KLT precision is robust to changes in KLT parameters. SIFT and Harris detectors both give acceptable and similar performance, with SIFT giving slightly higher recall in the high precision region.

Although KLT with tracking performs very well, differences between image viewpoints are small. This means the robustness of the extractors is not being thoroughly tested, and motivates the comparison performed in the next section.

4.3.4 Comparison of Feature Extractors with Subsampling

A desirable property in feature extractors is invariance to viewpoint changes. Image sequence subsampling was performed to analyze this by stepping over images during processing, increasing the distance between image viewpoints. Figure 4.9 compares SIFT, Harris, and KLT with and without tracking for the *hallway* dataset, using every fifth image, corresponding to approximately 750 mm or 25° between each frame. Figure 4.10 does the same for the *lab* dataset, using every fourth image, corresponding to a 400 mm or 20° change.

The recall for all extractors in Figure 4.9 and Figure 4.10 is noticeably lower than before, since some potential matches are no longer visible and there is no way to account for this effect in the automated procedure. The KLT with tracking remains nearly vertical, although there is some loss

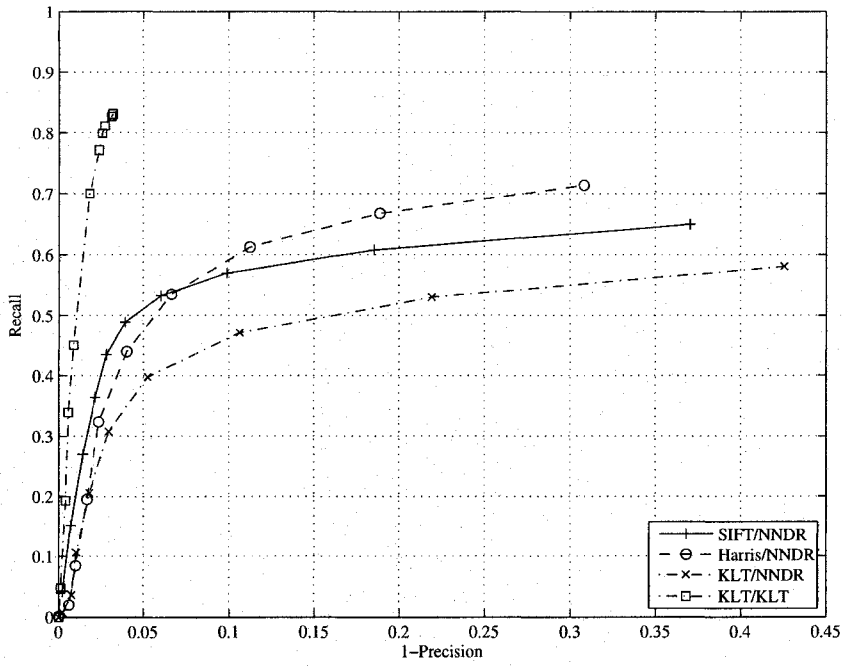


Figure 4.7: Comparison of Harris corner detector, Kanade-Lucas-Tomasi (KLT) tracker, and Scale-Invariant Feature Transform (SIFT) feature extractors on the *hallway* dataset using NNDR and KLT tracker matchers. Images are spaced approximately 150mm or 5° apart. KLT with tracking achieves the best performance in terms of recall for the same precision, followed by SIFT and Harris.

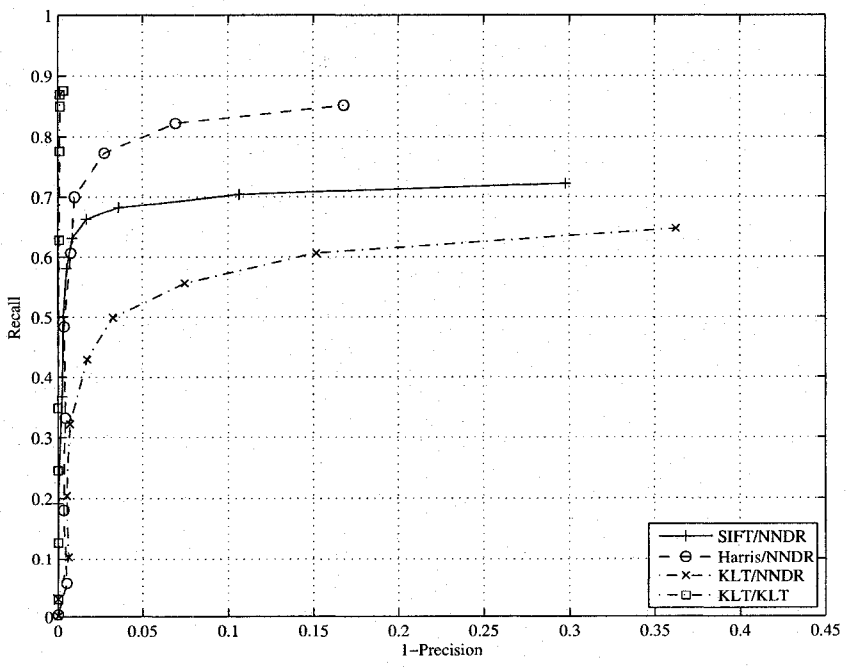


Figure 4.8: Comparison of Harris, KLT, and SIFT feature extractors on the *lab* dataset using NNDR and KLT tracking matchers. Images are spaced approximately 100 mm or 5° apart. The KLT with tracking curve rises almost vertically along the recall axis, representing superior performance, and is followed closely by Harris and SIFT.

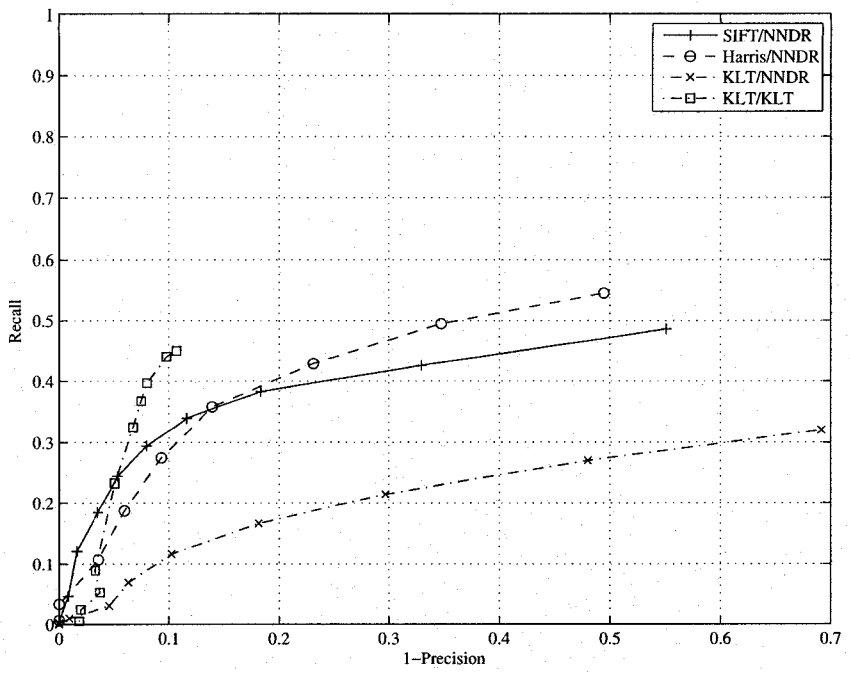


Figure 4.9: Comparison of Harris, KLT, and SIFT extractors on the *hallway* dataset using NNDR and KLT tracking matchers with image stepping. Every fifth image is considered, resulting in a 750mm or 20° change between images. SIFT achieves better performance in the very high precision region, while KLT is still able to obtain high recall with reasonable precision.

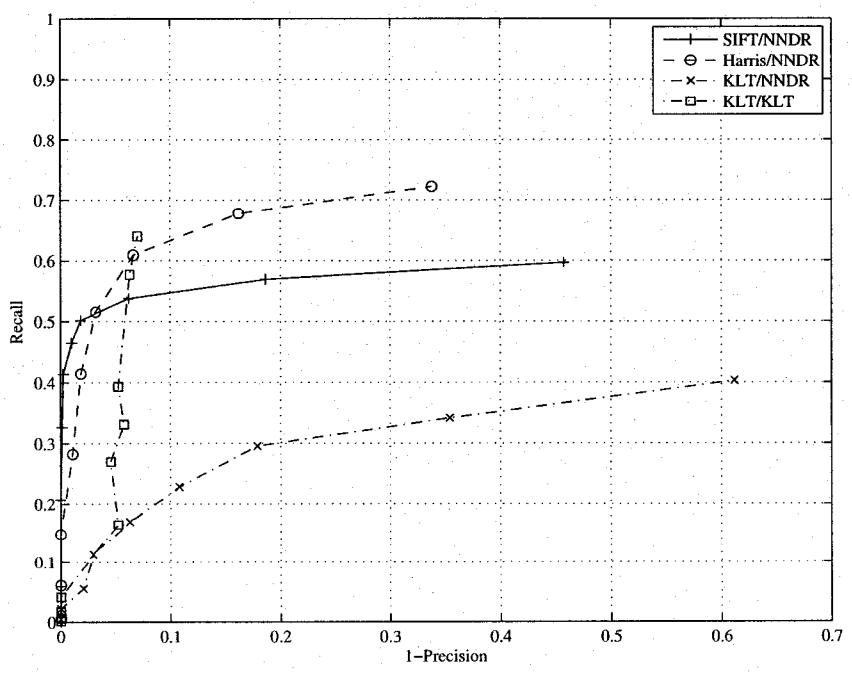


Figure 4.10: Comparison of Harris, KLT, and SIFT extractors on the *lab* dataset using NNDR and KLT tracking matchers with image stepping. Every fourth image is considered, resulting in a 400mm or 20° change between images. SIFT and Harris outperform KLT at high precision.

in precision. This indicates that the constraints that tracking places on matching are able to better discriminate between correct and incorrect matches. The relative performance of the extractors also appears to remain constant.

4.4 Summary

This chapter described experiments conducted to evaluate performance of feature extractors at the feature stage before SLAM processing. Datasets were collected with a mobile robot in an indoor setting. The geometric relationship between each pair of images was determined by an automated method, and used to verify matches. A metric based on recall and precision was used to compare performance. These experiments extend previous work in the area by considering recall and precision over a sequence of images rather than a single pair, with arbitrary rather than planar scenes.

The experimental results presented show that all three of the feature extractors can be made to perform well in the test environments given suitable choice of matching parameters, even when the change in image viewpoint reaches 750 mm translation or 25° rotation. However, distinctions can still be made in terms of performance. In conditions where images are taken with only a small change in viewpoint, KLT with tracking outperforms SIFT and Harris by achieving much higher recall due to the tracking algorithm. As distance between images increases, SIFT and Harris achieve greater performance than the KLT due to the larger distances between corresponding features causing KLT tracking to fail. However, KLT performance degrades at a slower rate than the other extractors, which indicates a robustness towards tuning parameters. It is also interesting to note that performance of the two datasets is similar even though they contain features at very different depths, which indicates that these conclusions hold at different scales.

Chapter 5

Performance Evaluation at the SLAM Stage

Ideally, SLAM performance would be assessed directly by comparing measured robot pose and landmark positions to the SLAM state vector estimate. Measuring a ground truth is difficult, especially in arbitrary environments. In the local office-style environment, it is possible to measure robot pose by tracking points on the ceiling (see Appendix C), but it is infeasible to measure landmark positions. With these constraints in mind, a method of evaluation is designed using only the robot pose. This chapter presents two contributions: a methodology for evaluating SLAM performance and discussion of metric choice, and the experimental results of a comparative study using this methodology to evaluate three feature extractors.

5.1 Metric Choice

Any filter, whether used for SLAM or not, should produce estimates that are *consistent* such that the estimates are compatible with ground truth. However, testing for consistency does not provide a ranking between different estimates, since estimates are either consistent or not. An additional metric must be found that provides a means of comparing the performance between estimates generated, in this case, using different feature extractors. In the methodology described later, consistency testing is used as a verification step before a comparative metric is applied to create a ranking.

5.1.1 Consistency Testing

Successful SLAM runs should be consistent in the sense that the "state errors should be zero-mean (unbiased) and compatible with the covariance yielded by the filter" [7, p. 71]. Consistency is tested by calculating the *normalized estimation error squared* (NEES)

$$\epsilon_k = \left(\mathbf{r}_k - \hat{\mathbf{r}}_k\right)^T \Sigma_{r,k}^{-1} \left(\mathbf{r}_k - \hat{\mathbf{r}}_k\right) \tag{5.1}$$

using the SLAM estimated robot pose $\mathbf{r}_k$, ground truth robot pose $\hat{\mathbf{r}}_k$, and estimated pose covariance $\Sigma_{r,k}$ [4, 7]. With the hypothesis of a consistent filter with correct assumptions of Gaussianity and

linearity, ϵ_k follows a χ^2 distribution with degrees of freedom equal to the dimensionality of $\mathbf{r}_k$.

A single trial does not yield enough information to determine if the system produces consistent results (a single run of an inconsistent filter may generate consistent results, and vice versa, depending on the environment) [4]. Instead, the average NEES value $\bar{\epsilon}_k$ is considered, calculated from N Monte Carlo runs of the filter

$$\bar{\epsilon}_k = \frac{1}{N} \sum_{i=1}^{N} \epsilon_k^{(i)} \tag{5.2}$$

where $\epsilon^{(i)}$ is the NEES for trial i. The robot pose predicted by the motion model and observation predicted by the observation model are randomly sampled according to the (Gaussian) robot pose covariance and innovation covariance, respectively. The χ^2 acceptance test from [7] is then used to check the hypothesis H_0 that the system errors are consistent with the estimated covariance. This hypothesis is accepted if the average NEES values lie within a confidence interval $\bar{\epsilon}_k \in [r_1, r_2]$, where the interval is calculated such that

$$P\{\bar{\epsilon}_k \in [r_1, r_2] | H_0\} = 1 - \alpha \,. \tag{5.3}$$

where α is a small number such as 5%. This defines a region for which NEES values are consistent. Values below the lower bound are conservatively inconsistent, meaning the estimated covariance is compatible with the estimation error, but could be made smaller and remain consistent. Above the upper bound, values become optimistically inconsistent, meaning that the error is outside a reasonable region defined by the estimated covariance.

5.1.2 Comparative Metric

A useful metric for comparing performance should consider the state error and uncertainty and allow for a quantitative ranking of estimates obtained with the different feature extractors. This section discusses the shortcomings of some possible comparative metrics before describing a novel metric that satisfies the desired properties.

Accumulated error is an obvious metric for comparing performance, as minimizing the magnitude of state error is part of the goal of a SLAM system, and the result is easily comparable. However, it does not account for the uncertainty in state estimates and is thus not directly applicable to probabilistic systems. In this application, the case of small absolute error but near-zero uncertainty that is inconsistent with ground truth should not be ranked more favorably than the case of large absolute error and large uncertainty that is consistent with ground truth. Accumulated error is therefore not appropriate.

To account for the uncertainty, error could be weighted by covariance, and normalized error (the NEES values) accumulated. However, it is difficult to rank performance based on accumulated NEES. The best performance is not the lowest score, since a zero score is obtainable by setting the covariance to infinity at all timesteps, which is obviously not a desirable solution. Instead, the best

performance would have to have consistent NEES values, but since NEES is classified in a binary manner (consistent or not) there is no concept of rank.

All things being equal, it is considered desirable to have less uncertainty, as the robot position becomes better known. Therefore, accumulating uncertainty rather than normalized error is chosen as a solution to the problem of choosing a metric. The best performance, ideally, is considered to be the system that remains consistent while obtaining the smallest uncertainty. The volume V of the ellipsoid that represents the estimated covariance matrix is considered to represent "uncertainty" at each timestep. V is found using the lengths of the principal axes r_i of the covariance, which are equivalent to the square roots of the eigenvalues λ_i, and is easily calculated using the determinant

$$V(\Sigma_r) = \frac{4}{3}\pi r_1 r_2 r_3 = \frac{4}{3}\pi\sqrt{\lambda_1\lambda_2\lambda_3} = \frac{4}{3}\pi\sqrt{\det(\Sigma_r)}\,. \tag{5.4}$$

Accumulated uncertainty is simply the sum of volumes over time, given N_s steps in a trial

$$AU = \sum_{k=1}^{N_s} V(\Sigma_{k,r})\,. \tag{5.5}$$

This statistic is calculated over the Monte Carlo runs, and the average of accumulated uncertainty

$$\overline{AU} = \frac{1}{N}\sum_{i=1}^{N} AU_i \tag{5.6}$$

is used to rank feature extractors, where the highest-ranked extractors will have the lowest accumulated uncertainty while remaining consistent.

5.2 Methodology

Images and odometry data are captured from a mobile robot with a single camera while it is driven through an environment. A ground truth is captured simultaneously. Data is captured for later offline processing so that multiple SLAM runs can be made on the same dataset.

After data acquisition, multiple Monte Carlo runs of the SLAM system are generated by randomly sampling the robot pose and innovation covariances at the prediction and observation steps, respectively. Evaluation is then carried out in two steps: the first to test estimate consistency, and the second to rank the feature extractors. Ideally, the consistency test should be used to reject feature extractors that result in inconsistent SLAM estimates. However, inconsistency in EKF-SLAM is a known problem [4], so inconsistent cases may still be considered if all feature extractors cause inconsistency. After rejecting based on consistency testing, the accumulated uncertainty metric is applied and used to rank feature extractors from lowest to highest uncertainty.

5.3 Experimental Setup

An ActivMedia Pioneer P3-AT mobile robot was used to record data for offline processing. A Dragonfly IEEE-1394 camera from Point Grey Research captures images at a resolution of 640×480

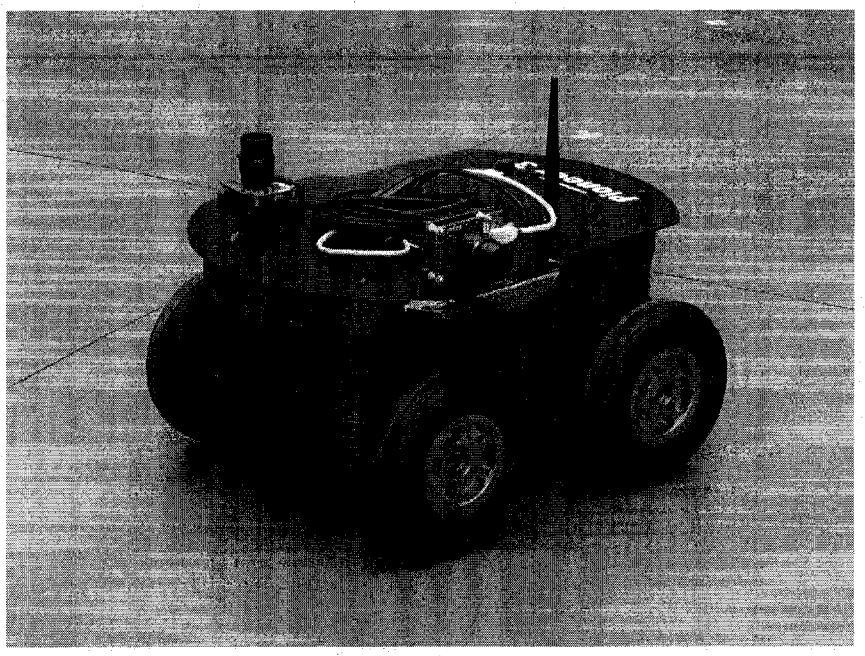


Figure 5.1: ActivMedia Pioneer P3-AT with Point Grey Research Dragonfly IEEE-1394 cameras. One camera points to the left of the robot and is used for SLAM observations, while the other is used for tracking the ceiling to generate a ground truth robot path.

pixels using Bayer encoding. Images are Bayer decoded by downsampling, resulting in a final image size of 320×240 pixels. Odometry data from the robot is recorded at the same time an image is captured. Taking advantage of the structured nature of the environment, ground truth is generated with a ceiling tracker that uses the regular grid of ceiling tiles to calculate robot pose, detailed in Appendix C. The hardware setup is shown in Figure 5.1.

The dataset is generated by driving the robot on the second floor of the Computing Science Centre at the University of Alberta. This environment represents a typical institutional foyer setting, shown in Figure 5.2. Data is recorded after every 4cm translation or 4° rotation, whichever is encountered first. A C-shaped path is taken by the robot, as shown in Figure 5.3. It is designed such that during the last leg of the path the robot re-observes the initial part of the environment. Although this is not loop closure in the sense of returning to the starting position, it is a type of closure since the initial features are visible. This is important as loop closure is a key test of a SLAM system, since it is only through re-observing previous features that pose covariance is reduced. Additionally, this path configuration allows for the evaluation of feature extractors when a scale change occurs, as the robot observes the same space at varying depth.

The SLAM system described in Chapter 3 is used to process the recorded data. The motion model in Section 3.3.1 is calibrated with two different parameter sets, one optimistic and one conservative. Of the motion model parameters, α_1 and α_3 are considered to be *direct* parameters, since they increase the variance in each odometry variable (ψ_1, d_x, ψ_2) directly based on that variable, and α_2 and α_4 to be *indirect* parameters, since they increase the variance in each odometry variable

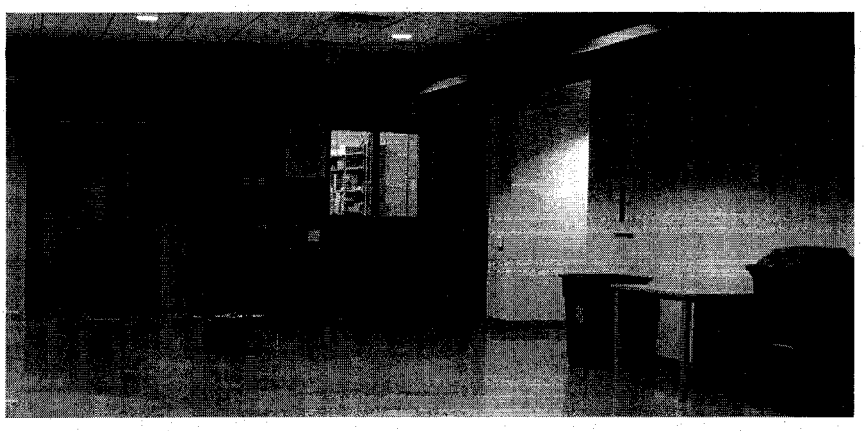


Figure 5.2: Robot in experimental environment. The setting is an elevator foyer in the Computing Science Centre at the University of Alberta.

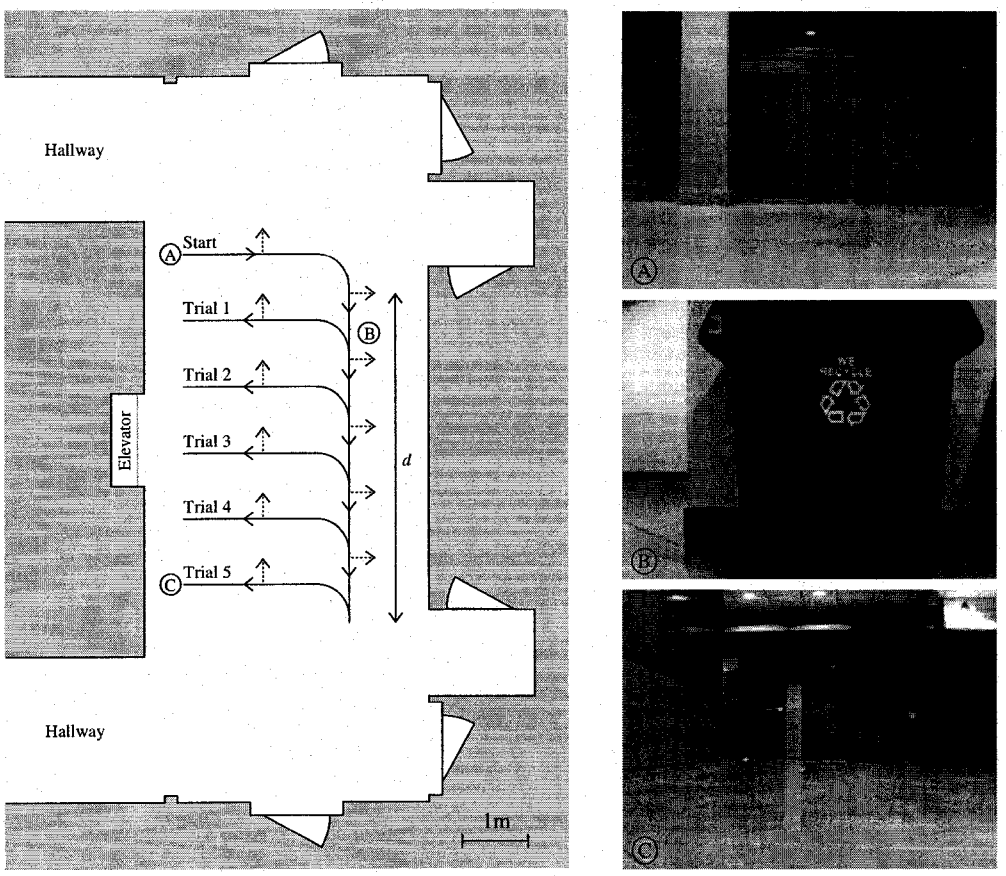


Figure 5.3: Experimental environment consisting of an elevator foyer surrounded by laboratory rooms. Dimensions are approximate. Robot starts at point Ⓐ, drives forward, turns through a 90° arc to the right, drives downwards a distance d that varies from one to five metres, passing point Ⓑ. It then reverses through a 90° arc and drives in reverse to point Ⓒ. The front of the robot always faces to the right or down and the camera is mounted 90° counter-clockwise from the front of the robot, so it faces up or right. Camera direction is indicated by the dashed arrow, while direction of motion is indicated by the solid arrowheads. During the final section when reversing to point Ⓒ, the robot moves left while facing to the right.

Model	α_1	α_2	α_3	α_4
Optimistic	1.111×10^{-3}	1.126×10^{-6}	1.111×10^{-3}	1.097×10^{-4}
Conservative	1.000×10^{-2}	1.013×10^{-5}	1.000×10^{-2}	9.870×10^{-4}

Table 5.1: Parameters for the odometry motion model, in both *optimistic* and *conservative* configurations. See text for parameter calculation, Section 3.3.1 for model description.

f_x [pix]	f_y [pix]	u_0 [pix]	v_0 [pix]	k_1	k_2
342.0 ± 1.3	345.5 ± 1.5	161.4 ± 2.8	133.9 ± 2.2	-0.383 ± 0.014	0.226 ± 0.061

Table 5.2: Observation model parameters with approximate standard deviations. Calibrated using [11]. See Section 3.4.1 for model description.

based on a different variable. The optimistic parameter set is derived such that direct parameters result in variance equal to of 3% of the odometry variables ψ_1, d_x, ψ_2 (equivalently, the 3σ or 99.7% confidence limit is equal to 10% of the odometry variables), while indirect parameters contribute $1/3\%$ to the variance. The conservative parameter set is derived in a similar manner, instead using setting 1σ or 68% confidence limits to 10% and 1% for direct and indirect. The values used are given in Table 5.1. Using the two models allows us to examine the effect of motion model parameters on SLAM results. The observation model in Section 3.4.1 is calibrated using an implementation of Zhang's camera model calibration method [11], and parameters are given in Table 5.2.

Three feature extractors described earlier were used in the SLAM system: SIFT with a 128 element descriptor, the Harris corner detector with an 11×11 image patch descriptor, and the KLT tracker with an 11×11 image patch descriptor. Nearest-neighbor with distance-ratio matching was used in preference to normalized cross-correlation since it was found to produce better results in Chapter 4.

5.4 Results and Discussion

The SLAM system was run offline on recorded data. Five trials were captured (Trials 1–5), where the trial number corresponds to the scale-change depth d in metres shown in Figure 5.3. Fifty Monte Carlo runs were performed for every trial using each feature extractor, and the average NEES, uncertainty, and accumulated uncertainty at each time step were calculated. Full results are displayed in Appendix D. The remainder of this chapter discusses key conclusions drawn from the results, with supporting figures drawn from the Appendix. In addition to the two steps described in the methodology above (consistency testing and accumulated uncertainty ranking), the effects of the two motion models and the scale change between Trials on SLAM performance with different feature extractors is considered.

5.4.1 Consistency and Divergence

Consistency is measured by examining the NEES plots for each trial (Figures D.1–D.5, (a)–(b)), an example of which is shown in Figure 5.4. Given the robot pose state size $n_x = 3$ and number of runs $N = 50$ in each trial, the NEES consistency interval is calculated to be $[2.36, 3.72]$. This interval is shown as dashed lines on the NEES plots.

Examining the average NEES values for all trials, it is seen that the system remains within or below the consistency region during Trial 1 (Figure D.1(a)–(b)), rises above the consistency region to become optimistic in Trial 2 (Figure D.2(a)–(b)), and finishes well above the consistency region in Trials 3-5 (Figures D.3–D.5, (a)–(b)). Ideally, values should remain within the consistency bounds, although a conservatively inconsistent estimate is considered acceptable since the robot pose and ground truth are still "compatible." All trials initially exhibit conservative inconsistency due in part to constant uncertainty added to the robot pose covariance to account for measurement error in the ceiling tracker used to measure ground truth. Additionally, all extractors follow the same general trends in the trials with some individual deviation, so it seems there is no advantage in terms of consistency to choosing a particular extractor.

Inconsistency in EKF-SLAM systems is a known problem that stems from the required linearization of non-linear models [4, 25]. Interested readers are directed to proposed solutions outside the scope of this thesis that achieve some success in minimizing this source of error [3].

Since divergence is inevitable over time, it is expected that shorter runs (Trials 1–2) remain consistent or conservative while longer runs (Trials 3-5) tend to diverge. Conclusions drawn in the following sections regarding the relative performance of feature extractors are less strong than if all trials were consistent, however it is quite possible to make statements about the relative performance of feature extractors within the limits of the experimental setup.

5.4.2 Motion Model Effects

Comparing SLAM results obtained using the two different motion model parameter sets, it can be seen that using the *optimistic* model results in lower uncertainty (Figures D.1–D.5, (e)–(f)) and accumulated uncertainty (Figures D.1–D.5, (c)–(d)). An example using SIFT showing both models on the same plot is given in Figure 5.5, with other cases yielding similar results. This behavior is expected since the *optimistic* model injects less uncertainty into the system at each prediction step.

In most cases of all trials, using the two different models results in very similar NEES values. An example from Trial 4 is shown in Figure 5.6. The difference is not particularly significant except in a few cases (SIFT in Trial 3, KLT in Trial 4). Additionally, Trial 2 is a transition point of sorts; Trial 1 and Trial 2 with the *optimistic* model result in mainly conservatively inconsistent or consistent NEES, while *conservative* Trial 2 and Trials 3–5 are optimistically inconsistent. The magnitude of pose uncertainty affects the extent to which the EKF will allow observations to "push" the pose away from the odometry prediction, minimizing the amount of correction performed when pose is already

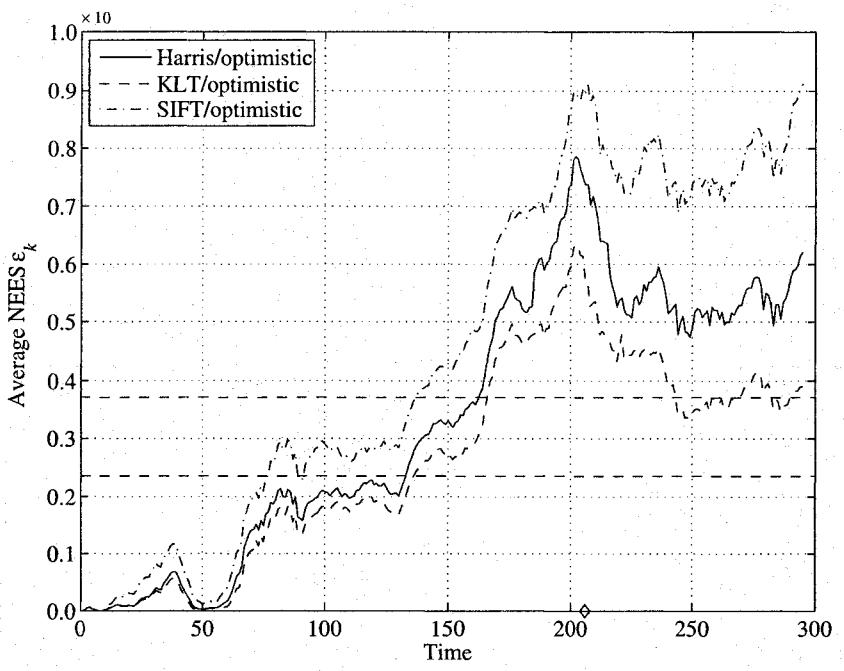


Figure 5.4: Average NEES values from Trial 3 with the *optimistic* motion model. Dashed lines indicate the 95% confidence bounds that demarcate the region in which the estimate is considered consistent, and ◇ indicates when the robot was able to re-observe the initial section of the environment. Initially the system is conservatively inconsistent, then passes through the consistent region, with optimistic inconsistency beginning shortly after the first corner in the robot path, between $t = 140$ and $t = 160$. Although the system recovers somewhat, it remains inconsistent at termination with all three extractors. Also appears as Figure D.3b.

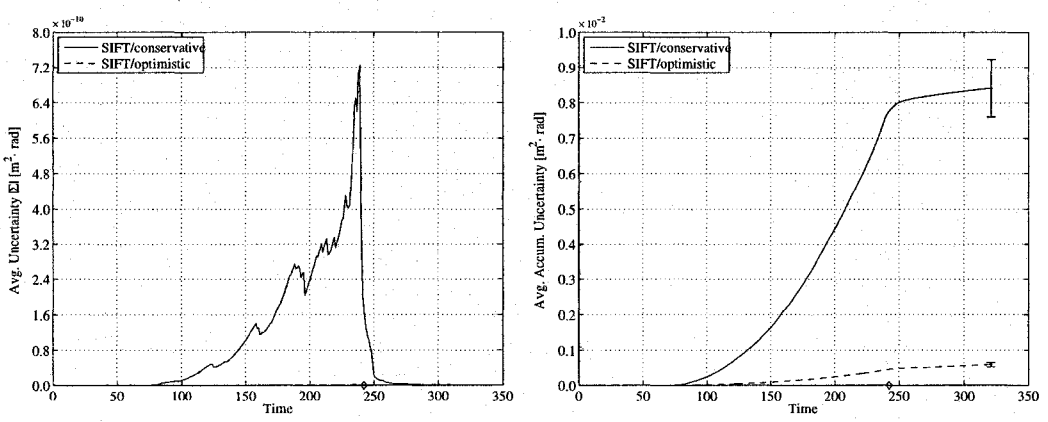


Figure 5.5: Comparison of SLAM system average uncertainty and average accumulated uncertainty with 1σ interval during Trial 4 using SIFT with two different motion models. The ◇ indicates when the robot was able to re-observe the initial section of the environment. Less uncertainty is injected by the *optimistic* model at each time step, resulting in lower accumulated uncertainty, as is expected. A sharp drop around $t = 250$ corresponds to re-observation of initial features and indicates a loop closure of sorts. Derived from Figure D.4 (c)–(f).

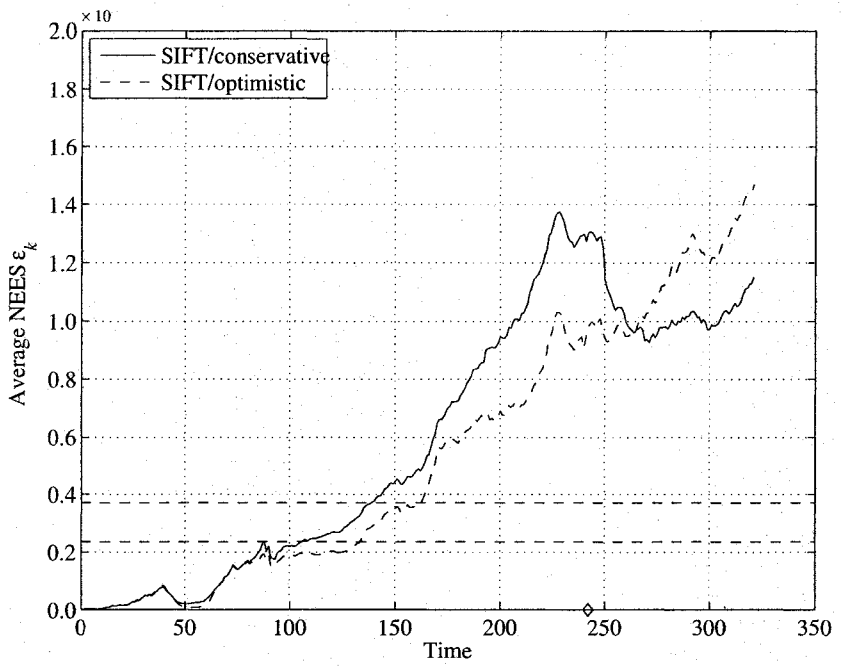


Figure 5.6: Average NEES with 95% confidence limits for SIFT in Trial 4 using different motion models, representative of other trials and extractors. The ◇ indicates when the robot was able to re-observe the initial section of the environment. Although there is some difference, both models follow the same general trend. Derived from Figure D.4 (a)–(b).

very certain. Since the *optimistic* model adds less uncertainty, robot pose is more constrained than with the *conservative* model. This could lead to problems if an optimistic model is used with a robot with poor odometry, however the Pioneer odometry is very accurate. Different behavior is expected on a robot with poorer odometry.

With the exception of SIFT in Trial 3, runs that become optimistically inconsistent transition to inconsistency around the same timestep regardless of motion model, as seen in Figure 5.6 and Figures D.3–D.5 (a)–(b). This further suggests that while the motion model has a role in determining the magnitude of uncertainty present in the SLAM system, the general behavior is not significantly affected, except in two cases out of fifteen. As such, the conclusion is drawn that in these experiments the choice of motion model is not important, although the *optimistic* model allows for slightly better performance in Trial 2.

5.4.3 Effect of Scale Change

SLAM performance will degrade according to how well the extractor handles scale change. Poor extractors will fail to make matches after a scale change, which limits the extent to which SLAM can reduce pose uncertainty. The C-shaped path followed by the robot is designed to test how well the extractors handle scale change by varying the change in depth at which features are re-observed from one to five metres.

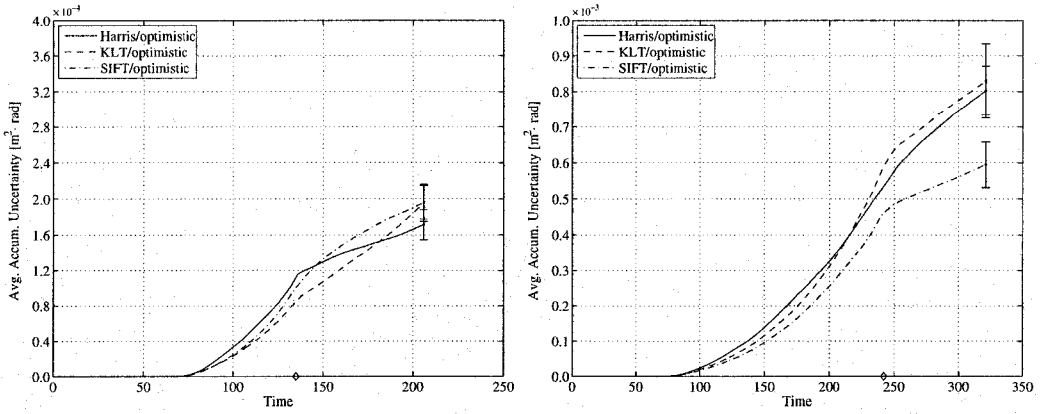
From the average uncertainty it can be seen that in all trials with all extractors there is a large reduction in uncertainty at the point corresponding to where the robot turns to face the initial direction. Since accumulated uncertainty is the sum of uncertainty, this also appears in the accumulated uncertainty figures as a decrease in slope. An example of this is shown in Figure 5.5. This indicates that earlier features are re-observed, since re-observation is the only way the SLAM observation step can reduce uncertainty. All three extractors yield similar performance, suggesting that scale does not have a significant influence in this environment and that a scale-invariant extractor like SIFT is not necessarily required. Constraints and clutter in typical indoor environments will often mean that features are not re-observed from a viewpoint more than five metres away, so these experiments should generalize to similar scenarios.

5.4.4 Difference Between Extractors in terms of Accumulated Uncertainty

Determining the extent to which choice of feature extractor affects the average accumulated uncertainty is the key focus of this chapter. It can be seen from Figures D.1–D.5 (c)–(d) that in many cases the average accumulated uncertainty for one extractor will lie within one standard deviation of another, and vice versa. An example is shown in Figure 5.7a. For all other cases, the average accumulated uncertainty of each extractor all lie within the same order of magnitude, an example of which is Figure 5.7b. While this does not in itself imply statistical insignificance, it is clear that SLAM performance with different extractors does not drastically vary. It appears instead that the choice is "lost in the noise" among the multitude of parameters and choices involved in designing the components of a visual SLAM system. As such, the system as a whole must be analysed to determine optimal choices and settings to maximize performance. Additionally, when the extractor is varied while holding trajectory and motion model constant, the NEES plots follow the same trend in almost all cases. This shows that SLAM consistency is largely independent of extractor choice.

If a ranking of accumulated uncertainty performance based strictly on extractor choice is desired, SIFT yields the lowest average accumulated uncertainty in nine of ten cases, although the standard deviation for SIFT may overlap with other extractors. In these nine cases, KLT is the second lowest five times, and Harris four. Since Harris and KLT are based on similar principles, it is not surprising that the results are very similar. Trial 1 with the *optimistic* model is the single anomalous case, with Harris yielding the lowest average accumulated uncertainty, with SIFT and KLT producing nearly identical values. It should be noted that there is significant overlap of standard deviations.

Because average accumulated uncertainty results are not significantly different in all cases, and NEES curves follow the same trend with different extractors as previously seen in Figure 5.6, these results show that any of the three extractors may be used to perform visual SLAM in similar close-range indoor environments.



(a) Trial 1 (b) Trial 4

Figure 5.7: Average accumulated uncertainty with 1σ intervals using the *optimistic* motion model on the indicated trials. The $\diamond$ indicates when the robot was able to re-observe the initial section of the environment. In (a), the standard deviations of different extractors overlap, implying a form of statistical insignificant. In (b), although there is no overlap, results are still within the same order of magnitude, which indicates that although statistically there is some difference, for practical purposes there is little advantage of one extractor over another. Also appears as Figures D.1b and D.4b.

5.5 Summary

This chapter described a methodology for comparing the performance of a SLAM system using several different feature extractors and presented an experimental study using this methodology to test three common extractors. The methodology consists of two parts: consistency testing using the average NEES values to determine if estimates are compatible with ground truth, and creating a ranking of those feature extractors that remain consistent using the novel average accumulated uncertainty metric.

A dataset was collected on a mobile robot in a typical indoor environment, with a ground truth estimate provided by a ceiling tracker. Odometry was captured for control input, with two different motion model parameter sets. The visual SLAM system described in Chapter 3 was used to process the data and the NEES and accumulated uncertainty values were calculated as described above. From the results obtained, four main conclusions have been presented:

1. Over time the SLAM system becomes optimistically inconsistent. This agrees with others prediction and simulation [4, 25] and stems from the linearization inherent to the extended Kalman filter, rather than effects of the feature extractors. Although in the ideal case this should cause the rejection of all three feature extractors by the consistency test, further analysis was performed since all feature extractors were affected equally, so no particular one has an advantage in the ranking.

2. Motion model parameters play a role in determining the *magnitude* of robot pose uncertainty, but the two parameter sets tested have little effect on general system trend and behavior.

3. Scale change does not hinder SLAM in the environment used for these experiments. In all cases with all feature extractors the system is able to reacquire previous landmarks after a change in depth of five metres. If this depth were dramatically increased, a scale-invariant feature extractor may be required.

4. No significant difference exists between feature extractors in terms of average accumulated uncertainty in the experimental environment.

These points lead to the final conclusion that choice of feature extractor is not critical, as the feature extractors tested all yield similar results. Other criteria or constraints in a particular system can therefore be used to choose a feature extractor. A rotationally-invariant feature extractor may be required for a robot with more degrees of freedom for example, or an embedded system may necessitate a computationally inexpensive feature extractor. These conclusions apply to test environments similar to that presented here, and natural or very large environments may invalidate the claims.

Chapter 6

Conclusion

This thesis has presented a framework for evaluating the performance of feature extractors in the context of visual SLAM. Choice of feature extractor is a key concern when implementing a visual SLAM system, however no work exists that compares extractors and suggests usage guidelines. The framework was used to provide the first comparison of feature extractors commonly used in SLAM systems. Evaluation was performed at two stages: directly after feature extraction, and after SLAM processing. The former removes dependence on SLAM implementation, while the latter directly measures the desired performance. At the feature stage, matches between images in sequence were verified using the fundamental matrix, and the recall and precision metrics were used to compare performance. After SLAM processing, performance was determined by consistency testing and the metric of accumulated uncertainty. A SLAM system was built to support the experiments.

Experiments were conducted at both stages on two mobile robots in a typical indoor environment. Three commonly used feature extractors were used: the Harris corner detector, the Kanade-Lucas-Tomasi tracker (KLT), and Lowe's Scale-Invariant Feature Transform (SIFT). After feature extraction, it was found that all three extractors could be made to perform in the acceptable recall and precision regime. A second experiment was performed with subsampling of the image sequence, yielding similar results even when image viewpoints were spaced up to 750 mm or 25° apart. Results were similar for two datasets containing features at different depths. Although the difference is small, KLT was seen to outperform SIFT and Harris by achieving the highest recall. After full SLAM processing, it was found that the SLAM system estimate became inconsistent over longer trials, an effect known from the literature. Despite this, relative comparisons of extractors were possible. Results show that while there are small differences between the accumulated uncertainty of different extractors, for practical purposes performance is identical. Additionally, all extractors were able to re-observe initial features after a change of depth from one to five metres. This suggests that for smaller-scale indoor environments, extractor choice is not important. Examining the small differences, SIFT comes out slightly ahead in most trials.

The final conclusion drawn from these experiments is that all three of the extractors can be made to work well in similar environments. This is supported by evaluation at the feature stage and

SLAM stage, as results from both agree with each other. These sorts of environments comprise a large part of the domain that mobile robots work in today, making this a valuable result. Another key observation is that a good SLAM system is able to reject many spurious matches and ensure that accepted matches are consistent with the current state estimate. This innovation gating, described in Section 3.5, acts to negate the effects of poor feature extraction and matching, and greatly reduces the influence of extractor choice. In light of this, it is hypothesized that other feature extractors will perform similarly in a visual SLAM system. The choice is "lost in the noise" of SLAM system design and parameter choice.

Since feature extractor choice is not critical to determining SLAM performance, other criteria may dictate the choice. If very large scale changes are present in a robot's trajectory, SIFT may be necessary. If the robot has additional degrees of freedom that result in rotating the optical axis, a rotationally-invariant descriptor such as SIFT is required. For embedded systems, a less computationally expensive extractor such as Harris is appropriate. KLT was shown to produce higher recall when viewpoint change is small, and may be appropriate for systems with a slow-moving robot or a very fast processor.

Three main contributions are presented in this thesis. First, previous feature stage performance evaluation work is extended to apply the recall and precision metrics to non-planar scenes in a video sequence. Second, accumulated uncertainty is introduced as a means to compare SLAM performance. This offers a metric that goes beyond consistency testing, which is unable to rank performance. Third, the first evaluation of common feature extractors in the context of visual SLAM at both the feature stage and after SLAM processing is presented, the results of which are valuable to those researching or implementing these systems. Finally, an orthogonal contribution to the main focus of performance evaluation is the feature initialization technique of Section 3.6. This provides a simple and efficient technique using triangulation and the unscented transform. More detail and comparison to other feature initialization techniques can be found in [30, 47].

6.1 Future Directions

A number of options and ideas for extension presented themselves during the course of working on this research, only to be limited by available time and resources. The major points are recorded here.

The most straightforward extension is to evaluate more feature extractors. While the three used here are the most common, researchers are beginning to apply feature extractors such as Harris-Affine [36], a scale-invariant version of the Harris detector, and SURF [9] to robot localization and mapping problems. It would be interesting to see if these verify the general conclusion presented here that extractor choice does not play a critical role in SLAM performance. Additionally, it would be interesting to use multiple heterogeneous extractors concurrently to see if the additional information helps SLAM performance.

Performance at the SLAM stage was measured using the accumulated uncertainty of robot pose.

In this case it was too time consuming to measure landmark positions, however there is nothing in the theory that forbids the accumulated uncertainty from including the entire state. In fact, the metric should be more accurate and complete if the full state uncertainty is considered for consistency and accumulation. Practically this is only difficult because knowledge of landmark positions is needed, requiring either a calibrated, known environment, or the manual measurement of all landmarks. Some care is also required when accumulating uncertainties with different numbers of landmarks.

While the metric of accumulated uncertainty was introduced to compare the performance of a single SLAM system with different feature extractors, there is no reason not to apply it to other situations. For example, it could be used to perform comparisons of different feature initialization or map management methods. It should also be possible to derive an equivalent metric in the case of the non-Gaussian representation present in particle filter algorithms.

Bibliography

[1] S. Agarwal and D. Roth. Learning a Sparse Representation for Object Detection. In *Proc. of the European Conference on Computer Vision*, pp. 113–130, 2002.

[2] T. Bailey. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. Ph.D. thesis, Austrailian Centre for Field Robotics, University of Sydney, August 2002.

[3] T. Bailey and H. Durrant-Whyte. Simultaneous Localization and Mapping (SLAM): Part II, State of the Art. *IEEE Robotics and Automation Magazine*, 13(3) pp. 108–117, September 2006.

[4] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM Algorithm. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing, China, 2006.

[5] T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM Algorithm. In *Proc. of IEEE International Conference on Robotics and Automation*. Orlando, USA, May 2006.

[6] S. Baker and I. Matthews. Lucas-Kanade 20 Years on: A Unifying Framework. *International Journal of Computer Vision*, 56(3) pp. 221–255, February 2004.

[7] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*, vol. 179 of *Mathematics in Science and Engineering*. Academic Press, Orlando, USA, 1988.

[8] T. D. Barfoot. Online Visual Motion Estimation using FastSLAM with SIFT Features. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*. Edmonton, Canada, 2005.

[9] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *Proc. of Ninth European Conference on Computer Vision*. Graz, Austria, 2006.

[10] S. Birchfield. KLT: An Implementation of the Kanade-Lucas Tomasi Feature Tracker. [Online] Available: `http://www.ces.clemson.edu/~stb/klt/`. Accessed February 2006.

[11] J.-Y. Bouget. Camera Calibration Toolbox for Matlab. [Online] Available: `http://www.vision.caltech.edu/bouguetj/calib_doc/`. Accessed January 2006.

[12] A. J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proc. of International Conference on Computer Vision*. Nice, France, October 2003.

[13] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), June 2007.

[14] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3), June 2001.

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

[16] L. Goncalves, E. Di Bernardo, D. Benson, M. Svedman, J. Ostrowski, N. Karlsson, and P. Pirjanian. A Visual Front-end for Simultaneous Localization and Mapping. In *Proc. of IEEE International Conference on Robotics and Automation*. Barcelona, Spain, April 2005.

[17] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice-Hall, Upper Saddle River, New Jersey, third ed., 2008.

[18] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proc. of Fourth Alvey Vision Conference*, pp. 147–151. Manchester, United Kingdom, 1988.

[19] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2) pp. 146–157, 1997.

[20] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[21] J. Heikkilä and O. Silvén. A Four-step Camera Calibration Procedure with Implicit Image Correction. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1106–1112. San Juan, Puerto Rico, 1997.

[22] A. Howard and N. Roy. The Robotics Data Set Repository (Radish), 2003. [Online] Available: `http://radish.sourceforge.net`.

[23] Intel Corporation. Open Source Computer Vision Library (OpenCV). [Online] Available: `http://www.intel.com/technology/computing/opencv/`. Accessed October 2007.

[24] S. Julier and J. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, 1997.

[25] S. J. Julier and J. K. Uhlmann. A Counter Example to the Theory of Simultaneous Localization and Map Building. In *Proc. of IEEE International Conference on Robotics and Automation*, pp. 4238–4234. Seoul, Korea, May 2001.

[26] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich. The vSLAM Algorithm for Robust Localization and Mapping. In *Proc. of IEEE International Conference on Robotics and Automation*. Barcelona, Spain, April 2005.

[27] Y. Ke and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[28] G.-H. Kim, J.-S. Kim, and K.-S. Hong. Vision-based Simultaneous Localization and Mapping with Two Cameras. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*. Edmonton, Canada, August 2005.

[29] J. Klippenstein and H. Zhang. Quantitative Evaluation of Feature Extractors for Visual SLAM. In *Proc. of Fourth Canadian Conference on Computer and Robot Vision*. Montreal, Canada, May 2007.

[30] J. Klippenstein, H. Zhang, and X. Wang. Feature Initialization for Bearing-Only Visual SLAM Using Triangulation and the Unscented Transform. In *Proc. of IEEE International Conference on Mechatronics and Automation*. Harbin, China, August 2007.

[31] P. D. Kovesi. MATLAB and Octave Functions for Computer Vision and Image Processing. School of Computer Science and Software Engineering, The University of Western Australia. [Online] Available: `http://www.csse.uwa.edu.au/~pk/research/matlabfns/`. Accessed February 2006.

[32] T. Lindeberg. Scale-space Theory: A Basic Tool for Analyzing Structures at Different Scales. *Journal of Applied Statistics*, 21(2) pp. 224–270, 1994.

[33] D. G. Lowe. SIFT Keypoint Detector. [Online] Available: `http://www.cs.ubc.ca/~lowe/keypoints/`. Accessed April 2006.

[34] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2) pp. 91–110, 2004.

[35] P. Maybeck. *Stochastic Models, Estimation, and Control*, vol. 141 of *Mathematics in Science and Engineering*. Academic Press, New York, 1979.

[36] K. Mikolajczyk and C. Schmid. Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1) pp. 64–86, 2004.

[37] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10) pp. 1615–1630, October 2005.

[38] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1/2) pp. 43–72, 2005.

[39] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping. In *Proc. of International Joint Conference on Artificial Intelligence*. Acapulco, Mexico, August 2003.

[40] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of Interest Point Detectors. *International Journal of Computer Vision*, 37(2) pp. 151–172, 2000.

[41] D. Shaw and N. Barnes. Regular Polygon Detection as an Interest Point Operator for SLAM. In *Australasian Conference on Robotics and Automation*, 2004.

[42] J. Shi and C. Tomasi. Good Features to Track. In *IEEE Conference on Computer Vision and Pattern Recognition*. Seattle, 1994.

[43] R. Sim, P. Elinas, M. Griffin, and J. J. Little. Vision-based SLAM using the Rao-Blackwellised Particle Filter. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*. Edinburgh, Scotland, 2005.

[44] R. Smith, M. Self, and P. Cheeseman. Estimating Uncertain Spatial Relationships in Robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, pp. 167–193. Springer-Verlag, New York, 1990.

[45] S. Thrun. Robotic Mapping: A Survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[46] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, USA, 2005.

[47] X. Wang and H. Zhang. Bearing-Only Landmark Initialization by Using SUF with Undistorted SIFT Features. In *Proc. of IEEE International Conference on Robotics and Automation*, 2006.

[48] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Tech. Rep. TR 95–041, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[49] Z. Zhang. A Flexible New Technique For Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11) pp. 1330–1334, 2000.

Appendix A

Jacobian Matrices

A.1 Prediction Jacobians

The Jacobian matrix F_x of the prediction function $\mathbf{f}$ from Section 3.3.1 with respect to the state $\mathbf{x}$ can be calculated as

$$F_x = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_r}{\partial \mathbf{r}} & 0 \\ 0 & I \end{bmatrix}, \qquad \frac{\partial f_r}{\partial \mathbf{r}} = \begin{bmatrix} 1 & 0 & -d_x \sin(\theta + \psi_1) \\ 0 & 1 & d_x \cos(\theta + \psi_1) \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.1}$$

Since noise in the motion model is modeled as noise in the control vector $\mathbf{u}$, the Jacobian matrix F_u of the prediction function $\mathbf{f}$ with respect to noise as the Jacobian taken with respect to the control vector $\mathbf{u}$ is derived as

$$F_u = \frac{\partial f}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial f_r}{\partial \mathbf{u}} \\ 0 \end{bmatrix}, \qquad \frac{\partial f_r}{\partial \mathbf{u}} = \begin{bmatrix} -d_x \sin(\theta + \psi_1) & \cos(\theta + \psi_1) & 0 \\ d_x \cos(\theta + \psi_1) & \sin(\theta + \psi_1) & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{A.2}$$

A.2 Observation Jacobians

Jacobians of the observation function $\mathbf{h}$ from Section 3.4.1 are more involved than the prediction Jacobians, and simpler to solve in multiple steps. Calculate H_x by first expanding with the chain rule (the subscript on map landmark $\mathbf{m}_i^{\mathcal{W}}$ is dropped for convenience)

$$H_x = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \frac{\partial \mathbf{h}}{\partial \mathbf{m}^{\mathcal{C}}} \frac{\partial \mathbf{m}^{\mathcal{C}}}{\partial \mathbf{x}} = \frac{\partial \mathbf{h}}{\partial \mathbf{m}^{\mathcal{C}}} \frac{\partial \mathbf{m}^{\mathcal{C}}}{\partial \mathbf{m}^{\mathcal{R}}} \frac{\partial \mathbf{m}^{\mathcal{R}}}{\partial \mathbf{x}}. \tag{A.3}$$

The first term of Equation (A.3) consists of the derivatives of the $\mathbf{h}$ function, Equation (3.12) (let $\mathbf{m}^{\mathcal{C}} = (x, y, z)$ instead of (m_x, m_y, m_z) for convenience)

$$\frac{\partial \mathbf{h}}{\partial \mathbf{m}^{\mathcal{C}}} = \frac{1}{z} \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix} \tag{A.4}$$

where the components, using $r^2 = (x^2 + y^2)/z^2$, are

$$\begin{aligned}
h_{11} &= f_x \left(1 + k_1 \frac{3x^2 + y^2}{z^2} + k_2 r^2 \frac{5x^2 + y^2}{z^2}\right) \\
h_{12} &= f_x \frac{2xy}{z^2} \left(k_1 + 2k_2 r^2\right) \\
h_{13} &= -f_x \frac{x}{z} \left(1 + 3k_1 r^2 + 5k_2 r^4\right) \\
h_{21} &= f_y \frac{2xy}{z^2} \left(k_1 + 2k_2 r^2\right) \\
h_{22} &= f_y \left(1 + k_1 \frac{x^2 + 3y^2}{z^2} + k_2 r^2 \frac{x^2 + 5y^2}{z^2}\right) \\
h_{23} &= -f_y \frac{y}{z} \left(1 + 3k_1 r^2 + 5k_2 r^4\right) .
\end{aligned} \tag{A.5}$$

The second term of Equation (A.3) that results from transforming from $\mathcal{C}$ to $\mathcal{R}$ and is trivial

$$\frac{\partial \mathbf{m}^{\mathcal{C}}}{\partial \mathbf{m}^{\mathcal{R}}} = \frac{\partial}{\partial \mathbf{m}^{\mathcal{R}}} \left({}^{\mathcal{C}}R_{\mathcal{R}}\, \mathbf{m}^{\mathcal{R}} + \mathbf{t}^{\mathcal{C}}\right) = {}^{\mathcal{C}}R_{\mathcal{R}} \tag{A.6}$$

using the the fact that the transform ${}^{\mathcal{C}}T_{\mathcal{R}}$ can be expressed in three-dimensional non-homogeneous coordinates as

$$\mathbf{m}^{\mathcal{C}} = {}^{\mathcal{C}}R_{\mathcal{R}}\, \mathbf{m}^{\mathcal{R}} + \mathbf{t}^{\mathcal{C}} . \tag{A.7}$$

The third and final term of Equation (A.3) involves the transformation of landmark $\mathbf{m}$ to the robot frame using the world to robot transformation (inverse of Equation (3.1))

$${}^{\mathcal{R}}T_{\mathcal{W}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & -x\cos\theta - y\sin\theta \\ -\sin\theta & \cos\theta & 0 & x\sin\theta - y\cos\theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{\mathcal{R}}R_{\mathcal{W}}(\theta) & \mathbf{t}^{\mathcal{R}}(x, y, \theta) \\ 0 & 1 \end{bmatrix} \tag{A.8}$$

which is expressed in non-homogeneous coordinates as

$$\mathbf{m}^{\mathcal{R}}(\mathbf{r}, \mathbf{m}^{\mathcal{W}}) = {}^{\mathcal{R}}R_{\mathcal{C}}\, \mathbf{m}^{\mathcal{W}} + \mathbf{t}^{\mathcal{R}} . \tag{A.9}$$

The Jacobian with respect to state $\mathbf{x}$ is formed from a concatenation of smaller Jacobian matrices

$$\frac{\partial \mathbf{m}_i^{\mathcal{R}}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{m}_i^{\mathcal{R}}}{\partial x} & \frac{\partial \mathbf{m}_i^{\mathcal{R}}}{\partial y} & \frac{\partial \mathbf{m}_i^{\mathcal{R}}}{\partial \theta} & 0_{3\times 3} & 0_{3\times 3} & \cdots & \frac{\partial \mathbf{m}_i^{\mathcal{R}}}{\partial \mathbf{m}_i^{\mathcal{W}}} & \cdots \end{bmatrix} \tag{A.10}$$

where the ellipses are the (zero) Jacobians with respect to other landmarks in the map $\mathbf{m}$. All that remains is to fill in the parts of the above equation. The derivatives with respect to robot position $\mathbf{r}$ and world-space landmark position $\mathbf{m}^{\mathcal{W}}$ are

$$\begin{aligned}
\frac{\partial \mathbf{m}^{\mathcal{R}}}{\partial x} &= \frac{\partial \mathbf{t}^{\mathcal{R}}}{\partial x} = \begin{pmatrix} -\cos\theta \\ \sin\theta \\ 0 \end{pmatrix} , \quad \frac{\partial \mathbf{m}^{\mathcal{R}}}{\partial y} = \frac{\partial \mathbf{t}^{\mathcal{R}}}{\partial y} = \begin{pmatrix} -\sin\theta \\ -\cos\theta \\ 0 \end{pmatrix} \\
\frac{\partial \mathbf{m}^{\mathcal{R}}}{\partial \theta} &= \frac{\partial {}^{\mathcal{R}}R_{\mathcal{W}}}{\partial \theta} \mathbf{m}^{\mathcal{W}} + \frac{\partial \mathbf{t}^{\mathcal{R}}}{\partial \theta} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ -\cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{m}^{\mathcal{W}} + \begin{pmatrix} x\sin\theta - y\cos\theta \\ x\cos\theta + y\sin\theta \\ 0 \end{pmatrix} \\
\frac{\partial \mathbf{m}^{\mathcal{R}}}{\partial \mathbf{m}^{\mathcal{W}}} &= {}^{\mathcal{R}}R_{\mathcal{W}} \frac{\partial \mathbf{m}^{\mathcal{W}}}{\partial \mathbf{m}^{\mathcal{W}}} = {}^{\mathcal{R}}R_{\mathcal{W}} .
\end{aligned} \tag{A.11}$$

Since the noise in the observation function is modeled as noise in the camera parameters $\pi = (f_x, f_y, u_0, v_0, k_1, k_2)$, the Jacobian matrix H_π of h with respect to π, again letting the camera-space landmark position $\mathbf{m}^C = (x, y, z)$ is derived as

$$H_\pi = \frac{\partial \mathbf{h}}{\partial \pi} = \frac{1}{z}\begin{bmatrix} \left(1 + k_1 r^2 + k_2 r^4\right) x & 0 & z & 0 & r^2 f_x x & r^4 f_y y \\ 0 & \left(1 + k_1 r^2 + k_2 r^4\right) y & 0 & z & r^2 f_y y & r^4 f_y y \end{bmatrix} \tag{A.12}$$

where $r^2 = (x^2 + y^2)/z^2$.

A.3 Triangulation Transformation

Jacobian matrices of the transform from $\mathcal{R}_o$ back to $\mathcal{W}$ given by Equation (3.32) are required. The Jacobian of $\mathbf{g}$ with respect to the map landmark $\mathbf{m}^{\mathcal{R}_o}$ is the product of two rotation matrices

$$G_m = \frac{\partial \mathbf{g}}{\partial \mathbf{m}^{\mathcal{R}_o}} = {}^{\mathcal{W}}R_{\mathcal{R}_o} \,. \tag{A.13}$$

The Jacobian of $\mathbf{g}$ with respect to initial robot position $\mathbf{r}_0^{\mathcal{W}} = (x_0, y_0, \theta_0)$ takes derivatives of the world to robot transformation given by Equation (3.1)

$$\begin{aligned} G_r &= \frac{\partial \mathbf{g}}{\partial \mathbf{r}_0^{\mathcal{W}}} = \frac{\partial ({}^{\mathcal{W}}R_{\mathcal{R}_o})}{\partial \mathbf{r}_0^{\mathcal{W}}} \mathbf{m}^{\mathcal{R}_o} + \frac{\partial \mathbf{t}^{\mathcal{W}}}{\partial \mathbf{r}_0^{\mathcal{W}}} \\ &= \begin{bmatrix} \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 1} & G_{\theta_0} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ G_{\theta_0} &= \begin{bmatrix} -\sin\theta & -\cos\theta & 0 \\ \cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{m}^{\mathcal{R}_o} \,. \end{aligned} \tag{A.14}$$

Appendix B

Calculation of Field of View from Camera Parameters

Field-of-view (FOV) for a calibrated camera is determined easily from the intrinsic calibration parameters. Looking along the optical axis, the half-angles formed by the ray to an image point and the optical axis are:

$$\begin{aligned} \tan\left(\frac{\theta_x}{2}\right) &= \frac{x}{z} \\ \tan\left(\frac{\theta_y}{2}\right) &= \frac{y}{z}\,. \end{aligned} \tag{B.1}$$

The intrinsic camera matrix yields the following projection equations, ignoring distortion

$$\begin{aligned} u &= u_0 + f_x\frac{x}{z} \\ v &= v_0 + f_y\frac{y}{z}\,. \end{aligned} \tag{B.2}$$

At the optical axis, the principal point offset (u_0, v_0) is zero, so these are rearranged as

$$\begin{aligned} \frac{u}{f_x} &= \frac{x}{z} \\ \frac{v}{f_y} &= \frac{y}{z}\,. \end{aligned} \tag{B.3}$$

The FOV is then calculated as twice the half-angles defined in Equation (B.1), using points at the edges of the image (e.g. $(u, v) = (320, 0)$ and $(u, v) = (0, 240)$ for a 640 × 480 image), and combining Equations (B.1) and (B.3)

$$\begin{aligned} FOV_x &= 2\arctan\left(\frac{u}{f_x}\right) \\ FOV_y &= 2\arctan\left(\frac{v}{f_y}\right)\,. \end{aligned} \tag{B.4}$$

Appendix C

Ceiling Tracker

Ground truth for the SLAM stage performance trials was generated using a ceiling tracker system that takes advantage of the structured ceiling tiles above the experimental environment. The ceiling in this environment comprises white tiles supported by dark metal beams, with the occasional light, fire sprinklers, and clock, as shown in Figure C.1. The corners of the square tiles form a repeating square grid that covers the entire robot workspace.

An EKF is used to track two-dimensional robot pose (x, y, θ) based on observations of the ceiling tile corners. This is purely a localization task, as the map is simply the rectangular grid of points which is easily measured. A feature detector was constructed from low-level image operations. The Canny edge detector is applied to images and detects edges on both sides of each dark support beam. The Hough transform is then used to find long lines in the edge image. The set of resulting lines is examined for the 90° intersections that occur at tile corners, and those in close proximity are grouped together to form possible tile corner points. These possible corner points are then constrained to a square grid with each square having the dimensions of a ceiling tile. Corner points are matched to the closest points in the square grid and used to update the EKF. The use of nearest-neighbor matching is necessary since the points are indistinguishable and implies that mismatches will occur if the robot drives too fast or processes images too slowly.

In practice the system works well and results in a measurable error on the order of a centimetre or less. It has been used to complete a 100 metre loop around the University of Alberta Computing Science Centre. Additionally, the constraints applied to detected tile corner position successfully filter out edges caused by clutter on the ceiling, although care is taken to drive around the skylights and overhangs that disrupt and obstruct the square grid.

Figure C.1: Sample image from ceiling tracker system. Intersections of the dark support beams at ceiling tile corners are detected and used as feature points in a square grid to perform robot localization. Lights and miscellaneous clutter do not meet the imposed square grid constraints and are ignored.

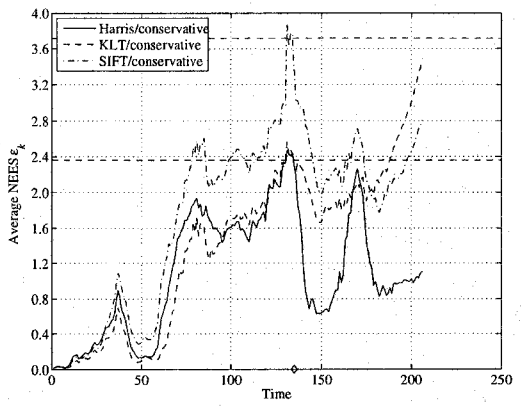
Appendix D

SLAM Results

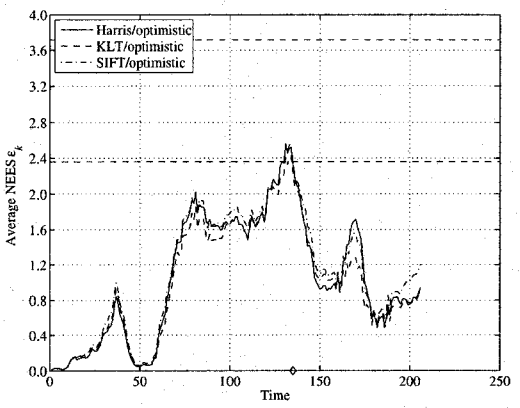
This appendix contains full results of the SLAM stage performance evaluation experiments in Chapter 5. Results are given for the five trials, each of which consists of the average results from 50 Monte Carlo runs. The results shown are:

- (a) and (b): Average NEES values using the *conservative* and *optimistic* motion models, respectively. Dashed lines represent the 95% consistency bounds. Values should lie within these bounds in a consistent system. The covariance estimate is too large if the NEES is below the lower bound, and too small if the NEES is above the upper bound.
- (c) and (d): Average accumulated uncertainty AU using the *conservative* and *optimistic* motion models, respectively. Error bars show the 1σ standard deviation calculated from the Monte Carlo runs. This is essentially the summation of the uncertainty described next.
- (e) and (f): Average uncertainty using the *conservative* and *optimistic* motion models. A measure based on the determinant of the covariance matrix that combines uncertainty in all pose variables, accounting for cross-correlation.

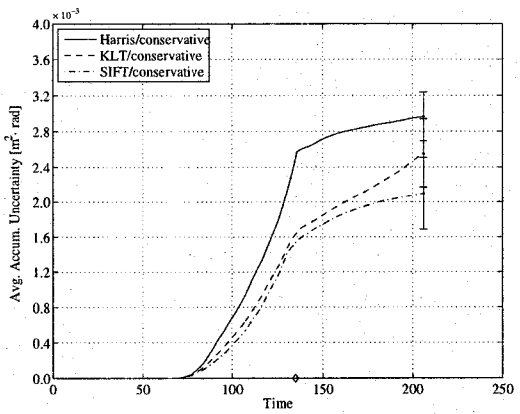
The time axis of each graph is further labeled with a $\diamond$ symbol. This indicates the approximate time at which the robot is able to re-observe the initial section of the environment.



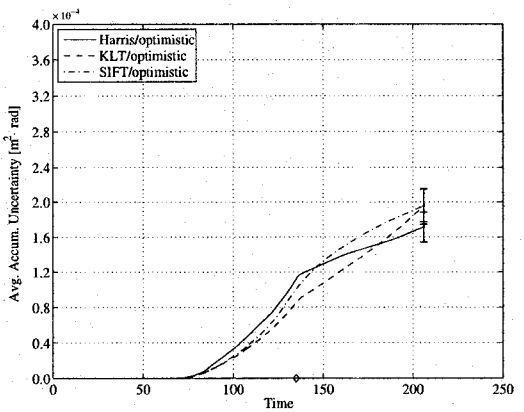
(a) Average NEES with *conservative* model.



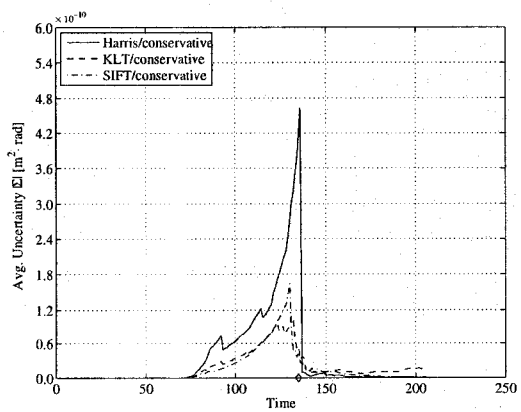
(b) Average NEES with *optimistic* model.



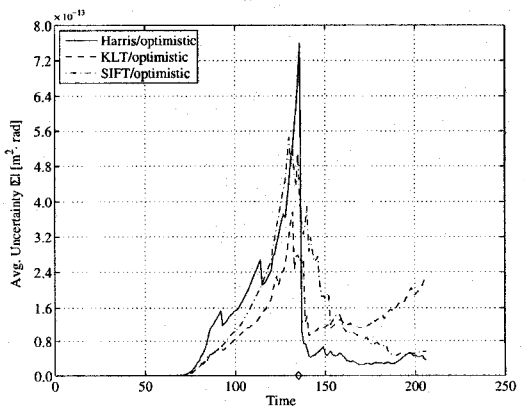
(c) Average accumulated uncertainty with *conservative* model.



(d) Average accumulated uncertainty with *optimistic* model.

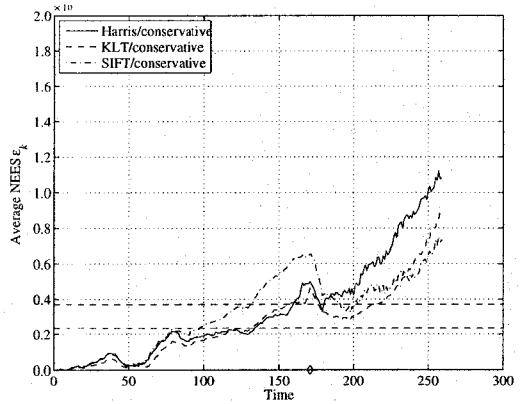


(e) Average uncertainty with *conservative* model.

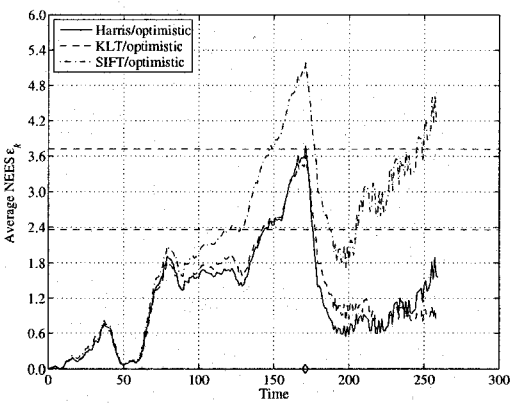


(f) Average uncertainty with *optimistic* model.

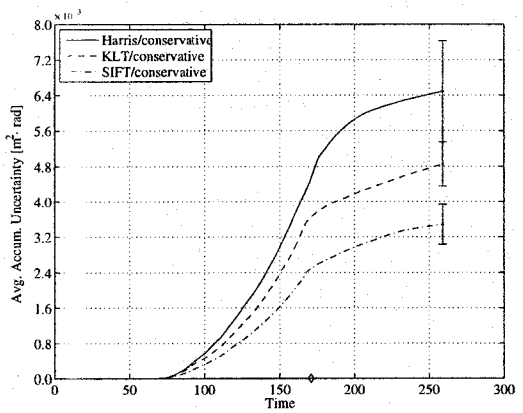
Figure D.1: Trial One



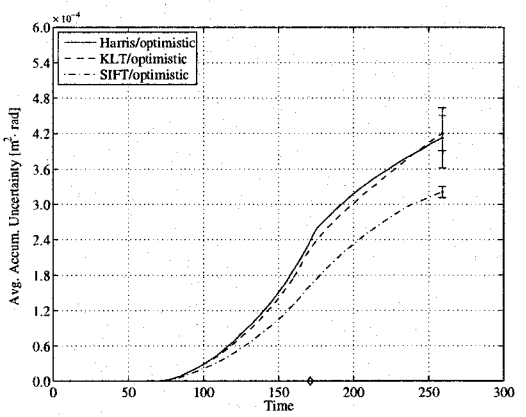
(a) Average NEES with *conservative* model.



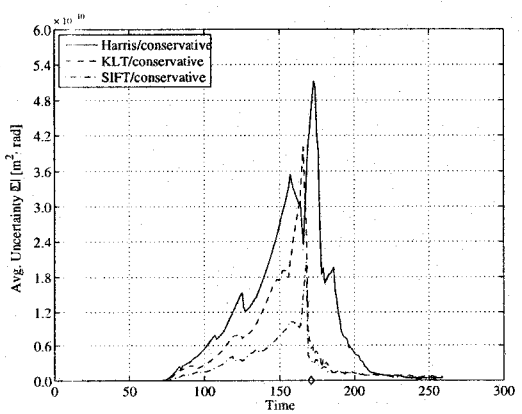
(b) Average NEES with *optimistic* model.



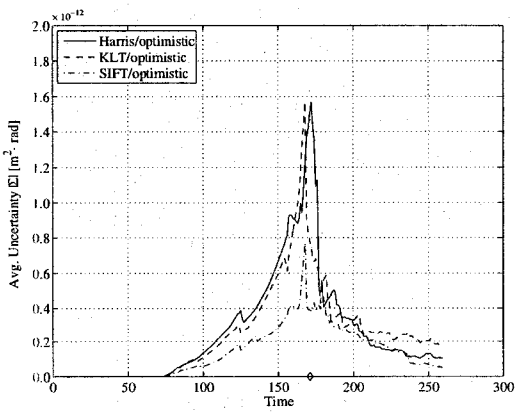
(c) Average accumulated uncertainty with *conservative* model.



(d) Average accumulated uncertainty with *optimistic* model.

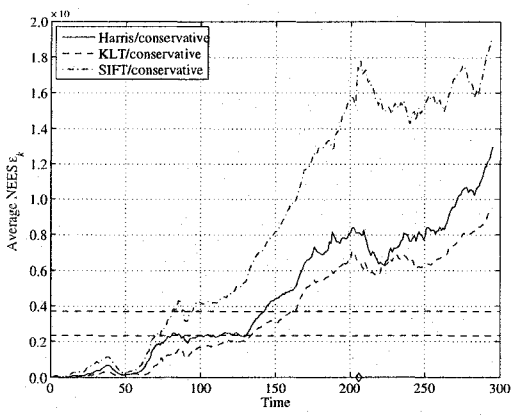


(e) Average uncertainty with *conservative* model.

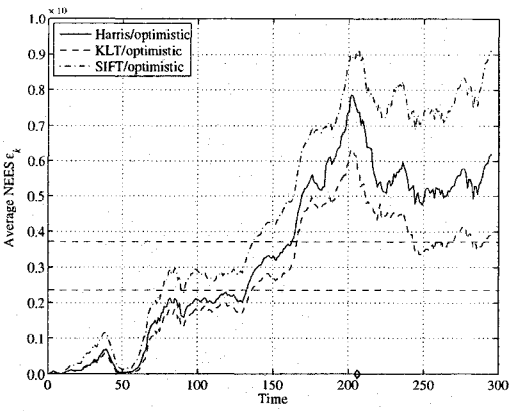


(f) Average uncertainty with *optimistic* model.

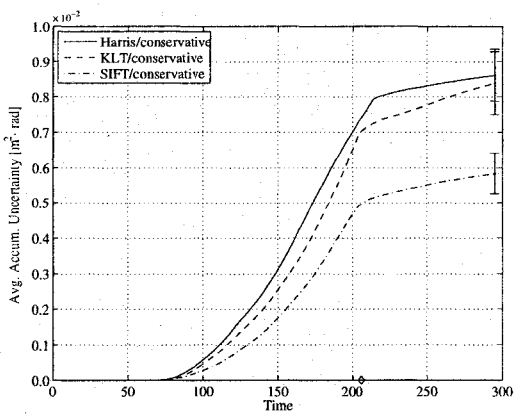
Figure D.2: Trial Two



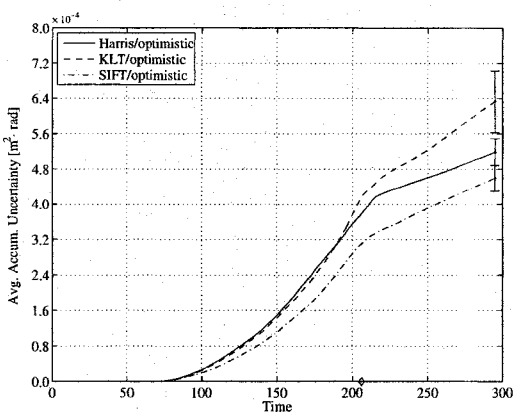
(a) Average NEES with *conservative* model.



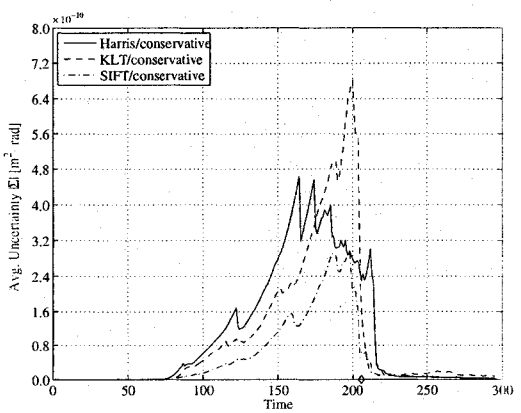
(b) Average NEES with *optimistic* model.



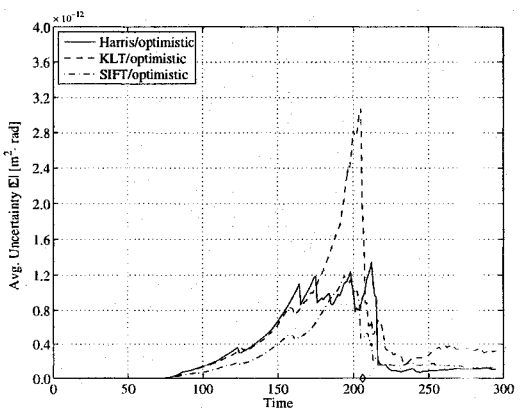
(c) Average accumulated uncertainty with *conservative* model.



(d) Average accumulated uncertainty with *optimistic* model.

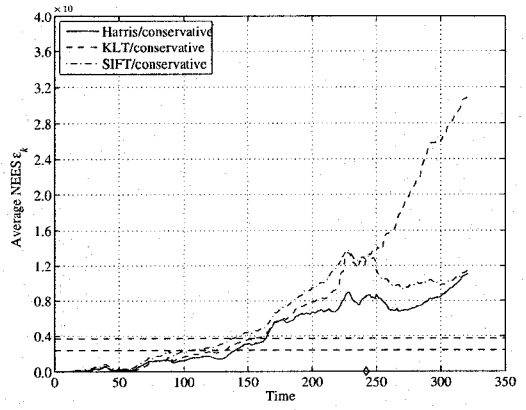


(e) Average uncertainty with *conservative* model.

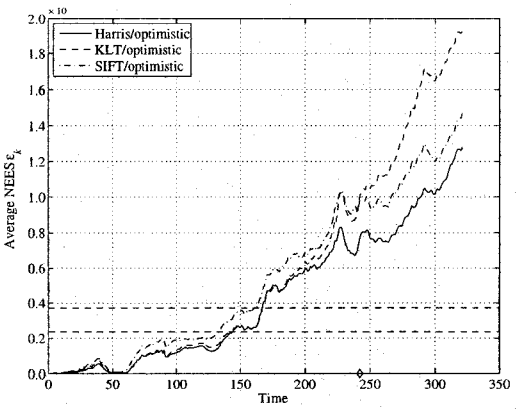


(f) Average uncertainty with *optimistic* model.

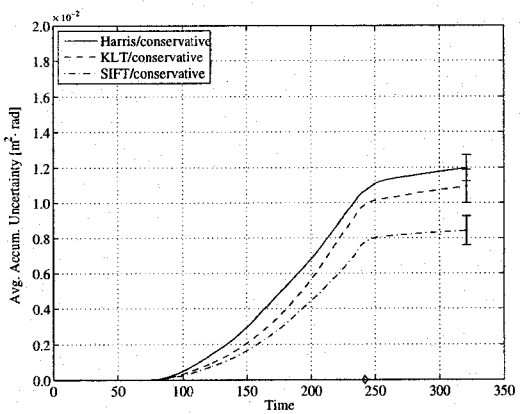
Figure D.3: Trial Three



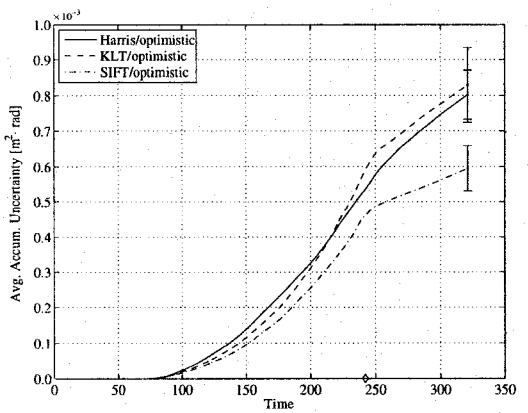
(a) Average NEES with *conservative* model.



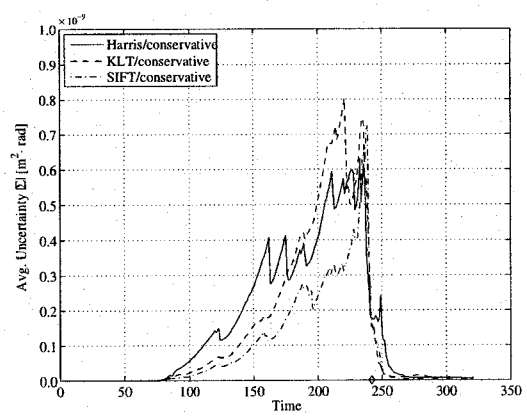
(b) Average NEES with *optimistic* model.



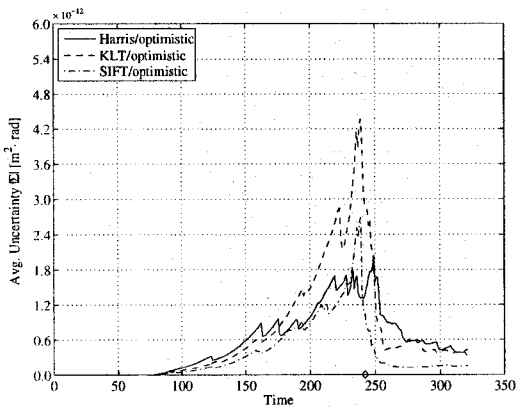
(c) Average accumulated uncertainty with *conservative* model.



(d) Average accumulated uncertainty with *optimistic* model.

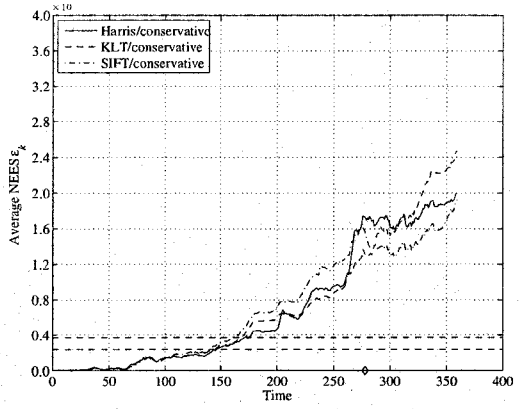


(e) Average uncertainty with *conservative* model.

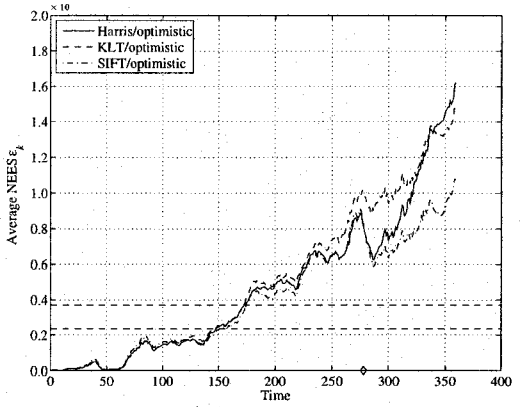


(f) Average uncertainty with *optimistic* model.

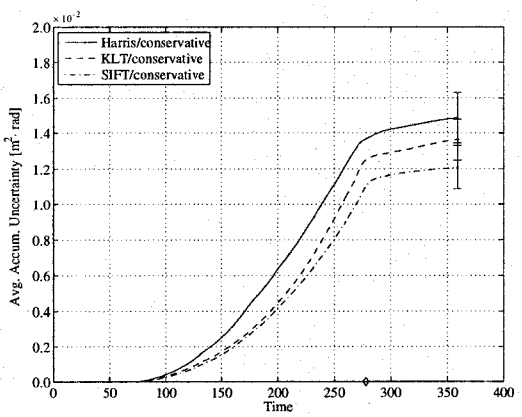
Figure D.4: Trial Four



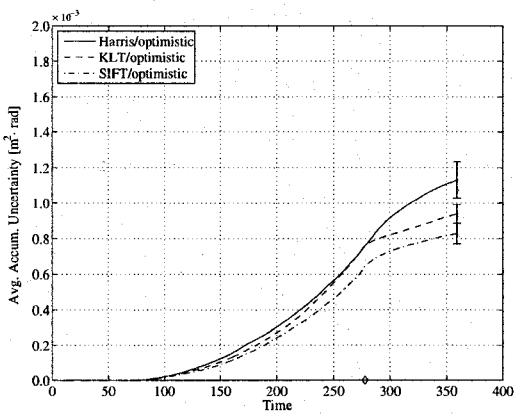
(a) Average NEES with *conservative* model.



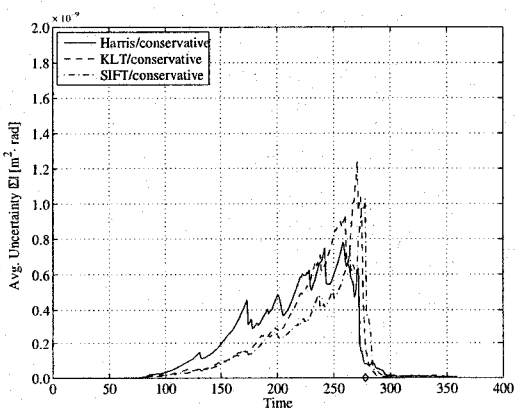
(b) Average NEES with *optimistic* model.



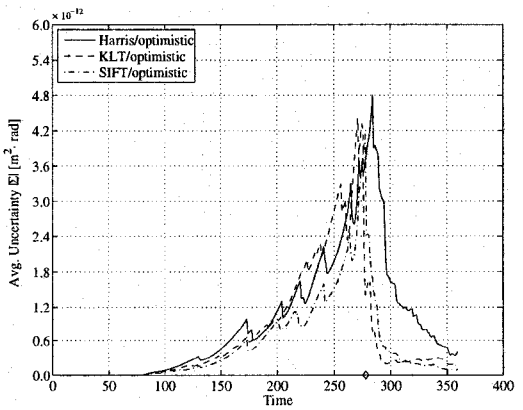
(c) Average accumulated uncertainty with *conservative* model.



(d) Average accumulated uncertainty with *optimistic* model.



(e) Average uncertainty with *conservative* model.



(f) Average uncertainty with *optimistic* model.

Figure D.5: Trial Five