IMPROVING VISUAL SLAM BY FILTERING OUTLIERS WITH THE AID OF OPTICAL FLOW

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Tolga Özaslan

July, 2011

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Uluç Saranlı(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Selim Aksoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Buğra Koku

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

IMPROVING VISUAL SLAM BY FILTERING OUTLIERS WITH THE AID OF OPTICAL FLOW

Tolga Özaslan
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Uluç Saranlı
July, 2011

Simultaneous Localization and Mapping (SLAM) for mobile robots has been one of the challenging problems for the robotics community. Extensive study of this problem in recent years has somewhat saturated the theoretical and practical background on this topic. Within last few years, researches on SLAM have been headed towards Visual SLAM, in which camera is used as the primary sensor. Superior to many SLAM application run with planar robots, VSLAM allows us to estimate the 3D model of the environment and 6-DOF pose of the robot. Being applied to robotics only recently, VSLAM still has a lot of room for improvement. In particular, a common issue both in normal and Visual SLAM algorithms is the data association problem. Wrong data association either disturbs stability or result in divergence of the SLAM process. In this study, we propose two outlier elimination methods which use predicted feature location error and optical flow field. The former method asserts estimated landmark projection and its measurement locations to be close. The latter accepts optical flow field as a reference and compares the vector formed by consecutive matched feature locations; eliminates matches contradicting with the local optical flow vector field. We have shown these two methods to be saving VSLAM from divergence and improving its overall performance. We have also described our new modular SLAM library, *SLAM++*.

Keywords: Visual Simultaneous Localization and Mapping (SLAM), optical flow, outlier elimination.

ÖZET

GÖRSEL EŞZAMANLI HARITALAMA VE KONUMLANDIRMA PROBLEMININ PERFORMANSINI AYKIRI GOZLEMLERI OPTIK AKI YARDIMIYLA ELEYEREK ARTIRMA

Tolga Özaslan
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Assist. Prof. Dr. Uluç Saranlı
Temmuz, 2011

Mobil robotlarla Eşzamanlı Haritalama ve Konumlandırma (EHK), robotik camiasının en zorlu problemlerinden biridir. Geçtiğimiz birkaç yılda, üzerine yapılan yoğun çalışmalar neticesinde, bu konu teorik ve pratik açılardan doyuma ulaşmıştır. Geçtiğimiz birkaç sene içerisinde, araştırmaların yönelimi EHK'den, ölçüm aygıtı olarak kameraların kullanıldığı Görsel EHK'ye doğru olmuştur. Düzlemsel uzayda çalışan birçok EHK uygulamasına kıyasen daha üstün olarak, GEHK, ortamın 3 boyutlu modelini ve robotun 6 serbestlik dereceli durumunu da kestirebilmektedir. Robotik çalışmalarına henüz uygulanmakla beraber, GEHK'nin geliştirilmesi gereken çok yönleri bulunmaktadır. Özellikle, EHK ve GEHK algoritmalarının ortak problemi bilgi eşlemesidir. Hatalı bilgi eşlemesi EHK'nin kararlığını olumsuz yönde etkileyebilir ya da tamamen ıraksamasına neden olabilir. Bu çalışmada, aykırı gözlemleri elemek için, tahmini izdüşüm hatasını ve optik akı bilgisini kullanan iki yöntem öneriyoruz. İlk yöntem, harita öğelerinin tahmini izdüşüm ve onlarla eşlenen ölçüm yerlerinin yakın olması gerektiği mantığını kullanmaktadır. İkinci yöntem ise, optik akı vektor alanını referans kabul edip, ardışık iki ölçüm ile belirlenen vektör ile, bölgesel optik akı alanını kıyas ediyor; ve optik akı alanı ile çelişen ölçümleri eliyor. Çalışmamızda, bu iki yöntemin, GEHK'nin ıraksamasını engellediğini ve genel performansını artırdığını gösteriyoruz. Ayrıca, modüler bir EKH kütüphanesi olan *SLAM++* yazılımımızı açıklıyoruz.

Anahtar sözcükler: Görsel Eşzamanlı Haritalama ve Konumlandırma, optik akı, aykırı gözlem eleme.

Acknowledgement

First of all I owe thanks to my supervisor Uluç Saranlı for his patience, encouragement and support through out my studies. I have learnt much from him on how an academic study is made. I would not be able produce such a work unless he has given me his great moral and material support.

I am grateful to Buğra Koku who has conduced to my joining the SensoRHex and Bilkent Dexterous Robotics and Locomotion Group. He has also given me advises as a teacher which warmed me towards the academy.

I am also grateful to Afşar Saranlı for his guidance and expertise, who has taught me a lot on robotics through the lectures in Middle East Technical University.

I thank to Mert Ankaralı for his moral help through my studies, Emre Ege for being a patient colleague through our studies with RHex, Sıtar Kortik for not leaving me alone in office, Kadir Akbudak for being a devoted friend, Mustafa Ürel for being good a pacemaker, Utku Çulha for being a chat-mate with whom I never get bored, Gökhan Gültekin for being a knowledgeable colleague, Bilal Turan for being a cheerful chatter, İsmail Uyanık and Özlem Gür and all Bilkent Dexterous Robotics and Locomotion Group and SensoRHex members.

I am also appreciative of the financial support from Bilkent University, Department of Computer Engineering, and TÜBİTAK, the Scientific and Technical Research Council of Turkey.

Finally I owe my loving thanks to my parents Cemanur and Hüseyin, and to my sister Tuğba for their patience and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Simultaneous Localization and Mapping has been one the most studied topics in mobile robotics [31, 41]. This problem involves estimating the location of the robot in the map while generating the map at the same time. The need for mapping an environment comes from the need for automating robots. Robots are designed so that they can achieve their tasks by themselves. Without knowledge of how the environment is, autonomy cannot be achieved. However, real maps are usually not available but even when they are, e.g. in the format of blueprints, what an object means to the robot can change. For this reason, it is advantageous for a robot to make its own map.

Due to this need, a large academic literature has grown in the last two decades on the SLAM topic [12, 13, 19, 26, 28, 32, 39]. Most studies are concerned with generating 2D maps using onboard sensors. In the last decade, these studies continued to generate 3D maps as well. Consumer level cameras became one of the commonly used sensors for building 3D maps of the environment. This type of SLAM is named Visual-SLAM, or shortly VSLAM. Nowadays there are studies in the literature which can do VSLAM in real time, i.e. at 30 fps [11, 25].

In SLAM, data association is one of the most common points of failure resulting in wrong maps and even divergence of the algorithm. In the context of Visual

SLAM, *data* is a set of features extracted from image frames. Consequently, in VSLAM, good data association means correct matching of these image features. Matching these features with existing map components can be done in a controlled way. In the literature, model based methods like RANSAC are used for outlier elimination [10], presuming that a model is available for how feature points are located. In our study, we use optical flow for eliminating false feature matches. This way feature matches are eliminated up to a certain level, increasing the overall performance of mapping and localization.

These goals also need a good VSLAM library and this has driven us to also implement a modular VSLAM library. This gives us the chance of testing our contributions both in simulated and real data sets.

1.2 Contributions

The two main contributions of this thesis are:

1. A new method to eliminate false interest point matches using optical flow.
2. Design and implement a C++ library for Visual SLAM tasks and applications.

We have performed outlier elimination using optical flow information and projection accuracy. In this study, we have also implemented a C++ library for Visual SLAM. This library includes EKF-SLAM, FastSLAM 1.0 and FastSLAM 2.0. For different purposes one of the SLAM versions can be run.

1.3 Organization of the Thesis

Chapter 2 starts the thesis with background on related topics. These topics include optical flow calculation, interest point extraction and Simultaneous Localization and Mapping (SLAM). In Chapter 3, we describe several interest point matching algorithms and introduce our two outlier elimination methods, optical flow aided and prediction error based outlier eliminators. Chapter 4 gives results

of several test scenarios on which the proposed outlier elimination methods are applied. In Chapter 5, we briefly describe software architecture of our modular SLAM library, SLAM++. Finally, we proceed with conclusion of our study and discussion on the proposed methods.

Chapter 2

Background and Related Work

Visual SLAM uses cameras as primary sensors for localization and mapping. Camera supplies color, texture and shape information from the environment. However, this raw data should be processed in order to obtain useful information in the form of 'local features'. Local features can be summarized as the set of distinctive image regions of an image. These are often tracked for estimating their 3D location in space. Then, these local features can also be transformed into map elements. For this reason, feature extraction and matching are important steps in VSLAM. In the following chapters, brief descriptions about some of the mostly used feature extraction and matching algorithms are given.

Optical flow calculation is another subtask within this thesis. The calculation of optical flow, which in itself is a huge research area, gives pixel displacements in a sequence of frames. Even though optical flow information is not used in existing Visual SLAM studies, it can be useful in eliminating false feature matches. In this thesis we use optical flow for this purpose. This chapter also gives brief descriptions of a number of optical flow calculation algorithms.

The nature of the SLAM problem does not change according to which sensors are used, so once they are modeled correctly, different types of sensors can be used. In this study, we implement Visual SLAM (VSLAM), which uses monocular vision. In Section 2.3 brief mathematical derivations of VSLAM are given as well.

2.1 Optical Flow

The calculation of optical flow is one of the fundamental problems in image processing [2]. The aim of optical flow calculation is to compute 2D projections of 3D velocities in the scene [22]. In other words, optical flow is the observed velocities of intensity patterns on an image. There are various areas of application for optical flow information such as motion estimation and surface reconstruction [1, 3, 21]. Depending on the application, *dense* or *sparse* flow fields may be needed. For instance, for surface reconstruction, dense flow is required. However, for object tracking, sparse flow may be adequate. Optical flow can also be used for extracting spatial arrangements of objects in the scene by inspecting flow discontinuities. In this study, we use flow vectors for eliminating false feature matches. As such, we aim to increase the ratio of true matches to the total number of matches and as a result, improve VSLAM performance.

Optical flow calculation techniques can be investigated under four main groups [2]: differential methods, region-based methods, energy-based methods and phase-based methods. Since in this study, we will only use differential methods, background on other methods will not be included.

Differential methods compute optical flow vectors using spatio temporal derivatives of image sequences. Image intensity constancy is the main idea behind these methods, with

$$I(\mathbf{x}, t) = I(\mathbf{x} - \mathbf{v}t, 0), \tag{2.1}$$

where $\mathbf{x}$ is the image pixel location, t is time and $\mathbf{v}$ is linear velocity [23]. Some methods use the first order derivatives of the image sequence. Applying Taylor's expansion rule on (2.1) we obtain

$$\nabla I(\mathbf{x}, t) \cdot \mathbf{v} + I_t(\mathbf{x}, t) = 0, \tag{2.2}$$

where $I_t(\mathbf{x}, t)$ denotes derivative of $I(\mathbf{x}, t)$ with respect to time, and $\nabla I(\mathbf{x}, t) = (I_x(\mathbf{v}, t), I_y(\mathbf{v}, t))^T$ where I_x and I_y are derivatives of I w.r.t. $\mathbf{v}_x$ and $\mathbf{v}_y$ respectively.

Some other methods use the second order Taylor's expansion of (2.1) to constrain

velocities [5]. In other words, the Hessian of the image is used, with

$$\begin{bmatrix} I_{xx}(\mathbf{x},t) & I_{yx}(\mathbf{x},t) \\ I_{xy}(\mathbf{x},t) & I_{yy}(\mathbf{x},t) \end{bmatrix} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \end{bmatrix} + \begin{bmatrix} I_{tx}(\mathbf{x},t) \\ I_{ty}(\mathbf{x},t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2.3}$$

Equation (2.3) can be derived from (2.1). This coincides with the conservation of $\nabla I(\mathbf{x},t)$ with $d\nabla I(\mathbf{x},t)/dt = 0$. The above equations presume that $I(\mathbf{x},t)$ is differentiable. For this reason, numerical differentiation should be done carefully.

2.1.1 Horn and Schunck

Horn and Schunck [23] use gradient constancy of (2.2) with a global smoothness term. For each pixel, we have only one known which is the intensity; but two unknowns which are the $\mathbf{v}_x$ and $\mathbf{v}_y$ velocities. Due to this fact, optical flow cannot be computed only using (2.2). For this reason, more constraints should be introduced into problem. Horn and Schunck assumes smoothness of flow almost everywhere in the image. The problem is handled as an energy minimization problem. The closed form equation of the problem is as follows:

$$\int_D (\nabla I(\mathbf{x},t) \cdot \mathbf{v} + I_t(\mathbf{x},t))^2 + \lambda^2(||\nabla \mathbf{v}_x||^2 + ||\nabla \mathbf{v}_y||^2)dx \tag{2.4}$$

where D is the domain, in this case the image, and λ is the importance weight of smoothness. They give an iterative solution to this energy minimization problem as

$$\mathbf{v}_x^{k+1} = \bar{\mathbf{v}}_x^k - \frac{I_x(I_x\bar{\mathbf{v}}_x^k + I_y\bar{\mathbf{v}}_y^k) + I_t}{\lambda^2 + I_x^2 + I_y^2} \tag{2.5}$$

$$\mathbf{v}_y^{k+1} = \bar{\mathbf{v}}_y^k - \frac{I_y(I_x\bar{\mathbf{v}}_x^k + I_y\bar{\mathbf{v}}_y^k) + I_t}{\lambda^2 + I_x^2 + I_y^2}, \tag{2.6}$$

where k denotes the iteration number, $\bar{\mathbf{v}}_x$ and $\bar{\mathbf{v}}_y$ are weighted averages of velocity component of the neighboring pixels and initial values are $\mathbf{v}_x^0 = 0$ and $\mathbf{v}_y^0 = 0$.

2.1.2 Lucas and Kanade

Lucas and Kanade [30] assume constancy of flow in a local neighborhood of the pixel under consideration. Using a least squares criterion, they solve (2.1) for all the pixels in that neighborhood. Lucas and Kanade minimize the energy function

$$\sum_{\mathbf{x}\in\Omega} W^2(\mathbf{x})[\nabla I(\mathbf{x},t)\cdot\mathbf{v} + I_t(\mathbf{x},t)]^2, \tag{2.7}$$

where $W(\mathbf{x})$ is a windowing function. This function has higher coefficients in the center of the window and smaller in peripherals. Solution to (2.7) is

$$\mathbf{A}^T\mathbf{W}^2\mathbf{A}\mathbf{v} = \mathbf{A}^T\mathbf{W}^2\mathbf{b}, \tag{2.8}$$

where

$$\mathbf{A} = [\nabla I(\mathbf{x}_1), ..., I(\mathbf{x}_n)]^T \tag{2.9}$$

$$\mathbf{W} = diag[W(\mathbf{x}_1), ..., W(\mathbf{x}_n)] \tag{2.10}$$

$$\mathbf{b} = -(I_t(\mathbf{x}_1), ..., I_t(\mathbf{x}_n))^T. \tag{2.11}$$

As a result, the solution is found as

$$\mathbf{v} = [\mathbf{A}^T\mathbf{W}^2\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{W}^2\mathbf{b}. \tag{2.12}$$

Since this is a local method, it may not give correct estimates for interiors of uniform regions. There are implementations of this algorithm which assume $W(\mathbf{x}) = 1$. In this case the solution is the common least squares of (2.7). In other cases, it becomes a weighted least squares problem.

2.1.3 General Variational Methods

In image processing, variational methods have attracted the attention of researcher in recent years. These methods provide good and clear formalization of flow model assumptions [44]. Once the mathematical model is formalized, the

problem boils down to an energy minimization problem, which gives the best result for the given model assumptions.

One of the earliest and pioneering methods, which uses variational methods for computation of optical flow, is the study of Horn and Schunk [23] In this method, the problem is minimizing the energy function which consists of data and smoothness terms. Data term includes flow constraints such as gray value constancy; and smoothness term constraints the flow to vary smoothly in space. The resultant energy function to be minimized yields to be of the form

$$E(\mathbf{v}) = \int (I(\mathbf{x}+\mathbf{v}) - I(\mathbf{x}))^2 + \alpha(\nabla I(\mathbf{x}+\mathbf{v}) - \nabla I(\mathbf{x}))dx. \tag{2.13}$$

Data term can be edited to include more constraints, such as Hessian and Laplacian constancy, which makes the problem harder but increases the accuracy in the resultant flow field.

$$E(\mathbf{v}) = \int (I(\mathbf{x}+\mathbf{v}) - I(\mathbf{x}))^2 + (H(I(\mathbf{x}+\mathbf{v})) - H(I(\mathbf{x})))^2 + \tag{2.14}$$

$$(\Delta(I(\mathbf{x}+\mathbf{v})) - \Delta(I(\mathbf{x})))^2 + \alpha(\nabla I(\mathbf{x}+\mathbf{v}) - \nabla I(\mathbf{x}))dx. \tag{2.15}$$

2.2 Feature Detectors

Local features are pieces of images, such as points, edgels or image patches, which differ from their immediate neighborhood [43]. In the literature, there are many feature extraction algorithms [14, 20, 37, 38, 40] some of which attach descriptors to these features. These descriptors can be obtained by using image properties such as gradients, curvatures, color, texture etc. Once a descriptor is associated to a feature, it can be used for a wide range if applications. To illustrate, edges can be interpreted as roads in a satellite image; blobs can be used as features in cancer cell detections; corners are usually good for tracking with algorithms like KLT [30]. Image mosaicking, camera calibration, pose estimation are some of the other areas of applications for features detectors.

In some applications, such as tracking or calibration, good localization of features

becomes important. In contrast, in some other problems, exact location is not so important but descriptors have more importance. Object recognition may be a good example for this case. In recognition problems, rather than individual features, statistics of a set of features becomes meaningful. As can be seen from the above statements, every application has its distinct constraints. According to these constraints, the best type of the feature and its descriptor differs.

Ideally a feature should be a point. However images are discrete signals with smallest elements as pixels. For this reason, sometimes subpixel localization is needed. In order to do subpixel localization pixels around a point should also be investigated. Furthermore, for attaching descriptors to features, an image region around the location of the feature is analyzed. As a result, the assumption of 'point feature' is confuted with the above facts. In some applications like camera calibration, 3D reconstruction descriptors are not needed. But in applications like object recognition, VSLAM such extended descriptors are a must.

In [43], properties of an ideal local feature are listed as follows:

- *Repeatability* : Similar results should be obtained from different images of a single scene. These images could be taken from different angles and locations and there may also be lighting changes.
- *Distinctiveness* : Patterns at feature locations should be distinguishable for better matching.
- *Locality* : The features should be local. A feature should not be defined with a region, but a point.
- *Quantity* : Enough number of features should be extracted from a single image. Too many and too few number of features are not desired.
- *Accuracy* : Location of features should be accurate both in image coordinates and scale. Subpixel and subscale localizations should be done.
- *Efficiency* : Time needed for extracting features should allow time-critical applications.
- *Invariance* : Under large deformations and intensity changes, description of the feature should not change significantly.

- *Robustness* : Accuracy of the extractor should not degrade under relatively small image deformations.

The importance of these properties differs according to the application. In the VSLAM literature, blobs, edgels, corners are among the mostly commonly used types. FAST features [37] and Harris corners [20] with patches as descriptors [25], SIFT [29] and SURF [4] features are among the most often used algorithms for the VSLAM problem [34]. Repeatability, invariance are two of the most important properties of a feature detector in a VSLAM application [16–18, 33], since a feature should be detected several times while the camera is moving. Also, features detected in different frames should be matched correctly. Efficiency is again an important property if real time applications are to be developed. With an increase of quantity, performance of VSLAM can degrade but this would increase number of map components. This way, dense maps can be obtained and with more features, localization performance also increases.

2.3 Simultaneous Localization and Mapping

In [31], the authors observe that "Simultaneous Localization and Mapping (SLAM) addresses the problem of acquiring an environment map with a roving robot, while simultaneously localizing the robot relative to this map". This problem has attracted enormous attention from many robotics researchers in recent years. In this context, the robot knows neither the map of the environment nor its own pose. However, the robot is fed with a series of commands and measurements, using which it should extract a map and estimate its own pose. Compared to its two siblings, *'mapping'* in which the pose of the robot is given and *'localization'* in which a map of the environment is given, it is obvious that SLAM is a significantly harder problem. Fortunately, large body of literature exists [8, 11, 15, 24, 35, 36, 42], as a result of which many difficulties in SLAM are solved. Nevertheless, there is much room for development, since robots still cannot be put out to a completely unknown environment and wander around.

In the SLAM problem, the pose of the robot at time t is denoted by s_t. In Visual

SLAM, this pose includes 3D position and orientation, as well as translational and rotational velocities. The state vector can change according to the type of motion model. In this thesis, a constant velocity motion model is used. This implies that, both rotational and translational velocities are assumed to remain constant between consecutive frames. The complete trajectory of the robot, which consists of the set of poses at each frame, is denoted with

$$\mathbf{s}^t = \{s_1, s_2, ..., s_t\}. \tag{2.16}$$

The environment of the robot is modeled as a set of N landmarks. These landmarks may be the output of a SIFT feature detector. The set of N landmarks represents a map Θ, denoted as

$$\Theta = \{\theta_1, \theta_2, ..., \theta_N\}. \tag{2.17}$$

In this thesis, we assume that the robot and camera have equal meanings from which one should understand a system with full state vector. The set of control inputs are denoted as

$$\mathbf{u}^t = \{u_1, u_2, ..., u_t\}. \tag{2.18}$$

These inputs can be obtained from odometry, inertial navigation units or the given commands may already be known.

While the robot moves, it takes measurements from its environment. Various types of sensors can be used for this purposes, such as laser scanners, sonars and cameras. In this thesis, a single camera is used as the primary sensor. The observation at time t is denoted by z_t and all of the measurements up to time t are written as

$$\mathbf{z}^t = \{z_1, z_2, ..., z_t\}. \tag{2.19}$$

Using the notations up to now, the pose distribution of the robot in probabilistic terms is denoted as

$$p(\mathbf{s}_t, \Theta | \mathbf{z}^t, \mathbf{u}^t). \tag{2.20}$$

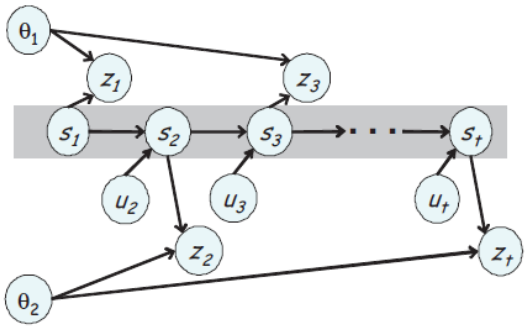


Figure 2.1: SLAM as a dynamic Bayes network

The SLAM problem can be best described as a probabilistic Markov chain [41]. Figure 2.1 visualizes this chain. The pose $\mathbf{s}_t$ of the robot is a function of its previous state $\mathbf{s}_{t-1}$ and the control executed $\mathbf{u}_t$. This function can be named as the *motion model* of the robot. The motion model not only applies control inputs to the robot but also integrates process noise which exists in control inputs. This model can be written as

$$p(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{u}_t). \tag{2.21}$$

As can be seen from Figure 2.1, sensor measurements gathered by the robot are included in this Markov chain. Each measurement is a function of the visible set of landmarks and state of the robot. This function is named as the *measurement model* and represented with the probability distribution

$$p(\mathbf{z}_t|\mathbf{s}_t, \Theta). \tag{2.22}$$

Using a Bayes filter and these two functions, namely motion and measurement models, the SLAM posterior at time t can be recursively estimated. This filter can be shown as

$$p(\mathbf{s}_t, \Theta|\mathbf{z}^t, \mathbf{u}^t). \tag{2.23}$$

Unfortunately, we cannot represent (2.23) in closed form. Some assumptions should be made about the motion and measurement models, as well as the type of noise in the system. The Extended Kalman Filter (EKF) represents this posterior as a multivariate Gaussian random variable with a mean μ and a covariance Σ.

$$\mu_t = \left\{\mu_{s_t}, \mu_{\theta_{1,t}}, ..., \mu_{\theta_{N,t}}\right\} \tag{2.24}$$

$$\Sigma_t = \begin{bmatrix} \Sigma_{s_t,t} & \Sigma_{s_t\theta_1,t} & \dots & \dots & \Sigma_{s_t\theta_N,t} \\ \Sigma_{\theta_1 s_t,t} & \Sigma_{\theta_1,t} & \Sigma_{\theta_1\theta_2,t} & \dots & \Sigma_{\theta_1\theta_N,t} \\ \dots & \ddots & \ddots & \ddots & \vdots \\ \Sigma_{\theta_N s_t,t} & \Sigma_{\theta_N\theta_1,t} & \dots & \dots & \Sigma_{\theta_N,t} \end{bmatrix}. \tag{2.25}$$

The size of the state vector and the covariance matrix depends on the type of motion model and measurement model. For a robot with a constant velocity motion model and landmarks parametrized with inverse depth [9] μ is a $6N + 13$ vector where robot state μ_{s_t} and landmark state μ_{θ_i} are

$$\mu_{s_t} = \begin{pmatrix} x^W \\ q^{WR} \\ v^W \\ w^R \end{pmatrix} \tag{2.26}$$

$$\mu_{\theta_i} = \begin{pmatrix} x, & y, & z, & \theta, & \phi, & \rho \end{pmatrix}^T. \tag{2.27}$$

Given the above state representations, the covariance becomes a *6N+13* square matrix. Thus, the representation of the SLAM posterior with the EKF has quadratic size complexity in the number of landmarks.

The EKF is, as its name suggests, an extension to the Kalman Filter, which linearizes the nonlinear functions at their most likely value. For this reason, in order for this linearization to give good performance, nonlinear functions should be approximately linear at the mean point. In this thesis, both motion and measurement models are nonlinear functions. Due to the quaternion multiplication in motion model, it needs linearization. Inverse depth parametrization is used as the measurement model and it includes trigonometric terms in which makes it a nonlinear function too. As described in [9], parameterizing the landmarks with inverse depth, better linearization is possible which improves the SLAM performance.

Chapter 3

Data Association in Visual SLAM

The data association problem is one of the most important problems in SLAM applications. Although the SLAM framework models the probabilistic nature of localization and mapping problems well, data association is not directly addressed within this framework. Associating new data with existing data needs special treatment. In the context of Visual SLAM, this process is handled under the topic of feature matching. Data association should be handled carefully for good performance.

In some studies, maximum-likelihood is used for feature matching [41], in which the probabilistic framework of SLAM is utilized for associating new information with an existing map. This subtype of SLAM problem is specifically named as 'SLAM with unknown data association' whose success rate is open for criticizing. This method for matching features is generally used in cases where no well defined or significant cues for identifying features exist. However, in many SLAM applications, rather that using unprocessed data, researchers try to fit descriptions and use them for feature matching.

As in many SLAM applications, Visual SLAM does not rely on maximum likelihood. Rather than this, VSLAM uses image features with patches or descriptors

for data association. In the Visual SLAM case, raw information is supplied as image frames and interest points are extracted from these images. Once an interest point is determined, a descriptor is fit to identify it. SIFT and SURF descriptors, FAST and Harris corners with warped image patches are among the mostly used alternatives. In this thesis, SIFT features are used.

3.1 Matching Features

SIFT and SURF features have their own descriptors that can be used. FAST and Harris corners do not come with descriptors and are usually used together with image patches. As mentioned above, we have used SIFT features as landmarks in this study. For each frame, new features are extracted from the image and these features are compared with all of the map elements that are estimated to be visible from the current pose. There are various metrics and algorithms for comparing and matching these features. Usually, pairwise distance between two SIFT features, which is simply the Euclidean distance between their descriptors, is used. Usually, from a 640×480 image about 1000 SIFT features can be extracted. Matching two such feature sets can be accomplished in various ways. In this study, we will mention three different algorithms for this task:

1. SIFT Matching
2. Married Matching
3. Minimum Distance Matching

3.1.1 SIFT Matching

This algorithm calculates all pairwise distances between elements of both descriptor sets S_1 and S_2. In order for a feature in S_1 to be matched with another in S_2, the ratio of distances between two closest features in set S_2 to the feature in S_1 should be greater than the given threshold T_{match}. Using this heuristic, features which have more than one similar feature are prevented from being matched. In other words, since such features can be easily mismatched, even though distance

between these features' descriptors might be very small, rather than making mismatches, these features are simply rejected. Algorithm 1 describes this method.

Algorithm 1 Algorithm_SIFT_Matching(S_1, S_2, Thr)

```
for all s1 ∈ S1 do
  minDist1 ← ∞
  minDist2 ← ∞
  minFeat ← NULL
  for all s2 ∈ S2 do
    dist ← |s1 − s2|
    if dist < minDist1 then
      minFeat ← s2
      minDist1 ← dist
    else
      if dist < minDist2 then
        minDist2 ← dist
      end if
    end if
  end for
  if minDist1/minDist2 < Thr then
    M ← {M : [s1, minFeat]}
  end if
end for
```

For nonempty sets of features, this algorithm either matches a feature or rejects a match if two nearest features are close to each other. For example consider the case where S_1 has $N_1 > 1$ number of features and S_2 has $N_2 = 1$ feature. In this case, all of the features in set S_1 will be matched with the only feature in S_2 which is a weakness of the above matching algorithm. All of the matches other than one possibly true match will be all outliers. This analysis can be generalized as follows : when $N_1 >> N_2$, many of the features from S_2 will be matched to more than one feature in S_1. In the reverse case, with $N_2 >> N_1$, since the possible choices of features in set S_2 is very high, it is unlikely for features in set S_2 to be matched with more than one feature in set S_1. From this inspection we can conclude that the above algorithm is not suitable for cases $N_1 >> N_2$ and $N_2 << C$ for $C > 0$. However it is, by many applications, established that this algorithm gives good results, usually when the sets include around a few hundreds of features.

3.1.2 Married Matching

This algorithm, as in previous section 3.1.1, calculates all pairwise distances between feature descriptors in both sets. In order for two features to be matched, both descriptors should be the closest to the other descriptor among all descriptors in the other set. In other words, consider two descriptors $s_1 \in S_1$ and $s_2 \in S_2$, s_1 should be the closest descriptor to s_2 among all other descriptors in set S_1 and vice versa. So this method checks the distance between descriptors twice before matching them. In Section 3.1.1 there was the possibility of matching a feature in S_2 with more than one features in S_1. However in this algorithm, every feature in both sets is assigned to one and only one feature in the other set. Algorithm 2 describes this method.

When using Algorithm 1, it was shown above that for $N_1 >> N_2$, resulting matches would include many wrong pairs. However, in this algorithm, since closest descriptors in both directions are calculated, cases where the number of features in sets differ much more, are handled better. In other words, for cases with $N_1 >> N_2$ and $N_2 >> N_1$, this algorithm will not give as many wrong matches as the previous algorithm.

3.1.3 Minimum Distance Matching

This algorithm calculates distances between descriptors only in one direction. The sufficient condition for two features to be matched is that the distance between that pair is smaller than the given threshold T_{match} and the distance between all other descriptor pairs. In other words, consider two descriptors $s_1 \in S_1$ and $s_2 \in S_2$; s_2 should be the closest descriptor to s_1 among all other descriptors in set S_2. One weakness of this method is that matches will be such that many features from set S_2 will be assigned to more than a single feature from set S_1. For cases where $N_1 >> N_2$ this result will be more obvious. This method is explained in Algorithm 3.

Algorithm 2 Algorithm_Married_Matching(S_1, S_2)

$N_1 \leftarrow length(S_1)$
$N_2 \leftarrow length(S_2)$
$m_1 \leftarrow vector(N_2)$
$m_2 \leftarrow vector(N_1)$
$D_{21} \leftarrow vector(N_2, \infty)$
for $i = 1 : N_1$ **do**
  $s_1 \leftarrow S_1(i)$
  $minDist \leftarrow \infty$
  $minFeat \leftarrow -1$
  for $j = 1 : N_2$ **do**
    $s_2 \leftarrow S_2(j)$
    $dist \leftarrow |s_1 - s_2|$
    if $dist < minDist$ **then**
      $minFeat \leftarrow s_2$
      $minDist \leftarrow j$
    end if
  end for
  $m_1(i) = minFeat$
  if $minDist < D_{21}(minFeat)$ **then**
    $D_{21}(minFeat) = minDist$
    $m_2(minFeat) = i$
  end if
end for
for $i = 1 : N_1$ **do**
  if $m_1(i)! = -1$ && $m_2(m_1(i)) == i$ **then**
    $M = \{M : [S_1(i), S_2(m1[i])\}$;
  end if
end for

Algorithm 3 Algorithm_Minimum_Distance_Matching(S_1, S_2, Thr)

```
for all s1 ∈ S1 do
  minDist ← ∞
  minFeat ← NULL
  for all s2 ∈ S2 do
    dist ← |s1 − s2|
    if dist < minDist1 then
      minFeat ← s2
      minDist ← dist
    end if
  end for
  if minDist < Thr then
    M ← {M : [s1, minFeat]}
  end if
end for
```

3.2 Estimating Feasible Features

In this study, the map of the environment consists of sparse landmarks encoded as SIFT features. Each feature represents a landmark in 3D space through an inverse depth parametrization [9]. These features are used as identifiers of landmarks for data association. In addition to using feature descriptors for matching task, we utilize projected positions of landmarks as well. By projecting landmarks, we obtain estimates of pixel coordinates of these landmarks. Subsequently, landmarks whose projected positions lie inside the image plane are marked as visible and only these features are used in matching with the features extracted from the new frames. Using one of the feature matching algorithms given in Section 3.1, new features are matched with existing landmarks. There are two potential problems in this process : There may be errors in pose estimation which would result in wrong estimation of landmark visibility and there may be erroneous matches between new features and landmarks. Marking some of the visible features as not visible will result in degrading of localization performance. In particular, visibilities of features close to image boundaries might be estimated wrongly. Mismatches in the feature matching task will result in wrong EKF updates and that will affect both mapping and localization performances. Such mismatches are expected to be minimized through optical flow aided outlier elimination.

Algorithm 4 describes the visibility determination process.

Algorithm 4 Check_Visibility(y_i)

$xyz \leftarrow idp2xyz(y_i)$ {Convert from inverse depth rep. to Cartesian rep.}
$h_d \leftarrow camProj(xyz)$ {Project landmark to image plane}
$h_u \leftarrow distort(h_d)$ {Apply distortion to projected point}
if $inImage(h_u)$ **then**
  $visible = true$
else
  $visible = false$
end if

3.3 Outlier Elimination

In this thesis, we use SIFT features, with their own descriptors as interest points. In the literature, especially in object recognition, identification is done using only descriptor information but not using feature locations. However, in the context of VSLAM, features seen in the previous frame are usually searched in the next frame and this introduces a new constraint in matching these features which is the optical flow between these consecutive frames. Ignoring this constraint and using only descriptors for matching means discarding existing information. We actually do the matching using one of the methods described in the section 3.1; but after that, since these algorithms only consider descriptors but not feature locations, we try to eliminate outliers with the help of optical flow information. It is known and also shown in this work that outliers degrade the performance of SLAM algorithms. At the end, by eliminating outliers, we hope to see VSLAM performance will increase compared to the base case.

There is a second problem that should be handled separately, that cannot be solved by only optical flow aided outlier elimination. Suppose that two landmarks resemble each other. While matching, assigning the feature corresponding to the first landmark to the second landmark is very likely to happen. In this case, the match vector may not violate the constraint induced by the optical flow and, even though it is a wrong match, this fault may not be detected. In such situations, we have further information that is still not used. This is the estimated projected

locations of landmarks. If the distance between the projected locations of a landmark and the matched feature is higher than a threshold, we can conclude that this match is wrong.

The first method for eliminating outliers needs only features and the optical flow between consecutive frames. However, the second method requires the knowledge of the map. This relation is summarized in Figure 3.1, showing whether the elimination methods are applicable in absence or existence of a map.

Table 3.1: Applicable outlier eliminators to with and without maps

	Outlier Elimination Method	
	Optical Flow Aided	Prediction Error
No Map	✓	×
Map Available	✓	✓

In subsequent sections we will explain both of these elimination methods in detail.

3.3.1 Outlier Elimination Using Optical Flow

In Section 3.1, three different methods for matching features were explained. As can be seen from each of these algorithms, matching heuristics do not consider feature locations and only use their descriptors in identification. However, feature locations can also be used for either matching or eliminating outliers. In this thesis, we first match features using one of the algorithms given in Section 3.1, and filter out matches which are in contradiction with the optical flow field information. This way we utilize the unused available optical flow information.

The probabilistic framework of SLAM algorithms do not handle wrong data association but directly integrate any information with the current belief. Wrong data association will either result in catastrophic failures and divergence or degrade the certainty of its belief.

In order to accomplish outlier elimination, we track features observed in the previous frames, find optical flow vectors and compare this vector with the displacement vector determined by one of the feature matching algorithms. Using

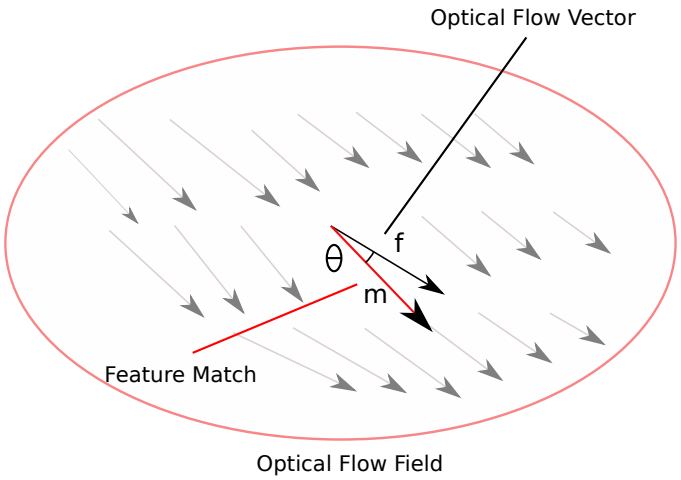


Figure 3.1: Possible optical flow vs feature match vector pairs, f and m resp. - inlier case. In this sample, f and m have similar orientations and magnitudes. This agreement results in marking the match as an inlier.

carefully designed metrics, we eliminate matches which contradict with the flow vector.

For outlier elimination, we compare the flow vector $\overrightarrow{f}$ and the feature displacement vector obtained as the result of the match algorithm, $\overrightarrow{m}$, both in magnitude and orientation. For analysis, we should look at the magnitude ratio and the angle between these two vectors. These quantities can be found using the formulae

$$\theta = cos^{-1} \frac{\overrightarrow{f} \cdot \overrightarrow{m}}{|\overrightarrow{f}||\overrightarrow{m}|} \tag{3.1}$$

$$r = \left| \frac{|\overrightarrow{f}| - |\overrightarrow{m}|}{|\overrightarrow{f}| + |\overrightarrow{m}| + c} \right|, \tag{3.2}$$

where $\overrightarrow{f} \cdot \overrightarrow{m}$ is the dot product of the two vectors and $|\overrightarrow{f}|$ is the length of the vector.

In Figure 3.1, the two parameters for vector similarity are $\theta \cong 0$ and $r \cong 0$. Looking at these values we can conclude that the flow vector and the vector obtained by feature matching coincide well with each other. Such kind of flow and matched vector pairs are marked as inliers.

In the second case, illustrated in Figure 3.2, vectors have similar orientations but

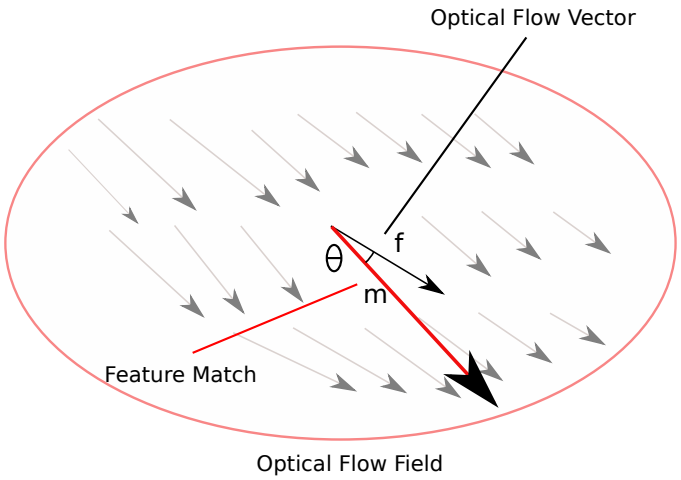


Figure 3.2: Possible optical flow vs feature match vector pairs, f and m resp. - outlier case 1. In this sample, f and m have similar orientations but their magnitudes differ too much. Such matches should be marked as outlier.

their magnitudes differ too much. In other words, we can still say that $\theta \cong 0$ but $r >> 0$. Vector pairs like these should be marked as outliers.

In the last case, illustrated in Figure 3.3, even though $r \cong 1$, the orientation difference is very high being a sufficient reason for marking this pair as an outlier.

Different cost functions can be used for determining whether the match is an outlier. One of the possible functions is a weighted sum of θ and r formulated as

$$C = w_1\theta + w_2 r. \tag{3.3}$$

Feature pairs which have a total cost greater than a threshold can be marked as outliers and those with smaller costs can be accepted as inliers.

A second alternative for outlier determination can be the function

$$C = max(w_0\theta, \frac{r}{\pi}). \tag{3.4}$$

.

In (3.4), if one of the parameters takes a large value, in other words either the magnitude difference or the orientation difference is high, match is marked as an outlier.

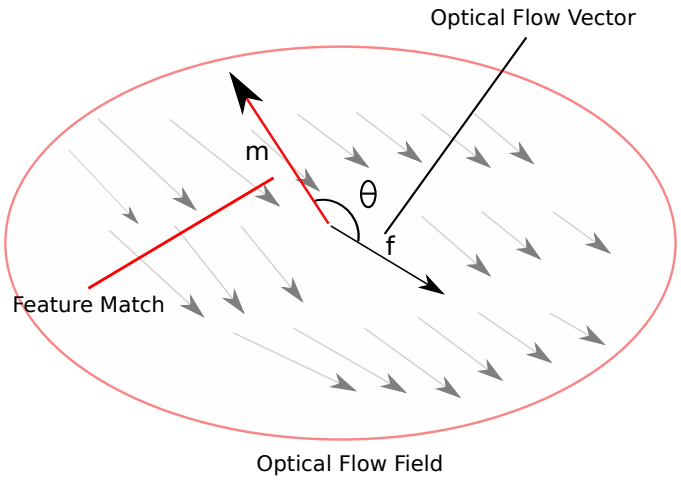
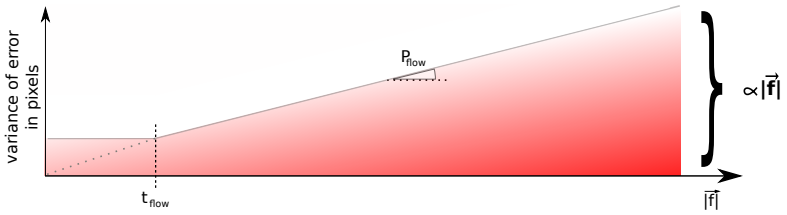


Figure 3.3: Possible optical flow vs feature match vector pairs, f and m resp. - outlier case 2. In this sample, f and m have their magnitudes and orientation different from each other. So such matches should be marked as outlier.

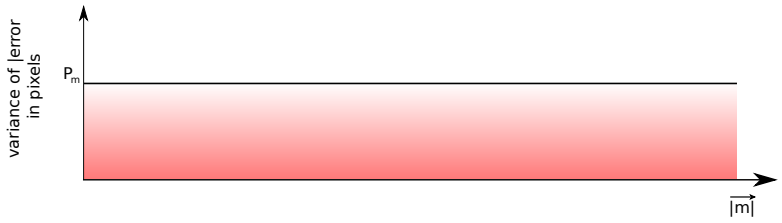
The above functions, (3.4) and (3.3) will give correct estimations in many flow field - feature match pairs. These situations include cases where either $|f| >> 1$, $|m| >> 1$ or both hold. But we try to eliminate outliers depending on the knowledge of optical flow information which might also have errors in it. Furthermore, it is a well known fact that since images are discrete signals, feature localization cannot be realized with perfect accuracy, which is also the case for SIFT extractors. So, these two sources of errors should be considered in order to make the elimination process handle inaccuracies in flow field estimation and feature localization. For situations with $|f| << C$ and $|m| << C$, where C is a real positive scalar, although the ratio of norms might be close to each other, due to errors in feature localization and flow field estimation, $\theta >> 0$ might be the case. Such a high θ value obviously dominates in both of the above functions resulting in the elimination of a correct match. In order to solve this problem, we propose the following method for checking matches.

Firstly, we have to determine error models for both the flow vector estimation and feature match vectors. For both of these, there are two independent dimensions in which errors can exist, namely their norms and orientations. These error models are plotted in Figures 3.4 and 3.5.

In these figures, $|\overrightarrow{f}|$ is the norm of the estimated flow vector and $|\overrightarrow{m}|$ is the norm

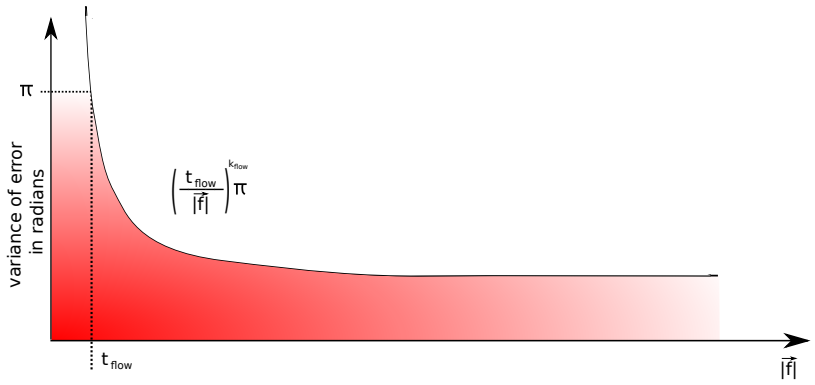


(a) Norm error model for flow vector estimation. $|f|$ is the norm of the calculated flow vector. Variance of error is the expected error in the norm of the flow vector. This plots says, up to a certain flow norm, t_{flow}, there is constant error. After that threshold, expected error in the norm increases with the increase in the calculated flow norm. The relation between error and the flow norm is assumed to be a constant multiplier, P_{flow}

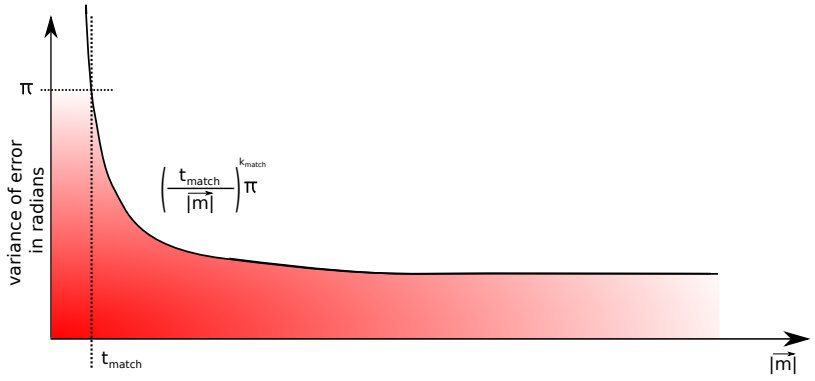


(b) Norm error model for feature localization. $|m|$ is the norm of the match vector connecting the two matched features. Variance of error is the expected error in the norm of the flow vector. The origin of error is the sub-pixel localization while extracting features. So the norm of match vector, whose end points are the two matched features, can miscalculated at most as much as the feature localization error. For this reason, error variance has been taken to be constant independent from the match vector norm.

Figure 3.4: Norm error models for flow vector estimation and feature localization



(a) Orientation error model for flow vector estimation



(b) Orientation error model for feature match vector

Figure 3.5: Orientation error models for flow vector estimation and feature match vector. These figures show that for both of flow and match vectors, when they are below a threshold (t_{flow} and t_{match} resp.) expected error in the orientation peaks. This is due to that if their norms are very small compared to the expected error values, small errors in one of the end point locations of these vectors, result in great changes in their orientations.

of the vector connecting matched pairs of features. Since many of the optical flow algorithms [6] [2] [23] [22] [30] use differential methods which utilize linearization of nonlinear functions and iteratively estimate the flow vector, there is a higher possibility of faulty norm estimation for larger displacements, hence more error for larger flow vectors. This behavior is depicted in Figure 3.4. However, the proposed error model for feature match vector asserts that the error in the norm of the match has constant variance, i.e. it does not change with how far the matched pairs of features are. Closed form definitions for the these error models are

$$\sigma_{|\overrightarrow{f}|} = \left\{ \begin{array}{ll} P_{flow} * |\overrightarrow{f}| & \text{if } |\overrightarrow{f}| > t_{flow} \\ P_{flow} * t_{flow} & otherwise \end{array} \right\} \tag{3.5}$$

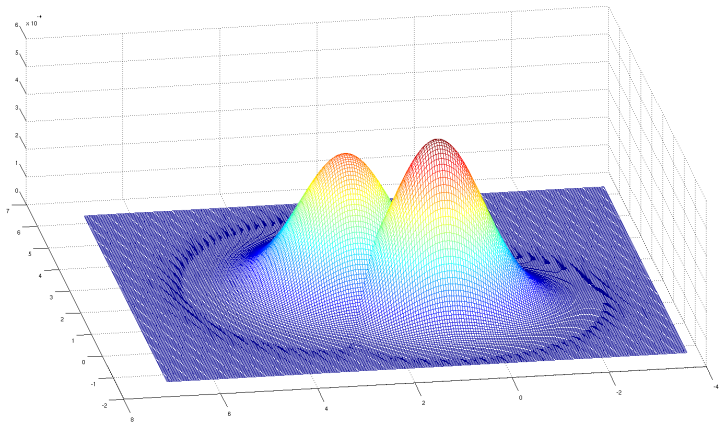
$$\sigma_{|\overrightarrow{m}|} = P_{match}. \tag{3.6}$$

In contrast to the increase in the norm of the error with $|\overrightarrow{f}|$, for orientation, the error variance becomes smaller with increasing norm of the flow vector. However, since for short flow vectors a small change in the Cartesian position of endpoint of the flow vector will cause great change in its orientation, and expected error amount spans the whole $[-\pi, \pi]$ range. Similar considerations are applicable to error variance in orientation of feature match vectors. These models can be formulated as

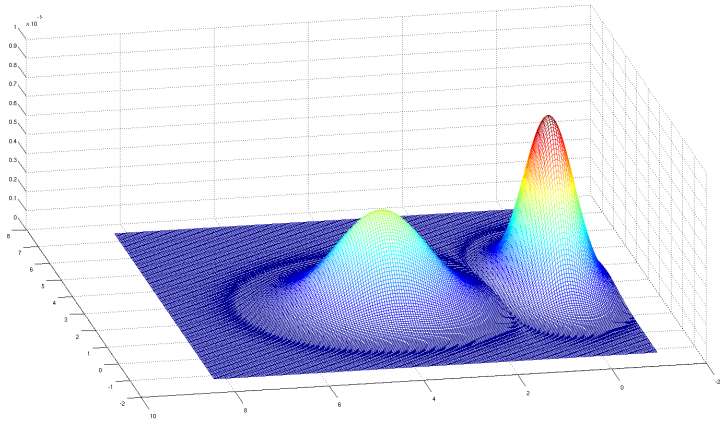
$$\sigma_{\angle\overrightarrow{flow}} = \left\{ \begin{array}{ll} \pi \left(\frac{t_{flow}}{|\overrightarrow{f}|} \right)^{k_{flow}} & \text{if } |\overrightarrow{f}| > t_{flow} \\ \pi & otherwise \end{array} \right. \tag{3.7}$$

$$\sigma_{\angle\overrightarrow{match}} = \left\{ \begin{array}{ll} \pi \left(\frac{t_{match}}{|\overrightarrow{m}|} \right)^{k_{match}} & \text{if } |\overrightarrow{m}| > t_{match} \\ \pi & otherwise. \end{array} \right. \tag{3.8}$$

Having the error models for flow and match vectors, we can define distributions representing the probability of these vectors' actual head positions and orientations in the image plane. Sample distributions are shown in Figure 3.6. In the perfect match case, both flow and match vectors should have the same means and uncertainties. As their means get separated from each other, their likelihood to be correct becomes less. Obtaining such a perfect match is obviously not likely



(a) In this sample, two distributions have their means close to each each other and their uncertainty regions overlap. When Equation 3.11 is applied, their distance yields to be d_{KLD}=2.48.



(b) In this sample, two distributions' means are distant and their uncertainty regions do not overlap much which result in a large KL distance of d_{KLD}=16.53

Figure 3.6: Sample distributions for norms and orientations of flow and match vectors. In this figure, among many possibilities, two cases are given and these show the result of Equation 3.11 applied to close and distant Gaussian distributions.

to happen. But, although their means fall away from each other, their uncertainty regions may overlap. For this reason, when comparing the two vectors, we include their variances into the calculation too. In comparing two distributions, Kullback-Leibler divergence [27] has been used, which is a non-symmetric measure of the difference between two probability distributions. For distributions P and Q of a continuous random variable, KL-divergence is defined as

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) log \frac{p(x)}{q(x)} dx. \tag{3.9}$$

For discrete P and Q distributions, KL-divergence is

$$D_{KL}(P||Q) = \sum_{i} P(i) log \frac{P(i)}{Q(i)}. \tag{3.10}$$

As can be seen from the Equations (3.9) and (3.10), KL-divergence is not symmetric. In other words, the KL-divergence from P to Q is not necessarily the same as the KL-divergence from Q to P. In order to eliminate this asymmetry, we used a modified KL-divergence metric which is

$$D_{KL}(P, Q) = \frac{1}{2}(D_{KL}(P||Q) + D_{KL}(P||Q)). \tag{3.11}$$

Since optical flow gives displacements between the previous frame and the current frame, we can only apply this filter to features those were observed in the previous frame. Features, which were not visible in the previous frame, but whose matches are outliers, cannot be filtered with this method. One way of extending this method to include such features might be tracking features in the last N frames. If a feature was not observed in the previous frame, but observed in one of the older frames, by this extension, such features can be included in the filtering process too.

3.3.2 Outlier Elimination Using Prediction Error

As explained in Section 3.1, matching algorithms do not use spatial information of interest points; but only use their descriptors in identification. In Section 3.3.1, we proposed a method which utilizes spatial information together with the optical flow information for outlier elimination. However, there is another case which may result in a wrong feature match, and cannot be detected with the aid of optical flow information.

Consider two landmarks L_1 and L_2, which are distant from each other; but their corresponding interest points, f_{L_1} and f_{L_2}, have similar descriptors. Using only descriptors, it is very likely to mismatch these two interest points. Since it may be the case that optical flow vector calculated at the projected pixel location (p_{L_i}) of a landmark may be similar, in magnitude and orientation, to its match vector; but their pixel locations may be distant from each other, optical flow aided outlier eliminator will not be able to filter out this false match. In order to eliminate such a false match, we should consider the expected projected location of the landmark together with the extracted feature's location. This scenario is depicted in Figure 3.7. Looking at the Euclidean distance of these two points might be intuitive, but although extracted feature's location has little uncertainty, projected landmark location may have a large uncertainty. This comes from the nature of EKF-SLAM that every landmark state has its mean and uncertainty (in our case an ellipsoidal volume). When current landmark estimate is projected to the image plane, together with its mean, this uncertainty region is also projected as an elliptical region. This uncertainty is represented with a 2×2 covariance matrix which intuitively guides us to use Mahalanobis distance. The projected covariance matrix is found as

$$S_i = \frac{\partial u}{\partial x_v}\Sigma_{xx}\frac{\partial u}{\partial x_v}^T + \frac{\partial u}{\partial x_v}\Sigma_{xy_i}\frac{\partial u}{\partial y_i}^T + \frac{\partial u}{\partial y_i}\Sigma_{y_i x}\frac{\partial u}{\partial x_v}^T + \frac{\partial u}{\partial y_i}\Sigma_{y_i y_i}\frac{\partial u}{\partial y_i}^T + R. \quad (3.12)$$

The Mahalanobis distance between projected landmark location, p_{L_i}, and matched feature location f_{L_i} is

$$d_{Mah} = (p_{L_i} - f_{L_i})S^{-1}(p_{L_i} - f_{L_i})^T. \quad (3.13)$$

We expect d_{Mah} to be smaller that a threshold in order to accept the match. It is obvious that, with landmarks already initialized, S will be larger allowing for more error in matching.

In this chapter, we have introduced three methods for matching features, which are *SIFT Matching*, *Married Matching* and *Minimum Distance Matching*. All of these methods use the descriptors attached to feature points, but do not used spatial information. But in VSLAM context we have further constraints to decide whether a feature pair is correctly matched or not. One of these constraints is optical flow field information, which says a feature seen in consecutive frames should follow the optical flow field. The second one asserts that when a map exists, a feature matched to an already initialized landmark should not fall apart from its expected projected location. These two principles can be used to eliminate outliers in VSLAM applications to improve its performance and prevent catastrophic failures.

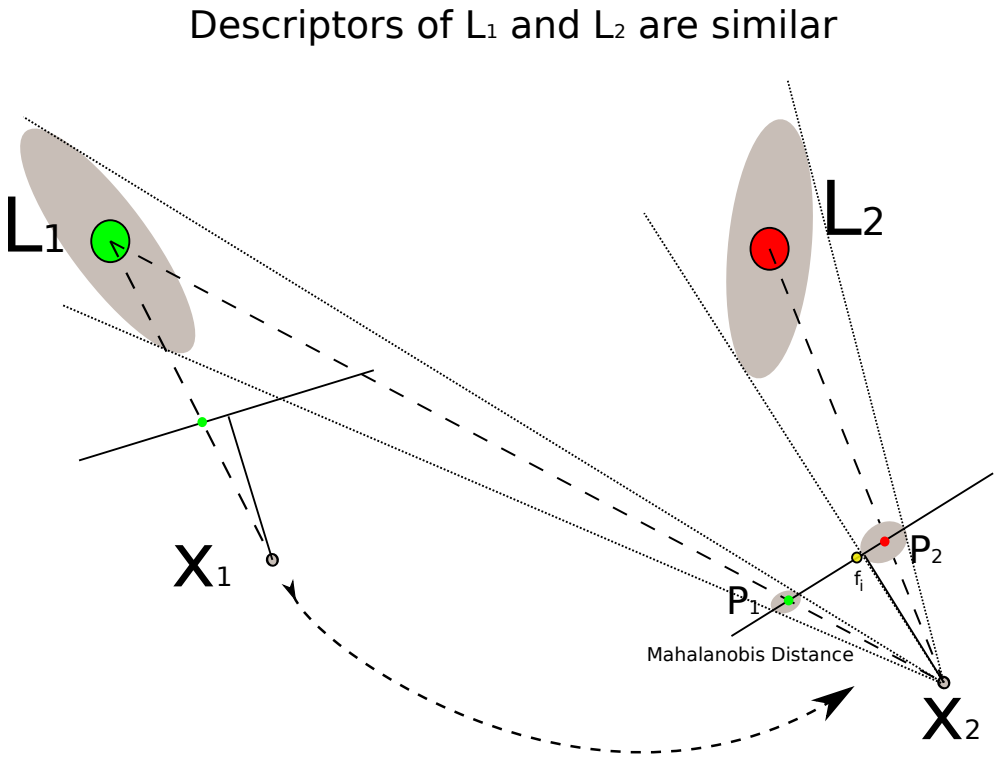


Figure 3.7: Outlier elimination using prediction error. In the figure, robot moves from pose x_1 to x_2. In both poses, robot sees landmark L_1, and at x_2 it also sees L_2 with both of the landmarks having uncertainties. If the two landmarks have similar feature descriptors, it is very likely to make a mismatch at x_2. Suppose the case that at x_2, extracted feature f_i is matched with L_1 which is a wrong match. When Mahalanobis distance between f_i and P_1 is calculated, distance will probably be greater than the eliminator threshold. But if f_i matches with L_2, which is a correct match, Mahalanobis distance between f_i and P_2 will be smaller than the threshold. This way, when f_i is matched with the wrong one of the similar L_1 and L_2 landmarks, match is marked as an outlier.

Chapter 4

Evaluation

In Chapter 4, two methods for eliminating outliers were introduced. One of them was using optical flow information as reference and relying on mismatches between predicted and observed displacements to detect whether they are correct or false matches. The second one utilizes the knowledge of the map in order to filter out erroneous data associations. In this chapter, we will show results of several experiments, testing both of these methods. These experiments realize different scenarios, including simulated and real image sequences, with and without maps. Our implementation was based on an existing VSLAM library [10].

Table 4.1 shows a list of test scenarios, elimination algorithms and data types used.

Table 4.1: Summary of outlier eliminator test scenarios and associated sections

	Outlier Elimination Methods			Data Type	
Section	No Elim.	OF Aided	Pred. Error Based	Real	Synthetic
4.1.1	✓	✓			✓
4.1.2	✓	✓		✓	
4.2.1	✓			✓	
4.2.2		✓		✓	
4.2.3		✓	✓	✓	

Throughout all experiments with real data, SIFT features were used as interest points [29, 45]. In computing the optical flow, the method described in [7] was

used. Maps were computed using available Matlab sources for the algorithm presented in [10]. Figure 4.1 shows a layout of the complete system, i.e. the VSLAM process with prediction error based and optical flow aided outlier elimination. In base case tests, optical flow calculation and outlier elimination processes are omitted; and when outlier elimination is done only with prediction based outlier eliminator, optical flow is not calculated.

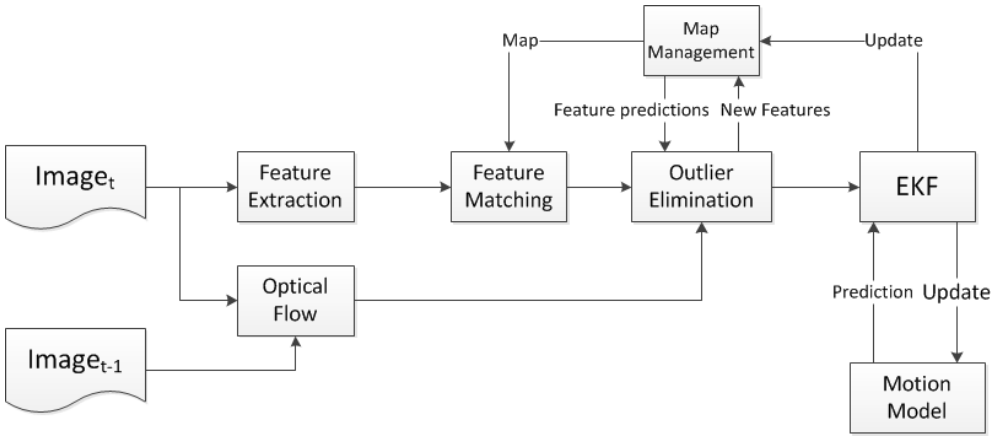


Figure 4.1: A layout of VSLAM with prediction error based and optical flow aided outlier elimination process

4.1 Outlier Elimination without a Map

In the following sections, test scenarios will be explained and their results will be presented. In order to save space, abbreviations were used for results. In Table 4.2, these abbreviations and their descriptions are given (all values are percentages). Also, the relation between these terms are depicted in Figure 4.2.

4.1.1 Synthetic Data

For synthetic data experiments, artificial data consisting of feature matches and a flow vector field have been generated in a simulation environment. A sample screenshot of the simulation environment is shown in Figure 4.3. In this simulation, the robot follows a manually defined path in 3D space, navigating through

Table 4.2: Abbreviations, descriptions and mathematical relations for metrics used in performance evaluation of Optical Flow Aided Outlier Elimination

Abbreviation	Description	Mathematical Relation
GTCM	Ground truth correct matches	GTCM + GTWM = 1
GTWM	Ground truth wrong matches	
TN	Unfiltered correct matches (true negatives)	FP + TN = GTCM
FP	Wrong outlier eliminations (false positives)	
TP	Correct outlier eliminations (true positives)	TP + FN = GTWM
FN	Unfiltered wrong matches (false negatives)	
IL	Information lost	IL = FP / GTCM
ES	Elimination success	ES = TP / GTWM

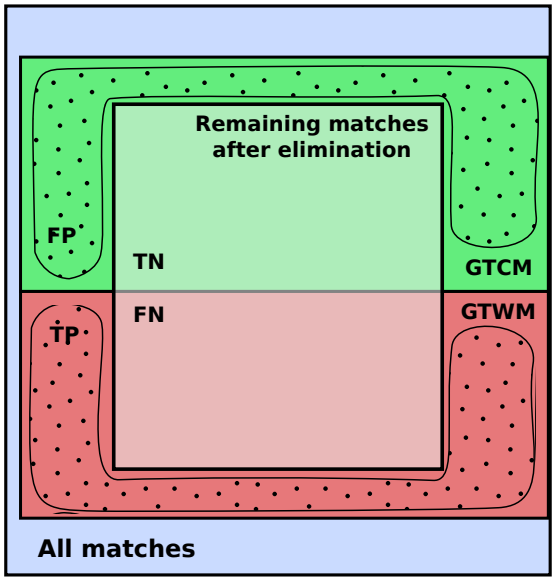


Figure 4.2: Confusion diagram showing the relation between outlier elimination performance metrics described in Table 4.2

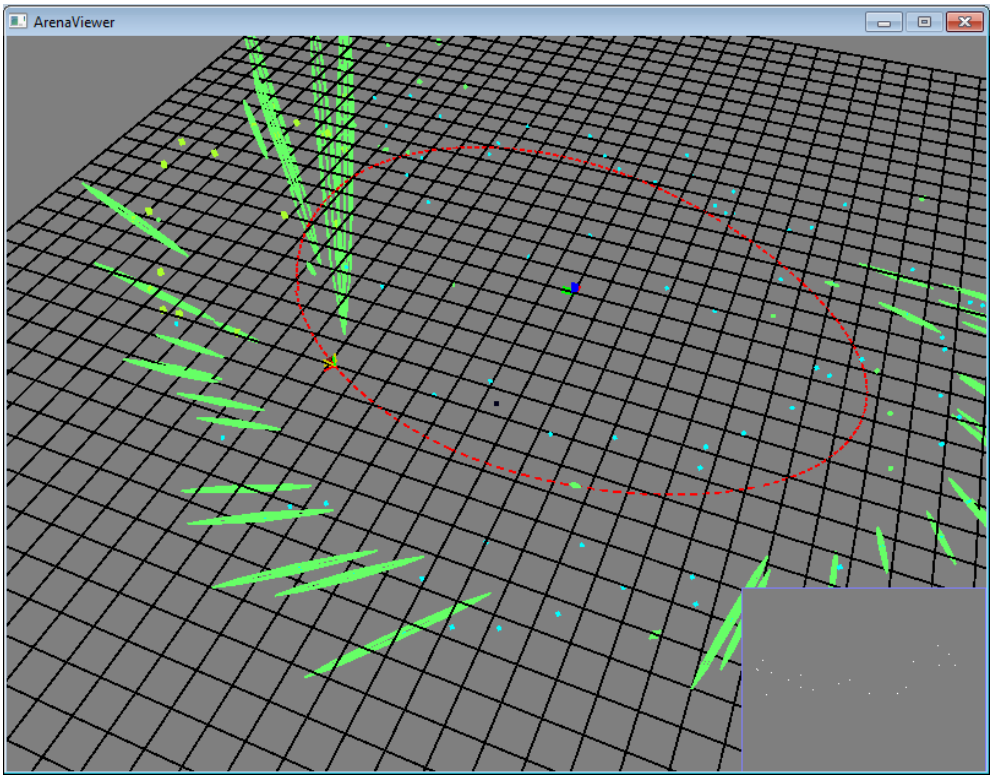


Figure 4.3: Simulation environment used in synthetic data generation

a set of landmarks, also defined manually. A camera model with mustache distortion was used for projecting landmarks. Since this test was done in a simulation environment, ground truth matches and flow vectors were known. However, in order to make the scenario more realistic, matches were disturbed with three different noise models, and optical flow vectors are approximated with a simple spherical environment assumption. We have disturbed the feature matches with three different noise models, which are

1. Swapping match pairs,
2. Modifying the location of the matched features,
3. Matching with non-existing features.

The first case simulates mismatches which can occur when descriptors of two features are very similar to each other. For example assume that in the n^{th} frame we have two features, f_1, f_2, which should be matched to f_1', f_2' in the $(n+1)^{st}$ frame. But assume that, since the descriptors of f_1 and f_2 are similar to

each other, their corresponding features in the $(n+1)^{st}$ frame will have similar descriptors too. In such a case, there is the possibility of matching f_1' to f_2 and f_2' to f_1.

The second noise model simulates errors in feature localization. It is a known fact that, due to noise, intensity and viewpoint changes, the pixel location of a feature can be slightly different than what it should be. By adding noise to the position of the matched feature, we aim to simulate this type of error. Finally, for the third type of noise, non-existing features were generated and some features were matched with these artificial features. In real scenarios, sometimes the descriptor of an unrelated feature may be closer to a feature than its correct match. Usually, such false matches are completely random in terms of their pixel locations.

Since we work in a simulation environment, we could extract the exact optical flow vector field for each feature location, but this would be very unrealistic. For this reason, we propose the following simple method for approximating the flow vectors. Assume that the camera travels from pose x_1 to x_2. Through its movement, it observes a landmark L at pixel locations p_1 and p_2. The vector from p_1 to p_2 is the exact optical flow vector. Rather than this, we back project p_1 onto a sphere centered at P_{sph} with a radius R_{sph} to give L'. In the next frame, we project L' and obtain p_2', then use p_1 and p_2' to approximate the flow vector for landmark L. P_{sph} is taken to be the current position of the robot, and R_{sph} to be the mean depth of the visible landmarks from the current robot state. This method is depicted in Figure 4.4.

We tested our optical flow aided outlier elimination method in a single dataset with three different parameter sets. Table 4.3 shows the results for these three tests. As can be seen in Table 4.3, all erroneous data has been eliminated, meaning that, after applying the filter, we no longer have feature mismatches. However, due to our approximate optical flow, some correct matches (approximately 15%) were eliminated as well. In order to improve this situation, we have modified eliminator parameters, decreasing false alarms.

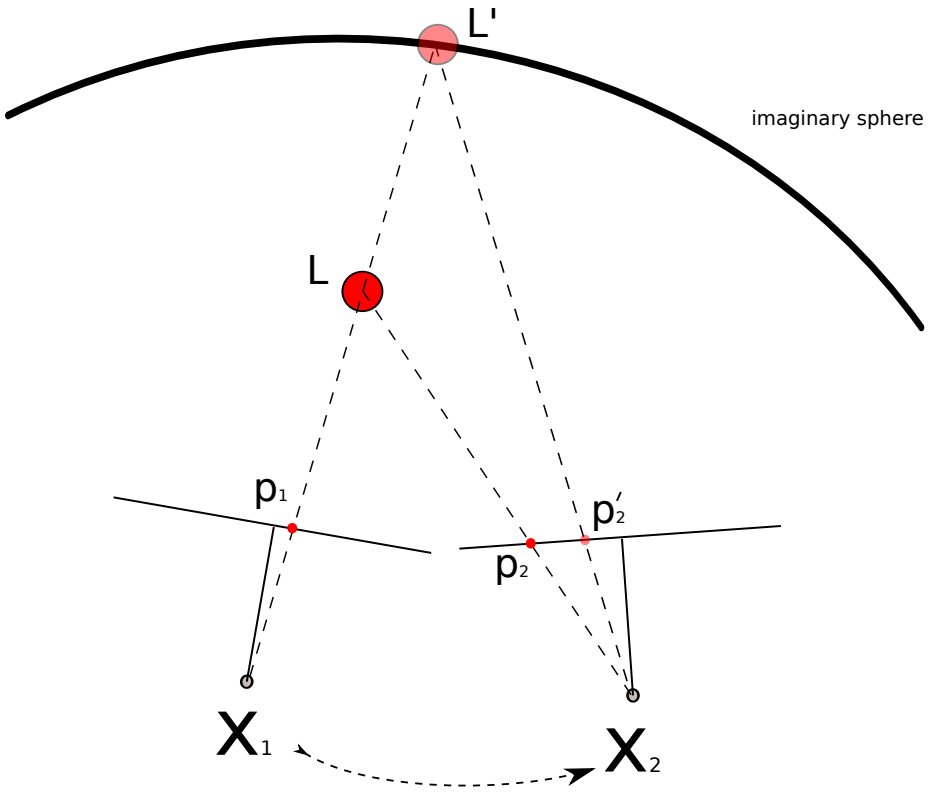


Figure 4.4: Method used in optical flow approximation for synthetic data

Table 4.3: Results of optical flow aided outlier elimination applied on synthetic data

Average Values(%)	$1^{st}Test$	$2^{nd}Test$	$3^{rd}Test$
GTCM	50.96	50.96	51.38
GTWM	49.04	49.04	48.62
TP	49.04	49.04	48.62
FP	6.6	4.34	1.24
FN	0	0	0
TN	44.36	46.62	50.14
IL	13.02	8.55	2.36
ES	100	100	100

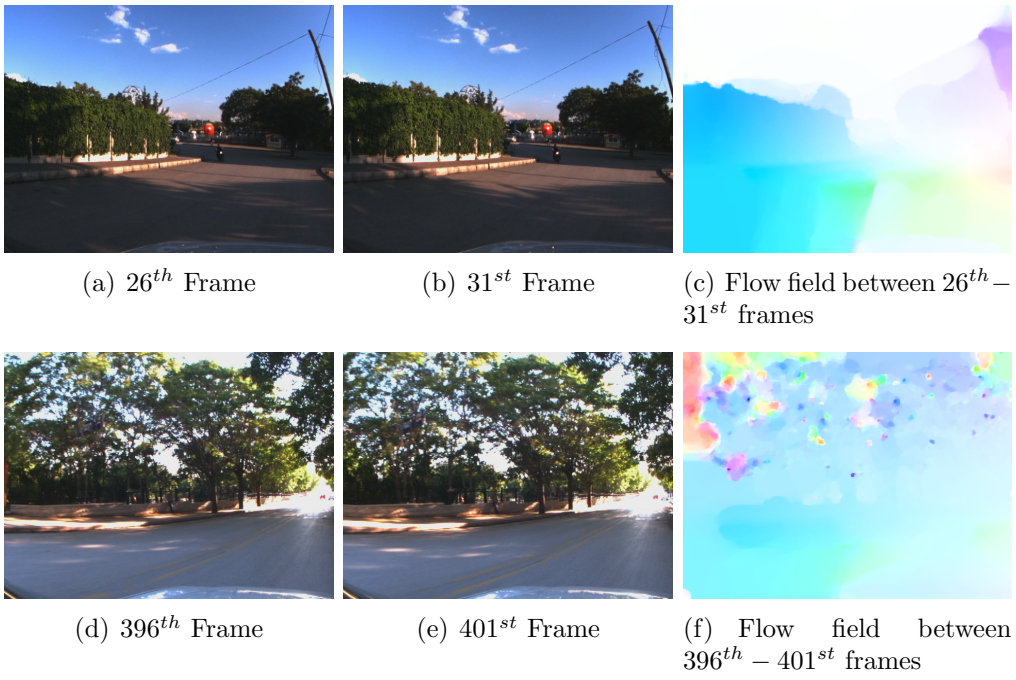
4.1.2 Real Data

These experiments test optical flow aided outlier elimination with real image sequences. Dataset were collected at 30 fps using a FLEA2 camera with a wide angle lens mounted on top a car driven in urban area. Sample frames from the dataset and optical flow images between these frames are shown in Figure 4.5. In Figure 4.6, color codes for optical flows are shown. As can be noted from this figure, rather than using consecutive pairs of images, images separated by five frames were used to compute optical flow and feature matches. The reason for this is that at 30 fps, there is little movement and correctly matched features are very close to each other. In such cases, the performance of the elimination algorithm would not be tested with moderate length flow and match vectors, and majority of the these vectors would be very small in length, easily passing the elimination test.

In the elimination process, we take the optical flow field as a reference and compare match vectors to it. However, images with repetitive textures do not give good results when fed to the optical flow calculation algorithm. An example can be seen in the second row of Figure 4.5. The corresponding flow image, shows how the flow field estimation diverges from the expected field in the regions occupied by trees.

In this dataset, there are two phases, differing with accuracy of optical flow calculation. From the start to the 250^{th} frame, the flow field was more accurate than it was for frames 252^{nd} to the end. The reason behind this was, the clear texture in the first phase and repetitive texture in the second phase due to trees. Since we compare the match vectors with flow vectors, errors in the flow calculation would yield wrong results in the elimination process. Consequently, in these two phases we expect slightly different success rates. Table 4.4 show some global statistics on average values and separate statistics for the two different phases phases.

Ideally, the eliminator should detect all wrong matches and should not eliminate any correct match as outliers. In other words, we expect “FP=0” and “TP=GTWM”. As can be seen from Table 4.4 these two constraints are approximately satisfied.



(a) 26^{th} Frame (b) 31^{st} Frame (c) Flow field between 26^{th} – 31^{st} frames

(d) 396^{th} Frame (e) 401^{st} Frame (f) Flow field between 396^{th} – 401^{st} frames

Figure 4.5: Several frames and optical flows fields from car_dataset1

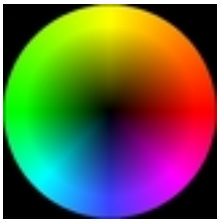


Figure 4.6: Optical flow vector color codes. Direction of the flow is coded with colors, and the magnitude is coded with intensities.

Table 4.4: Optical flow aided outlier elimination results without map applied on real data

Average Values(%)	Whole Sequence	1^{st} Phase	2^{nd} Phase
GTCM	96.77	97.66	96.16
GTWM	3.23	2.34	3.84
TP	3.12	2.16	3.79
FP	4.37	1.17	6.59
FN	0.11	0.18	0.06
TN	92.4	96.49	89.56
IL	5.16	1.21	9.50
ES	94.84	91.22	97.37

4.2 Outlier Elimination with a Map

In Section 3.3.1, outlier elimination was carried using only optical flow information and no map. In this section, we apply the two different outlier elimination algorithms to the VSLAM problem and investigate the chances in performance. Since VSLAM, also estimates camera pose and landmark locations, we now have additional information to use in the elimination process. As described in Section 3.3.2, when the pose of the robot and location of the landmarks are available, we can also make estimates of pixel locations on which landmarks should be reobserved. If matched features lie far from their estimated locations, we can conclude that these matches are outliers.

In the following tests, we apply prediction error based and optical flow aided outlier elimination algorithms to four different datasets. The sequence of application of these algorithms is as:

1. Performance of VSLAM in absence of any outlier filters
2. Prediction error based outlier elimination applied
3. Optical flow aided outlier filter applied together with the previous filter

Name of the datasets used the following tests are

1. *car_dataset*1 (Figure 4.7),
2. *lab_dataset*1,
3. *lab_dataset*2 (Figure 4.8),
4. *lab_dataset*3 (Figure 4.8).

4.2.1 VSLAM without Outlier Elimination

In these tests, we did not use any outlier eliminators and used only *Married Matcher* for data association. All of the tests resulted in divergence of the VSLAM algorithm due to successive data association errors for large percentages of the features. In Figures 4.9(a), 4.10(a), 4.11(a) and 4.12(a), estimated robot paths

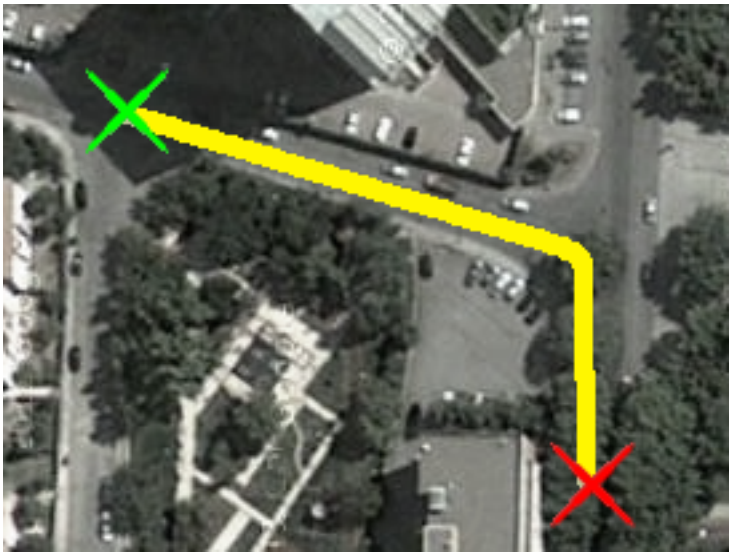
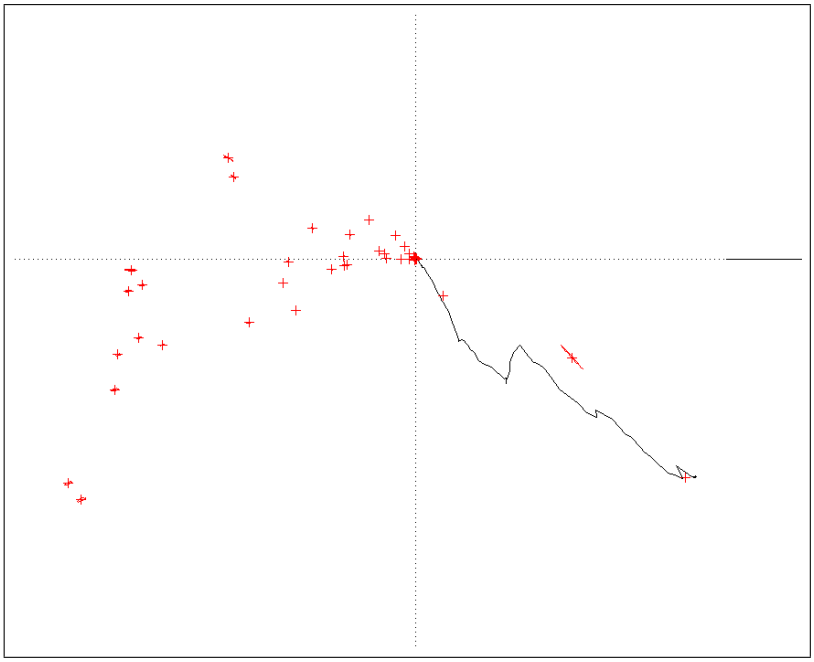


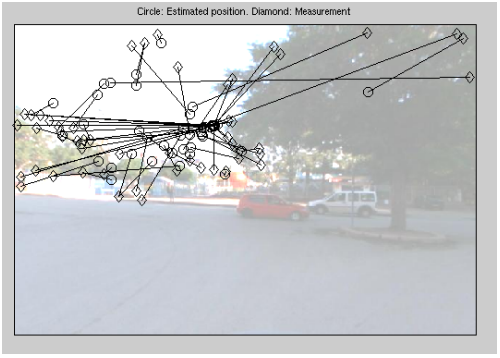
Figure 4.7: Google Earth image showing the path followed in car_dataset1

Figure 4.8: Google Earth image showing the path followed in lab_dataset2 and lab_dataset3

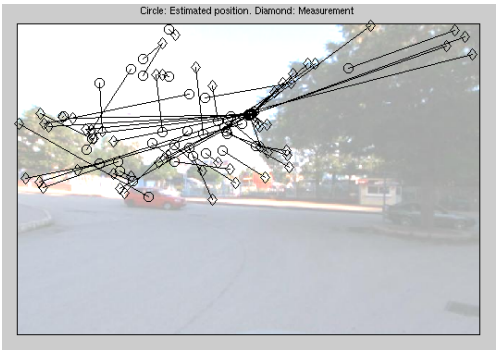
are plotted. When compared to Figures 4.7 and 4.8, it is obvious that VSLAM's path estimates, and the maps, are far from what they should be. If we look at sample screenshots in Figures 4.9, 4.10, 4.11 and 4.12, it can be seen why such catastrophic fails occurred. In these screenshots, lines connecting image pixels are match vectors, showing that very large errors in conflicting directions occur, destroying stability.



(a) Path estimate for car_dataset1 zoomed for base case



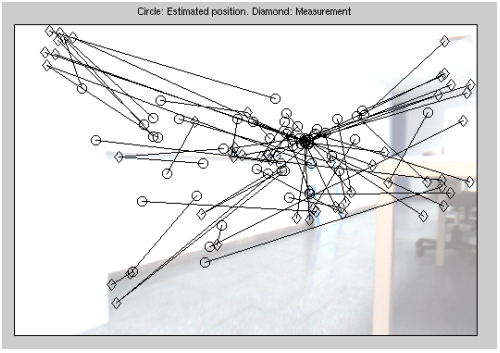
(b) Frame 199



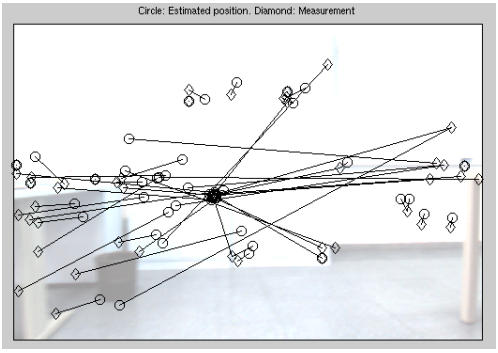
(c) Frame 176

Figure 4.9: Estimated path and several frames and feature match vectors from car_dataset1 for base case. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

(a) Path estimate for lab_dataset1 zoomed for base case

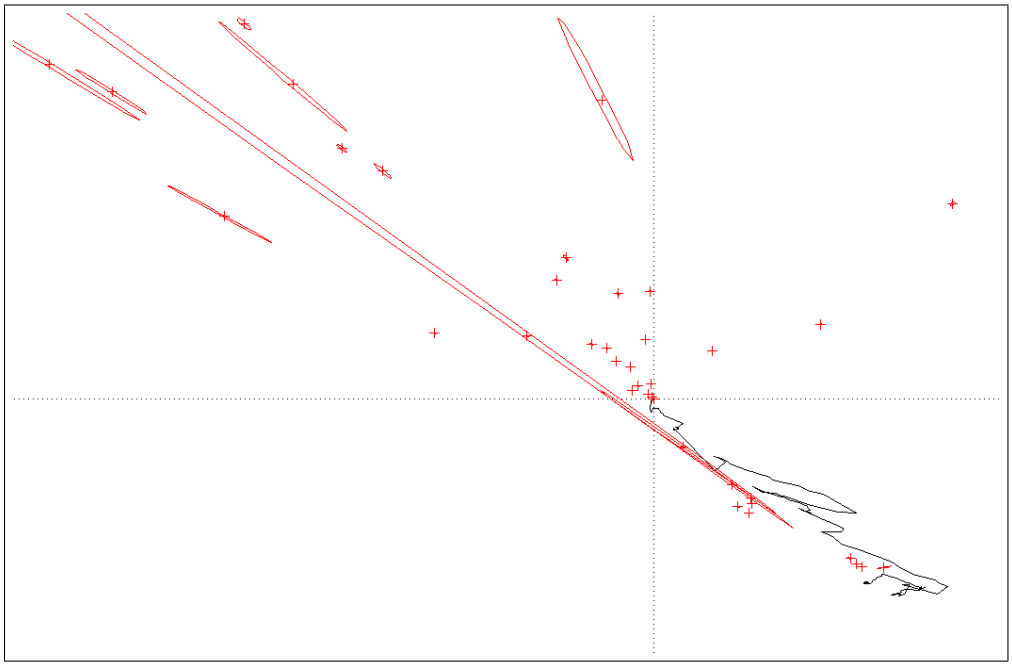


(b) Frame 847

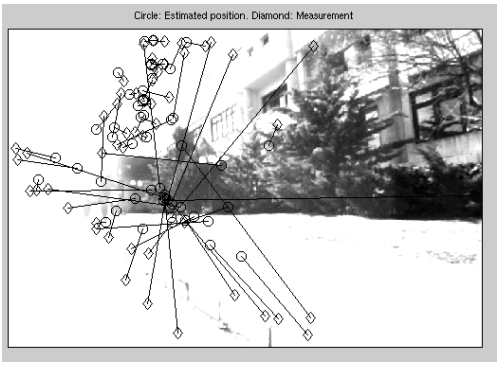


(c) Frame 767

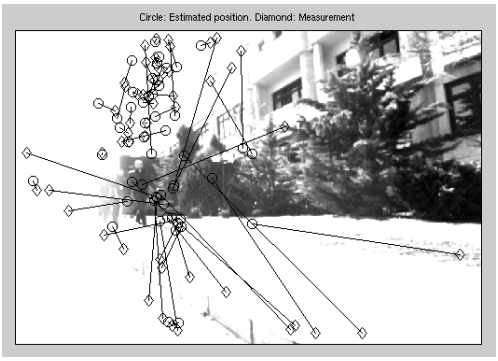
Figure 4.10: Estimated path and several frames and feature match vectors from lab_dataset1 for base case. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.



(a) Path estimate for lab_dataset2 zoomed for base case

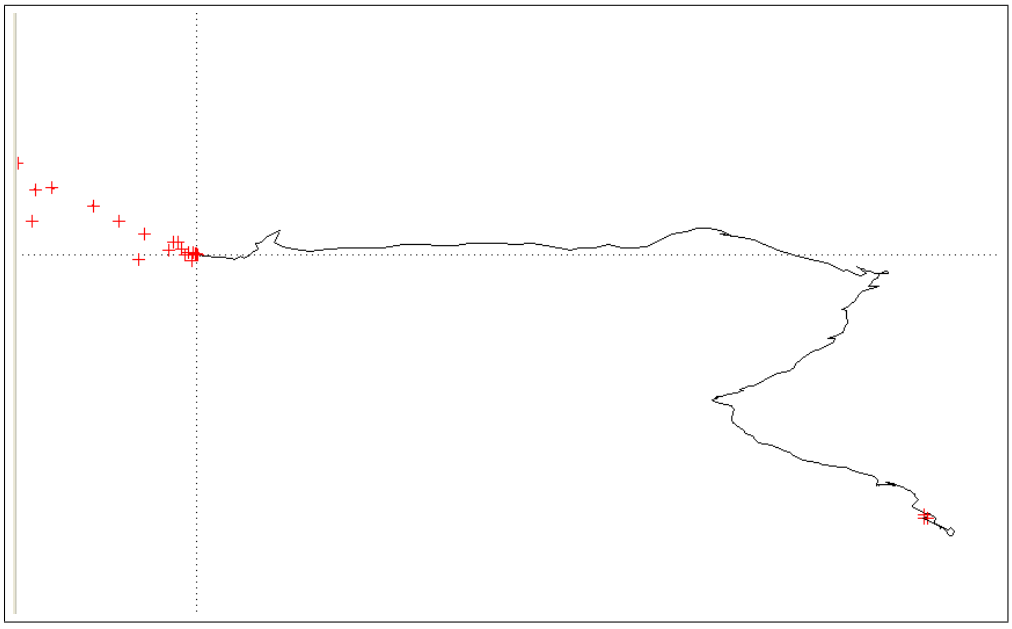


(b) Frame 3098

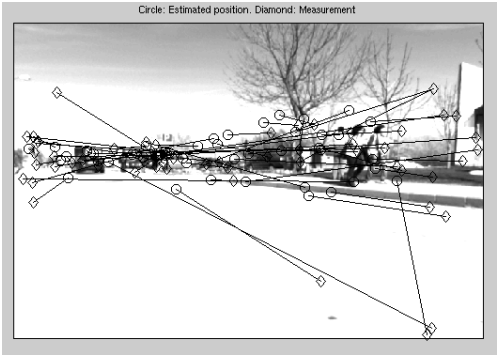


(c) Frame 3138

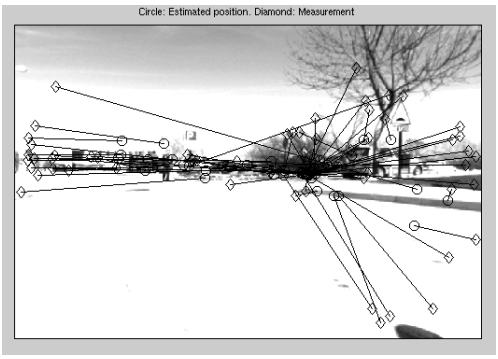
Figure 4.11: Estimated path and several frames and feature match vectors from lab_dataset2 for base case. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.



(a) Path estimate for lab_dataset3 zoomed for base case



(b) Frame 4038



(c) Frame 4168

Figure 4.12: Estimated path and several frames and feature match vectors from lab_dataset3 for base case. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

4.2.2 VSLAM with Prediction Error Based Outlier Elimination

In this section, we consider an outlier eliminator based on feature location prediction error. As can be seen from Figures 4.13(a), 4.16(a), 4.19(a) and 4.22(a), estimated robot paths are closer to ground truth paths depicted in Figures 4.7 and 4.8. Looking at these figures, we can roughly conclude that the prediction error based outlier eliminator works well. However, as can be seen from Figures 4.13, 4.16, 4.19 and 4.22, showing instances with largest average projection errors, there are still some outliers that were not eliminated. As in previous tests, effect of these outliers on the update stage of SLAM is deteriorating the belief of the robot rather than improving it. However, since the number of outliers is a low percentage of all matches, these effects are very small compared to those of Section 4.2.1.

The estimated path for *car_dataset1* obviously resembles more to its ground truth path (Figure 4.7) than the paths of other datasets which include sharper discontinuities. There may be various reasons for such a performance difference. One reason for that may be that *car_dataset1* includes less repetitive textures, resulting in less descriptor similarity between features extracted throughout the whole sequence. Frames in this dataset include a number of different regions such as pavement, tarmac, trees, creepers, clouds and park which result in richer texture variety. However, the other three datasets do not have frames with that much texture variety. For example, *lab_dataset1* has been collected in a laboratory with walls and furnitures which have little texture variety. Also, frames in *lab_dataset2* and *lab_dataset3* contain repetitive textures of faculty building, tarmac and clear sky resulting in poor data association and worse path estimates.

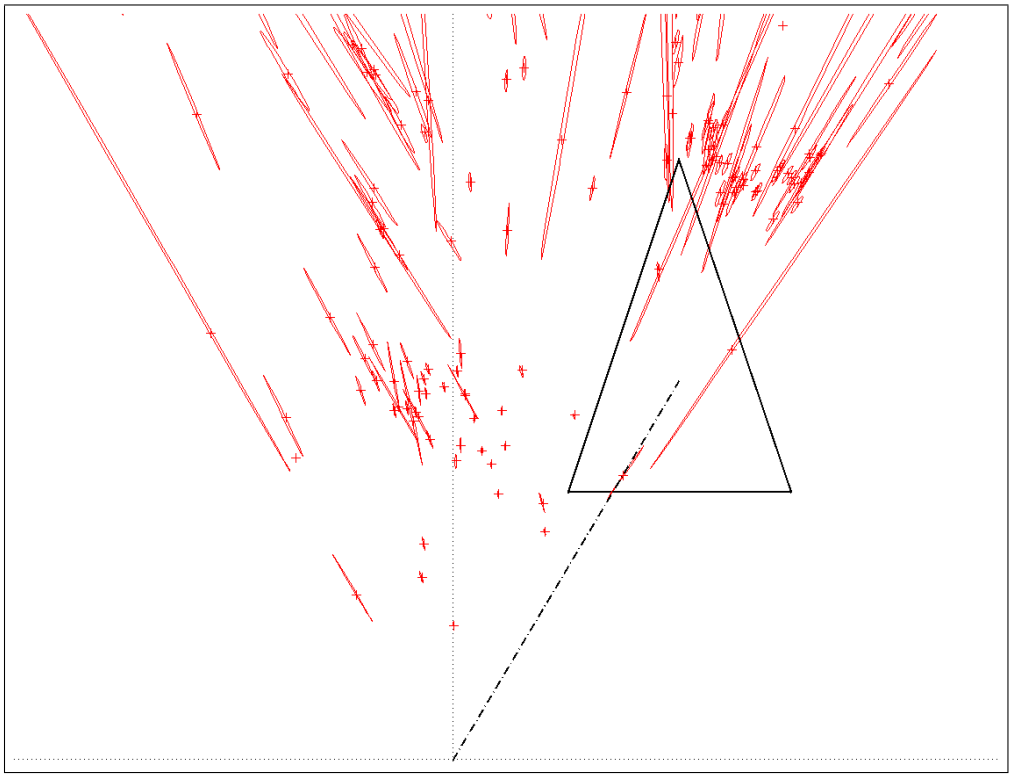
Another reason for the quality of estimated paths may also be average depth of visible regions, rather than data association problems. For example, landmarks on clouds give almost no information about the translational motion. To express generally, if distant landmarks dominate close-to-middle range landmarks in number, position estimation fails. In *lab_dataset2* and *lab_dataset3*, both sequences consist almost only of such scenes. Also, landmarks that are too close

are usually not visible for long periods, and cannot contribute to the belief before their position estimations get sufficiently certain. So, after traversing short paths through the test area, such landmarks disappear and new ones are added to the map with high uncertainty, which cannot contribute to pose estimation much. One reason for the performance degradation in *lab_dataset1* may be that such situations occur very often.

Figures 4.13, 4.16, 4.19 and 4.22 are example frames with largest projection errors throughout all sequences. Compared to the base case (Figures 4.9, 4.10, 4.11 and 4.12), the improvement from using projection error based outlier eliminator is clear. In these tests, false matches are usually done with landmarks that are recently initialized. To eliminate a match, Mahalanobis distance between the landmark's estimated projected location and the feature's location should be greater than a threshold. However, recently detected landmarks have large uncertainties, so the projected uncertainty of such landmarks cover a larger image area. So the Mahalanobis distance between matched pairs may be very low, even though distance between their pixel locations may be large.

Figures 4.14, 4.17, 4.20 and 4.23 show average projection errors in pixels versus frame numbers. Ideally, these errors should be very close to zero. However, outliers and rapid camera movements cause larger projection errors. For example, *car_dataset1* was collected with a camera mounted on top of a car, which is not affected much by irregularities in the road. But *lab_dataset2* and *lab_dataset3* were collected with a two wheeled trolley, magnifying the effect of even small pebbles. These disturbances change the orientation of the camera significantly, so projection errors come up to be large.

Figures 4.15, 4.18, 4.21 and 4.24 show the ratio of outliers to all matches. Ground truth rates for correct matches were manually extracted from each frame. These figures show strong relation, to their corresponding projection error plots from which we can conclude that average error is mainly due to outliers. In the absence of outliers, we can expect 2-6 pixel errors, which is acceptable for a camera not moving rapidly and capturing at 30fps. Why outliers diminish in certain ranges of frames may be, through these frames camera sees landmarks which have been tracked enough frames to make their estimates certain, so that for smaller match vector norms, Mahalanobis distance yields to be higher than the threshold.



(a) Path estimate for car_dataset1 zoomed with prediction error based outlier elimination. Triangle represents the robot with its sharp corner indicating its heading. Plus markers indicate landmark locations with associated uncertainty ellipses.

(b) Frame 45

(c) Frame 194

Figure 4.13: Estimated path and several frames and feature match vectors from car_dataset1 with prediction error based outlier elimination. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

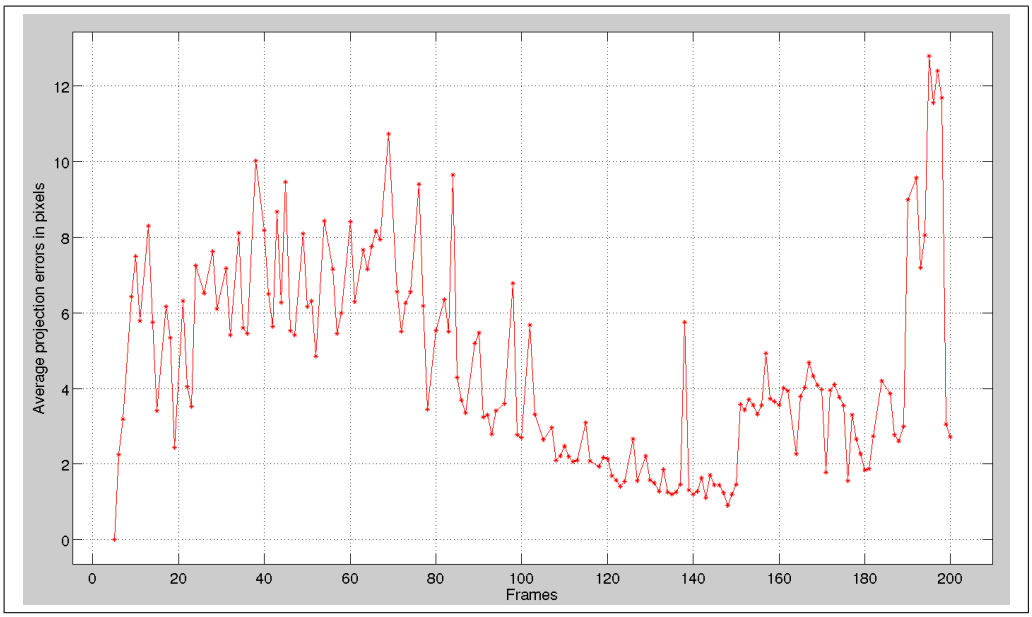


Figure 4.14: Average projection errors vs frames for car_dataset1 with prediction error based outlier elimination

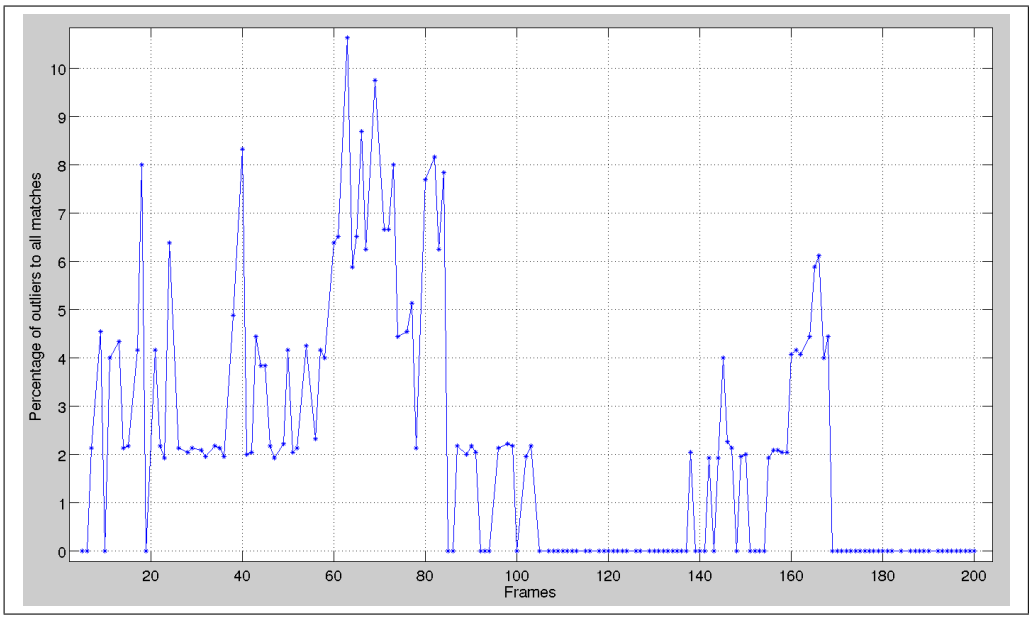
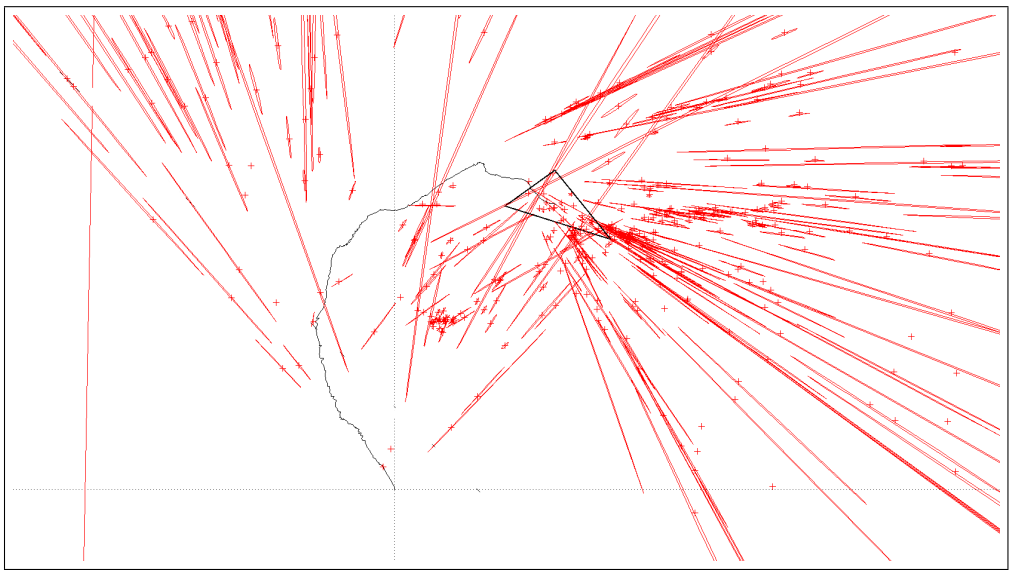
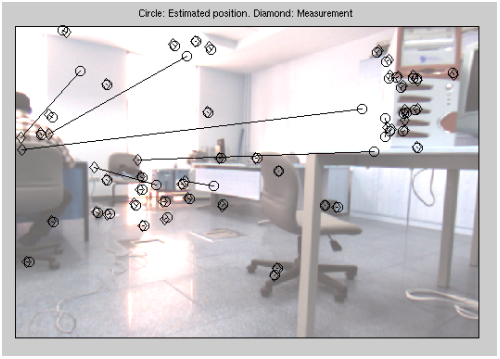


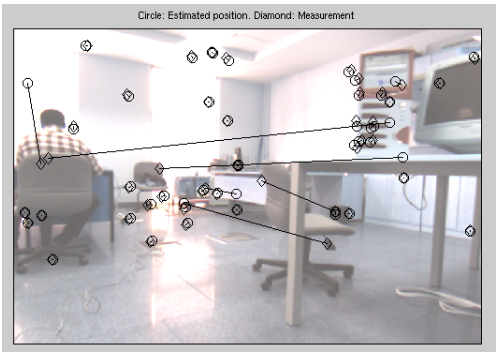
Figure 4.15: Percentage of residual outliers to visible and matched features vs frames for car_dataset1 with prediction error based outlier elimination



(a) Path estimate for lab_dataset1 zoomed with prediction error based outlier elimination. Triangle represents the robot with its sharp corner indicating its heading. Plus markers indicate landmark locations with associated uncertainty ellipses.



(b) Frame 570



(c) Frame 551

Figure 4.16: Estimated path and several frames and feature match vectors from lab_dataset1 with prediction error based outlier elimination. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

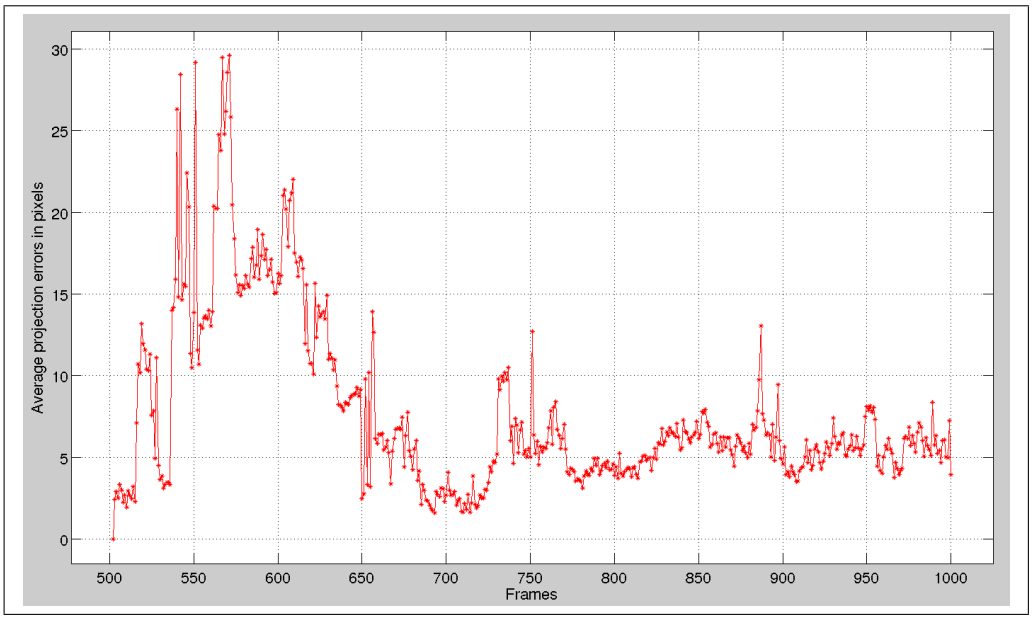


Figure 4.17: Average projection errors vs frames for lab_dataset1 with prediction error based outlier elimination

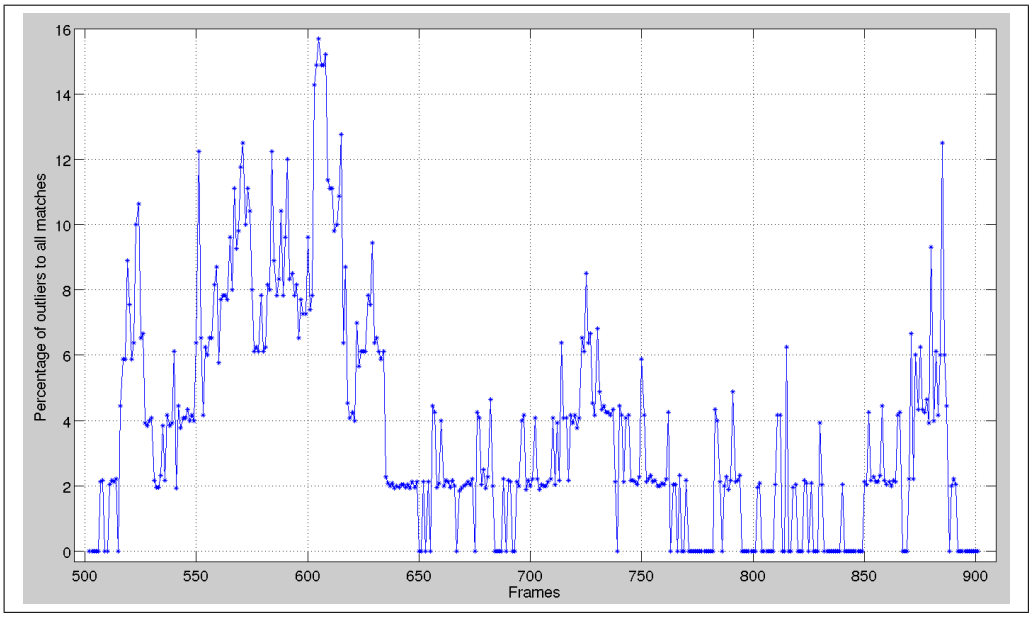
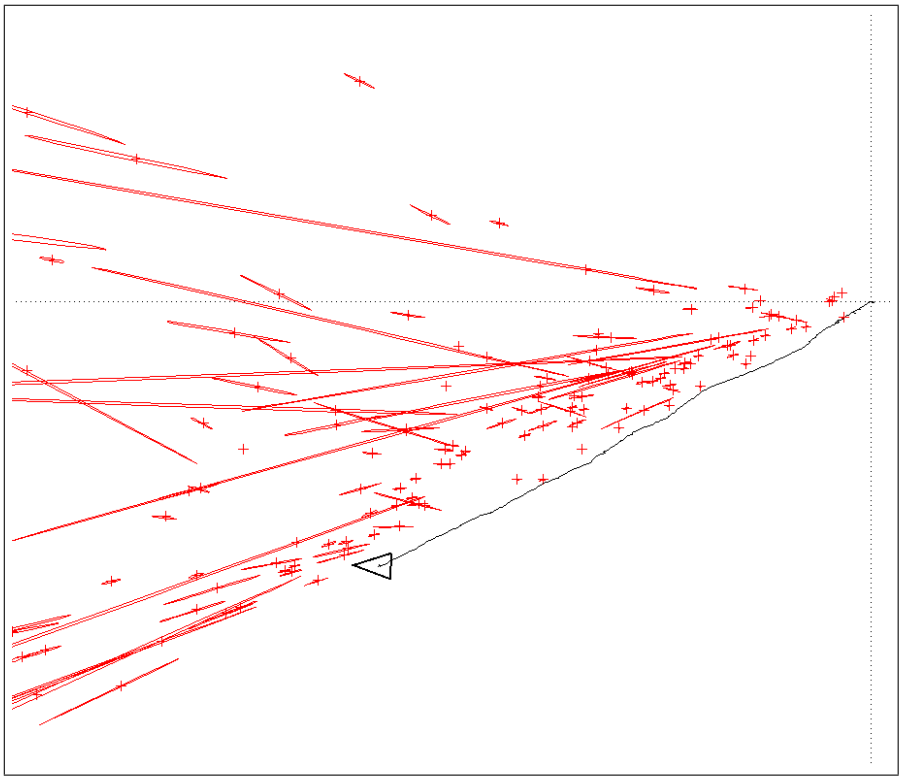
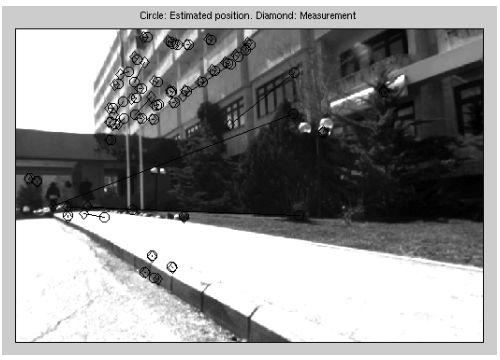


Figure 4.18: Percentage of residual outliers to visible and matched features vs frames for lab_dataset1 with prediction error based outlier elimination



(a) Path estimate for lab_dataset2 zoomed with prediction error based outlier elimination. Triangle represents the robot with its sharp corner indicating its heading. Plus markers indicate landmark locations with associated uncertainty ellipses.



(b) Frame 3306

(c) Frame 3298

Figure 4.19: Estimated path and several frames and feature match vectors from lab_dataset2 with prediction error based outlier elimination. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

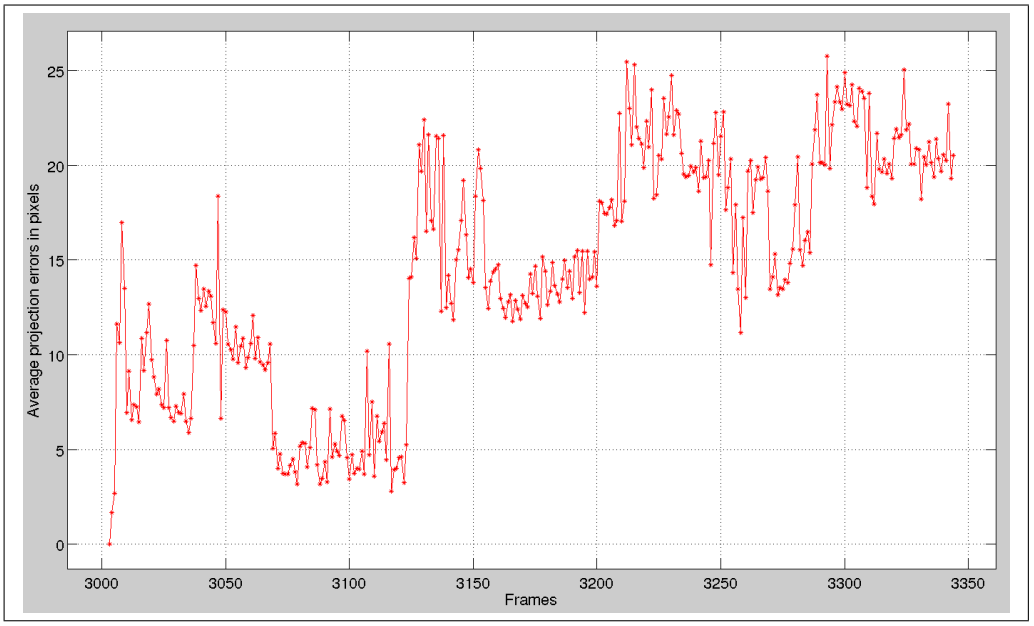


Figure 4.20: Average projection errors vs frames for lab_dataset2 with prediction error based outlier elimination

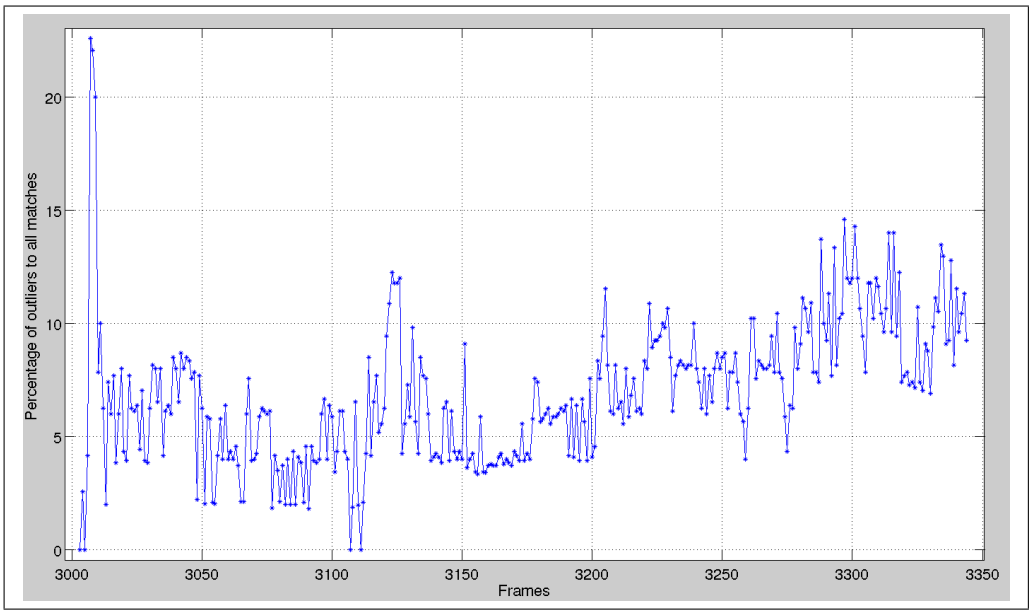
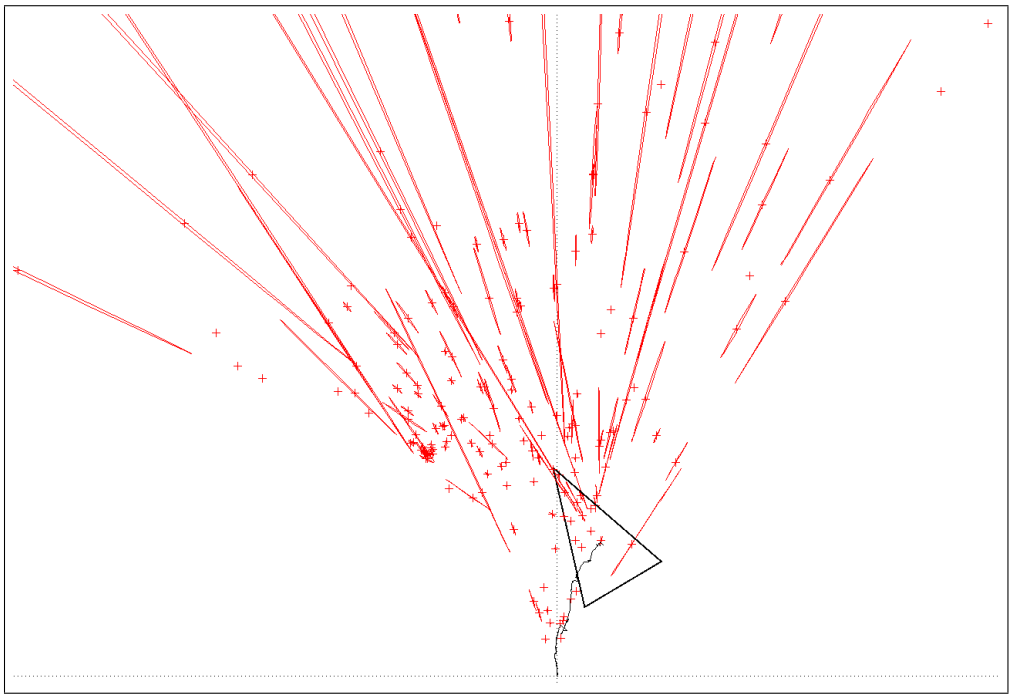
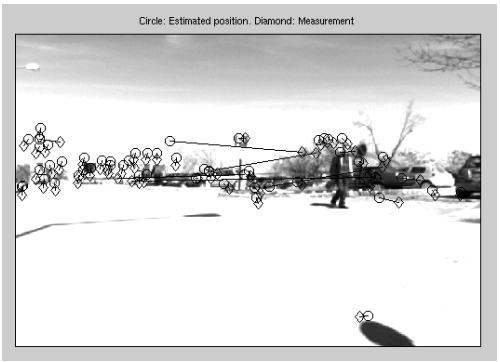


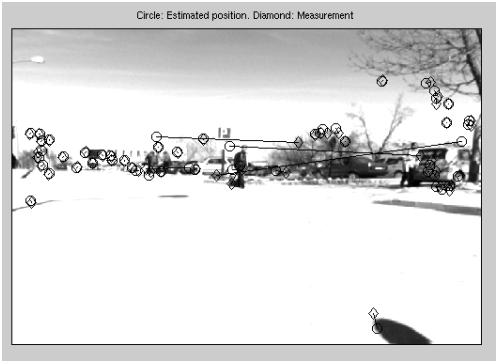
Figure 4.21: Percentage of residual outliers to visible and matched features vs frames for lab_dataset2 with prediction error based outlier elimination



(a) Path estimate for lab_dataset3 zoomed with prediction error based outlier elimination. Triangle represents the robot with its sharp corner indicating its heading. Plus markers indicate landmark locations with associated uncertainty ellipses.



(b) Frame 4297



(c) Frame 4238

Figure 4.22: Estimated path and several frames and feature match vectors from lab_dataset3 with prediction error based outlier elimination. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

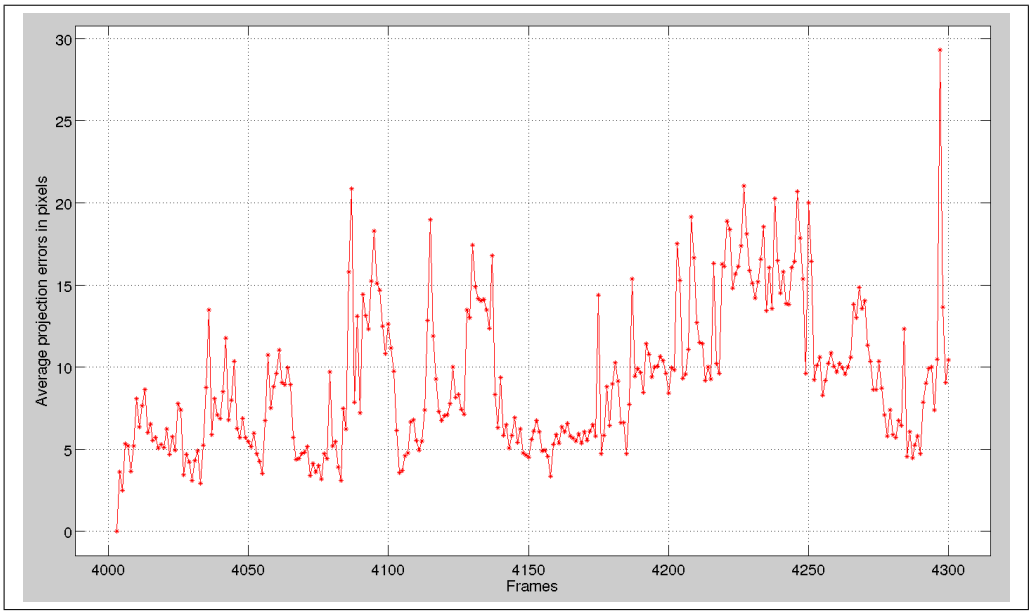


Figure 4.23: Average projection errors vs frames for lab_dataset3 with prediction error based outlier elimination

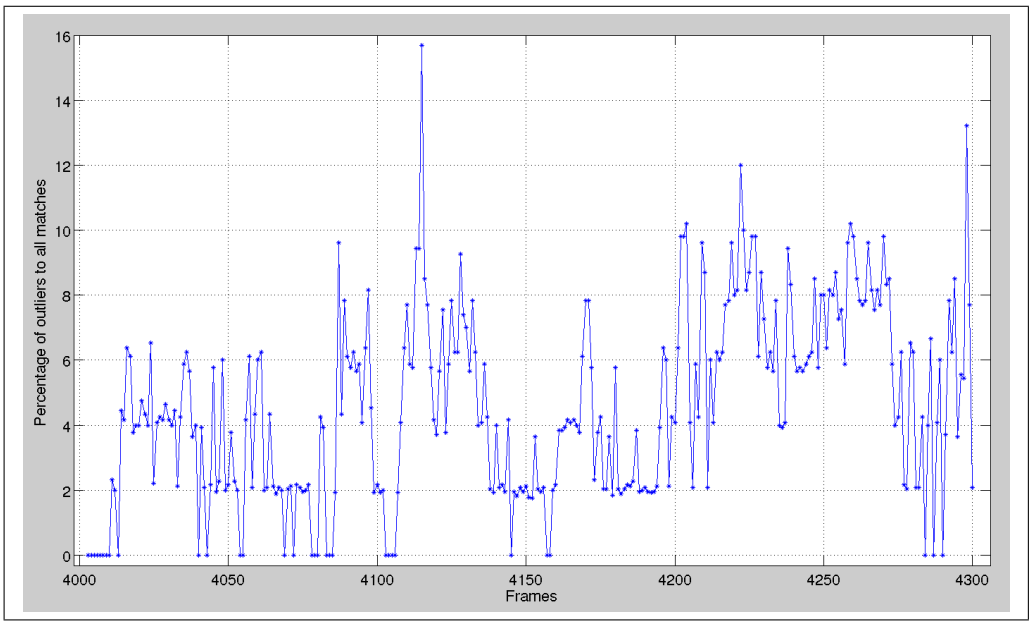


Figure 4.24: Percentage of residual outliers to visible and matched features vs frames for lab_dataset3 with prediction error based outlier elimination

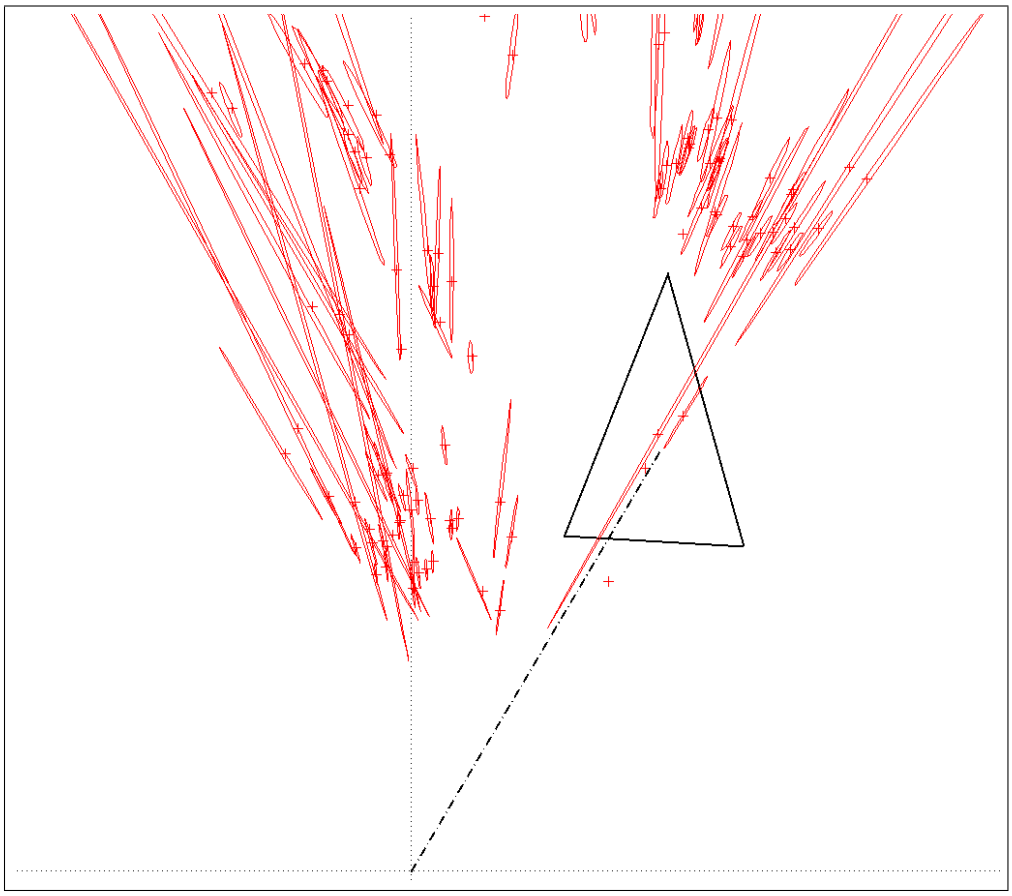
4.2.3 VSLAM with Optical Flow Aided and Prediction Error Based Outlier Elimination

In this section, we report performance results with our second filter, the Optical Flow Aided Outlier Eliminator, used together with Prediction Error Based Outlier Eliminator. As we will shortly explain, we observe improvements in path estimates, average projection errors and outlier percentages. When estimated paths are compared to those in Section 4.2.2, the smoothness of the paths for these last tests, can be seen from Figures 4.28(a), 4.31(a) and 4.34(a). Since the estimated path of *car_dataset1* in Section 4.2.2 was already free of glitches, there is little improvement for this dataset (Figure 4.25(a)). Local irregularities for *lab_dataset1* were eliminated with the second filter, but there is still too much curvature towards the end of path (Figure 4.10(a)). Although the ground truth path (Figure 4.8) follows a line, a curved path was estimated for *lab_dataset2* (Figure 4.11(a)). After adding the second outlier eliminator, we obtained a smoother path (Figure 4.31(a)), which fits better to the actual path. Better improvements were observed for *lab_dataset3*. In the previous section, there were local loops, turn backs and glitches (Figure 4.22(a)). After applying our second filter, even though the end result is different than the ground truth path, many irregularities were fixed; with an erroneous curvature towards left still persist.

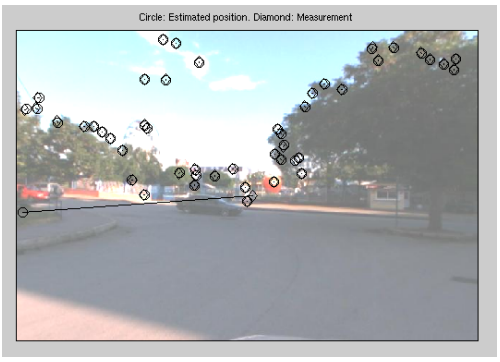
Example frames with largest projection errors are shown in Figures 4.25, 4.28, 4.31 and 4.34. Compared to the previous case (Figures 4.13, 4.16, 4.19 and 4.22), the number of outliers is halved together with the total effect on noise approximately halved as well. One weakness of the Optical Aided Outlier Eliminator was that if the predicted location of a landmark, p_L, and its matches features location drift apart from each other, but the flow vector at p_L has similar orientation and norm with the match vector, although that match is an outlier, it cannot be detected. In fact, the Prediction Error Based Outlier was designed to eliminate such outliers, but as it was described in Section 4.2.2, it may fail, and the Optical Flow Aided Outlier Eliminator may also allow this match, resulting in elimination failure.

Average projection errors, in pixels versus frame numbers, are plotted in Figures

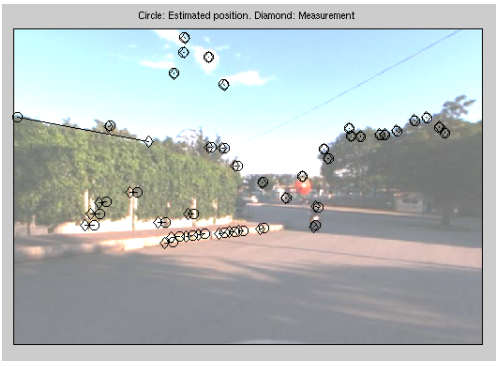
4.26, 4.29, 4.32 and 4.35. In Section 4.2.2, reasons for projection errors were described, which were rapid camera movements and false matches. The only source of such errors that can be eliminated on those were that come from false matches. When we compare Figures 4.27, 4.30, 4.33 and 4.36 with Figures 4.15, 4.18, 4.21 and 4.24 respectively, adding the second filter reduces the percentage of false matches. The effect of this reduction on the projection error is twofold: firstly, the outliers are themselves are eliminated; secondly effect of false matches on the robot pose, such as disturbing its orientation which can cause large errors even with small changes, are prevented.



(a) Path estimate for car_dataset1 zoomed with prediction error based and optical flow aided outlier elimination. Triangle represents the robot with its sharp corner indicating its heading. Plus markers indicate landmark locations with associated uncertainty ellipses.



(b) Frame 142



(c) Frame 34

Figure 4.25: Estimated path and several frames and feature match vectors from car_dataset1 with prediction error based and optical flow aided outlier elimination. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

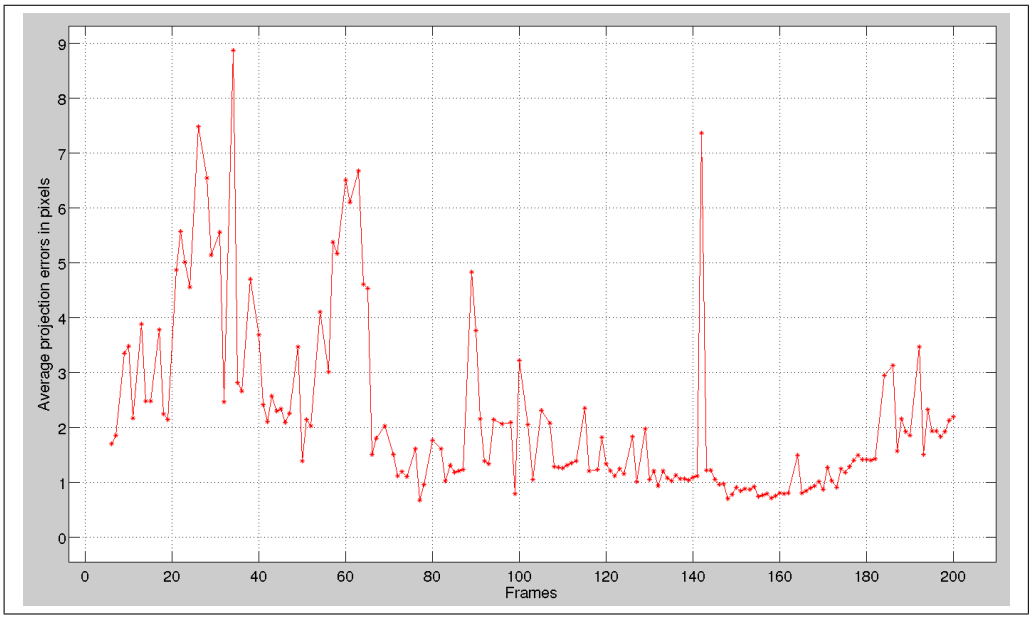


Figure 4.26: Average projection errors vs frames for car_dataset1 with prediction error based and optical flow aided outlier elimination

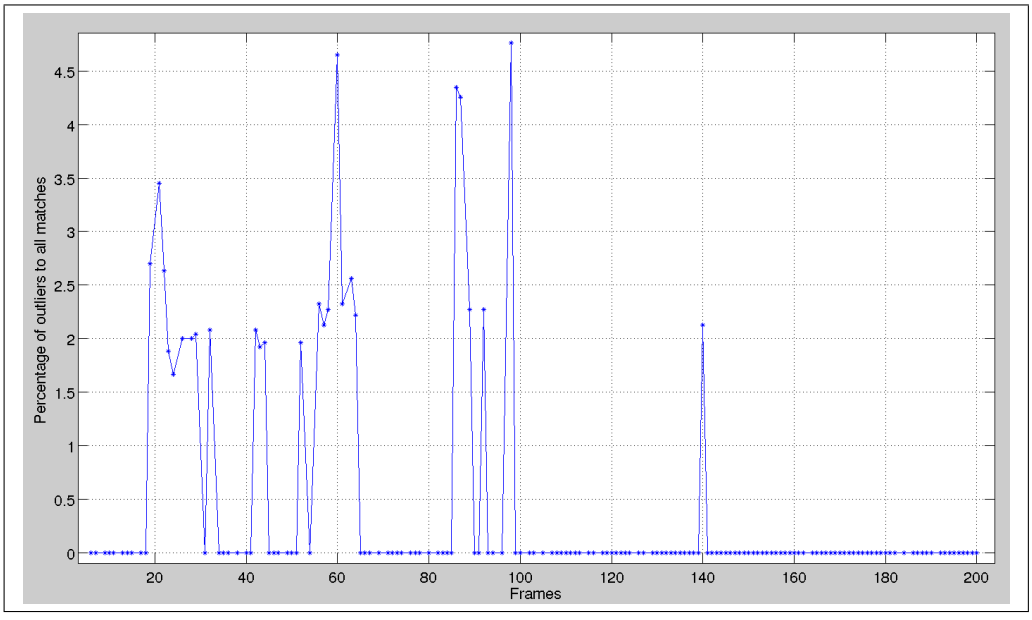
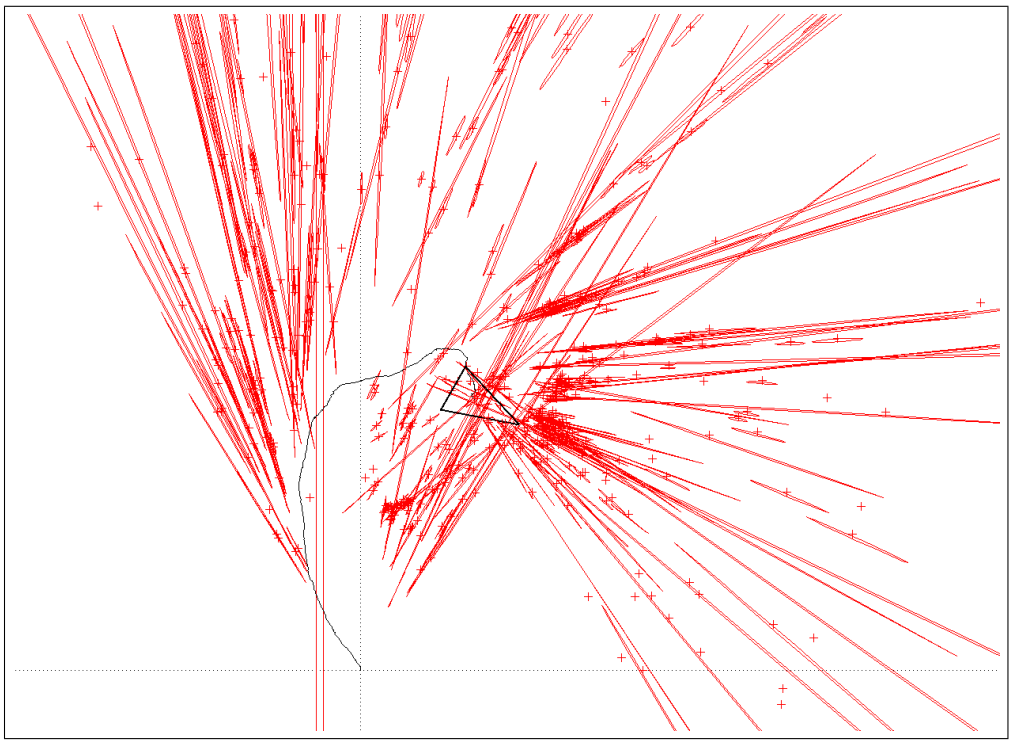
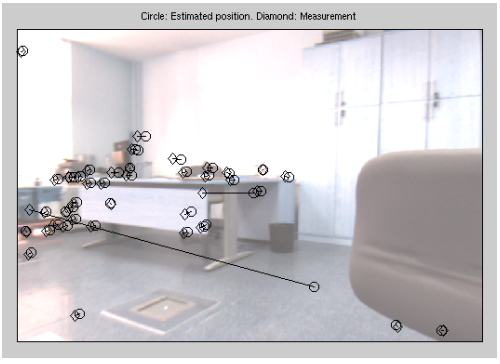


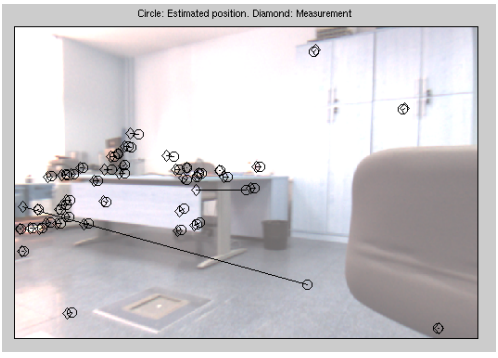
Figure 4.27: Percentage of residual outliers to visible and matched features vs frames for car_dataset1 with prediction error based and optical flow aided outlier elimination



(a) Path estimate for lab_dataset1 zoomed with prediction error based and optical flow aided outlier elimination. Triangle represents the robot with its sharp corner indicating its heading. Plus markers indicate landmark locations with associated uncertainty ellipses.



(b) Frame 677



(c) Frame 678

Figure 4.28: Estimated path and several frames and feature match vectors from lab_dataset1 with prediction error based and optical flow aided outlier elimination. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

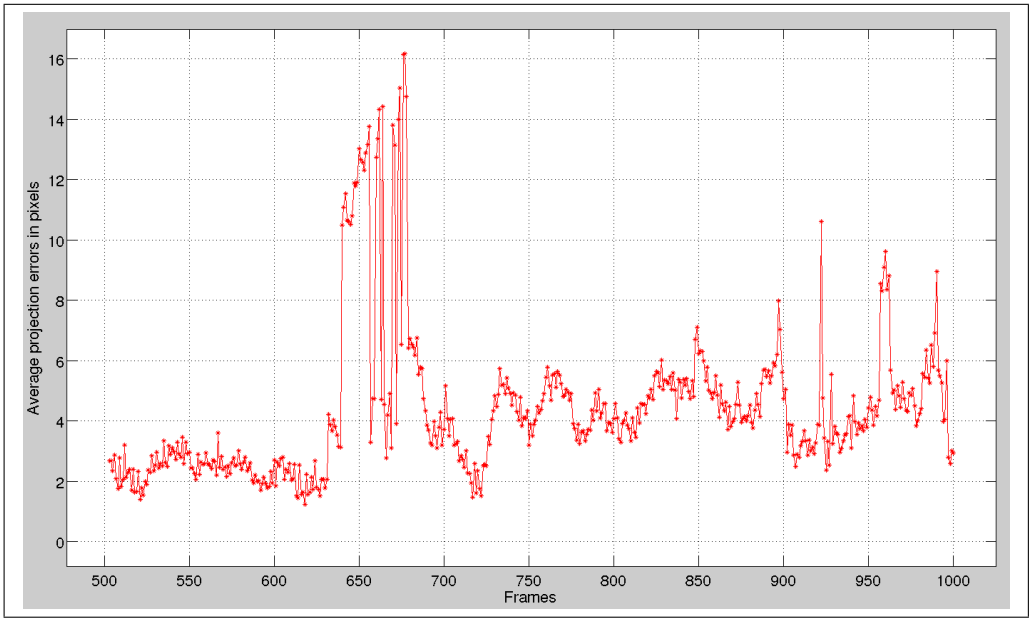


Figure 4.29: Average projection errors vs frames for lab_dataset1 with prediction error based and optical flow aided outlier elimination

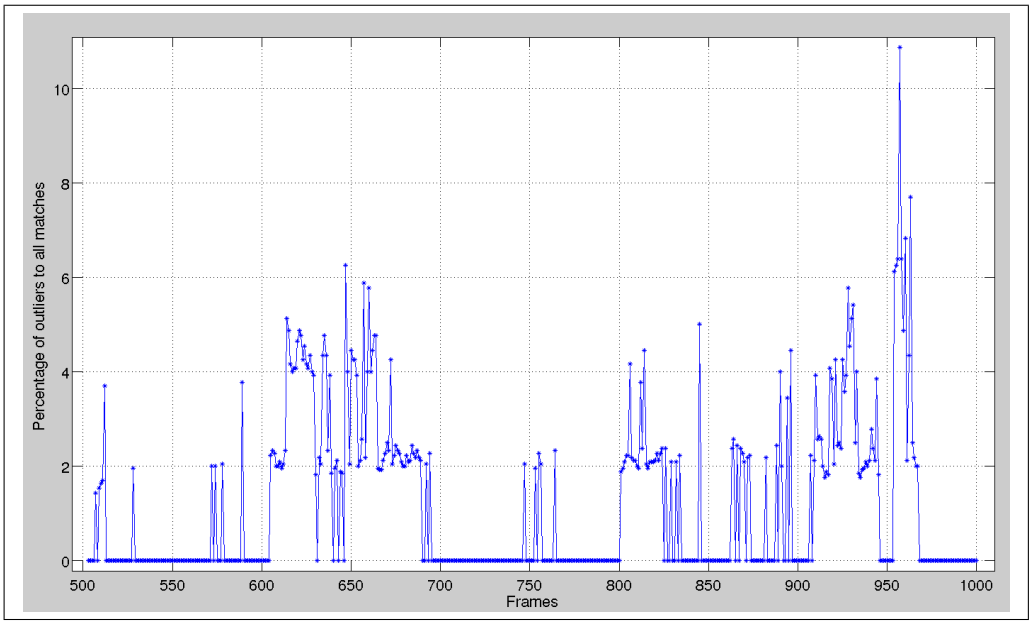
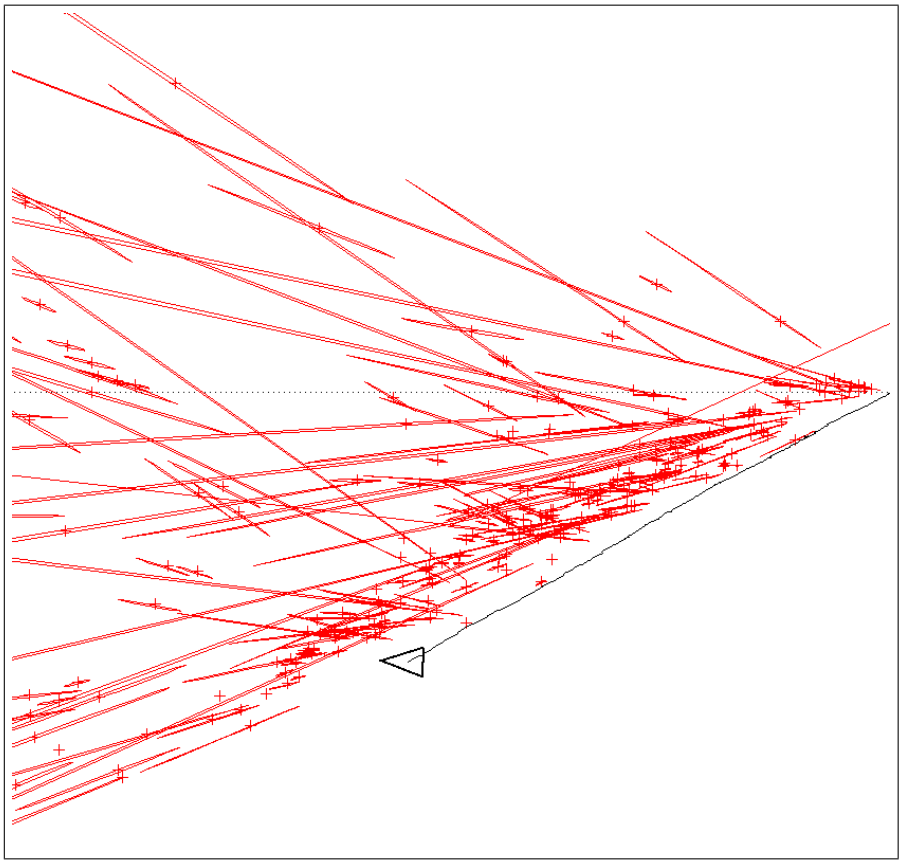
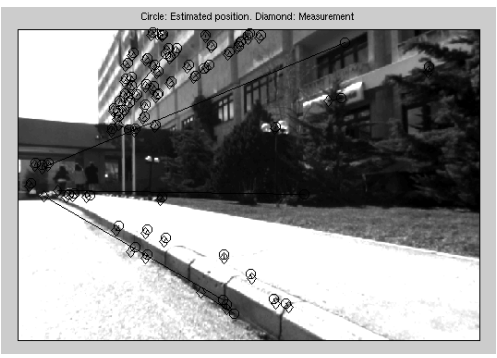


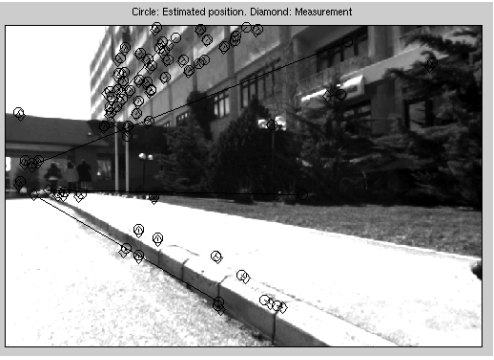
Figure 4.30: Percentage of residual outliers to visible and matched features vs frames for lab_dataset1 with prediction error based and optical flow aided outlier elimination



(a) Path estimate for lab_dataset2 zoomed with prediction error based and optical flow aided outlier elimination. Triangle represents the robot with its sharp corner indicating its heading. Plus markers indicate landmark locations with associated uncertainty ellipses.



(b) Frame 3250



(c) Frame 3251

Figure 4.31: Estimated path and several frames and feature match vectors from lab_dataset2 with prediction error based and optical flow aided outlier elimination. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

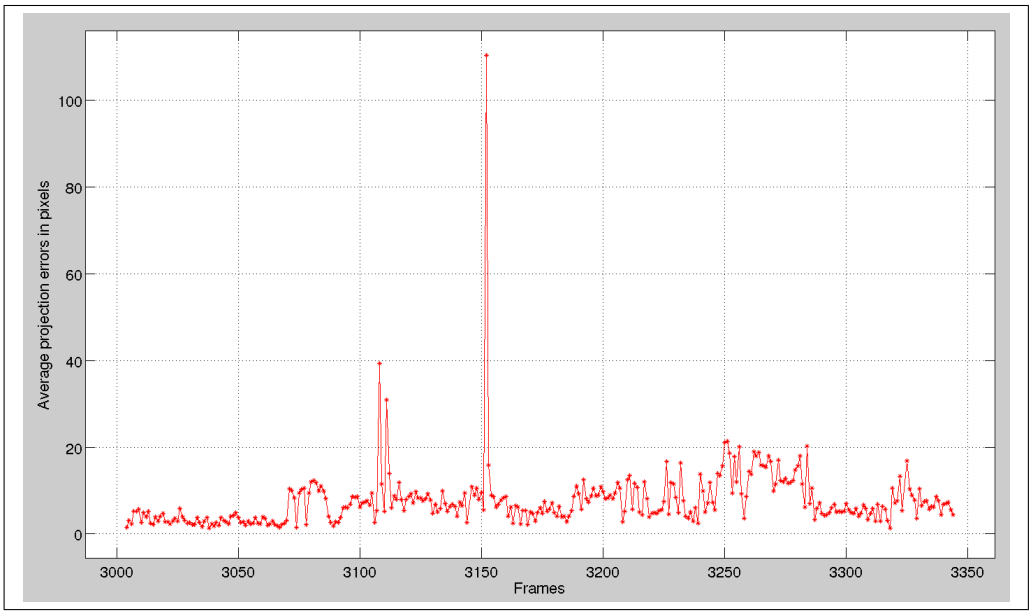


Figure 4.32: Average projection errors vs frames for lab_dataset2 with prediction error based and optical flow aided outlier elimination

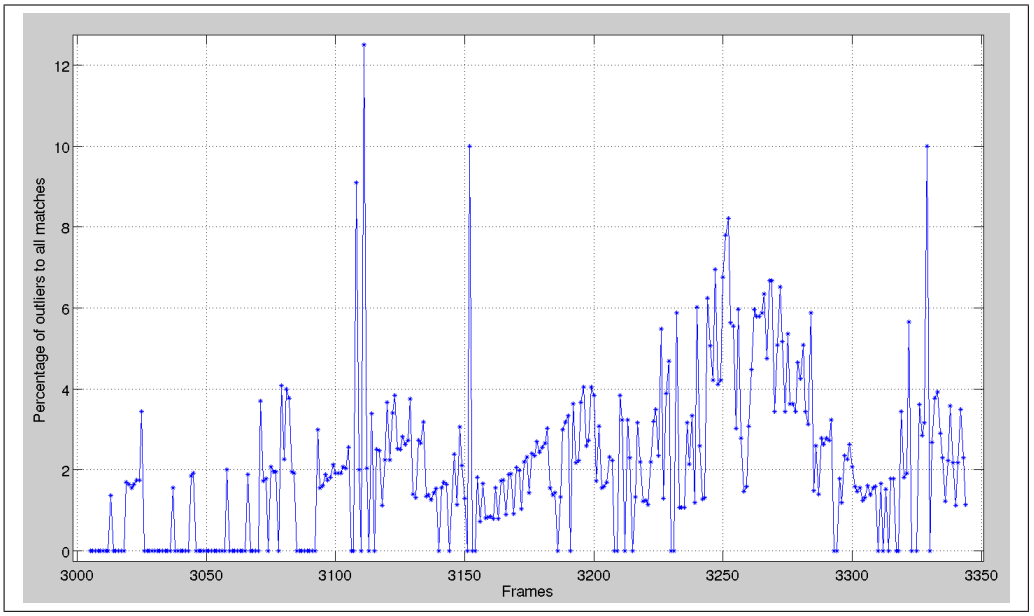
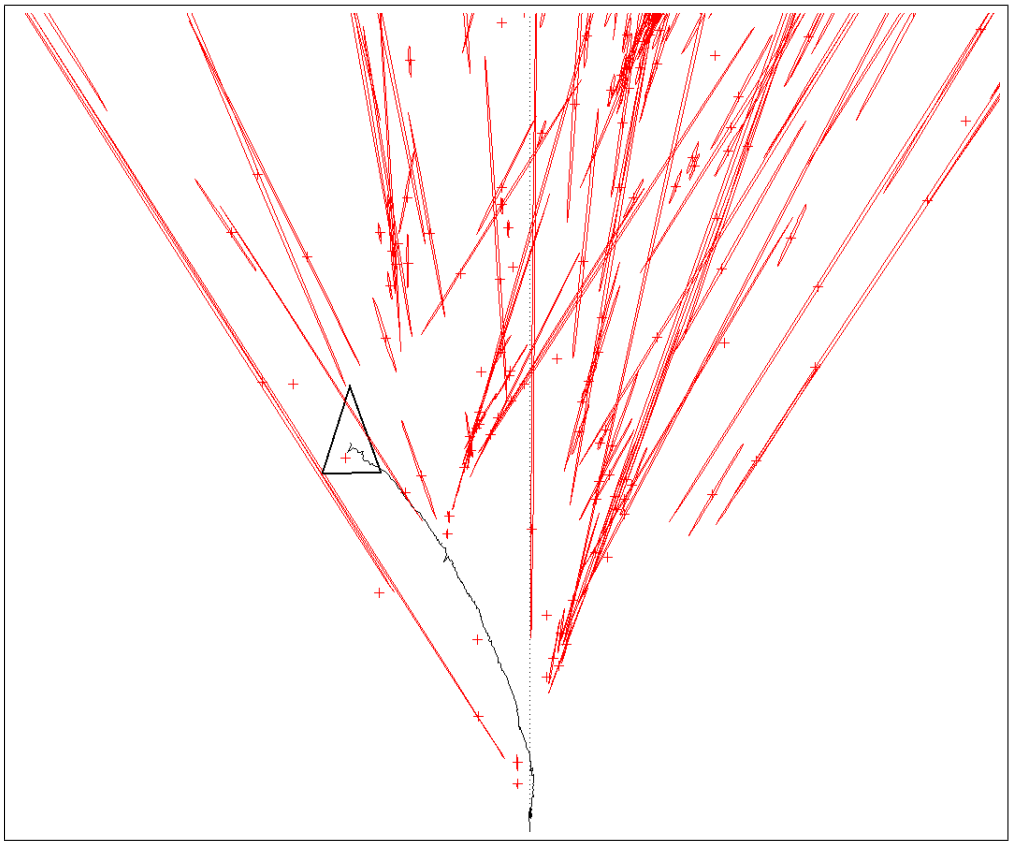
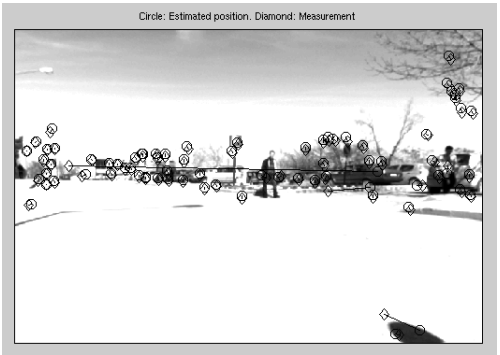


Figure 4.33: Percentage of residual outliers to visible and matched features vs frames for lab_dataset2 with prediction error based and optical flow aided outlier elimination



(a) Path estimate for lab_dataset3 zoomed with prediction error based and optical flow aided outlier elimination. Triangle represents the robot with its sharp corner indicating its heading. Plus markers indicate landmark locations with associated uncertainty ellipses.



(b) Frame 4254

(c) Frame 4297

Figure 4.34: Estimated path and several frames and feature match vectors from lab_dataset3 with prediction error based and optical flow aided outlier elimination. Circles and diamonds, connected with lines, are estimated landmark projection positions and their corresponding measurements respectively.

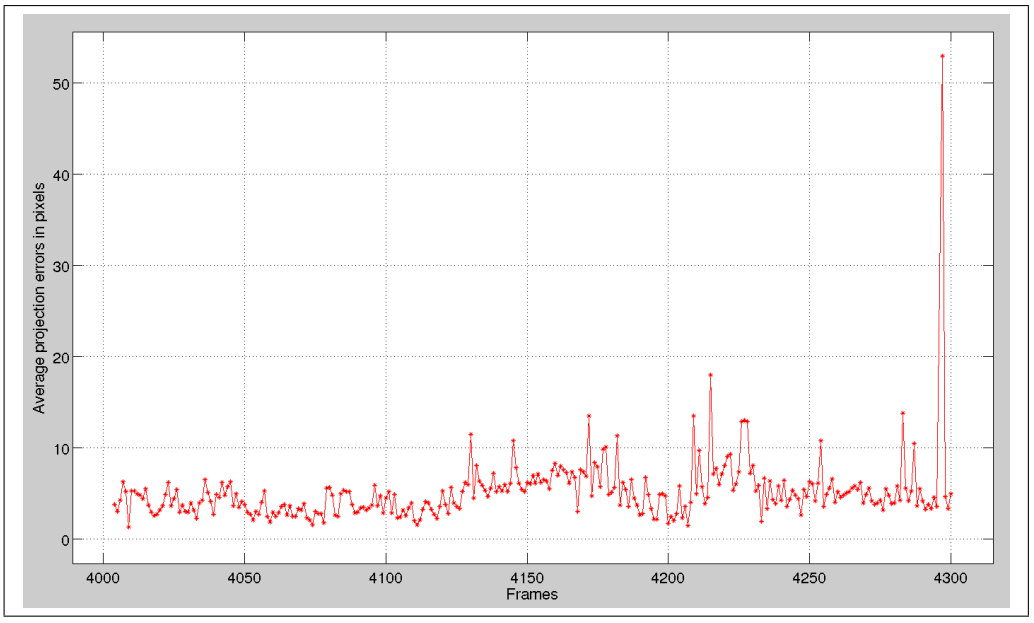


Figure 4.35: Average projection errors vs frames for lab_dataset3 with prediction error based and optical flow aided outlier elimination

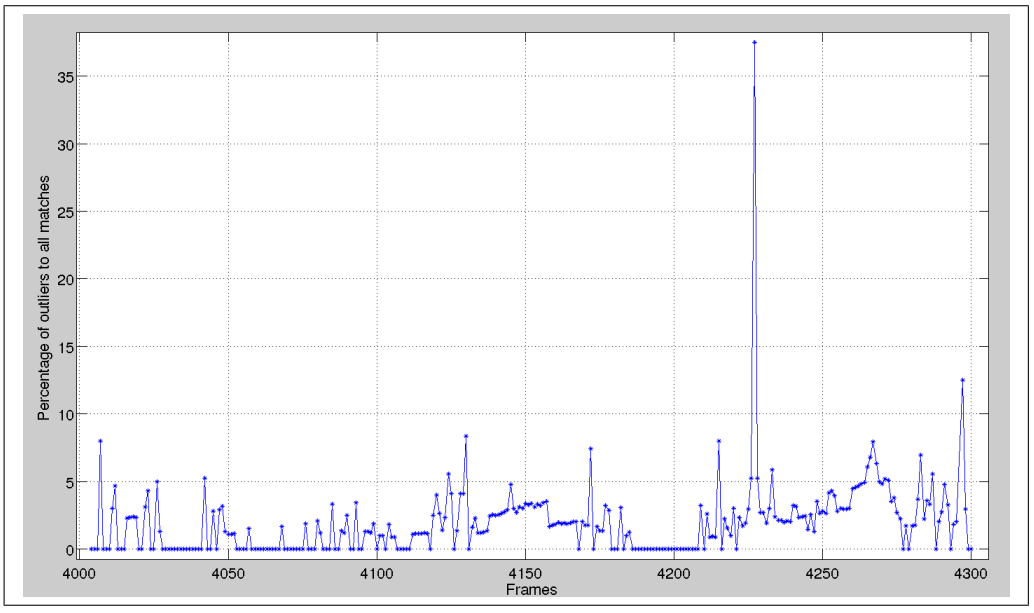


Figure 4.36: Percentage of residual outliers to visible and matched features vs frames for lab_dataset3 with prediction error based and optical flow aided outlier elimination

In table 4.5, a summary of all tests with their performance metrics are given. In this table, eliminators that are used were abbreviated with letters B, P and O corresponding to $BaseCase$, $PredictionErrorBasedOutlierEliminator$ and $OpticalFlowAidedOutlierEliminator$ respectively. For all datasets, the base case failed to converge, so no further statistics were given. When we look at the tests with eliminators used, we observe a decrease in average projection errors and the ratio of outliers. The average projection error was found as the mean of all projection errors, and the ratio of outliers was the percentage of outliers to all feature matches throughout the whole sequence. More importantly perhaps, using eliminators, VSLAM's behavior changes to convergence. Looking at these results, we can conclude that prediction based outlier eliminator saves SLAM from divergence, and optical flow aided outlier eliminator improves performance when applied with the former eliminator.

Table 4.5: Summary of the outlier elimination tests applied on real dataset with map together with their performances. The abbreviations used for eliminators mean *(B)ase Case*, *(P)rediction Error Based Outlier Eliminator* and *(P)rediction Error Based Outlier Eliminator and (O)ptical Flow Aided Outlier Eliminator used together*. Result are given for four different datasets and on each of them all the eliminator alternatives are applied. For base case, none of the datasets converged so no further details were given. *Average Prediction Error* has units of pixels. Percentage values are w.r.t. the number of all matches in a frame.

Eliminators Used	B	P	P+O	B	P	P+O	B	P	P+O	B	P	P+O
Dataset	car_dataset1			lab_dataset1			lab_dataset2			lab_dataset3		
SLAM Converged	×	✓	✓	×	✓	✓	×	✓	✓	×	✓	✓
Average Projection Error	-	4.48	2.11	-	7.81	4.35	-	14.42	7.17	-	9.26	4.83
% of Matches with Projection Errors Greater than 10 pixels	-	6.92	1.36	-	8.72	3.47	-	14.08	4.77	-	16.61	3.56
Average Number of Outliers	-	0.98	0.18	-	1.42	0.54	-	3.45	1.47	-	2.27	1.14
% of Outliers	-	2.05	0.4	-	2.99	1.15	-	6.92	2.27	-	4.6	1.77

Chapter 5

SLAM++ Software Architecture

In this chapter, the modular C++ library, *SLAM++*, which implements basic requirements of Visual SLAM algorithm will be explained. SLAM++ provides application developers the ability to implement several SLAM algorithms which may use different types of robots and sensors. Using the built-in interfaces, users can implement THEIR own drivers for sensors, new motion models and embed these into a SLAM algorithm. These scenarios may include EKF-SLAM of a robot moving in plane, equipped with a laser sensor; or FastSLAM2.0 with a camera moving in 3D space equipped with an IMU. SLAM++ is still under development, but at the moment, some of the commonly used SLAM algorithms, image processing modules, data containers and simulation tools are completed. Also, in simulation environment, we can run Visual EKF-SLAM stably.

5.1 Motivation

In the open source community, there exists many SLAM libraries written in different languages like C++, Matlab and Java, proven to be working stably. Many of these libraries focus on scenarios with robots moving in a plane, equipped with range-bearing sensors; and few of these implement SLAM with robots moving in 3D space and using cameras as measurement devices. Change in the type of the robot, its degrees of freedom, measurement devices usually requires completely

different mathematical models to which switching needs fundamental modifications. Furthermore, some existing VSLAM implementations are clumsy to be used in real-time scenarios or lack of modularity. For this reason, we started developing SLAM++, aiming it to be modular and run at real-time.

5.2 Software Architecture

Regardless of the type of the SLAM algorithm, robot design and measurement devices, all SLAM libraries require some basic routines to be implemented. These include SLAM's predict-measure-update loop, motion and measurement models, routines for information extraction from raw data gathered with measurement devices and data association. For this reason, we have implemented such interfaces using which more specific drivers can be coded. To illustrate, inheriting the SLAM interface class, one can implement any of EKF-SLAM, FastSLAM1.0, SEIF-SLAM etc. without losing or changing generality of the 'SLAM' concept. Furthermore, using motion model interface, modules which implement motion characteristics of a planar robot, a 6DOF robot or even a snake robot can easily be generated. Such considerations are applicable to measurement models, data association routines.

5.2.1 VSLAM Modules

As explained in Section 2.3, what SLAM means does not change with the scenario applied. All scenarios include the concepts of *Map*, *Trajectory*, *MotionModel*, *Measurement Model* either as outputs of this process or as its requirements. Furthermore, SLAM is an iteratively solved problem, which at each time step executes a set of processes such as prediction, getting measurements, solving for data association and updating belief. By providing appropriately derived drivers for all the above procedures, without losing generality, VSLAM process can be defined with a single, generic interface. In the current release, SLAM++ has Visual EKF-SLAM, FastSLAM1.0 and FastSLAM2.0 modules derived from the VSLAM interface class. Figure 5.1 shows the relation of these VSLAM algorithms with the

interface class. What changes between different SLAM algorithms is the details in an iteration. For example, if EKF-SLAM is the case, one should implement the generic Extended Kalman loop which is a series of matrix operations, or for the case of FastSLAM, a particle filter should be implemented together with voting heuristics.

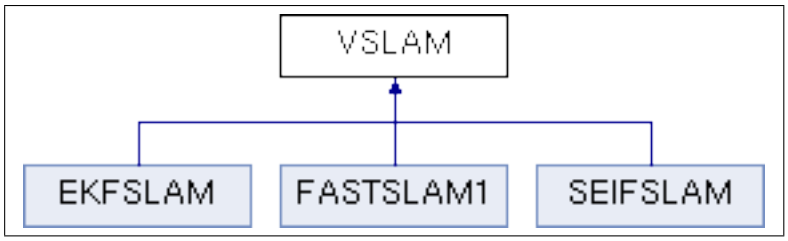


Figure 5.1: Relation between VSLAM interface and derived SLAM classes

Some of the VSLAM member functions and their descriptions are as follows:

- `VSLAM (MotionModel *mm, CameraModel *cm)`

  Constructor for VSLAM requires *MotionModel* and *CameraModel* as its inputs. Depending on the type of robot, *MotionModel* may have several constraints such as motion in plane or 3D space, constant velocity or acceleration etc. Different *CameraModel*s can also have various properties such as image size, type of distortion etc.

- `const Map * getMap (void)`

  Map is a container class which may consist of different type of map elements such as image regions, 3D meshes or range readings.

- `virtual void  step (const InterestPointVector &ipv, double dt)`

  This function implements the generic predict-measure-update loop. In case of EKF-SLAM, these are a series of matrix operations and for FastSLAM, it implements a particle filter.

5.2.2 MotionModel Modules

As explained earlier, according to the type of robot and expected motion characteristics, motion models may vary. For a robot moving in plane, only 3DOF, x-y

coordinates and heading, is considered; but a robot freely moving in 3D space has 6DOF, x-y-z coordinates and orientations in the three axes, which demands a different motion model. When possible SLAM scenarios are considered, the following functions are found to be common or musts for these scenarios.

- `MotionModel (double sigmaTrans, double sigmaRot)`

  In practice there is always error in motion, and this necessitates the parameters σ_{Trans}, error in translation, and σ_{Rot}, error in orientation, to be fed to any type of motion model.

- `virtual void predict (RobotState &robotState, double dt)`
  `virtual void noisyPredict (RobotState &robotState, double dt)`

  These two functions are implemented considering the two filtering methods, Kalman filters and particle filters, which requires noise-free and noisy pose predictions respectively. *predict(..)* function estimates the next pose and uncertainty is represented with a covariance matrix separately. However, when particle filters are used, the set of particles contain the uncertainty information. So, while drawing new particles, prediction step should also incorporate uncertainty. This behavior is implemented in *noisyPredict(..)*.

- `virtual cv::Mat dfv_dxv (const RobotState &robotState, double dt)`
  `virtual cv::Mat dfv_derr (const RobotState &robotState, double dt)`

  In EKF-SLAM, since non-linear mathematical models should be linearized, the functions *dfv_dxv()*, *dfv_derr()*, which calculates the Jacobians of motion model w.r.t. robot pose and system noise respectively, are included in the set of interface functions.

For the time being, since requirements do not force us for other models, only constant velocity motion model has been derived, whose relation with the *MotionModel* class is shown in Figure 5.2.

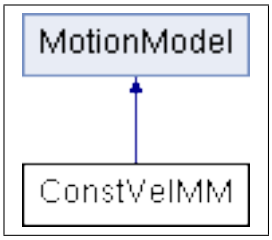


Figure 5.2: Relation between MotionModel interface and derived ConstantVelocityMotionModel class

5.2.3 Measurement Modules

The meaning of a landmark changes according to the type of SLAM. For a robot moving in plane, equipped with a range sensor, a landmark may be a single range reading, or lines, circles, corners extracted from the raw reading; but for a robot equipped with a camera, landmarks become image patches or interest points. Such changes in definition of a landmark, alters the mathematical model greatly. But, as it was for other interfaces, there are some common requirements of SLAM algorithms, like retrieving current estimate and its uncertainty for a landmark, updating its state etc. Also for SLAM algorithms running Extended Kalman Filter, Jacobians of landmark state with respect to measurement, robot pose and those of predicted measurement w.r.t. landmark state and robot pose are needed. Some of these Jacobians are also needed for FastSLAM2.0 and SEIF-SLAM. Some of the functions considered to be included in Measurement Model interface are as follows:

- `virtual cv::Mat getCovariance (void)`
  `virtual cv::Mat getMean (void)`
  `virtual cv::Mat getCartesianCovariance (void)`
  `virtual cv::Point3f getCartesianPosition (void)`

  These two set of getter functions are different in terms of in which parametrization space they return means and covariances. First two getters are to retrieve exact mean and covariance of a landmark. For example, if landmark is Inverse Depth Parametrized, then a 6-vector and 6×6 matrix is returned for mean and covariance respectively. However, in some situations, like estimating the projected feature location or projected uncertainty

region onto the image plane, independent from how the landmark is modeled, Cartesian means and covariances are needed. The latter two functions supply this information.

- `virtual void  refresh (const RobotState &robotState, CameraModel &cam)`

  Depending on the type of measurement model, updating the estimate of a landmark may require a series of operations. Furthermore, within a single update stage some other miscellaneous operations, like keeping history of states, may be needed. *refresh(..)* function packs all such requirements.

- `cv::Mat Dy_Dh (void)`
  `cv::Mat Dy_Dxv (void)`
  `cv::Mat Dh_Dy (void)`
  `cv::Mat Dh_Dxv (void)`

  These functions calculate the Jacobians of landmark state w.r.t. a measurement, robot pose; and Jacobians of predicted measurement w.r.t. landmark state and robot state respectively.

In the current release of SLAM++, only *IDPLandmark* was added. This implements Inverse Depth Parametrization (IDP) explained in [10]. IDP's alternative is XYZ parametrization [10] which is more efficient than IDP; but its initialization step needs special treatment. Also for stages with high uncertainty in depth, IDP performs better than XYZ parametrization, so in early stage it is preferable to use IDP. After an IDP landmark is localized, switching to XYZ parametrization will reduce the calculation. In future releases, such an addition is planned. Relation between IDPLandmark and Landmark classes is shown in 5.3.

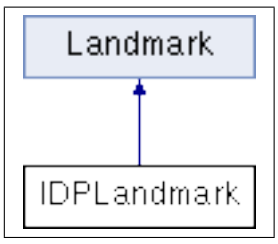


Figure 5.3: Relation between Landmark interface and derived IDPLandmark class

Chapter 6

Conclusion

In this thesis, we presented two methods for minimizing erroneous data association in Visual SLAM, in order to both prevent the VSLAM process from diverging and improving quality of the estimated map and robot pose. One of these methods uses the consistency assumption between current map estimate and the measurements. In other words, a feature point that is matched to a landmark, should be located in the vicinity of the estimated projected location of that landmark. When checking this condition, we also consider the uncertainty in the landmark's position via using Mahalanobis distance as the metric rather than comparing pixel locations. The second method utilizes the unused optical flow information which is intuitive to be used in VSLAM since its inputs are consecutive frames between which optical flow field defines a constraint. This method checks whether the vector defined by the feature locations those are matched to a landmark in consecutive frames is in agreement with the optical flow field around the first matched feature. In order to compare these two vectors, we modeled uncertainties in orientation and norm of these vectors; and than looked for their resemblance through Kullback-Leibler divergence.

The above methods are explained together with three different feature matching algorithms. We asserted that these matching algorithms have weaknesses, since they only use descriptor information for data association; and claimed that the proposed two outlier elimination methods can be used to resolve this weakness. On four different datasets, we run VSLAM without any outlier eliminator, with

map-measurement consistency based outlier eliminator and using the two eliminators together. Results have shown that, in the above given outlier eliminator application order, performance improved from divergence to more quality maps and trajectory estimates. Also background on optical flow calculation, image feature points and SLAM were given. Brief documentation of our modular SLAM library, *SLAM++*, was given explaining some of its important aspects. Also, aims in implementing such a library was given which include modularity and run-time performance.

Our intent in the near future is to complete the implementation of the *SLAM++* library together with the outlier elimination capabilities expecting it to run in real time and with high accuracy. However, in the long term, we would like to extract further information from optical flow field to detect moving objects and remove regions with such objects from input frames in order to further improve VSLAM performance by rejecting out dynamic objects in the environment and obtain a map that is a static subset of the environment.

We believe that using optical flow information through the above given methods will result in better localization and map building in the VSLAM problem and give a robot the ability to navigate autonomously in environments with poor texture quality.

Bibliography

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-7(4):384 –401, July 1985.

[2] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994.

[3] J. L. Barron, A. D. Jepson, and J. K. Tsotsos. The feasibility of motion and structure from noisy time-varying image velocity information. *International Journal of Computer Vision*, 5:239–269, 1990.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin / Heidelberg, 2006.

[5] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. Technical report, Intel Corporation Microprocessor Research Labs, 2000.

[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision - ECCV 2004*, volume 3024 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin / Heidelberg, 2004.

[7] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):500 –513, March 2011.

[8] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kummerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. Tardos. A comparison of slam algorithms based on a graph of relations. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2089 –2095, Oct 2009.

[9] J. Civera, A. Davison, and J. Montiel. Inverse depth parametrization for monocular slam. *Robotics, IEEE Transactions on*, 24(5):932 –945, oct 2008.

[10] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry. *J. Field Robot.*, 27:609–631, September 2010.

[11] A. Davison, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052 –1067, June 2007.

[12] A. J. Davison. *Mobile Robot Navigation Using Active Vision.* PhD thesis, University of Oxford, 1998.

[13] A. Doucet, N. d. Freitas, K. P. Murphy, and S. J. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 176–183, 2000.

[14] P.-E. Forssen. Maximally stable colour regions for recognition and matching. In *Computer Vision and Pattern Recognition*, 2007.

[15] U. Frese. Efficient 6-dof slam with treemap as a generic backend. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4814 –4819, April 2007.

[16] S. Gauglitz, T. Hllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94:335–360, 2011.

[17] A. Gil, O. Mozos, M. Ballesta, and O. Reinoso. A comparative evaluation of interest point detectors and local descriptors for visual slam. *Machine Vision and Applications*, 21:905–920, 2010. 10.1007/s00138-009-0195-x.

[18] A. Gil, O. Mozos, M. Ballesta, and O. Reinoso. A comparative evaluation of interest point detectors and local descriptors for visual slam. *Machine Vision and Applications*, 21:905–920, 2010.

[19] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *Robotics and Automation, IEEE Transactions on*, 17(3):242 –257, jun 2001.

[20] C. Harris and M. Stephens. *A combined corner and edge detector*, volume 15, pages 147–151. Manchester, UK, 1988.

[21] D. J. Heeger and A. D. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *International Journal of Computer Vision*, 7:95–117, 1992.

[22] B. K. Horn. *Robot Vision*. McGraw-Hill Higher Education, 1st edition, 1986.

[23] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185 – 203, 1981.

[24] G. Klein. *Visual Tracking for Augmented Reality*. PhD thesis, Oxford University, 2006.

[25] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[26] J. Knight, A. Davison, and I. Reid. Towards constant time slam using postponement. *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 1:405 –413 vol.1, 2001.

[27] S. Kullback and R. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22:79–86, 1951.

[28] J. Leonard and H. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 1442 –1447 vol.3, nov 1991.

[29] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[30] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[31] M. Montemerlo and S. Thrun. *Tracts in Advanced Robotics*, volume 27. Springer, 2007.

[32] P. Moutarlier and R. Chatila. An experimental system for incremental environment modelling by an autonomous mobile robot. *Experimental Robotics I*, 139:327–346, 1990.

[33] O. M. Mozos, A. Gil, M. Ballesta, and O. Reinoso. Interest point detectors for visual slam. In D. Borrajo, L. Castillo, and J. M. Corchado, editors, *Current Topics in Artificial Intelligence*, pages 170–179, Berlin, Heidelberg, 2007. Springer-Verlag.

[34] . Mozos, A. Gil, M. Ballesta, and O. Reinoso. Interest point detectors for visual slam. In D. Borrajo, L. Castillo, and J. Corchado, editors, *Current Topics in Artificial Intelligence*, volume 4788 of *Lecture Notes in Computer Science*, pages 170–179. Springer Berlin / Heidelberg, 2007.

[35] M. A. Paskin. Thin junction tree filters for simultaneous localization and mapping. *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1157–1164, 2003.

[36] L. Paz, P. Jensfelt, J. Tardos, and J. Neira. Ekf slam updates in o(n) with divide and conquer slam. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1657 –1663, April 2007.

[37] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[38] J. Shi and C. Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593 –600, jun 1994.

[39] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

[40] S. Smith and J. Brady. Susan - a new approach to low level image processing. *Int. Journal of Computer Vision*, 23(1):45–78, May 1997.

[41] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[42] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Ng. Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. In *Algorithmic Foundations of Robotics V*, volume 7 of *Springer Tracts in Advanced Robotics*, pages 363–380. Springer Berlin / Heidelberg, 2004.

[43] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3:177–280, July 2008.

[44] J. Weickert and C. Schnrr. Variational optic flow computation with a spatio-temporal smoothness constraint. *Journal of Mathematical Imaging and Vision*, 14:245–255, 2001.

[45] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). 2007.