Vision-Inertial SLAM using Natural Features in Outdoor Environments

by

Daniel C. Asmar

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Simultaneous Localization and Mapping (SLAM) is a recursive probabilistic inferencing process used for robot navigation when Global Positioning Systems (GPS) are unavailable. SLAM operates by building a map of the robot environment, while concurrently localizing the robot within this map. The ultimate goal of SLAM is to operate anywhere using the environment's natural features as landmarks. Such a goal is difficult to achieve for several reasons. Firstly, different environments contain different types of natural features, each exhibiting large variance in its shape and appearance. Secondly, objects look differently from different viewpoints and it is therefore difficult to always recognize them. Thirdly, in most outdoor environments it is not possible to predict the motion of a vehicle using wheel encoders because of errors caused by slippage. Finally, the design of a SLAM system to operate in a large-scale outdoor setting is in itself a challenge.

The above issues are addressed as follows. Firstly, a camera is used to recognize the environmental context (*e.g.*, indoor office, outdoor park) by analyzing the holistic spectral content of images of the robot's surroundings. A type of feature (*e.g.*, trees for a park) is then chosen for SLAM that is likely observable in the recognized setting. A novel tree detection system is introduced, which is based on perceptually organizing the content of images into quasi-vertical structures and marking those structures that intersect ground level as tree trunks. Secondly, a new tree recognition system is proposed, which is based on extracting Scale Invariant Feature Transform (SIFT) features on each tree trunk region and matching trees in feature space. Thirdly, dead-reckoning is performed via an Inertial Navigation System (INS), bounded by non-holonomic constraints. INS are insensitive to slippage and varying ground conditions. Finally, the developed Computer Vision and Inertial systems are integrated within the framework of an Extended Kalman Filter into a working Vision-INS SLAM system, named VisSLAM.

VisSLAM is tested on data collected during a real test run in an outdoor unstructured environment. Three test scenarios are proposed, ranging from semi-automatic detection, recognition, and initialization to a fully automated SLAM system. The first two scenarios are used to verify the presented inertial and Computer Vision algorithms in the context of localization, where results indicate accurate vehicle pose estimation for the majority of its journey. The final scenario evaluates the application of the proposed systems for SLAM, where results indicate successful operation for a long portion of the vehicle journey. Although the scope of this thesis is to operate in an outdoor park setting using tree trunks as landmarks, the developed techniques lend themselves to other environments using different natural objects as landmarks.

Acknowledgments

As I write these last lines of my thesis, I look back at the past four years of my life and contemplate my achievements. I realize that none of this would be possible without the help of several people and institutions whom I find myself indebted to.

Thank you to Dr. John Zelek and Dr. Samer Abdallah, who are my friends before being my supervisors. Thank you guys for always being there for me, for guiding me and supporting me in all possible fashions. Thank you for looking out for my interest and pushing me towards success. I would also like to thank Dr. David Clausi, Dr. Hamid Tizhoosh, and Dr Daniel Smilek for their time and guidance. Thank you to my external examiner Professor Eduardo Nebot for his valuable advice and comments, which helped improve the quality of this thesis.

Thank you to my wife Aline for her love, support, and understanding throughout the hardest times of my life. I could not have done it without you Allo. Thank you to my daughter Nour and my son Tony for just being who they are. Their smiling faces every day remind me how valuable my life is.

Thank you to my mom and dad, who have been a source of power, motivation, and all kinds of support to me. To my mom who religiously made me coffee every morning for the past 4 years. To my dad who has always believed in me and motivated me to always better myself. Thank you to my sister Mary and my brother Manuel for their love and understanding.

Thank you to my friend Adel for all his help in compiling this thesis and for the mind lifting conversations we have had together. Thank you to my friends Ahmad El Shameli, Eyad Barakat, Chris Maciejko, Abdunnasser Younes, and Ahmad Farshoukh for the great times together.

Thank you to the Ontario government for its financial support through two consecutive Ontario Graduate Scholarships (OGS). Thank you to the Canadian government for its partial support through the National Sciences and Research Council (NSERC). I would also like to thank the University of Waterloo and the American University of Beirut for partial financial support.

Dedication

To Aline, Nour, and Tony.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Hypotheses

1. Land-based Simultaneous Localization and Mapping (SLAM) is feasible in outdoor environments using a low-cost Inertial Measurement Unit (IMU) for dead-reckoning purposes and a camera for detecting, recognizing, and localizing natural objects.
2. SLAM using natural features produces meaningful maps that can be used for surveying or for human navigation.
3. By recognizing landmarks from their appearance, conventional statistical-based data association can be avoided.

These three hypotheses are demonstrated by presenting a Vision-Inertial SLAM system implemented in an outdoor park setting, using tree trunks as landmarks for SLAM.

1.2 Motivation

With the turn of the twenty first century, the barriers between robots and humans are falling. Humans are less intimidated by electronic equipment, which is evidenced by the ubiquitous presence of computers and digital equipment in consumer households. Robots are evolving from toys to necessities, from being remotely controlled to being fully autonomous. Roboticists have shifted their research from insuring

robot motion control, which is considered a solved problem today, to fulfilling a human need and thereby insuring the robots' economic justification. Examples of such robots include the robot vacuum cleaner ROOMBA [1] which operates autonomously and recharges itself without any human intervention. Search and rescue robots, such as the PACKBOT by iRobot [2], are also very popular and are dispersed in catastrophe type scenarios such as in an earthquake aftermath. Other examples of autonomous robot applications include extra-terrestrial exploration such as the Spirit Mars rover [3] that successfully navigated, sampled, and mapped areas of the planet Mars and relayed back this information to humans located millions of miles away. These robots share in common the fact that they must navigate autonomously with no help from a human operator. Current mobile robot systems are limited in their capacity to achieve this task.

The simplest form of robot navigation is line-following, where the robot is equipped with sensors that are able to track a pre-laid electromagnetic, optical or thermal track (Figure 1.1a). This type of navigation is very successful in constrained indoor environments with a slow-moving robot. The disadvantage of line-following robots is that they are dependent on the line they follow. Bad ground conditions or crossed tracks could cause the robot to loose its fix on the track and become completely lost. Skidding and quick turns could also cause de-tracking. In other words, no lateral movement is tolerated. In general, methods that require environmental modification are limited by economics, generalization, and places of application. More successful navigation algorithms are based on robot localization. In these algorithms the pose (location and orientation) of the robot is calculated at each time step. Dead-reckoning using wheel encoders or inertial sensors is a good positioning methodology for short paths; however, the errors associated with such methods accumulate with distance and eventually localization fails if no corrective measures are taken. Localization can alternatively be estimated via triangulation (Figure 1.1b) by using range sensors to estimate the distance from the robot to artificial or natural landmarks having known coordinates. In an effort to improve robot localization, robot pose estimates obtained from dead-reckoning sensors on one hand and from landmark relative positions on the other hand can be combined to yield more precise pose estimates. The Kalman Filter (KF) [4, 5] is optimal at fusing such data sources if both the dynamic and measurement models are linear and the associated errors can be represented by white Gaussian noise. Unfortunately, in practice it is rare when linearity can be assumed and alternative filters that can deal with non-linearities are used, such as the Extended Kalman Filter (EKF) [5] or the Particle Filter (PF) [6, 7].

In robot localization, differentiation is made between local robot localization

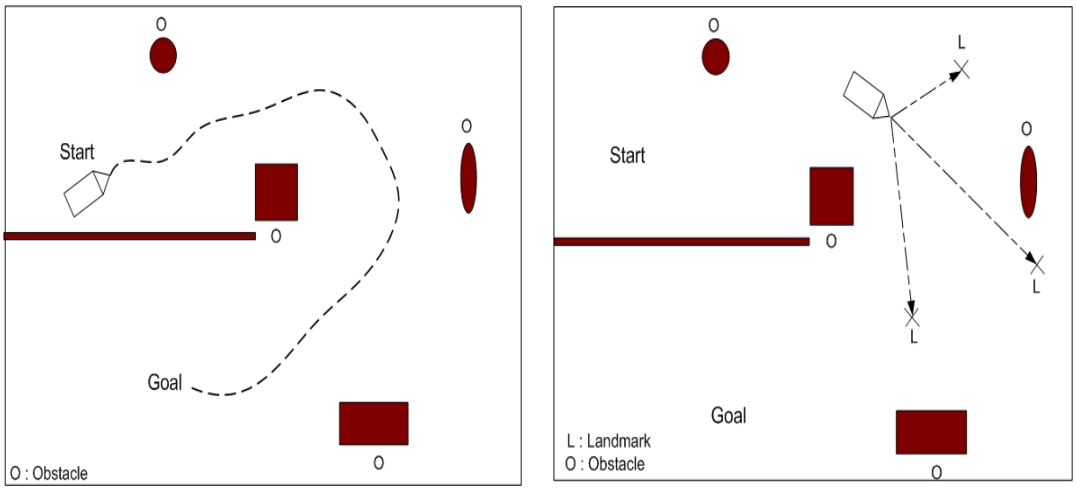


Figure 1.1: Robot navigation modalities. (a) Line following; (b) landmark triangulation.

and global robot localization. In local robot localization, also known as tracking, the initial pose of the robot is known. In global robot localization, or the kidnapped robot scenario, the problem is more challenging since the robot is given no information about its initial whereabouts and has to find itself without any prior information. Markov Localization (ML) [8] and Monte Carlo Localization (MCL) [9] are very common in the solution of such problems. In outdoor navigation, good global localization can be achieved via Global Positioning Systems (GPS). In some environments, such as forests, urban canyons, sub-sea, extra-terrestrial, or inside underground mines, GPS information is not available. Nevertheless, it is still possible for a robot to globally localize itself by a process known as Simultaneous Localization and Mapping (SLAM).

1.2.1 Simultaneous localization and mapping

In SLAM, the robot builds a map of the environment in which it is navigating, while simultaneously localizing itself with respect to that map (Figure 1.2). When a map of the robot's environment is available, navigation is relatively simple by localizing the robot using its onboard sensors [10, 11] and path planning towards the goal [12]. Conversely, when the absolute position of the robot is known at every time frame (*e.g.*, using GPS in an outdoor environment), mapping of the environment is

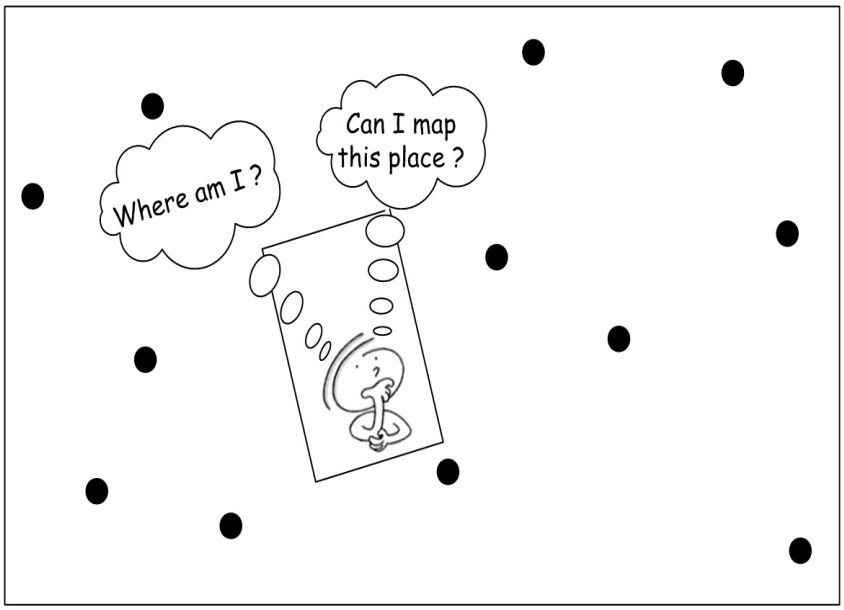


Figure 1.2: SLAM is the problem of localizing a robot while simultaneously building a map of its surrounding milieu. Black dots are landmarks that can be used to build a map.

also considered a solved problem [13]. It is when both the pose of the robot and the location of the landmarks are unavailable (*i.e.*, SLAM scenario) that the problem becomes convoluted because the errors in pose of the robot and landmark locations are correlated. More specifically, if at time 't', the pose and the kinematics of a robot are known, then at time '$t+1$' its pose can be estimated by integrating the angular and linear acceleration between t and $t+1$ and adding the result to the pose at time t. At the robot's new location the position of any observed landmark is estimated based on its relative distance from the robot. Since the robot's estimated coordinates are not precise, the landmark position estimates are subject to an error which is dependent on that of the robot pose (Figure 1.3). As the robot continues to navigate, the error in its ego motion estimate increases, which results in a larger error in the estimated position of new landmarks. SLAM addresses this error growth problem by maintaining a covariance matrix, which correlates between the errors in all the variables of the state vector, including the position and bearing of the robot and surrounding landmarks. At each time step, the covariance matrix is propagated forward in time, indicating the inter-dependence between variables. The Kalman

Filter (KF), the Extended Kalman Filter (EKF), or the Particle Filter (PF) are the traditional mechanisms by which the state and covariances are propagated between time steps. If at some time later during the robot's journey, it returns to a previously viewed scene (a situation ubiquitously know in SLAM as loop closure), and the robot recognizes previously viewed landmarks which have low uncertainty, the position of the robot can be updated with high certainty. More importantly, the uncertainty in the position of all the landmarks which have been seen so far can also be reduced since they are correlated (through the covariance matrix) to the uncertainty in robot pose.

SLAM is difficult because of three main reasons: (1) dimensionality explosion, (2) robust landmark detection and (3) *data association*. The processing requirements for traditional SLAM systems using a Kalman filter are $O(N^2)$, where N is the number of landmarks. As the number of landmarks increases, SLAM implementation becomes intractable due to the quick growth in the dimension of the problem; an issue referred to as dimensionality explosion. Since the seminal paper of Smith and Cheeseman [14], which sparked research in SLAM, the focus of most research in this field [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28] has been directed towards developing the underlying theory of SLAM and reducing its order of complexity in order to avoid dimensionality explosion. Towards this end, researchers relaxed the landmark issue of SLAM by using simple features which are easily detected and recognized as landmarks,. After resolving many of the issues of dimensionality explosion [24, 29, 30], growing interest mounted in applying SLAM in more challenging environments, such as in aerial [31], underwater [32], and large land-based [23] outdoor settings. Although the SLAM methodology remains the same in such settings, implementing the SLAM filter is more difficult because of the increased size of the mapped area and the inherent uneven and unstructured terrain conditions. In such settings, the problem of landmark detection has to be addressed.

Each iteration of the SLAM filter is either a predictive or an update step. In a predictive iteration, SLAM predicts the ego-motion of the navigating vehicle via an onboard dead-reckoning sensor. In an update iteration, SLAM uses range and bearing (or bearing alone) information to update the vehicle ego-motion estimate and the position of the landmarks surrounding the vehicle. SLAM uses the discrepancy (a measure called innovation) between its prediction of the pose of a landmark and the instantaneous observation of the pose of the same landmark to perform an update iteration. Therefore, for SLAM to succeed, the system must be capable of detecting landmarks and associating a previously-viewed landmark to new observations of the same landmark; a problem known as data association.

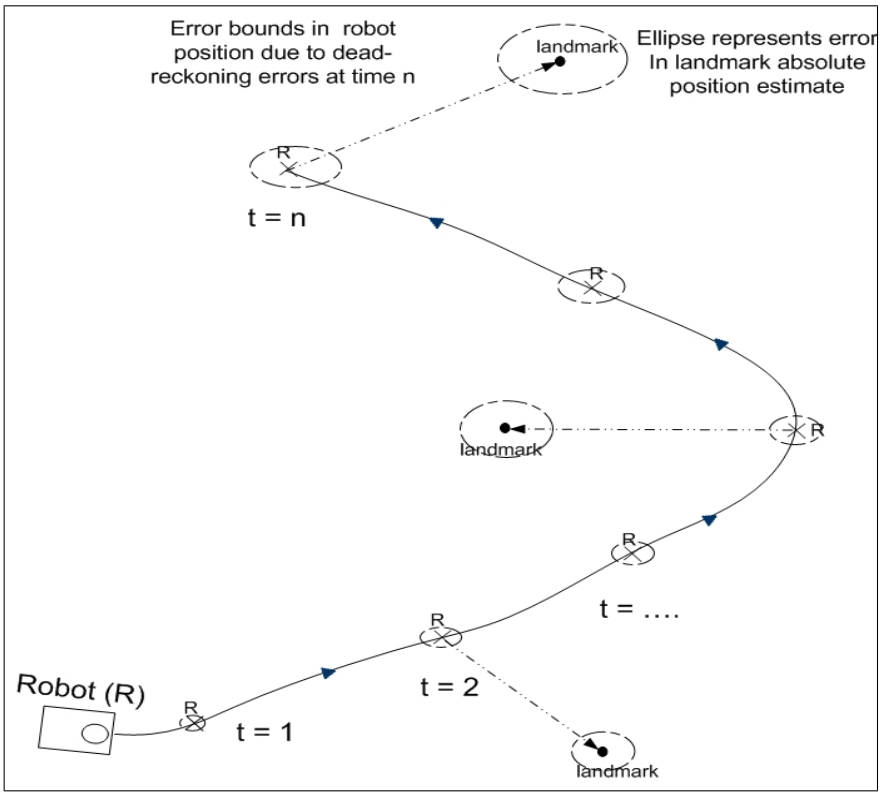


Figure 1.3: Error growth in landmark position estimate is associated with the error in robot position. The ellipses around the robot position and Landmarks represents the errors in their estimate. Notice how the error in the landmark estimate grows as the uncertainty in the robot position grows.

Data association of a landmark is dependent on the complexity of the shape of that landmark and the capabilities of the exteroceptive sensors onboard the robot. In small scale SLAM, data association is not a problem because the test site can be set up with a number of artificial beacons that are salient and highly distinctive from the background. In large environments, the cost and time of setting up a test site with artificial beacons, is not justifiable. Natural landmarks, such as trees, rocks, and shrubs are readily available outdoors but are more difficult to detect than artificial landmarks. Natural features exhibit large variance in their size, shape, orientation, color, and texture. It is therefore difficult for sensors with limited bandwidth of information to successfully detect and recognize natural features.

The most common type of range-bearing sensor that is used for SLAM is the sonar for indoor and underwater settings and the Laser Range Finder (LRF) for

outdoor land settings. Cameras have been recently proposed as an alternative to these sensors. With the growing commercialization of digital cameras, good quality cameras can be found which are cheap, small, and weigh very little. Cameras are passive sensors which consume very little power and are easily integrated into embedded navigating systems. Furthermore, there has been much work in the Computer Vision community on segmentation, and although this remains an unsolved problem, such routines can be utilized to help detect objects. In Computer Vision, landmarks can be matched in feature space, by extracting high level primitives of each landmark and comparing them to those of model landmarks. Cameras can provide rich information about the color and texture of the environment; information which can be used as top-down *a priori* knowledge about the type of landmark expected in such a setting [33].

1.2.2 SLAM using vision

The majority of Vision SLAM systems implemented to date [34, 35, 36, 37, 38, 39, 40, 41, 42, 43] use local image saliencies called Interest Points (IP) as landmarks for SLAM. IPs are regions within the image which are distinctive from the rest of the image. Although such systems are typically fast to implement, the maps that they generate serve little purpose other than to perform SLAM because IPs are too abstract for any human to use as a reference. Figure 1.4 (a) shows the map generated by Se *et al.* [43] using Scale Invariant Feature Transform (SIFT) features as landmarks. Notice that these features (shown in 1.4 (b)) lack the semantics to be used as references for humans.

Alternatively, some Vision SLAM systems [44, 31, 45] exist which use real objects as landmarks; however, all such systems use artificial rather than natural objects as landmarks. Although efficient for SLAM, artificial landmarks do not add any new information about their environment. In some settings such as in an earthquake aftermath or on the planet Mars, the test site is not accessible to man and natural landmarks which are indigenous of their setting are more appropriate than artificial landmarks to perform SLAM. Natural features have been traditionally avoided because they vary considerably in shape and size amongst each other; nevertheless, when one considers the above issues and the cost of setting up the site with a sufficient number of artificial beacons, natural features become an attractive alternative, especially if the landmarks are simple enough to be quickly detected.

In this context, the first motivation of this work is to develop a Vision system that can detect natural objects for the sake of using them as landmarks for SLAM. Examples of natural features could include trees, shrub, ponds, and rocks. Although

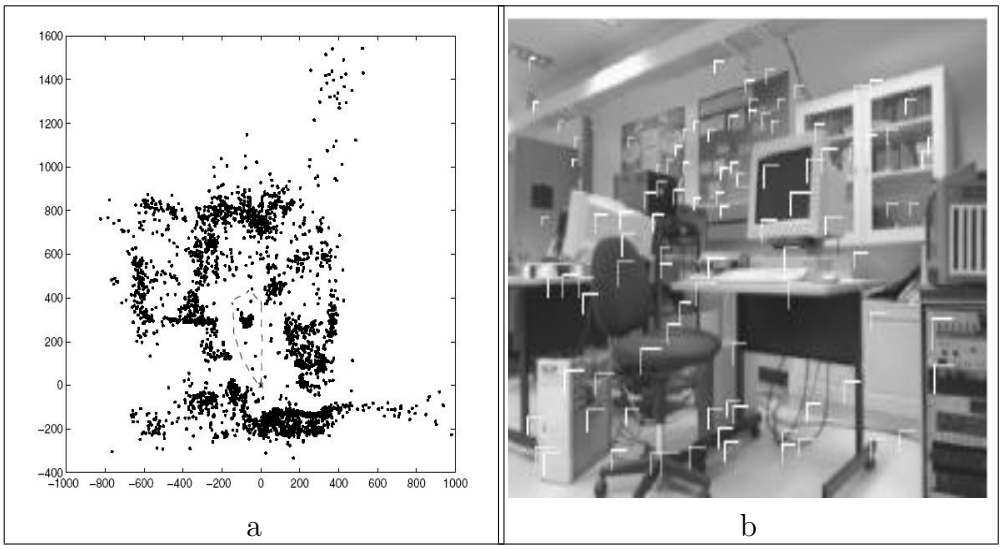


Figure 1.4: (a) Map generated during SLAM of (a) Se *et al.* [43] with the corresponding SIFT features (b). Notice that such maps and features would be of no use for a human navigator.

the system in this work is tailored for tree trunks, the developed techniques are easily extendible to other natural objects.

In most SLAM implementations, landmarks are matched to previously viewed landmarks (an issue referred to as data association) by means of some statistical procedure such as the Nearest Neighbor (NN) [46], or Joint Compatibility Branch and Bound (JCBB) [47]. Such systems become problematic once the error in the vehicle pose estimate is high. For instance, in Figure 1.5, the robot observes a feature and estimates its position with an associated uncertainty (ellipse), which makes both landmarks 'A' and 'B' viable candidates for a match. In such situations, SLAM systems are usually programmed to avoid making an association and to ignore the observation altogether; such methods suffer from too few updates, which can compromise the consistency of the SLAM filter. It is clear at this point that a system capable of recognizing landmarks without using statistical inference at all would be a large contribution to the field of SLAM. This goal constitutes the second motivation of this thesis.

Vision SLAM systems can be classified based on the type of vision sensor used: range-bearing vision SLAM using multiple cameras or bearing-only vision SLAM using a single camera. In range-bearing SLAM, stereo or trinocular vision is used to calculate the bearing as well as depth of a landmark from one viewpoint. The

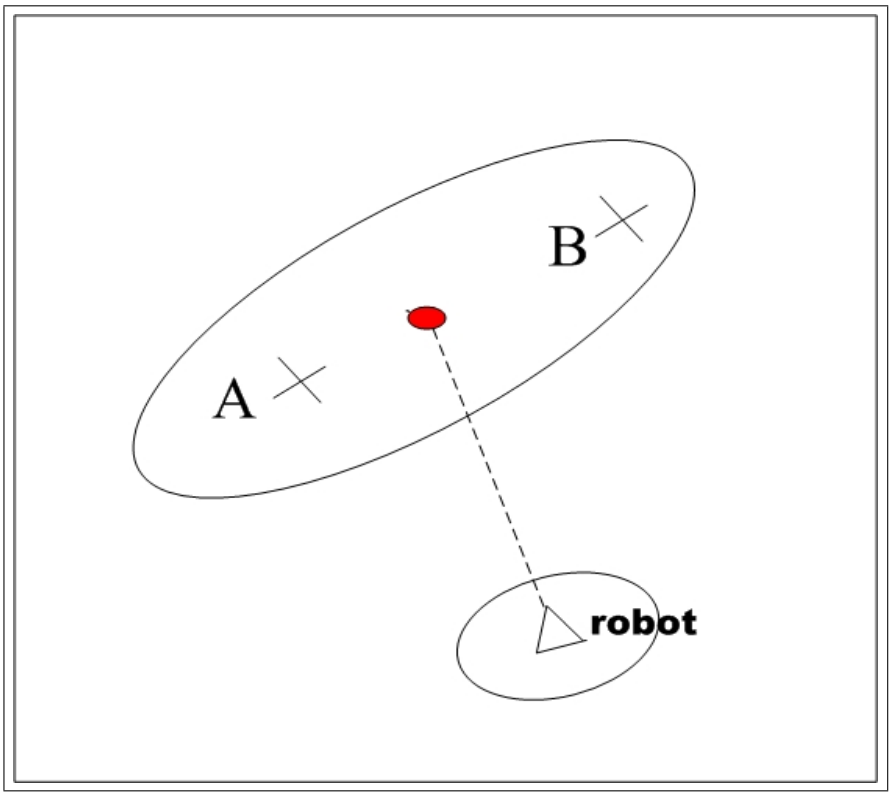


Figure 1.5: Data association failure. If at given time a robot's pose is estimated with a given uncertainty (ellipse around triangle) and observes a landmark (solid circle), it tries to match the observation to one of two known landmarks A or B.

main contributions here consist of the work of Davison *et al.* [35, 37], Se *et al.* [43], Panzieri *et al.* [44], Fitzgibbons [45], Jung and Lacroix [48, 38], and Kim and Sukkarieh [31]. Range-bearing SLAM systems have demonstrated good results indoor and outdoor; however, the main pitfall of this method is the requirement of a relatively large baseline between the cameras of the stereo rig. The precision of depth values obtained from a stereo rig is proportional to its baseline and inversely proportional to the square of the observed depth [38]. Tests have shown [38] that depth values extracted via stereo cameras degrade for baseline to depth ratios smaller than 1/30, which corresponds to depths of the order of 6 meters for conventional stereo rigs, whose baselines are less than 20 centimeters. While such depth capabilities are acceptable for indoor purposes, outdoor range sensors used for SLAM are expected to detect landmarks that are sometimes located 15 to 30 meters away, requiring stereo rigs featuring baselines of the order of 50 centimeters

to 1 meter to insure precise depth estimates. This minimum baseline size constraint precludes performing outdoor Stereo-Vision SLAM on small vehicles, which can not accommodate such wide stereo camera rigs, and promotes the idea of Bearing-only (B-only) Vision SLAM, using a single camera for detecting landmarks. In B-only SLAM systems the depth of a landmark is determined by correlating the bearing readings from several viewpoints with the distance the robot travels between bearing observations. Bearing-only Vision SLAM contributions include the work of Deans and Hebert [49], Davison [36], Fitzgibbons [45], Kwok *et al.* [39, 40, 41], Kim and Sukkarieh [31], Sola *et al.* [50, 42], and Bailey [51]. None of these systems presents a conclusive solution to the problem of B-only SLAM, where each of them suffers from either intractable processing requirements or dimensionality issues.

The third motivation is to develop a hybrid between range-bearing and bearing-only systems, where range information is used only to initialize a landmark into the SLAM map when a tree trunk is within the range capabilities of the stereo camera. When the robot navigates out of this confidence region, the SLAM system uses its bearing-only information (*i.e.*, azimuth and elevation of landmark) to continue performing SLAM. In this fashion, bearing-only and range-bearing complement each other to produce a robust initialization and tracking system. This idea agrees favorably with that of human depth perception, where studies [52] have shown that humans use stereo as the dominant depth source for up to one meter. Above this range, other cues are used to infer depth such as occlusion, height in the visual scene, relative size, relative density, shading, and texture.

Uneven terrain conditions create issues for both the proprioceptive and exteroceptive sensors onboard the navigating vehicle. The most common dead-reckoning sensors are based on wheel encoders, which are sensitive to slippage; an issue so common when the terrain is not flat. Traditional approaches to deal with slippage are to use an Inertial Measurement Unit (IMU) as an alternative dead-reckoning sensor or to supplement the encoder with an IMU to correct errors caused by the encoders [53].

The forth and final motivation is to integrate inertial predictions and visual observations into a working Visual-Inertial SLAM system and to test it on real data collected during a run in an outdoor unstructured environment. The integration of such a system constitutes a significant step forward in the development of a stand-alone product to be applied in non-wheeled kinematics such as for humanoid robotics or to act as a navigation aid for the visually impaired. Many SLAM systems avoid testing in unstructured outdoor environments because it is difficult to account with the large variabilities in these settings. Nevertheless, such tests are necessary to comprehend the limitations and strength of one's system. In the

next sections, the objectives of the developed system are presented, as well as its architecture and anticipated challenges.

1.3 Objectives, architecture, and challenges

1.3.1 Objectives

The long term intent of the author's research is to develop a system capable of navigating between different environmental settings (*e.g.*, park, urban), recognizing the milieu it is in, choosing landmarks that are appropriate for the recognized environment (*e.g.*, trees for park and lampposts for urban), and performing SLAM using an inertial sensor to predict motion and a camera to detect landmarks. Initial efforts aim at solving this problem in a single environment; namely, an outdoor park environment, using tree trunks as landmarks for SLAM. Towards this end, the first objective of the developed system is to recognize that the navigating vehicle is indeed located in a park setting. Once the system has verified its surrounding context, the second objective is to be capable of segmenting and detecting tree trunks, and determining their 3D pose (*i.e.*, position and bearing) in the vehicle's Field of View (FoV). The third objective constitutes what is known as data association, where the system is required to differentiate between observed trees and associate a previously viewed tree to new observations. The SLAM filter uses the discrepancy between the predicted location of the landmark and its new observation to update the ego-motion estimate of the navigating vehicle.

1.3.2 Architecture

A Vision SLAM system is proposed, named VisSLAM, whose structure is presented in Figure 1.6. VisSLAM operates within the framework of an Extended Kalman Filter (EKF); fusing state predictions with landmark observations to propagate the state and covariance matrices at each time step. The state vector includes the position, velocity, and orientation of the navigating vehicle, as well as the 2D coordinates of landmarks around the vehicle. State predictions are achieved via an Inertial Navigation System (INS) and detection and localization of landmarks are obtained via a stereo camera. VisSLAM is comprised of three fundamental building blocks; namely, the Inertial Navigation System (INS), the Computer Vision (CV) System, and the SLAM system.

The INS system is responsible for predicting the state of the robot at each SLAM predictive iteration. The INS uses a low-cost strapdown Inertial Measurement Unit (IMU) to measure the linear accelerations and angular velocities of the moving platform and subsequently predict the motion of the vehicle. Care must be taken to calibrate the IMU before navigation commences to avoid detrimental effects that the biases of an IMU have on its predictive capabilities.

The CV system is responsible for two independent tasks: the first task is to perform Environment Recognition (ER) before SLAM commences in order to insure that the robot's environment is indeed that of an outdoor park. Future research could involve performing ER online during SLAM to give the vehicle dynamic feedback of its instantaneous milieu. ER is implemented in a holistic fashion, based on the spectral composition of each image. The details of the ER system are presented in Chapter 4. The second task of the CV system is to detect natural objects for the sake of using them as landmarks for SLAM. The type of landmarks to use is defined by the context of the scene; in this work the context is that of a park and tree trunks are the most appropriate type of natural landmarks for such as setting. Tree detection is performed by segmenting images, using a mixture of brightness and texture cues, into symmetric and continuous lines and grouping these lines into potential trees. Landmarks are initialized into the SLAM map based on the quality of the depth map at the landmark positions. Trees that are within the confidence range of the stereo camera (set to 7.48 meters in Section 4.5) are immediately initialized into the SLAM state and covariance matrix. Landmarks which are outside the range of the stereo camera are still detected and their bearing to the camera is recorded. All initialized landmarks are compiled in a database along with their corresponding coordinates. Each tree trunk that is sighted in subsequent images is first tested for a match against all initialized trees in the compiled database. If a match is successful, the difference between bearing observation of the landmark and the predicted bearing observation (a metric called Innovation) is used to update the vehicle pose and landmark position. Trees are matched and recognized via a system which compares Scale Invariant Feature Transform (SIFT) features within the boundaries of the query tree to SIFT features corresponding to model tree trunks. If the observed tree does not successfully match any trees in the database, and it is within the confidence range of the stereo system, an attempt is made to initialize it as a new landmark for the SLAM map.

The prediction-update loop continues until the robot performs an entire cycle inside the test site and returns to its starting point. The ability of the system to recognize that a loop has ended and that it is back at its starting point is known as loop closure.

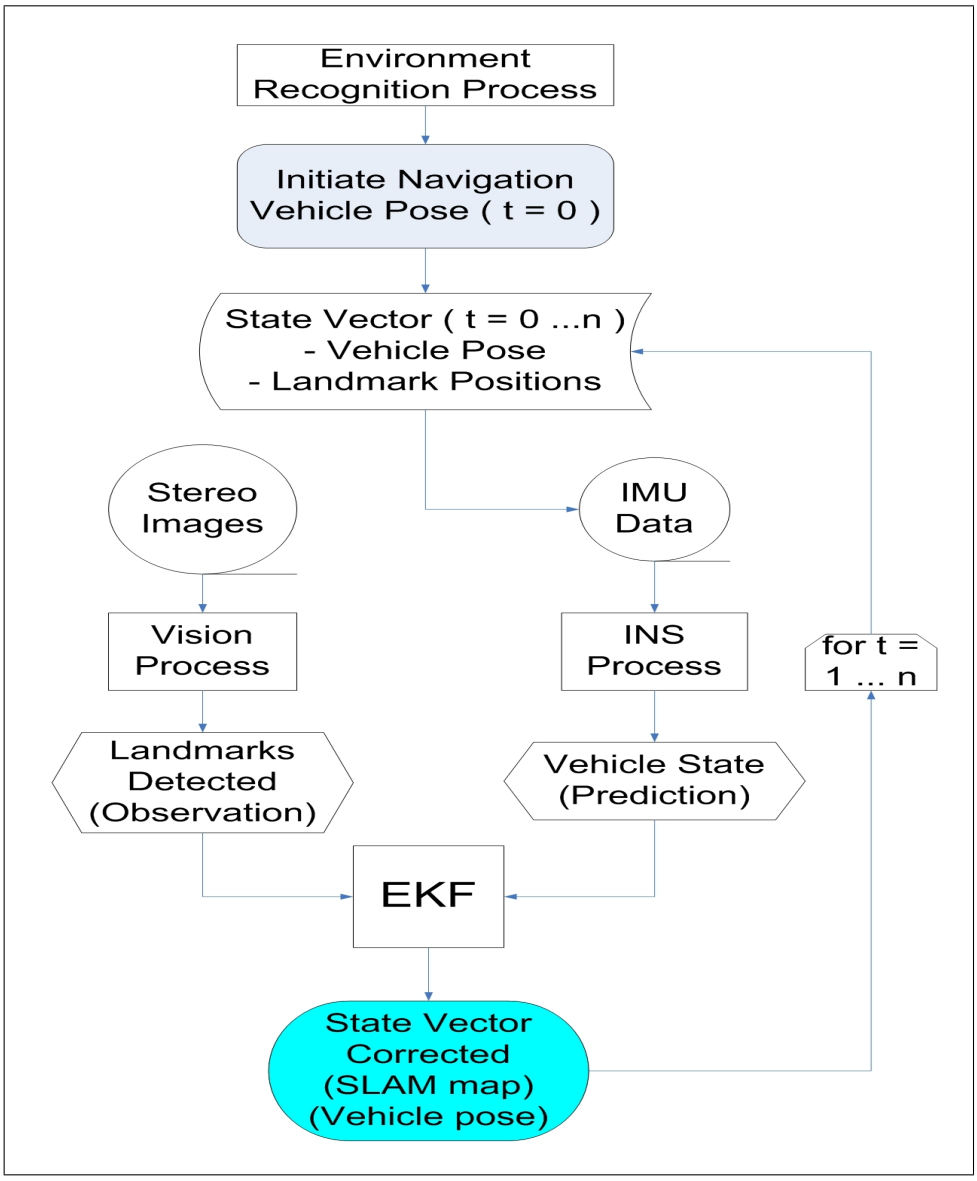


Figure 1.6: VisSLAM system architecture. Environment recognition is first conducted to determine the context and type of landmarks to track. An EKF is used to integrate dead-reckoning information with observations of landmarks. Dead-reckoning is done via an IMU. Detection and localization of landmarks is done via a stereo camera.

1.3.3 Challenges

The challenges facing VisSLAM are numerous including ground truth establishment, tree detection, tree recognition, tree initialization, and real-time issues.

Ground truth. VisSLAM performance is established by comparing its results to ground truth, which includes the position of landmarks and the position and heading of the vehicle during its complete journey. The traditional approach used by surveyors for fixing the coordinates of an object outdoors is to position oneself at the desired location, while holding a hand held Global Positioning System (GPS), and recording the longitude and latitude at that position. This approach is not possible for locating trees since they exhibit dense foliage which causes faulty GPS readings due to the deflection of the incoming GPS electromagnetic signal by the overhead foliage.

Tree detection. Object recognition has received extensive attention in the Computer Vision community for several decades and remains far from being a solved problem. Furthermore, none of these systems have attempted to segment natural features such as trees from background foliage. Trees are natural objects which are conventionally considered as part of the background in images. Trees share similar brightness and texture patterns as their surrounding background, which makes it difficult to place thresholds for segmentation. These difficulties are compounded by the issue that most trees also change drastically in appearance between seasons.

Tree recognition. In order for SLAM to succeed, it must be capable of recognizing the landmark in order to calculate an innovation based on the saved state of this landmark. Even if the CV system successfully detects the presence of a tree, there is no guarantee that it will successfully identify which of these trees it is. Object recognition is a very complex issue in Computer Vision which is an open problem as of yet.

Tree initialization. Tree initialization signifies calculating the coordinates of a tree and adding these coordinated with their corresponding covariance into the SLAM state and covariance. The coordinates of a landmark to a camera are directly related to the depth of the landmark to the camera. Depth can be calculated using stereo vision if multiple cameras are available or by calculated depth from two bearing readings if only a single camera is available.

Real time issues. The largest challenge facing SLAM systems in general is their real-time requirements. Although it is very appealing to perform VisSLAM in real-time, this constraint is relaxed here since the main concern is functionality of the system. Real-time issues can be dealt with in future work.

1.4 Principal contributions

The impact of the proposed research is twofold: first, to the Computer Vision community, detection and recognition of natural objects in cluttered outdoor environments, and secondly to the SLAM community, the application of the Computer Vision system to track natural landmarks for SLAM. A system is proposed which is original and unique due to the following contributions

1. VisSLAM presents a Computer Vision system that segments trees from background clutter. Images are segmented into quasi-vertical lines which are subsequently grouped into trees based on symmetry, continuity and minimization of image entropy. Trees are recognized by matching high level primitives extracted from the query tree to those features corresponding to model trees. VisSLAM also contributes to the field of Computer Vision by introducing an environment recognition system built on an Artificial Neural Network (ANN), which uses the holistic spectral content of images to predict the environmental context.

2. VisSLAM presents an INS system specifically designed for land-based navigation. Experimental analysis of a approximate IMU calibration method is performed and the results compared to real calibration results. Two theories from the literature of non-holonomic constraints are implemented and evaluated. While the first is refuted due to shortcomings in the method, the second, which is based on and EKF, is proved to be successful.

3. Another major contribution is the integration of different technologies into an operational land-based Vision-Inertial SLAM system which is tested on real data from a navigation run in an outdoor park. The VisSLAM Computer Vision system including landmark detection, recognition, and initialization succeeds at bounding the INS errors first in a localization scenario and then in a full localization and mapping scenario.

4. The experimental part of this thesis puts forward a methodology for establishing ground truth for Vision SLAM systems and makes available a database

of synchronized IMU, Images and GPS data. This database constitutes the infrastructure for a benchmark where past and future Vision SLAM systems can be compared.

1.5 Thesis overview

This chapter presents a brief synopsis on robot localization and mapping with a specific focus on SLAM. The current challenges facing SLAM are dimensionality explosion and issues related to landmark detection, landmark matching and data association. The focus of research in this thesis are landmark issues, which are addressed by developing a SLAM system, named VisSLAM, which uses a camera to detect, recognize landmarks, and initialize landmarks; and an IMU to predict the robot motion at each time step. An EKF is used to integrate dead-reckoning state vector estimates with camera observations, yielding corrected state and covariance matrices at each time step. The remainder of this thesis is structured as follows.

Chapter 2 reviews the state of the art of Vision SLAM, first by describing the different methodologies of implementing Vision SLAM, and then by analyzing the various contributions to date in Vision SLAM.

Chapter 3 deals with the first of VisSLAM's building blocks, the dead-reckoning system, which predicts vehicle motion via an Inertial Measurement Unit (IMU). The system is tested on an IMU dataset corresponding to a real experimental run.

Chapter 4 explains the fundamental building block of VisSLAM, which is the Computer Vision system, including environment recognition, landmark representation, segmentation, and detection, landmark initialization and matching. The CV system is tested on a database of images of trees.

Chapter 5 demonstrates the developed inertial and Computer Vision systems in the context of SLAM by integrating them into a working Vision-Inertial SLAM system named VisSLAM. VisSLAM is tested on a data sequence recorded during a run in an outdoor area using tree trunks as landmarks. Three tests are performed to evaluate each of the CV systems (*i.e.*, tree detection, tree recognition, and the tree initialization).

Chapter 6 summarizes the contributions of this thesis and introduces the focus of future research.

Chapter 2

Background

2.1 Introduction

There is growing interest in the SLAM community to use vision rather than laser or sonar as the robot's exteroceptive sensory modality. Cameras are passive sensors, which are virtually jam proof. They are light, cheap, and consume relatively low power. Furthermore, cameras transfer a large bandwidth of information, which if properly managed can be used to detect landmarks in cluttered environments, where laser or sonar fails.

Vision SLAM has been researched in the Computer Vision community since the early eighties under a variant name called Structure from Motion (SfM) [54, 55, 56, 57, 58]. SfM refers to the problem of recovering the 3D structure of a scene and the motion of a camera inside it by analyzing a sequence of images taken by the camera of this scene. In SfM, the camera is used to both estimate its ego-motion and detect the position of surrounding landmarks. In contrast, many of the Vision SLAM systems developed to date use some other form of sensor (*i.e.*, wheel encoder or IMU) for dead-reckoning purposes and use the camera exclusively to detect landmarks. The main differences between SfM and Vision SLAM are related to loop closure requirements and processing time. In Vision SLAM, a global map of the vehicle setting must be constructed using landmarks that the system detects during navigation. Once a landmark is initialized into the SLAM map, its position is continuously updated, such that when it is revisited, the system can re-predict its location with high accuracy. Drift occurs when the position of a landmark is predicted at an offset from its original estimate. In SfM, there is no need for such a global map; rather, SfM produces only local 3D information which is not preserved

when the vehicle navigates away from the scene. Therefore, when a previously viewed scene is traversed the SfM system does not recognize it and treats it as a new environment. Vision SLAM requires the detection and recognition of landmarks in real-time using Computer Vision; a task which creates an inherent challenge since object detection algorithms in Computer Vision are far from becoming a solved problem and yet further from being run in real-time. SfM does not necessitate any real-time processing and can be run in an off-line scenario using a batch process.

In Vision SLAM implementation to date, landmarks have either been real objects, or image saliencies (known as Interest Points) that do not necessarily possess any semantic grounding, but are distinctive enough to be repetitively detected and recognized. The advantage of using object-based landmarks is that they generate sparse maps that are not susceptible to dimensionality explosion issues which are common in Interest-Point based systems. The intent of this chapter is to provide the background for Vision SLAM methodologies and then to present the state of the art in Vision SLAM systems. More specifically, in the next four sections, a Vision SLAM taxonomy is presented which is based on four criteria: (1) the type of the landmark (*i.e.*, Interest-Point-based or object-based), (2) the type of sensor (*i.e.*, range-bearing or bearing-only), (3)the setting (*i.e.*, indoor or outdoor), and (4) the type of dead-reckoning sensor (*i.e.*, IMU or wheel encoder). Section 2.6 then presents the state of the art in Vision SLAM, segregating system based on the forgoing four criteria.

2.2 Type of landmark

Vision SLAM systems rely on the camera to capture images of the surrounding scene and on systems developed in Computer Vision to detect and recognize landmarks in these images. The camera's ability to detect landmarks is dependent on the complexity in shape, color and texture of the sought landmarks, and on the sophistication of the visual object detection system. Unfortunately, the state of the art of object detection and recognition in Computer Vision is far from being applicable to generic objects in any setting. In light of this weakness, developers of Vision SLAM systems use easily detectable landmarks such as image saliencies that do not necessarily correspond to objects but that can be quickly detected, or real objects which are highly salient. The former type of landmarks are referred to as Interest-Point-based or IP-based, while the latter are referred to as object-based.

IP-based landmarks include features such as the Scale Invariant Feature Transform (SIFT) features [59], moment invariants [60], differential invariants [61], com-

plex filters [62], Steerable filters [63], Harris Corners (HC) [64], Scale Adaptive Harris Corners (SAHC) [65], edges [66], or Salient Normalized Intensity Regions (SNIR) [67]. SIFT features [59] are the most stable of the IP types [68], and are invariant under changes in camera viewpoint and variable lighting conditions. These invariant properties ascertain dependable repeatability (*i.e.*, detecting a previously viewed landmark) and good data association (*i.e.*, recognizing a previously viewed landmark and maintaining the correspondence between a measurement and that landmark). The second best IP type is based on the Steerable filter [63], which produces stable results but is computationally expensive. Corners and intensity based regions are fast to compute but are not as robust as SIFT or the Steerable filter. Of the possible IP types, only SIFT, HC, SAHC, SNIR, and edges have been used as landmarks in Vision SLAM implementations. The justification for these choices is a compromise between IP robustness and the speed of detecting the IP. Landmarks represented by IPs can either be the IPs themselves, combinations of IPs into stable clusters which do not necessarily correspond to real objects, or combinations of IPs that represent objects.

Object-based landmarks are either artificial man-made landmarks or natural landmarks. The former type have been traditionally selected for Vision SLAM because they are relatively easy to detect due to their predictable and structured properties. Artificial beacons are chosen that are salient in color, texture, shape and size. The disadvantage of using these man-made objects for SLAM is the cost and time of setting up the test site with a large enough number of them to satisfy SLAM. Natural features do not require any site preparation but are difficult to detect because they exhibit large variance in their properties. Nevertheless, if natural features with relatively structured shapes are used (*e.g.*, tree trunks) it is postulated that they can be detected at rates that satisfy SLAM. In order for SLAM to succeed, the system must be capable of repeatedly detecting the same landmarks and associating a previously-viewed landmark to new observations, regardless of the lighting conditions and viewing direction of the camera (*i.e.*, scale and orientation of the landmark); a detailed account of these issues is differed to Chapter 4 that deals with object recognition in Computer Vision.

Table 2.1 compares the advantages and disadvantages of IP-based verses Object-based landmarks. The entries in the table are labeled as true 't', false 'f', inclined to be true '$\hat{t}$', and inclined to be false '$\hat{f}$'. Most of the IP-based landmarks can be quickly detected but are prone to dimensionality explosion issues, some are invariant to camera viewing direction (*e.g.*, SIFT) and others are not (*e.g.*, edges, Harris corners). Most of the IP-based SLAM systems generate dense occupancy maps comprising of landmarks with no underlying semantic significance. Researchers in

IP-based SLAM deal with the issue of denseness by either reducing the number of IPs by selecting the top 'n' most salient of them in each image [40], or by gating the IPs to a certain size [37, 35, 36].

	Landmark Type		
	IP-based	Object-based	
		Artificial	Natural
Fast Detection	$\hat{t}$	$\hat{f}$	$\hat{f}$
Scale Invariant	$\hat{t}$	$\hat{f}$	$\hat{f}$
Small # landmarks	f	t	t
Dimensionality Explosion	$\hat{t}$	f	f
Model Required	f	t	t
Sparse Map	f	t	t
Dense Map	t	f	f
Infrastructure Preparation	f	t	f

Table 2.1: Comparison of the advantages and disadvantages of IP-based verses object-based landmarks for SLAM. The entries in the table are rated as t (true), f (false), $\hat{t}$ (inclined to be true), and $\hat{f}$ (inclined to be false).

Object-based SLAM systems generate sparse maps, comprising of landmarks that are real objects. The implementation time of these systems is at par with that of object detection routines which are generally slower than IP-based detection. Object-based SLAM requires the additional work of setting up a model for each type of landmark but avoids dimensionality explosion issues due to the relatively low number of landmarks that are tracked. Using artificial landmarks for SLAM is costly and time consuming because of the need to set up the site with a large enough number of beacons to satisfy SLAM. Natural landmarks are readily available inside their environment but are difficult to model, detect, and recognize due to a large variance in their shape, size, color, and texture.

2.3 Type of vision sensor

Vision SLAM systems are developed differently based on the number of cameras that are used. If more than one camera is available (stereo or trinocular), the vision system is capable of reporting the bearing as well as the depth (*i.e.*, via stereopsis) of surrounding landmarks; this is known as Range-Bearing Vision SLAM. If, on

the other hand, only one camera is available, the vision system looses its depth perception but retains its ability to determine the relative bearing of a landmark; such systems are known as Bearing-only (B-only) Vision SLAM. B-only Vision SLAM is an idea that stems from bearing-only tracking [69, 70, 71] but is easier to implement since it only requires the estimation of position, whereas bearing-only tracking also estimates the velocity of the vehicle. Because of their inability to infer depth, B-only sensors can not initiate a landmark into the SLAM map (*i.e.*, determine its coordinates) from one sighting; rather, at least two readings are required from different viewpoints to subsequently calculate the landmark position. These two positions must be separated by a large enough baseline in order to avoid ill-conditioned situations (Figure 2.1). The 2D position of a landmark is located at the intersection of two lines, drawn from the robot position to that of the landmark at different times i and j [51]; these lines are expressed analytically as (2.1a) and (2.1b) respectively.

$$y_L - y_{Vi} = tan(\alpha_i + \beta_i)(x_L - x_{Vi}) \tag{2.1a}$$

$$y_L - y_{Vj} = tan(\alpha_j + \beta_j)(x_L - x_{Vj}) \tag{2.1b}$$

where x_{Vi}, y_{Vi} and x_{Vj}, y_{Vj} are the coordinates of the vehicle at time i and j, α is the bearing of the vehicle in the global coordinate frame and β is the landmark bearing in the vehicle coordinate frame. Manipulating (2.1a) and (2.1b) and rearranging their components, the intersection of the lines determines the 2D position of the landmark (x_L, y_L) expressed as

$$\begin{bmatrix} x_L \\ y_L \end{bmatrix} = \begin{bmatrix} \frac{x_{vi}s_ic_j - x_{vj}s_jc_i + (y_{vj} - y_{vi})c_ic_j}{s_ic_j - s_jci} \\ \frac{y_{vi}s_ic_j - y_{vj}s_jc_i + (x_{vj} - x_{vi})c_ic_j}{s_ic_j - s_jci} \end{bmatrix} \tag{2.2}$$

where

$$\begin{aligned} s &= \sin(\alpha + \beta) \\ c &= \cos(\alpha + \beta) \end{aligned}$$

Research in B-only Vision SLAM [34] reports that (2.2) is ill-conditioned for bearing sightings separated by an angle θ less than 40 degrees. As a result of this minimum baseline criterion, most B-only SLAM systems involve a wait period from the time a landmark is first observed until a sufficient baseline is insured before it

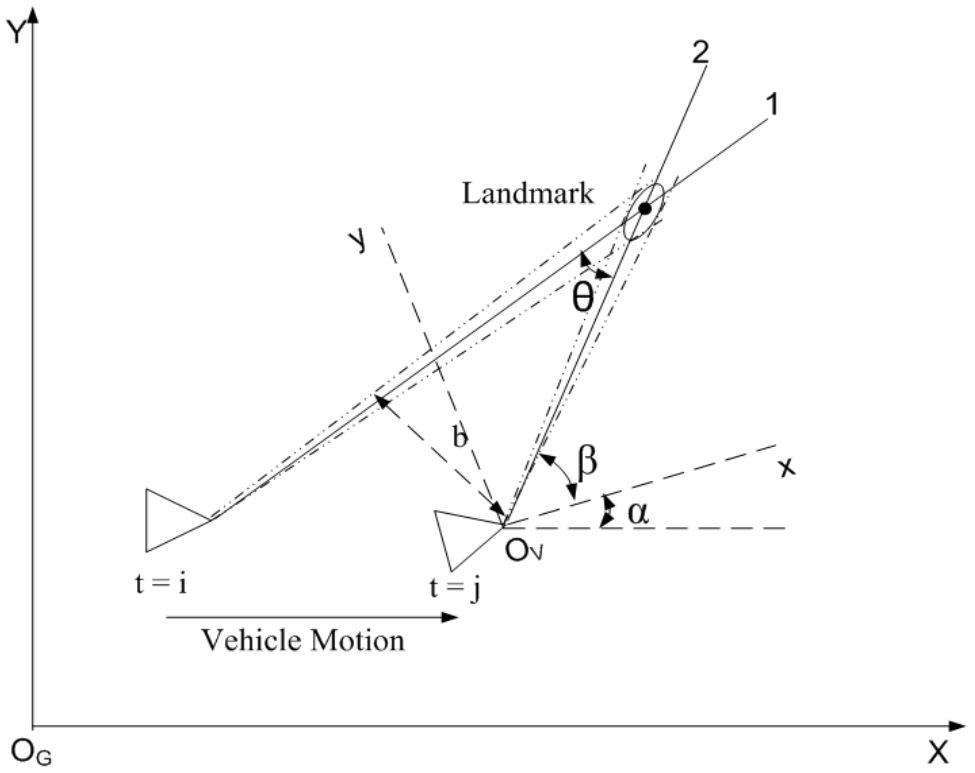


Figure 2.1: Landmark initialization requires a minimum bearing. Θ is the angle between different sightings, α is the bearing of the vehicle in the global coordinate frame O_G, β is the bearing of the landmark in the vehicle coordinate frame O_V, b is the baseline between sightings, and t is the time at which the sightings are taken.

may initialized into the SLAM map. These delayed systems are known as delayed-initialization B-only SLAM [42, 34, 51, 49, 50, 45]. In undelayed initialization B-only SLAM systems [36, 40, 41, 50], landmarks are initialized into the SLAM map in real-time by hypothesizing multiple depths for each landmark and subsequently pruning bad hypothesis from the map based on consequent observations. There are several flavors of delayed and undelayed B-only Vision SLAM systems, which are presented in Sections 2.6.3 and 2.6.4.

Table 2.2 presents the differences between R-B and B-only SLAM with their corresponding advantages and disadvantages. In R-B Vision SLAM the visual system determines the depth and bearing of the landmark from one sighting. Furthermore, in R-B systems the depth of a landmark can be used as an additional cue for data-association; landmarks are matched based on their characteristic features

and on their 3D position relative to the camera. In B-only SLAM, the landmarks can be matched based on their characteristic properties or on bearing triplets [51] and the depth of each landmark is subsequently inferred. B-only SLAM is more processor intensive than R-B SLAM because of the delayed initialization operation, which involves keeping track and correlating the different poses of the robot and landmarks until conditions are appropriate for initialization. The advantage of B-only systems is primarily flexibility because a single camera is sufficient to implement SLAM. Furthermore, a single camera does not have to worry about loss of calibration (such as in stereo camera) caused by physical abuse of the camera.

	Sensor Type		
	Range-Bearing	Bearing-only	
		Delayed	Undelayed
Initialize from 1 sighting	t	f	f
Data association using depth	t	f	f
A single camera	f	t	t
Requires 2 or more cameras	t	f	f
Processor intensive	$\hat{f}$	t	t

Table 2.2: Range-bearing verses bearing-only sensors. The entries in the table are rated as t (true), f (false), $\hat{t}$ (inclined to be true), and $\hat{f}$ (inclined to be false).

2.4 Vision SLAM setting

The third criterion by which Vision SLAM systems are classified is their environmental setting, which affects both the prevailing lighting conditions and the nature of the terrain for the navigating vehicle.

Cameras are sensitive to the lighting conditions of the scene they are viewing. In indoor settings, lighting conditions can be made fairly homogenous regardless of the time of the day; in outdoor settings, objects reflect differently throughout the day. Furthermore, shading can cause detrimental effects on a segmentation system, suggesting boundaries that do not correspond to true separations of physical regions. Objects viewed in homogenous lighting conditions are more likely to be repeatedly detected than under changing lighting conditions.

Terrain conditions are a second matter of concern when considering different environments. In indoor spaces, the ground is relatively flat and slippage can be

avoided. Moreover, a camera that is fixed to a vehicle remains approximately at the same height, thereby reducing the viewing direction parameters from six ($pitch, roll, yaw, x, y$, and z) to five (*i.e.*, z is constant). Outdoor SLAM is performed in either an aerial, underwater, or land-based setting. In aerial SLAM [31, 48, 45, 34], range capabilities of the vision system are an issue and therefore promote using B-only SLAM rather than R-B SLAM. In underwater SLAM [72], issues including unstructured terrain, low-overlap imagery, and a moving light source must be considered. In outdoor navigation, dead-reckoning via wheel encoders is either not possible (aerial, underwater) or is prone to severe errors (on land due to slippage). In such environments, an Inertial Measurement Unit (IMU) or camera is used for ego-motion estimation.

Table 2.3 re-iterates the advantages and disadvantages of indoor verses outdoor settings in tabular form.

	Setting			
	Indoor	Outdoor		
		Land	Aerial	Underwater
Homogenous light	t	f	f	f
Shading issues	$\hat{f}$	t	t	t
Moving light source	f	$\hat{f}$	$\hat{f}$	$\hat{t}$
low-overlap imagery	f	f	f	$\hat{t}$
Uneven terrain	$\hat{f}$	$\hat{t}$	t	t
Unstructured terrain	$\hat{f}$	t	t	t
Range issues	$\hat{f}$	$\hat{t}$	t	$\hat{t}$

Table 2.3: Range-Bearing verses Bearing-only Vision SLAM. The entries in the table are rated as t (true), f (false), $\hat{t}$ (inclined to be true), and $\hat{f}$ (inclined to be false).

2.5 Type of dead-reckoning sensor

A forth and final criterion by which Vision SLAM systems are classified is the type of dead-reckoning sensor that is used. Different environments constrain the navigating vehicle to use different types of ego-motion sensors. In indoor settings, where slippage is rare and the ground is relatively flat, wheel encoders are used. In outdoor aerial and underwater settings, wheel encoders are not possible, and

IMUs are used instead. IMUs have traditionally not been used for land navigation due to their excessive cost; however, with the recent commercialization of low cost strapdown IMUs, such sensors are being considered for land navigation. IMUs are not affected by slippage, as in the case of wheel encoders, but require precise pre-calibration in order to reduce the detrimental effects of IMU sensor biases.

Cameras can also be used for dead-reckoning, akin to the method of Structure from Motion (SfM) [57]. In such systems care must be taken if a Kalman filter is used to merge ego-motion estimates and observations of landmarks. Kalman filters require that the two sources of information that it fuses are statistically independent. If the camera is used to both predict the motion of the vehicle and update its position by detecting landmark, both sources of information are correlated since they both come from the same camera; such situations can lead to failure of the filter if care is not taken to compensate for these effects. One possible remedy is to use different landmarks for motion prediction than those used to build the SLAM map. The comparison between dead-reckoning sensors is shown in Table 2.4.

	Dead-Reckoning Sensor		
	Encoder	IMU	Camera
Slippage Issues	$\hat{t}$	f	f
Expensive	$\hat{f}$	$\hat{t}$	$\hat{f}$
Correlated to Kalman Filter	f	f	t

Table 2.4: Type of dead-reckoning sensor, encoder-based verses IMU-based verses camera-based. The entries in the table are rated as t (true), f (false), $\hat{t}$ (inclined to be true), and $\hat{f}$ (inclined to be false).

2.6 State of the art

In the following sections, the most significant Vision SLAM contributions to date are presented. Systems are grouped based on the camera used (range-bearing or bearing-only) and then into subgroups based on the experimental setting (indoor or outdoor). The critical analysis of each system is focused on highlighting the type of landmark that each system uses. Furthermore, systems are evaluated based on their aptitude to achieve SLAM in an large-scale outdoor setting such as that of VisSLAM. The expected outcome of this section is to show that a new Vision SLAM system based on using natural landmarks is pioneering work that provides an attractive solution to large-scale outdoor Vision SLAM. In this context, the author

does not attempt to highlight the contributions of each of the proposed system; nevertheless, information that is found necessary is included for additional clarity.

2.6.1 Range-bearing indoor

Davison and Murray [37] present the first successful real-time indoor implementation of R-B Vision SLAM using active stereo vision to detect and track naturally occurring features. Landmarks that are used for SLAM are interest regions, detected by finding the principal directions inside patches of the high frequency components of the tested images, akin to the method of Shi and Tomasi [67] but differs in the fact that the tested interest regions are 15 by 15 rather than 5 by 5 pixel regions. Once initialized, these patches are tracked with an active stereo head for several frames to insure that they are stable landmarks. SLAM is executed in the framework of an Extended Kalman Filter. During navigation, a measure of attractiveness for each landmark is calculated based on the amount of information the robot would gain from observing it. The camera is then directed towards landmarks that exhibit the highest attractiveness. Efficient map management decide which landmarks to keep, which to add, and which to prune. Depth is inferred by stereopsis via a stereo camera pair. Experiments are done in an indoor corridor 6 meters long following a straight path and moving forwards and back along this path. Images are processed at 5 frames per second. No information is provided regarding the resolution of the captured images, and to the linear and angular velocities of the navigating vehicle. Davison and Kita [35] build on their previous work [37] to conduct SLAM using vision on uneven terrain. The contribution of this paper resides in conducting SLAM in 3D, where random slope variations of the terrain are allowed, although inclinations higher than 10° are not attempted. The same interest regions as above are used as landmarks and the experimental site is a simulated 2 by 2 meter square indoor area. No information is provided regarding the resolution of the captured images, the frame rate, and the linear and angular velocity of the navigating vehicle. The robustness and validity of this systems requires more extreme terrain variations. These systems can not be applied in an outdoor unstructured setting such as that of VisSLAM, because of the following three reasons. Firstly, the landmarks that are used are not scale invariant, which makes it difficult to recognize them from different depths. Secondly, it is observed that the systems are not tested under significant changes in lighting conditions and viewing directions such as in large outdoor settings. Thirdly, it is expected that the method would suffer from dimensionality issues when applied in large scale settings, where a large number of landmarks have to be tracked.

Se, Lowe, and Little [43] use Scale Invariant Feature Transform (SIFT) local features as landmarks to conduct vision SLAM in an indoor environment. Each SIFT feature has associated with it a vector representing the coordinates, scale, orientation, and disparity (or depth) of the image at the feature location. Dead-reckoning information is used to estimate the robot motion and thereby predict the location of tracked SIFT features in the images of the environment during navigation. Wheel encoders are used for dead-reckoning and a triclops camera arrangement is used to estimate landmark depth. Features are matched according to position, scale, orientation, and disparity. Using the matched SIFT features, a least-squares procedure corrects the camera ego-motion, thereby rectifying the estimate of the 3D coordinates of the SIFT features. SIFT features are screened according to several criteria before using them as landmarks. Appropriate map management is conducted to add, prune, and keep these landmarks as the robot navigates. The effect of viewpoint variance is accounted for by keeping track of the view direction of each landmark. The experimental setting is an indoor office, where the robot travels in a loop approximately 10 meters long on a flat surface. The image resolution is 320 by 240, the linear and angular velocities of the robot are 0.4 m/s and $10^{\circ}/s$ respectively. The loop closure results are $3.9cm$ in position and 2.1° in bearing. SLAM is successfully implemented in this paper using SIFT features, which prove to be stable landmarks which satisfy repeatability (*i.e.*, same landmarks detected from different viewpoints) and data association (*i.e.*, associating a feature in a 2D image to a saved 3D landmark). The main criticism here is the dimensionality issue, one important requirement for SLAM is that the generated map be sparse, allowing tractable map management. The experimental setting the authors use is too small and constrained to permit generalization of its results to large unstructured environments, exhibiting non-uniform lighting conditions. In outdoor environment, Se *et al.*'s method would detect a large number of SIFT features, requiring some form of pruning to avoid the dreaded dimensionality explosion problem of SLAM.

Panzieri, Pascucci, and Ulivi [44] present a vision-based SLAM implementation in an indoor environment using trinocular vision. SLAM is implemented in the framework of an Extended Kalman Filter (EKF). The environmental setting is fairly constrained and consists of an office-like setting, where planar motion is assumed. Artificial light sources located in the ceiling, above the robot, are used as landmarks. Image segmentation, grouping and recognition is performed to recognize and locate landmarks. Since the light sources are at a fixed vertical location, a single camera is used to locate them, using a homography to map the 3D landmark locations to their respective image positions. The experimental setting is a 3 by 4.5 meter area in an indoor office-like setting. The resolution of the images is 352 by 288. No information is provided regarding the frame rate, and the linear and angular

velocity of the navigating vehicle. The loop closure errors are shown graphically but are not quantified. Although the authors show acceptable results for this setting, their system can not be applied in more realistic environments such as that of VisSLAM, which exhibit varying lighting conditions and camera viewpoint as well as detecting natural landmarks that vary substantially in size and shape.

In his thesis dissertation, Jung [38] presents an EKF Vision SLAM system which uses a stereo camera as its only onboard sensor to predict the camera motion and detect landmarks near the camera. Vehicle ego-motion is estimated by matching 3D points between consecutive frames [73, 74]. Landmarks are groups of local interest points, which are invariant to scale, viewpoint, and lighting conditions. Interest Points (IPs) are scale adaptive Harris corners [64, 65] and are grouped based on their physical proximity. Landmarks, represented by IP groups, are matched between images by comparing how similar they are. A similarity metric for IP groups is developed based upon the geometry of IPs inside each group and the similarity of the IPs themselves. Once a reliable group match is established the predicted image transformation are used to focus the search space of candidate group matches, thereby reducing the computational requirements. Data association is successful under changes in camera viewpoint and lighting conditions. Since prediction and observation are based on the same sensor (*i.e.*, the stereo camera) special care is taken to avoid violating the independence condition of the Kalman filter. This is achieved by minimizing calibration errors of the stereo system and by selecting IP for dead-reckoning that are different than those for landmarks. The main concern here is the extreme dependence of the SLAM system on the stereo rig. If the relative position of one camera is accidently displaced with respect to the other, which is highly probable in outdoor settings, the Kalman Filter would fail. This method solves the problem of dimensionality explosion, which is so common in IP based landmarks, by grouping IPs together and using these groups as landmarks rather than the IPs themselves. As a result the number of tracked landmarks is reduced to a manageable number. Jung implements his system in three settings: indoor, outdoor off-road, and outdoor aerial. The experimental setting is a 3.5 by 3.5 square indoor office setting. Image resolution is 512 by 384. The frame rate is $1.1 fps$. No information is provided regarding the linear and angular velocity of the navigating vehicle. This system produces very good results for all three settings; however, its major shortcoming is the requirement for a relatively large baseline between the cameras of the stereo rig. Figure 2.2 displays the stereo geometry, where depth of a 3D point P is calculated based on the disparity in coordinates of the images I_L and I_R of point P on the image planes of the left and right cameras respectively. The precision of the three dimensional points which are determined from image points is calculated according to the following equations [75]:

$$z = \frac{b \cdot f}{p} \tag{2.3}$$

where z is the depth, b is the camera baseline, f is the camera focal length in pixels, and p is the parallax or disparity. Differentiating this equation yields

$$zdp + pdz = fdb + bdf \tag{2.4}$$

The relation for the relative standard deviation squared is then expressed as [75]:

$$(\frac{\sigma_z}{z})^2 = (\frac{\sigma_b}{b})^2 + (\frac{\sigma_f}{f})^2 + (\frac{\sigma_p}{p})^2 \tag{2.5}$$

where σ stands for standard deviation and its inverse is the precision. In the case of perfect calibration and orientation, σ_b and σ_f can be ignored and the relative standard deviation becomes

$$\frac{\sigma_z}{z} = \frac{\sigma_p}{p} \tag{2.6}$$

Subbing $p = \frac{f \cdot b}{z}$ into (2.6) yields

$$\frac{\sigma_z}{z} = \frac{z}{f}\frac{\sigma_p}{b} \Longrightarrow Precision = \frac{1}{\sigma_z} = \frac{f \cdot b}{z^2 \cdot \sigma_p} \tag{2.7}$$

Equation (2.7) states that, given a calibrated stereo rig with fixed focal length and baseline, the precision of three dimensional points obtained via stereopsis is inversely proportional to the depth squared, and the standard deviation of the disparity values which is directly related to the precision of the camera. Therefore, one can obtain more precise depth values by using cameras with higher precision (*i.e.*, lower σ_z) but comes at the cost of more expensive hardware and high processing requirements (*i.e.*, larger images). Empirical tests have shown, that for a camera with a resolution of 1024x768, baseline to depth ratios less than 1/30 return inconsistent depth values [38]; which implies that off-the-shelf stereo cameras with similar resolution, exhibiting a baseline of the order of $10 - 15cm$ return imprecise depth estimates at distances larger than 4.5 meters. As a result, stereo rigs used in long range applications are designed with large baselines, where the cameras are placed far away from each other. When space requirements are an issue, such configurations are not possible and one has to resort to using a single camera with bearing-only technology.

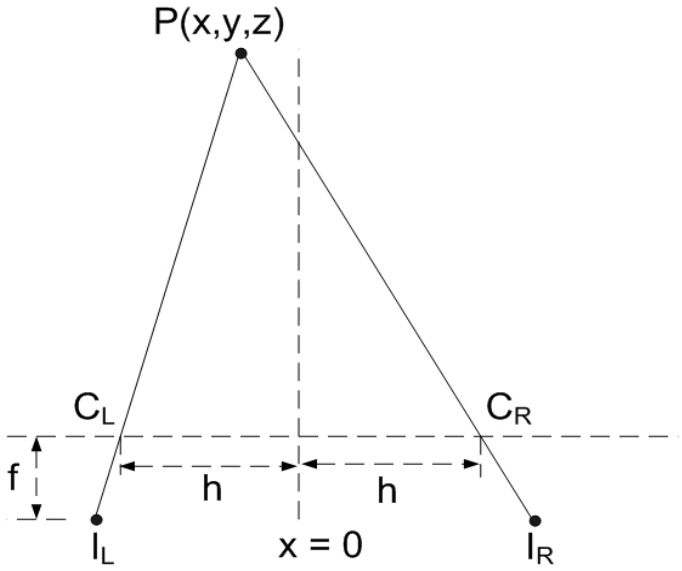


Figure 2.2: Stereo geometry. I_L and I_R are the images of the point P in the left (C_L) and right (C_R) cameras respectively, f is the focal length of the cameras and $2h$ is the baseline between the two cameras.

2.6.2 Range-bearing outdoor

Fitzgibbons [45] performs both R-B SLAM and B-only SLAM in an outdoor flat open area using bright white parking dots as landmarks. SLAM is performed on a utility vehicle, equipped with wheel encoders to predict the vehicle ego motion. In the R-B SLAM system, the vehicle is fitted with a stereo camera which reports the range and bearing of landmarks. The landmarks are detected by thresholding images of the scene, and selecting regions that are brightest in these images. The experimental test site is 60 by 60 meter outdoor flat area. No information is provided regarding the frame rate, and the linear and angular velocity of the navigating vehicle. This system could not be applied in the VisSLAM setting for two reasons. First, no artificial beacons exist in the VisSLAM test site and second, the wheel encoders would probably suffer from large errors due to slippage in the outdoor setting. Fitzgibbons' B-only implementation is described below along with other B-only systems in Section .

Kim and Sukkarieh [31] demonstrate a Vision SLAM system on an unmanned aerial vehicle using an inertial sensor for dead-reckoning prediction and a monocular camera for landmark detection. Landmarks are artificial beacons with known shapes and sizes, whose 3D positions are determined based on their projected size

in images collected during navigation of the test site. The experimental setting is a 1200 by 600 meter outdoor area. The aerial vehicle travels at 40 m/s and is capable of $20°/s$ rotation rates. The processing rate is $50fps$. The main contribution of this work is the proof that Visual information is sufficient to bound inertial drift. The inherent problem of this system is that it relies on artificial beacons as landmarks, the cost of preparing such an infrastructure would be intractable for large scale settings.

Jung [38] presents two successful outdoor R-B SLAM implementations, the first is aerial-based and the second in land-based. The resolution of the images in both cases is 512 by 384. The aerial test site is 100 by 100 meters and the land-based site is 16 by 12 meters. The loop closure results for the aerial set up is $212cm$ in position and 3.56° in bearing. The loop closure results for the land-based experiment is $6.5cm$ in position and 1.38° in bearing. The frame rate is $1.1fps$. No information is provided regarding the linear and angular velocity of the navigating vehicle.

In stereo rigs, care must be taken to prevent any relative motion between the cameras during navigation to avert the need for re-calibrating the stereo rig. Clearly, an algorithm capable of using a single camera to locate landmarks is useful. However, in such systems depth information is lost and an alternate method is used to infer the 3D position of landmarks. In the next section, the various contributions of B-only SLAM are discussed.

2.6.3 Bearing-only indoor

Davison [36] performs ego-motion estimation in real-time using a single camera, using a separate Particle Filter to estimate distance. Once a feature is measured for the first time, a 3D line 'L' is initialized into the map from the estimated camera position, and heading to infinity along the direction of the feature. Each feature is then represented by a vector containing the origin of 'L' and its unit direction vector in the world coordinate frame. Then, for each feature, a number of depth hypotheses is suggested along their respective directions. Davison suggests using 100 depth hypotheses for each feature. As the robot moves to another position, these hypotheses are tested by projecting them into the image. Each hypothesis consists of an elliptical search region. Features that match within each ellipse produce a likelihood for each which are used to adjust the weight of each hypothesis and used to update is probability of occurrence. As the robot navigates, each feature's 3D position hypothesis is updated until its PDF peaks at a consistent depth value. At that time, the feature is initialized as a landmark in the SLAM map. Landmarks are local interest regions detected as in the Davison's previous work [35,

37]. Loop closure is not performed here. This SLAM implementation is performed indoors on a desktop setting, where lighting is almost uniform and the scale of environment is small; implementing it in an outdoor environment such as ours, does not seem feasible because of issues related to variable lighting conditions and poor data association (landmark variance under scale). Furthermore, The number of landmarks needed for an outdoor SLAM implementation for depth detection might increase prohibitively in large-scale outdoor settings.

Kwok and Dissanayake [39] implement B-only SLAM on a mobile robot in an indoor unstructured environment. Their SLAM filter is implemented within the framework of a particle filter; fusing dead-reckoning data from wheel-encoders with landmark positions estimated via a monocular camera. The vision system uses edge features as landmarks. Edge features are not invariant from the standpoint of scale, viewpoint, and lighting conditions. Data association is achieved via a Nearest Neighbor approach, by determining the difference between real-life and expected measurements and thresholding the result against some confidence level in order to determine the most likely match. If the measurement is below a given threshold, a landmark is declared matched. Landmarks are ranked according to their edge height and image quality and only the top 15 landmarks are preserved in order to avoid dimensionality issues. The robot is steered in a circular path to insure maximum convergence due to the low Field of View (FOV) of the onboard camera. A better approach is to mount the camera on a Pan Tilt Unit (PTU) or to use an omnidirectional camera. The robot is steered in a circular path 2 meters in diameter. The experimental setting is too small to ascertain the effectiveness of the method. The system promotes the particle filter SLAM approach which is a more robust vision module for non-linear cases.

Kwok and Dissanayake [40] present an undelayed initialization B-only SLAM, where the initial state is approximated as a sum of Gaussians [76, 77] and is added to the state vector of the Kalman filter. Multiple depth hypothesis are postulated for each landmark along its line of sight at the time it is first observed. Each of the hypotheses is integrated into the filter and treated as a separate feature. Bad hypotheses are subsequently pruned as the vehicle moves around the feature and only persistent hypotheses are kept. Once a depth hypothesis for a landmark is validated from several sightings, the landmark is initialized into the SLAM map. This approach is computationally efficient but suffers from poor data association because information from observations made before initialization is discarded. The experimental setting is the same as in their previous work [39]. No information is reported to the frame rate, linear and angular velocities of the vehicle. The loop closure performance is shown graphically but is not quantified in numbers.

Kwok, Dissanayake, and Ha [41] extend their previous work [40] in devising an efficient initialization procedure for landmarks. Each depth hypothesis has associated with it an EKF which operates independently and its performance is evaluated from the likelihood values calculated based on the innovations. A Sequential Ratio Probability Test (SPRT) [78, 79] is used to decide whether to keep the EKF or remove it from the EKF bank. The SPRT is a decision-making technique in which the ratio of innovations from EKFs is used as a decision metric. SPRT allows for the choice of a delayed decision in addition to acceptance or rejection decisions. Each time a new landmark is observed the EKF bank is re-initialized and the decision process repeats. The authors present results in a simulated setting, as well as an indoor setting on an Pioneer robot using a Laser Range Finder (LRF) to determine bearing-only values to the landmarks. The experimental setting is an office like environment as in their previous work [39, 40]. Loop closure results are shown graphically but are not quantified. Although the results are promising, it remains to investigate this method in a larger unstructured settings, using a camera to obtain bearing information.

Kwok, Dissanayake, and Ha [41] extend their previous work [40] in devising an efficient initialization procedure for landmarks. Each depth hypothesis has associated with it an EKF which operates independently and its performance is evaluated from the likelihood values calculated based on the innovations. A Sequential Ratio Probability Test (SPRT) [78, 79] is used to decide whether to keep the EKF or remove it from the EKF bank. The SPRT is a decision-making technique in which the ratio of innovations from EKFs is used as a decision metric. The authors present results in a simulated setting, as well as an indoor setting on an Pioneer robot using a Laser Range Finder (LRF) to determine bearing-only values to the landmarks.

Deans and Hebert [49] demonstrate a Visual B-only SLAM system. In their work, landmarks are initialized based on a modified bundle adjustment system. Bundle adjustment consists of storing all the observations and respective robot poses and then correct the robot pose and landmark positions by performing a batch update with all the stored observations. The advantage of this system is that the estimates are well conditioned, the disadvantage is that the computational complexity of the algorithm scales with the number of observations, make it intractable for real-time applications.

2.6.4 Bearing-only outdoor

Lemaire, Lacroix, and Solà [42] introduce their implementation of a B-only SLAM system which is based on an EKF. Once a landmark is first observed, its 3D location is modeled as a sum of Gaussians. This landmark is then updated during subsequent observation until the landmark can be initialized into the SLAM map. The initial state of the robot is expressed in the robot frame until it is initialized into the SLAM map and expressed in the global frame. Many features can be added to the initial map at low computational cost and then the best feature is selected. All landmark observations and their corresponding robot pose at the time of observation are saved and used to update the map in a batch process once the landmark is deemed stable. The experimental tests are done in a simulated environment and landmarks are corners detected via a Harris corner detector. The indoor test is run on a circular path 3 meters in diameter. The outdoor test is done on an aerial blimp along a straight path 120 meters in length. For the indoor setting loop closure performance is not reported and the outdoor loop closure is not performed.

Bailey [51] presents a B-only SLAM system implemented within an EKF framework. Landmarks are initialized into the SLAM map in a delayed procedure known as *constrained initialization* [80], where past vehicle poses are stacked into the SLAM state-vector, together with associated measures of landmark bearing until the baseline is sufficient to enable Gaussian initialization. At that time, the saved pose and landmark bearing measurements are used to correct the entire map. The condition for well-conditioned situations relies on relative entropy between the analytical PDF of the feature location and its linearized Gaussian approximation, where a close match implies a near-linear transformation from measurement-space to feature-space. The primary concern in this method is the validity of the entropy threshold methodology. Bailey states that the optimum threshold is not constant; rather, it is dependent on the degree of uncertainty of the analytical PDF. Another issue is the computational burden inherent in a numerical Monte Carlo solution such as this one. Finally, the observations that were discarded between the first and final reading contain information that might add robustness, precision, and accuracy to the heuristic depth estimating system. The experiments are run in a simulated 100 by 100 meter outdoor flat area and the landmarks are point features. Loop closure information is not quantified but the estimated error of one of the vehicle's coordinates is plotted verses time.

Bryson and Sukkarieh [34] implement their B-only SLAM on an Unmanned Aerial Vehicle (UAV) over unstructured, natural environments. They implement SLAM within an EKF environment, where IMU dead-reckoning information is fused with camera bearing information. Delayed initialization is used to fix the 3D posi-

tion of landmarks from several bearing readings obtained from different robot poses. The bearing of each landmark and the corresponding robot pose are stacked into a state vector and their corresponding covariances into a dynamic covariance matrix. In other words, the correlations between the saved states and observations, and the current vehicle pose are maintained until ill conditions subside. At that time, the accumulated state and covariance matrices are used to initialize the landmark and update the entire SLAM map. The authors use a angle threshold of 40 degrees between observations to decide when conditions are satisfactory to initialize a landmark. Data association is achieved via statistical inference by hypothesizing several feature 3D positions along a line of sight. The mean and covariance of each hypothesis is calculated in incremental depths from a minimum to a maximum range. The multi hypothesis distribution is maintained separately from the state vector and is used only to assist in the data association of un-initialized features. The Malhalanobis distance between features and landmarks is used to predict the optimal match. The vision system finds the normalized intensity of each pixel in the image, applies an intensity threshold and finds which pixels lie above the threshold. Groups of interconnected pixels whose pixel count and dimensions fall within given bounds, are treated as image saliencies and initialized as landmarks into the SLAM map. The system is implemented in an off-line scenario using logged vision and IMU data and real-time implementation remains to be evaluated.

The bearing-only systems detailed so far suffer from two major drawbacks. Firstly, they all rely on a criterion to decide when ill-conditions subside, and it is safe to initialize a landmark. Good generic criteria are not available yet and navigation constraints must be enforced to keep the wait period with reasonable limits. Secondly, a wait period is necessary until the criteria of initialization are met. It is clear from this discussion that it would be beneficial that alternate methods which avoid the initialization criterion and transient period would be beneficial. Such methods are known as undelayed-initialization and are discussed next.

In the work of Fitzgibbons [45] the B-only system is preceded by an analysis of the different methodologies for landmark initialization in B-only SLAM; namely, triangulation-based and particle filter based initialization. In his analysis, the author concludes that the former method (*i.e.*, triangulation) yields more robust solutions than the latter (*i.e.*, Particle filter). B-only SLAM is then performed in the same outdoor area using the same utility vehicle as in the R-B case; however, landmarks are salient colored poles rather than parking dots and the system uses a panoramic camera to detect the relative bearing of landmarks to the vehicle. In both R-B and B-only implementations, the author mentions the reduction of uncertainty of the systems at the end of each run but no mention is given to loop

closure capabilities and run-time of the systems.

Solà *et al.* [50] implement an undelayed initialization B-only SLAM that can be run in real-time using a bearing-only sensor with a narrow Field of View (FOV). The system works by generating a multi-hypothesis Gaussian map that includes the entire ray that represents the PDF of the landmark's position. Prior to initializing a feature as a landmark, observations of this feature are transferred to each hypothesis using concepts acquired from the Federated Filter (FF) [81]. However, in using such an approach there is no guarantee that estimates are consistent due to the updating of hypotheses that might not exist. The experimental setting is a simulated one and no mention is given to the type of landmarks that are used.

Table 2.5 encapsulates the foregoing literature review in tabular form. Two shortcomings are immediately obvious: 1) none of the systems uses natural objects as landmarks and 2) none of the systems uses an IMU for land-based SLAM. Furthermore, none of the systems except Jung [38] constitutes a serious candidate for large-scale outdoor SLAM. However, Jung's system relies on a stereo rig featuring a long baseline between cameras. Such a system can not be used if physical constraints on board the mobile platform are an issue. Moreover, it is interesting to develop a system that can be implemented on any handheld camera, rather than having to purchase an expensive stereo rig or having to set up and calibrate your own stereo system.

2.7 Context of this thesis

A Vision SLAM system named VisSLAM is proposed that aims to use real objects as landmarks in an outdoor environment. Tree trunks are used for testing the system, although the developed techniques lend themselves to other situational relevant landmarks, possessing known structure, color and texture. It is the author's belief that environment recognition can offer valuable information that can aid in the detection of sought landmarks for SLAM. If we submit to the idea of using physical objects as landmarks, environment recognition can help decide with high probability what landmarks are available in the current setting to confine the landmark search space to landmarks that are characteristic of that setting. The long term goal is to develop a SLAM algorithm capable of operating across multiple environments with good loop closure.

		Authors	Landmark			Dead-Reck.			Objective		
			I.P.	O.A.	O.N.	W. E.	IMU	C	Feas.	Init.	Sparse
Range-Bearing	Indoor	Davison [37]	x			x			x		
		Davison [35]	x			x			x		
		Se [43]	x					x	x		
		Panzieri [44]		x		x			x		
		Jung [38]	x			x			x		
	Outdoor	Jung [38]	x				A		x		
		Kim [31]		x			A		x		
		Jung [38]	x			x			x		
		Fitzgibbons [45]		x		x			x		
Bearing-only	Indoor	Williams [80]		x						D	
		Davison [36]	x					x		U	
		Kwok [39]		x		x				U	
		Kwok [40]		x		x				U	
		Kwok [41]		x		x				U	
		Solà [50]		s						U	
		Lemaire [42]		x						D	
	Outdoor	Lemaire [42]		s			A			D	
		Bryson [34]		x	x		A			D	
		Deans [49]		s		x				D	
		Deans [49]		x		x				D	
		Bailey [51]		s		x				D	
		Solà [50]		s		x				D	
		Fitzgibbons [45]		x		x				D	
		Eustice [72]		x			W		x		x

Table 2.5: Comparison of Vision SLAM systems. Systems are grouped into Range-Bearing and Bearing-only systems and then into subgroups based on the setting (Indoor or Outdoor). Each of the systems is then classified based on three criteria: (1) the landmark type is either IP-based (I.P.), Object Artificial (O.A.) or Object Natural (O.N); (2) the dead-reckoning sensor is either a Wheel Encoder(W.E.), an IMU, or a Camera (C); (3) the objective of each system is either Feasibility (Feas.), Landmark Initialization (Init.), or Sparseness (Sparse). The landmark initialization is either Delayed(D) or Undelayed(U). The IMU is applied in either an aerial (A) or underWater(W) setting.

2.8 Summary

In this chapter a critical analysis is presented of the state of the art in the field of Vision SLAM. There are two main research streams in Vision SLAM: range-bearing and bearing-only. Range-bearing SLAM systems use binocular or trinocular camera rigs to determine the range and bearing of landmarks in one sighting. Although efficient, they are limited to vehicles which can accommodate relatively large camera baselines.

Bearing-only systems use a single camera to report the bearing of a landmark from several sightings and subsequently calculate depth based on the bearing readings and vehicle motion estimate between sightings. Bearing-only systems provide flexibility to the vision system but suffer from high computational requirements or real-time issues due to delay times inherent in initializing a landmark.

Landmarks used for Vision SLAM systems are either based on image saliencies of artificial landmarks. While the former type of landmarks can suffer from dimensionality issues, the latter type suffers from the inherent overhead of setting up the SLAM test site. Although difficult to detect, there is a definite advantage in using the environment's natural features as landmarks.

A new Vision SLAM system named VisSLAM is proposed, which begins by recognizing the context of the surrounding environment in order to develop a prior probability on the type of landmarks to use. Once the context is established, the SLAM system selects natural features that are commonly found in such settings. VisSLAM is implemented on tree trunks inside a park environment. VisSLAM detects, recognizes, and localizes tree trunks using a Computer Vision system detailed in Chapter 4. VisSLAM predicts the ego-motion of a mobile platform using a Inertial Navigation System (INS) which is introduced in the next Chapter.

Chapter 3

Land-based inertial navigation system

3.1 Introduction

The Inertial Navigation System (INS) constitutes the first building block of VisSLAM. Inertial navigation is a form of dead-reckoning which predicts the translation and rotation of a platform by processing readings of linear accelerations and angular rotations obtained via an Inertial Measurement Unit (IMU) fixed to this platform. IMUs possess many important advantages for navigation; they are insensitive to vehicle slip, they are jam proof, and do not radiate anything. IMUs have been traditionally used by the military (*e.g.*, aircrafts, submarines, missiles) and not used elsewhere due to their prohibitive cost; however, with the recent growing interest in using IMUs in the automotive industry [82] for applications such as vehicle dynamic control, rollover, airbag deployment, navigation, and chassis control, small sized, low-cost IMUs are being developed [82, 83, 84, 85] that are not as precise as their military counterparts but are good enough for robot navigation applications.

IMUs are especially appealing to aerial and underwater robotics, where dead-reckoning via wheel encoders is not possible. In outdoor land-based navigation, IMUs are also attractive primarily because they are not affected by slippage, which is a major source of error in wheel encoders. In fact, there has been recent incorporation of IMUs on board some commercial wheeled robots [53] to compensate for dead-reckoning errors incurred by the robot in its wheel encoders. Other benefits of developing land-based INS, is the application of robotics navigation technologies to non-wheeled scenarios such as humanoid robotics or as an aid for the visually impaired.

Although, it is possible to purchase relatively cheap, out-of-the-box IMUs that are self-contained, small, and precise [86], these devices are not plug-&-play and special care must be taken to calibrate them, and to set up appropriate coordinate systems before using them. Unfortunately, regardless of the calibration that is done, the best one can achieve is to reduce dead-reckoning errors and not completely eliminate them, for these errors will grow without bound unless the IMU readings are supplemented by some external absolute positioning system such as GPS or they are incorporated into a SLAM framework.

The contributions of the proposed INS system to the inertial navigation community are threefold. First, to present a clear and detailed methodology for designing an INS based on strapdown IMUs. Although there are a myriad of papers in the literature regarding INS design, very little information is provided regarding those based on low cost IMUs. Second, this chapter addresses IMU bias effects and evaluates an approximate calibration method by comparing its results to those of an exact method. Third, the effect of non-holonomic constraints on an INS are analyzed. Specifically, two recently proposed theories for imposing such constraints are implemented and evaluated. The first theory is contested because a shortcoming is discovered during its implementation. The second theory, based on enforcing constraints via an EKF is found effective and is adopted for the VisSLAM INS.

The remainder of this chapter is structured as follows. First, Section 3.2 introduces IMUs and discusses the two types available on the market, along with their advantages and disadvantages. Section 3.3 discusses the coordinate frames that are used in the VisSLAM INS. Section 3.4 discusses the biases that are present in IMUs and presents two methods to account for them. Section 3.5 discusses the *nonholonomic* constraints, which can be imposed on land-based INS when no lateral skidding or slipping occurs. Section 3.6 presents the architecture of the VisSLAM INS system. In Section 3.7 several experiments are performed to investigate the INS dead-reckoning capabilities, where shortcomings in inertial-based navigation promotes the idea of an INS-SLAM navigation system.

3.2 Inertial measurement units

The background information for IMUs is taken from the technical report of Nebot [87]. Inertial sensors include accelerometers and gyros. Accelerometers report the linear acceleration of a body in one direction by sensing the value of the inertial force applied to this body. Gyroscopes determine the rotational velocity of a body in one direction and are either of the vibratory type or the fiber optic type. Vibratory

gyros calculate angular velocity based on the measured coriolis [1] accelerations; fiber optic gyros calculate angular velocity based on the Sagnac [2] effect.

IMUs are constructed by assembling three mutually orthogonal accelerometers and three mutually orthogonal gyros in one of two configurations: *gimballed* or *strapdown*. The traditional and most precise IMUs are based on the gimballed configuration (Figure 3.2), where the platform that carries the accelerometers and gyros is actively controlled on three mutually orthogonal gimbals, such that the platform maintains a constant orientation (*i.e.*, pitch, roll, and yaw). Motion of the vehicle in each direction (x, y, and z) is then obtained by simply integrating the accelerometer reading in the corresponding directions (a_x, a_y, and a_z). Although precise, these systems are extremely expensive running in the order of tens of thousands of U.S. dollars [83], whereas strapdown IMUs can be purchased at around two thousand U.S. dollars [86, 84].

In strapdown IMUs, the accelerometers and gyros are fixed to a common frame whose orientation changes with that of the vehicle. Many manufactures of strapdown IMUs use Micro Electro Mechanical Systems (MEMS) parts to build very small systems such as the one shown in Figure 3.1 (Crista IMU by Cloud Cap [86]). The disadvantage of strapdown systems is the relative complexity of the underlying algorithms used to transform acceleration and rotation rates to position and bearing and for the necessity to account for biases and gravity effects that must be removed from the sensor readings before integration. In strapdown IMUs, readings are indicative of accelerations and rotation rates in a frame attached to the body (body frame). These values must be expressed in an inertial frame before they are integrated. The transformations from body to inertial frame are detailed in section 3.3. True acceleration of the vehicle must be separated from external disturbances including gravity effects, coriolis forces incurred from the rotation of Earth, and biases (discussed in Section 3.4). In practice, it is impossible to completely remove all IMU errors, which grow without bound if the system is not augmented by some external source of information.

[1] The coriolis acceleration is apparent at a point which is translating on a body which is itself in rotation.

[2] The angular velocity is proportional to the phase shift in two waves of light circulating in opposite direction around a given path.

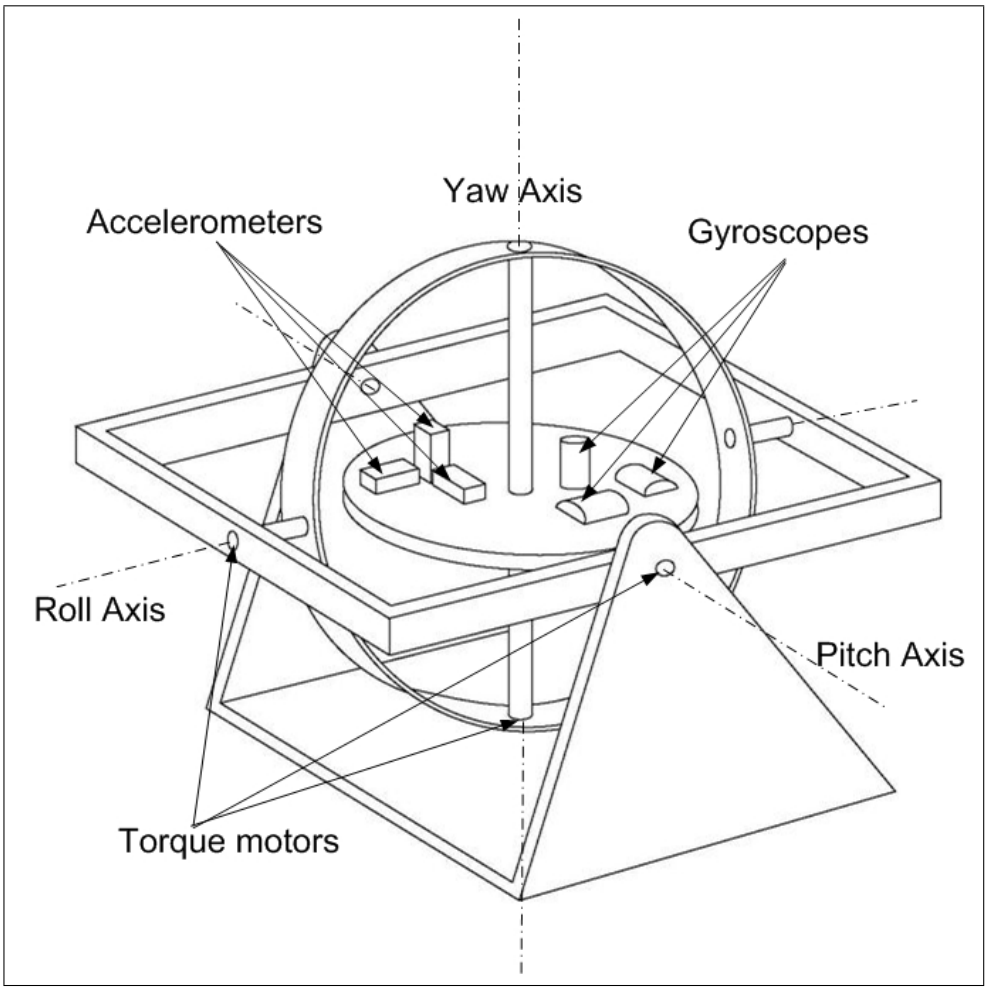


Figure 3.2: Gimballed IMU. During navigation the gyroscopes sense any angular motion in pitch, roll, and yaw, which subsequently activates the relevant torque motor to maintain the platform at a constant angular orientation. Motion is then predicted by a straight integration of the accelerometer readings.

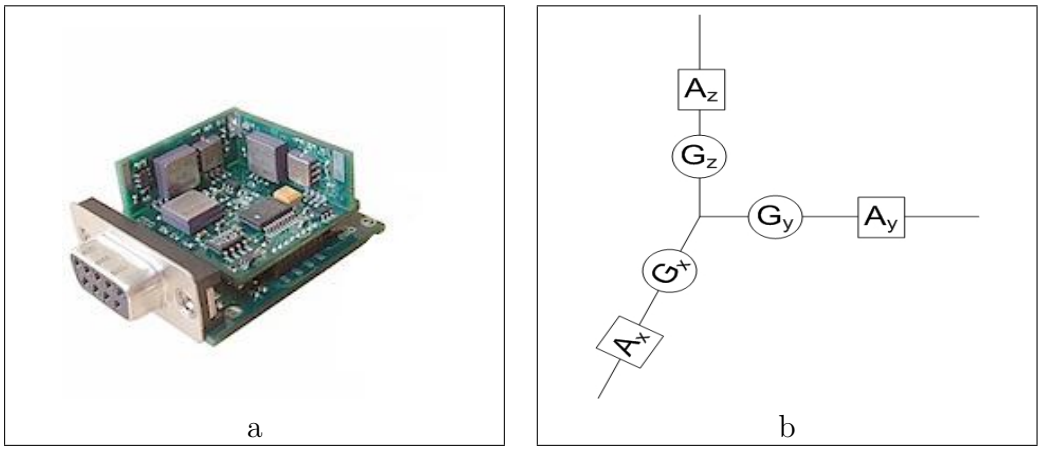


Figure 3.1: (a) Strapdown IMU sold by Cloud Cap [86], (b) sketch of a strapdown IMU.

3.3 Coordinate frames

Figure 3.3 shows the coordinate frames used in the INS, including navigation (N) frame, which is a North, East, Down (NED) coordinate frame and the body (B) frame.

During navigation, transformation from body to Inertial frame is performed following the Z-Y-X Euler convention, which starts with the body frame coincident with the navigation frame, followed by a rotation of the body frame about the $\hat{Z}_b$ by an angle ψ, followed by a rotation θ about the current y axis $\hat{Y}_B$, followed by a rotation of an angle ϕ about the current x axis $\hat{X}_B$. These rotations must occur in the above order, since rotations are not commutative. The resulting rotation from body to navigation frame would then be calculated as:

$$C_b^n = R_z(\psi)R_y(\theta)R_x(\phi) = \begin{bmatrix} c\phi & -s\phi & 0 \\ s\phi & c\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\psi & -s\psi \\ 0 & s\psi & c\psi \end{bmatrix} \tag{3.1}$$

$$C_b^n = \begin{pmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{pmatrix} \tag{3.2}$$

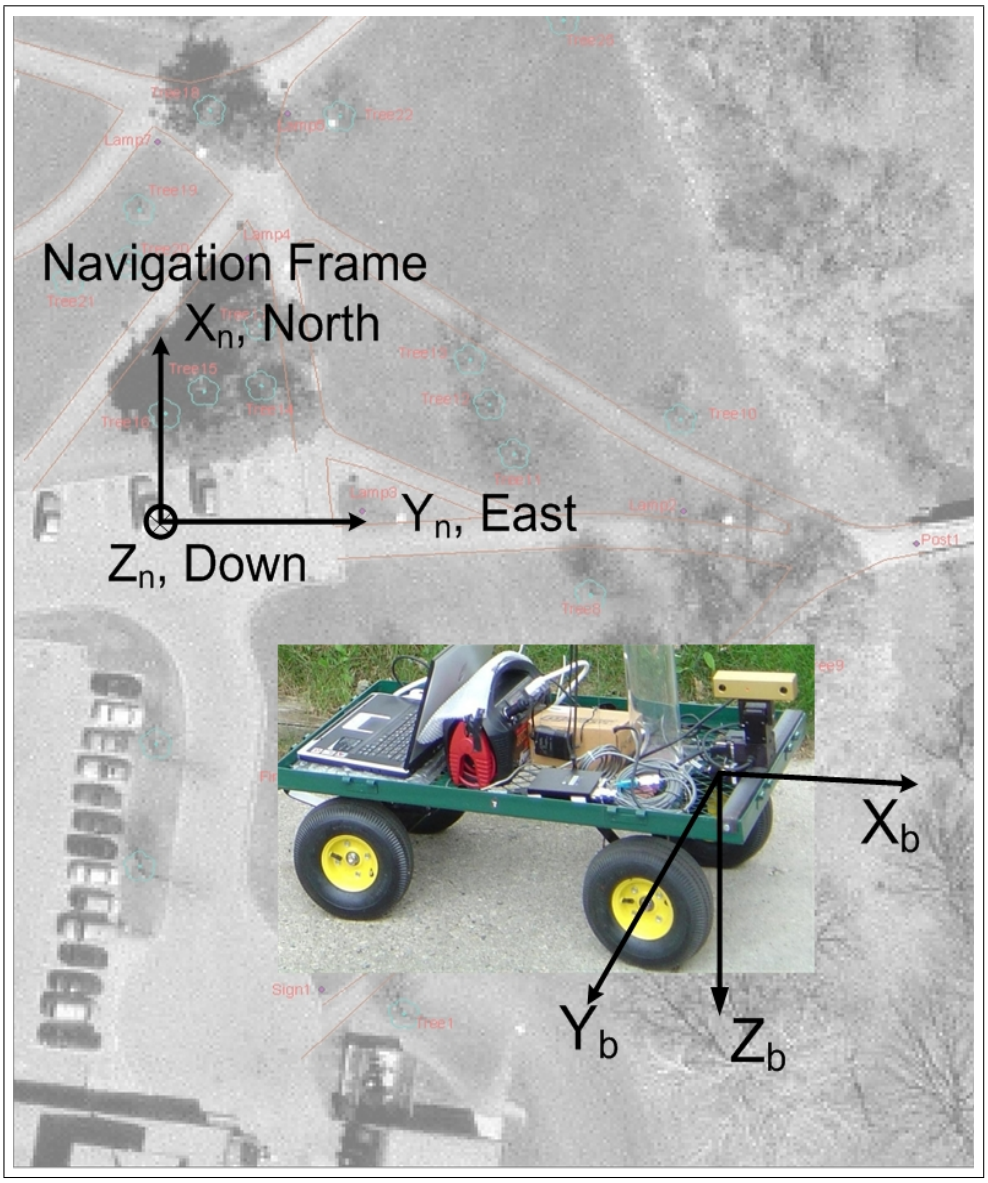


Figure 3.3: Coordinate frames used for INS dead-reckoning. Subscripts B and N stand for the body and navigation frames respectively.

where c and s are the *cos* and *sin* of the Euler angles. Transforming a vector (*e.g.*, A_{Bx}) from the body frame to the navigation frame is done by multiplying A_{Bx} by the direction cosine

$$A_{nx} = C_b^n A_{Bx}. \tag{3.3}$$

It is apparent that the Euler angles must be tracked throughout the navigation process in order to allow for the accelerometer readings to be expressed in the navigation frame at each iteration. The Euler angles at any time t are calculated by integrating the respective gyro readings from the previous time step $t-1$ until t and adding the result of each of the three integrals to the Euler angles at time $t-1$. Numerically, this amounts to

$$\begin{bmatrix} \phi_t \\ \theta_t \\ \psi_t \end{bmatrix} = \begin{bmatrix} \phi_{t-1} \\ \theta_{t-1} \\ \psi_{t-1} \end{bmatrix} + E_B^N \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} dt \tag{3.4}$$

where E_B^N is the matrix that transforms gyroscope readings into Euler angles

$$E_b^n = \begin{bmatrix} 1 & s\phi \tan\theta & c\phi \tan\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi sec\theta & c\phi sec\theta \end{bmatrix} \tag{3.5}$$

One issue that warrants attention is the determination of the initial Euler angles. One method to calculate them is to include two pendulum gyros on board the navigating vehicle to determine the bank and elevation of the unit at standstill. Alternatively, if such pendulum gyros are not available, one can estimate these initial Euler angles by correlating the accelerometer readings at standstill to the known gravitational forces.

$$\theta_0 = \arcsin(a_x/g) \tag{3.6a}$$

$$\phi_0 = \arcsin(-a_y/(g * \cos(\theta))), \tag{3.6b}$$

where a_x and a_y are the average accelerometer readings (calculated by averaging accelerometer readings at standstill) in the x and y directions respectively. The remaining Euler angle is the heading of the vehicle at startup, which can be estimated via a compass, via a gyro with enough sensitivity to determine the rotation of the Earth, or as is done in this work via two consecutive GPS readings at startup,

$$\psi_0 = \arctan(\frac{GPS_y(t_2) - GPS_y(t_1)}{GPS_x(t_2) - GPS_x(t_1)}), \tag{3.7}$$

where ψ_0 is the heading at t_0, GPS_y and GPS_x are the latitude and longitude readings of the GPS respectively, t_1 and t_2 are two consecutive time stamps from the point the vehicle initiates navigation. Determining the correct t_1 is important, for if it is set to a pre-startup time stamp, t_1 and t_2 would mistakenly represent random noise in the GPS unit; using the corresponding GPS values at t_1 and t_2 would therefore result in an erroneous initial heading of the unit. If on the other hand, t_1 and t_2 are recorded some time after the correct startup, and the vehicle moves in a curved trajectory, an erroneous initial heading is recorded. The solution to this issue is to choose t_1 closest to the time the first image is taken; which is when navigation initiates.

Once the Euler angles are calculated, the new direction cosine matrix C_b^n is updated by subbing in the new Euler angles into 5.7. The updated C_b^n is then used to transform the accelerometer readings from the current body frame to the navigation frame and the velocity and position of the vehicle are calculated as

$$V_{new} = A_N dt + V_{old}, \tag{3.8}$$

and

$$X_{new} = \frac{1}{2} A_N dt^2 + V_{old} dt + X_{old} \tag{3.9}$$

The equations presented above in (3.6a) and (3.6b) assume that the accelerometer readings a_x and a_y include no bias and therefore report true values. In reality, these readings are corrupted by errors, called biases that must be accounted for before any integration occurs.

3.4 IMU biases

Linear and angular displacements of an IMU are measured by integrating accelerometer and gyroscopic readings respectively. Biases in the accelerometers and gyros can introduce artifacts that have devastating effects on the integration process. For example, if we consider a bias in the angular velocity reading w_z, it introduces false acceleration components in a_x and a_y which are proportional to the navigation time cubed (t_{nav}^3) [88]. Biases in the gyros are modeled as

$$\dot{\theta}_{meas} = \dot{\theta}_{true} + b + \mu, \tag{3.10}$$

where $\dot{\theta}_{meas}$ is the measured gyroscopic rate, $\dot{\theta}_{true}$ is the true rate, b is the gyro bias (assumed independent of time), and μ is the noise in the measurement. Integrating

(3.10) yields

$$\theta_{meas} = \theta_{true} + bt + \int \mu dt, \tag{3.11}$$

where θ_{meas} is the integral of the measured gyroscopic rate, θ_{true} is the true angular rotation, bt is the angular bias offset which is a linear function of time, and $\int \mu dt$ is known as *random walk* (*RW*), which is proportional to the standard deviation of gyro noise σ_μ and square root of navigation time t, expressed as

$$RW = k\sigma_\mu \sqrt{t}. \tag{3.12}$$

The biases that are present in accelerometers and gyroscopes include temperature-dependent bias, turn-on to turn-on bias, cross-axis or inline bias, and scale bias. Gyroscopes further include acceleration-dependent biases, which cause the gyroscope bias to shift based on the sensed accelerations. The exact value of these biases is different from sensor to sensor and depend on manufacturing imperfections. For this reason, IMU manufacturers usually report only nominal bias values as part of their product specifications (Appendix A.1). In order to determine more precise and unit-specific biases, some manufacturers calibrate each of their units individually and provide the corresponding calibration file to the customer. This file contains information regarding cross-axis acceleration and gyroscope biases, as well as a matrix to deal with gyroscope acceleration bias.

Calibration of an IMU is performed by placing the unit in a controlled environment, varying the parameter of interest and measuring the corresponding bias. There is a strong correlation between temperature and sensor bias; the better the quality of the sensor the smaller the variability of the bias with temperature and the more linear the variation of the bias in the temperature range. Temperature-dependent bias is estimated by keeping the sensor stationary and measuring the readings of the sensors while varying the temperature. Sensors are also affected by a hysteresis effect; the bias determined during one temperature range and during another is different. This is called turn-on to turn-on bias and is measured by conducting several batch bias calculations and comparing their output. Inline, or cross-axis biases are caused by internal misalignment of the sensor input axis. Inline biases of gyros are estimated by placing the gyro on a rate table such that its axis is perpendicular to that of the rate table. Any measurement that is subsequently reported by the gyro is due to misalignment. Once these biases are estimated, the scale biases of the gyros are estimated by aligning the gyro axis with that of the rate table and comparing gyro readings to those of the input. Acceleration-dependent gyro biases are estimated by placing the gyro inside a centrifuge machine and com-

paring the reported gyro readings (minus the previous biases) and the inputs at different rotation rates (*i.e.*, centrifugal forces are proportional to the square of the angular velocity). Accelerometer biases are determined in a similar fashion as that described for gyros.

Nebot [87] presents the aforementioned biases in a more formal manner shown in (3.13) and (3.14). Gyro bias is a function of a residual bias, linear accelerations, angular rotation rates and random noise (3.13). The residual bias includes the temperature-dependent bias and the turn-on to turn-on bias. Accelerometer bias is a function of residual bias, linear accelerations, and random noise (3.14).

$$\delta w_x = bias_w + bias_g \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} + s_f w_x + m_y w_y + m_z w_z + \eta, \tag{3.13}$$

$$\delta f_x = bias_f + s_f a_x + m_y a_y + m_z a_z + \eta \tag{3.14}$$

where δw_x and δf_x are the gyro and accelerometer biases respectively. $bias_w$ and $bias_f$ are the residual biases for gyros and accelerometers. $bias_g$ is the gyro acceleration bias. s_f is the scale factor term. m is the mounting and internal misalignments of the IMU sensors. η is random noise on the sensor signal.

In order to better understand bias and noise, an experiment is performed on a dataset of IMU readings (See Appendix A.3) recorded during a real experimental run in an outdoor environment. Gyro readings in the pitch direction are first integrated without removing any biases. In Figure 3.4a. the integrated gyroscopic rate about the Y axis (*i.e.*, pitch), is shown. Notice the pronounced error in pitch in the standstill region(0 to $28s$). In this region Pitch should be represented a by straight line which is approximately horizontal since no rotation rate should exist. The bias component 'b' from (3.10) is represented by the slope of the line in the standstill region. In Figure 3.4b the as-is data, with uncorrected biases is used to run the INS algorithm during standstill. Although the vehicle is stationary during this period, the figure indicates that it is moving from the 'x' at the middle towards the left (bold line). This reconfirms the presence of biases that are detrimental to the INS performance. In Figure 3.4c the biases are removed from the data sets. Notice that the previously sloped line is now horizontal, indicating that biases are removed. The line is not perfectly flat because of the random noise in the system. In Figure 3.4d the rectified data is used in the INS and the results are much improved. However, there still exist a drift from the true position due to random noise in the gyro sensor.

In addition to temperature dependent biases, inline biases, scale biases, and

turn-on to turn-on biases, accelerometers are susceptible to gravity effects, which must be accounted for and removed from the accelerometer readings before integrating them. Such errors are dependent on the inclination of the IMU axes with respect to the gravitational force and are particularly important in land-based navigation where the applied accelerations are comparable to those caused by gravity. In other words, if the vehicle is pitched at an angle θ, the accelerometer in the direction of the vehicle falsely reports an acceleration of $g * sin\theta$; for tilt angles in the order of five to ten degrees, this gravitational error in the forward direction is in the order of 0.1g to 0.2g which is comparable to those accelerations encountered in land based navigation. Therefore, any error in the estimated pitch θ can seriously effect the precision of the predicted forward acceleration.

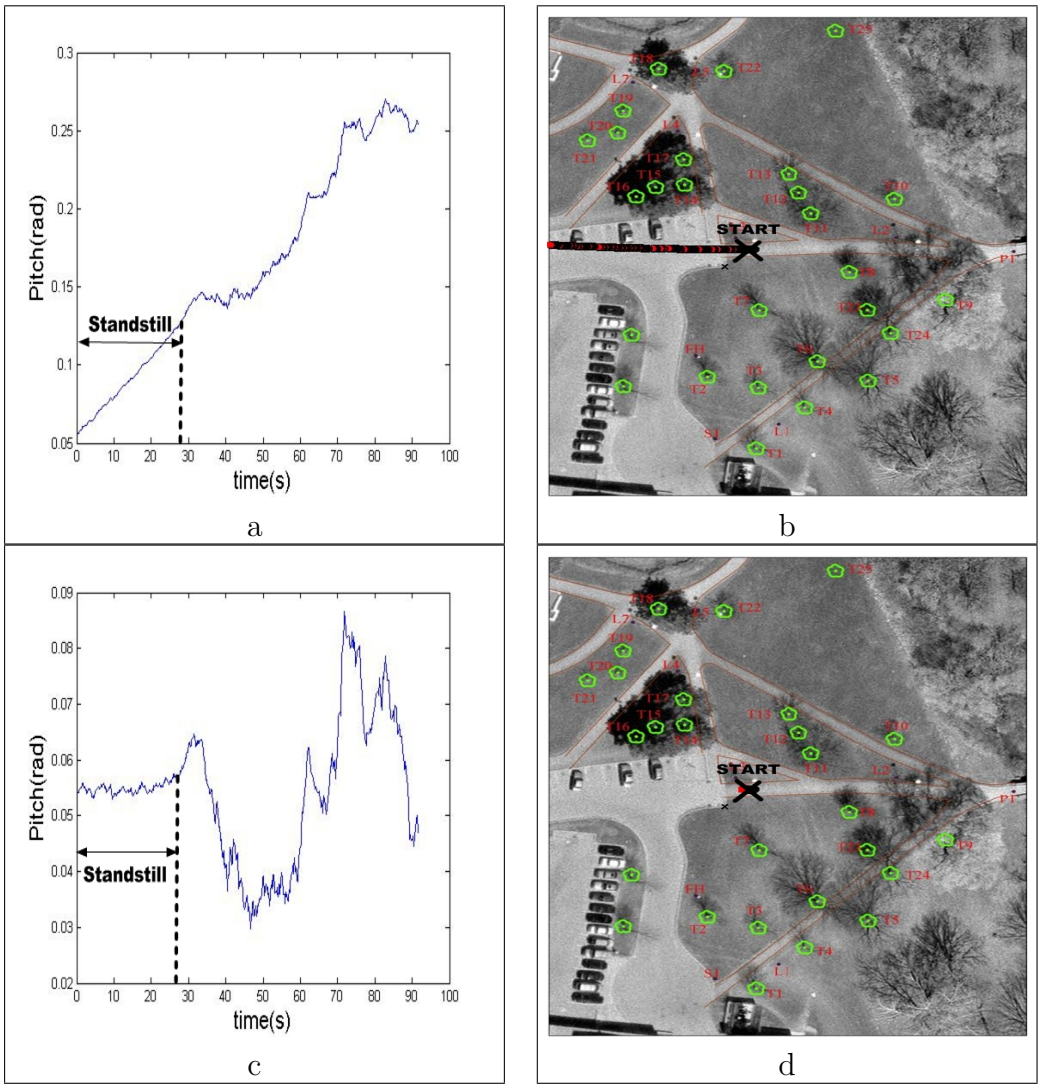


Figure 3.4: Effect of gyro bias on the acceleration and dead-reckoning estimate. (a) Pitch angle throughout the first part of a sample run. The cart is stationary until approximately 28 seconds. This should be evidenced by a horizontal line in the Pitch-time graph, rather than a sloped line; indicative of a linear time dependent bias. (b) INS prediction using the as-is data; although the cart is stationary, the image shows a severe drift of the cart to the let of the image (bold line). (c) Pitch after correcting for bias. Notice that the bias has been removed from the stationary region (horizontal line). (d) Some drift due to random noise still exists in the system.

3.4.1 IMU data correction

Two methods are presented here for correcting IMU data. The first method is the most precise of the two and is applied if the IMU unit has been calibrated and a resulting calibration file exits. The second method is less precise and is used to approximate the IMU biases when no calibration information is provided.

Calibration method

In this first method, hereafter referred to as 'calibration method', IMU are calibrated based on the manufacturer instructions [89]. The calibration file that is supplied by the manufacturer contains four pieces of information, which are used to post process and subsequently correct data collected by the IMU. These items include:

1. A cross-axis correction matrix for the Accelerometer data K_A.
2. A cross-axis correction matrix for the Gyroscope data K_G.
3. A Gyroscope Acceleration Bias matrix $K_{G_{AB}}$.
4. A Gyro Gain vector GG.

In the IMU used by VisSLAM, temperature-dependent biases are taken into account by the software of the IMU manufacturer and do not have to be addressed. Removing the remaining biases is achieved according to the following calibration sequence. First, the accelerometer data is corrected by multiplying it by the cross-axis correction matrix K_A:

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}_{true} = K_A \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}_{meas}, \tag{3.15}$$

where K_A is taken from the calibration file, the subscript *true* refers to the corrected accelerations and the subscript *meas* refers to the as-is measured accelerations from the IMU.

Once the true acceleration is calculated the gyro acceleration bias is calculated:

$$G_{ABVolts} = K_{G_{AB}} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}_{true}, \tag{3.16}$$

where GAB_{Volts} is the calculated gyro acceleration bias in Volts, A_{true} is the true acceleration calculated in (3.15), K_{GAB} is the gyro acceleration matrix taken from the calibration file. The gyro acceleration bias is then expressed in $^\circ/s$ by multiplying it by the Gyro Gain

$$\begin{bmatrix} G_{ABx} \\ G_{ABy} \\ G_{ABz} \end{bmatrix}_{^\circ/s} = \frac{1}{0.8085} GG \begin{bmatrix} G_{ABx} \\ G_{ABy} \\ G_{ABz} \end{bmatrix}_{Volts}, \tag{3.17}$$

where 0.8085 is scaling due to a voltage divider. The gyroscope acceleration bias causes a shift in the gyroscope cross-axis bias when the system is accelerating. Therefore, this shift must be accounted for and removed from the measured gyroscope readings before correcting for gyroscope cross-axis error.

$$G_{true} = K_G(G_{meas} - G_{AB}), \tag{3.18}$$

where G_{true} is the true gyroscope readings, G_{meas} is the as-is measured gyroscope readings from the gyroscopes, and G_{AB} is the gyro acceleration bias calculated in (3.17).

As a final corrective measure, the turn-on to turn-on bias is removed by plotting the gyroscope reading at standstill and removing any offset from zero. Figure 3.5 shows an example of this procedure where the calibrated GyroZ is plotted with turn-on to turn-on bias included (left) and without the turn-on to turn-on bias (right). Notice how the average gyro reading in the uncorrected image is around -0.25 rad/s, whereas the average in the correct one is approximately zero.

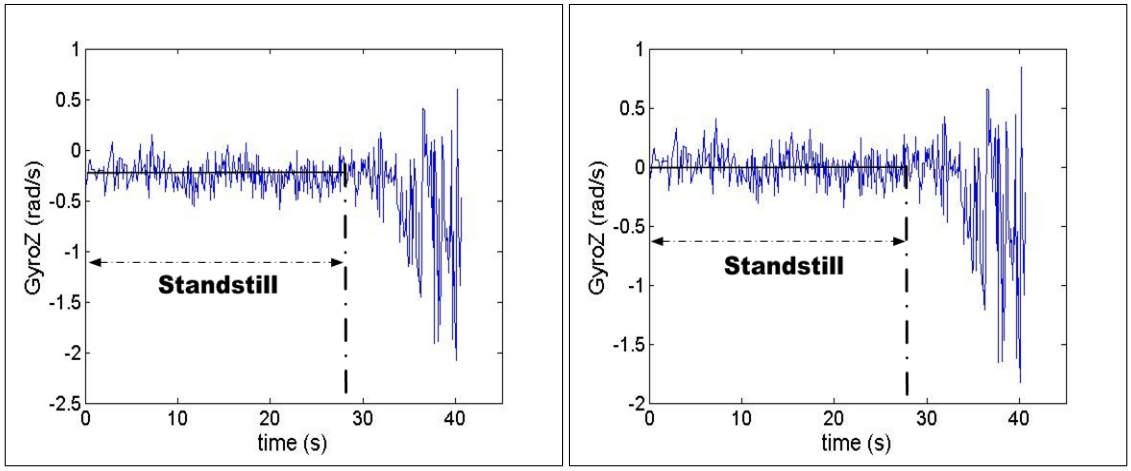


Figure 3.5: Removing turn-on to turn-on bias. In the left image the gyroZ data is calibrated but the turn-on to turn-on bias is not removed. In the right image the turn-on to turn-on is removed.

Approximate method

Alternatively, if a calibration file is not available, the biases can be approximated as suggested by Nebot and Durrant-Whyte [88], in which accelerometer bias is calculated as follows

$$a_x^{bias} = a_x^{meas} - g * \sin(elevation) \tag{3.19a}$$

$$a_y^{bias} = a_y^{meas} + g * \sin(bank) \tag{3.19b}$$

$$a_z^{bias} = a_z^{meas} - g * (1 - \cos(elevation)\cos(bank)) \tag{3.19c}$$

where the subscripts *bias* and *meas* correspond to the bias and measured accelerations respectively. The *elevation* and *bank* are the initial elevation and bank of the vehicle. Nebot and Durrant-Whyte [88] determine elevation and bank by adding an additional two pendulum gyros to their vehicle. Since the system here is not equipped with such sensors, the original bank and elevation of the vehicle are calculated by ignoring these biases (3.20) and assuming that the accelerometer reading used in 3.6a and 3.6b are true accelerations.

$$bank = \arcsin(\sin(\phi_0) * \cos(\theta_0)) \tag{3.20a}$$

$$elev = \theta_0 \tag{3.20b}$$

where ϕ_0 and θ_0 are the euler angles averaged at standstill (calculated in (3.6a) and (3.6b)). This method is error prone because it does not take into account the bias errors when estimating the initial Euler angles. Nevertheless, in the absence of tilt and bank sensors it is the only possible alternative. The biases in the gyros are calculated by estimating the gyro readings at standstill, where any readings from the gyros are due to their inherent biases and not to any motion.

The effects of using these simplifying assumptions on the INS system are investigated next.

Comparison of methods

In this section, the two methods discussed above are analyzed and compared by running them on a data set of IMU readings collected during an experimental run (See Appendix A.3 for a description of this data set). Both methods are used to compute the corrected accelerometer and gyroscopes readings and subsequently plot and compare them in order to understand the predictive capabilities of both. One would predict that the re-calibration based on actual data has to be at least as good as the tables provided by the sensor manufacturer. The problem is that if the system keeps running for a long period of time then continuous adjustment with the manufacturer table could provide better results.

In Figure 3.6 the gyroscope data about the X, Y, and Z directions are plotted for both the 'calibration' method (in blue) and the 'approximate' method (in red). These graphs show that both methods yield almost identical results (no major offset of one color over the other). This graph validates the use of the approximate method for the estimation and removal of gyro bias.

In Figure 3.7, the accelerometer data in the X,Y, and Z directions in the navigation frame are plotted for the two calibration methods. Notice that both method closely agree for all three methods. Notice how all acceleration are zero at standstill before the vehicle commences to navigate.

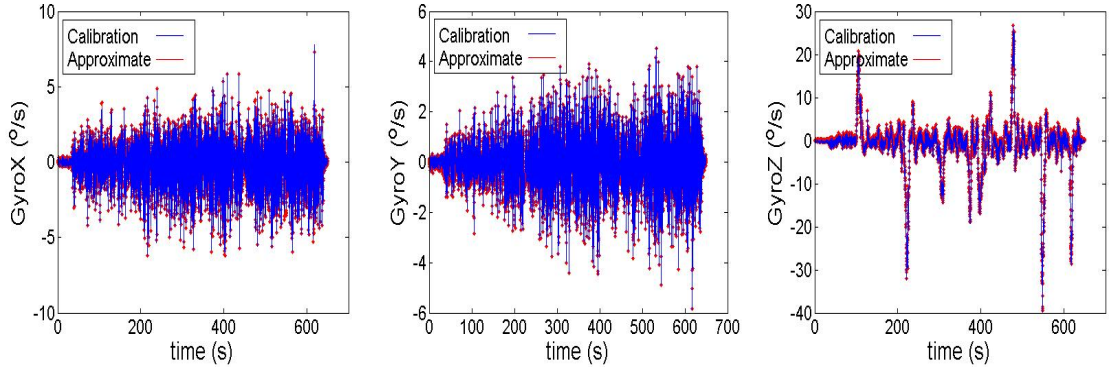


Figure 3.6: Comparison of GyroX, GyroY, and GyroZ between the 'Calibration' method (in blue) and the 'approximation' method (in red). There is strong agreement between the two methods for all three Gyro sensors.

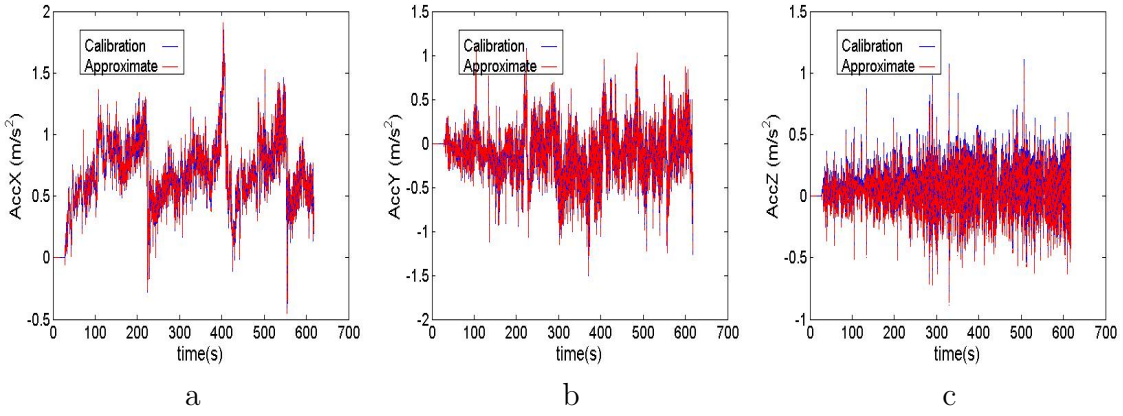


Figure 3.7: Comparison of AccX, AccY, and AccZ between the 'Calibration' method (in blue) and the 'approximation' method (in red). Both methods closely agree for all three accelerations.

Since the IMU that is used in this work is in fact calibrated and a calibration file is supplied from the manufacturer, the calibration method is used to remove biases from the IMU sensors for the remaining of the VisSLAM experiments.

3.5 Enforcing constraints

In land-based navigation, the INS can be made more robust by enforcing what is known as *non-holonomic* constraints, which imply that the navigating vehicle can move in only one direction at any one time. In other words, if the vehicle is moving in a forward direction, no lateral movement occurs. A further constraint is that the vehicle remains in contact with the ground (*i.e.*, no motion in the vertical direction). Dissanayake *et al.* [90] introduce two methods to enforce these constraints: (1) the direct method, and (2) the EKF method.

3.5.1 Direct method

A simple method to enforce non-holonomic constraints is to reformulate the state transition matrix in function of the distance along the path of the vehicle. The velocity vector of the vehicle in the navigation frame is aligned with the forward velocity of the vehicle body (note this direction as $\vec{b_x}$). Let s, $\dot{s}$, and $\ddot{s}$ denote the respective displacement, velocity, and acceleration of the vehicle in the forward direction. As a result,

$$V = \dot{s}\vec{b_x} \tag{3.21}$$

Similarly, the acceleration of the vehicle is taken as the derivative of forward velocity

$$\vec{A} = \dot{V} = \ddot{s}\vec{b_x} + \dot{s}\dot{\vec{b}}_x \tag{3.22}$$

Taking the derivative of b_x as the cross product of its rotation vector and itself and expanding the cross product yields

$$\begin{aligned} \vec{A} &= \ddot{s}\vec{b_x} + \dot{s}w_b \times \vec{b_x} \\ \vec{A} &= \ddot{s}\vec{b_x} + \dot{s}w_z\vec{b_y} - \dot{s}w_y\vec{b_z} \end{aligned} \tag{3.23}$$

The values $\ddot{s}$, $\dot{s}w_z$, and $-\dot{s}w_y$ in (3.23) are the magnitudes of the accelerations of the vehicle in the $\vec{b_x}$, $\vec{b_y}$, and $\vec{b_z}$ directions of the body frame. Expressing the forward velocity $\dot{s}$ as V_f, allows one to write these magnitudes as $\dot{V_f}$, $V_f w_z$, and $-V_f w_y$ respectively. In the absence of Gravity, these three values represent the accelerations felt by the accelerometers mounted on the body of the vehicle. The effect of the gravity is added to these values by transforming the gravitational force $F_g = [0; 0; g]$ to the body coordinate system by simply multiplying F_g by the inverse of the direction cosine as follows:

$$\begin{bmatrix} A_{bx} \\ A_{by} \\ A_{bz} \end{bmatrix} = \begin{bmatrix} \dot{V}_f \\ V_f w_z \\ -V_f w_y \end{bmatrix} + \begin{bmatrix} C\theta C\psi & C\theta S\psi & -S\theta \\ -C\phi S\psi + S\phi S\theta C\psi & C\phi C\psi + S\phi S\theta S\psi & S\phi C\theta \\ S\phi S\psi + C\phi S\theta C\psi & -S\phi C\psi + C\phi S\theta S\psi & C\phi C\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \quad (3.24)$$

$$\dot{V}_f - A_{bx} + gsin\theta = 0 \quad (3.25a)$$

$$V_f w_z - A_{by} - g \sin\phi \cos\theta = 0 \quad (3.25b)$$

$$V_f w_y - A_{bz} + g \cos\phi \cos\theta = 0 \quad (3.25c)$$

Dissanayake *et al.* present results for experiments in a simulated setting where the velocity is set to a constant value and therefore $\dot{V}_f$ in 3.25a is zero and the pitch θ is then directly observable. The forward velocity V_f and roll θ are then found by solving the two equations with two unknowns 3.25a and 3.25c. Unfortunately in a real life setting, it is not possible to assume constant velocity and there always exist some fluctuations in the velocity. In such situations, Dissanayake *et al.* recommend manipulating 3.25a and 3.25c to obtain an expression of pitch and subbing it into 3.25a in order to obtain a first order differential equation for V_f. What the authors fail to notice is that Pitch can only be obtained as a function of its trigonometric Cosine, which when converted to its equivalent Sine value (such as required in 3.25a) no information is given to the sign (*i.e.*, plus or minus) of this angle. In a more mathematical form, squaring 3.25a and 3.25c and rearranging:

$$g^2 \sin^2\phi \cos^2\theta = Aby^2 + V_f^2 w_z^2 - 2A_{by} V_f W_z \quad (3.26a)$$

$$g^2 \cos^2\phi \cos^2\theta = Abz^2 + V_f^2 w_y^2 - 2A_{bz} V_f W_y \quad (3.26b)$$

Adding 3.26a and 3.26b and rearranging yields an expression for the Cosine of the Pitch as

$$\cos^2\theta = \frac{(A_{by}^2 + A_{bz}^2) + V_f^2(w_z^2 + w_y^2) - 2V_f(A_{by} w_z + A_{bz} w_y)}{g^2} \quad (3.27)$$

The Sine of the Pitch angle θ is calculated as $\pm\sqrt{1 - \cos^2\theta}$ and is subbed into (3.25a) to yield

$$\frac{dV_f}{dt} - A_{bx} \pm g\sqrt{g^2 - (A_{by}^2 + A_{bz}^2) + V_f^2(w_z^2 + w_y^2) - 2V_f(A_{by}w_z + A_{bz}w_y)} = 0 \quad (3.28)$$

In the first order non-linear differential equation (3.28), the value of V_f is different depending on the sign of the radical, which is directly related to the sign of the Pitch angle. It is therefore apparent, that without further knowledge of the sign of the Pitch angle, the 'Direct' method just proposed is lacking.

This constraint method could work very well if a wheel encoder were available and V_f is directly observable. In this case the non-holonomic constraints are $V_y = V_z = 0$ and $V_x = V_{\mathbf{f}orward}$. These equations render the problem extremely simple since the velocities can now be expressed in the navigation frame through the transformation matrix C_b^n (from body to navigation frame). The system thus becomes linear and a simple Kalman Filter or Information Filter can be used to predict ego-motion.

3.5.2 Extended Kalman Filter method

Enforcing land-based constraint can also be performed using an Extended Kalman Filter. The no lateral movement is imposed via the EKF observation model, by assuming that an observation represents two vehicle velocities, one in the lateral direction and the other in the vertical direction. The motion constraints are then imposed by setting these velocities to zero at each iteration. Mathematically, this implies:

$$V_{by} - \nu_y = 0 \quad (3.29a)$$

$$V_{bz} - \nu_z = 0 \quad (3.29b)$$

where V_{by} and V_{bz} are the velocity components of the vehicle in the body coordinate frame, ν_y and ν_z are the respective noise values associated with these velocities. V_b is expressed in the navigation frame as

$$V_b = [C_b^n]^T V_n \quad (3.30)$$

where C_b^n is the direction cosine matrix shown in (3.2). Expanding (3.30) yields

$$z(k) = h(x(k), w(k)) = \begin{bmatrix} V_{by} \\ V_{bz} \end{bmatrix} =$$
$$\begin{bmatrix} V_{nx}(s\phi s\theta c\psi - c\phi s\psi) + V_{ny}(c\phi c\psi + s\phi s\theta s\psi) + V_{nz} s\phi c\theta \\ V_{nx}(c\phi s\theta c\psi + s\phi s\psi) + V_{ny}(-s\phi c\psi + c\phi s\theta s\psi) + V_{nz} c\phi c\theta \end{bmatrix} + \begin{bmatrix} \nu_y \\ \nu_z \end{bmatrix}, \quad (3.31)$$

where s and c are the trigonometric Sine and Cosine, and ϕ, θ, and ψ are the Euler angles.

Equation (3.31) represents the velocity of the vehicle in the body coordinate frame; it is desired to set the two rows of (3.31) to zero at each time step in order to satisfy the non-holonomic and ground navigation constraints. The navigation system is designed in the framework of an EKF, where predictions are obtained from the INS state transition equations, which includes three entries for position (P_n), three for linear velocities (V_n) and three for the Euler angles (ϕ, θ, ψ).

$$\dot{x} = f(x, u) = \begin{bmatrix} \dot{P}_n \\ \dot{V}_n \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V_n \\ C_b^n A_b - G \\ w_{bx} + (w_{by} \sin\phi - w_{bz} \cos\phi) \tan\theta \\ w_{by} \cos\phi - w_{bz} \sin\phi \\ \frac{w_{by} \sin\phi + w_{bz} \cos\phi}{\cos\theta} \end{bmatrix}, \quad (3.32)$$

Kalman Filters and Extended Kalman Filters are applied by integrating two sources of information. First a prediction of a queried state and corresponding covariance is performed. Next, an observation model, which is a function of the state, is used to predict a set of variables. At the same time, these variables are observed using an alternate sensor. The difference between the observations and the predicted observation constitutes a metric known as the 'Innovation', which is used to update the initial predicted state and covariance. The equations for prediction, observation, and update of the INS EKF are presented next.

Prediction

$$\hat{x}(k|k-1) = f(\hat{x}(k-1|k-1), u(k-1), (k-1)) \quad (3.33a)$$
$$\hat{z}(k|k-1) = h(\hat{x}(k|k-1)) \quad (3.33b)$$
$$P(k|k-1) = J_s(k-1) P(k-1|k-1) J_s(k-1)^T + J_u(k-1) Q J_u(k-1)^T \quad (3.33c)$$

where $\hat{x}$ is the predicted state $(x, y, z, V_x, V_y, V_z, \phi, \theta, \psi)$ of the vehicle, $f(\hat{x}, ...)$ is the

state transition function shown in (3.32), $u(k)$ is the process $(a_x, a_y, a_z, w_x, w_y, w_z)$. $\hat{z}(k|k-1)$ is the predicted observation at time t_{k-1} obtained from the observation model h calculated in (3.31). J_s and J_u are the Jacobians of the state transition function $f(\hat{x})$ with respect to the state and process respectively. The calculation of these Jacobians is performed numerically as shown in Appendix B.2. Q is the noise associated with the process (*i.e.*, expected noise in IMU gyros and accelerometers).

Observation The observation model here is used to impose non-holonomic constraints through the mechanics of an EKF. This objective is achieved by supposing that the system observes, at each iteration, the lateral and vertical vehicle velocities, expressed in the body reference frame. The system further supposes that these two velocities are observed to be zero at every iteration and then update the EKF accordingly. At each update iteration, an innovation is calculated from the difference between the predicted observation (*i.e.*, found via (3.31)) and the new observation (*i.e.*, it is set to zero). More formally, the innovation μ is expressed as

$$\mu = 0 - \hat{z}(k|k-1) \tag{3.34}$$

At this stage an innovation covariance can also be calculated as

$$S(k|k-1) = J_x(k-1)P(k|k-1)J_x(k-1)^T + J_w(k-1)R(k-1)J_w(k-1)^T \tag{3.35}$$

where J_x and J_w are the Jacobians of the observation model with respect to the state and observation (v_{by},v_{bz}) respectively. J_x and J_w are calculated numerically as shown in Appendix B.2. R is the noise in the assumed non-holonomic model, or in other words, the precision of the non-holonomic assumptions.

Update The update is based on the EKF formulation as follows:

$$\hat{x}(k|k) = \hat{x}(k|k-1) + W(k)(\mu) \tag{3.36a}$$

$$P(k|k) = P(k|k-1) - W(k)S(k)W^T(k) \tag{3.36b}$$

where $W(k)$ is the gain matrix and is given by

$$W(k) = P(k|k-1)J_x(k-1)^T S^{-1}(k) \tag{3.37}$$

where $P(k|k-1)$ is the predicted covariance obtained through (3.33c) and S is the innovation covariance calculated in (3.35).

Figure 3.8 shows the effect of the non-holonomic constraints on the velocity in the lateral vehicle direction Y . Ideally, one would want the velocity to remain zero but in reality the best the EKF can do is to have this value fluctuate about zero. In the left image, the system includes no constraints, notice how V_y increases very quickly and ends up at around 25 m/s at 100 seconds, whereas in the constrained plot V_y remains approximately zero except at approximately 70 seconds where there is a pronounced jump to 0.4 m/s in the lateral velocity. This is probably due to excessive slippage of the vehicle at this point, causing the non-holonomic constraints to be violated.

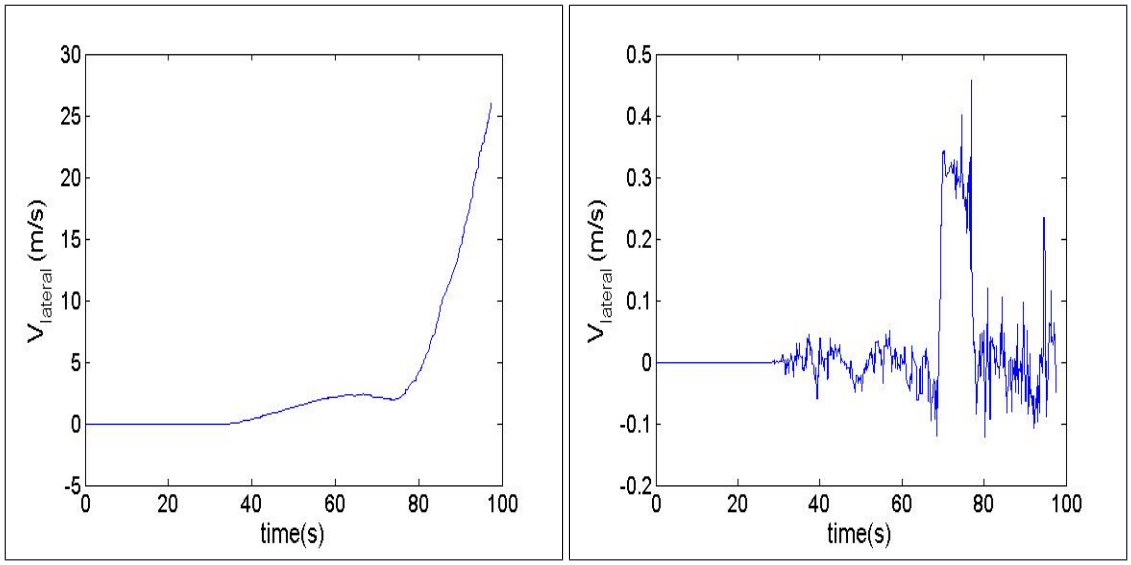


Figure 3.8: Effect of enforcing non-holonomic constraints on the lateral velocity. In the left image no constraints are imposed whereas in the right image, the non-holonomic constraints are imposed via the EKF.

3.6 Architecture

The above discussion regarding the INS is summarized in the flowchart of Figure 3.9. The INS is structured within the framework of an EKF, whose prediction step estimates the position, velocity and bearing of the vehicle by integrating rectified accelerometer and gyroscope readings. The covariance of the system is estimated

using the Jacobians of the state transition function J_s and J_u with respect to the state and process variables respectively. The observation part of the EKF (Section 3.5.2) is used to impose, through the mechanics of the EKF, a set of constraints regarding the allowable modes of motion of the vehicle. This is done by formulating the observation as a vector containing the lateral and vertical velocities V_{by} and V_{bz} of the vehicle and setting these velocities to zero (plus noise ν) at each time step. Finally, the update step uses the difference between the expected values of the velocities (*i.e.*, zero) and the prediction of these velocities from the observation model h, to calculate a new state X_{new} and new covariance P_{new}.

The output of the INS system is used as the prediction step for the VisSLAM system. In effect, what the INS system offers is a prediction of state in the form of a Probability Distribution Function (PDF) with a mean X_{INS} and a covariance P_{INS} expressed as follows:

$$X_{INS} = \begin{bmatrix} x \\ y \\ z \\ V_x \\ V_y \\ V_z \\ \phi \\ \theta \\ \psi \end{bmatrix}, \tag{3.38}$$

where x, y, z are the predicted coordinates of the vehicle, V_x, V_y, V_z are velocities of the vehicle expressed in the navigation frame, and ϕ, θ, ψ are the Euler angles representing the 3D orientation of the vehicle.

$$P_{INS} = \begin{bmatrix} \sigma_x\sigma_x & \cdots & \sigma_x\sigma_{V_x} & \cdots \sigma_x\sigma_\psi \\ \vdots & \ddots & & \\ \sigma_{V_x}\sigma_x & & & \\ \vdots & & & \\ \sigma_\psi\sigma_x & & & \sigma_\psi\sigma_\psi \end{bmatrix}, \tag{3.39}$$

where σ stands for standard deviation. The state coordinates 'x' and 'y' are initialized to the GPS readings at the GPS time stamp closest to that of the first captured image, since this time indicates when navigation commences. The state coordinate 'z' is the height of the IMU at startup and is determined to be 0.4 meters. V_x, V_y,

and V_z are zero at startup since the vehicle begins its journey at standstill. The calculation of the initial Euler angles ϕ, θ, and ψ is shown in 3.6a 3.6b, and 3.7 respectively.

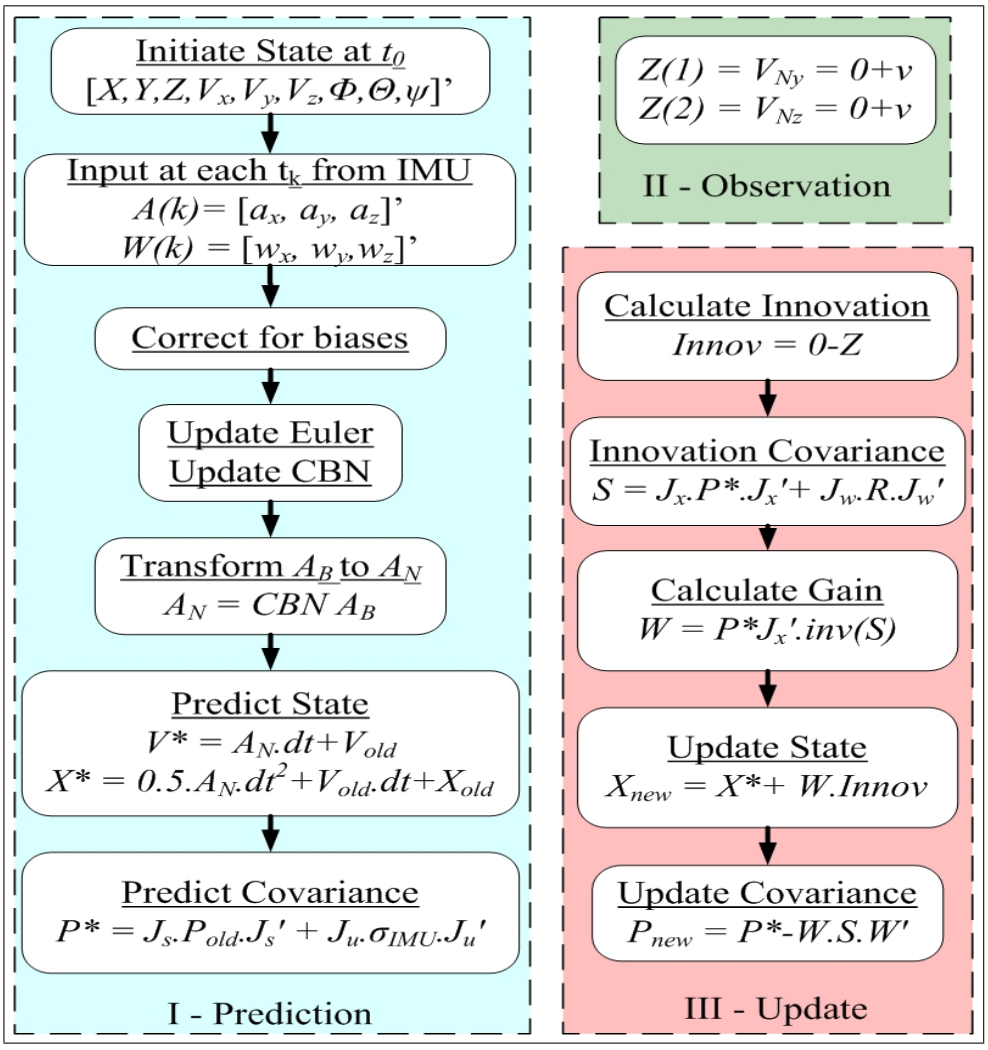
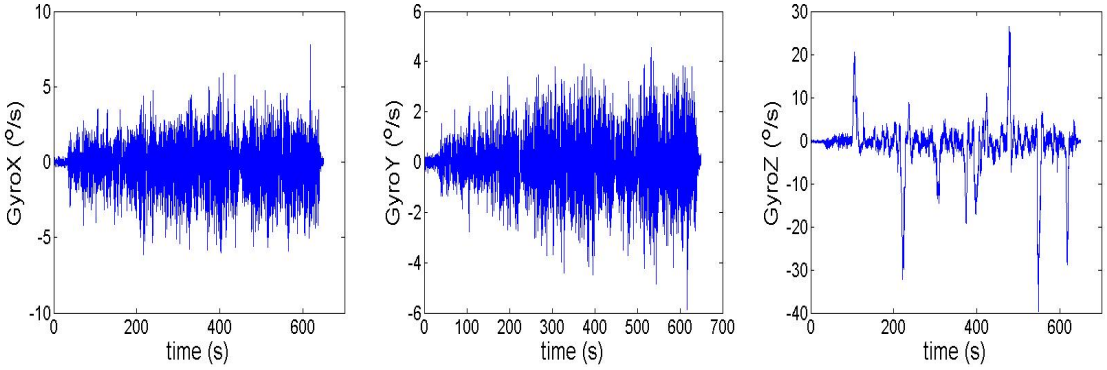


Figure 3.9: The INS is structured in the framework of an EKF. The state variables include position (X, Y, Z), velocity(V_x, V_y, V_z), and Euler angles (ϕ, θ, ψ). $A(k)$ and $W(k)$ are the IMU accelerometer and gyroscope readings at time k. A_N and A_B are the accelerations in the navigation and body frames respectively. V^*, X^*, and P^* are the predicted velocity, position, and covariance respectively of the system. J_s and J_u are the Jacobians of the state transition function with respect to the state and process model respectively. Z is the observation model comprising the lateral and vertical velocities V_{Ny} and V_{Nz} respectively and ν is the associated noise. $Innov$, S, and W are the innovation, innovation covariance and Kalman Gain respectively. J_x and J_w are the Jacobians of the observation model with respect to the state and observation respectively. σ_{IMU} and R are the IMU and observation covariances. All these variables are defined in Section 3.5.2.

In the next section, the INS system described above is tested on a dataset of IMU readings, taken from a real experimental run in an outdoor environment. See Appendix A for a description of the experiment and dataset. The point of this exercise is to twofold: first, to show the benefits of the non-holonomic constraints on the IMU predictive capabilities of the IMU; and second to show that INS based navigation using low cost IMUs fail on their own, thereby promoting the idea of INS-SLAM system.

3.7 Experiments

In the first part of this experimental section, IMU corrected data is plotted verses time. The point of this exercise is to investigate the worth of the IMU data from their respective graphs alone. In Figure 3.10 the three sets of gyroscope readings are plotted against time. The two Graphs 3.10 'a' and 'b' indicate that the IMU remains relatively flat during its journey and any change in roll or pitch (*i.e.*, GyroX and GyroY) is due to random noise in the sensors. The GyroX readings fluctuate between plus or minus $3°/s$ with very few impulses reaching peaks at around $6\ °/s$. In Graph 3.10 'c' the sensor readings are of higher order than the random noise since the readings are indicative of planar rotations of the vehicle around bends during the journey of the vehicle. In order to get a better perspective on the worth of these results, the GyroZ data is plotted along with a corresponding aerial image and shown in Figure 3.11. Critical points representing sharp bends are marked on both the the gyro image and its corresponding location on the aerial image. This graphs indicates that the IMU GyroZ data corresponds to reality, where each peak matches to a change of orientation of the vehicle. At point '1' the vehicle initiates navigation and preserves a constant heading until it reaches point '2', where the vehicle makes a sharp clockwise rotation (IMU coordinate system shown in Figure 3.3 is such that the Z axis points down and a clockwise rotation is positive). At point '3' the vehicle makes a U-turn while rotating counter-clockwise (peak '3') and then retraces its path backwards until it reaches point '4' where it makes a counter-clockwise rotation (peak'4') and travels up and left to peak '5'. The vehicle than closes the loop after going through another two bends (peaks '5' and '6'). Peaks '8' and '9' are repeats of '2' and '3'.



a b c

Figure 3.10: GyroX, GyroY, and GyroZ verses time. It is difficult to fully ascertain the goodness of the IMU data from these graphs.

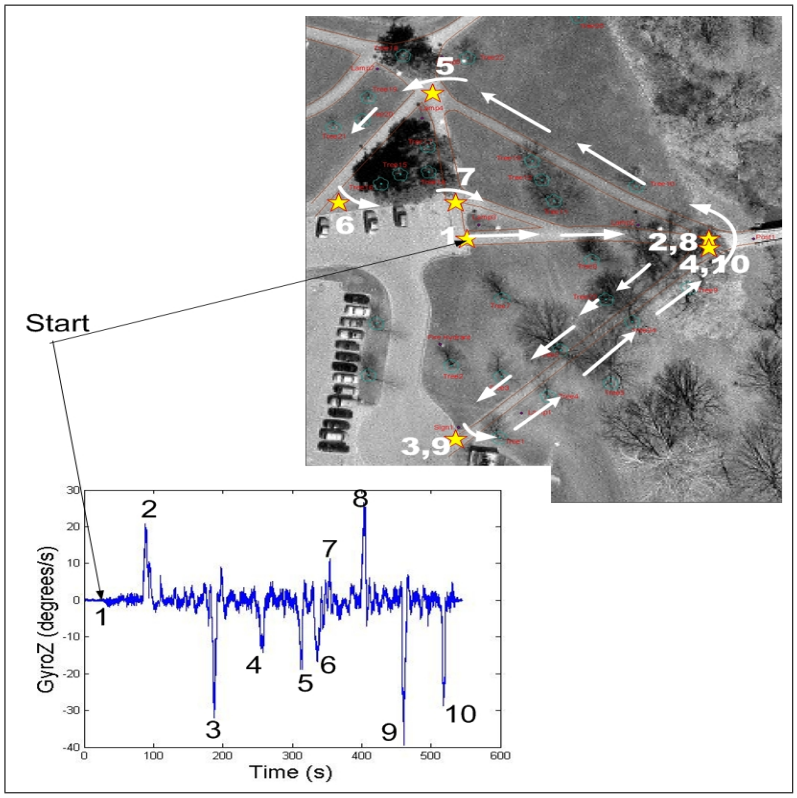


Figure 3.11: Correlating the Gyro readings in the Z direction to their corresponding locations on the aerial image of the test site. Experiment performed on a data set of an experimental run.

Figure 3.12 shows the corrected accelerometer data verses time. Although it is not possible to fully ascertain the quality of the accelerometer data from these graphs, several observations can be made. At startup, all accelerometer data is not centered around zero; this is due to the effect that gravity has on accelerometers in all three directions. Graphs 3.12 'a' and 'b' indicate that the vehicle to which the IMU is fixed is not level at startup; rather, there exists certain bank and elevation angles at startup which cause components of gravity to be recorded by the accelerometers in the X and Y directions. Indeed, it is these values upon which the approximate calibration method described above (Section 3.4.1) is based. Graph 3.12 'c' further validates the fact that the vehicle is not level at startup since the recorded accelerations in the Z directions, which represent gravitational acceleration, are approximately -9.75 m/s^2 instead of -9.81 m/s^2, which is the known gravitational acceleration at the nominal longitude and latitude positions of the test site.

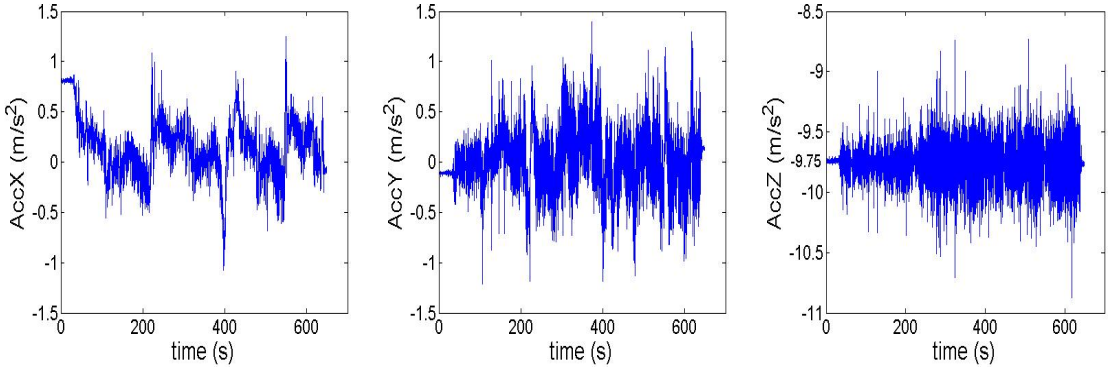
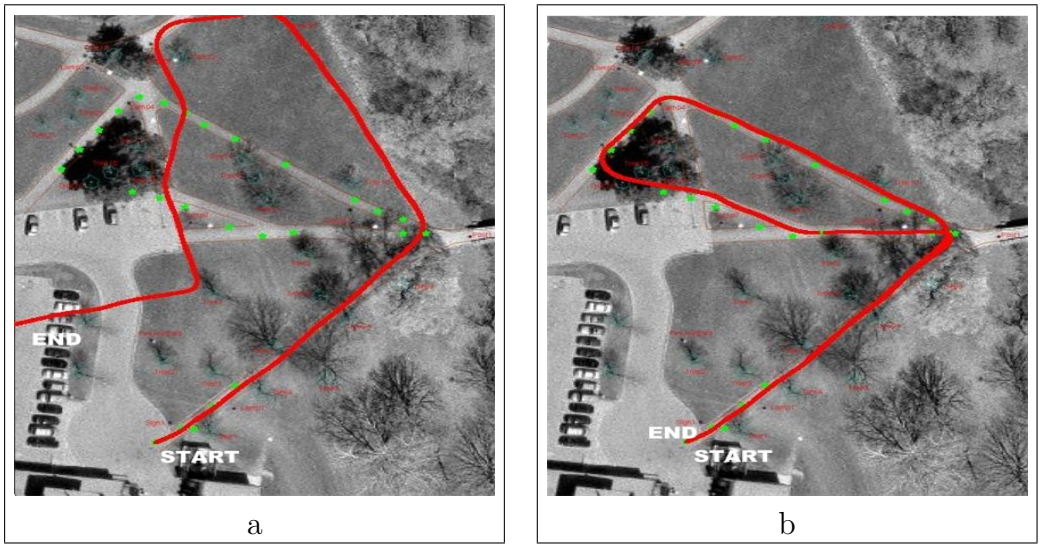


Figure 3.12: AccX, AccY, and AccZ verses time. It is difficult to fully ascertain the worth of the IMU data from these graphs.

In addition to the above analysis, one can check the quality of IMU data by plotting the vehicle path on the ground truth and verifying how closely the INS matches the ground truth. The left image of Figure 3.13 shows a sample INS run while enforcing non-holonomic constraints. The employed strategy also involves capping the velocity at 1 m/s since accelerometer data is bound to diverge if not supplemented by an external source of information (*e.g.*, GPS). This approach is successful on straight paths, but at extreme bends the algorithm does not predict actual motion of the vehicle , thus causing it to deviate from its actual path. In order to highlight this effect, gyroscopic rates are manually compensated at the

first three turns and the result is shown in right image of Figure 3.10. In this experiment the vehicle moves from the bottom-left to the the middle-right and then to the top-left before heading back to the starting point.



a b

Figure 3.13: (a) Enforcing non-holonomic constraints on INS dead-reckoning and capping the forward velocity at 1 m/s. (b) Manually increasing the yaw angles at the first, second, and third turns. The vehicle runs counter-clockwise starting from the bottom of the figure, following the bold line. Experiments are performed on data set '3' of the database of experimental data.

Although INS dead-reckoning succeeds to some extent, INS errors grow without bound if not augmented by some form of absolute positioning systems such as GPS or integrated within a SLAM framework. This issue is evidenced by the quick divergence of the vehicle state estimate on the position verses time plot in Figure 3.14.

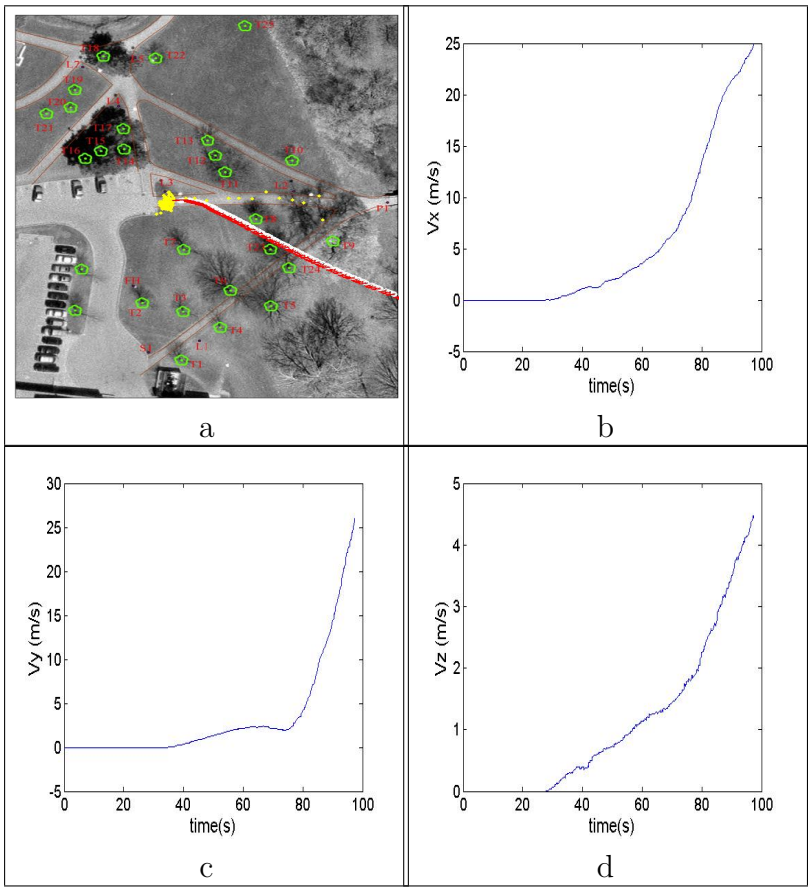


Figure 3.14: Vehicle position (a) and velocities (V_x, V_y, and V_z) without constraints. The velocities are expressed in the navigation frame. Notice how quickly the system diverges when no constraints are enforced.

The effect of adding non-holonomic constraints is shown in Figure 3.15. Although the vehicle does indeed avoid skidding sideways, the IMU errors are not bounded. In fact the system erroneously moves backwards at the outset before moving forwards. A similar error is shown before the vehicle makes its turn where it overshoots the curb and then moves backwards and turns. From the above images in 3.14 and 3.15 it is evident that the INS system alone is lacking and that an alternative navigation system is needed.

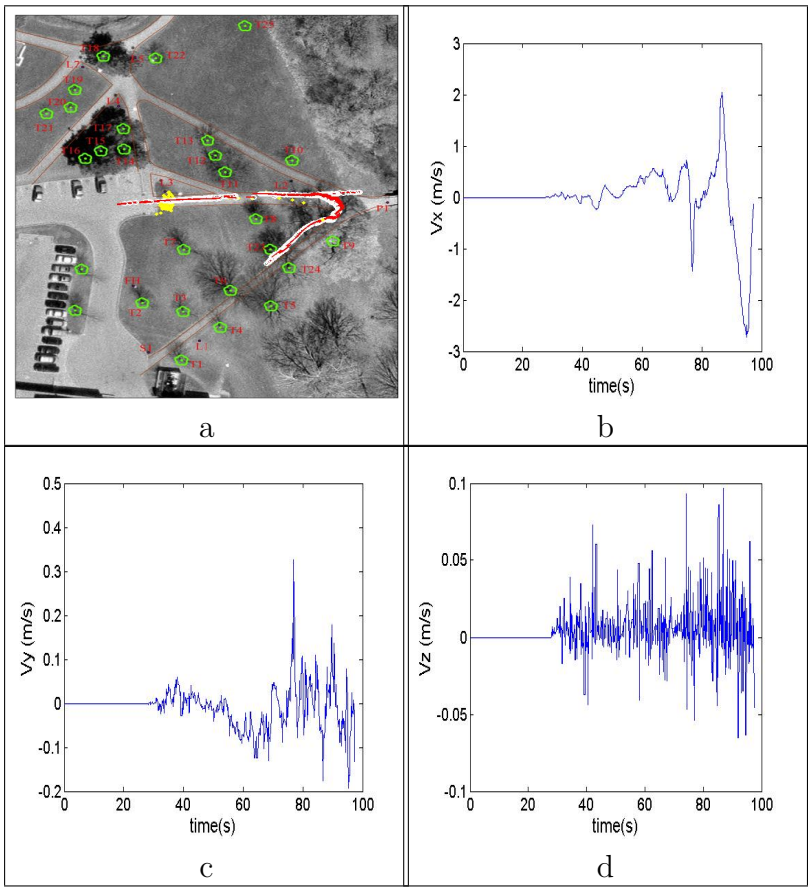


Figure 3.15: Vehicle position (a) and velocities (V_x, V_y, and V_z) with constraints. The velocities are expressed in the navigation frame. Notice how the non-holonomic constraints bound the lateral and vertical vehicle velocities.

3.8 Summary

The INS is the prediction entity for VisSLAM. The predicted state $X = [x, y, z, V_x, V_y, V_z, \phi, \theta, \psi]^T$ and its corresponding covariance matrix are the output of the INS system, and are fed into the VisSLAM system.

The first contribution of this chapter is a comparison of an approximate calibration system proposed in the literature to an exact method used when a unit is calibrated in shop. The approximate method involves correlating the average accelerometer and gyroscope readings at standstill to known gravitational forces.

The two methods yield comparative corrections for gyroscopes data but different corrections for accelerometer data. This is due to an incorrect estimate in the vehicle orientation at standstill.

The second contribution is an evaluation of two theories from the literature for imposing non-holonomic constraints on a land-based inertial navigation system. The first theory involves expressing the vehicle state transition equations in function of the forward velocity alone and thereby eliminating one integration from the position estimation process. This systems is found lacking since the final differential equation involving vehicle forward velocity can only be found up to an unknown sign. In the second method, non-holonomic constraints are enforced via an Extended Kalman Filter (EKF), by formulating the EKF observation model as a vector containing the lateral and vertical velocities of the mobile platform in the body coordinate system, and setting these velocities to zero at each iteration. This method is implemented and indeed helps stabilize the system considerably.

Unfortunately, as evidenced by the experiments in this chapter, regardless of the improvements that might be achieved, INS alone is lacking, and an alternate mode of autonomous navigation such as INS-based SLAM is recommended. Towards this end, a new Inertial-SLAM system named VisSLAM is proposed. VisSLAM uses the INS EKF system developed in this chapter to predict vehicle ego motion and a Computer Vision system, developed in the next chapter for landmark detection, recognition, and initialization.

Chapter 4

Computer vision system

4.1 Introduction

The fundamental contribution of this thesis is the Computer Vision (CV) system presented in this chapter, whose function is to first classify the vehicle milieu and second to detect, recognize, and localize natural landmarks within this milieu.

A fast and robust Environment Recognition (ER) system is developed that can infer the context of the environment (*e.g.*, indoor office, outdoor park) in which an image is taken. Before a vehicle initiates SLAM, ER processes a number of images of the setting in which the vehicle is located and subsequently suggests the context of this scene. VisSLAM can then use this top down information for its map building task, by selecting a type of landmark that is most probably available in the recognized environment. The details of the ER technique are discussed in Section 4.2. In this work, the experimental setting is an outdoor park area and the SLAM landmarks are tree trunks. Nevertheless, the developed techniques lend themselves to other environments featuring different landmarks.

The second Computer Vision module is designed to detect, recognize and localize tree trunks in the vicinity of the vehicle. Tree trunks are detected by first segmenting the query images into quasi-vertical structures and marking those structures that are close to the Ground-Sky (G-S) separation line as tree trunks. The detection system is designed for trees exhibiting visible trunks such as those in Figure 4.1 except for trees of types (E) and (G). Tree trunks are recognized based on a number of their discriminative features, which are extracted via a Scale Invariant Feature Transform (SIFT) filter.

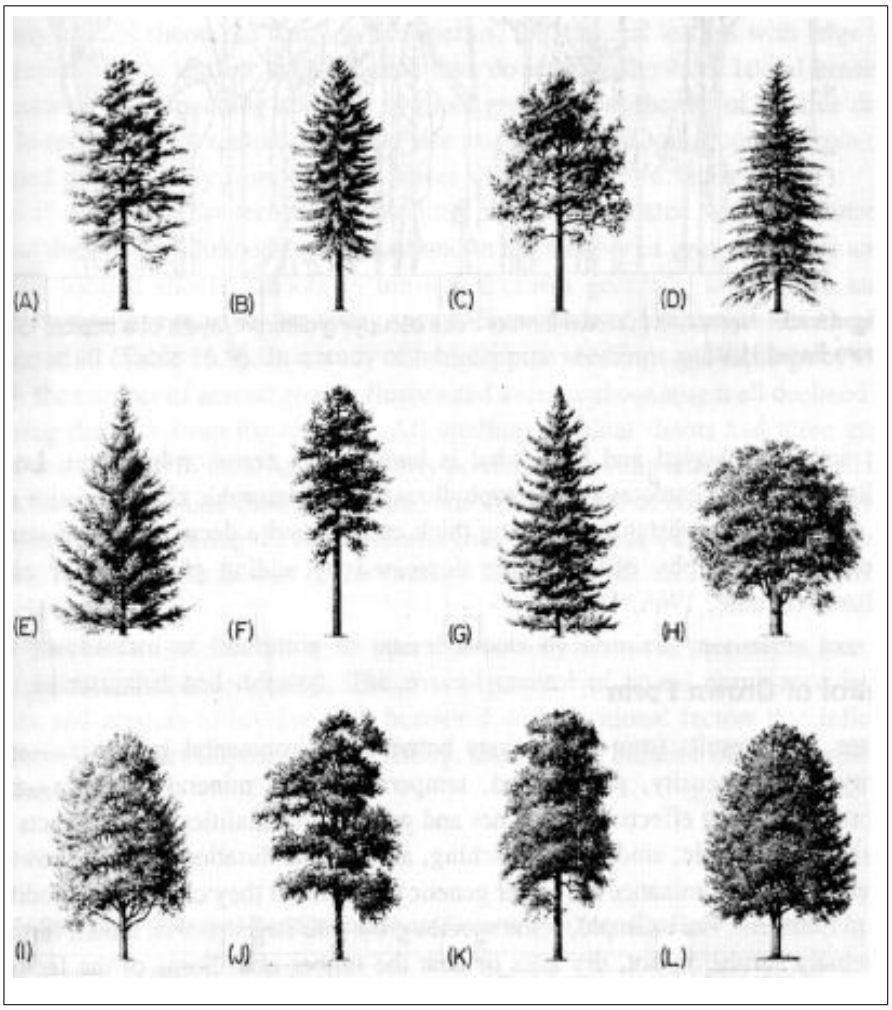


Figure 4.1: Different shapes of trees. The Computer Vision system is intended to detect trees with salient tree trunks. Trees of type (E) and (G) are not expected to be detected by the vision system. Image courtesy of Steve Nix [91].

VisSLAM is a bearing-only system, in the sense that only bearing observations are used to update the SLAM predictions. Nevertheless, stereo vision is used to initialize tree trunks into the SLAM state. Stereo vision offers precise depth information, which reduces uncertainties in the landmark estimates when they are initialized. Alternatively, landmark initialization could be performed in the standard bearing-only techniques but VisSLAM experimental conditions (*i.e.*, straight line trajectories) are poor for the application of such initialization techniques. Stereo information can further provide invaluable information that can assist in object

detection and recognition. In this context, the advantages of using stereo is investigated for each of the Computer Vision systems proposed in this chapter.

The remainder of this chapter is structured as follows: Section 4.2 introduces Environment Recognition techniques, including top-down and bottom-up approaches before presenting the ER system developed for VisSLAM. Section 4.3 introduces two variants for object detection, one based on Interest Points and the other based on Segmentation. The Interest Point method is discarded due to unacceptable shortcomings in the technique. Section 4.4 addresses object recognition and introduces a fast and efficient tree trunk recognition system based on matching objects in feature space. Section 4.5 describes landmark initialization in the context of SLAM, and presents a methodology for initializing trees based on stereo vision. Section 4.6 summarizes the architecture of the Computer Vision system. Section 4.7 discusses experiments performed on the proposed system, starting with a description of the experimental methodology and ending with a presentation of the experimental results. Finally, Section 4.8 draws relevant conclusions and recommendations.

4.2 Environment recognition

Context is valuable knowledge that can aid a robot to autonomously navigate across different environments. Before performing SLAM, it is beneficial for the robot to analyze and thereby categorize the setting around it (*i.e.*, office, street, underwater, *etc.*). Once this knowledge is acquired, the robot uses landmarks that are typical of the recognized environment to perform SLAM. A database of environment-landmark pairs in the form of a look up table can then be set up at a preprocessing stage. Looking at the four images in Figure 4.2 it is difficult to specify features that are common to all four settings; while in the urban setting lamp posts or windows are good features, in underwater settings rocks are better candidates. The general idea in environment recognition is to reduce the search space to features that are typical of that environment. In this manner, the system can be designed as a set of rules rather than a complex learning paradigm. The rules dictate the type of landmarks to use for SLAM based on the recognized environment. It goes without saying that an exhaustive list of environments is not possible; however, the list can be general enough to encompass most of the areas that the robot is likely to visit. In this thesis, SLAM is performed in only one environment and therefore ER is only used to confirm that the mobile platform is indeed located in a park. It can be easily deduced how such a system can be applied to multiple environments using different landmarks for each environment.

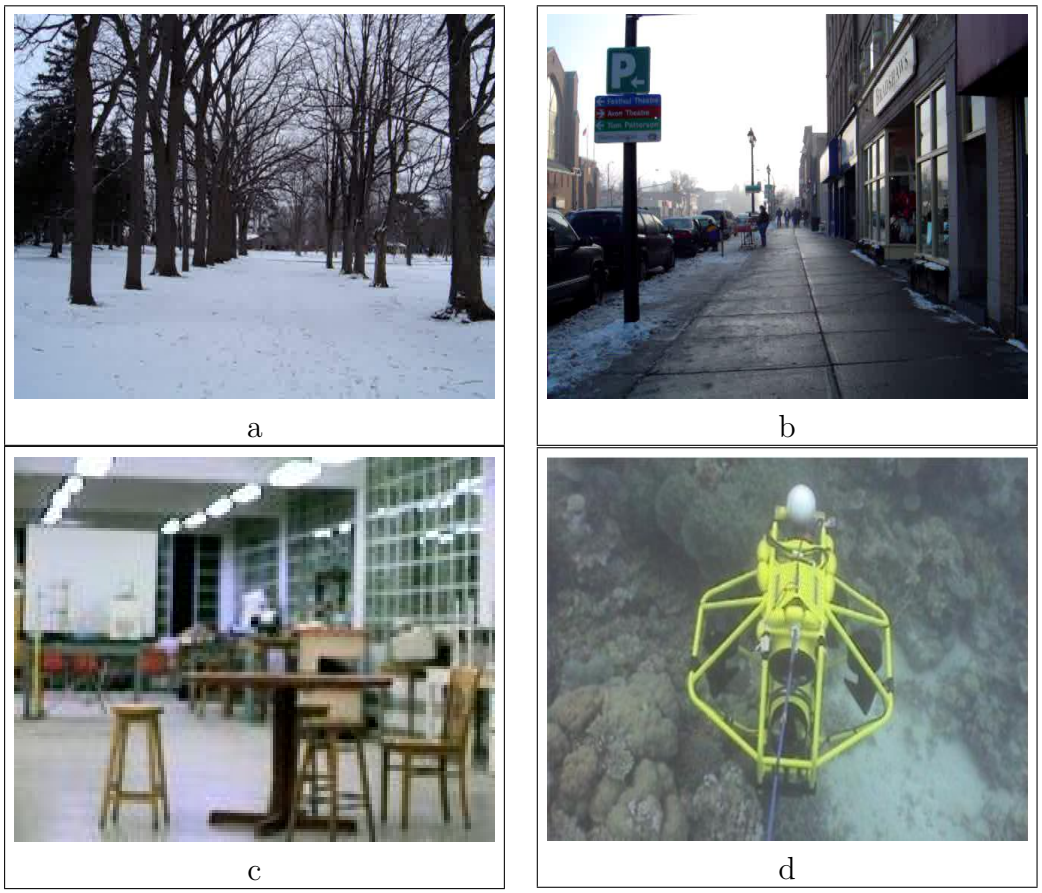


Figure 4.2: Images taken in four different setting. (a) park, (b) urban, (c) lab, (d) underwater (taken from Williams [92]).

In Computer Vision, ER is done in either a bottom-up or top-down approach. In the former case the environment is inferred from a set of objects that are recognized in the environment. This is considered a hard problem in the Computer Vision community and is still unresolved. The complexity resides in the ability to represent objects in such a way that computer vision algorithms are capable of recognizing them from different views, scales and lighting conditions. The order of complexity of such scene recognition algorithms is on par with that of object recognition, which is too costly to be implemented in real-time.

Alternatively, ER is performed in a top-down fashion, by analyzing the holistic content of each image of a queried environment. Top-down methods are appearance-based, where scenes are recognized by analyzing the spatial organization of struc-

tural elements of the images of the scene. Each image class exhibits a distinctive structural signature that is sufficient to discriminate it from other scene classes. Top-down approaches are robust and fast enough to be run in real-time. Structure is relevant to the type of image processing that is undertaken. Windowed Fourier Transforms, Gabor filters or Wavelet Decomposition filters are the most common frequency decomposition methods that are used to extract image spatial structure. These are holistic approaches, where all pixels in the image contribute to each representation.

Section 4.2.1 reviews the various context recognition systems to date. Section 4.2.2 formalizes the framework for ER: extracting the feature vectors of images, training the neural network, and classifying query images. Section 4.2.3 presents the results of the tests conducted on the ER system.

4.2.1 Top-down environment recognition

Torralba [33] presents an excellent paper on environment recognition in which he states that there exist several discriminatory features between scenes and cites three of them:

1. '**The statistics of structural elements.** The second order statistics of natural images are correlated with simple scene attributes and differ between distinct environmental categories.'

2. '**The spatial organization of structural elements.** Structural elements have particular spatial arrangements. Each context imposes certain organization laws. The different organization laws introduce spatial non-stationarities in statistics of low-level features that provide differential signatures between scene categories.'

3. '**Color distribution.** Color histograms and coarse spatial distribution provide discriminant information between scene categories.'

Torralba's features are extracted by filtering images with a bank of oriented bandpass filters (Gabor filters) at six orientations and four scales . The resulting image representation encodes spatially localized structural information in the form of a Feature Vector (FV). The dimensionality of the feature space is reduced by performing PCA and extracting the first 64 eigen vectors. Each scene context is estimated by averaging the FV of 500 images belonging to that context. Torralba

et al. [93] further extend their work by presenting a context-based vision system for place and object recognition. Their goal is to categorize environments and to use that information to improve object recognition. Their FV is extracted via a Wavelet image decomposition. In the first part of the paper, the system is trained on 64 specific places (*i.e.*, office A, office B, lobby C, lobby D, *etc.*) and is then tested by querying it with images of similar locations. In the second part of their tests, the system is queried with images of areas that have not been visited before and asked to determine the image class (*e.g.*, office, corridor, street). Seventeen classes in total are available for the program to choose from. It is of no surprise that the success rate is lower in the second test since many images with different structure can belong to the same class, whereas in the first test very specific structure is sought.

In a similar vein, Oliva and Torralba [94] propose another top-down ER system that is based on a very low dimensional representation of the scene, termed *Spatial Envelope*. Five perceptual dimensions called naturalness, openness, roughness, expansion, and ruggedness are used to develop the spatial representation of a scene. These dimensions are estimated using spectral information that is extracted from images using a Windowed Fourier Transform (WFT). Scenes belonging to the same semantic categories are identified by similar dimensional values. The energy spectrum (Fourier Transform (FT) squared) of images gives the distribution of the signal's energy among the different spatial frequencies. The global and local energy spectrums provide high dimensional representations of the input image. Feature extraction and dimensionality reduction is achieved using the Karhunen-Loeve Transform and the Principal Component Analysis, respectively. Different kinds of environments such as buildings, highways, mountains, forests, *etc.*, exhibit very specific and distinctive power spectrum forms. The Discriminant Spectral Template (DST) is a function that describes how each spectral component of each scene energy spectrum should be weighted in order to determine its perceptual dimension. Similarly, the Windowed Discriminant Spectral Template (WDST) describes how the spectral components at different spatial locations contribute to a spatial envelope property. The DST and WDST are a function of the principal components of the Fourier Transform and the WFT of each image. Their coefficients are learned by training the algorithm on a large set of images that are representative of each scene category.

Tieu and Viola [95] present a paper in which they postulate that images are generated by a sparse set of visual causes and that images that are visually similar share causes. They introduce a method for automatically generating a very large number of *selective features*. A technique known as boosting is then used to learn

a simple classifier that relies on approximately 20 features. As a result, a very large number of images can be scanned in a very short time. After query learning, each image is evaluated by only examining the 20 features. The highly selective features are a natural extension of the simple features that are used in other image processing techniques. For example, first order features such as oriented edges or color are used to construct selective features that measure how the first order features are related. Each level of processing discovers arrangements of features in the previous level. This work is based on previous work by Debonnet and Viola [96]. The process starts by extracting a feature map for twenty five simple features (*e.g.*, oriented edges, center surround, *etc.*). Each feature map is then rectified and down-sampled by two. The process is then repeated for all 25 maps yielding 625 maps. Again this process is repeated to yield $15,625$ feature maps. That number is multiplied by three if colored images are used (Red, Green and Blue). Finally each feature map is summed to yield a single feature value. In mathematical terms the above is expressed as:

$$g_{i,j,k,c} = \sum_{pixels} M_{i,j,k,c}, \tag{4.1}$$

where:

$$\begin{aligned} M_{i,j,k} &= \downarrow_2 (|f_k \otimes M_{i,j}|) && (4.2)\\ M_{i,j} &= \downarrow_2 (|f_j \otimes (M_i)|) && (4.3)\\ M_i &= \downarrow_2 (|f_i \otimes X|), && (4.4) \end{aligned}$$

where $\downarrow_2$ means down sampling by 2, X is the image, f is the primitive operator and $\otimes$ is the convolution operator. AdaBoost [97] is used to combine several weak learners into one strong classifier. The learner is called weak because it is not expected to classify the training data well on its own. The final strong classifier is a weighted average of the weak classifiers. One important point in this method is that it entails knowing what you are looking for (e.g. cars, trees, jets, cloudy skies, waterfalls), finding the most selective features and then convoluting those features with the test images to check if the images contain such objects. It entails classifying scenes (100 images of each) and running AdaBoost 20 times for each scene category to extract 20 selective features for each category. Now these features can be used to determine if a scene belongs to any of the classes. Experimental results show that there is indeed a high correlation between these highly selective features and scenes. Although this system is not implemented in our work, it could be useful as a tool for ascertaining the presence of a specific landmark in the surrounding environment.

All the above approaches are based on representing images in a holistic fashion; where all pixels in the image contribute to each representation. The difference between the approaches resides in the way features are extracted. Torralba [33] performs Gabor filtering, Torralba et al. [93] perform a wavelet transformation on their images, Oliva and Torralba [94] use a discrete Fourier Transform and a Windowed Discrete Fourier Transform. Tieu and Viola [95] apply several low level filters to discover salient features in images.

VisSLAM implements its own top-down system, which is similar in spirit to the work of Torralba [33] since both use a Steerable Pyramid (SP) to extract its Feature Vector (FV), but differs in its learning paradigm; while Torralba uses a Hidden Markov Model (HMM), VisSLAM uses an Artificial Neural Network (ANN) combined with Histogram weighing. The justification for this preference is the higher robustness of the ANN to non-linearities. The details of the VisSLAM ER system are presented in the next section.

4.2.2 ER framework

The framework of the ER system proposed in this work is presented in Figure 4.3 and is interpreted as follows. At a preprocessing stage, a large number of images of various classes (e.g. indoor office, indoor corridor, indoor hall, outdoor street) are collected. A steerable pyramid filter is applied to each of these images in order to extract its corresponding FV, which includes 384 dimensions: 6 orientations times 4 scales times 16 sub-regions per image. Principal Component Analysis (PCA) is then performed on all these FVs to reduce their dimensionality from 384 to 40. The reduced FVs, along with the class of each image are then used to train a 3-layered Feed-Forward Artificial Neural Network. Once the ANN is sufficiently trained the robot can initiate navigation. Equipped with a vision system, the robot acquires images from its surrounding environment and classifies 100 of them using the trained ANN and accumulating the resulting votes in a histogram. The class with the highest votes is picked as the recognized environment. Although the developed technique is applicable to multiple environments types, the scope of this work is limited to classifying an image as an Outdoor Park or not. Once environment classification is achieved, the robot is assigned features that agree with the recognized setting (*i.e.*, tree trunks for a park area). During SLAM, the robot periodically scans its surroundings to insure that it is still located within the same environment.

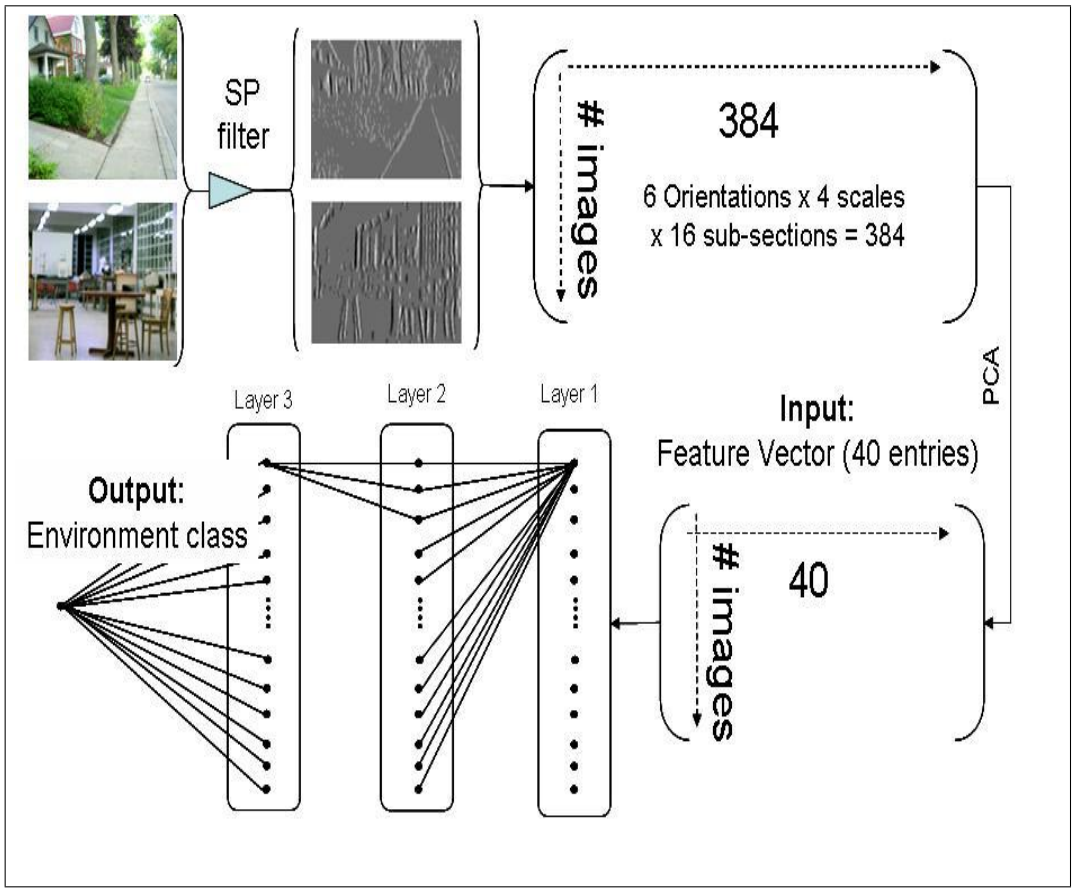


Figure 4.3: Framework for the ER system. Images are filtered with a Steerable Pyramid at 6 orientations and 4 scales. Each filtered image is then averaged by a 4x4 lattice resulting in 16 entries per filtered image. The resulting feature vector contains 384 (4 scales x 6 orientations x 16 entries). The dimensionality of each feature vector is then reduced to 40 via PCA. These entries are then used as inputs to an ANN for scene classification.

Figure 4.4 presents an example of two environment classes with their low-level features extracted via a steerable pyramid. The filtered images are obtained using the code of Simoncelli and Freeman [98].

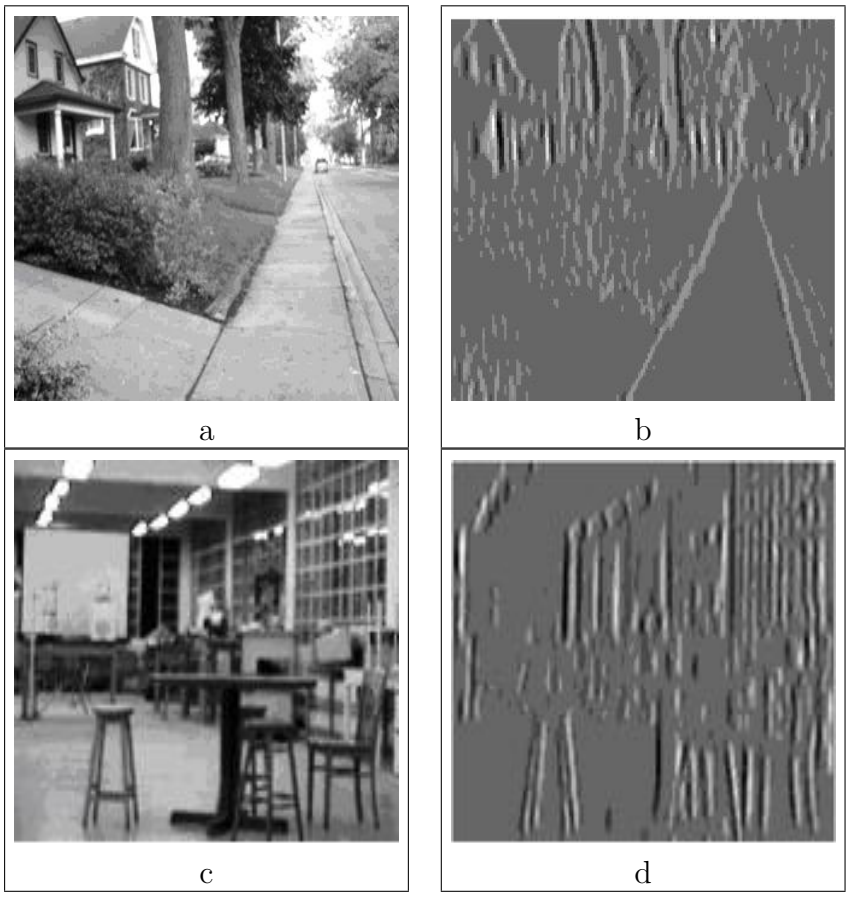


Figure 4.4: Images filtered via the steerable filter code of Simoncelli [98];(a-c) input images, (b-d) filtered images at finest scale and vertical orientation.

4.2.3 Experiments

Methodology

All the ER code is written in Matlab and is implemented on an Intel Pentium 4 processor, 3.2 GHz, 1 GByte RAM workstation. In the preprocessing phase one thousand images of various indoor and outdoor urban environments are collected and labeled by hand according to their environment class. The images are filtered and PCA applied to extract their feature vectors. Testing is performed on batches of 100 images each of three settings: the first batch of images is taken from the same setting as that of the test images, the second batch is taken from a different setting than that of the test images. The ANN assigns a weight between zero and

one to each image based on its class, where 1 is an outdoor park and zero is not. Any weighting between zero and one corresponds to an unsure classification. In this work, a threshold of 0.5 is used to binarize the classification process. The number of votes of the ANN are accumulated in a Histogram and the environment class is assigned to the label of the bin with the highest number of votes.

Results

The computational time for the preprocessing phase is 574s. The feature vectors that are used as training instances for the ANN, trained successfully in 20 epochs or the equivalent duration of 540s. Once the training is complete, the ANN is tested on 100 images from an indoor setting and 100 images from an outdoor urban setting.

The ANN classifies the indoor images with a success rate of 83% (Figure 4.5a) and the outdoor images with a success rate of 80% (Figure 4.5b). A classification is considered 'correct' when the error is less than 0.5. The total time required to extract the feature vectors of 100 images and classify them amounts to $61s$, which corresponds to $0.61s$ per image.

In the second test, images from an outdoor park (rather than urban) setting are queried using the same network as above. The ANN performs poorly and yields a success rate of 51 percent which is almost equivalent to random guessing. These poor results are expected because the ANN is not exposed to any suburban images in its training set. Comparing the urban picture in Figure 4.7a to the park picture in Figure 4.7b, it is intuitive why the above ANN failed. Although both are images of external environments, there is significant difference in the structure of both. In the latter case there are natural objects such as trees and grass that exhibit a distinctive spectral signature which is different than that of buildings and streets, which are common in urban environments. To remedy this setback, the ANN had to be retrained by including images of outdoor park locations in the training data. This retrained ANN yields 81 % correct classification. In the third and final test, a video sequence of an indoor lab at the American University of Beirut (AUB) and another sequence of an outdoor area at the AUB campus is tested on the ANN. The result is an impressive 95 % correct recognition rate for the indoor images and 79% for the outdoor images. Figures 4.6a and 4.6b display the performance of the ANN on these images. It is interesting to note that the variability in the classification error is much lower (*i.e.*, closer to 0) in the indoor than the outdoor environment. The reason for this discrepancy is that an indoor environment is more structured than an outdoor one; a fact that enables the ANN to classify indoor

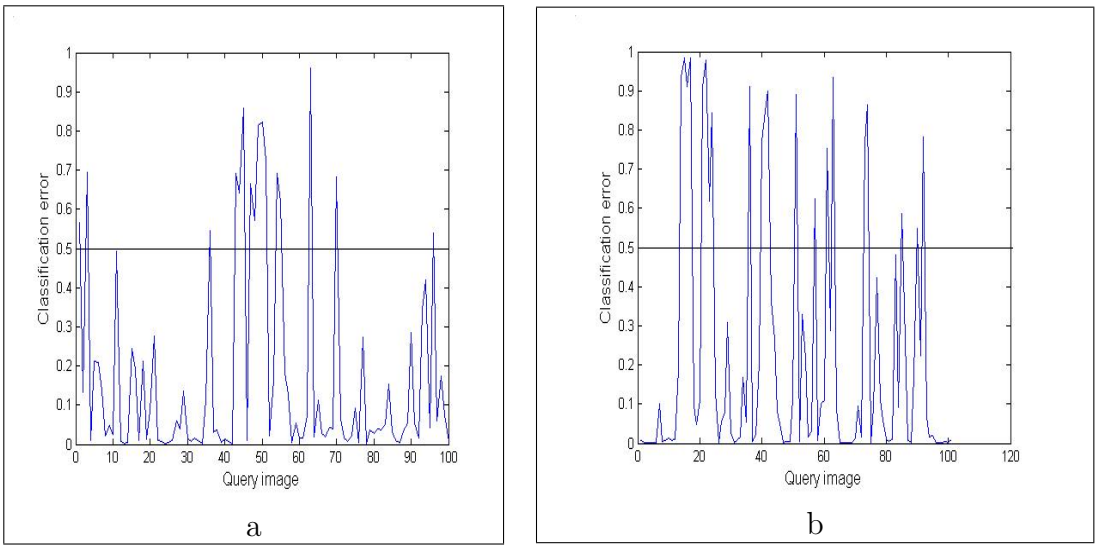


Figure 4.5: Classification performance of the ER system on test images of same setting as the training images on (a) indoor and (b) outdoor images. The x axis represents the image number and the y axis is the classification error, calculated as the difference between the true class and the one predicted via the ANN. Any classification with an error below 0.5 is considered correct. The system results in 83% correct classification for indoor images and 80% for outdoor images.

settings with more certainty than outdoor ones. In any case, classification results for both environments are acceptable and are indicative of the strong inductive bias of the ANN toward correct classification, which allows for good generalization of the recognition system.

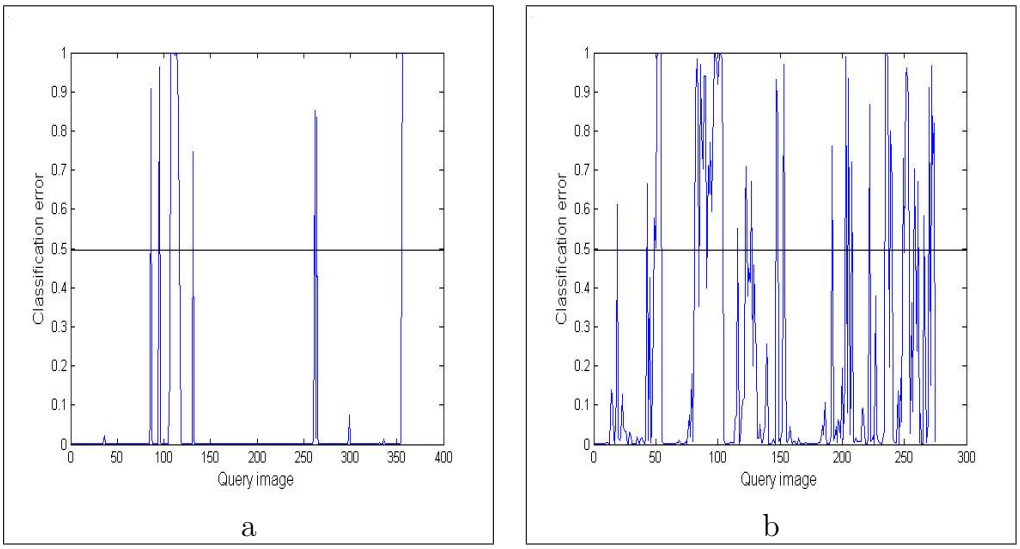


Figure 4.6: Classification performance of the ER system on images taken from the AUB campus on (a) indoor and (b) outdoor images. The x axis represents the image number and the y axis is the classification error, calculated as the difference between the true class and the one predicted via the ANN. Any classification with an error below 0.5 is considered correct. The system results in 95% correct classification for indoor images and 79% for outdoor images.

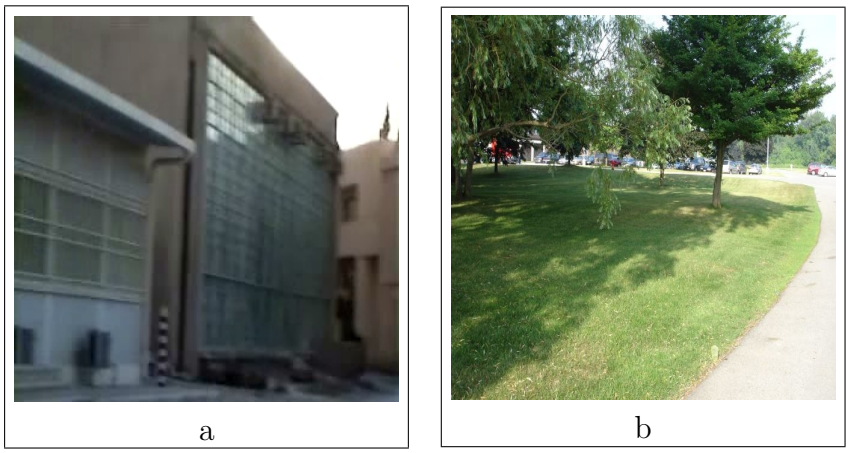


Figure 4.7: (a) Sample urban and (b) park images.

4.2.4 Can stereo help environment recognition?

It is argued here that stereo vision can aid environment recognition by adding a prior probability that depends on the clutter and dimension of the area surrounding the camera. For instance, in an indoor office, the stereo camera detects objects that are relatively close and most of the regions surrounding the camera are within the range of the stereo system. On the other hand, in an outdoor open area, most of the camera surrounding is probably out of the range of the stereo camera, which detects very few objects in its vicinity. Stereo vision for environment recognition is not investigated in this work but constitutes a open problem and an interesting field for future research.

4.3 Proposed tree detection techniques

Tree detection is a new problem, which to the author's knowledge has received very little attention in the Computer Vision community. Some recent work is proposed by Ramos *et al.* [99] who use statistical representations for natural objects but the resulting segmentations are too coarse to serve for object detection. Object detection systems can be grouped according to the following taxonomy: Local-Image-Descriptor (LID)-based object detection or SEGmentation (SEG)-based object detection. In the first technique [59, 33, 100, 101, 102, 103], distinctive and salient image features [60, 61, 62, 63, 64, 65, 102, 104] are extracted from images at a pre-processing phase and archived in a database. Each of these features is represented by its corresponding feature vector. Objects are then modeled by a combination of these local features that are arranged in a specific geometric configuration. Query images are classified by matching their feature vectors to the nearest ones in feature space. David Lowe [59, 104] introduced the Scale Invariant Feature Transform (SIFT) descriptor. A 16x16 grid is constructed around every interest point. Each entry into this grid represents the orientation of a Difference of Gaussian (DoG) map, taken at the scale of the interest point. The grid is then grouped into 4x4 sub-grids, where each sub-grid element represents a histogram of orientations of the original grid elements inside each sub-grid. Other descriptors include steerable filters, differential invariants, complex filters, and moment invariants. Mikolajczyk and Schmid [68] compare the performance of the aforementioned descriptors and find that SIFT descriptors perform best, followed by steerable filters. Although SIFT descriptors are very distinctive, it is still possible to find similar SIFT features between two different objects or between an object and its surrounding background, thereby resulting in false positives. One method

to improve matching is proposed by Lowe [104], where a triplet of descriptors are associated with each object part and query objects are matched to model objects only if three of its descriptors match up with a corresponding triplet in the model database. The probability of a mismatch with three descriptors between a query object and those of an image in a database is much lower than that of a mismatch with one descriptor. In a similar vein, Dorkó and Schmid [101] suggest representing objects by clusters of descriptors, where objects are represented by a Gaussian Mixture Model (GMM) and each Gaussian represents a part of the object. The mean, covariance, number of components, and prior probabilities of the GMM components are learned in a weakly supervised fashion — images that contain query objects are labeled positive but individual objects are not labeled. Learning of the mixture components is achieved using Expectation-Maximization (EM). The clusters of Gaussians are then ranked using classification likelihood and mutual information. While the former ranking method is a measure of classification efficiency, the latter is good for sparse representation and focus of attention scenarios. During testing, candidate images are first processed to extract their local descriptors, which are then paired to the closest Gaussian (or part) using Maximum à Posteriori (MAP). If enough of the query object descriptors are matched to the n top-ranked clusters of the model object then it is safe to conclude that the object is present in the image. A natural extension to this work is the addition of spatial constraints between the ranked parts to improve the recognition rate and reduce false positives.

Unfortunately, although LID-based techniques are fast, they require that the sought object be comprised of features which can be repeatedly detected and matched and that these features be organized similarly across all objects. These requirement rules out using LID-based object detection systems for detecting natural features such as tree trunks. The only commonality between tree trunks is that all of them stem from the ground and that their structure is quasi-vertical and quasi-symmetric. None of the internal features of one tree trunk are likely to be found on another trunk. One possible ramification of these methods is to use local descriptors to locate all salient regions inside a query image and then to determine which of these salient features are tree trunks and which are not. Alternatively, IPs can be used for recognition rather than detection. Once an object is detected, the regions bounded by the perimeter of the object in question can be searched for distinctive features (such as SIFT) which can then be used to differentiate between 2 objects (*e.g.*, Tree A verses Tree B). This method is further investigated in Section 4.4.

In segmentation-based object detection the query image is segmented before

attempting to detect an object within it. Image segmentation returns the most probable interpretation of the content of the primal image. Ideally, segmentation seeks to group pixels that belong to same objects together. The exact details of the segmentation process is dependent on the type of object that is sought and consists of extracting a useful object representation for the detection system. The disadvantage of such systems is that they are slow because of their high processing requirements. Nevertheless, if the sought objects are simply structured, such as tree trunks, it is possible to model them by a few parameters such as the position of their center, their diameter and their orientation. Furthermore, full 3D reconstruction of a landmark is not necessary, since SLAM only uses this feature as a point landmark and is not interested in the exact shape of its profile. Under such simplifying conditions, the implementation of such detection techniques can approach real-time barriers. The issue that remains to be solved is the nature of the segmentation process that yields the most informative and useful representation for detecting tree trunks. The choice of possible segmentation methods is either appearance-based or structure-based. In the former method images are regarded as arrays of pixels rather than a picture of objects and pixels sharing similar properties (*e.g.*, brightness, color, texture) are combined together. The latter segmentation method consists of searching the image for lines, ellipses, and other geometries, which are later combined into candidate objects and compared to model objects. In Figure 4.8, three images are shown that are typical examples of the type of images in which VisSLAM seeks to detect tree trunks. The images are difficult to segment because of the varying brightness patterns across the trees and because of the similarities in texture between trees in the foreground and background clutter.

Img 181

Img 185

Img 186

Figure 4.8: Challenging segmentation problems. (a) Tree trunks in the background causes merging of foreground and background regions; (b) The textured regions inside the tree trunk in front creates line artifacts which could be falsely labeled as separations between two different objects; (c) Hanging foliage of the tree triggers line artifacts around the tree.

4.3.1 SEG-based tree detection (appearance-based)

Segmentation-based object detection involves segmenting the image into a representation that is useful for the subsequent object detection task. In this section two appearance-based segmentation methods are attempted: the Brightness-Texture clustering method and Statistical Region Merging (SRM) [105]; both methods are tested on a database of images collected during an experimental run in an outdoor park environment (See Appendix A for a description of the experiments).

Integrating brightness and texture

This segmentation procedure consists of grouping into regions pixels which exhibit similar brightness, texture, or a combination of both. Brightness is represented by the gray-scale intensity values of the pixels inside the query image. The brightness of a 3D object in an image is a result of the illumination of the 3D object and its reflectance properties. An object that is illuminated in an inhomogeneous fashion exhibits inhomogeneous brightness patterns even though its reflectance properties across its entire surface might be the same. It is therefore desirable to remove the illumination component from the calculated values of brightness and express brightness as a function of the reflectance properties of the body alone. Once illumination variation is accounted for, it is then possible to segment objects by

clustering pixels exhibiting similar brightness values. In gray scale images, the effects of changing lighting conditions are reduced by histogram equalizing each image before it is processed. Histogram equalization employs a monotonic, non-linear mapping which re-assigns the intensity values of pixels in the input image such that the output image contains a uniform distribution of intensities (*i.e.*, a flat histogram).

Object reflection is dependent on the surface property of the object and is expressed as the Bidirectional Reflectance Distribution Function (BRDF)[1] [106]. Segmentation is done by clustering pixels into one of four bins (several numbers are tested and 4 produces the best results for these images) based on their intensity values. The result of Brightness segmentation on three images containing trees is reported in the middle images of Figure 4.9. Trees that are located far away are segmented well, whereas close-by trees are badly segmented. In the close-by trees, the textures patterns are highly visible and the tree trunks exhibit inhomogeneous BRDF across the tree surface, thereby causing inconsistent segmentation. The textured regions on the far away tree trunks are less pronounced; therefore, the tree surfaces exhibit homogenous BRDF and reflect similarly.

Texture is an indication of the spatial arrangement of patterns inside the image. Studies in psychology have shown that filtering an image with a Gabor filter bank produces responses similar to the ones sensed by V1 cortical cells [107]. The filter bank consists of four scales and six orientations and a nominal texture value is calculated at each pixel site by simply adding the response at all scales and orientations. Pixels exhibiting similar texture response are grouped together. The result of texture segmentation on three images is reported in the right images of Figure 4.9. In texture based segmentation, close-by regions are better segmented than in Brightness-based segmentation since similarly textured regions are grouped together. The problem here occurs when background regions exhibit similar texture regions as the trees in the foreground, subsequently causing the segmentation process to merge trees with background.

Given the limitations of the previous approaches it is proposed to combine the evidence from both brightness and texture to yield a better quality segmentation.

$$Seg = w_b * Seg_b + w_t * Seg_t, \tag{4.5}$$

[1]Forsyth and Ponce [106] define the BRDF as: 'The ration of the radiance in the outgoing direction to the incident irradiance'

$$\rho = \frac{L_o}{L_i}$$

where ρ is the BRDF, L_i is the incident irradiance, and L_o is the outgoing radiance.

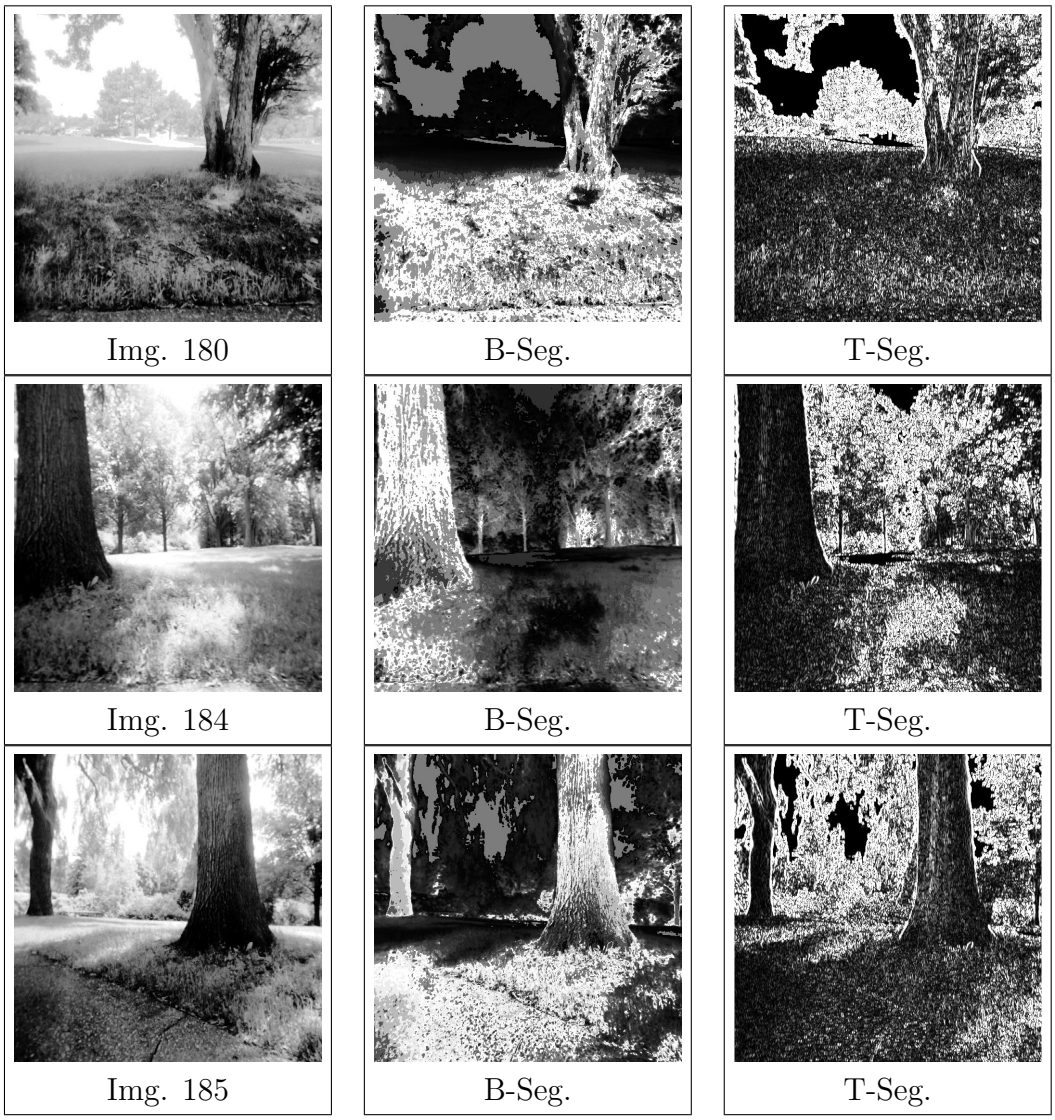


Figure 4.9: Result of performing Brightness and Texture segmentations on images 180, 184, and 185 of our first database of tree images.

where Seg is the combined response at each pixel which is used to ultimately segment the image. Seg_b and Seg_t are the responses obtained from the brightness and texture respectively, w_b and w_t is the weighting the user chooses for combining brightness and texture respectively ($w_b = 1 - w_t$). Unfortunately, this additive effort does not improve segmentation results. An attempt is also made to segment using color (*i.e.*, Hue component from the Hue, Saturation, and Intensity (HSI)

space) and texture on color images taken from the same test site. Results do not improve over those obtained in Figure 4.9.

Statistical region merging

Statistical Region Merging (SRM) is an appearance based segmentation method developed by Nock and Nielsen [105] which segments images via region merging, following a particular order in the choice of regions. SRM works by creating what the authors refer to as *"theoretical image"* $I*$ in which pixels are referred to as *"statistical pixels"* and are represented by a family of distributions, from which the observed image is sampled. Pixels that share similar statistical expectations in the theoretical image are grouped together. Different regions exhibit different expectations for their statistical pixels. In each pixel of $I*$, each color channel is replaced by a set of exactly Q independent random variables, taking positive values on domains bounded by g/Q (g is the dimension of the color space *e.g.*, 0 to 255) such that any possible sum of outcomes of these Q random variables belongs to $1, 2, ...g$. Tuning Q modifies the statistical complexity of the scene, and makes it possible to tune the coarseness of the segmentation. SRM is implemented as follows. First, a list S_I is compiled of all possible couples of pixels (4-connexity) of the image. The couples are sorted in increasing order of $f(p, p')$, where f can be any real-valued function (described below) and p and p' are pixels of I. A test $P(R(p), R(p'))$ of compatibility is made between the regions $R()$ to which p and p' belong to. P is the merging predicate of SRM and is the key to its success. For gray-scale images

$$P(R, R') = \begin{cases} true & \text{if } |\bar{R}' - \bar{R}| \leq \sqrt{b^2(R) + b^2(R')} \\ false & otherwise \end{cases} \tag{4.6}$$

where $\bar{R}$ is the average intensity value of the region R, $|.|$ stands for cardinal, $b(R) = g\sqrt{(1/(2Q|R|))\ln(|R|/\delta)}$, δ is a value between 0 and 1, and g is equal to 256.

and the same applies for color images

$$P(R, R') = \begin{cases} true & \text{if } \forall a \in \{R, G, B\}, \\ & |\bar{R}'_a - R_a| \bar{\leq} \sqrt{b^2(R) + b^2(R')} \\ false & otherwise \end{cases} \tag{4.7}$$

If $P(R(p), R(p'))$ returns true the regions R(p) and R(p') are merged. In this SRM implementation f implements radix sorting based on color differences.

$$f_a(p, p') = |p_a - p'_a| \tag{4.8}$$

SRM can be applied on colored (*i.e.*, 3 channels) or gray (*i.e.*, 1 channel) images. The segmentation is slightly better for colored images but since color is not available in our images, SRM is implemented based on gray level information. The implication of using only one channel versus three is a coarser segmentation. This is confirmed by testing SRM on several colored images obtained from the test site. SRM requires the user to specify the segmentation resolution, which is a problem in itself for tree detection. If the resolution is set too low, large tree trunks are segmented as one entity but small tree trunks are not detected. On the other hand, if the resolution is set high, small tree trunks are detected but the larger ones are partitioned at highly textured locations. In the results presented below in Figure 4.10, an intermediate scale is used that compromises between large and small tree segmentation requirements. Unfortunately, the results are unsatisfactory, where segmentation errors occur similar to those in the Brightness-Texture segmentation described above. The middle images in Figure 4.10 show the results obtained by segmenting trees using SRM-Brightness. Notice the overlapping regions between trees and background.

Nock and Nielsen [105] also suggest performing SRM on different color spaces such as the Hue, Saturation, Intensity (HSI) color space. This work attempts SRM on a textured image in order to cluster homogenous texture patches together. The results are presented in the right images of Figure 4.10. Again,overlapping regions are observable in the segmentation results of all three tree images.One observation worth mentioning is that the SRM-Brightness segmentation performs better on trees which are far away than SRM-Texture and the contrary occurs for trees that are close by. This is an expected result since close by trees exhibit similar texture values but different brightness values, whereas in far away trees the texture of the trees and background foliage is comparable and gets merged together.

Both appearance-based segmentation methods presented in this section are lacking. In some instances, tree regions are mixed with the background (*e.g.*, the 2 attached trees in Image 180 of Figure 4.9); in other instances a single tree is divided into two parts (*e.g.*, the large tree in the left of Image 184 of Figure 4.10). Furthermore, in both SRM and Brightness-Texture segmentations the proposed regions are coarse and small tree trunks are not detected. In spite of these shortcomings, these segmentations can be used as a starting point for a system that further processes them in order to produce more precise results. Tu *et al.* [108] integrate top-down and bottom-up segmentation techniques to recursively reconfigure segmentations via reversible Markov Chain Jumps [109]. Unfortunately, the

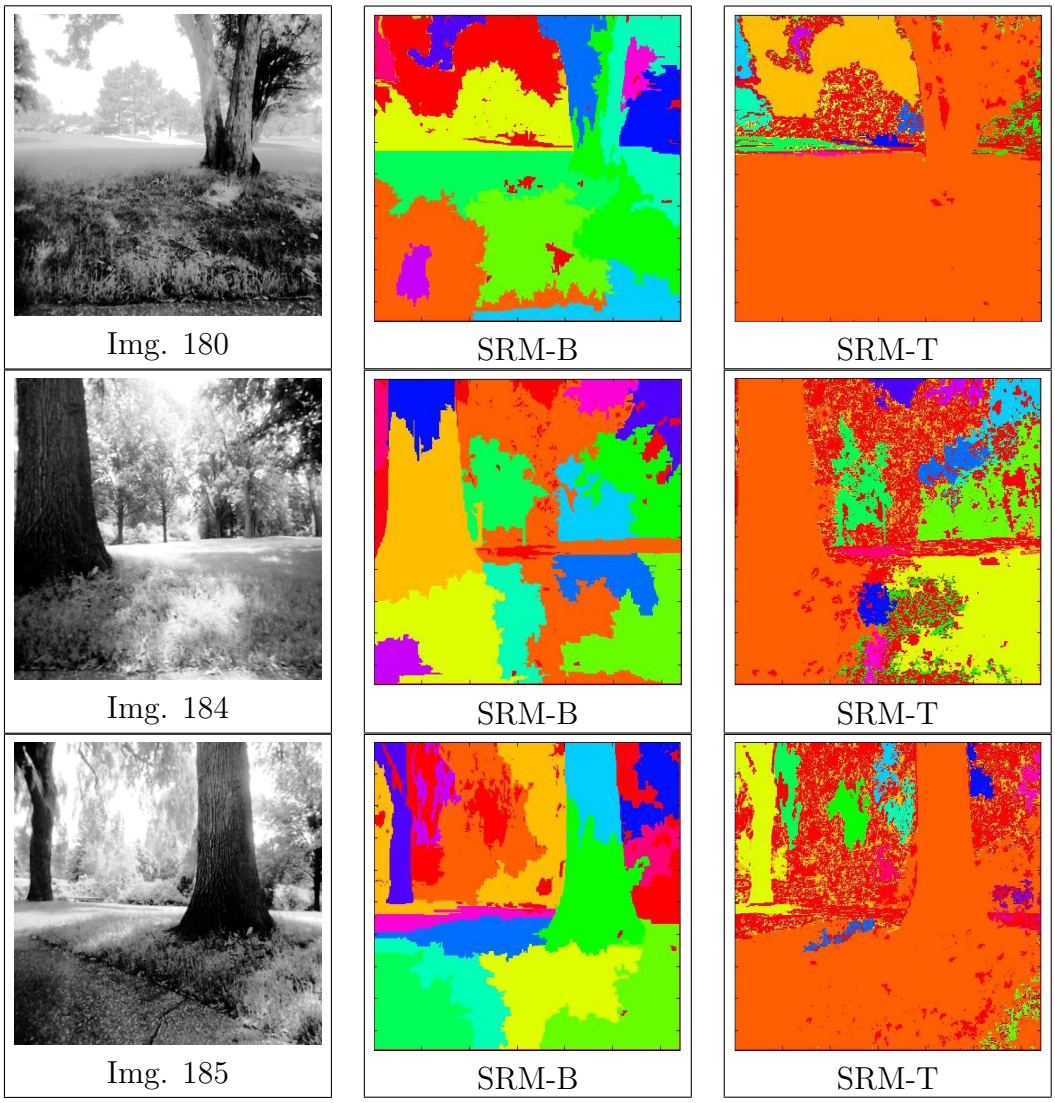


Figure 4.10: Statistical Region Merging (SRM) segmentation. (left) Original gray images showing trees that are close to the camera and others that are far. (center) SRM using brightness, notice that the far away trees are nicely segmented whereas the close ones are not. (right) SRM using texture, this segmentation performs well on trees that are close but poorly on far away trees.

run time of such methods is too high to be used for SLAM, which requires object detection in near-realtime.

Can stereo help appearance-based segmentation?

Depth information from stereo vision can indeed help in correcting and refining both appearance-based techniques. Figure 4.11 proposes a method by which depth and height maps inside the segmented regions can be used to refine an appearance-based segmentation. First, SRM is applied to the query image to segment it into several regions. The height map of the same scene, extracted via stereopsis is used to further refine the segmentation into ground, sky, and interesting regions 'A' and 'B. Depth histograms are then computed for 'A' and 'B. Note that the stereo camera used for VisSLAM (Appendix A.1) returns a value of 0.204 meters as a depth value when there are inconsistencies in a disparity at a certain point in the image. These values are reflected by the peaks near zero that are evident in both the histograms. If a segmented region represents an object such as a tree, it is expected that the depth map in this region have very little variations and without any independent pixel clusters (histogram B). If on the other hand, there is a cluster of pixels at one depth (histogram A) and the remaining pixels are distributed at other depths, the depth map suggests that the original segmentation is not precise. Depth information can then be used to refine the original SRM segmentation. One would argue that only depth should be used for segmentation and not bother with the initial segmentation. The problem is that depth segmentation is coarse and would lead to many discontinuities in the segmentation.

Although this depth-based method is appealing, the problem resides in segmenting trees that are not in the range of the stereo camera, where the system would have to rely on SRM alone and segmentation errors can not be corrected. In the next section an alternative segmentation method is proposed, which is a structure-based system, and detects trees by perceptually organizing lines into symmetric and continuous lines and grouping these lines into trees.

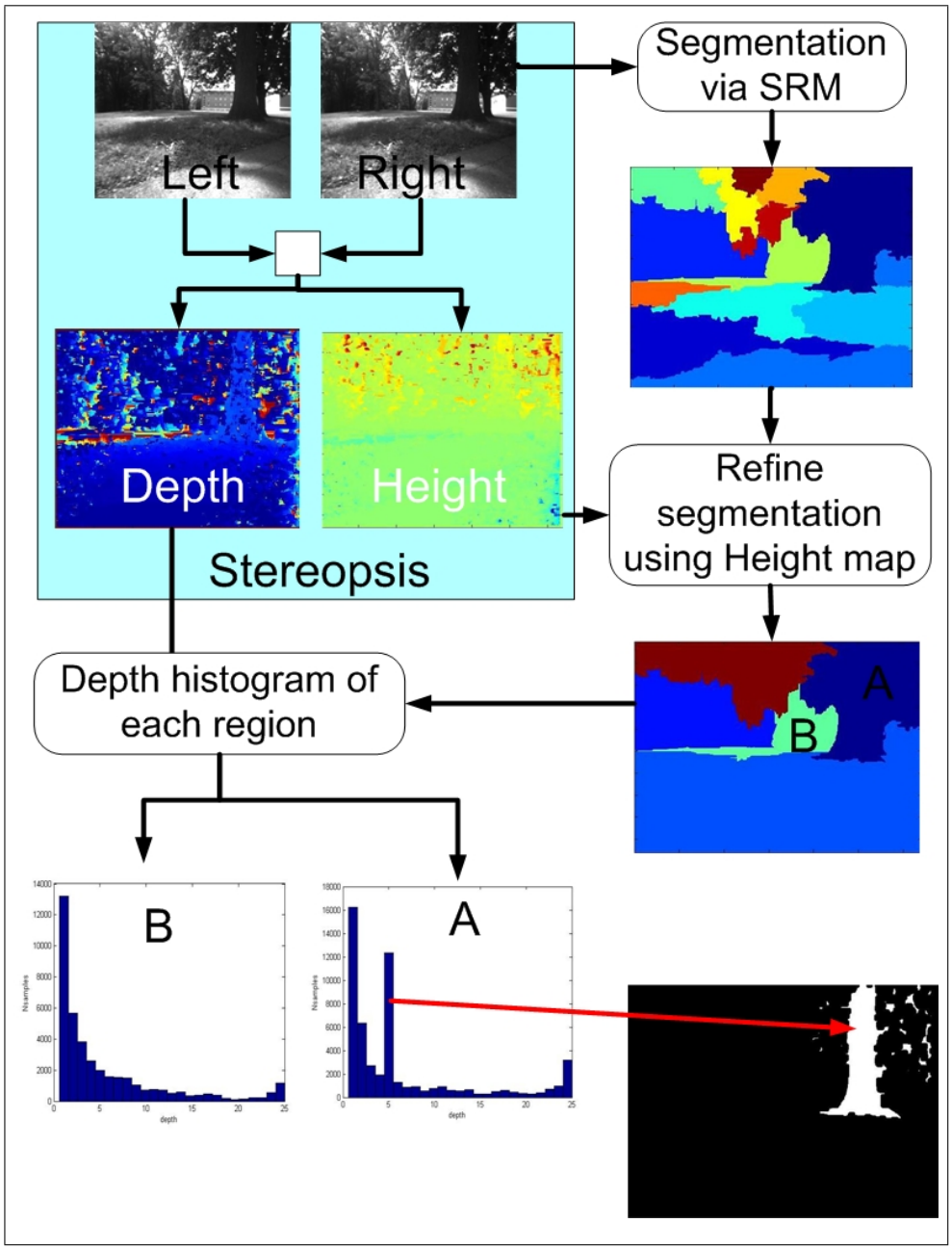


Figure 4.11: Stereo aiding appearance-based segmentation.

4.3.2 SEG-based tree detection (structure-based)

Steve Nix [91] defines a tree as follows:

- 'A tree is a woody plant with a single erect perennial trunk at least 3 inches in Diameter at Breast Height (DBH).'

  One could add the following definitions:

- 'The profile of a tree is constructed by quasi-symmetric lines that are approximately vertical.'
- 'The base of a tree trunk is always connected to ground.'
- 'The aspect ratio of a tree trunk is generally larger than 2.'

These definitions are used as guidelines for generating an efficient tree detection system. First, edges dominant in the vertical direction are sought by filtering the input image with an edge detection routine that is sensitive to vertical edges. Next, the dominant edges are grouped into continuous and symmetric lines, which are subsequently grouped into trees based on two criteria: entropy reduction and location of the tree trunk base. A Ground-Sky tracking system is used to yield a strong prior on the position of the tree trunk base. The details of each system are presented in the following sections.

Figure 4.12 shows the different parts of a tree. It is the 'TRUNK' that the vision system is designed to detect.

Edge detection

The first step in this process involves detecting edges which are dominant in the vertical direction. The justification for searching for vertical lines is that the camera remains horizontal to the ground plane during all experimental runs and since trees are quasi-vertical structures they would always appear quasi-vertical in images taken by the camera. This is a valid assumption since the ground upon which the vehicle navigates is relatively flat and smooth. Alternatively, in off-road conditions, one could take advantage of the tilt and bank information obtained from the onboard IMU to determine the relative orientation of a tree to the camera and search for structures in that orientation.

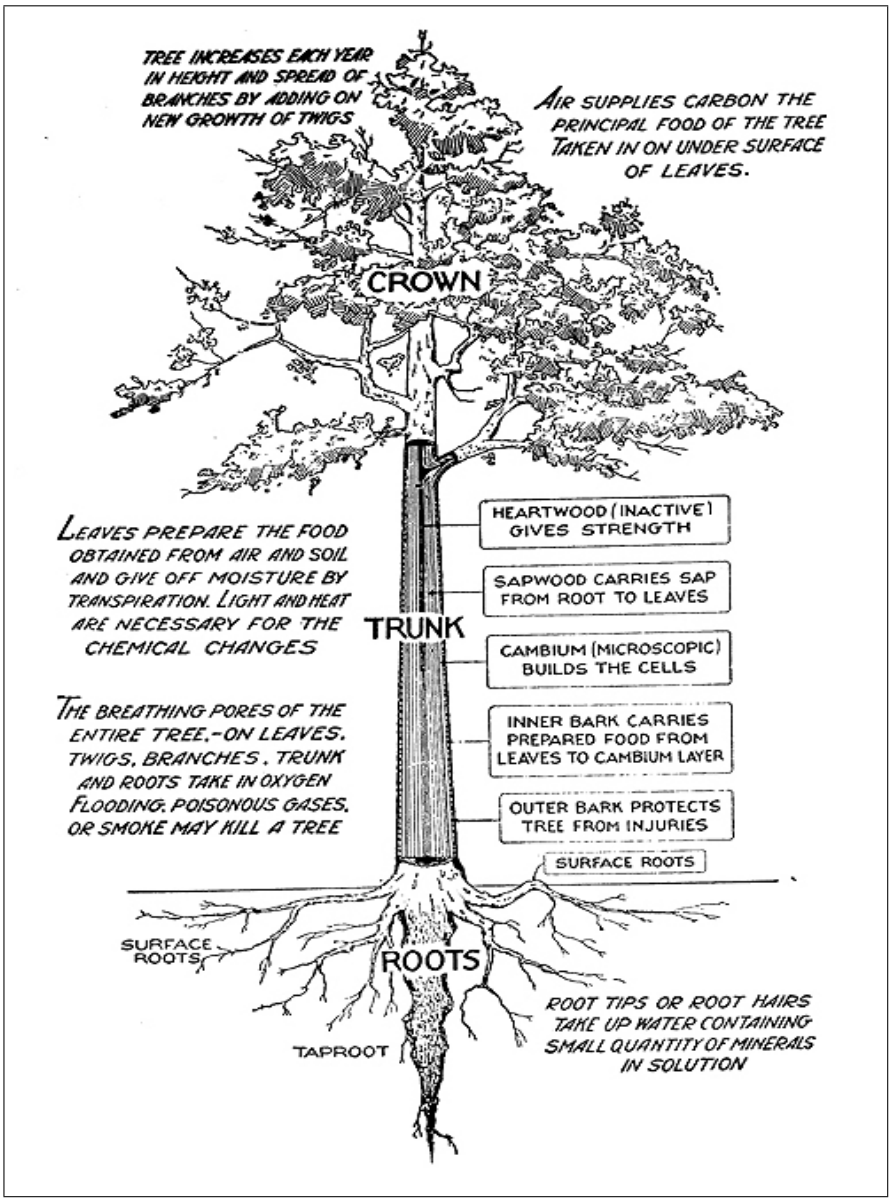


Figure 4.12: Tree parts including the roots, trunk and crown. The vision system is interested in detecting the trunk of each tree. Image courtesy of Steve Nix [91].

In the traditional Canny edge detector [66], lines are tracked in a random direction before any non-max suppression is applied. After applying a Gaussian filter to the image, it is convolved with a gradient kernel in the horizontal, vertical and diagonal directions.

In the first stage of VisSLAM, images are processed by a Canny edge detector tuned for vertical sensitivity. This is achieved by simply discounting the effect of the horizontal gradient from the sum of gradients before non-max suppression and hysteresis thresholding is performed. Discounting horizontal gradients results in giving more weight to vertical structures and make it simpler to segment them from background clutter. In Figure 4.13 the red circles overlaid on top of the image show location where the traditional Canny erroneously follows a horizontal edge rather than a vertical one. In the modified Canny the vertical edges are given more weight, which results in a better trace of the tree profile. For the sake of clarity edge tracking and linking is also included in these images. Following edge detection, non-max suppression is performed as in the traditional Canny, followed by hysteresis thresholding.

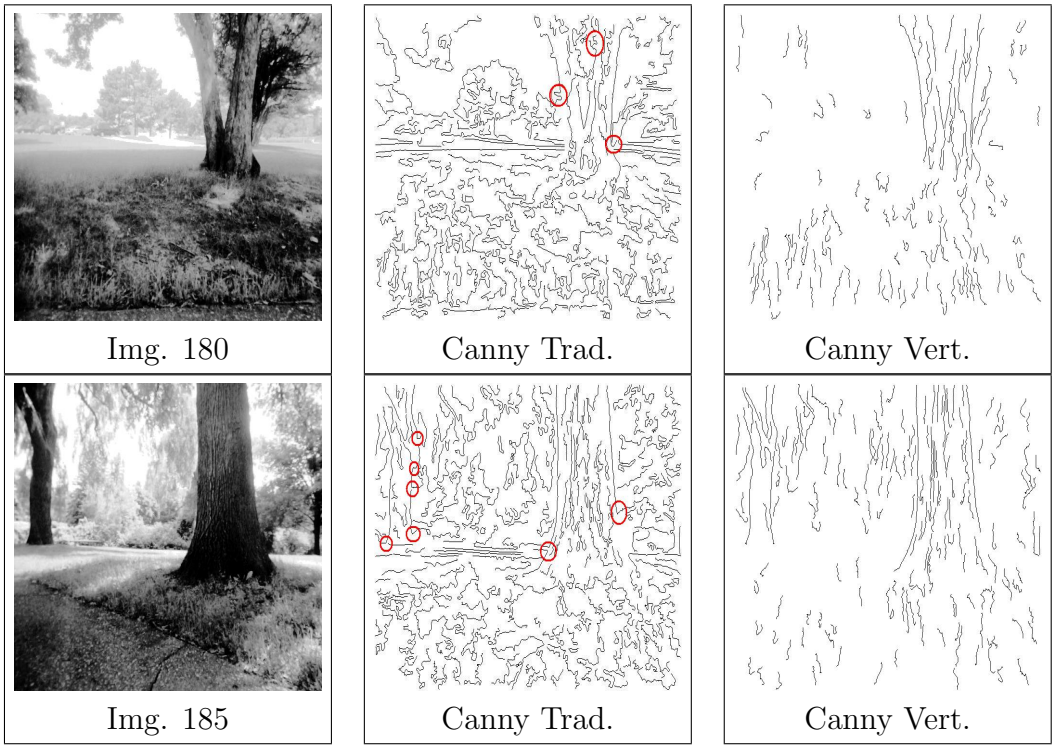


Figure 4.13: Applying the traditional Canny and one sensitive to vertical lines. The images also include a line following system for clarity purposes.

Line tracking and refining

The output of the edge detection system is fed to a line tracking system which uses an eight point neighborhood scheme to build lines in a recursive fashion. At junctions the system tracks all lines of the junction and selects the longest of the possible lines in a greedy fashion (Figure 4.14).

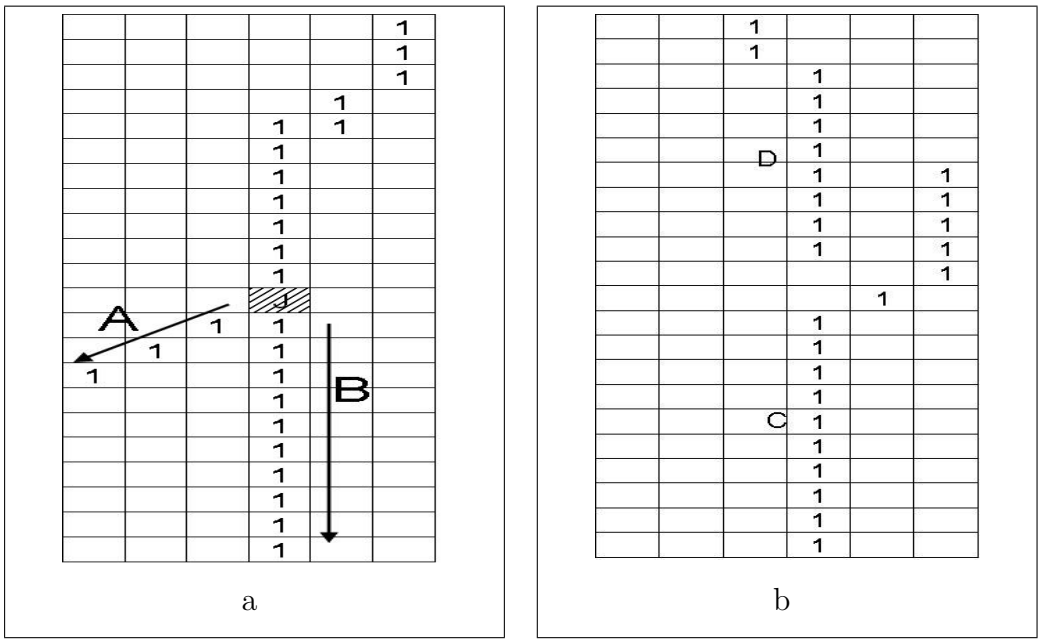


Figure 4.14: (a) At junctions the system tracks all possible alternatives and chooses the longest line (line B); (b) Check line neighborhoods at their extremities, if line D continues line C at their intersection is better than the original line D, join C and D at their intersection and trim line D.

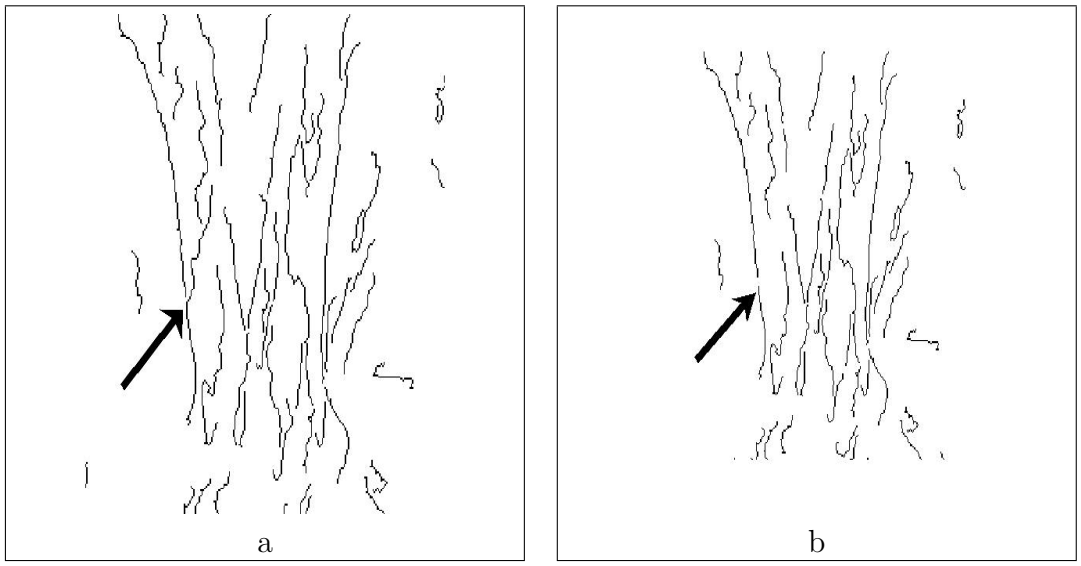


Figure 4.15: Trimming bad lines at intersections. (a) Arrow shows location of bad line direction; (b) Algorithm trims bottom lines at intersection and merges the two lines at their intersection, thereby trimming the first line.

By following this greedy approach noisy pixels that tend to deviate the tree profile in the wrong direction are eliminated.

Line pruning

Of the remaining lines, a number of them are pruned according to two heuristics: minimal length and average texture response. Firstly, all lines which are smaller than a minimal length are pruned. Experiments run on a database of tree images, indicate that lines less than 15 pixels long can be safely pruned without loosing any significant lines. 15 pixels corresponds to a tree 1 meter high located approximately 34 meters away from the camera. Secondly, since it is quasi-vertical lines we are interested in, the images are filtered with a Gabor filter, using the coefficients sensitive to the vertical orientation at the finest scale. Then, the average texture value (4.9) for each of the tracked lines above is calculated and those below a threshold h are discarded.

$$\overline{T} = \frac{1}{Ni} \sum_{i}^{Ni} T(f, v) \tag{4.9}$$

where i is a point of the queried line, Ni is the total number of points of that line, $\overline{T}$ is the average texture value of the line, and $T(s, or$ is the texture value of the queried point i at the finest scale f and vertical orientation v.

The idea is that tree profiles exhibit significantly larger average texture values in the vertical direction than those of lines generated by bark texture. h is found empirically by testing the system on data sets 'one', 'two' and 'three' of the database of experimental images (Appendix A.3). A value of h equal to 0.8 yields the best compromise between detected tree profile and spurious lines.

Figure 4.16 shows the effectiveness that texture pruning has on eliminating insignificant lines. Notice how the lines representing grass are eliminated and those inside the tree trunk are reduced.

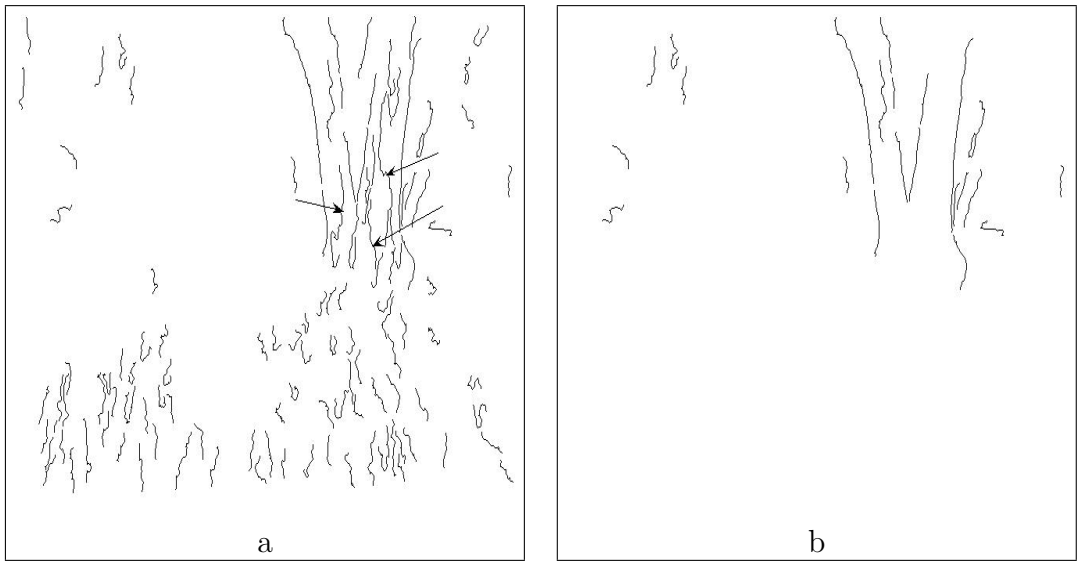


Figure 4.16: Effect of applying pruning based on the texture response. (a) Before texture pruning; (b) after texture pruning.

Once all lines are tracked, repaired, and bad lines are pruned, the system is ready to group lines based on Continuity and Symmetry constraints.

Continuity and symmetry

The procedure here is inspired by the Gestalt laws in psychology [107] to design a system that detects trees. Based on the theory of perception, the Gestalt principles

were developed in the nineteenth century by a psychologist who believed that whole images are often perceived as more than the sum of their parts. The Gestalt laws include nine rules that are stated as follows:

1. Continuity
2. Symmetry
3. Proximity
4. Similarity
5. Closure
6. Surroundedness
7. Smallness/area
8. Figure/ground
9. Pragnanz

All the lines that are obtained after line tracking and pruning are grouped based on four of the Gestalt laws; namely, Continuity, Symmetry, Similarity, and Proximity. First, all detected lines are compared for continuity. The developed equation for Continuity and Symmetry are inspired by the work of Mohan and Nevatia [110], although the equations proposed here include modification that are found necessary to insure better grouping. Continuity between two lines is proportional to the proximity between the extremities of the endpoints, to the difference in orientation (similarity) between the lines and inversely proportional to the length of the continuing line as detailed in (4.10)

$$Continuity = \frac{\sqrt{\alpha^2 + \beta^2} * (1 + 0.9 * \delta_s)}{\sqrt{L_D / L_S}}, \tag{4.10}$$

where *Continuity* is the continuity strength of between two lines, α is the change in orientation the source line takes to join the destination line, β is the change in orientation the destination line makes to join the source line,δ_s is the value of the spacing between the two source and destination line, L_{so} and L_D are the lengths of the source and destination lines respectively (Figure 4.17).

Equation (4.10) indeed succeeds to represent the continuity affinity between lines since α and β indicate the compatibility of the end of a source line to the beginning

of a destination line. If any of these two angles is large the continuity value increases and lines have low affinity. Furthermore, even if both lines are compatible matches based on orientation but are far away, then they are not matched.

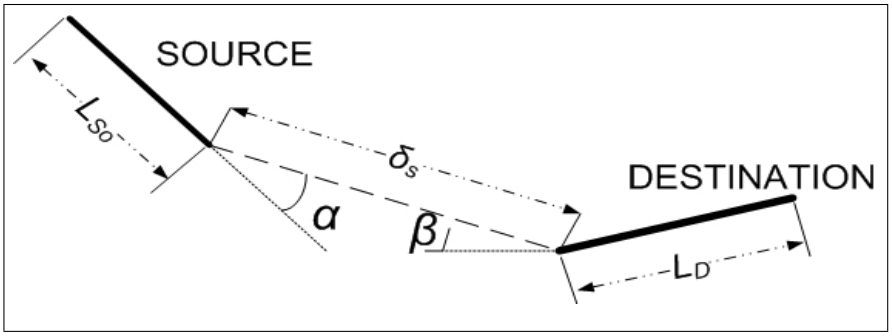


Figure 4.17: Variables for continuity.

Lines with α, β, δ_s above given thresholds are pruned to ensure that the source lines only connect to destination lines which are in front of them and not behind them or too far away from them. These thresholds are found empirically by running a series of experiments on the first three data sets of the experimental database. Values of ± 20 degrees for α and β, and 30 pixels for δ_s yielded the most consistent results across the database of tree images. The minimum Continuity value is preserved for each line and lines which are with 1/3 of this value are preserved as potential continuous lines.

A measure of symmetry is also used to determine the symmetry strength between two lines (4.11).

$$S = 5AR - 20\delta_\theta - 20\delta_l - \delta_e - 3\delta_{sk} + L_s - \delta_{al}, \tag{4.11}$$

where AR is the aspect ratio between the two lines, $\delta\theta$ is the difference in orientation between them in rad, δ_l is the normalized difference in their length, δ_e is the difference in the orientation of their ends in rad, δ_{sk} is the difference in their maximum skewness, L_s is the length of the symmetry line, δ_{al} is the difference in the alignment between their ends (Figure 4.18).

The weights in (4.11) are found to produce the most consistent results in the database of tree images. A better approach would be to learn the weights using a machine learning algorithm such as Expectation Maximization (EM). It is worthwhile to note that the calculation of the symmetry and continuity is very fast due to

the fact that only the extremities of the lines are used to calculate all the variables in (4.10) and (4.11). The symmetry line is the line joining the midpoints of the lines joining the top and bottom extremities respectively.

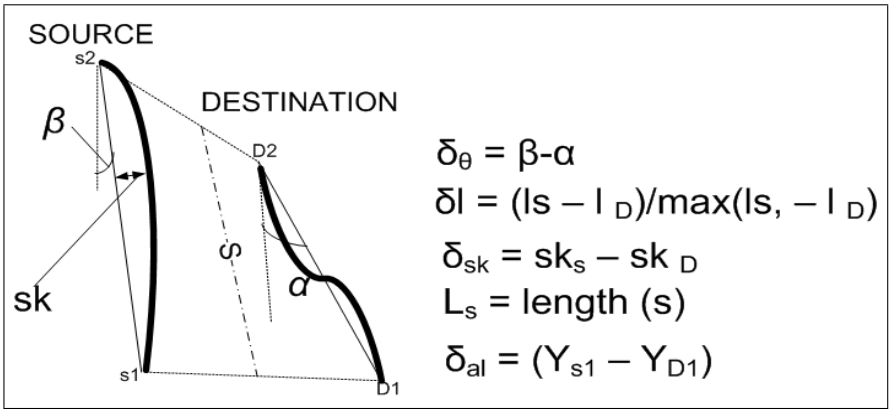


Figure 4.18: Variables for symmetry.

Testing this equation on a pair of lines that are close and parallel produces a change in orientation δ_θ equal to zero; therefore resulting in a high affinity between the two parallel lines. On the other hand, if two perpendicular lines are tested for grouping, the difference in their orientation results in a large negative number $(40 * Pi)$ subtracted from the Symmetry value, and the tow perpendicular lines have a very lwo affinity for each other.

Ground-sky line tracking

In addition to their quasi-vertical shapes, tree trunks are also constrained to be attached to ground. A tree is either located far away and the base of its trunk is adjacent to the Ground-Sky (G-S) separation line, or it is located close by and its trunk intersects G-S. Figure 4.19 shows the two possible configurations of tree trunks. It is not possible for the base of the tree trunk to be located higher than G-S.

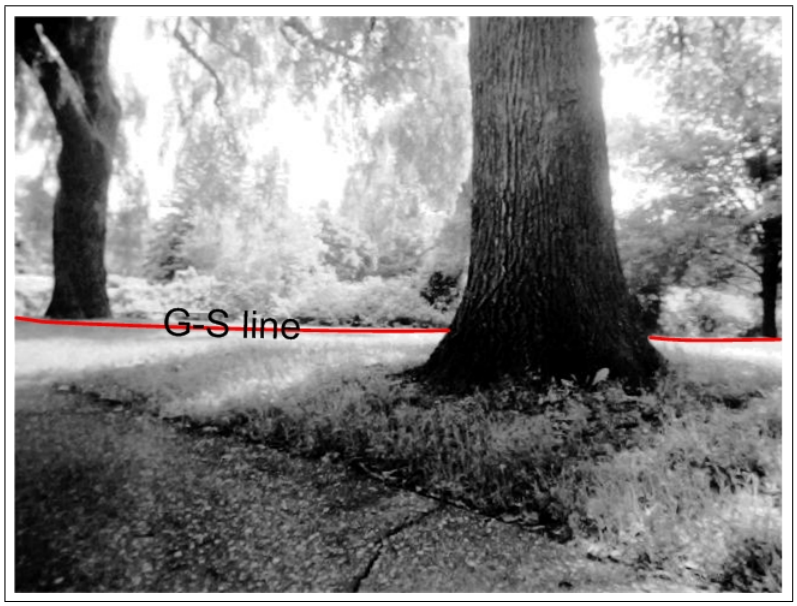


Figure 4.19: Image showing the two possible locations for a tree trunk with the Ground-Sky (G-S) line overlaid by hand. The tree on the right is close and its trunk intersects the G-S line, whereas the tree on the left is far and its base is adjacent to the G-S line. It is not possible for a tree base to be located higher than G-S.

Determining G-S is a difficult problem, and is one that is directly tied to Perception: where does the ground end and the sky begin? To the author's knowledge, there has been no work to date specifically focused on finding G-S. The closest work is aerial based ground-sky detection [111], which is more focused on finding the horizon for the sake of navigation of Unmanned Aerial Vehicles (UAV). This is a different problem than G-S for in the aerial scenario trees, shrub, rocks and similar ground based structured are assumed part of the ground; whereas in land-based navigation these structures are part of the Sky. In other words, the G-S system here is interested in segmenting 'Ground 0' from the sky and other structures protruding from the ground.

G-S line is attributed to the longest horizontal line in the image since the camera is assumed to remain oriented parallel to the ground. Its calculation is similar to that of vertical line tracking explained in Sections 4.3.2 and 4.3.2 but in this case it is the horizontal lines that are sought. Edges are detected and tracked in the horizontal direction and all horizontal lines below a threshold h are preserved. Tests on the database of experimental images indicate that a value of 90 pixels

for h preserves most of the G-S information while avoiding spurious horizontal information. The result of this intermediate operation is shown in the left image of Figure 4.20. The next step consists of taking the mean row height of all horizontal lines in each column

$$Hor(j) = mean(Hor_1(j), Hor_2(j), Hor_3(j), ..), \tag{4.12}$$

where $Hor_1(j)$ is row height of the horizontal line '1' at column 'j'. The final result is shown in the right image of Figure 4.20.

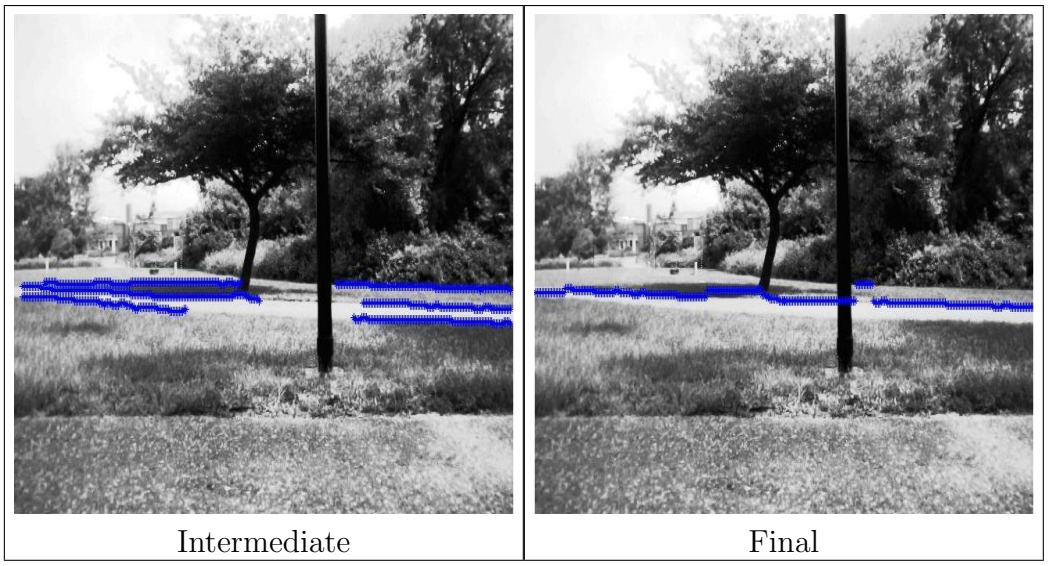


Figure 4.20: G-S detection system based on finding the dominant horizontal lines.

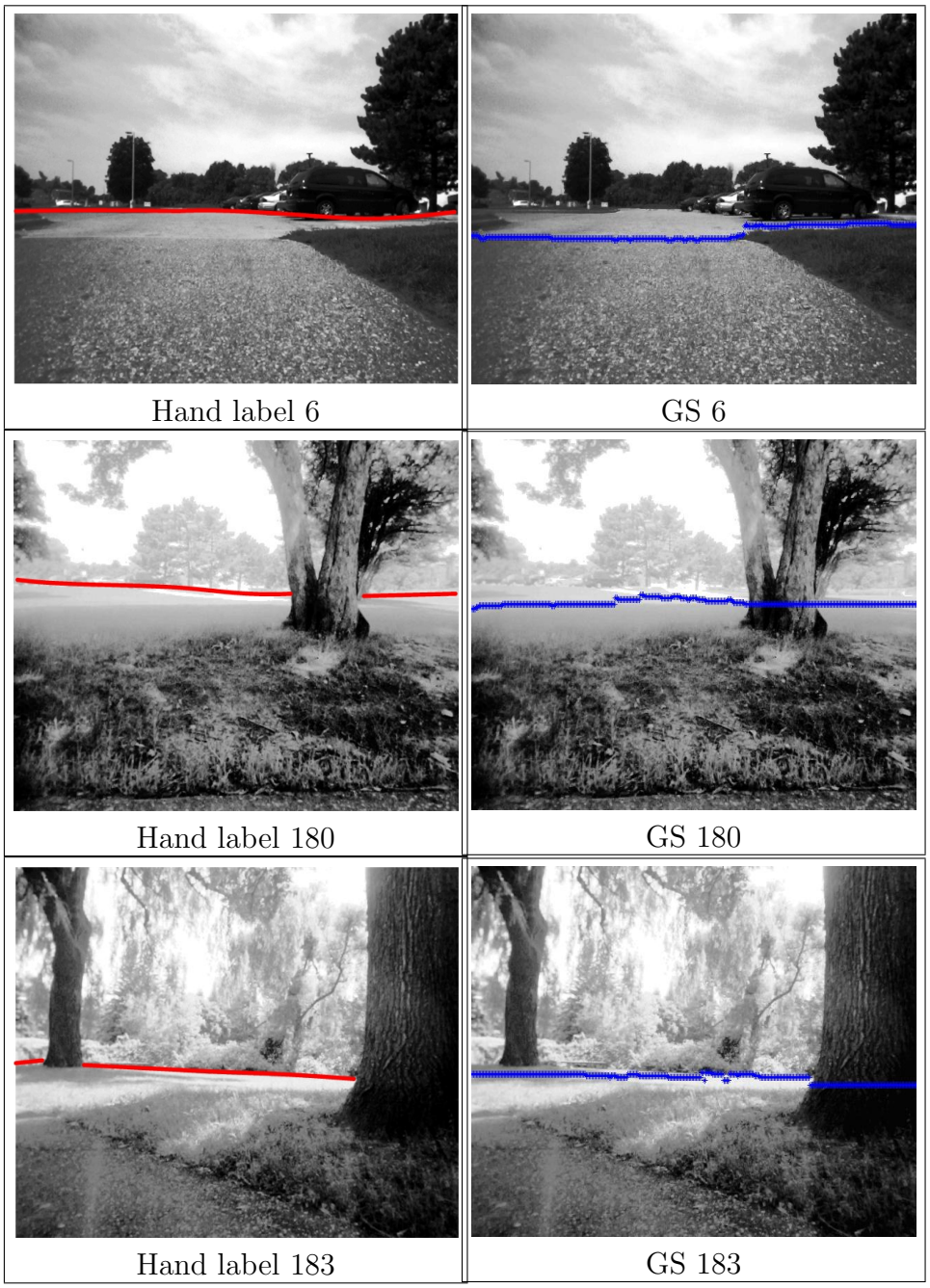


Figure 4.21: Results of the Ground-Sky detection system (right), compared to ground truth (left). The numbers 6, 180, and 183 refer to the image number in the database of collected images.

Unfortunately, this G-S system sometimes fails due to long horizontal lines in the image that do not represent G-S. It is found that setting a nominal value for the G-S line and using a range of acceptable G-S lines produces better results. More specifically, all lines with their endpoints below the horizon range and with no continuous or symmetric lines above the horizon range are pruned. All lines with their beginning above the horizon and with no lines which are continuous or symmetric within the horizon range are pruned. The range of acceptable G-S lines is determined empirically by hand segmenting all the collected images during an experimental run and recording the position of G-S. It is found that most of the G-S lines are located within 50 pixels of the middle of the images (*i.e.*, 240 ± 50 pixels).

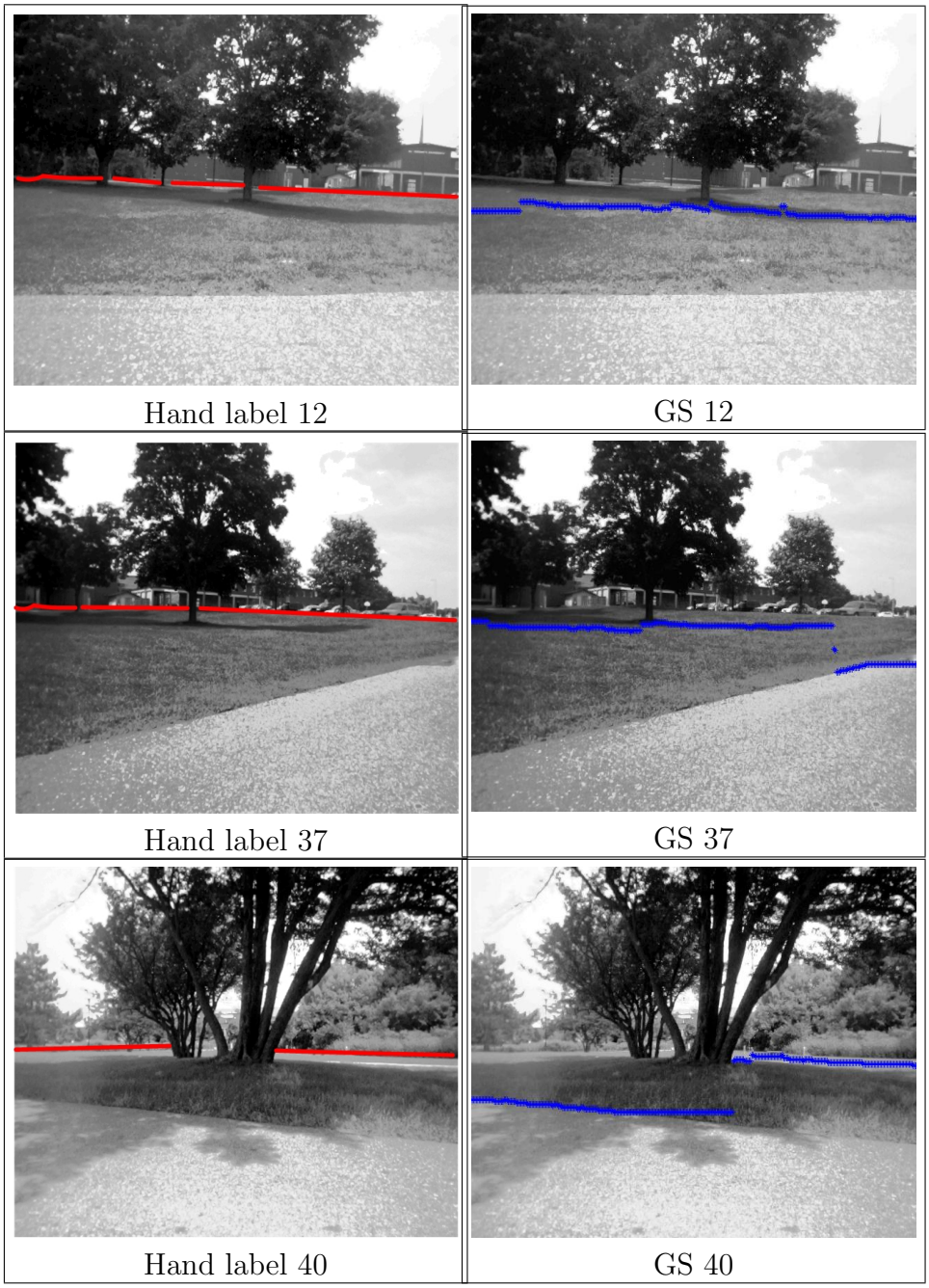


Figure 4.22: Failures of the Ground-Sky detection system (right), compared to ground truth (left). The numbers 12, 37, and 40 refer to the image number in the database of collected images.

Grouping lines into trees

Continuous and symmetric lines are now grouped into potential trees based on two criteria; namely, entropy reduction and proximity of the base to horizon. The remaining lines are grouped into trees by reducing the entropy of the pixels in the image, where the entropy is defined in (4.13)as follows

$$E = -\sum(P_n)log2(P_n) \tag{4.13}$$

and P_n is the probability of a pixel having a label n and is calculated as

$$P_n = \frac{n_c}{T_P} \tag{4.14}$$

where n_c is the count of pixels in the image labeled n and T_P is the total count of pixels inside the image.

Results

Figures 4.23 and 4.24 show the result of applying the PO-based tree detection system described above to a temporal sequence of images (180-197) taken from data set 1 of the experimental data (Section A.3). A quantitative analysis of this tree detection system is presented in Section 4.7, where its performance is evaluated on an entire run during which the navigating vehicle traverses a complete loop around the test site.

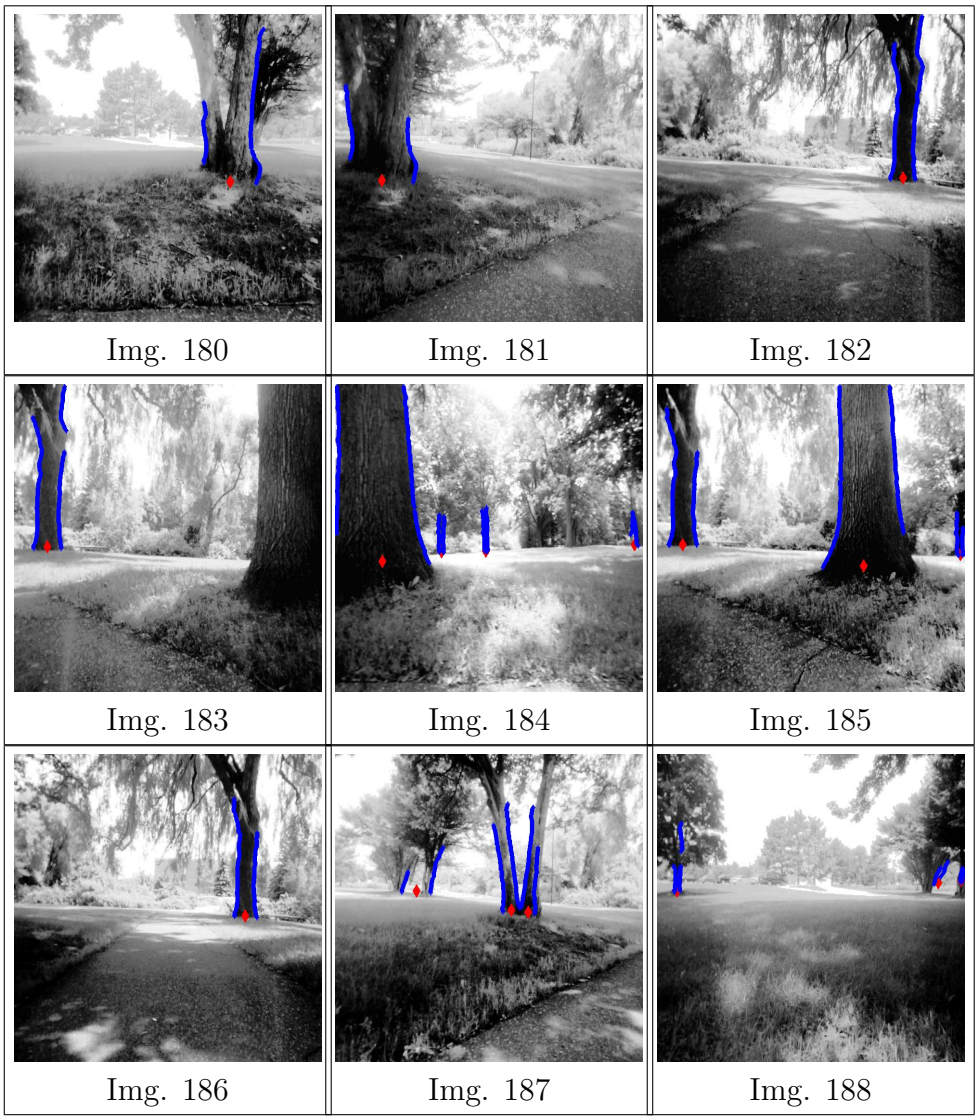


Figure 4.23: Tree detected via perceptual organization. The system appears to detect all instances of trees when they are isolated and relatively closely situated. Problems occur when multiple trees intersect (Img. 180) and when trees are far away and poorly contrasted (Img. 187).

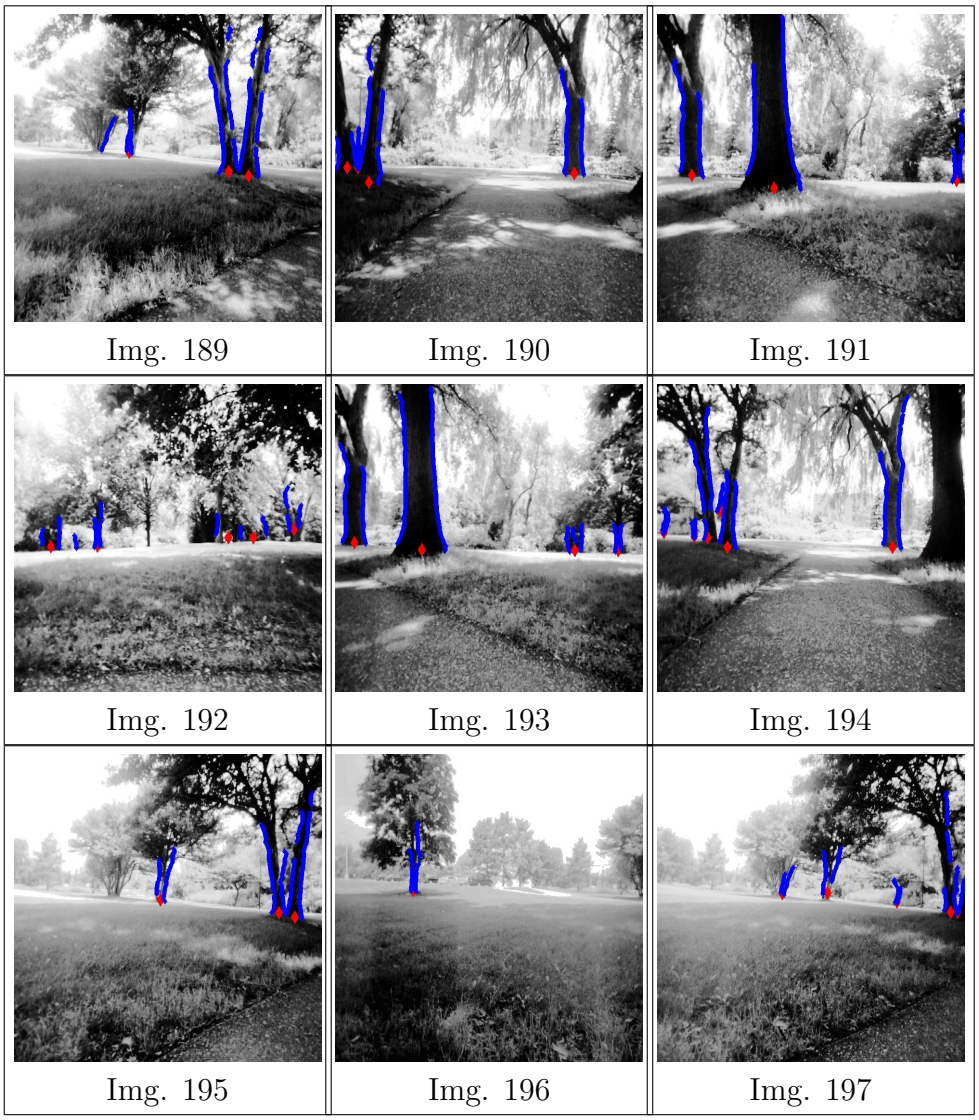


Figure 4.24: Tree detected via perceptual organization. The system appears to detect all instances of trees when they are isolated and relatively closely situated. Problems occur when multiple trees intersect (Img. 192) and when trees are far away and poorly contrasted (Img. 195).

4.4 Tree recognition

In VisSLAM, tree trunks are recognized and matched to previously viewed trees based on the features which they exhibit. These features must be very distinctive in order to avoid false positives, and must be recognizable from different camera viewpoints and varying lighting conditions. Mikolajczyk and Schmid [68] compare the performance of many descriptors (See discussion in Section 4.3) and find that SIFT features are the most stable and distinctive of them all. For this reason, SIFT features are used in VisSLAM as the basis for the tree recognition system. Once a tree is detected, the SIFT features located within its boundaries are matched to those in a database, and the corresponding match indicated the identity of the queried tree. In a real-time SLAM, the SIFT database can be compiled online and updated each time a new tree is observed by augmenting the SIFT database with the SIFTs and their corresponding identity. In VisSLAM, the SIFT database is prepared offline, by extracting the SIFTs of all landmarks from several viewpoints and adding them to the SIFT database.

Due to the importance of SIFT to the success of the recognition system, the following sections review the extraction of SIFTs and the calculation of their descriptors. In Section 4.4.2 SIFT features are extracted and overlaid on images of trees taken from various viewpoints.

4.4.1 SIFT extraction and description

Extraction

SIFT features are detected via a Difference of Gaussian filter (see Figure 4.25). First, the input image is filtered with a 2D Gaussian kernel ($2^{1/k}$) to produce the first entry at the bottom of the image. Then, each processed image is convoluted with this same 2D Gaussian kernel to produce the image one level higher on the same octave. The DoG images on the right are obtained by subtracting two adjacent images. Once the desired number of images per octave are obtained, the image corresponding to the original Gaussed by a kernel equal to 2 (*i.e.*, $2^k/k$) is down-sampled by two to produce the first image on the next octave. The procedure is repeated for each scale.

Once all the DoG images are obtained, the maximum values inside each image are determined and compared to those obtained at lower and higher scales to yield the most salient regions in space and scale. In this work, the DoG uses two Gaussed images (*i.e.*, one DoG image) for each octave at a spatial frequency of $\sigma = 1.6$ and

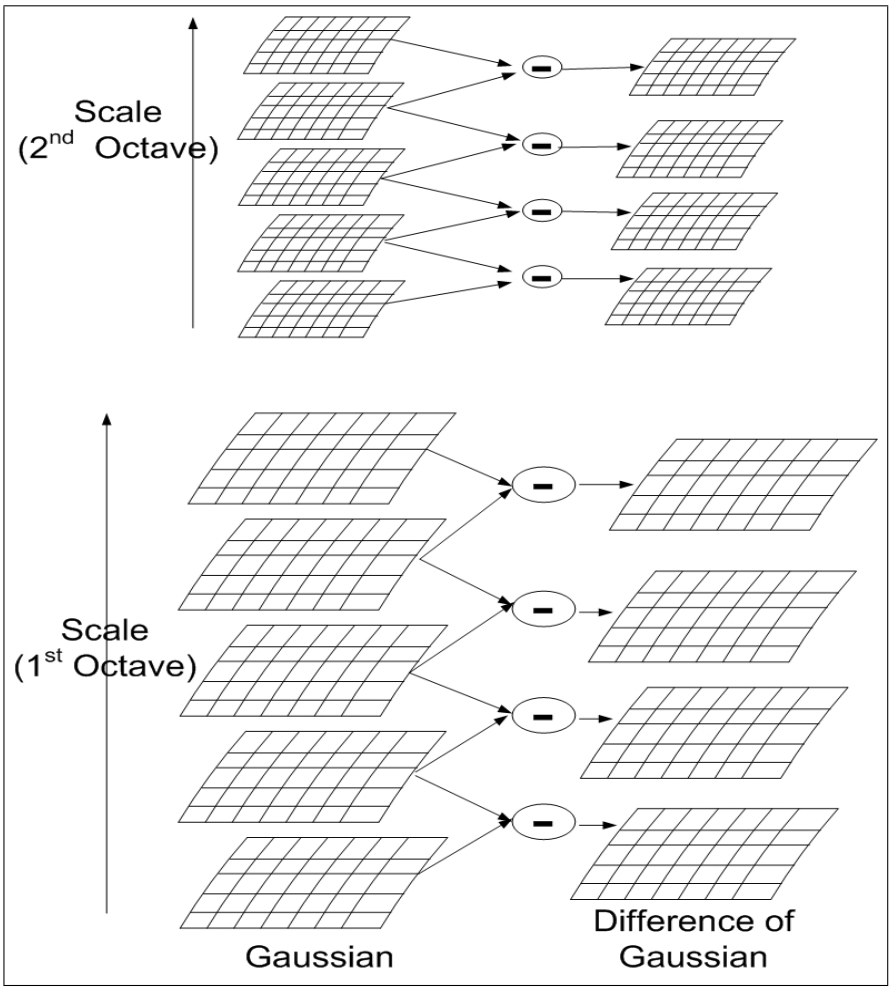


Figure 4.25: Procedure for extracting the Difference of Gaussian (DoG) image saliencies. The input image is first convolved with a 2D Gaussian (the bottom image). Next the convolved image is convolved to produce the image higher on the scale. This procedure is repeated to produce all the images on one octave. The DoG images on the right are obtained by taking the difference between two adjacent Gaussed images. After each octave, one of Gaussed images (chosen to avoid aliasing) is down-sampled by a factor of two and the process is repeated for that octave. This image is copied from that produced by Lowe [104].

ten octaves in total. Figure 4.27 shows five of the ten DoG images for a sample image where the '+' sign indicate the location of DoG maxima at the current scale. The final large image shows the original image with all the detected IPs overlaid,

where the size of each square reflects the scale of the IP.

Descriptor

While the DoG operator insures invariance of the SIFT features to changes in lighting conditions and camera viewpoint, it is its descriptor that ascertains its distinctiveness from other DoG features. At each DoG point, a window of 16 x 16 pixels is taken surrounding this point (see Figure 4.26). This window is then divided into 16 sub-windows (4x4) and inside each sub-window all orientations are weighed in a histogram. All the weights of each of these 16 sub-windows for each of the 8 orientations are grouped into a feature vector known as the SIFT descriptor. The SIFT descriptor is 128 dimensional, including 8 orientations times 16 sub-windows.

The success of SLAM is hinged on recognizing a landmark (*i.e.*, data association) and using the difference between the observation of that landmark (range and/or bearing) on one hand and the prediction of that same observation on the other hand, to correct the ego-motion estimate of the navigating vehicle and the estimate of the SLAM map. It is therefore imperative that the system not only detect a landmark but also recognize which specific landmark it is (*e.g.*, tree 'one' verses tree 'nine'). One option for tree recognition is to list SIFT features that appear inside the tree region from several viewpoint, and then to match a query tree by matching one of its SIFT to those in the database. Since SIFT features are invariant to camera viewpoint and lighting conditions, the system should be able to match SIFT features of one tree to very few template SIFT features taken from that tree. A more robust recognition system is to use multiple SIFT for matching, but care must be taken not to use too many of these SIFTs to avoid false rejection. The best compromise between correct recognition and false positives is to require 3 SIFT matches to infer a correct tree match.

4.4.2 Results

Figures 4.28 and 4.29 show the images that are used to extract the SIFT descriptors for Tree 'six' and 'seven' respectively. Images are taken for nine different viewpoints to capture sufficient SIFT features to represent the tree from all viewpoints. A quantitative analysis is differed to Section4.7 where the system is tested on a database of images taken while the navigating vehicle completes a full loop around a track.

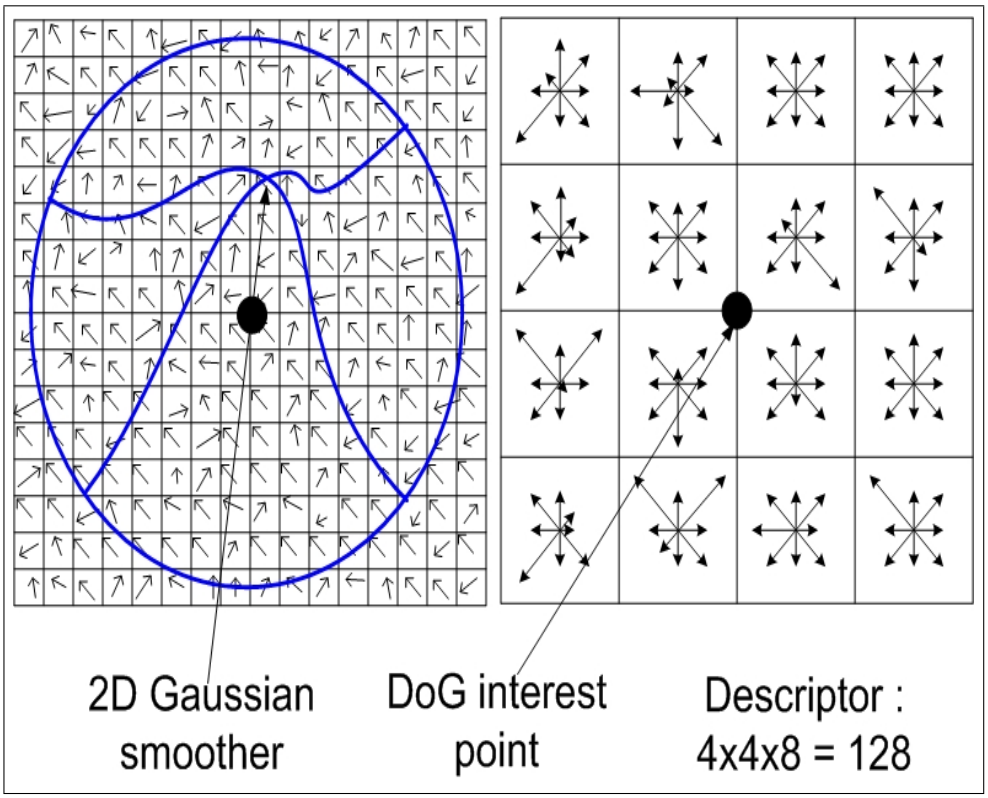


Figure 4.26: SIFT descriptor. After each DoG interest region is found, it is expressed by its corresponding SIFT descriptor. A 16x16 window is take around each interest point and the orientations in eight directions are recorded at these locations. The large 16x16 window is then subdivided into 4x4 sub regions where an orientation histogram is set up in each of these regions. The value of each histogram entry is reflected by the size of the arrow in the eight arrow cluster inside each of the subregions. Finally each of the orientation entries in each of the subregions are accumulated in a feature vector known as the SIFT descriptor (4x4 subregions x 8 orientations = 128 entries). This image is inspired by a similar one produced by Lowe [104].

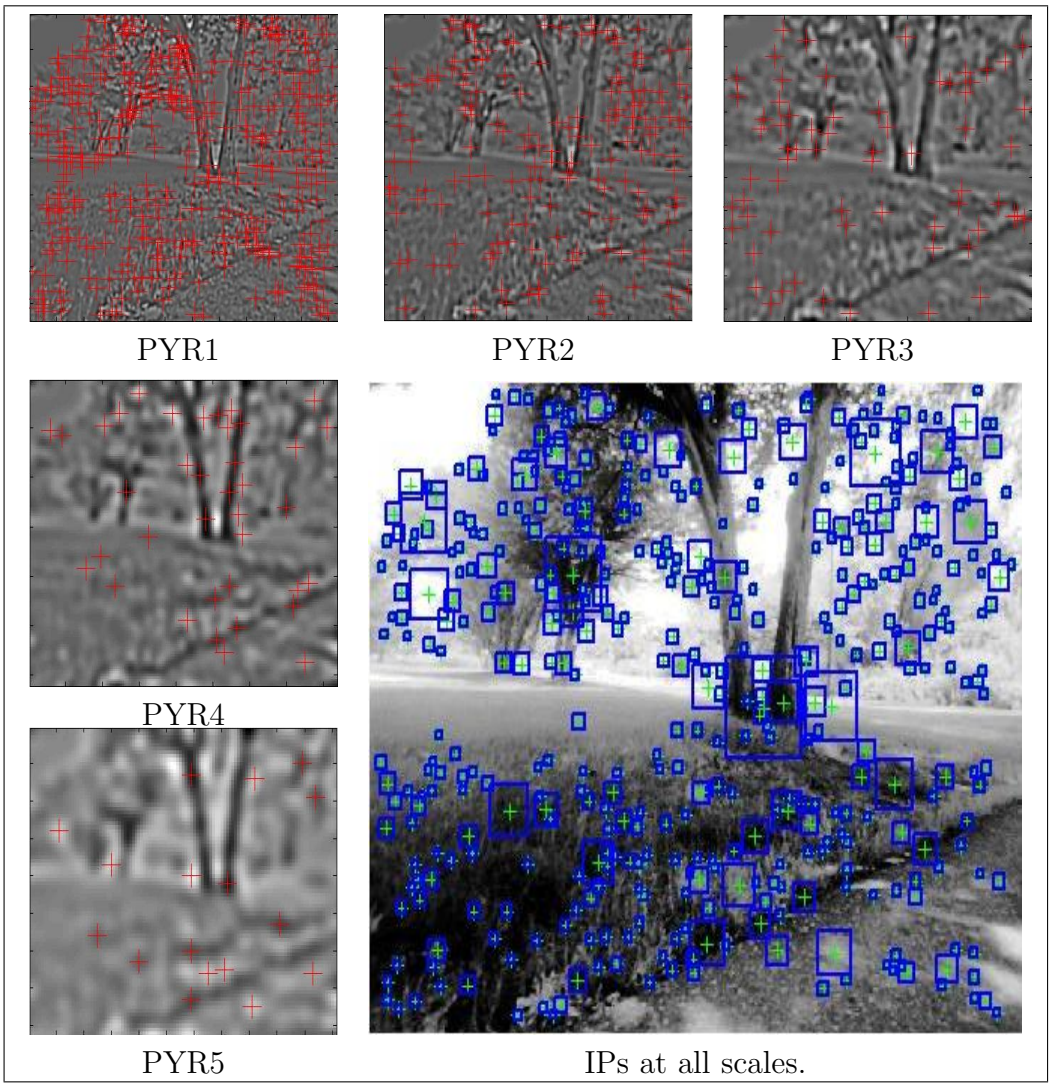


Figure 4.27: DoG Interest Points (IPs) shown at 5 scales (the + signs). Although the images appear to be the same size, the actual size of PYR2 is half of PYR1 and that of PYR3 is half of PYR2. For this reason, when they are shown at the same scale the smaller images appear blurred due to the required averaging. The final large image with squares overlaid shows all the IPs on the original image, where the size of the square reflects the scale of the IP.

Figure 4.28: Tree 'six' shown from nine different viewpoints, with overlaid SIFT features on every view. The SIFT features provide distinctive marks by which trees are identified.

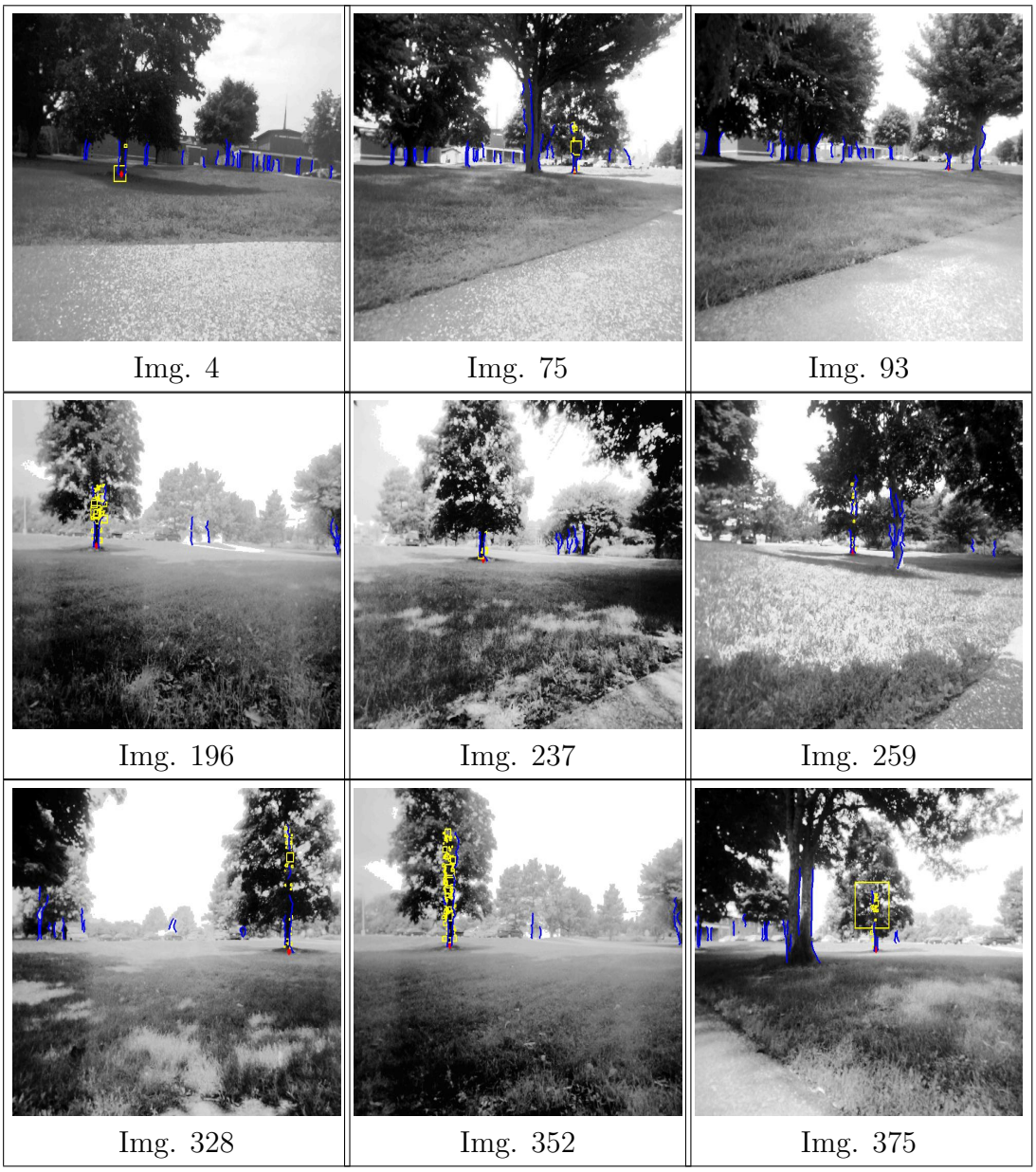


Figure 4.29: Tree 'seven' shown from nine different viewpoints, with overlaid SIFT features on every view. The SIFT features provide distinctive marks by which trees are identified

4.4.3 Can stereo help tree recognition?

One method by which stereo depth information can assist recognition is by reducing false negatives and false positives when the queried tree is within the range of the stereo camera. For instance, when a tree is associated to another tree, the depth of the recorded tree and that of the observed tree are compared to validate if indeed the match is correct or not.

4.5 Tree initialization

Another issue that is crucial for SLAM's success is the initialization of landmarks into the SLAM map. Initialization involves adding the landmark position into the SLAM state and its corresponding covariance into the SLAM covariance matrix. One must be cognisant that the EKF upon which SLAM is built imposes Gaussian representation for all SLAM state variables and works by linearizing a non-linear estimate of vehicle pose and the position of the landmarks. Landmark locations must exhibit small variances in order to approximate their non-linear functions with their respective linearized forms. Therefore, it is imperative that the initial guess of the landmark's position be as precise as possible in order to avoid large non-linearities that can cause the EKF to diverge. Landmark initialization can be performed in a bearing-only or range-bearing fashion.

4.5.1 Bearing-only initialization

In bearing-only SLAM it is not possible to localize a landmark and initialize it into the SLAM map from a single observation; rather, it takes at least two observations of the same landmark and knowledge of the vehicle motion between the two observations in order to localize a landmark (Figure 4.30).

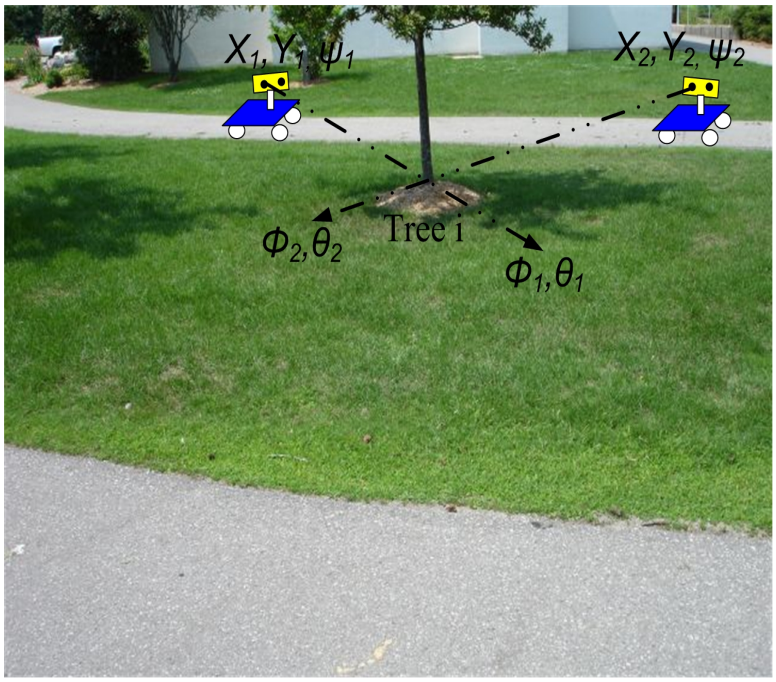


Figure 4.30: Initializing a landmark via bearing-only. Once a tree is detected, its bearing to the camera θ and φ is saved as well as the pose of the robot. If the tree is detected in subsequent images, its bearing is saved with the respective position of the robot, however it is not initialized into the SLAM map until sufficient angular gradient is sufficient to avoid ill-conditioned problems.

The problem with the bearing-only procedure is the requirement for a large baseline between the positions at which landmarks are sighted at the time of initialization (This issue is detailed in Section 2.3). Unfortunately, in the experiments run in VisSLAM, the vehicle navigates for long periods of time in a straight line and only rotates when changing its heading and making another straight line trajectory. It would therefore take a long time before the initialization condition be met and the system would have most probably already diverged because of the INS growing errors.

4.5.2 Can stereo help tree initialization?

Alternatively, landmark initialization can be performed in range-bearing fashion if a stereo camera is available. The stereo camera is used to infer the depth of a

landmark via stereopsis, where the parallax between the two lenses of the stereo pair is used to calculate the depth of a landmark. One important issue is that depth quality decreases with range of the camera. This issue is evidenced by the images compiled by the camera manufacturer [112] shown in Figure 4.32, where the long and short range accuracy are plotted verses the range of the camera. VisSLAM uses a range cut-off point, beyond which potential trees are not initialized. This point is chosen at 7.5 meters, where a maximum error of 1 meter can occur. One could allow for higher depth values but must be cognisant of the risk of filter divergence if too large an error exists in the initial landmark position estimate. Figures 4.32 and 4.33 show images where the trees within are initialized. Tree 'one' is initialized when it is at 5.165 meters away. Tree 'six' is initialized at 2.4 meters. These trees are then initialized into the SLAM map by adding half the diameter of each tree to its corresponding range reading.

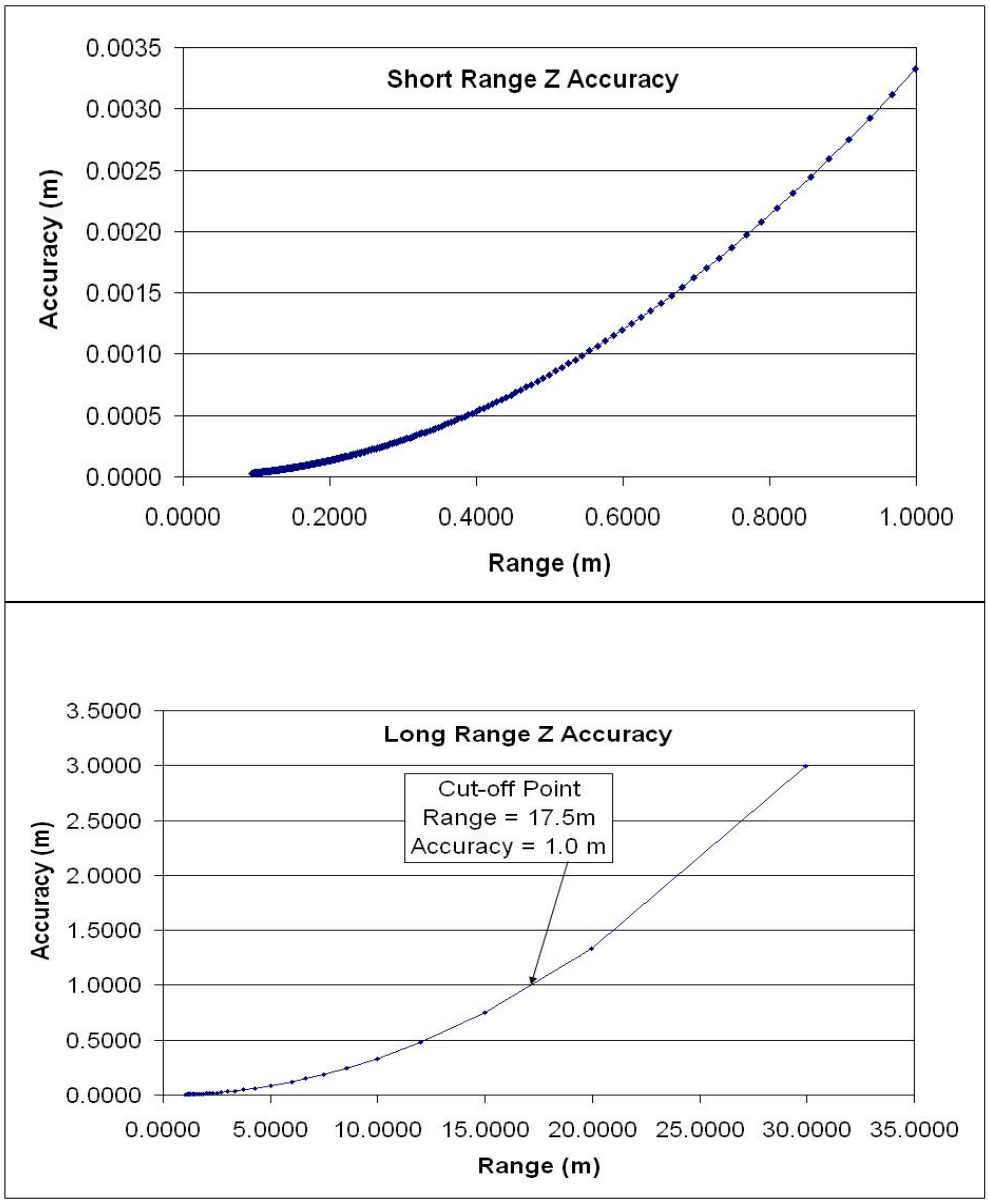
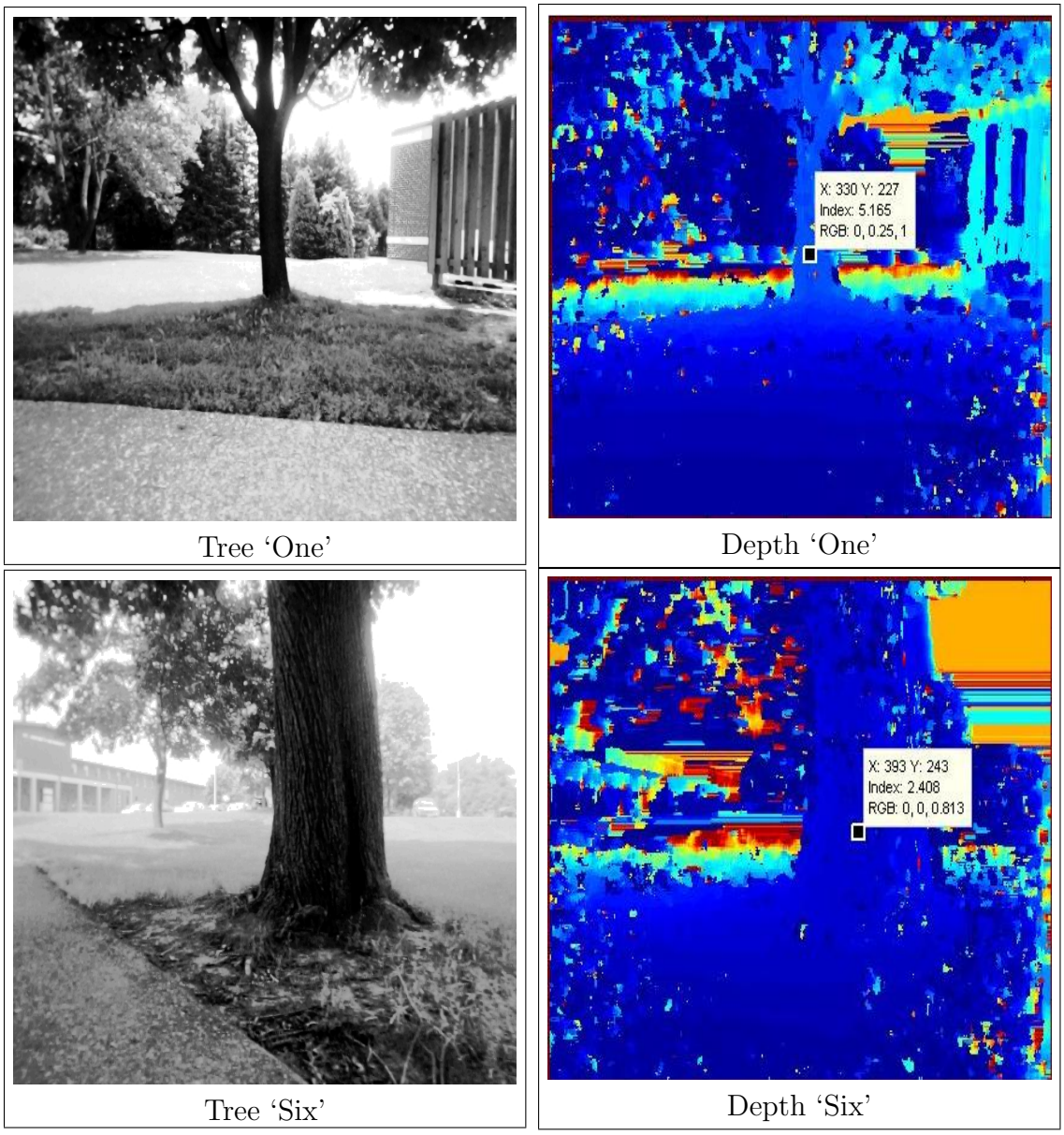


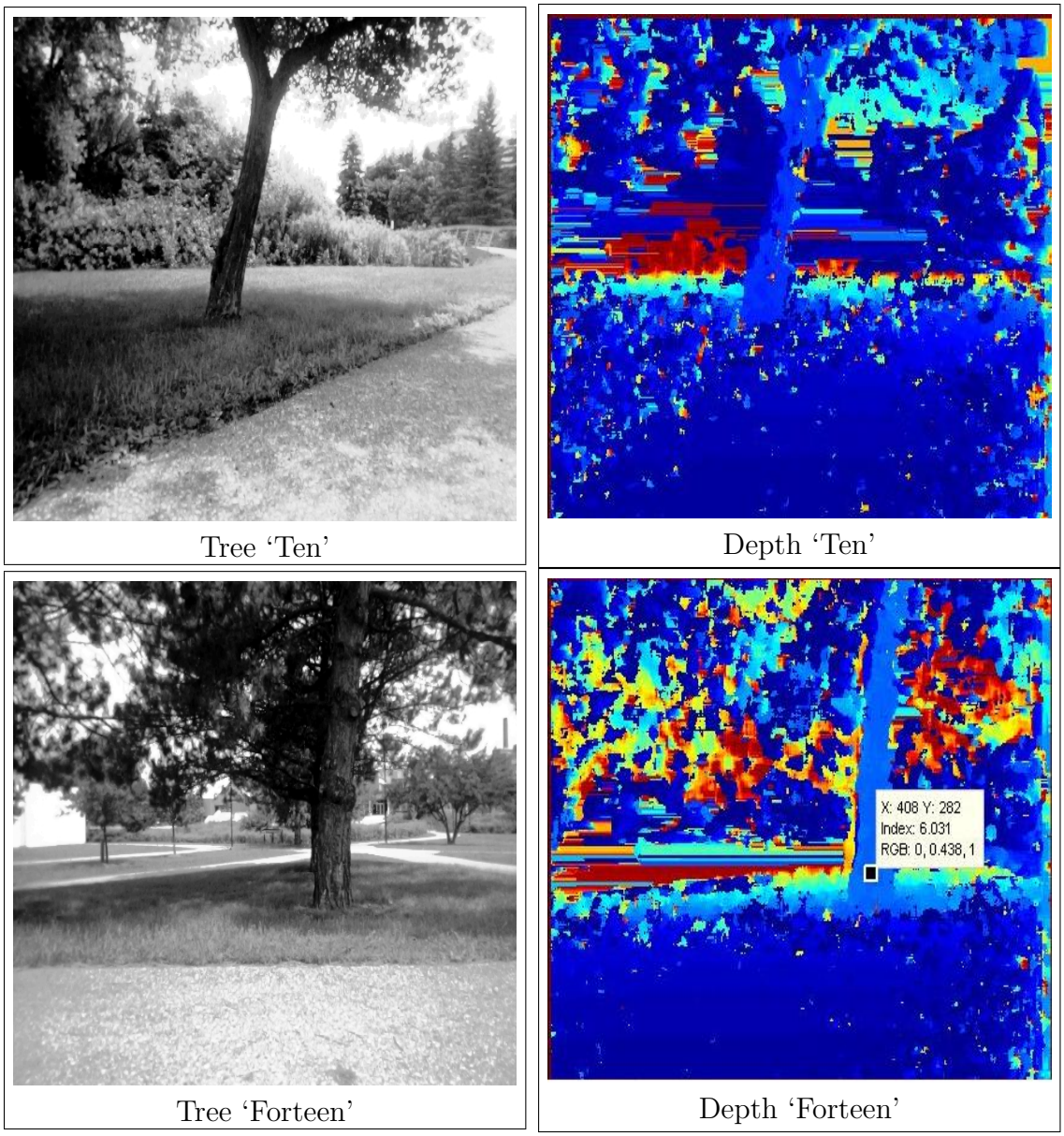
Figure 4.31: Stereo accuracy of the bumblebee camera. The accuracy is plotted verses the range of the camera for short range (top) and long range (bottom). Any landmark observed at a depth value beneath the cut-off point (shown in the bottom graph) of $17.5m$ is deemed acceptable and the landmark is initialized. The accuracy of the depth estimate at the cut-off point is $1m$. If the landmark is situated beyond this point, it is not initialized.



Tree 'One' | Depth 'One'

Tree 'Six' | Depth 'Six'

Figure 4.32: Trees initialized using depth from stereo. Trees 'One' and 'Six' initialized when they are within the range of stereo camera.



Tree 'Ten'

Depth 'Ten'

Tree 'Forteen'

Depth 'Forteen'

Figure 4.33: Trees initialized using depth from stereo. Trees 'Ten' and 'Fourteen' initialized when they are within the range of stereo camera.

4.6 Architecture

The above system, including tree detection, recognition and initialization, is summarized in the flowchart in Figure 4.34. The vision system first captures a stereo image of the scene, which it unwraps and rectifies, performs stereo correspondence and performs 3D reconstruction of that scene. At the same time, the image is processed by the tree detection system, where edge detection, line tracking, line pruning, and line grouping is performed to detect potential tree trunks. The image is simultaneously filtered by a SIFT filter and those inside each tree boundary are associated to that tree. These SIFT features are then matched to a library of SIFT descriptors of previously viewed trees. If 3 SIFT descriptors match to the same tree, the system concludes that the query tree is recognized and initiates a SLAM update using that tree. If, on the other hand, the queried tree does not match any in the database, an attempt is made to initialize it into the SLAM map, where acceptable conditions are determined by the stereo system as described in Section 4.5. If the queried tree is within the *acceptable* range of the stereo camera, it is initialized; if it is not within the acceptable range it is discarded and the next tree is processed.

4.7 Experiments

The Computer Vision system proposed above is tested on the first data set of the experimental runs (described in A.3) where the vehicle completes a full loop and the system is evaluated based on correct detection, recognition and initialization rates. Other measures of interest include false positive detections and false positive recognitions.

4.7.1 Results

Images are collected from a camera onboard a mobile vehicle while it is navigating inside a park area, featuring relatively flat and smooth paved paths that circulate between a number of conifer trees, exhibiting salient tree trunks (See Appendix A.2 for images of the trees that are used as landmarks). In addition to the trees, 7 lampposts are observed at the test site. These posts are treated as tree trunks by the detection and recognition system. This is done due to the scarcity of trees in some locations. 580 images are collected in total as the vehicle makes a complete cycle around a pre-determined path. These images are hand segmented in order to

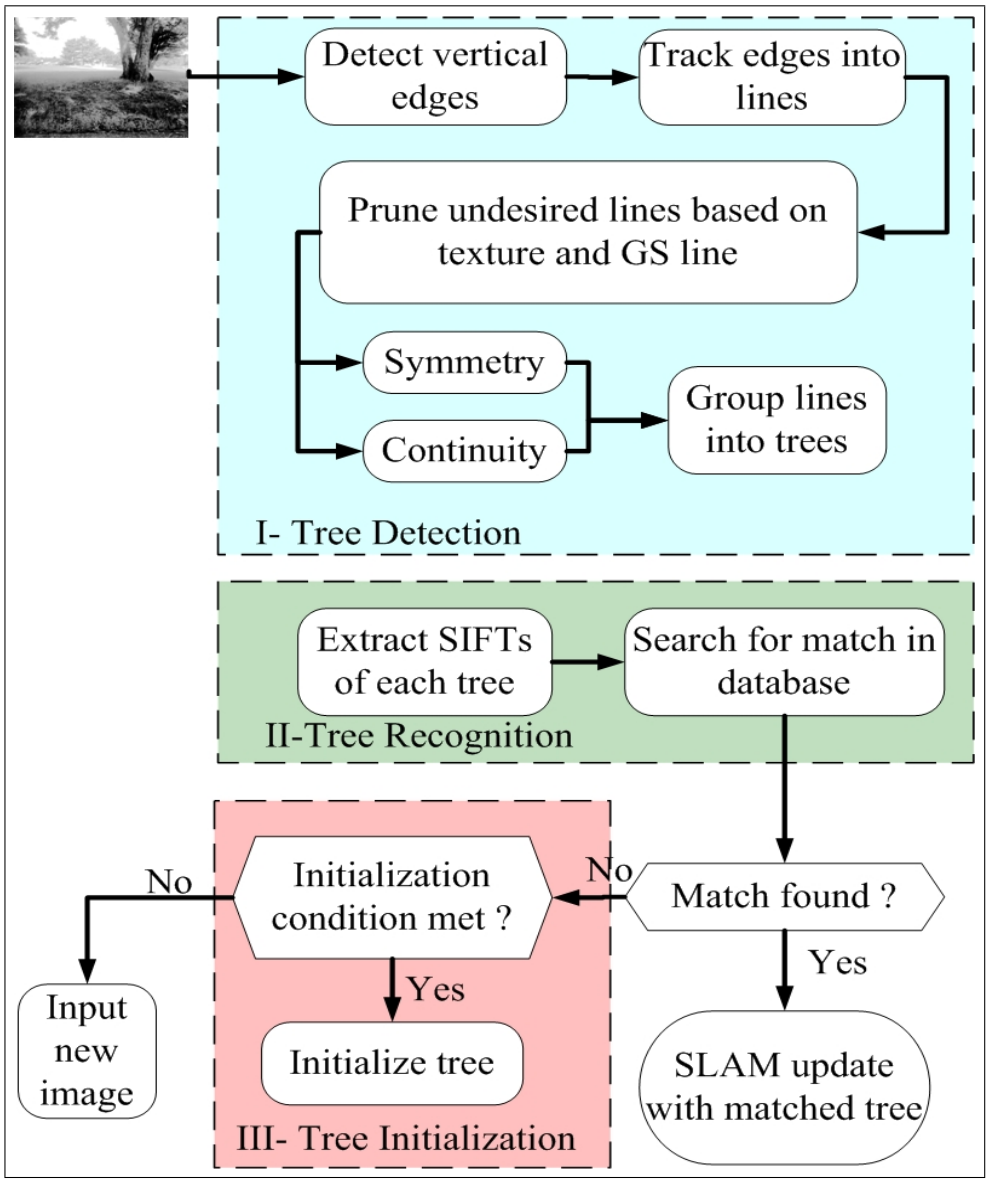


Figure 4.34: Architecture of the Computer Vision system.

determine the number of times each tree is sighted as well as the total number of these trees, where 932 tree trunks are marked as potential landmarks for the vision system.

The performance of the vision system on the aforementioned tree images is summarized in tabular form in Table 4.1 and then graphically in Figure 4.35. The

average detection rate is 71.24% and the average recognition rate of these detected trees is 85.54%.

Trees '2' and '7' are marked as initialized, which signifies that during an entire cycle of the vehicle, trees 2 and 7 are never close enough to the camera to satisfy the initialization criterion (*i.e.*, closer than 7.48 meters away). A picture of these trees is shown in Figure A.3 and it is apparent that the path along which the vehicle circulates is far from these trees. A solution to this shortcoming is to combine range-bearing and bearing-only initialization techniques; where trees that satisfy the bearing-only initialization criteria (Section 2.3) are initialized if they have not yet been initialized through range-bearing methods. This method is not attempted here but is left for future research.

Landmark	Total # of Sightings	Detection		Recognition		Initialization
		True	%	True	%	(Y,N)
Tree 1	28	20	71.43	17	85.00	Y
Tree 2	40	35	87.50	26	74.29	N
Tree 3	50	32	64.00	25	78.13	Y
Tree 4	41	31	75.61	26	83.87	Y
Tree 5	51	33	64.71	24	72.73	Y
Tree 6	54	39	72.22	32	82.05	Y
Tree 7	58	45	77.59	38	84.44	N
Tree 8	49	35	71.43	33	94.29	Y
Tree 9	31	20	64.52	17	85.00	Y
Tree 10	61	55	90.16	53	96.36	Y
Tree 11	33	22	66.67	15	68.18	Y
Tree 12	27	17	62.96	13	76.47	Y
Tree 13	28	21	75.00	15	71.42	Y
Tree 14	20	11	55.00	9	81.82	Y
Tree 15	23	16	69.57	15	93.75	Y
Tree 16	26	14	53.85	13	92.86	Y
Tree 17	20	18	90.00	16	88.89	Y
Tree 18	11	9	81.82	7	77.78	Y
Tree 19	6	4	66.67	3	75.00	Y
Tree 20	10	7	70.00	7	100.00	Y
Tree 21	9	6	66.67	5	83.33	Y
Tree 22	4	1	25.00	1	100.00	Y
Tree 23	100	53	53.00	48	90.57	Y
Tree 24	48	33	68.75	30	90.91	Y
Lamp 1	26	19	73.08	17	89.47	Y
Lamp 2	39	35	89.74	34	97.14	Y
Lamp 3	11	10	90.91	9	90.00	Y
Lamp 4	11	9	81.82	7	77.78	Y
Lamp 5	6	5	83.33	5	100.00	Y
Lamp 6	1	1	100.00	1	100.00	Y
Lamp 7	2	1	50.00	1	100.00	Y
Post 1	8	7	87.50	6	85.71	Y
Total	932	664	71.24	568	85.54	

Table 4.1: Performance of Computer Vision system during a complete cycle of the robot. A tree is marked as not initialized (N) if during the entire run it does not enter into the confidence range of the stereo camera (7.48 meters).

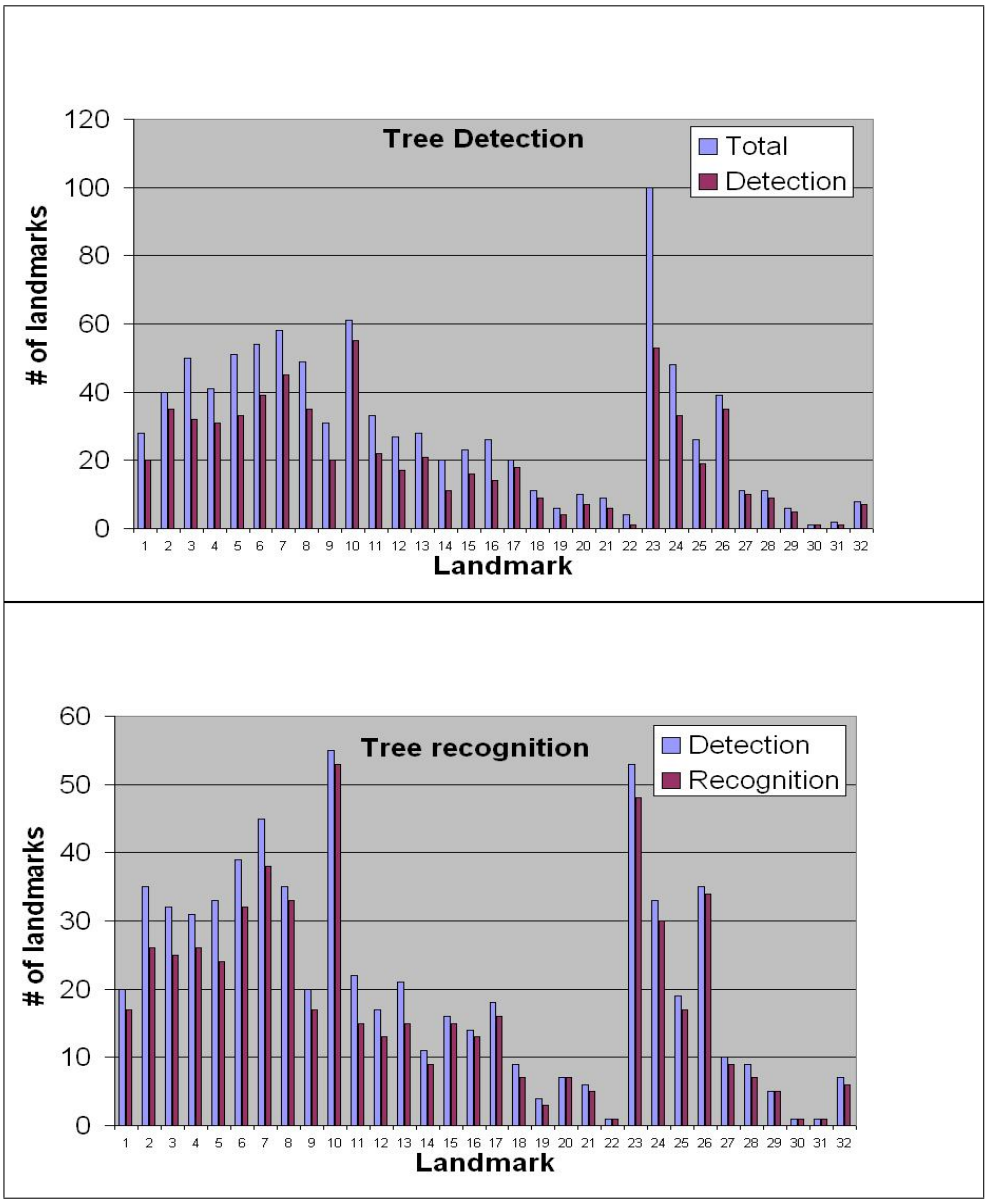


Figure 4.35: Tree detection and recognition rates.

Trees that are connected at the base including Tree '11', Tree '12', Tree '13',Tree '23', and Tree '22' are merged and treated as one tree. The tree detection manages with such trees by keeping track of the distance between the centers of each tree pair '$d_{tree1-tree2}$' and the width of each tree 'W_{tree1}' and 'W_{tree2}'. The merging

predicate of two trees is as follows:

$$P(tree_1, tree_2) = \begin{cases} true & \text{if } \frac{W_1+W_2}{2} \geq d_{12} \\ false & otherwise \end{cases} \tag{4.15}$$

where $P(tree_1, tree_2)$ is the merging predicate between $tree_1$ and $tree_2$.

If the sum of the widths of two trees is equal to or larger than the distance between the centers of these trees, the trees are merged and considered as one tree (See Figure 4.36). Figure 4.37 shows Tree '23', which in essence is two trees joined at the base. In the image on the left the result of the tree detection system without merging is shown, on the left the result is shown after the merging.

The system is written in Matlab and implemented on a Pentium M $1.8GHz$ processor with 1GB of RAM. The time to detect trees on a 640x480 image, including edge detection, line tracking, continuity and symmetry detection, line and tree pruning is 6 seconds.

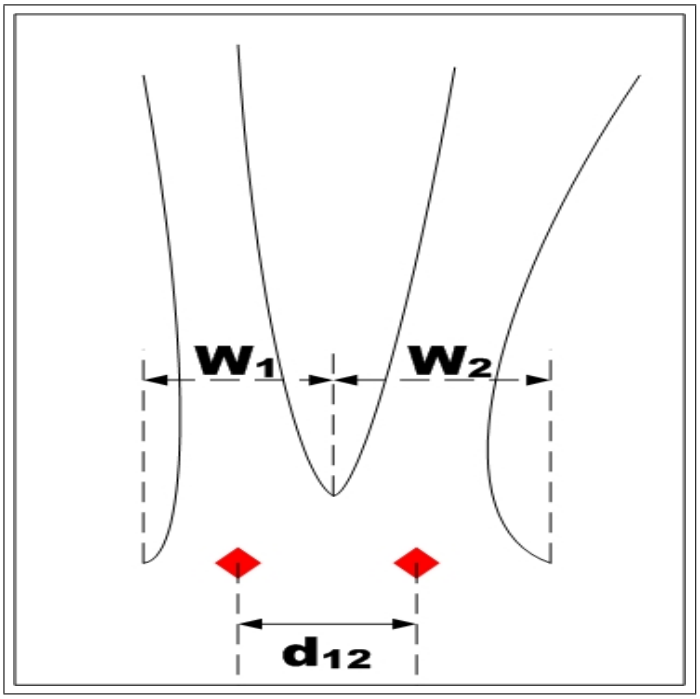


Figure 4.36: Merging trees connected at the base. If $\frac{W_1+W_2}{2} \geq d_{12}$ the two trees are merged.

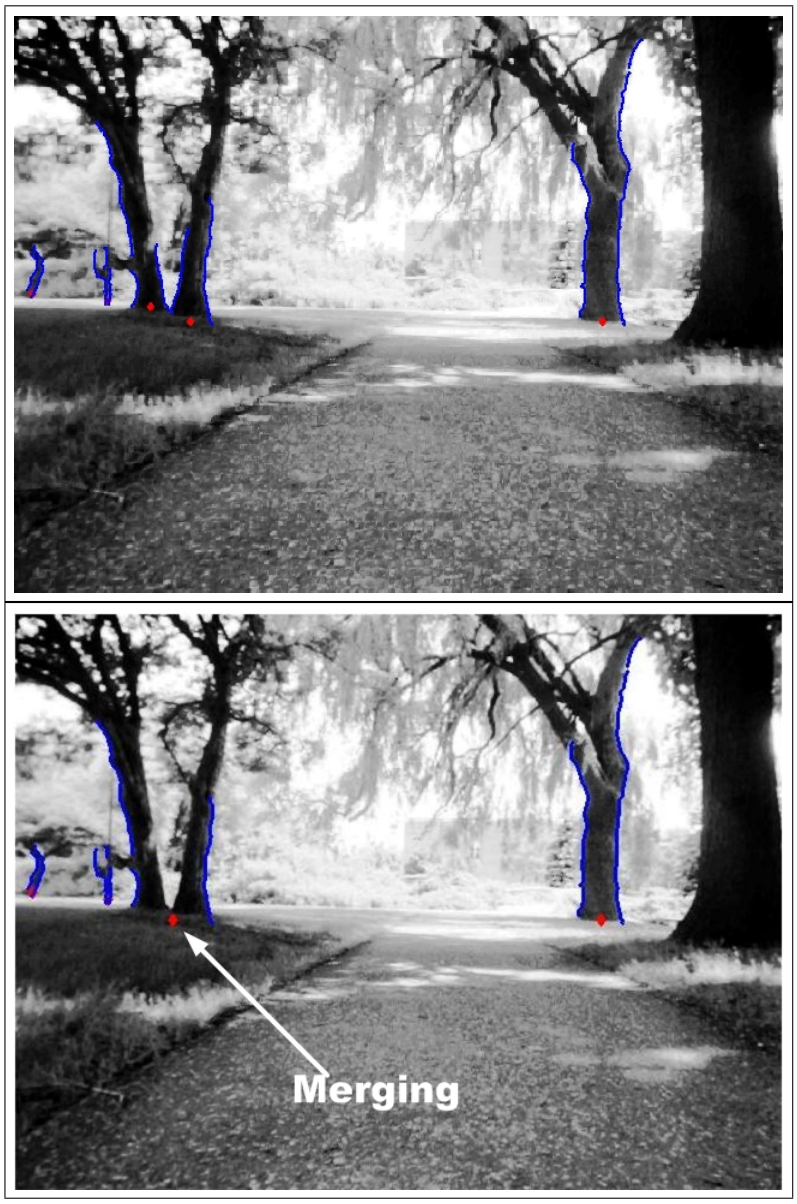


Figure 4.37: Merging of the two trees of Tree '23'.

4.8 Summary

This chapter constitutes the fundamental contribution of this thesis. Three Computer Vision systems are introduced including environment recognition, tree detection, and tree recognition.

Environment recognition is based on extracting high-level primitives inside images via a steerable pyramid and using the resulting feature vector as the image's signature. An Artificial Neural Network is trained to classify each image based on its discriminative signature. Results of experiments conducted on 350 indoor images and 270 outdoor images show correct classification rates of 95% and 79% respectively.

The detection system is based on segmenting images into quasi-vertical structures and selecting those structures that are close to the Ground-Sky separation line as tree trunks. The segmentation is a structure-based segmentation which consists of detecting vertical edges and tracking them into vertical lines. A measure of continuity and symmetry is then calculated for each pair of lines. In each image, vertical structures are then constructed using compatible lines (continuity and symmetry) by reducing the total entropy of the image. The system is tested on a database of 580 images containing 932 trees. The total number of trees detected is 664 trees, which is equivalent to 71.24%.

The recognition system is based on matching trees in feature space. Each tree that is detected, is scanned for SIFT features within its borders and these features are matched to those of previously viewed trees. If three SIFTs match to the same tree the query tree is declared matched. The system is tested on the same database as that of the tree detection system and results in 85% correct recognition rate.

The localization system uses depth via stereopsis to localize a landmark in 3D space. Trees that are located within 7.48 meters of the camera are localized with an accuracy below 18.72 cm. During the experiments, where the navigating vehicle makes a complete cycles, all trees except for two of them 'T7' and 'T22' are initialized. The other two are never initialized since they are never located within the confidence range of the stereo camera.

Chapter 5

VisSLAM system

5.1 Introduction

This chapter is intended to demonstrate the application of the fundamental contributions presented in Chapters 3 and 4 in the context of both autonomous localization and SLAM. The developed inertial and Computer Vision systems are integrated into a working SLAM system, which is named VisSLAM. VisSLAM is built in the framework of an Extended Kalman Filter, where the prediction model is the INS system described in Chapter 3 and the update model is based on the Computer Vision system presented in Chapter 4. The mechanics of VisSLAM are first described in Section 5.2, including specifics of the EKF prediction, observation, and update models. Three experiments are conducted to demonstrate the performance of the object detection, recognition, and localization techniques proposed in Chapter 4. Results from these experiments are presented in Section 5.3. Finally, conclusions are made in Section 5.5.

5.2 Architecture

The VisSLAM architecture is shown in the flowchart of Figure 5.1. First, an Environment Recognition system described in Chapter 4 (Section 4.2) is used to determine the environmental context of the navigating vehicle. In this work, the vehicle is constrained to an outdoor park area, where trees are abundant. Once the system is confident that it is indeed located within a park environment, it chooses tree trunks as potential landmarks and initializes the state vector and covariance

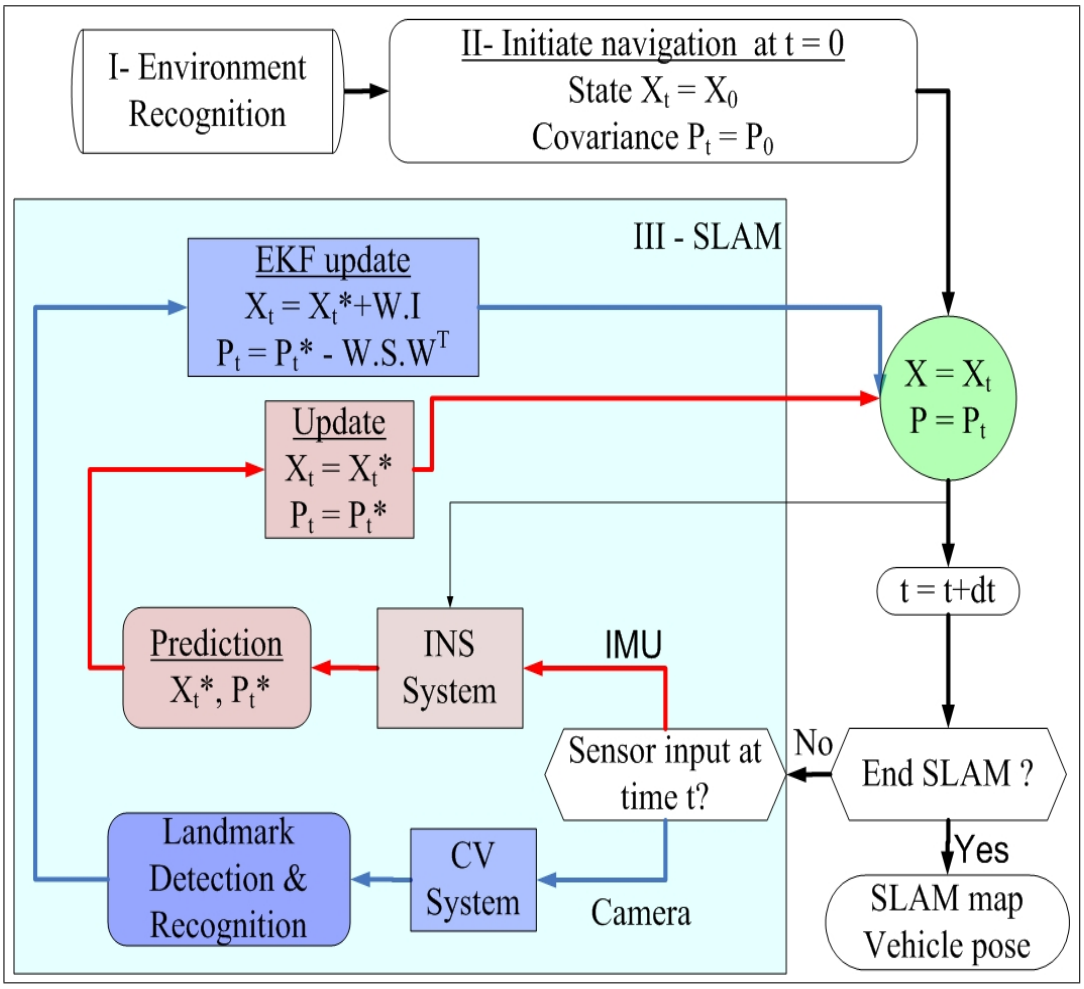


Figure 5.1: Architecture of VisSLAM system. In the first step of VisSLAM, the Environment Recognition system is executed to determine the context of robot milieu in order to know what type of landmarks to search for. Navigation begins and the state vector and covariance matrix are initialized. A prediction is then performed through the INS system of Chapter 3. After each prediction, the system checks if a new image is captured; if so the image is used to detect tree trunks and perform a SLAM update (or initialization); if not the INS predictions for state and covariance are adopted as the new state and covariance.

matrix. VisSLAM does not attempt the kidnapped robot problem[1] [113]; rather, the initial position and heading of the vehicle are calculated from two consecutive

GPS readings at startup:

$$Pos(t_0) = (GPS_x(1), GPS_y(1)) \tag{5.1}$$

$$Heading(t_0) = \arctan(\frac{GPS_y(2) - GPS_y(1)}{GPS_x(2) - GPS_x(1)}), \tag{5.2}$$

where $GPS(1)$ and $GPS(2)$ are two consecutive GPS readings at the time the vehicle initiates navigation. $GPS(1)$ is the GPS reading immediately following the first captured image, since as previously explained in Section 3.3, navigation initiates immediately after the first image is taken. Furthermore, the vehicle travels on a straight line, thereby allowing calculation of initial heading from 2 consecutive GPS readings. The state at startup includes the vehicle position (x, y, z), the vehicle velocity (V_x, V_y, V_z) and the Euler angles (ϕ, θ, ψ). The covariance at startup includes the components of the vehicle states

$$\begin{bmatrix} \sigma_x\sigma_x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y\sigma_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z\sigma_z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{Vx}\sigma_{Vx} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{Vy}\sigma_{Vy} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{Vz}\sigma_{Vz} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\phi\sigma_\phi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\theta\sigma_\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\psi\sigma_\psi \end{bmatrix}, \tag{5.3}$$

where σ stands for the standard deviations of the relevant state at startup and are all initialized to a random small number. σ is chosen as 0.01 here since the initial positions, velocities, and Euler angles are known with high certainty (*i.e.*, low σ). The process of adding and removing landmarks from the state and covariance, also known as map management, is described in Section 5.2.2.

Once the states and covariances are initialized, they are propagated forward in time in a recursive fashion via an EKF, whose specifics are discussed next.

[1]The kidnapped robot refers to the situation where the robot is given no information regarding its initial position and heading.

5.2.1 The EKF System

Extended Kalman Filters are very popular for SLAM in situations where the non-linearities in the process and measurement models are small enough to allow approximating these models by the linear terms of their corresponding Taylor series expansions.

Process model

VisSLAM's process model is based on the equations developed for the INS in Chapter 3, where a prediction of the vehicle motion is made from

$$\hat{x}(k) = F(\hat{x}(k-1), u(k), k) + w(k), \tag{5.4}$$

where $F(.,.,k)$ is the non-linear state transition function and $w(k)$ is the white Gaussian noise associated with this transition function; $\hat{x}$ is the state and u is the process input including accelerometer and gyroscope readings from the IMU.

$$\begin{bmatrix} p^n(k) \\ v^n(k) \\ \psi^n(k) \end{bmatrix} = \begin{bmatrix} p^n(k-1) + v^n(k)\Delta t \\ v^n(k-1)) + [C_b^n(k-1)f(b(k) + g^n]\Delta t \\ \psi^n(k-1) + E_b^n(k-1)w^b(k)\Delta t \end{bmatrix}, \tag{5.5}$$

where p^n, v^n are the position and velocity of the vehicle in the navigation frame, and ψ^n are the Euler angles. More explicitly,

$$\begin{bmatrix} p_x^n(k) \\ p_y^n(k) \\ p_z^n(k) \\ v_x^n(k) \\ v_y^n(k) \\ v_z^n(k) \\ \phi(k) \\ \theta(k) \\ \psi(k) \end{bmatrix} = \begin{bmatrix} p_x^n(k-1) + v_x^n(k)\Delta t \\ p_y^n(k-1) + v_y^n(k)\Delta t \\ p_z^n(k-1) + v_z^n(k)\Delta t \\ v_x^n(k-1) + (C_b^n(1,:)[a_x^b; a_y^b; a_z^b] + g_x)\Delta t \\ v_y^n(k-1) + (C_b^n(2,:)[a_x^b; a_y^b; a_z^b] + g_y)\Delta t \\ v_z^n(k-1) + (C_b^n(3,:)[a_x^b; a_y^b; a_z^b] + g_z)\Delta t \\ \phi(k-1) + E_b^n(1,:)[w_x^b; w_y^b; w_z^b]\Delta t \\ \theta(k-1) + E_b^n(2,:)[w_x^b; w_y^b; w_z^b]\Delta t \\ \psi(k-1) + E_b^n(3,:)[w_x^b; w_y^b; w_z^b]\Delta t \end{bmatrix}, \tag{5.6}$$

where Δt is the time difference between two IMU readings; a^b and w^b are the body-frame referenced vehicle accelerations and rotation rates which are sensed by the IMU accelerometers and gyroscopes respectively; C_b^n is the direction cosine matrix that transforms accelerometer readings from the body coordinate frame to

the navigation frame

$$C_b^n = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}, \tag{5.7}$$

and E_b^n is the matrix used to transform gyroscope readings to Euler angles,

$$E_b^n = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi sec\theta & c\phi sec\theta \end{bmatrix}. \tag{5.8}$$

Observation model

The second source of information used by the EKF are the observations. A stereo camera is mounted on board the navigating vehicle to detect the bearing of landmarks around the vehicle. The observation model is as follows

$$z_i(k) = \mathbf{H}_i(p^n(k), \Psi^n(k), m_i^n(k), k) + v(k), \tag{5.9}$$

where $\mathbf{H}_i$ is the observation model, which is a function of the position $p^n(k)$ and bearing $\Psi^n(k)$ of the vehicle in the navigation frame, as well as the position of the landmarks $m_i^n(k)$ in the navigation frame; $v(k)$ is the zero-mean observation noise error with covariance R. R is dependent on the quality and resolution of the camera and is approximated in an empirical fashion in Section 5.3.1. The observation model $\mathbf{H}_i$ is expressed as

$$\mathbf{H}_i(\hat{x}(k|k-1) = \begin{bmatrix} \varphi_i \\ \vartheta_i \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{y^c}{x^c}\right) \\ \arctan\left(\frac{z^c}{\sqrt{(x^c)^2+(y^c)^2}}\right) \end{bmatrix}, \tag{5.10}$$

where φ_i and ϑ_i are the predicted azimuth and pitch angles of the landmarks with respect to the camera. Pitch is usually included in the observation model for aerial-based SLAM; however, for land-based SLAM it is sufficient to use the azimuth since the vehicle is constrained to navigate on the ground. The observation model is therefore reduced to:

$$\mathbf{H}_i(\hat{x}(k|k-1) = \begin{bmatrix} \varphi_i \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{y^c}{x^c}\right) \end{bmatrix}, \tag{5.11}$$

x^c and y^c, are the 2D coordinates of landmarks in the camera reference frame and are determined from

$$p^c_{mc} = \begin{bmatrix} x^c \\ y^c \end{bmatrix} = C^c_b C^b_n [m^n_i - p^n - C^n_b p^b_{cb}], \tag{5.12}$$

where p^b_{cb} is the position of the camera lens with respect to the body coordinate frame center (*i.e.*, inertial sensor location); m_i and p^n are the respective position of a landmark 'i' and the vehicle in the navigation frame. The expression between the square brackets in (5.12) represents the distance between the landmark and camera Δ_{LC} expressed in the navigation frame. C^b_n is the matrix expressing the distance Δ_{LC} in the body frame (Figure 5.3). C^b_n is in effect the inverse of the direction cosine C^n_b, and can be simply calculated from the transpose of C^n_b since only rotations are involved in this transformation; p_n is the position of the vehicle in the navigation frame and m^n_i is the position of a landmark 'i', once initialized, in the navigation frame.

Equations (5.11) and (5.12) present an expression for the *expected* bearing (azimuth) of a landmark as seen from the camera. The difference between the expected and *actual* bearing measurements yields a measure called *Innovation*, which is used in the update stage of the EKF to improve the state and covariance estimates. C^c_b transforms Δ_{LC} from the body frame to the camera frame. During vehicle motion, the camera pans back and forth in increments of $Pi/4$ between $+Pi/2$ and $-Pi/2$ in the body reference frame (white frame in Figure 5.2), taking an image at each increment. This is done to increase the range view of the navigating vehicle. Since the camera rotates throughout the vehicle motion, the position of the camera lens depends on the camera shot. For simplicity, the center of PTU Yaw axis is taken as the camera origin rather than the actual camera center. This amount to an error of 6 cm at most in the position of the camera origin.

Let α represent the camera angular rotation with respect to the Z axis of the body frame. The camera center offset, seen in Figure 5.2, is 5 cm in the x direction, 0 cm in the y direction and -20.6 cm in the z direction. Therefore, C^b_c is best expressed in homogenous coordinates as follows:

$$C^b_c = \begin{bmatrix} c\alpha & -s\alpha & 0 & 0 \\ s\alpha & c\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & +0.05 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -0.206 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

or

$$C_c^b = \begin{bmatrix} c\alpha & -s\alpha & 0 & 0.05 \\ s\alpha & c\alpha & 0 & 0 \\ 0 & 0 & 1 & -0.206 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.14}$$

where C_b^c and C_n^b are the inverse of C_c^b and C_b^n respectively.

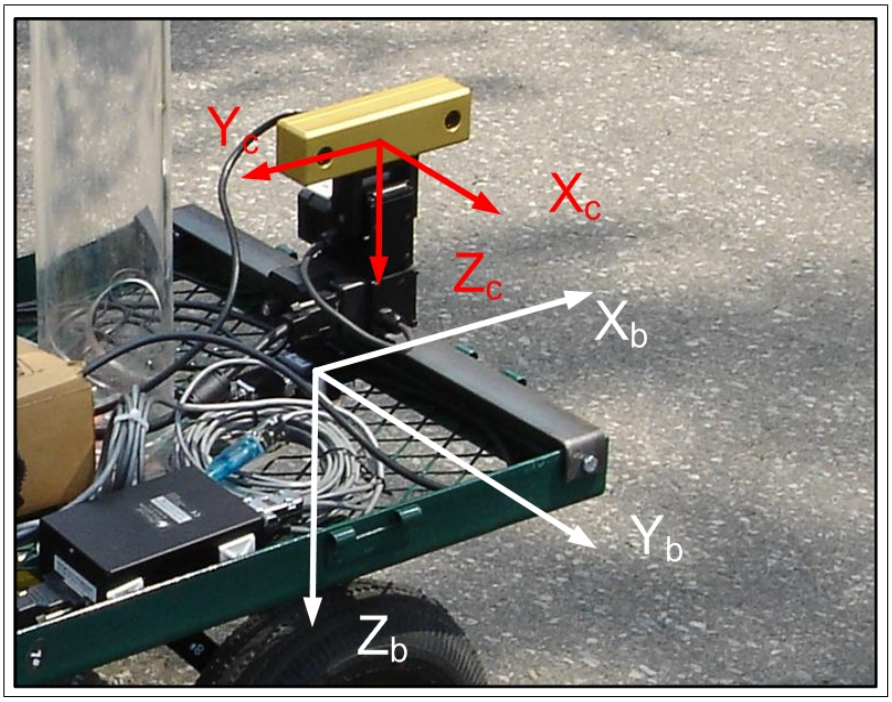


Figure 5.2: Body and camera coordinate frames. Subscripts b and c stand for the body and camera frames respectively.

The actual bearing measurements (φ_a and ϑ_a) as seen from the camera are calculated based on their image coordinates as follows

$$z_i(k) = \begin{bmatrix} \varphi_a \\ \vartheta_a \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{u-u_0}{f_u}\right) \\ \arctan\left(\tan\left(\frac{v-v_0}{f_v}\right)\cos\varphi\right) \end{bmatrix}. \tag{5.15}$$

Since the system is only concerned with the azimuth, the observation only includes:

$$z_i(k) = [\varphi] = \left[\arctan\left(\frac{u-u_0}{f_u}\right)\right], \tag{5.16}$$

where u_o, v_0 are the coordinates of the image center in pixels, u, v are the coordinates of the landmarks in pixels and f_u and f_v are the focal lengths in u pixels and v pixels respectively (Figure 5.4). The focal length for the camera is 4mm or 514.92 pixels at a resolution of 640 x 480 pixels (See camera specifications in Appendix A.1.

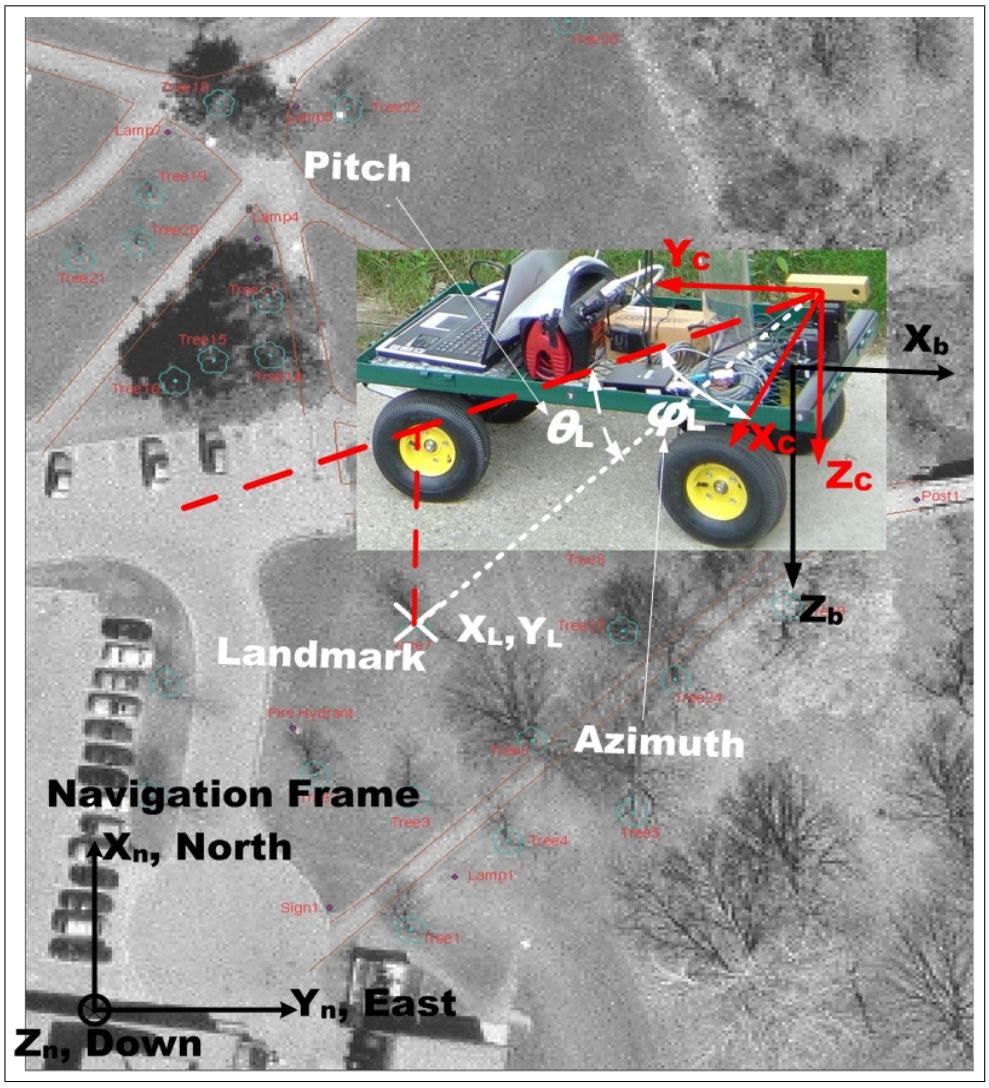


Figure 5.3: Coordinate frames used for INS dead-reckoning. Subscripts n, b and c stand for the navigation, body and camera frames respectively. X_L and Y_L are the coordinates of the landmarks.

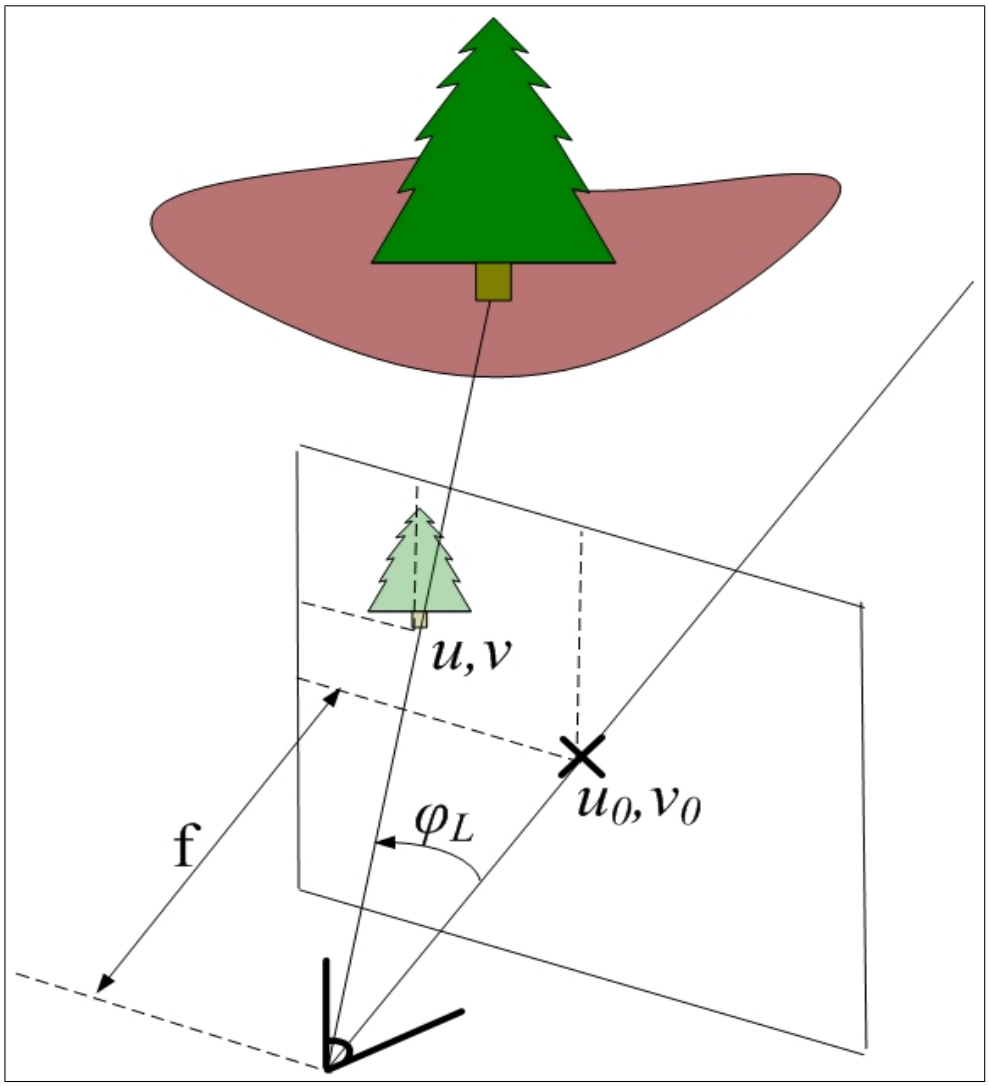


Figure 5.4: Calculating bearing from an image. f is the focal length of the camera, u and v are the coordinates of the image of the base of the tree. u_0 and v_0 are the coordinates of the center of the image, and φ_L is the azimuth angle to the landmark.

The EKF recursive process consists of a prediction (INS), an observation (tree bearing from camera), and an update (EKF equations). The mathematical details of these three steps are presented next.

Recursive process

A SLAM prediction involves more work than an INS prediction since it also must take into account the prediction of correlations between the vehicle state and map.

Prediction:

$$\mathbf{P}(k|k-1) = \mathbf{J}_x \mathbf{P}(k-1|k-1)\mathbf{J}_x^T + \mathbf{J}_w \mathbf{Q} \mathbf{J}_w^T, \tag{5.17}$$

where $\mathbf{J}_x$ and $\mathbf{J}_w$ are the Jacobians of the state transition function with respect to the state vector and noise input respectively. $\mathbf{J}_x$ and $\mathbf{J}_w$ are calculated numerically as shown in Appendix B.2. Equation (5.19) can be written more explicitly as:

$$\begin{bmatrix} \mathbf{P}^-_{vv}(k) & \mathbf{P}^-_{vm}(k) \\ \mathbf{P}^{-T}_{vm}(k) & \mathbf{P}^-_{mm}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{J}_v \mathbf{P}^+_{vv}(k-1)\mathbf{J}_v^T + Q_{vv} & \mathbf{J}_v \mathbf{P}^+_{vm}(k-1) \\ \mathbf{J}_v^T(k)\mathbf{P}^+_{vm}(k-1) & \mathbf{P}^+_{mm}(k-1) \end{bmatrix}, \tag{5.18}$$

where the diagonal entries in (5.18) represent the variances in the vehicles state and map and the off-diagonal represent the correlations between the vehicle and map uncertainties. The update step occurs every time the system observes a landmark and recognizes it from a previous sighting.

Update:

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{W}(k)\nu(k) \tag{5.19}$$

where

$$\begin{aligned} \nu(k) &= z_i(k) - \mathbf{H}_i(\hat{x}(k|k-1)) && (5.20) \\ \mathbf{W}(k) &= \mathbf{P}(k|k-1)\mathbf{J}[\mathbf{H}_x^T(k)]\mathbf{S}^{-1}(k) && (5.21) \\ \mathbf{S}(k) &= \mathbf{J}[\mathbf{H}_x(k)]\mathbf{P}(k|k-1)\mathbf{J}[\mathbf{H}_x(k)]^T + \mathbf{R} && (5.22) \end{aligned}$$

where $\mathbf{J}[\mathbf{H}_x(k)]$ is the Jacobian of the observation model with respect to the state vector. $\mathbf{J}[\mathbf{H}_x(k)]$ is calculated numerically as shown in Appendix B.2.

The state covariance is finally updated as follows

$$\mathbf{P}(k|k) = \mathbf{P}(k|k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^T(k). \tag{5.23}$$

If a landmark is observed for the first time, care must be taken in the fashion in which it is added into the SLAM map. This process is known as map management.

5.2.2 Map management

Once a landmark satisfies the initialization criterion (explained in Section 4.5), it is initialized in both the SLAM state and covariance. The following equations are based on those presented by Williams [92]. Each new feature is initialized via a feature initialization model $g_i(.,.)$ as follows:

$$\hat{\mathbf{x}}_i^+ = g_i(\hat{\mathbf{x}}_v^-(k), \mathbf{z}(k)), \tag{5.24}$$

where $\hat{\mathbf{x}}_i^+$ is the estimated feature coordinates when it is initialized, $\hat{\mathbf{x}}_v^-$ is the current vehicle state estimate before the observation, and $\mathbf{z}(k)$ is the observation of the new landmark. g_i is the initialization function obtained from the stereo system:

$$\begin{bmatrix} \hat{x}_i^+(k) \\ \hat{y}_i^+(k) \end{bmatrix} = \begin{bmatrix} \hat{x}_v^-(k) + z_r(k) * cos(\hat{\psi}_v^-(k) + z_\theta(k)) \\ \hat{y}_v^-(k) + z_r(k) * sin(\hat{\psi}_v^-(k) + z_\theta(k)) \end{bmatrix}, \tag{5.25}$$

where $z_r(k)$ and $z_\theta(k)$ are the depth and bearing of the observed landmark at time k. The new state $\hat{\mathbf{x}}_i^+$ is then appended to the current state as follows:

$$\mathbf{x}_{aug} = \begin{bmatrix} \mathbf{x}_{old} \\ \hat{\mathbf{x}}_i^+ \end{bmatrix}, \tag{5.26}$$

where $\mathbf{x}_{old}$ is the SLAM state vector before the new landmark is initialized. At this point, the covariances of the new feature estimates must also be initialized. The manner in which this is done is critical because the initial landmark estimate is dependent on the uncertainty of the current vehicle pose and is therefore correlated to the vehicle pose and landmark position uncertainties. Given that the current covariance matrix $P^-(k)$ is

$$P^-k = \begin{bmatrix} P_{vv}^-(k) & P_{vm}^-(k) \\ P_{vm}^{-T}(k) & P_{mm}^-(k) \end{bmatrix}, \tag{5.27}$$

The covariance is updated via two steps. In the first step the covariance is augmented with the observation covariance, $R(k)$ as follows:

$$P^{*-}(k) = \begin{bmatrix} P_{vv}^-(k) & P_{vm}^-(k) & 0 \\ P_{vm}^{-T}(k) & P_{mm}^-(k) & 0 \\ 0 & 0 & R(k) \end{bmatrix}, \tag{5.28}$$

In the second step, the covariance is corrected by projecting $P^{*-}(k)$ through the Jacobian $J_x g(k)$ of the initialization function g_i with respect to the states,

$$P^{+}(k) = J_x g(k) P^{*-}(k) J_x g^T(k), \tag{5.29}$$

The Jacobian J_x is calculated numerically as shown in Appendix B.2.

Pruning of landmarks is done based on the expectancy of their detection. Landmarks that are within the *detectable* range and bearing of the camera and are not detected are marked as missed. Recalling in Section 4.3.2 that the line detection system prunes lines that are shorter than fifteen pixels long. Fifteen pixels corresponds to a line (or tree trunk) measuring 1 meter in height, viewed at a depth of 34 meters. Given this constraint, the maximum detectable range is set to 30 meters. Landmarks that are consecutively missed for three times are marked as 'false' and removed from the state and covariance matrix. The SLAM state is updated by deleting the coordinates of the 'false' landmark, and SLAM covariance is updated by deleting the row and column of the corresponding 'false' landmark.

Although VisSLAM uses a stereo camera to initialize landmarks into the SLAM system, it is in effect a bearing-only SLAM system where only bearing information is used as observations to update the EKF. Initialization could alternatively be made in the standard delayed or undelayed bearing-only initialization techniques discussed in Section 2.3 but is done using stereo because conditions for bearing-only initialization are poor. More specifically, bearing initialization may be ill-conditioned due to three factors : (1) the pose estimates are not precise, (2) the uncertainty in the bearing measurements, and (3) the minimum angle between two sightings. The first two conditions are inherent of the SLAM problem and nothing can be done to remedy these issues. The problem of minimum angle is the issue that is addressed by researchers in Bearing-only Vision SLAM, by waiting until such angles are satisfied. In land-based INS, the system is not tolerant of long wait periods and diverges quickly if the errors of the IMU are not bounded quickly. For this reason, range-bearing initialization is preferred for VisSLAM.

No system is complete before it is tested in a veridical setting, where its performance is compared to established ground truth. Towards this end, in the next section VisSLAM is tested on several data sets collected during real experimental runs.

5.3 Experiments

VisSLAM is tested on a data set (Section A.3) of time-stamped GPS, IMU and images collected during an experimental run of a vehicle navigating in a park setting (Section A.2). VisSLAM is evaluated based on (1) the agreement of VisSLAM state estimate to ground truth, (2) the covariance values for the position of landmarks, (3) the covariance values during navigation for position and bearing of the vehicle, and finally (4) the innovations calculated at each observation.

This information is compiled for three scenarios. In the first scenario, trees are automatically detected but manually recognized and initialized into the SLAM state and covariance. The aim here is to investigate the efficiency of the landmark detection part of VisSLAM and isolate the issues of landmark recognition and initialization by performing them manually. In the second scenario, the system automatically detects trees and also automatically recognizes them using a technique proposed in Section 4.4. In this test, landmarks are still initialized manually using the known ground truth coordinates for each landmark. These first two tests should really be classified as localization problems rather than SLAM problems because of the fashion in which the landmarks are initialized. In the third and final scenario, VisSLAM performs SLAM while automatically detecting, recognizing, and initializing trees.

5.3.1 Automatic detection, manual recognition and initialization

The first step in these experiments is to estimate the values of the sensor variances used to predict and update the VisSLAM EKF and to estimate the variance (or confidence) of the non-holonomic model. Towards this end, several values for the above variances are proposed and the resulting performance of VisSLAM is evaluated; the values that result in the closest agreement of the VisSLAM map to ground truth (highlighted row in Table 5.1) are chosen for the remainder of the experiments that follow. In this parameter tuning phase, all trees are detected automatically, but are manually recognized and initialized by hand labeling each tree during the entire run and initializing its position to its known ground truth location. Figure 5.5 shows the SLAM map and vehicle motion obtained using these selected variances. The system performs well at estimating the SLAM state (*i.e.* vehicle pose and positions of landmarks). Notice how closely the outbound and inbound paths follow the center of the pathways on the map. Furthermore, notice how the landmark estimates (white dots) remain for the most part of them at their true

locations (*i.e.*, middle of green patches). It is true that the landmarks are initialized manually but nevertheless, the fact that they remain at their position during SLAM indicates the stability of VisSLAM and the efficiency of the tree detection system at bounding the IMU dead-reckoning errors.

	IMU		Camera	Constraints
Test	acc (m/s^2)	gyro $(^{rad}/s)$	Azimuth (rad)	V(m/s)
1	0.12	0.3	0.2	0.3
2	0.16	0.06	0.02	0.1
3	0.05	0.05	0.1	0.1
4	0.1	0.03	0.07	0.07

Table 5.1: Covariance tuning for IMU, camera, and non-holonomic model. The values shown represent the expected standard deviations of each sensor, and the corresponding variances are the square of these entries.

Figure 5.7 shows vehicle covariance estimates for the position of the vehicle during the run shown in Figure 5.5. One interesting observation is the brisk jump of the vehicle position near 'T13', which occurs because the camera In the worse case scenario, the variance in the x direction reaches $20m^2$ which corresponds to a standard deviation of approximately 4.6 meters in the x position (at t = 330 seconds). This peak corresponds to the final bend in Figure 5.5 near Tree 16 (T16). A closer observation of this region of the test site reveals that the ground is inclined beyond the normal ground inclination of the rest of the test site. Therefore, the error is probably due to an incorrect estimate of the vehicle pitch, which in turn results in a over-estimation of the gravity component. This error would appear to the INS system as if the vehicle were accelerating backwards, where in fact it is undergoing forward acceleration. The highest covariance in the y direction is approximately 8 m^2 (std = 2.8m) and occurs at 200 seconds, which corresponds to the region right after the U-turn at the bottom left of the image in Figure 5.5 where the estimated vehicle trajectory slightly looses control (green line goes off track). This large variance is due to the deficiency of the non-holonomic slip model at regions where high rotation rates occur and some lateral movement does in fact exist. VisSLAM immediately recovers and re-stabilizes after this. Figure 5.5 is repeated from Figure 3.11 for the purpose of getting better perspective on the position of the vehicle at each time step.

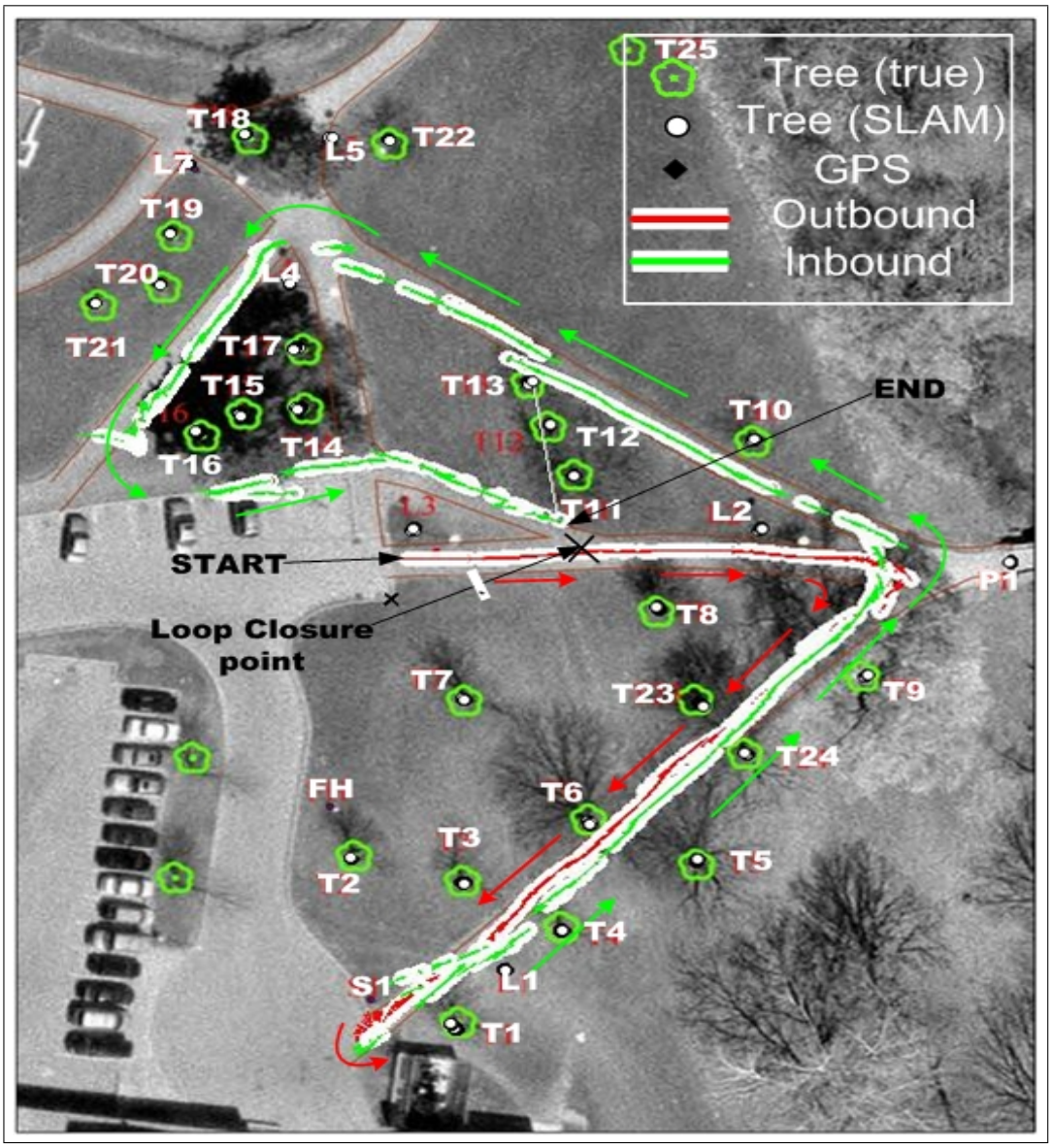


Figure 5.5: VisSLAM map developed during Experiment 'one' (auto tree detection, manual recognition and initialization). 'T' refers to a Tree, 'L' refers to a Lampost, 'P' refers to a Post, 'S' refers to a Sign, and 'FH' refers to a Fire Hydrant. 'Start' is the starting point of the vehicle, 'end' and 'loop closure point' are the true and estimated (via VisSLAM) location of the vehicle at the point of loop closure. The red and green arrows are overlaid on the map to simply help follow the trajectory of the vehicle from 'Start' to 'End'.

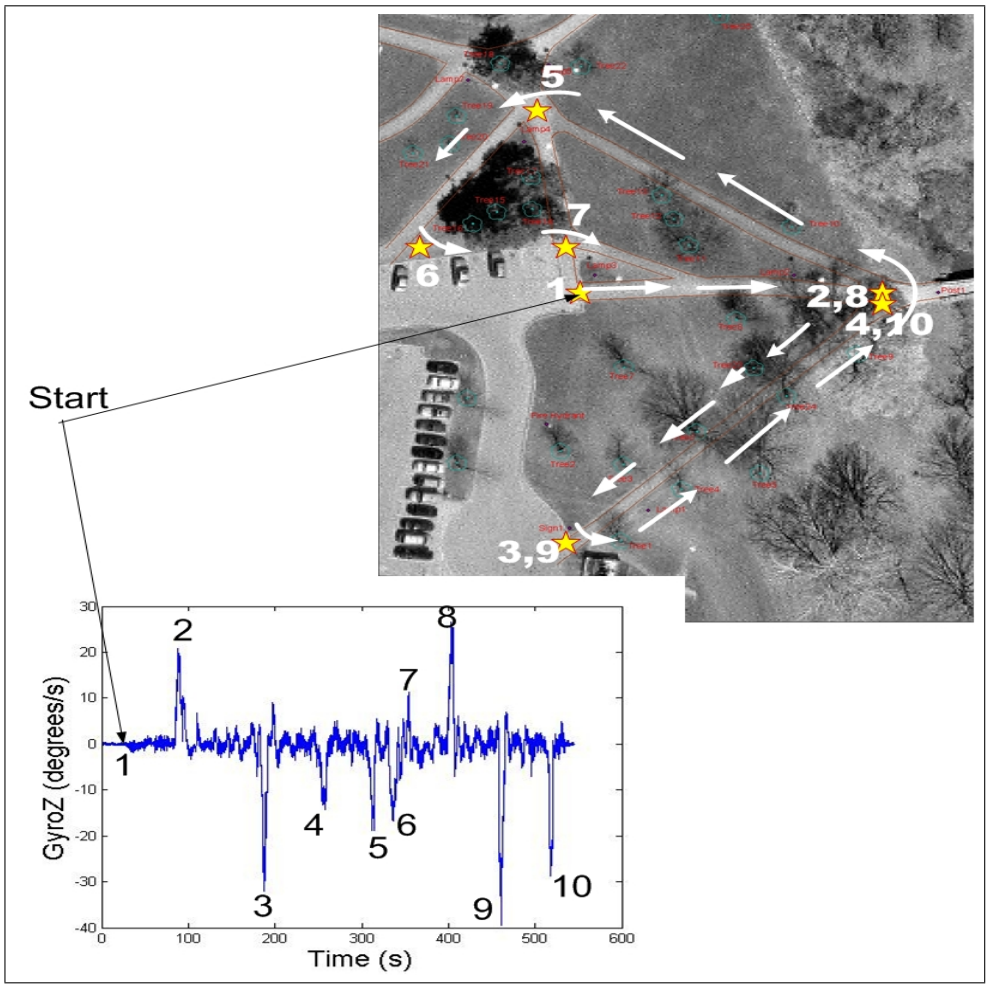


Figure 5.6: Correlating the Gyro readings in the Z direction to their corresponding locations on the ortho-referenced image of the test site. Experiment performed on data set 1 of the database of experimental data. Figure 3.11 Chapter 3.

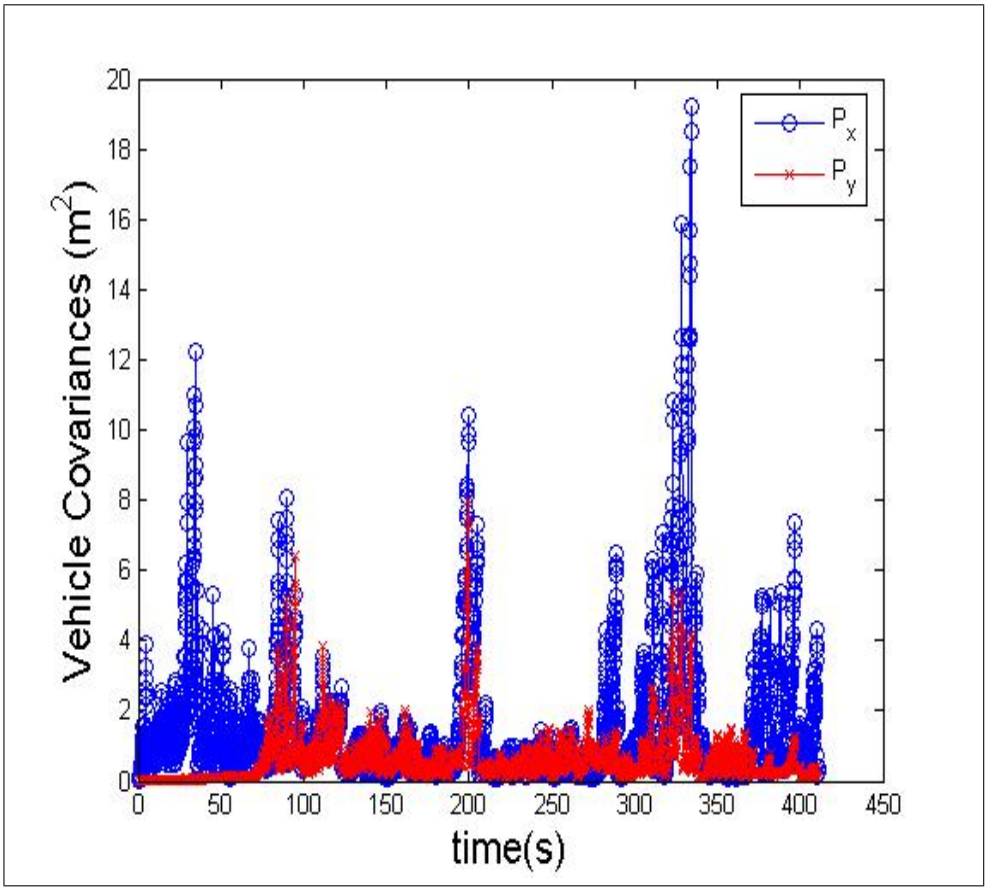


Figure 5.7: Covariance for vehicle position for Experiment 'one' (auto tree detection, manual recognition and initialization). P_x and P_y are the covariances in the x and y directions respectively.

Figure 5.8 shows the variance of the vehicle heading direction during the experiment. This value is small throughout the journey of the vehicle, indicating the continued large confidence in the heading direction. Indeed, observation of the vehicle motion in Figure 5.5 shows good turning predictions at all bends. Note that even at the 'T16' tree the vehicle correctly rotates (look at the orientation of the tail of the vehicle) but moves backwards rather than forwards after the rotation.

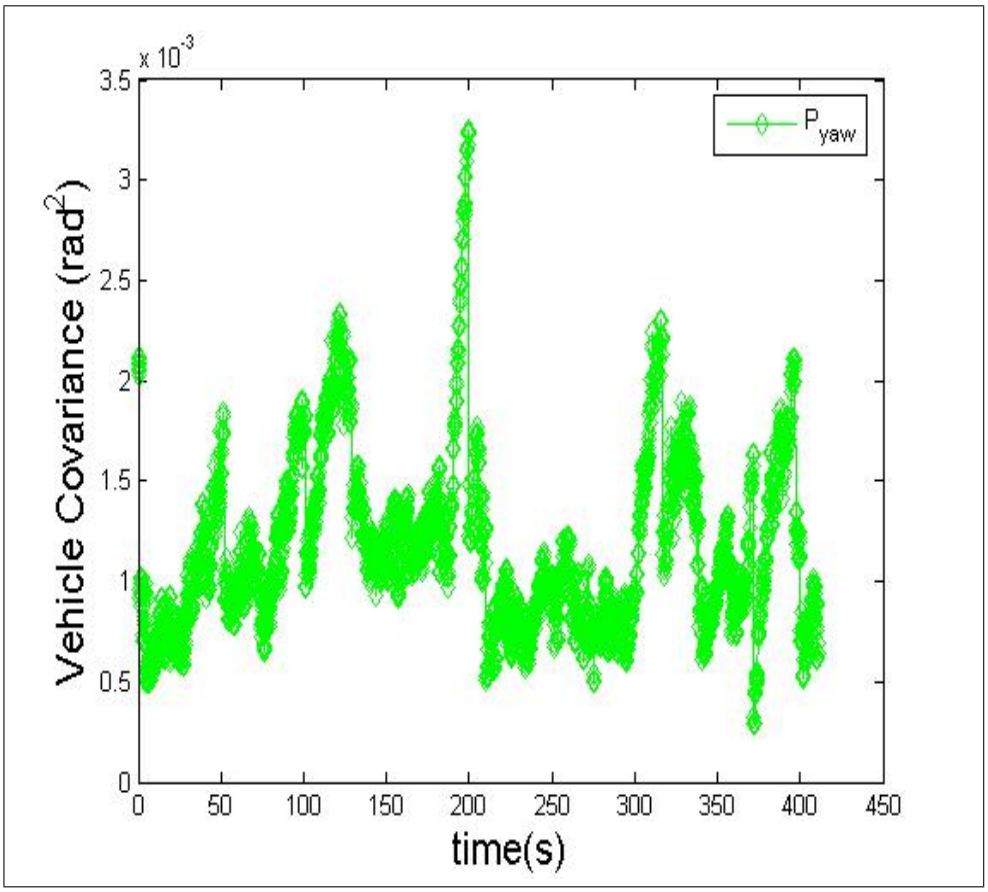


Figure 5.8: Covariance for vehicle bearing for Experiment 'one' (auto tree detection, manual recognition and initialization). P_{yaw} is the covariance in the heading of the vehicle.

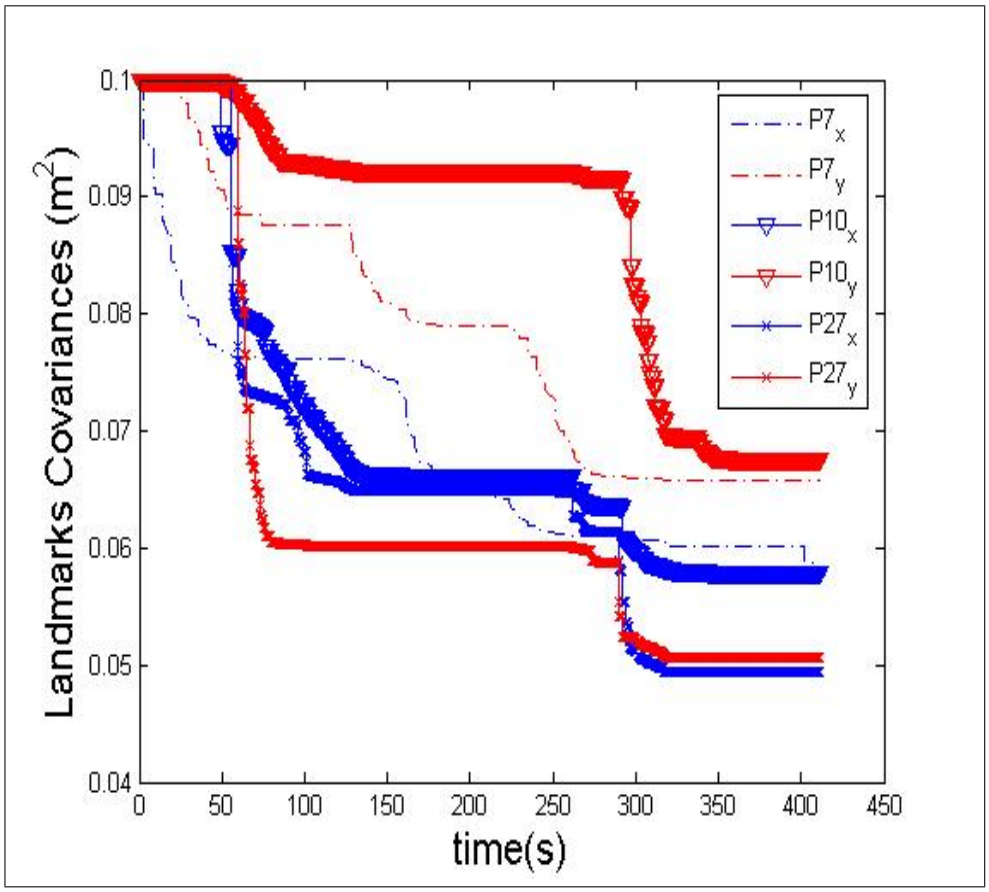


Figure 5.9: Covariance for the position of three landmarks 'T7', 'T10', and 'T27' for Experiment 'one' (auto tree detection, manual recognition and initialization). Pk_x and Pk_y are the covariances for Landmark 'k' in the x and y directions respectively.

Finally, in Figure 5.10, the Innovation of landmark bearing is shown throughout the entire run, where the values fluctuate between plus or minus 0.1 rad (5.7 degrees) with occasional peaks at 0.25 to 0.5 rad (14.32 to 28.64 degrees) and very few peaks at 1 rad (57.29 degrees). These peaks occur in areas where the vehicle does not detect enough landmarks and must count on its dead-reckoning capabilities. When a landmark is eventually detected, the innovations are high and result in the observed peaks.

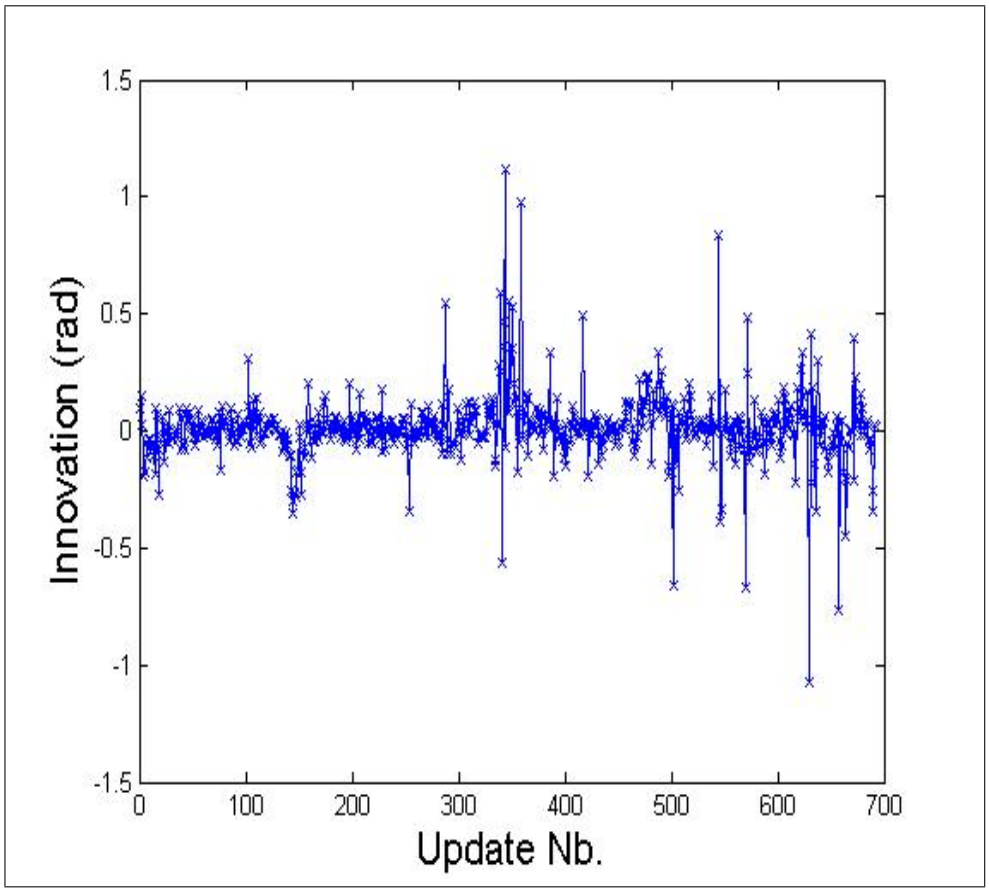


Figure 5.10: Innovation for the bearing of landmarks during Experiment 'one' (auto tree detection, manual recognition and initialization). Update Nb. stands for the number of innovations.

In Experiment 'two' the manual recognition is replaced by the VisSLAM recognition system. Landmarks are still manually initialized.

5.3.2 Automatic detection and recognition; manual initialization

The objective of the second experiment is to investigate the effectiveness of the tree recognition system developed in Section 4.4. In Figure 5.11, although the SLAM map is not as close to ground truth as in the first test, the vehicle trajectory is still well estimated. The exception is around tree 'T13' where the vehicle follows a

false trajectory and then rectifies itself. This error is due to the low number of tree trunks that are recognized by the automatic recognition system in this part of the map. A low recognition rate implies a low number of EKF updates and insufficient bounding of the INS errors. A similar error occurs at loop closure, where the system reverses track instead of heading to its correct destination.

The de-tracking issue at tree 'T13' is evidenced in Figure 5.12, where the vehicle position covariance exhibits a high impulse of approximately 100 m^2 (at 280 seconds), corresponding to a standard deviation of 10 meters. An important observation in this figure is that the covariance of the vehicle position is high at startup and then reduces gradually as more and more landmarks are initialized into the SLAM state.

In Figure 5.13, the covariance in vehicle heading is still low (average of 0.002 rad^2); albeit, not as low as the first experiment (average of 0.0012 rad^2). The heading covariance in heading is not affected due to the fact that the vehicle remains on a straight path for the majority of the journey with six bends. The highest orientation covariance occurs at the bottom left of the map, when the vehicle performs a U-turn.

Figure 5.14 shows the covariance variation for three landmarks during SLAM. As expected, the covariance in landmark position reduces due to increasing confidence in their position after several sightings. The difference between between this and the previous results is that in this case the landmarks covariances reduce at a slower rate. Notice that the final covariances here are not as low as those in Figure 5.14. This result is expected since all landmarks, including these three ('T7', 'T10', and 'T27') are recognized fewer times during the journey, resulting in less reduction of covariance.

Looking at Figure 5.15, it is apparent that a much smaller number of innovations occur than in the previous experiment (270 verses 700), reflecting the fact that many less landmarks are recognized than in the manual recognition case. Less landmarks signifies less SLAM updates and a less stable system.

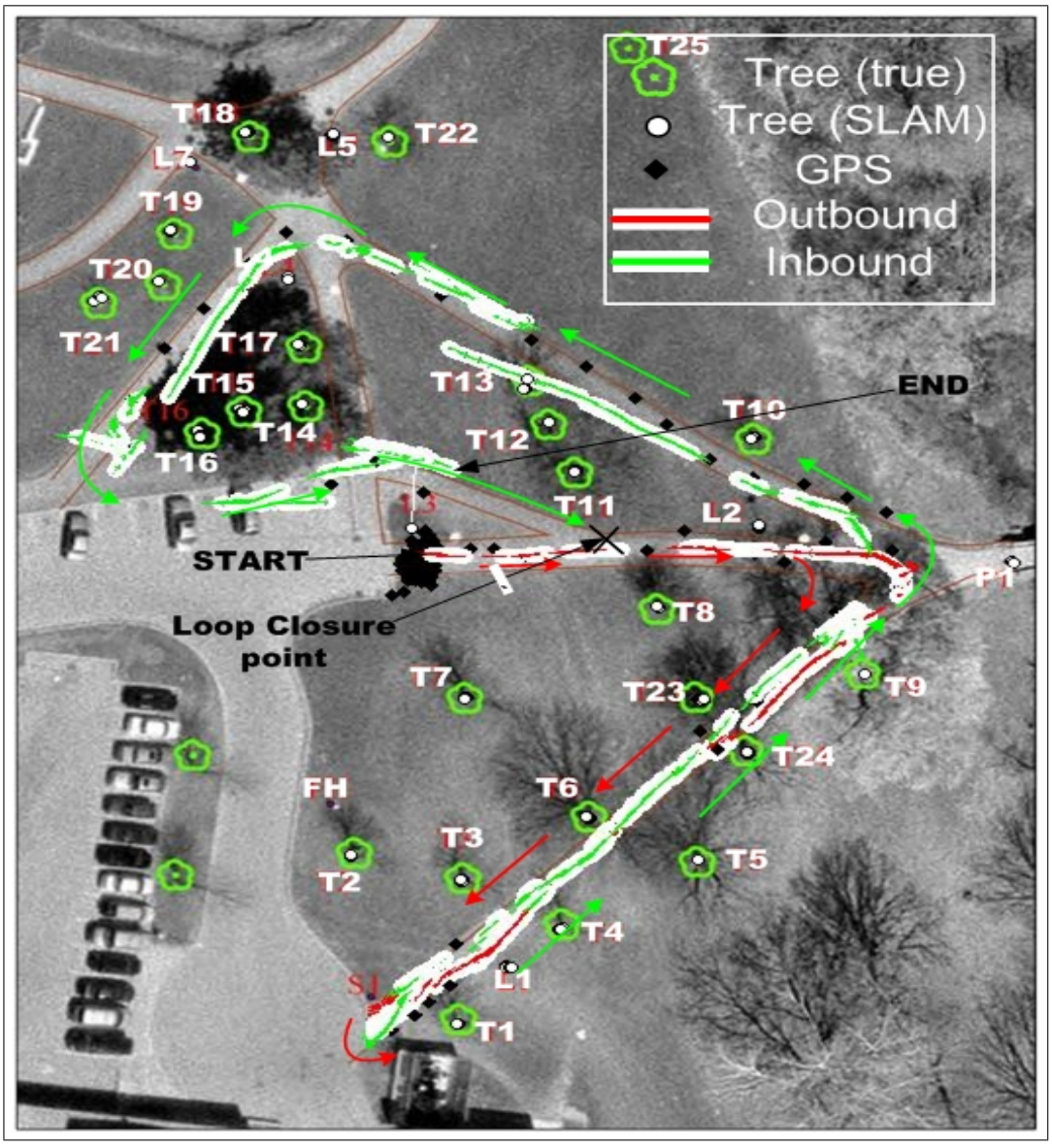


Figure 5.11: VisSLAM map developed during Experiment 'two' (auto tree detection and recognition, manual initialization). 'T' refers to a Tree, 'L' refers to a Lampost, 'P' refers to a Post, and 'FH' refers to a Fire Hydrant. The red and green arrows are overlaid on the map to simply help follow the trajectory of the vehicle from 'Start' to 'End'.

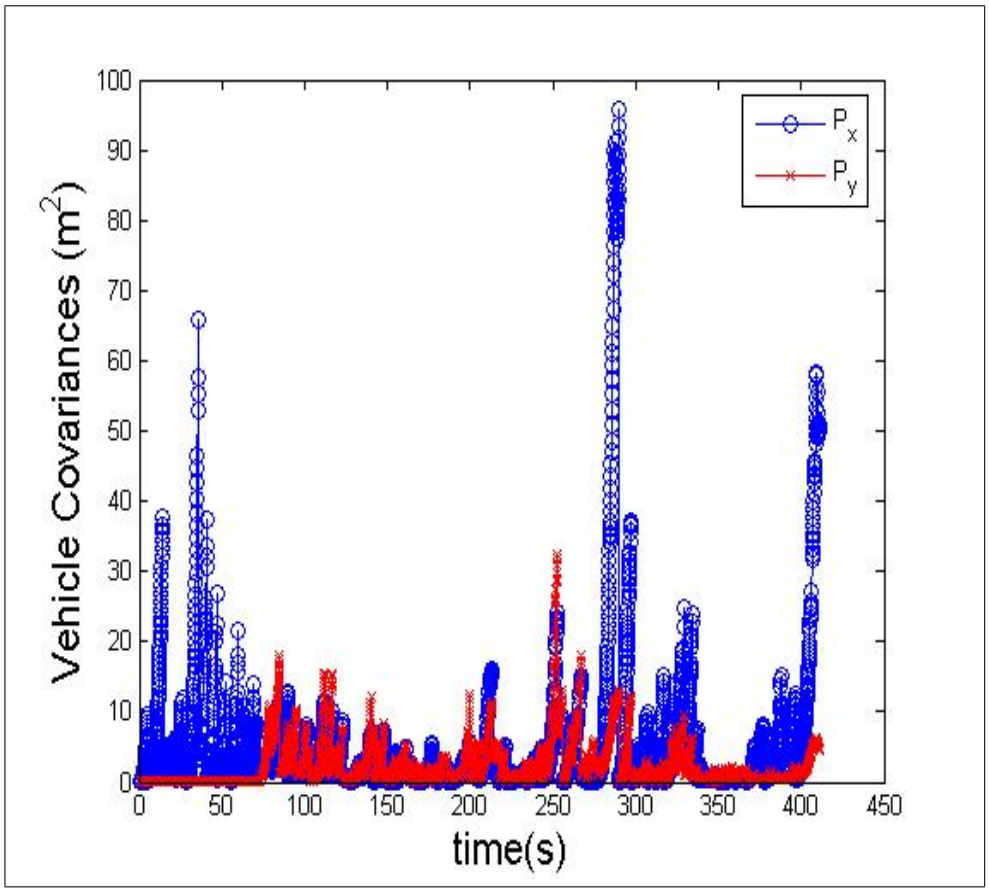


Figure 5.12: Covariance for vehicle position for Experiment 'two' (auto tree detection and recognition, manual initialization). P_x and P_y are the covariances in the x and y directions respectively.

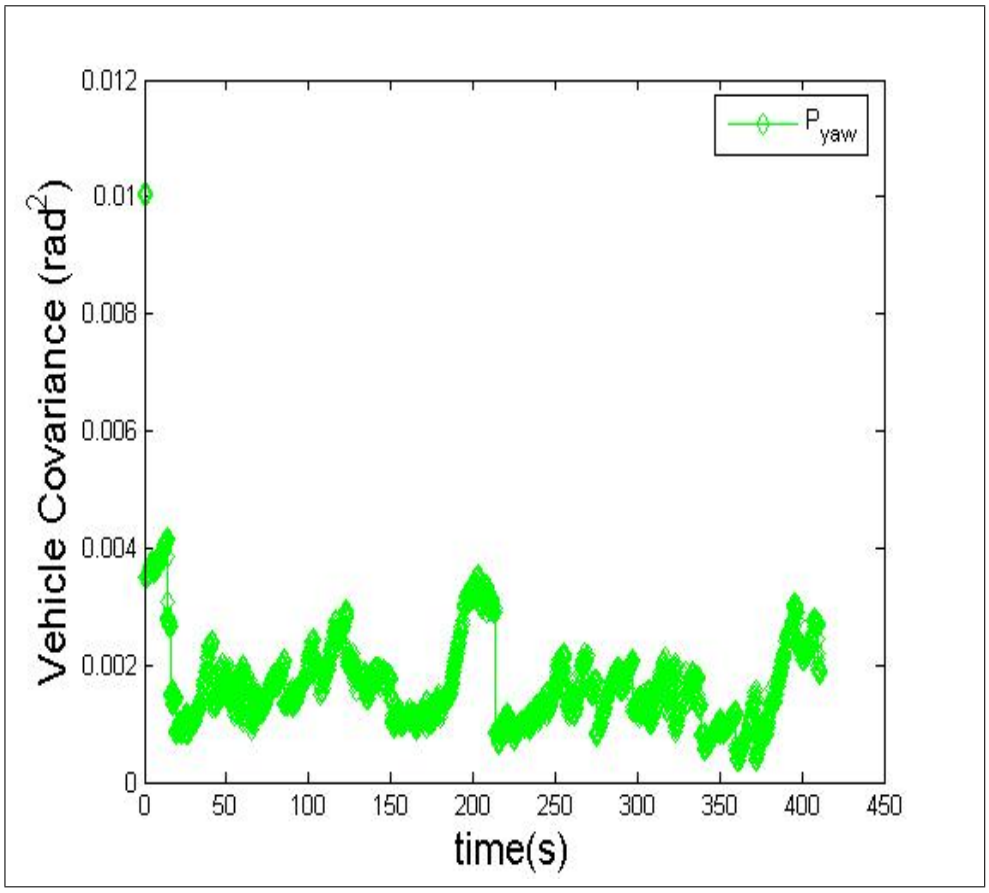


Figure 5.13: Covariance for vehicle bearing for Experiment 'two' (auto tree detection and recognition, manual initialization). P_{yaw} is the covariance in the heading of the vehicle.

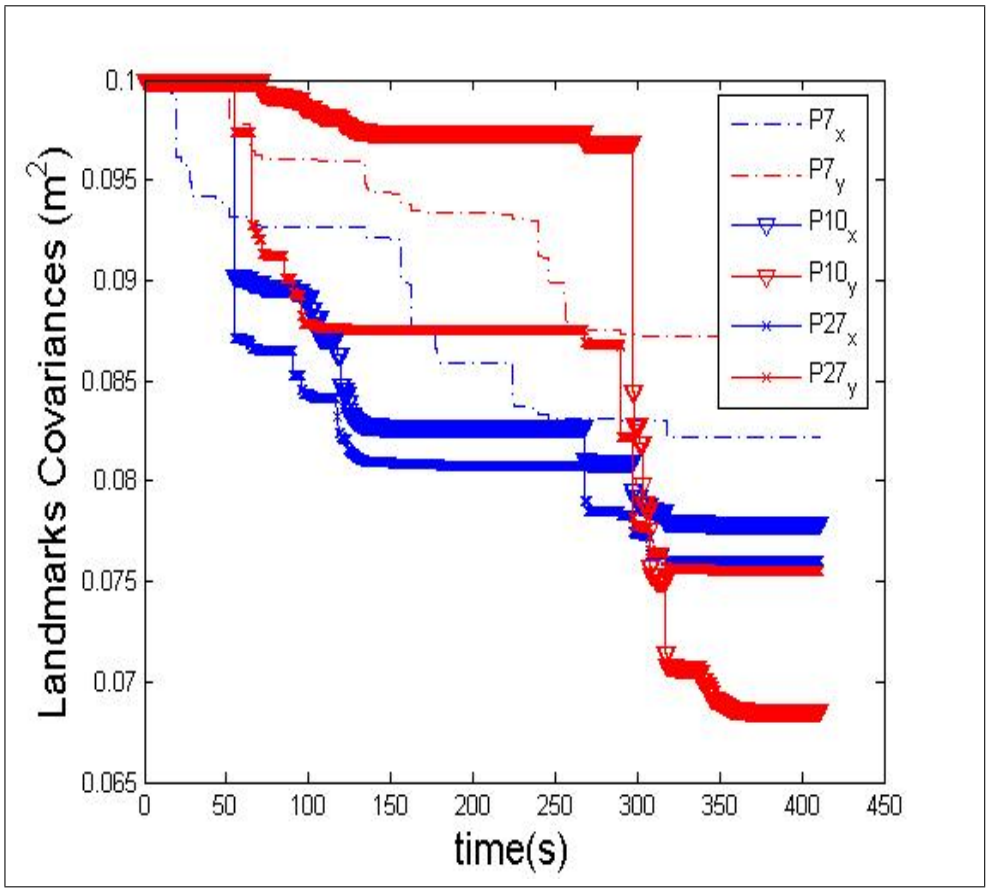


Figure 5.14: Covariance for the position of three landmarks 'T7', 'T10', and 'T27' for Experiment 'two' (auto tree detection and recognition, manual initialization). Pk_x and Pk_y are the covariances for Landmark 'k' in the x and y directions respectively.

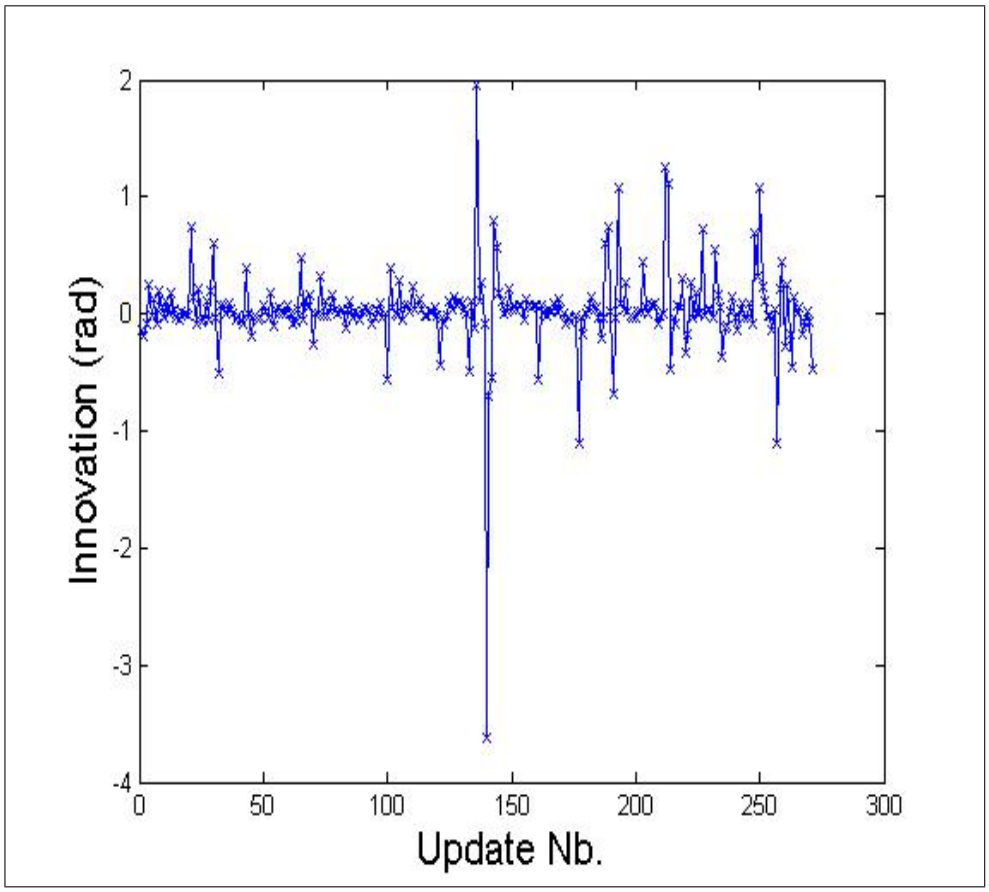


Figure 5.15: Innovation for the bearing of landmarks during Experiment 'two' (auto tree detection and recognition, manual initialization). Update Nb. stands for the number of innovations.

These first two test scenarios demonstrate the successful application of the Computer Vision and inertial systems developed above to localization. In the third and final test scenario presented in the next section, the systems are evaluated based on their ability to perform SLAM.

5.3.3 Automatic detection, recognition, and initialization

In this last test, VisSLAM automatically detects, recognizes, and initializes trees. Landmark initialization is performed using depth information extracted from a stereo camera (Section 4.5). Since the stereo system only initializes trees that enter

into the range of the camera, a smaller number of landmarks are initiated into the SLAM state during the journey of the mobile platform. The SLAM map shown in Figure 5.16 shows a degeneration in the results of VisSLAM from the previous two tests. Landmarks are no longer initialized at their correct position, which results in eventually destabilizing the EKF. In the outbound journey (red and white) VisSLAM faithfully tracks the correct vehicle trajectory while initializing landmarks into SLAM. Many of the landmarks (*e.g.*, 'L2', 'T9', 'T24') are initialized at their correct positions. Those landmarks falsely initialized (*e.g.*, 'T3', 'T4', 'L1') appear to slowly converge towards their correct positions. On the inbound journey, VisSLAM is de-stabilized near 'T9' and the INS dead-reckoning is lacking. The reason for this failure is primarily caused by not initializing 'P1', which in the previous tests contributed to the stability of VisSLAM around the bend after tree 'T9'. The system does not recover after that and the SLAM filter diverges. This weakness in landmark initialization suggests that either an alternate method is necessary or the current method should be aided by a second initialization technique.

Figures 5.17 and 5.18 show the vehicle position and bearing covariance verses time. The peaks at approximately 270 seconds and beyond indicate the divergence of the SLAM filter, where it can no longer estimate vehicle position with high confidence. In fact, the uncertainty in vehicle position covariance grows in both x and y directions.

Figure 5.19 shows the covariance variation for the same three landmarks as in the previous two experiments ('T7', 'T10', and 'T27'). The first observation is that 'T7' covariance does not change. This is simply due to fact that 'T7' is never initialized throughout the vehicle journey. The position covariance of 'T10' reduces much later than in the previous tests because it is not initialized until a later time in the inbound journey.

In Figure 5.20, innovations are plotted at all EKF updates. The first observation is that a much smaller number of innovations than before (175 versus 260 and 700), caused by fewer landmarks initialized and subsequently observed. Second, notice that many spikes exist, which is indicative of the much longer time the system uses it dead-reckoning estimation.

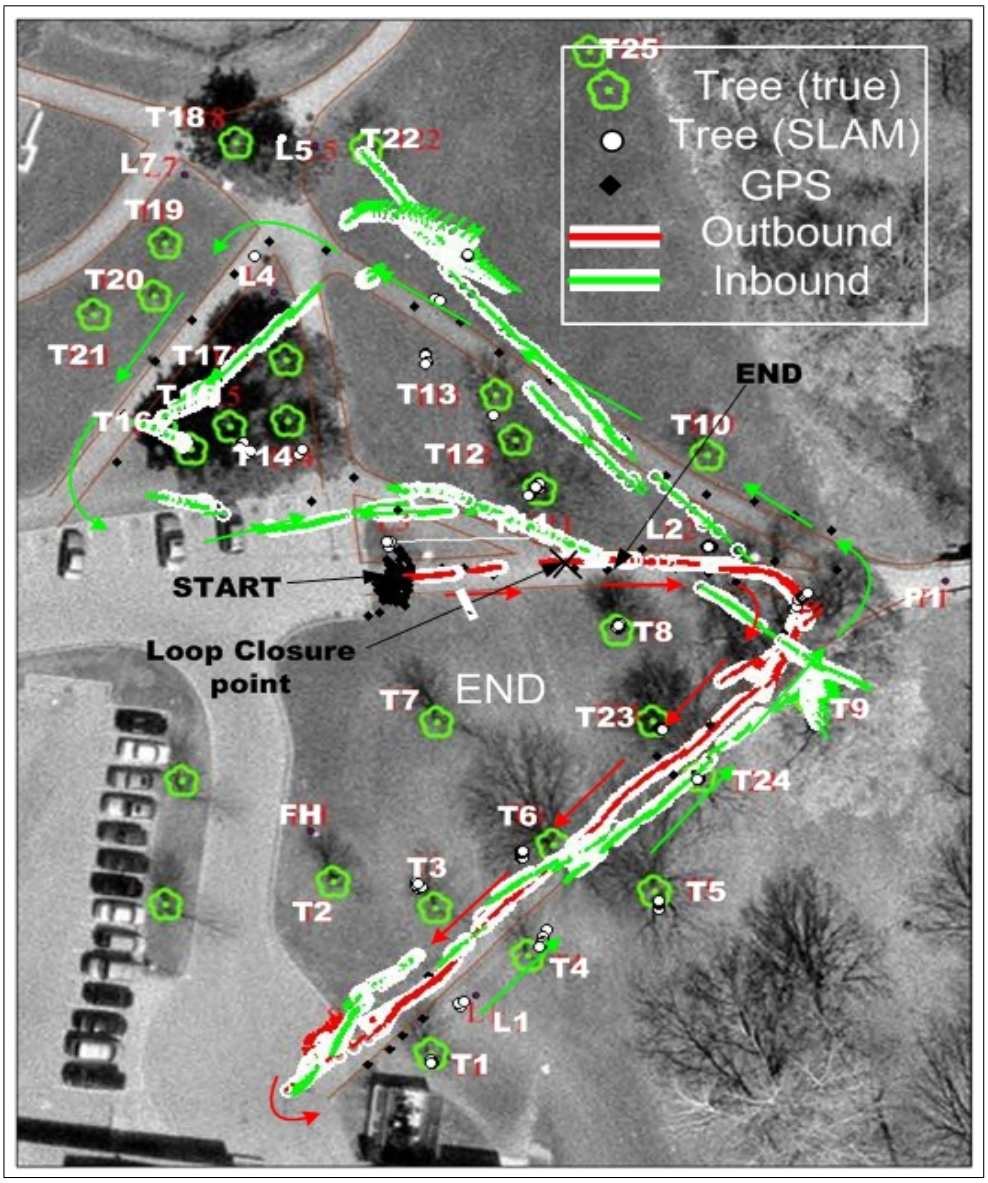


Figure 5.16: VisSLAM results during Experiment 'three' (auto tree detection, recognition, and initialization).

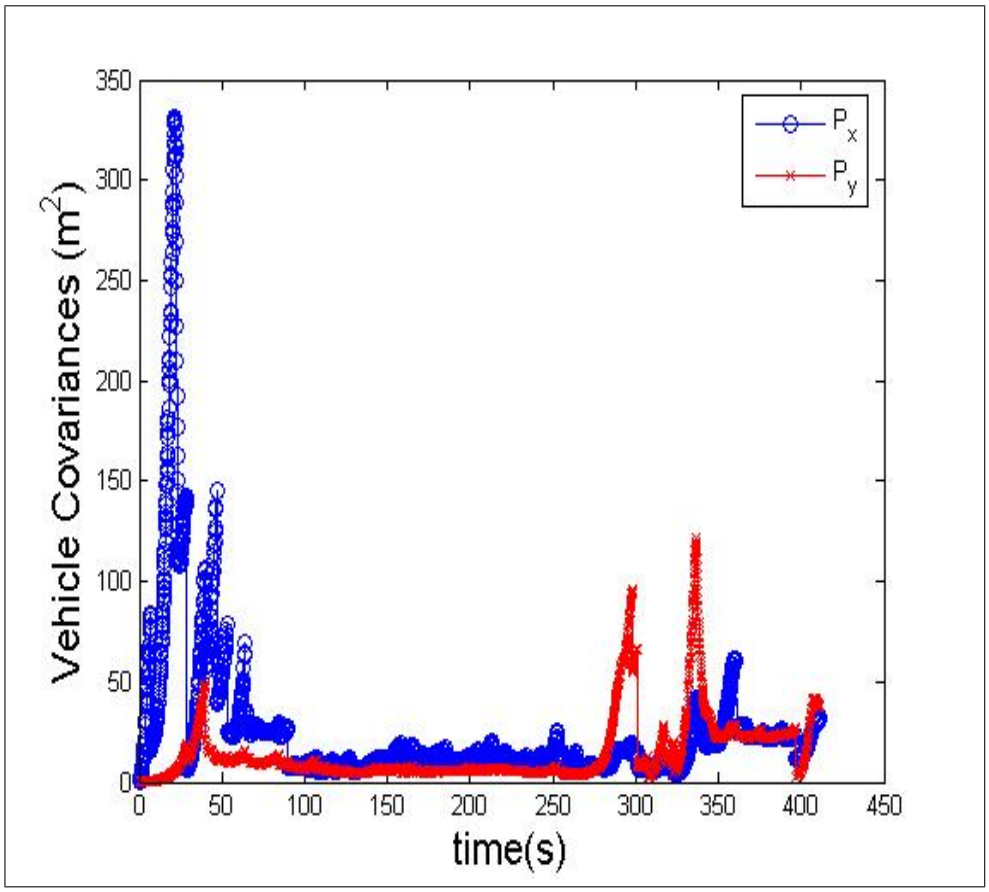


Figure 5.17: Covariance for vehicle position for Experiment 'three' (auto tree detection, recognition, and initialization). P_x and P_y are the covariances in the x and y directions respectively.

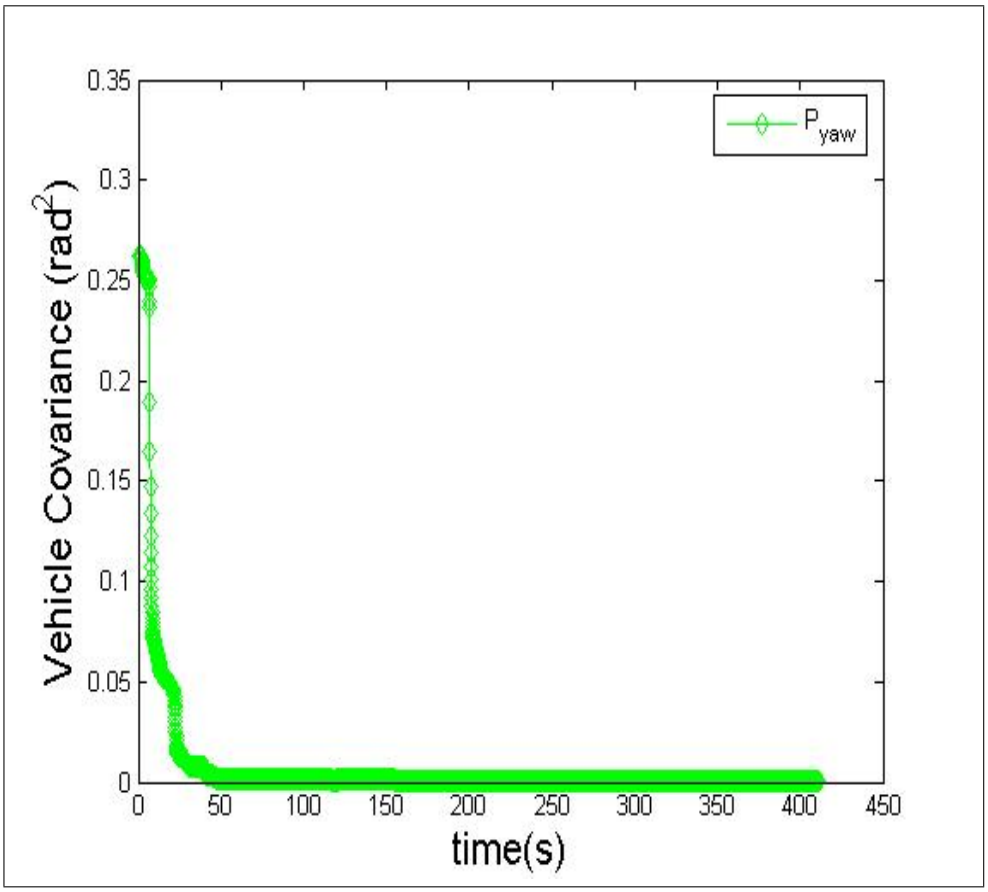


Figure 5.18: Covariance for vehicle bearing for Experiment 'three' (auto tree detection, recognition, and initialization). P_{yaw} is the covariance in the heading of the vehicle.

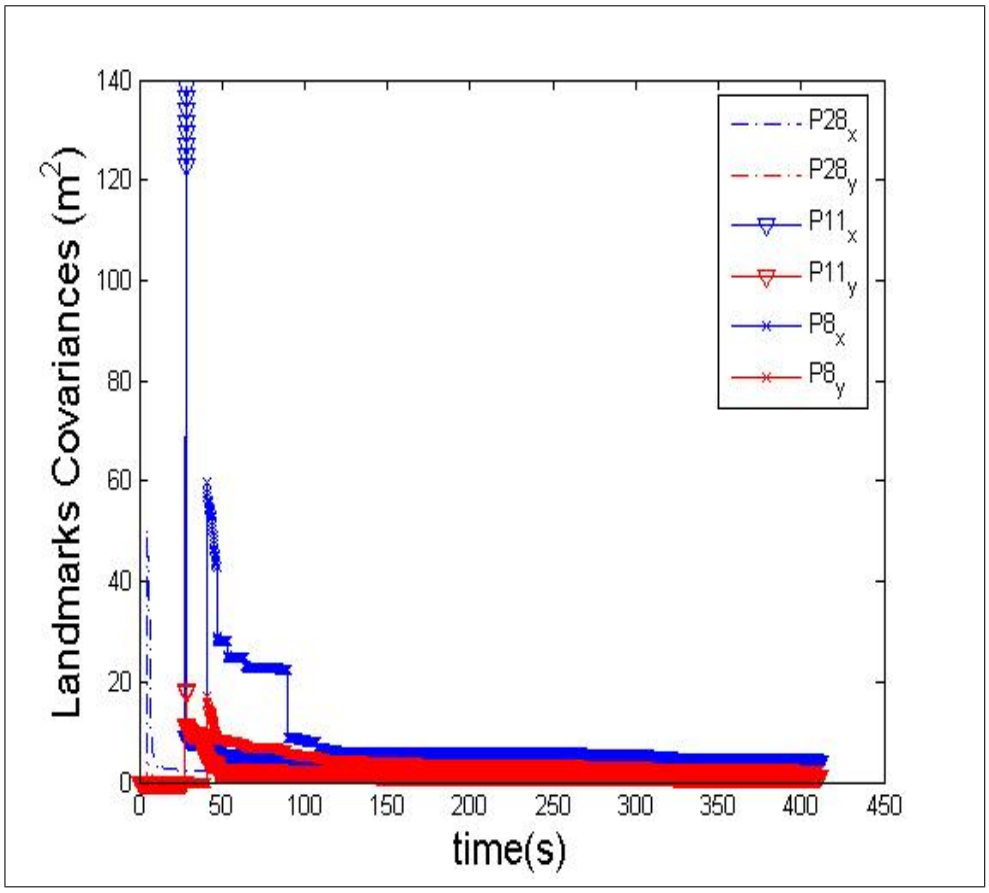


Figure 5.19: Covariance for the position of landmarks 'T28' or Lamp 3, 'T11', and 'T8' for Experiment 'three' (auto tree detection, recognition, and initialization). Pk_x and Pk_y are the covariances for Landmark 'k' in the x and y directions respectively.

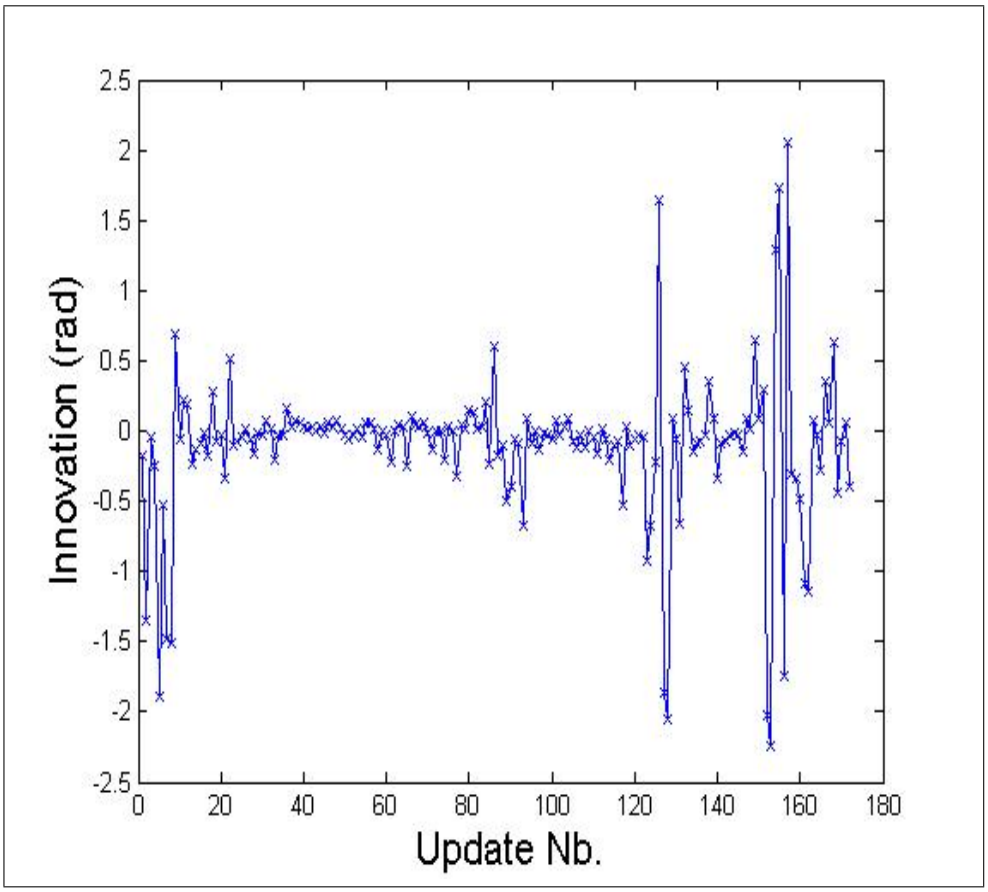


Figure 5.20: Innovation for the bearing of landmarks during Experiment 'three' (auto tree detection, recognition, and initialization). Update Nb. stands for the number of innovations.

5.3.4 Run-time

All of the VisSLAM modules, except for the SIFT feature extraction, are written in Matlab and implemented off-line on a laptop featuring a 1.8 GHz Pentium M processor, and 1GB of RAM. All processing times are shown in Table 5.2. SIFT features are extracted using the code of Lowe [104] which is written in C. The time for extracting SIFTs is added to the tree recognition time. The size of all the images is 640x480 pixels.

Each INS prediction requires 0.25 seconds. This time includes the IMU prediction as well as the application of non-holonomic constraints.

The average time spent for tree detection is approximately 20.22 seconds for a typical image containing two trees. This time includes importing an image, edge detection, line tracking, line pruning, continuity and symmetry calculations and line grouping. Furthermore, this time also includes filtering the image with a Gabor filter (2 scales and 2 orientations).

Each tree recognition step requires approximately 7 seconds. This time includes SIFT extraction, and matching 3 of them to the those saved in the SIFT database.

Tree initialization requires approximately 4.78 s and is in effect the time for the stereo camera API to unwrap a stereo image, rectify the resulting right and left images, extract the disparity map, and calculate depth.

A SLAM update involves executing the EKF equations and are executed in real time.

The total time for a SLAM iteration, including prediction, observation, and update is 32.25 seconds.

Operation	time (s)
INS prediction	.25
Tree detection	20.22
Tree recognition	7.00
Tree initialization	4.78
SLAM update	~ 0
Total SLAM iteration	32.25

Table 5.2: Run-time for VisSLAM processes.

5.4 Discussion

VisSLAM is tested in three experiments, where the mobile platform navigates on a path 314 meters in length, featuring one U-turn, and 6 curbs in which the vehicle makes 2 clockwise rotations and 4 counterclockwise. The results of the tests are shown in Table 5.3 which are interpreted as follows.

In the first test, only the detection part of VisSLAM is automated, while manually recognizing and initializing landmarks. Results are comparable to those achieved in range-bearing systems. The maximum covariance in vehicle position is 20 m^2 although the average value is considerably lower, at approximately 1 m^2. The

covariance in vehicle orientation is approximately 0.0001 rad^2 which is indicative of high confidence in this measure. Landmark position covariance reduces during navigation, where the minimum attained value is 0.05 m^2 for 'T27'. The Innovation in landmark position also remains within low bounds with a mean innovation close to zero (due to manual initialization) and very few spikes at approximately 1.2 rad.

In the second test, automatic tree recognition is implemented in addition to automatic detection. VisSLAM successfully converges and the vehicle estimated trajectories faithfully follow ground truth for the majority of the journey. The maximum covariance in vehicle position during the journey is 100 $m2$ although the average value is much lower at approximately 3 m^2. Although the vehicle covariance in orientation is higher in this test, it is still low at a average of 0.002 rad^2. Landmark position covariance reduces during navigation to a minimum of 0.068 for landmark 'T27'. This reduction is not as much as the previous test because the landmark is not observed as frequently as before. The peak innovation here is approximately 3.4 rad, although the average innovation is much lower during the trip (approximately 0.3 rad).

In the third and final test, all manual intervention is removed and the system is left to automatically detect, recognize, and initialize landmarks. Although the system destabilizes at one point due to insufficient tracked landmarks, VisSLAM allows the vehicle to autonomously navigate for the majority of its journey.

Vision System			Traj_Dev	Cov_V	Cov_Yaw	Cov_L	Inn
Det.	Rec.	Init.	(m)	(m^2)	(rad^2)	(m^2)	(rad)
A	M	M	2.4	20	0.0001	0.05	1
A	A	M	9.5	100	0.002	0.068	3.5
A	A	A	7.9	75	0.0003	0.072	3.2

Table 5.3: Performance of VisSLAM under three tests. First the Detection (Det.), Recognition (Rec.), and Initialization (Init.) of each test are labeled as Automatic (A) or Manual (M). The entries shown are the maximum values for Deviation in trajectory (Traj_Dev), maximum vehicle position covariance (Cov_V), maximum vehicle covariance in heading (Cov_Yaw), minimum covariance in the position of a landmark, and finally the innovation at each update step.

VisSLAM can not be implemented in real-time in its current form. The major bottleneck is the detection system but it is predicted that VisSLAM can be run in

near-real time if ported to a lower-level computer language.

5.5 Summary

This chapter demonstrates the systems developed in Chapters 3 and Chapter 4 in the context of localization and SLAM. A Vision-Inertial SLAM system is developed entitled VisSLAM, structured in the framework of an EKF, using natural features as landmarks.

Three experimental scenarios are presented. In the first two scenarios, the vision-inertial systems are tested on a localization problem, where landmarks are manually localized in the SLAM map. The first scenario differs from the second one in that landmarks are also recognized manually. While, the objective of the first scenario is to test the detection system of Section 4.3, the second scenario aims to test both the detection and recognition systems. In both the first and second scenarios, the CV systems guarantee high enough detection and recognition rates to allow accurate localization. In the third scenario, all three Computer Vision systems, including detection, recognition, and localization are tested on a SLAM problem. Result are encouraging for most of the vehicle trajectory but the filter eventually diverges due to incorrect data associations and low detection and recognition rates.

Chapter 6

Contributions, conclusions, and future research

This primary focus of this thesis is the development of a Computer Vision system for detecting natural landmarks for the sake of both localization and SLAM. A new Computer Vision system is developed which can classify the environment, and detect and recognize natural objects within that environment. The secondary objective of this thesis is to investigate the issues involved in the application of inertial land-based ego-motion estimation using low-cost IMUs. The developed Computer Vision and inertial systems are integrated into a working Vision-Inertial SLAM system, named VisSLAM, that is tested on real data collected during a test run in a park setting. Although VisSLAM is designed for a park setting using tree trunks as landmarks for SLAM, the developed techniques are extendable to other environments using different landmarks.

In the first section of this chapter, the main contributions of VisSLAM are accentuated and relevant conclusions are made. A section then follows which discusses the direction of future research avenues founded in VisSLAM.

6.1 Contributions and conclusions

Environment recognition. An Environment Recognition (ER) system is designed which is built on an Artificial Neural Network. Knowledge of the environment context constitutes top-down information which dictates the type of landmark to use for SLAM. The ER system is tested by querying it with a set of indoor and

outdoor images, where 95% of the indoor images and 79% of the outdoor images are correctly classified.

Object detection and recognition. A Computer Vision system is proposed for segmenting tree trunks within their natural environment. This system operates by segmenting quasi-vertical structures in images and choosing those structures that intersect the Ground-Sky separation line. The second contribution in the field of Computer Vision is the proposed tree recognition system which matches trees based on distinctive features within the borders of their trunks. Images are filtered with a Scale Invariant Feature Transform (SIFT) filter and those SIFT features within the boundaries of each tree trunk are used as identifiers by which the query tree can be matched in feature space. Tests performed on a database of 580 images containing 932 trees result in 71.24% detection rate. Of the detected trees 85.5% of them are recognized by the tree recognition system. Tests performed on VisSLAM in an outdoor park indicate that these detection and recognition rates meet the required standards of a SLAM observation model. The main limitation of the detection system is tracking the the Ground-Sky (G-S) separation line. Although an attempts is made to automate this process, the developed G-S detection system is lacking.

Inertial navigation system. While Inertial Navigation Systems (INS) systems have been designed for decades, very little interest has been given to land-based INS using low cost strapdown IMUs. VisSLAM implements a land-based INS system, which integrates IMU predictions with Non-Holonomic (NH) constraints to improve the INS dead-reckoning capabilities. The contribution of VisSLAM in the field of inertial navigation is twofold: first, to contest a theory based on enforcing NH constraints via forward velocity expressions. Second, to implement a second theory for enforcing NH constraints which is built in the framework of an EKF. The INS system is tested on real data, which indeed reveals constrained motion for the mobile platform.

Vision-inertial SLAM system. Building a Vision-Inertial SLAM system is in itself an accomplishment. VisSLAM succeeds at integrating the foregoing INS and Computer Vision technologies into a working SLAM system capable of operating in an outdoor park, using tree trunks as landmarks.

VisSLAM is tested on real data collected onboard a mobile platform while it is navigating in a park setting. Three scenarios for these tests are proposed. In

the first test, trees are detected automatically, but are recognized and initialized manually. In the second test, automatic detection and recognition is insured but manual initialization is done. In the third and final test, VisSLAM automatically detects, recognizes and initializes landmarks. In the first two tests where landmarks are manually initialized, the detection and recognition systems are good enough to insure loop closure errors that are of the same order of range-bearing systems. While the first test results in an loop closure error of 2.4 meters in position and 2 degrees in orientation, the second test achieves an error of 9 meters in position and 3 degrees in bearing. Furthermore, in both of these tests, the covariance in vehicle position is low (approximately $4m^2$) with the exception of some spikes which are evident in the second test due to erroneous data associations. The vehicle covariance in heading is low for both tests ranging around 0.002 rad^2. For the last test, where VisSLAM is completely autonomous, the performance is considerably degraded. Loop closure error in position is not acceptable (30 meters in position), although the error in heading is still low. The covariance in vehicle position during the journey is high, indicating the need for a better initialization technique.

6.2 Future Research

This thesis prepares the ground work for future research, some directly related to issues raised in this thesis in the fields of Computer Vision, INS and SLAM and others related to the larger scheme of autonomous robot navigation.

In Computer Vision, research is encouraged towards detecting and tracking Ground-Sky separation lines. This is a problem related to perception [107]. How do we judge where the ground end and the sky begins? It is the author's belief that such systems might have to be solved in a holistic approach, where images are parsed into it constituent parts [108]. The implication of this knowledge is the detection of any vertical and erect structure contacting ground.

In SLAM, research is encouraged in landmark initialization. VisSLAM initialization via range readings is lacking. A proposed solution is to use a hybrid between range and bearing systems. Bearing initialization requires a 40 degree angle between two sightings and range initialization requires the landmark to be within the range of the stereo system. These systems appear to complement each other. A system can be designed such that whichever method has its conditions met first is used to initialize a landmark.

Real time issues are not considered for VisSLAM. Any aspiration of porting VisSLAM to a mobile platform for real-time implementation must be preceded by

a conversion of the code to a lower level language than Matlab.

The natural extension of VisSLAM is in executing it in more than one environment using more than one type of landmarks. For instance, when a robot navigates between an park environment of a rural city center VisSLAM would transition from using tree trunks as landmarks to using buildings, or traffic lights. The INS system would effectively remain the same unless one of the environments to which the vehicle travels to is an aerial, underwater, or off-road setting where the non-holonomic constraints no longer hold and a more generic INS system is warranted.

The long term implication of VisSLAM is true autonomy for navigating robots via self-localization inside various environments using natural features. As an example of the application of such technology, an amphibious robot could be made to localize itself on land and underwater, without the need to stop in transition and hard code landmarks according to its surroundings. The developed techniques should be valid for a range of scales and applications including NANO-biomedical, undersea and space robots, in addition to various terrestrial applications.

Bibliography

[1] http://www.irobot.com/consumer/.

[2] http://www.irobot.com/home.cfm.

[3] http://marsrovers.jpl.nasa.gov/home/.

[4] R. E. Kalman. A new approach ot linear filtering and prediction. *ASME Journal of Basic Engineering*, pages 35–45, March 1960.

[5] P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, Inc., 1979.

[6] N. J. Gordon, D. J. Salmon, and A. F. M. Smith. A novel approach for nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings of Conference on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

[7] A. Doucet, N. J. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump markov linear systems. *IEEE Transactions on Signal Processing*, 49(3), March 2001.

[8] D. Fox, W. Burgard, and S. Thrun. Markov localisation for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[9] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE Proceedings on the Conference on Robotics and Automation*, 1999.

[10] H. Durrant-Whyte. An autonomous guided vehicle for cargo handling applications. *International Journal of Robotics Research*, 15(5):407–441, October 1996.

[11] W. Burgard, A. B. Cremers, D. Fox, D. Ahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55, January 1999.

[12] R. Siegwart and I. R. Nourbakhsh. *Introduction to autonomous mobile robots*. MIT press, April 2004.

[13] A. Elfes. Using occupency grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

[14] R. Smith and P. Cheeseman. On the representation and esitmation of spatial unceratainty. *International Journal of Robotics Research*, 5(4):56–68, Winter 1986.

[15] R. Chatila and J. P. Laumond. Position referencing and consistent world modeling for mobile robots. In *IEEE International Conference on Robotics and Automation*, March 1985.

[16] N. Ayache and O. Faugeras. Maintaining representations of the environment of a mobile robot. In *International Symposium on Robotics Research*, August 1987.

[17] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonmomous Robot Vehicles*, pages 167–193, 1990.

[18] M. Csorba. *Simultaneous Localisation and Map Building*. PhD thesis, University of Oxford, 1996.

[19] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 1442–1447, May 1991.

[20] *Autonomous navigation of a Mobile Robot Using Intertial and Visual Cues*, July 1993.

[21] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots (joint issue)*, 31(1-3):29–53, 1998.

[22] G. Dissanayake, S. Williams, H. Durrant-Whyte, and T. Bailey. Map management for efficient simultaneous localization and mapping SLAM. *Autonomous Robots*, 12:267–286, 2002.

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

[24] M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, June 2003.

[25] S. Thrun, D. Koller, Z. Ghahmarani, H. Durrant-Whyte, and A. Ng. Simultaneous mapping and localization with sparse extended information filters: theory and initial results. Technical report, Computer Science Technical Report CMU-CS-02-112, Carnegie Mellon University, Pittsburgh, PA 15213, September 2002.

[26] P. Newman and J. J. Leonard. Consistent, convergent and constant time SLAM. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.

[27] S. Williams, G. Dissanayake, and H. Durrant-Whyte. Efficient simultaneous localization and mapping using local submaps. In *Proceedings of the Australian Conference on Robotics and Automation*, pages 128–134, 2001.

[28] H. J. Chang, C. S. G. Lee, Y.-H. Lu, and Y. C. Hu. A computational efficient slam algorithm based on logarithmic-map partitioning. In *Proceedings of International Conference on Intelligent Robots and Systems*, Sendai, Japan, September 28 - October 2 2004.

[29] J. Guivant, J. Masson, and E. Nebot. Navigation and mapping in large unstructured environments. *International Journal of Robotic Research*, 2003.

[30] J. Nieto, J. Guivant, and E. Nebot. A novel hybrid map representation for DenseSLAM in unstructured large environments. Technical report, Australian Center for Field Robotics, Department of Mechanical and Mechatronic Engineering, The University of Sydney, Sept. 2003.

[31] J.-H. Kim and S. Sukkarieh. Airborne simultaneous localization and map building. In *IEEE Proceedings on the Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[32] P. Newman and J. L. Rikovski. Towards constant-time SLAM on an autonomous underwater vehicle using synthetic aperture sonar. In *International Symposium of Robotics Research*, Siena, Italy, September 2003.

[33] A. Torralba. Contextual priming for object detection. *Int. J. Comput. Vision*.

[34] M. Bryson and S. Sukkarieh. Bearing-only SLAM for an airborne vehicle. Technical report, ARC Centre for Excellence in Autonomous Systems, Australian Centre for Field Robotics, University of Sydney, 2005.

[35] A. J. Davison and N. Kita. 3D simultaneous localisation and map-building using active vision for a robot moving on ondulating terrain. In *IEEE Proceedings of Conference on Computer Vision and Pattern Recognition*, 2001.

[36] A. J. Davison. Real-time simultaneous localization and mapping with a single camera. In *Proceedings of the Conference on Computer Vision*, Nice, France, October 2003.

[37] A. J. Davison and D. W. Murray. Simultaneous localisation and map-building using active vision. *IEEE Proceedings of Conference on Pattern Analysis and Machine Intelligence*, October 2003.

[38] I.-K. Jung. *Simultaneous localization and mapping in 3D environments with stereo vision*. PhD thesis, CNRS, Toulouse, France, 2004.

[39] N. M. Kwok and G. Dissanayake. Bearing-only SLAM in indoor environments using a modified particle filter. In *Proceedings of the Australian Conference on Robotics and Automation*, Brisbane, Australia, December 2003.

[40] N. M. Kwok and G. Dissanayake. An efficient multiple hypothesis filter for bearing-only SLAM. In *IEEE Proceedings of the Conference on Intelligent Robots and Systems*, Sendai, Japan, September 28 - October 2 2004.

[41] N. M. Kwok, G. Dissanayake, and Q. P. Ha. Bearing-only slam using an SPRT based gaussian sum filter. In *IEEE Proceedings of the Conference on Robotics and Automation*, 2005.

[42] T. Lemaire, S. Lacroix, and J. Sola. A practical 3d bearing-only SLAM algorithm. In *IEEE Proceedings of the Conference on Intelligent Robots and Systems*, August 2005.

[43] S. Se, D. Lowe, and J. Little. Simultaneous localisation and map-building using active vision. *The International Journal of Robotics Research*, 21(8):735–758, August 2002.

[44] S. Panzieri, F. Pascucci, and G. Ulivi. Vision based navigation using Kalman approach for SLAM. Technical report, Universitá degli Studi La Sapienza, 2001.

[45] T. Fitzgibbons. *Visual-based simultaneous localisation and mapping*. PhD thesis, University of Sydney, Sydney, Australia, October 2004.

[46] Y. Bar-Shalom and T. E. Fortman. *Tracking and data association*. Academic, Boston, MA, 1988.

[47] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transcations of Robotics and Automation*, 17(6):890–897, 2001.

[48] I.-K. Jung and S. Lacroix. Simultaneous localisation and mapping with stereovision. In *International Symposium on Robotics Research*, Siena, Italy, October 2003.

[49] M. Deans and M. Hebert. Experimental comparison of thechniques for localization and mapping using a bearings only sensor. In *International Symposium on Experimental Robotics*, December 2000.

[50] J. Solà, A. Monin, M. Devy, and T. Lemaire. Undelayed initialization in bearing only SLAM. In *IEEE/RSJ Proceedings of the Conference on Intelligent Robots and Systems*, Edmonton, Canada, 2005.

[51] T. Bailey. Constrained initialisation for bearing-only SLAM. In *IEEE Proceedings of International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[52] J. E. Cutting. How the eye measures reality and virtual reality. *Behavior Research Methods, Instruments, & Computers.*, 29(1):27–36, 1997.

[53] http://www.activrobots.com/ROBOTS/p2at.html.

[54] H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, pages 133–135, September 1981.

[55] A. Azerbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, June 1995.

[56] T. Jebara, A. Azerbayejani, and A. Pentland. 3D structure from 2D motion. *IEEE Transactions on Signal Processing Magazine*, 16(3):66–84, May 1999.

[57] J. Oliensis. A critique of structure-from-motion algorithms. *Computer Vision and Image Understanding*, 80(2):172–214, November 2000.

[58] J. Oliensis. Exact two-image structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1618–1633, 2002.

[59] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 1150–1157, 1999.

[60] J. Shi and C. Tomasi. Affine / photometric invariants for planar intensity patterns. In *Proceedings of the European Conference on Computer Vision*, pages 642–651, 1996.

[61] J. Koenderink and A. Van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:867–375, 1987.

[62] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets. In *European Conference on Computer Vision, Proceedings of the 7th*, pages 414–431, Copenhagen, Denmark, 2002.

[63] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[64] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.

[65] Y. Dufournaud, C. Schmid, and R. Horaud. Matching images with different resolutions. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 612–618, Head Island, SC, USA, June 2000.

[66] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[67] J. Shi and C. Tomasi. Good features to track. In *IEEE Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[68] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 257–263, June 2003.

[69] D. Lerro and J. D. Tardos. Bearing-only tracking for maneuvering targets in clutter. *International Federation of Automatic Control*, pages 267–272, 1996.

[70] N. Peach. Bearing-only tracking using a set of range-parameterized extended kalman filter. In *IEEE Proceedings of the Conference on Control Theory Applications*, volume 142, pages 73–80, January 1995.

[71] T. R. Kronhamn. Bearing-only target motion analysis based on a multihypothesis kalman filter and adaptive ownership motion control. In *IEEE Proceedings of the Conference on Radar, Sonar Navigation*, volume 145, pages 247–252, August 1998.

[72] R. Eustice. *Large-Area Visually Augmented Navigation for Autonomous Underwater Vehicles*. PhD thesis, Massachusetts Institute of Technology / Woods Hole Oceanographic Joint-Program, June 2005.

[73] A. Mallet, S. Lacroix, and L. Gallo. Position estimation in outdoor environments using pixel tracking and stereovision. In *IEEE Proceedings of the International Conference on Robotics and Automation*, pages 3519–3524, San Francisco, Ca, USA, April 2000.

[74] C. Olson, L. Matthies, M. Schoppers, and M. Maimone. Stereo ego-motion improvements for for robust rover navigation. In *IEEE Proceedings of the International Conference on Robotics and Automation*, pages 1099–1104, May 2001.

[75] R. M. Haralick and L. G. Shapiro. *Computer and robot vision, volume II*. Addison Wesley Publishin Company, 1993.

[76] H. W. Sorenson and D. L. Alspach. Recursive bayesian estimation using gaussian sums. *Automatica*, 7:465–479, 1971.

[77] D. L. Alspach and H. W. Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control,*, AC-17(4):439–448, August 1972.

[78] A. Wald. *Sequential analysis*. Wiley, New York, 1947.

[79] A. Wald and J. Wolfowitz. Optimal character of the sequential ratio probability test. *Ann. Math. Statist.*, 19:326–339, 1948.

[80] S. Williams, G. Dissanayake, and H. Durrant-Whyte. Constrained initialization of the simultaneous localisation and mapping algorithm. In *International Conference on Field and Service Robotics, Proceedings of the*, pages 315–330, 2001.

[81] E. M. Foxlin. Generalized architecture for simultaneous localization, auto-calibration, and map-building. In *IEEE/RSJ Proceedings of the Conference on Intelligent Robots and Systems*, 2002.

[82] http://www.systronauto.com/products.html.

[83] http://www.summitinstruments.com.

[84] http://www.xbow.com.

[85] http://www.memsense.com.

[86] http://www.cloudcaptech.com/.

[87] Eduardo Nebot. Navigation system design. Technical report, ARC Centre for Excellence in Autonomous Systems, Australian Centre for Field Robotics, University of Sydney, 2004.

[88] E. Nebot and H. Durrant-Whyte. Initial calibration and alignment of low cost inertial navigation units. *Journal of Robotics Systems*, 16(2):81–92, February 1999.

[89] http://www.cloudcaptech.com/download/IMU/Crista%20Release%201.2.0/Documents/IMUCommunicationsSpecification.1.2.0.UserVersion.pdf.

[90] Gamini Dissanayake, Salah Sukkarieh, Eduardo Nebot, and Hugh Durrant-Whyte. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. In *IEEE Transactions on Robotics and Automation*, volume 17, October 2001.

[91] http://forestry.about.com/cs/treeid/a/tree_id_web.htm.

[92] S. Williams. *Efficient solutions to autonomous mapping and navigation problems*. PhD thesis, The University of Sydney, Sydney, Australia, September 2001.

[93] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In *Proceedings of the International Conference on Computer Vision*, 2003.

[94] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation fo the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[95] K. Tieu and P. Viola. Boosting image retrieval. In *IEEE Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 228–235, 2000.

[96] J. S. De Bonet and P. A. Viola. Structure driven image database retrieval. In *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[97] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, IEEE Proceedings of the Conference on*, volume 1, pages 511–518, Kauai, Hawaii, USA, 2001.

[98] E. P. Simoncelli and W. T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *IEEE Proceedings on the 2nd International Conference on Image Processing*, volume III, pages 444–447, Washington, DC, 1995.

[99] Fabio Ramos, Suresh Kumar, Ben Upcroft, and Hugh Durrant-Whyte. Representing natural objects in unstructured environments. Technical report, ARC Center of Ecvellence for Autonomous Systems, The University of Sydney, Australia, 2005.

[100] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision*, 56(3):151–177, 2004.

[101] G. Dorkó and C. Schmid. Object class recognition using descriminative local features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):1–13, 2004.

[102] T. Kadir and M. Brady. Scale, saliency, and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[103] G. Dorkó and C. Schmid. Selection of scale-invariant parts for object class recognition. In *Proceedings of the International Conference on Computer Vision*, 2003.

[104] D. G. Lowe. Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[105] R. Nock and F. Nielson. Statistical region merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11), November 2004.

[106] D. Foresyth and J. Ponce. *Computer Vision: A modern approach*. Prentice-Hall, New York, 2000.

[107] S. Coren, L. M. Ward, and J. T. Enns. *Sensation and Perception*. Harcourt-Brace, Orlando, 1996.

[108] Z.-W. Tu, X. R. Chen, A. L. Yuille, and S.-C. Zhu. Image parsing: unifying segmentation, detection and recognition. *International Journal of Computer Vision*, 63(2):113–140, September 2005.

[109] P.J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[110] R. Mohan and R. Nevatia. Perceptual organization for scene segmentation and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6), June 1992.

[111] S. Todorovic, M. C. Nechyba, and P. G. Ifju. Sky/ground modeling for autonomous flight. Technical report, Department of Electrical and Computer Engineering, University of Florida, 2002.

[112] http://www.ptgrey.com/.

[113] P. Newman and J. L. Rikovski. IEEE proceedings of the conference on robotics and automation. pages 2555–2560, Nice, France, 1992.

[114] http://www.geoplane.com/trimble/geoexplorer.html.

[115] http://www.dperception.com/.

[116] http://www.canadiantire.ca/assortments/.

[117] http://www.fes.uwaterloo.ca/geography/index.html.

[118] http://www.acfr.usyd.edu.au/homepages/academic/tbailey/index.html.

Appendix A

Experimental setup: equipment, setting, frames of reference, and data sets

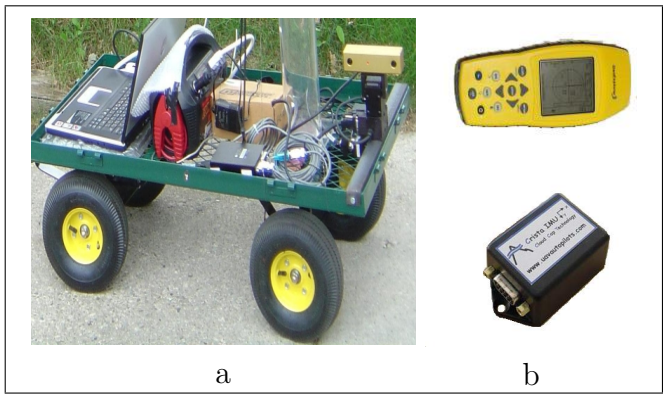
This Chapter deals with issues related to VisSLAM experiments including the equipment used in Section A.1, the experimental setting in Section A.2, and the collected data sets in Section A.3.

A.1 Equipment

The experimental equipment consists of a hand held Global Positioning System(GPS), an Inertial Measurement Unit (IMU) and a stereo camera. These three units are mounted on a mobile platform (Figure A.1), which is manually guided around a track .

The GPS unit is a carrier phase hand held unit made by Trimble, model named "GeoExplorer3" [114]. Accurate and precise absolute position is assured by tracking between 4 to 12 satellites at one time. Carrier phase processing is possible for sub-meter precision. The GeoExplorer3 is a stand alone item, where data is collected on site and later downloaded to the computer. The collected data is differentially corrected during post-processing.

The IMU is a three-axis strap-down inertial sensor made by Cloud Cap Technologies, model named "Crista" [86]. It comprises three orthogonal Micro Electro Mechanical System (MEMS) accelerometers and three gyroscopic rate sensors. The



a b

Figure A.1: Experimental equipment. (a) Cart used to collect data, equipped with PTU, stereo camera, hand held GPS (not shown) and strap-down IMU. (b) hand held GPS and strap-down IMU.

gyros and accelerometers feature $\pm 300^{\circ}/sec$ and $\pm 10g$ angular velocity and linear acceleration ranges respectively. The gyros operate below 0.05% $Full\ Scale(FS)$ bias error and below 1% scale error. The accelerometers operate below $0.025\% FS$ bias error and below 1% scale error. This unit is calibrated over $-40^{\circ}C$ to $80^{\circ}C$ temperature range. The output rate and over-sample averaging rates of output data can be controlled by the user. Over-samples are the number of A/D measurements made and averaged for each signal for each IMU update. The suggested IMU rates are $10Hz$ sampling rate and $200Hz$ over-sampling rate. The IMU specifications are shown in Table A.3:

Gyros	
Range	$\pm 300^{\circ}/sec$
Scale Factor Error	$< 1\%$
In-Run Bias Error	
Fixed Temperature	$< 0.05\%$FS
Over Temperature	$< 0.2\%$FS
Turn-on to Turn-on bias	$< 0.25\%$FS
Linear acceleration Effects	$0.2^{\circ}/sec/g$ typical
Resolution	$0.009^{\circ}/sec$
Bandwidth	2^nd order filter. Fc = 100Hz

Table A.1: Specifications of the Crista gyroscopes.

Accelerometers	
Range	$\pm 10g$
Scale Factor Error	$< 1\%$
In-Run Bias Error	
Fixed Temperature	$< 0.025\%$FS
Over Temperature	$< 0.2\%$FS
Turn-on to Turn-on bias	$< 0.3\%$FS
Alignment Error	$< 0.0025g$
Resolution	$0.3mg$
Bandwidth	Simple RC LPF. Fc = 100Hz

Table A.2: Specifications of the Crista accelerometers.

The camera is a stereo camera made by Pt. Grey, model named "Bumblebee" [**?**]. This unit features two $1/3''$ progressive scan CCD's. Accurate pre-calibration is done for lens distortions and camera misalignments. The frame rate is controllable, peaking at a maximum frame rate of $30fpm$. Connection to the camera is insured via $IEEE1394$ firewire. The focal length of the cameras is $4mm$, which corresponds to 70° Field of View (FOV). Images are aligned within 0.05 pixel RMS error. The unit is equipped with a calibration retention system that prevents it from losing calibration when the device is subject to mechanical shock and vibration.

Cameras	two Sony ICX084 Color, 1/3 inch progressive scan CCDs
Resolution	640x480
Frame Rate	$30Hz$
A/D Sampling	10 bit
Shutter Speed	$1/8000sto1/30s$
Baseline	12 cm
Focal Length	4 mm
Field of View	70°
Size	160x40x50 mm
Weight	375 g
Power Consumption	2.1 W

Table A.3: Specifications of the Bumblebee camera.

The camera is mounted on a Pan Tilt Unit (PTU). The PTU is the "PTU-D46-17" model manufactured by Directed Perception [115]. It features a 78° tilt angle range and a 318° pan angle range. Its maximum speed is $300°/s$ with on-the-fly speed and position changes.

A laptop is mounted on the vehicle to control the PTU unit and collect IMU data and images of the surroundings during navigation. The laptop features a Pentium M, 1.5 GHz processor with 1MB of onboard RAM. The IMU, stereo camera, and PTU unit are powered by a $12V$ battery pack [116].

A.2 Environmental setting and ground truth

An outdoor area is sought that is relatively flat and smooth (no inclinations higher than 10°) with few bumps, populated with large trees exhibiting visible trunks that could be used as landmarks. Keeping in mind these constraints, the open area outside St Jerome's building on the University of Waterloo campus is chosen (Figure A.2). This area spans a width of 70 meters by a length of 93 meters. A total of 25 conifer trees occupy the area, through which run several intertwining paved paths. The trees exhibit tree trunks that are large and salient. Determining the effectiveness of any SLAM system requires first establishing ground truth for the coordinates of the tracked landmarks and the pose of the vehicle as it navigates. These issues are tackled in the following section.

Ground truth for the experimental setting is acquired from a geographically referenced aerial image of the site [117]. This consists of an image similar to a photographic image, where each pixel is reference to a latitude and longitude. Next, a Computer Aided Drawing (CAD) of the site is obtained and overlaid on top of the geographically referenced image. The CAD image comprises the features of the site, such as paths, trees, and buildings. Vector data images (*i.e.*, CAD) are often out of date and lacking. For the sake of this work, the 2D location of trees trunks and their diameter is of interest for developing ground truth. This information is not available and had to be collected. The standard method to locate features in Geographic Information Systems (GIS) is to locate oneself next to the object with a hand held Global Positioning System (GPS) and fixate the object via the GPS. This procedure is not possible to locate trees since GPS suffers from outages and *multipath* errors under dense foliage such as conifers. Due to this limitation, it is necessary to fixate alternative features (lampposts) that are distant from foliage using GPS and then locate the position of trees by triangulation. The images of each of the trees that is used as a landmark are shown in Figures A.3 and A.4.

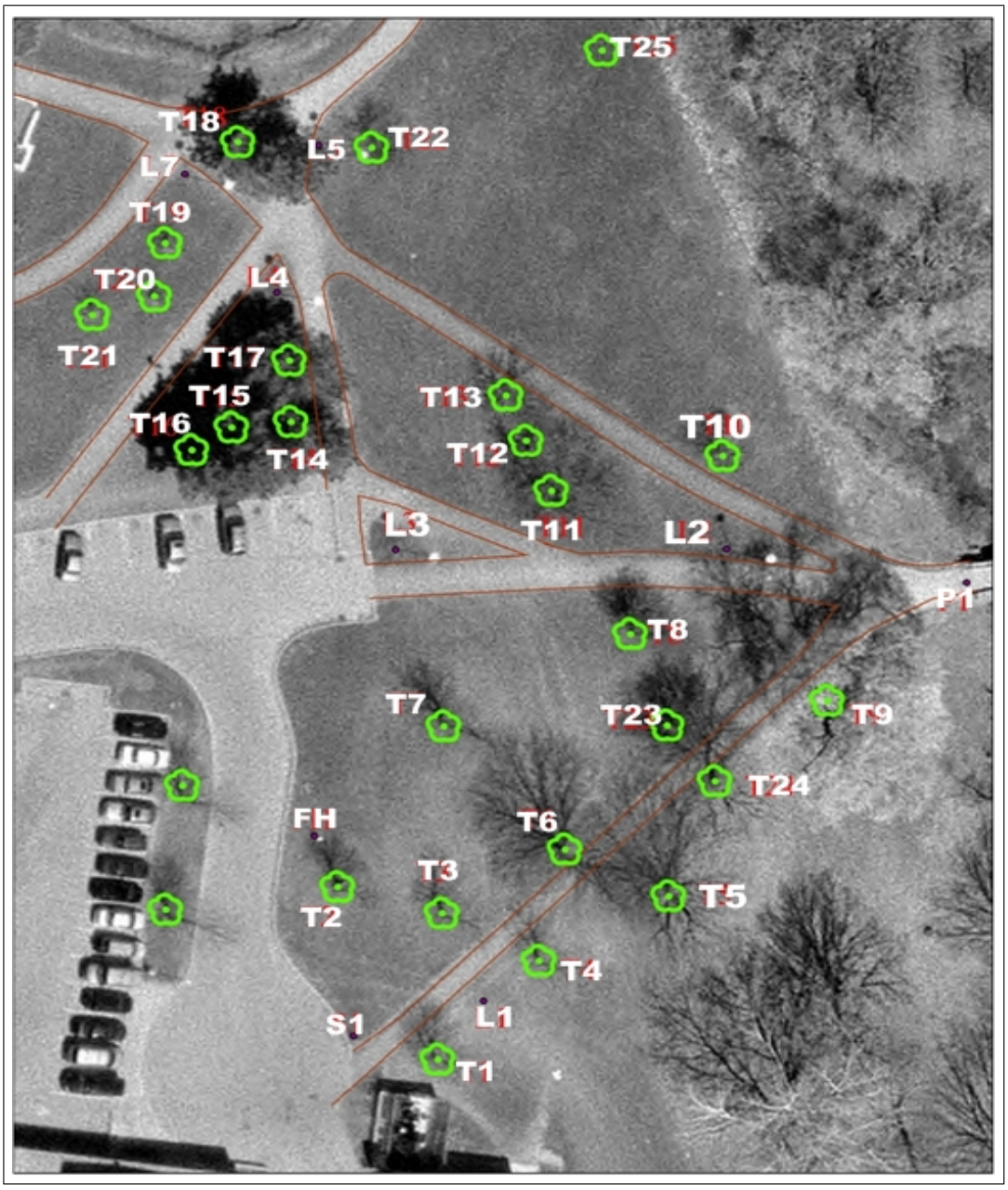


Figure A.2: Aerial image of the test site where IMU and GPS data, as well as stereo images were collected for Vision SLAM navigation runs. The location is the open area near St Jerome's building at the University of Waterloo campus. The overlaid circles are the landmarks (tree trunks) that will be used for Vision SLAM.

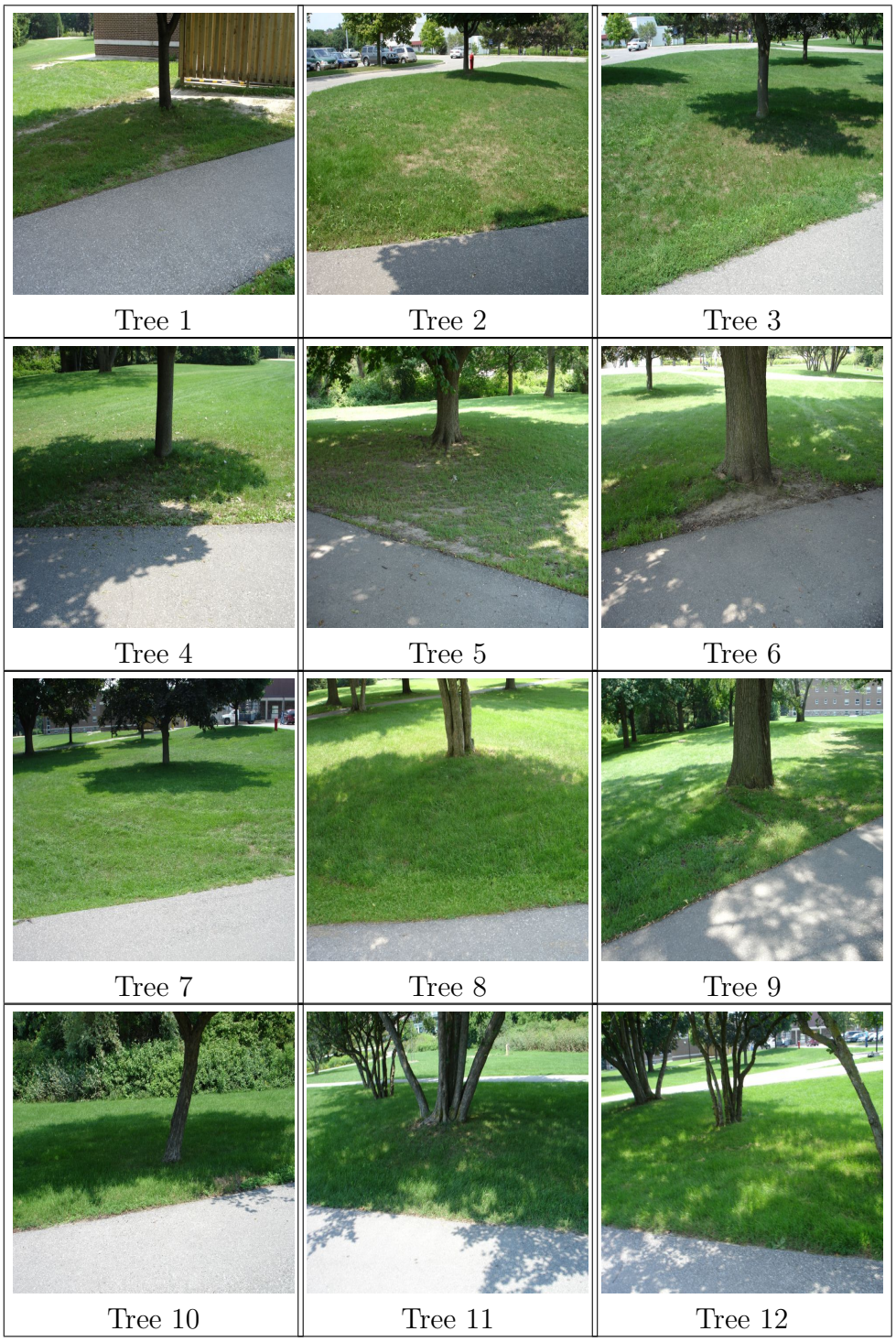


Figure A.3: Trees in the test site

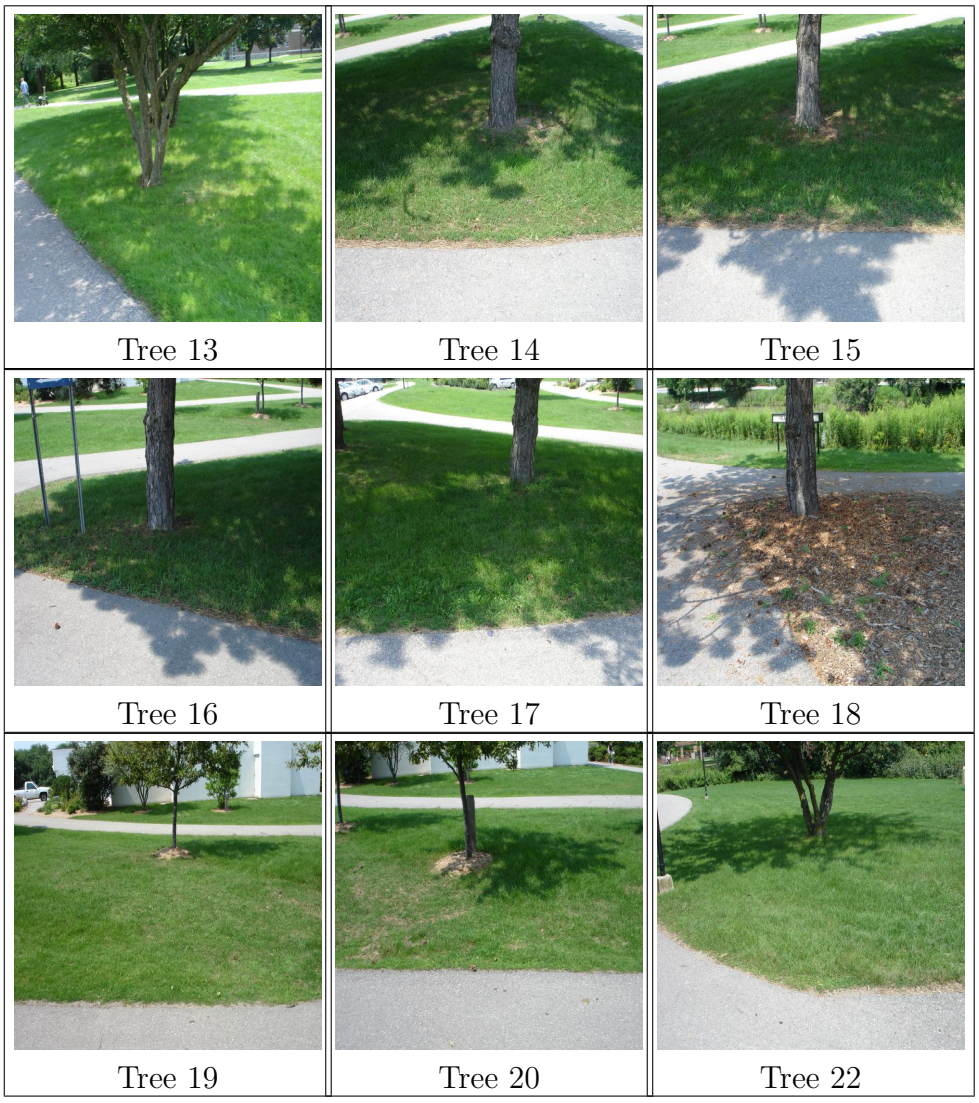


Figure A.4: Trees in the test site

Ground truth for the pose of the vehicle during navigation is established via a differential GPS. The vehicle follows a path that is distant from trees for the majority of the time, where GPS reception is adequate for positioning.

A.3 Data sets

The information collected from the IMU, camera, and gyroscope consist of three data files and a sequence of images. The first data file, entitled 'IMU_*.txt', includes information pertinent to the IMU sensor and its format is presented in Table A.4.

N	Clks	S	N	h	m	s	Gx	Gy	Gz	Ax	Ay	Az	N	N	N
-	Clks	#	-	h	m	s	$^\circ/s$	$^\circ/s$	$^\circ/s$	m/s^2	m/s^2	m/s^2	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table A.4: IMU data file format. 'N' signifies that this column is not used. 'Clks' stands for the number of clock cycles that have passed since the IMU was initiated. The clock is a 10 MHz and recycles each 429 seconds. 'S' is an 8 bit sequence. 'h', 'm', and 's' are the PC clock in hours, minutes and seconds respectively. 'Gx','Gy', and 'Gz' are the gyroscope readings in $^\circ/s$ and 'Ax', 'Ay', and 'Az' are the accelerometer readings in m/s^2.

The time stamp at which an IMU reading is taken can be calculated in one of two methods. In the first method, the time stamp is calculated as the simple addition of the hour, minute, and second entries of the PC clock. This time stamp represents the time at which the IMU reading is processed by the PC and includes some delay in it. The second, more precise method is to use the PC clock only to obtain an initial time stamp t_0 and then calculate subsequent time stamps by adding the number of PC clocks to t_0

$$t_k = t_0 + Clks * \frac{1}{10^7}, \tag{A.1}$$

where t_k is the time stamp at IMU reading k, Clks is the number of clock cycles since t_0 and the number 10^7 refers to the clock frequency. Although this second time stamp is more representative of the time an IMU reading is taken, better synchronization with the camera images are obtained using the first method.

The second data file 'IMG_*.txt', shown in Table **??** relates to the images captured during navigation and each entry is time stamped based on the PC clock. The time stamps in the IMU and IMG data files are synchronized since they are both based on the PC clock.

Iter	h	m	s	N	N	N
#	h	m	s	-	-	-
-	-	-	-	-	-	-

The final data file is the one related to GPS readings. The first two columns include longitude and latitude readings. The remaining column represent the GPS time stamp, which is acquired from the atomic clock onboard the GPS satellite and is extremely precise.

Long	Lat	hh	mm	ss
m	m	hh	mm	ss
-	-	-	-	-

Table A.5: GPS data file format. 'N' signifies that this column is not used. 'Long' stands for longitude, 'Lat' stands for latitude. 'h', 'm', and 's' are the GPS clock in hours, minutes and seconds respectively.

The GPS time stamps are synchronized to those of the IMU and IMG files by observing the time differential between the PC clock and that of the GPS unit.

In addition to the three text files, stereo images are saved in a folder to be later uncoupled into their corresponding right and left images.

Appendix B

Details of jacobian calculation

The Jacobians that are used in Chapters 3 and 5 can be calculated in one of two methods. In the first method, analytical expressions are developed for the Jacobians in an explicit fashion and those equations are used to calculate the Jacobians at each time step. It turns out that these equations are quite involved as can is shown in the next section. Alternatively, the Jacobians can be calculated numerically as shown in Section B.2, which is relatively simple to implement on a computer.

B.1 Closed-form analytical method

B.1.1 Process model

$$\begin{bmatrix} p_x^n(k) \\ p_y^n(k) \\ p_z^n(k) \\ v_x^n(k) \\ v_y^n(k) \\ v_z^n(k) \\ \phi(k) \\ \theta(k) \\ \psi(k) \end{bmatrix} = \begin{bmatrix} p_x^n(k-1) + v_x^n(k)\Delta t \\ p_y^n(k-1) + v_y^n(k)\Delta t \\ p_z^n(k-1) + v_z^n(k)\Delta t \\ v_x^n(k-1) + (C_b^n(1,:)[a_x^n; a_y^n; a_z^n] + g_x)\Delta t \\ v_y^n(k-1) + (C_b^n(2,:)[a_x^n; a_y^n; a_z^n] + g_y)\Delta t \\ v_z^n(k-1) + (C_b^n(3,:)[a_x^n; a_y^n; a_z^n] + g_z)\Delta t \\ \phi(k-1) + E_b^n(1,:)[w_x^n; w_y^n; w_z^n]\Delta t \\ \theta(k-1) + E_b^n(2,:)[w_x^n; w_y^n; w_z^n]\Delta t \\ \psi(k-1) + E_b^n(3,:)[w_x^n; w_y^n; w_z^n]\Delta t \end{bmatrix}, \tag{B.1}$$

$$\begin{bmatrix} p_x(k) \\ p_y(k) \\ p_z(k) \\ v_x(k) \\ v_y(k) \\ v_z(k) \\ \phi(k) \\ \theta(k) \\ \psi(k) \end{bmatrix} = \begin{bmatrix} p_x(k-1) + v_x(k)\Delta t \\ p_y(k-1) + v_y(k)\Delta t \\ p_z(k-1) + v_z(k)\Delta t \\ v_x(k-1) + [(c\psi c\theta)a_x + (c\psi s\theta s\phi - s\psi c\phi)a_y + (c\psi s\theta c\phi + s\psi s\phi)a_z + g_x)]\Delta t \\ v_y(k-1) + [(s\psi c\theta)a_x + (s\psi s\theta s\phi + c\psi c\phi)a_y + (s\psi s\theta c\phi - c\psi s\phi)a_z + g_y)]\Delta t \\ v_z(k-1) + [(-s\theta)a_x + (c\theta s\phi)a_y + (c\theta c\phi)a_z + g_z)]\Delta t \\ \phi(k-1) + [w_x + (s\phi t\theta)w_y + (c\phi t\theta)w_z]\Delta t \\ \theta(k-1) + [(c\phi)w_y - (s\phi)w_z]\Delta t \\ \psi(k-1) + [(s\phi sec\theta)w_y + (c\phi sec\theta)w_z]\Delta t \end{bmatrix}, \tag{B.2}$$

$$\mathbf{J}_x = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{J}_{4,7} & \mathbf{J}_{4,8} & \mathbf{J}_{4,9} \\ 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{J}_{5,7} & \mathbf{J}_{5,8} & \mathbf{J}_{5,9} \\ 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{J}_{6,7} & \mathbf{J}_{6,8} & \mathbf{J}_{6,9} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{J}_{7,7} & \mathbf{J}_{7,8} & \mathbf{J}_{7,9} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{J}_{8,7} & \mathbf{J}_{8,8} & \mathbf{J}_{8,9} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{J}_{9,7} & \mathbf{J}_{9,8} & \mathbf{J}_{9,9} \end{bmatrix}, \tag{B.3}$$

where

$$
\begin{aligned}
\mathbf{J}_{4,7} &= [(c\psi s\theta c\phi + s\psi s\phi)a_y + (-c\psi s\theta s\phi + s\psi c\phi)a_z]\Delta t \\
\mathbf{J}_{4,8} &= [(-c\psi s\theta)a_x + (c\psi c\theta s\phi)a_y + (c\psi c\theta c\phi)a_z]\Delta t \\
\mathbf{J}_{4,9} &= [(-s\psi c\theta)a_x + (-s\psi s\theta s\phi - c\psi c\phi)a_y + (-s\psi s\theta c\phi + c\psi s\phi)a_z]\Delta t \\
\mathbf{J}_{5,7} &= [(s\psi s\theta c\phi - c\psi s\phi)a_y + (-s\psi s\theta s\phi - c\psi c\phi)a_z]\Delta t \\
\mathbf{J}_{5,8} &= [((-s\psi s\theta)a_x + (s\psi c\theta s\phi)a_y + (s\psi c\theta c\phi)a_z]\Delta t \\
\mathbf{J}_{5,9} &= [(c\psi c\theta)a_x + (c\psi s\theta s\phi - s\psi c\phi)a_y + (c\psi s\theta c\phi + s\psi s\phi)a_z]\Delta t \\
\mathbf{J}_{6,7} &= [(c\theta c\phi)a_y + (-c\theta s\phi)a_z]\Delta t \\
\mathbf{J}_{6,8} &= [(-c\theta)a_x + (-s\theta s\phi)a_y + (-s\theta c\phi)a_z]\Delta t \\
\mathbf{J}_{6,9} &= 0 \\
\mathbf{J}_{7,7} &= 1 + [(c\phi t\theta)w_y + (-s\phi t\theta)w_z]\Delta t \\
\mathbf{J}_{7,8} &= [(s\phi sec^2\theta)w_y + (c\phi sec^2\theta)w_z]\Delta t \\
\mathbf{J}_{7,9} &= 0 \\
\mathbf{J}_{8,7} &= [(-s\phi)w_y + (-c\phi)w_z]\Delta t \\
\mathbf{J}_{8,8} &= 1 \\
\mathbf{J}_{8,9} &= 0 \\
\mathbf{J}_{9,7} &= [(c\phi sec\theta)w_y + (-s\phi sec\theta)w_z]\Delta t \\
\mathbf{J}_{9,8} &= [(c\phi sec\theta t\theta)w_y + (-s\phi sec\theta t\theta)w_z]\Delta t \\
\mathbf{J}_{9,9} &= 1
\end{aligned}
$$

$$
\mathbf{J}_u = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi & 0 & 0 & 0 \\
s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi & 0 & 0 & 0 \\
-s\theta & c\theta s\phi & c\theta c\phi & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & s\phi t\theta & c\phi t\theta \\
0 & 0 & 0 & 0 & c\phi & -s\phi \\
0 & 0 & 0 & 0 & s\phi sec\theta & c\phi sec\theta
\end{bmatrix}, \qquad \text{(B.4)}
$$

B.1.2 Observation model

$$\mathbf{H}_i(\hat{x}(k|k-1) = \begin{bmatrix}\varphi_i \\ \vartheta_i\end{bmatrix} = \begin{bmatrix}\arctan\left(\frac{Y^c}{X^c}\right) \\ \arctan\left(\frac{Z^c}{\sqrt{(X^c)^2+(Y^c)^2}}\right)\end{bmatrix}, \tag{B.5}$$

$$p^c_{mc} = \begin{bmatrix}X^c \\ Y^c \\ Z^c\end{bmatrix} = C^b_n[m^n_i - p^n - C^n_b p^b_{cb}], \tag{B.6}$$

$$C^n_b = \begin{bmatrix}c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi\end{bmatrix}, \tag{B.7}$$

$$C^b_n = inv(C^n_b) = (C^n_b)^T = \begin{bmatrix}c\psi c\theta & s\psi c\theta & -s\theta \\ c\psi s\theta s\phi - s\psi c\phi & s\psi s\theta s\phi + c\psi c\phi & c\theta s\phi \\ c\psi s\theta c\phi + s\psi s\phi & s\psi s\theta c\phi - c\psi s\phi & c\theta c\phi\end{bmatrix}, \tag{B.8}$$

$$p^c_{mc} = \begin{bmatrix}X \\ Y \\ Z\end{bmatrix} = \begin{bmatrix}P11(m_x - P_x) + P12(m_y - P_y) + P13(m_z - P_z) \\ P21(m_x - P_x) + P22(m_y - P_y) + P23(m_z - P_z) \\ P31(m_x - P_x) + P32(m_y - P_y) + P33(m_z - P_z)\end{bmatrix} \tag{B.9}$$

where where $d = X^2 + Y^2$, $\delta x = m_x - P_x$, $\delta y = m_y - P_y$, and $\delta z = m_z - P_z$. In the same manner, the terms of the second row of the Jacobian are expressed as

$$\begin{aligned}
P11 &= c\psi c\theta \\
P12 &= s\psi c\theta \\
P13 &= -s\theta \\
P21 &= c\psi s\theta s\phi - s\psi c\phi \\
P22 &= s\psi s\theta s\phi + c\psi c\phi \\
P23 &= c\theta s\phi \\
P31 &= c\psi s\theta c\phi + s\psi s\phi \\
P32 &= s\psi s\theta c\phi - c\psi s\phi \\
P33 &= c\theta c\phi
\end{aligned}$$

$$J(\mathbf{H}_i) = \begin{bmatrix}\frac{\partial \mathbf{H}_1}{\partial p_x} & \frac{\partial \mathbf{H}_1}{\partial p_y} & \frac{\partial \mathbf{H}_1}{\partial p_z} & \frac{\partial \mathbf{H}_1}{\partial v_x} & \frac{\partial \mathbf{H}_1}{\partial v_y} & \frac{\partial \mathbf{H}_1}{\partial v_z} & \frac{\partial \mathbf{H}_1}{\partial \phi} & \frac{\partial \mathbf{H}_1}{\partial \theta} & \frac{\partial \mathbf{H}_1}{\partial \psi} \\ \frac{\partial \mathbf{H}_2}{\partial p_x} & \frac{\partial \mathbf{H}_2}{\partial p_y} & \frac{\partial \mathbf{H}_2}{\partial p_z} & \frac{\partial \mathbf{H}_2}{\partial v_x} & \frac{\partial \mathbf{H}_2}{\partial v_y} & \frac{\partial \mathbf{H}_2}{\partial v_z} & \frac{\partial \mathbf{H}_2}{\partial \phi} & \frac{\partial \mathbf{H}_2}{\partial \theta} & \frac{\partial \mathbf{H}_2}{\partial \psi}\end{bmatrix} \tag{B.10}$$

$\mathbf{H}_1$ and $\mathbf{H}_2$ are taken from (5.11) and are equal to $\arctan\left(\frac{Y^c}{X^c}\right)$ and $\arctan\left(\frac{Z^c}{\sqrt{(X^c)^2+(Y^c)^2}}\right)$ respectively. The partial derivatives in (B.10) are evaluated as

$$
\begin{aligned}
\frac{\partial \mathbf{H}_1}{\partial p_x} &= [\frac{\partial Y}{\partial p_x}X - \frac{\partial X}{\partial p_x}Y]\frac{1}{X^2}\frac{1}{(1+\frac{Y^2}{X^2})} \\
\frac{\partial \mathbf{H}_1}{\partial p_x} &= [\frac{\partial Y}{\partial p_x}X - \frac{\partial X}{\partial p_x}Y]\frac{1}{d} \\
\frac{\partial \mathbf{H}_1}{\partial p_y} &= [\frac{\partial Y}{\partial p_y}X - \frac{\partial X}{\partial p_y}Y]\frac{1}{d} \\
\frac{\partial \mathbf{H}_1}{\partial p_z} &= [\frac{\partial Y}{\partial p_z}X - \frac{\partial X}{\partial p_z}Y]\frac{1}{d} \\
\frac{\partial \mathbf{H}_1}{\partial v_x} &= \frac{\partial \mathbf{H}_1}{\partial v_y} = \frac{\partial \mathbf{H}_1}{\partial v_z} = 0 \\
\frac{\partial \mathbf{H}_1}{\partial \phi} &= [\frac{\partial Y}{\partial \phi}X - \frac{\partial X}{\partial \phi}Y]\frac{1}{d} \\
\frac{\partial \mathbf{H}_1}{\partial \theta} &= [\frac{\partial Y}{\partial \theta}X - \frac{\partial X}{\partial \theta}Y]\frac{1}{d} \\
\frac{\partial \mathbf{H}_1}{\partial \psi} &= [\frac{\partial Y}{\partial \psi}X - \frac{\partial X}{\partial \psi}Y]\frac{1}{d}
\end{aligned}
$$

$$\frac{\partial \mathbf{H}_2}{\partial p_x} = [\frac{\partial Z}{\partial p_x} d^{1/2} - \frac{1}{2} d^{-1/2} (2X \frac{\partial X}{\partial p_x} + 2Y \frac{\partial Y}{\partial p_x}) Z] \frac{1}{e}$$
$$\frac{\partial \mathbf{H}_2}{\partial p_y} = [\frac{\partial Z}{\partial p_y} d^{1/2} - \frac{1}{2} d^{-1/2} (2X \frac{\partial X}{\partial p_y} + 2Y \frac{\partial Y}{\partial p_y}) Z] \frac{1}{e}$$
$$\frac{\partial \mathbf{H}_2}{\partial p_z} = [\frac{\partial Z}{\partial p_z} d^{1/2} - \frac{1}{2} d^{-1/2} (2X \frac{\partial X}{\partial p_z} + 2Y \frac{\partial Y}{\partial p_z}) Z] \frac{1}{e}$$
$$\frac{\partial \mathbf{H}_2}{\partial v_x} = \frac{\partial \mathbf{H}_2}{\partial v_y} = \frac{\partial \mathbf{H}_2}{\partial v_z} = 0$$
$$\frac{\partial \mathbf{H}_2}{\partial \phi} = [\frac{\partial Z}{\partial \phi} d^{1/2} - \frac{1}{2} d^{-1/2} (2X \frac{\partial X}{\partial \phi} + 2Y \frac{\partial Y}{\partial \phi}) Z] \frac{1}{e}$$
$$\frac{\partial \mathbf{H}_2}{\partial \theta} = [\frac{\partial Z}{\partial \theta} d^{1/2} - \frac{1}{2} d^{-1/2} (2X \frac{\partial X}{\partial \theta} + 2Y \frac{\partial Y}{\partial \theta}) Z] \frac{1}{e}$$
$$\frac{\partial \mathbf{H}_2}{\partial \psi} = [\frac{\partial Z}{\partial \psi} d^{1/2} - \frac{1}{2} d^{-1/2} (2X \frac{\partial X}{\partial \psi} + 2Y \frac{\partial Y}{\partial \psi}) Z] \frac{1}{e}$$

$$\frac{\partial X}{\partial p_x} = -P11; \ \frac{\partial X}{\partial p_y} = -P12; \ \frac{\partial X}{\partial p_z} = -P13$$
$$\frac{\partial X}{\partial \phi} = \frac{\partial P11}{\partial \phi} \delta x + \frac{\partial P12}{\partial \phi} \delta y + \frac{\partial P13}{\partial \phi} \delta z$$
$$\frac{\partial X}{\partial \theta} = \frac{\partial P11}{\partial \theta} \delta x + \frac{\partial P12}{\partial \theta} \delta y + \frac{\partial P13}{\partial \theta} \delta z$$
$$\frac{\partial X}{\partial \psi} = \frac{\partial P11}{\partial \psi} \delta x + \frac{\partial P12}{\partial \psi} \delta y + \frac{\partial P13}{\partial \psi} \delta z$$

Similarly

$$\frac{\partial Y}{\partial p_x} = -P21; \ \frac{\partial Y}{\partial p_y} = -P22; \ \frac{\partial Y}{\partial p_z} = -P23$$
$$\frac{\partial Y}{\partial \phi} = \frac{\partial P21}{\partial \phi} \delta x + \frac{\partial P22}{\partial \phi} \delta y + \frac{\partial P23}{\partial \phi} \delta z$$
$$\frac{\partial Y}{\partial \theta} = \frac{\partial P21}{\partial \theta} \delta x + \frac{\partial P22}{\partial \theta} \delta y + \frac{\partial P23}{\partial \theta} \delta z$$
$$\frac{\partial Y}{\partial \psi} = \frac{\partial P21}{\partial \psi} \delta x + \frac{\partial P22}{\partial \psi} \delta y + \frac{\partial P23}{\partial \psi} \delta z$$

and

$$\frac{\partial Z}{\partial p_x} = -P31;\ \frac{\partial Z}{\partial p_y} = -P32;\ \frac{\partial Z}{\partial p_z} = -P33$$
$$\frac{\partial Z}{\partial \phi} = \frac{\partial P31}{\partial \phi}\delta x + \frac{\partial P32}{\partial \phi}\delta y + \frac{\partial P33}{\partial \phi}\delta z$$
$$\frac{\partial Z}{\partial \theta} = \frac{\partial P31}{\partial \theta}\delta x + \frac{\partial P32}{\partial \theta}\delta y + \frac{\partial P33}{\partial \theta}\delta z$$
$$\frac{\partial Z}{\partial \psi} = \frac{\partial P31}{\partial \psi}\delta x + \frac{\partial P32}{\partial \psi}\delta y + \frac{\partial P33}{\partial \psi}\delta z$$

$$\frac{\partial P11}{\partial \phi} = 0$$
$$\frac{\partial P11}{\partial \theta} = -c\psi s\theta$$
$$\frac{\partial P11}{\partial \psi} = -s\psi c\theta$$

$$\frac{\partial P12}{\partial \phi} = 0$$
$$\frac{\partial P12}{\partial \theta} = -s\psi s\theta$$
$$\frac{\partial P12}{\partial \psi} = c\psi c\theta$$

$$\frac{\partial P13}{\partial \phi} = 0$$
$$\frac{\partial P13}{\partial \theta} = -c\theta$$
$$\frac{\partial P13}{\partial \psi} = 0$$

$$\frac{\partial P21}{\partial \phi} = +c\psi s\theta c\phi + s\psi s\phi$$
$$\frac{\partial P21}{\partial \theta} = +c\psi c\theta s\phi$$
$$\frac{\partial P21}{\partial \psi} = -s\psi s\theta s\phi - s\psi c\phi$$

$$\frac{\partial P22}{\partial \phi} = +s\psi c\theta c\phi - c\psi s\phi$$
$$\frac{\partial P22}{\partial \theta} = +s\psi c\theta s\phi$$
$$\frac{\partial P22}{\partial \psi} = +c\psi s\theta s\phi - s\psi c\phi$$

$$\frac{\partial P23}{\partial \phi} = +c\theta c\phi$$
$$\frac{\partial P23}{\partial \theta} = -s\theta s\phi$$
$$\frac{\partial P23}{\partial \psi} = 0$$

$$\frac{\partial P31}{\partial \phi} = -c\psi s\theta s\phi + s\psi c\phi$$
$$\frac{\partial P31}{\partial \theta} = c\psi c\theta c\phi$$
$$\frac{\partial P31}{\partial \psi} = -s\psi s\theta c\phi + c\psi s\phi$$

$$\frac{\partial P32}{\partial \phi} = -s\psi s\theta s\phi + c\psi c\phi$$
$$\frac{\partial P32}{\partial \theta} = s\psi c\theta c\phi$$
$$\frac{\partial P32}{\partial \psi} = c\psi s\theta c\phi + s\psi s\phi$$

$$\frac{\partial P33}{\partial \phi} = -c\theta s\phi$$
$$\frac{\partial P33}{\partial \theta} = -s\theta c\phi$$
$$\frac{\partial P33}{\partial \psi} = 0$$

B.2 Numerical method

In this second method, taken from Bailey [118], the Jacobian is calculated numerically as follows. Let $S1(x, y, z)$, $S2(x, y, z)$, ... $Sn(x, y, z)$ represent n vectors of the state transition matrix for whom the Jacobians with respect to the variables x,y, and z are sought. Each entry of the Jacobian J is found as

$$J = \begin{bmatrix} J_{S1x} & J_{S1y} & J_{S1z} \\ J_{S2x} & J_{S2y} & J_{S2z} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ J_{Snx} & J_{Sny} & J_{Snz} \end{bmatrix} \tag{B.11}$$

where

$$\begin{aligned} J_{S1x} &= \frac{S1(x+\delta x, y, z) - S1(x, y, z)}{\delta x} \\ J_{S1y} &= \frac{S1(x, y+\delta y, z) - S1(x, y, z)}{\delta y} \\ J_{S1z} &= \frac{S1(x, y, z+\delta z) - S1(x, y, z)}{\delta z}, \end{aligned}$$

and δx, δy,δz are very small increments added to the states for the sake of finding the Jacobians. In this work, a value of 10^{-9} is used as an increment.