USING SHAPE INFORMATION FROM NATURAL TREE LANDMARKS FOR IMPROVING SLAM PERFORMANCE

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Bilal Turan

March, 2012

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

Assist. Prof. Dr. Selim Aksoy(Advisor)

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

Assist. Prof. Dr. Ali Aydın Selçuk

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

Assist. Prof. Dr. Uluç Saranlı

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

USING SHAPE INFORMATION FROM NATURAL TREE LANDMARKS FOR IMPROVING SLAM PERFORMANCE

Bilal Turan
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Selim Aksoy
March, 2012

Localization and mapping are crucial components for robotic autonomy. However, such robots must often function in remote, outdoor areas with no a-priori knowledge of the environment. Consequently, it becomes necessary for field robots to be able to construct their own maps based on exteroceptive sensor readings. To this end, visual sensing and mapping through naturally occurring landmarks have distinct advantages. With the availability of high bandwidth data provided by visual sensors, meaningful and uniquely identifiable objects can be detected. This improves the construction of maps consisting of natural landmarks that are meaningful for human readers as well.

In this thesis, we focus on the use of trees in an outdoor environment as a suitable set of landmarks for Simultaneous Localization and Mapping (SLAM). Trees have a relatively simple, near vertical structure which makes them easily and consistently detectable. Furthermore, the thickness of a tree can be accurately determined from different viewpoints. Our primary contribution is the usage of the width of a tree trunk as an additional sensory reading, allowing us to include the radius of tree trunks on the map. To this end, we introduce a new sensor model that relates the width of a tree landmark on the image plane to the radius of its trunk. We provide a mathematical formulation of this model, derive associated Jacobians and incorporate our sensor model into a working EKF SLAM implementation. Through simulations we show that the use of this new sensory reading improves the accuracy of both the map and the trajectory estimates without additional sensor hardware other than a monocular camera.

Keywords: Simultaneous Localization and Mapping, Computer Vision, Visual Tracking.

ÖZET

EKH VERİMİNİ ARTIRMAK İÇİN AĞAÇLARIN BİÇİM BİLGİSİNİN KULLANILMASI

Bilal Turan
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Y. Doç. Dr. Selim Aksoy
Mart, 2012

Robotların özerk hareketi için konumlanma ve haritalama çok önemli birimlerdir. Lakin, bu tip robotlar genelde yerleşimden uzak, dış alanlarda çevre hakkında bir önsel bilgi olmadan çalışmak zorundadır. Buna bağlı olarak, saha robotları algılayıcıları doğrultusunda kendi haritalarını çıkarmak zorundadırlar. Bu doğrultuda, görsel algılayıcılar ile haritalamanın çeşitli avantajları vardır. Yüksek bant genişliğine sahip görsel algılayıcılar ile, insanların da anlayabileceği haritalar anlamlı ve birbirinden ayırt edilebilir nesneler tanımlanarak çıkartılabilir.

Bu tezde, Eşzamanlı Konumlanma ve Haritalama (EKH) için uygun işaretler grubu olarak, dış alanlarda ağaçların işaret olarak kullanılması üzerine yoğunlaştık. Ağaçlar göreceli olarak kolay, neredeyse dikey bir yapıya sahiptir ve bu, onların kolay ve istikrarlı olarak tesbit edilebilmelerini sağlar. Daha önemlisi, ağacın kalınlığı farklı görüş açılarından hassas ve istikrarlı olarak belirlenebilmektedir. Bizim ana katkımız, ağaç kalınlığı bilgisini algılayıcılar ile algılayarak, haritadaki her ağaca yarıçap bilgisini de dahil edebilmektir. Bu doğrultuda, görüntüdeki ağaç kalınlığı ile onun yarıçapını birbirine bağlayan yeni bir algılayıcı modeli geliştirdik. Bu modelin matematiksel formüllerini çıkarıp, EKH için gerekli olan türevlerini aldik. Sonra da, çalışan bir EKH üzerine algılayıcı modelini ve türevleri ekledik. Yeni algılayıcı modelinin yalnızca bir kamera kullanarak, konumlanma ve haritalama hassasiyetini artırdığını simülasyonlar ile gösterdik.

Anahtar sözcükler: Eşzamanlı Konumlanma ve Haritalama, Görüntü İşleme, Görsel Takip.

Acknowledgement

First of all I owe great thanks to my supervisor Uluç Saranlı for his patience, encouragement and support through out my studies. It was an honor for me to work under such a knowledgeable, diligent and insightful mentor. I can truly say that, I learned a lot through out my academic journey.

I also greatly appreciate the guidance of Selim Aksoy and Ali Aydın Selçuk and I thank them for being part of my thesis committee.

I would like to thank Utku Çulha for always being there for me and being a friend whom I could always count on, Tolga Özarslan for being a beacon of guidance through out my research and a dear brother. I would also like to thank Ali Nail İnal, İsmail Uyanık, Tuğba Yıldız, Özlem Gür, Deniz Güven and all members of the Bilkent Dexterous Robotics and Locomotion Group and all SensoRHex members for their moral and academic support.

I am also appreciative of the financial support from TÜBİTAK, the Scientific and Technical Research Council of Turkey.

Finally I owe my loving thanks to my parents Şerife and Mehmet, and my sisters Havva and Hatice, and my brothers Ekrem and İbrahim for their patience and encouragement. I thank Ryoko Yamamoto for always being by my side, for supporting and comforting me every step of the way. And I thank Alexandra Zehra Aksu for making me smile when I most needed it and also for helping me with my English. And I thank Fatih Bilge Atar, Yavuz Mester, Uğur Yılmaz, Enver Kayaaslan, M. Emin Öztürk, Joy Anna Crow, Minkee Kim, Oğuz Şahin, Abdullah Gülle and all my dear friends for being morally supportive and making this process bearable!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Autonomous navigation is very important for mobile robotics community. For autonomy, sensing the environment and estimating relative position within the environment is very important. Most of the time map of the environment is not available, thus robot is required to build an estimated map of the environment, then localize itself in this relative map. To generate a map, we need to know relative positions of features to be mapped. A camera as a sensor gives large amounts of information to detect and identify features in any environment, which makes it a perfect sensor for this purpose. However, it is not possible to derive distance information directly using a single pixel feature in an image. We know that features from an image lie on some ray starting from the camera and extending to infinity. Using stereo vision however, distance information can be found using disparity between images from the two cameras by finding the intersection of the two rays from these cameras. However, this method doubles the processing time of images and the distance information for distant landmarks may be erroneous. Another method is using only one camera while using measurements from different locations to estimate distance information of a feature. If two images from different locations have enough disparity, then distance to a feature can be estimated.

This delays initialization of a feature in the estimated map, and complicates the process. This method is more difficult to use in a SLAM environment.

Stereo vision seems to be the only way to immediately derive depth information and is used by many animals and humans to estimate how far away objects are. Using two eyes and utilizing the parallax from their separation, we can compute the distance of objects which are not too distant. When we do not use both eyes, our depth perception dramatically decreases but we can still recover some depth information by looking at objects we are already familiar with. Size gives important information to guess the distance of objects. If we can extract information concerning the shape of an object, we could obtain additional information that we can utilize in a SLAM environment.

Extracting depth information from an object using only one camera hence requires the knowledge of its structure. If the structure is known beforehand, then we could measure some features of the object using its image to estimate depth information. To be able to do this, we must be able to estimate the shape of the object of interest after detecting it in the image.

Knowing the structure of an object and estimating its shape in real time might be difficult depending on the type of the object. Once we detect the object, we have to determine its orientation, since some features to be measured might depend on orientation. Particularly while extracting depth information, small errors in features will result in large differences. These two reasons make the problem even harder, and the processing time they add to our implementation might be too long for us to consider within a SLAM framework. A solution to this might be using objects which have known simple structures that can be easily defined and detected. We choose to use tree trunks for this purpose, since they have a well defined shape whose properties can be estimated consistently from different viewpoints. Moreover, we can simply measure the width of the tree trunk in an image, which indirectly gives us depth information if we know the actual thickness of the tree.

1.2 Contributions

In this thesis, we introduced a new sensor model for use in a SLAM framework, which relies on the observed widths of tree trunks in the image plane. We use tree trunks as landmarks and show that we can use the position and the radius of a tree landmark to find the corresponding width of a tree trunk in the image using a projective camera model. Width information enables us to use monocular vision as both a range and a bearing sensor. Compared to other sensor models which do not use this width information, we see significant improvements.

We have derived and implemented a limited but adequate SLAM environment to test our new sensor model. We have derived mathematical models for the width sensor, its inverse and the Jacobians for both models. Our sensor model gives the position and the width information of a tree trunk visible in the image plane using structural parameters of a tree located in the 3D world. The inverse sensor model is used to find the radius of a tree whose position is known in the 3D world, using the tree width from image. We also created a simulation environment to emulate visual measurements for use in our SLAM environment. Using data from this simulation environment, we have been able to test the performance of our new method.

1.3 The Organization of the Thesis

In Chapter 2, we review existing research and basic background to help the reader familiarize with SLAM problems and our work. Then in Chapter 3, we present the details of our sensor model. In Chapter 4, we give details about how we test our new model and show results for its use within a SLAM environment. Finally, in Chapter 5, we conclude our work and discuss possible future directions.

Chapter 2

Background and Related Work

2.1 Autonomous Robot Navigation

Autonomous operation has been one of the central goals for the mobile robotics community. This goal naturally encompasses numerous components, including challenges in achieving adequate mobility [31, 7] and sufficiently accurate sensing of both the robot and environment state [36] to support decision making for goal-oriented behavioral control. In this thesis, we focus on the latter, investigating how visual sensing could be used to track the movement of a mobile robot.

Traversability is one of the most fundamental components of autonomous navigation. A mobile robot has to know which part of the map it can pass through, and which parts it should avoid. For large scale navigation in the outdoors, feasible paths can be identified using laser scanner data, a method often used by autonomous vehicles [37]. In indoor goal-based navigation, a mobile robot can plan its trajectory towards a goal by segmenting the image of the environment and identifying components such as the floor, the wall, doors and so on [29]. In forest like outdoor environments where bushes and thick and thin trees are present as obstacles. If the width of the trees can be estimated, it can be used to decide which trees can be run over to identify a feasible path [18]. Most of the time, traversability information is not used alone, but used as a part of a bigger

framework.

Another basic component in this context is the knowledge of displacement. Estimation of the displacement of a robot in a given time interval is called odometry. Integration of these differences can be used to estimate the trajectory of the robot. However, integrating over different odometry measurements results in an increasing position error over time, also known as drift. Odometry is generally used with localization in a known map which in return prevents drifting [21].

Knowing a robot's absolute position in a given map enables trajectory planning. A mobile robot, can locate itself in a known map using its sensors if one is given. Localization can also be used with odometry whenever possible, but if not then tracking the pose of the robot over time and deciding most likely location is still possible [33].

Most of the time, the map of the environment cannot be given to the robot. However, a robot can sense its environment and start building a local map if it has knowledge of its own position, which is called mapping. However, if the position of the robot is also not known, it can try to localize itself with respect to a local map it is building. A robot's act of building a local map of the environment and localizing itself in the same map is called Simultaneous Localization and Mapping (SLAM), which received substantial attention in the mobile robotics community. Many approaches are available, but we focus on a probabilistic approach where the location of the robot is defined by a Gaussian probability density function [36].

The Extended Kalman Filter (EKF) SLAM is one of the most frequently used methods for SLAM problem [9, 4, 38]. EKF SLAM defines the robot and landmark positions as multi variate Gaussian distributions, and correlations between landmarks and robot trajectories are defined as a covariance matrix. Moreover, different landmarks are correlated with each other, which results in a big covariance matrix in computations, resulting in $O[n^2)]$ complexity, where n is the total number of parameters used for landmarks. This complexity limits the number of landmarks in the environment. To solve this problem, different solutions are available. Fast SLAM is one of these, where the robot location is tracked through a

particle filter where each particle has its own landmark map with a local EKF instance inside [23]. Fast SLAM extends the limitation in the number of landmarks compared with EKF SLAM, but the correlation information between landmarks are now lost. Two layered approaches are also used to have EKF SLAM running on lower layers and a higher, topological layer defining different maps connected together. This addresses limitations of the EKF SLAM approach without losing correlation information between landmarks [32].

2.2 Sensors for Mobile Robot Navigation

Autonomous robots uses sensor to sense the environment and decide on the traversibility of a path, locate themselves in known maps and build new maps accordingly. Many different sensors can be used in a mobile robot, and every sensor type has a different sensor model which informs the robot how to use them. In this thesis, we focus on sensors which can be divided into two categories: range sensors and visual sensor - only bearing sensors.

Raw sensor data may not be useful as it is in some cases. Particularly for sensors with high bandwidth data, such as laser range scanners and cameras, it is impossible to use raw data as is. One of the passible ways in which semantically relevant information can be extracted from such raw data is through the definition of landmarks. A landmark could be based on computational criteria as image saliencies [22], meaningful structures as corners or curvatures [27] or based on objects with known geometries [42, 17].

2.2.1 Range Sensors

Range sensors are sensors which can measure distance to objects in the environment n a given direction. Using a range sensor, the position of a landmark can be known relative to the robot. Range sensors are active sensors, in that they emit sound, light or other signals. Calculating the time between emission and

reception of a signal gives the distance of the object. The active nature of range sensors makes them energy inefficient.

Laser range scanners are one of the most commonly used sensors in this category. They give quite accurate position information and have high data bandwidth, which in turn is very valuable for detecting objects by covering large fields of view [37]. When acquired data is treated as a point cloud, scan-matching methods can be used to match a part of the laser scan data into an estimated map within SLAM [26]. If laser range data is processed into landmarks, it can give very useful data to use in localization and mapping [27, 21].

Sonars are another type of range sensor that rely on sound emission. Unfortunately, they do not give very accurate position information. However, recognizing natural landmarks like trees through sequential echolocation and acoustic image analyzing is possible [40], which makes sonar sensor is a possible option for SLAM.

2.2.2 Visual Sensors

Cameras are bearing only sensors, in that each pixel on the image plane defines a semi-infinite ray starting from the focal point of the camera. Distance information cannot be directly computed from raw data provided by the camera. However, if the same pixel could be seen from two different locations, we could compute the distance to that pixel by intersecting two different rays from the two different images. This angle difference between the two rays is called parallax. With increasing parallax, accuracy of distance computations increase. Wrongly using raw data from the camera is computationally infeasible, so landmark based methods are preferred. Image data can be segmented in order to find traversability maps [29]. Image saliencies [12, 11] or detected objects [2, 1] can also be used as landmarks.

Stereo vision is widely used both in computer vision and robotics. The baseline separation of two cameras enables direct distance measurements for pixels.

However, due to errors in pixel locations, the range in which this distance remains accurate is quite limited. Yet, it is very useful to have direct access to such distance measurements. Measured landmarks could be directly incorporated into map [4, 3].

In contrast, monocular vision has the disadvantage of delay in computing distance of the features since a mobile robot needs to move until sufficient parallax is observed. However for a variety of practical reasons such as power and computational efficiency, it might be preferable to only use a single camera [12, 9, 13].

2.3 Object Detection

Image saliency is a good starting point for identifying features to be used in SLAM, but even with the best descriptors there is still a significant data association problem. In an image, there may be many salient features with the same descriptor. A better approach for defining landmarks is through objects, which are associated with semantic meanings. This way, the detection and identification of object type landmarks becomes more feasible. Many salient features together could define an object [5, 1]. Another approach is to segment an image and use texture or color information to identify individual regions as objects. If necessary, segments can be refined afterwards [24, 35]. Using a line detector, and connecting lines accordingly an object can also be created from a set of lines. For example a tree can be defined as collection of two vertical nearby lines [6, 43].

2.3.1 Artificial Landmarks

Artificial landmarks are generally simple engineered objects placed manually in an environment. They should be designed to make object detection very easy. Moreover, since their size and shape is known, the information they represent is more than natural landmarks. Artificial landmarks can be used to accurately localize a robot [41]. However, artificial landmarks are not always available and

placing them may be costly. They are primarily useful when the robot system has limited computational resources.

2.3.2 Natural Landmarks

Natural landmarks are preferred over artificial landmarks, because they are already in the environment and it is not always possible to place artificial landmarks in remote outdoor environments. Image saliencies are provide simplest natural landmarks. Nevertheless, any object that can be found naturally in an indoor or outdoor environment can be defined as natural landmark.

In this thesis, we focus on trees as natural landmarks. Trees are easily detectable and their position on image plane is well defined [43, 18, 6], which makes them a good landmark choice for SLAM [4]. Moreover, bark surfaces associated with each tree are distinct enough to allow identification of each tree to address the data association problem for SLAM implementations. However, having a limited number of trees in the environment makes the SLAM problem harder, which is the main reason why the method in [4] failed. The first obvious extension to this idea is to consider the radius of the tree trunk as an additional feature to be measured. Measuring tree radius is not a new idea [18, 2], but it is not used in SLAM framework yet. In this thesis, we focus on utilizing tree radius information, within EKF SLAM framework.

2.4 Simultaneous Localization and Mapping with an Extended Kalman Filter

EKF SLAM uses multivariate Gaussian distributions to parametrize the state within the whole SLAM framework and associated uncertainties. EKF yields an optimal solution for SLAM, when the sensors and the motion model of a system can be expressed through linear equations, provideing the best way to fuse different Gaussian distributions. EKF maintains mean vectors for the robot

pose and landmark parameters. Correlations between the elements of this mean vector are stored in a covariance matrix. EKF SLAM has two main components: A sensor model and a motion model. These are used to update the multivariate Gaussian function at each step during the operation of the filter.

2.4.1 The Sensor Model

The sensor model provides an interface between sensor measurements and the EKF SLAM algorith. Intrinsic parameters of this sensor model should be decided, based on landmarks in a map and measurements from sensors. Using the mean vector of EKF SLAM, the sensor model predicts measurements. The difference between predicted and actual measurements is called innovation and is used to update EKF state predictions. To be able to use the innovation for this update, we need to have Jacobians associated with the sensor model.

Adding new landmarks into the EKF requires the inverse of the sensor model. When adding new landmarks, both the mean vector and the covariance matrix of the EKF must be extended. To do this properly, the Jacobian of the inverse of the sensor model is also required.

2.4.2 Choosing Landmarks

The choice of landmarks is a very important component in SLAM. The sensor model uses the parameters of the landmark for predicting measurements. The parametrization of a landmark and the type of a landmark are different in this context. Here, we focus on the minimum representation of a point type, which are the most common type of landmarks. A point landmark can be specified as a point in 3D. Most image saliency landmarks [22, 14] and some object-based landmarks [4] are point type landmarks. Another type of landmark is the line landmark [39]. The difference between line landmarks and point landmarks is that, line landmarks represent a line in 3D, their projection to the image plane would also yield to a line under most circumstances. Other, more specialized type

of landmarks could also be introduced according to corresponding sensor models. For instance, using laser range scanners, edges, corners and curve segments could be used as special types of landmarks [28]. Two intersecting lines could define a landmark, which in return would yield better SLAM performance than separate point and line type landmarks [25]. In this thesis, we also introduce a special type of landmark for tree trunks. Every tree landmark will have an associated radius parameter making them circle landmarks.

Landmark parametrization, on the other hand, is an orthogonal issue. For point type landmarks, we can have XYZ, inverse depth, homogeneous point and anchored homogeneous point representations all of which have different consistency and complexity values for SLAM [34]. The XYZ parametrization is the most commonly used parametrization, particularly with a range sensor model [23]. However, since the location of landmarks cannot be initialized when they are first seen, in monocular vision SLAM, a delayed initialization approach is required [12]. In such cases, the Inverse Depth Parametrization (IDP) can be used, enabling EKF to initialize a landmark at first sight [9]. The Inverse depth parametrization initializes a ray, which starts at the position of the robot where the landmark is first seen, and the direction of the ray is towards the landmark. The point landmark lies on this ray, with its position defined by an inverse depth parameter. The inverse depth parametrization has a 4 times more increased complexity relative to an XYZ parametrization. However, EKF can start to use landmarks in update equations at first sight.

2.4.3 Motion Model

The motion model in an EKF framework is used to derive the expected position of the robot in the current time step. This expected position is important because it is used for predicting landmark observations using the sensor model. If a mobile platform has wheels, then odometry is the most obvious choice for the motion model. Using measurements from sensors connected to wheels, the turning angle could be measured. Using these angles displacement and new orientation of the robot can be calculated [4]. However, odometry is still a very approximate

method, since it leads to errors in the orientation of the robot. However, if odometry is used with a vision system, the orientation of the robot can be corrected easily [20].

Unfortunately, odometry is not an option for legged robots. In such cases inertial sensors can also be used [19]. An inertial sensor gives acceleration and angular velocities in all dimensions, which can be integrated to get position and orientation information. If the robot system also has a vision system, displacement information could be derived from from an image sequence. Tracking landmarks in an environment gives displacement information for robot [30]. And yet another way to succeed same thing is optical flow [14].

But in some cases an extra sensor for motion model might not be preferred. And also, if using extra computation time on visual odometry is infeasible, then pure vision SLAM with a simple motion model is possible [9]. Having constant velocity motion model, and having impulse like acceleration noises, it is possible to make SLAM.

Chapter 3

Using Tree Width as a Sensor

3.1 Motivation and Framework

In stereo vision, point coordinates of the same landmark in two images can be used to calculate the depth of that landmark. Accuracy of this depth estimation depends on the baseline separation of these two cameras as well as the distance of the landmark from these cameras. Depth estimation is more accurate if the separation of cameras increases and the distance of the landmark from the cameras decreases. However, the separation of cameras has certain physical limits when building robotic systems. In contrast, when using monocular vision, one can know in which direction the landmark really is, but cannot exactly estimate its distance. However, we can still use the shape of a known object to calculate its distance from the camera. In our case, we are using the width of a tree trunk to estimate its depth in an image. Given the position and the radius of a tree and using a projective camera model, we can find the width of the tree trunk in the image plane. As an additional piece of information, the tree width can also directly be used in a SLAM based system. We do not directly measure depth using the radius and the visible width of a tree from the image, but we rely on the SLAM procedure to indirectly estimate depth.

To be able to find the width of a tree trunk, our sensor model requires the

Symbol	Definition
$\mathbf{t}_r^W$	Robot position.
q_r^{WC}	Robot orientation as quaternion.
R_r^{WC}	Robot orientation as rotation matrix.
$\mathbf{t}_t$	Tree position.
r_t	Tree radius.

Table 3.1: Table of Definitions.

knowledge of the tree radius as an extra information compared to other sensor models which give only the position of the tree trunk in an image. This can be done by estimating the tree radius over time in a SLAM framework. Intuitively, using our sensor model, if we can find the width of the tree trunk in the image plane and if our sensor model is a one-to-one function, then we could find the unknown radius by using the inverse of our sensor model. Our inverse sensor model hence can give an estimate of the radius of tree trunk. Then, within the SLAM framework, the radius of tree trunk can be updated and will hopefully converge to its real value.

Our sensor model and its inverse give us the ability to predict the width of a tree trunk and initialize its radius. We need the Jacobian of these sensor models to integrate them into a SLAM based system. The Jacobian of the sensor model will be used to update the landmark information using innovation values, and the Jacobian of the inverse sensor model is required to initialize covariance components within the SLAM algorithm.

In our framework, we have two coordinate systems, one is the world frame W and the other one is a moving camera frame C as shown in Figure 3.1. The moving camera frame C can be defined as an orientation and a displacement defined in the world frame W. In addition to the camera frame, we also have the image plane I and the conversion between these two frames is done through a projective camera model. We assume that tree trunks as landmarks sit on a plane parallel to the x-y plane in W, and a tree trunk can be defined as a coordinate system in the world frame together with a radius value. Table 3.1 shows definitions we use in subsequent sections.

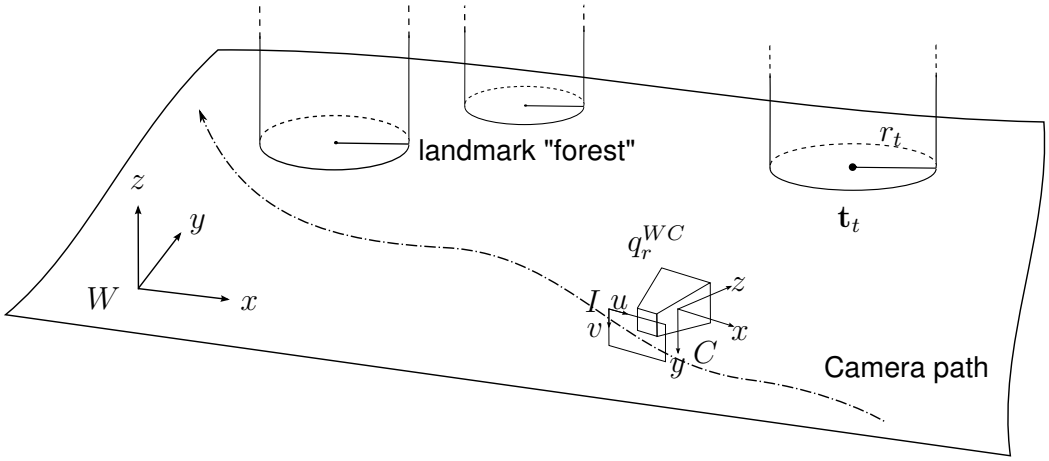

Figure 3.1: An illustration of a monocoular camera traveling in an environment populated with trees. Each tree landmark is located at $\mathbf{t}_t$ with trunk radius r_t.

3.2 Sensor Model for Tree Width Measurements

Our sensor model, for measuring the width of a tree trunk uses the position and the radius of the tree trunk and finds the coordinates of its center as well as the width of its trunk base in the image plane. We rely on the fact that the base of a tree is easily observable and relatively well defined. Figure 3.1 represents the robot and the tree trunk landmarks. Existing work enables us to extract this information from an image [43]. In order to derive a mathematical model for this sensor, we begin by assuming that the tree trunk can be approximated by an upright cylinder with the base coincident with the ground plane. We also assure that the root of the tree lies parallel to the x-y plane of the world coordinate system. Using only the bottom circle of the tree trunk cylinder, we can then find the center coordinates and the width of the tree trunk in the image plane. An overview of our sensor model is shown in Figure 3.2.

Mathematically, if a 2D conic section lies on a plane in 3D, it can easily be projected onto the image plane using a projective camera model [16]. Consequently, the projection of the base circle of the tree trunk will be an ellipse in the image plane. Since this projection occurs between two planes in space, it simplifies into

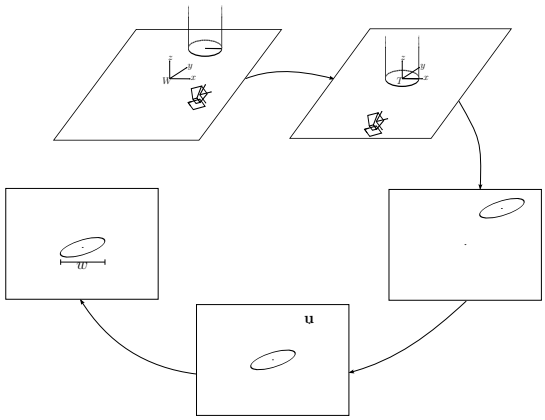

Figure 3.2: An overview of sensor model for tree width measurements.

a homography. Therefore, all we need is to find a relative homography between these two planes. Firstly, we need to have a transformation matrix which relates the tree plane to the image plane defined as

$$T := \begin{bmatrix} (R_r^{WC})^T & -(R_r^{WC})^T \mathbf{t}_r^W \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} I & \mathbf{t}_t \\ \mathbf{0} & 1 \end{bmatrix} , \tag{3.1}$$

where R_r^{WC} is orientation matrix corresponding to the quaternion q_r^{WC}. Since this equation already incorporates the position of the tree trunk base $\mathbf{t}_r^W$, the matrix defining the conic for the base of the tree trunk simplifies into

$$C_c := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -r_t^2 \end{bmatrix} . \tag{3.2}$$

Using the transformation matrix in (3.1), we obtain the desired homography matrix as

$$H := K A_1 T A_2 , \tag{3.3}$$

where K is the camera matrix [16]. Desired rows and columns of T are selected using extraction matrices A_1 and A_2. Since we now have a homography between two planes, we do not need all rows and colums of the transformation matrix T

and A_1 and A_2 are defined as

$$A_1 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \ , \ A_2 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \ . \tag{3.4}$$

Using this homography matrix H, we can project the base circle of tree trunk into the corresponding ellipse in the image plane [16] by

$$C_e = H^{-T} C_c H^{-1} \ . \tag{3.5}$$

After projecting the base circle of the tree trunk into the image plane, we get an ellipse equation of the form $x^T C_e x = 0$. A generic ellipse equation like this one can be represented in matrix form as

$$C_e := \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \ . \tag{3.6}$$

We now wish to find the center point of this ellipse so that we can transform its center point to the origin. Using the generic equation of an ellipse we can find its center point as

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = - \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}^{-1} \begin{bmatrix} d/2 \\ e/2 \end{bmatrix} \ . \tag{3.7}$$

Our sensor model uses this point as the position of the landmark in the image plane. Using this point we can translate the center of the ellipse to the origin to yield

$$C_{ce} = \begin{bmatrix} a & b/2 & 0 \\ b/2 & c & 0 \\ 0 & 0 & f + \frac{dx_c + ey_c}{2} \end{bmatrix} \ . \tag{3.8}$$

After centralizing ellipse we scale it first as

$$C_{ce} = \begin{bmatrix} a & b/2 & 0 \\ b/2 & c & 0 \\ 0 & 0 & -1 \end{bmatrix} \ . \tag{3.9}$$

From this point on, we can find the extrema of this ellipse in the x axis using the solution

$$x_{ext} = \pm\sqrt{\frac{-4c}{b^2 - 4ac}}\,, \tag{3.10}$$

which will be used to find horizontal length of ellipse as

$$w = 2\sqrt{\frac{-4c}{b^2 - 4ac}}\,. \tag{3.11}$$

In summary, this sensor model first uses the homography from the tree base plane to the image plane to project the base circle of a tree trunk into an ellipse, then finds extreme points of this new ellipse to get the width of the tree trunk. Unfortunately, we have a problem when the camera and the base circle of tree trunk lies on same plane, which makes H singular. If the camera even slightly away from the plane of the base circle, this is not a important problem in the sensor model, but when we take the jacobian using these equations derivatives became quite erroneous due to numerical issues. To solve this problem, we decided not to take the inverse of the homography matrix H, but instead solved equations using the inverse of the ellipse matrix. Detailed formulation of our sensor model which uses the inverse of the ellipse matrix could be found in the Appendix A.1.

3.3 The Inverse Tree-Width Sensor Model

Our sensor model requires knowledge of the radius for the tree landmark. Therefore, we have to initialize the radius of the tree landmark at some point. When the position of a tree landmark is known with enough certainty, we can initialize the radius of the tree landmark using the position of the tree landmark in world coordinates and the width measurement from the image sensor. We estimate the inverse of the centered ellipse matrix in the image plane, which we then project back to a circle in the world coordinate frame to give us the radius of the tree landmark. Using the width from the sensor, we can calculate the inverse of the

centered ellipse using (A.19)

$$C_{ce}^{-1} = \begin{bmatrix} -\frac{w^2}{4} & b & 0 \\ b & c & 0 \\ 0 & 0 & 1 \end{bmatrix} , \tag{3.12}$$

where w is the width from the image sensor. Since width information does not give us an exact ellipse, b and c are unknown. Fortunately these values can be estimated using the position of the landmark in world coordinates. We assume that the radius of the tree landmark does not affect the orientation of the ellipse in the image plane. Using this assumption, we can get the inverse of an ellipse using our sensor model as

$$C_{rce}^{-1} = \begin{bmatrix} A & B & 0 \\ B & C & 0 \\ 0 & 0 & F \end{bmatrix} . \tag{3.13}$$

Then, using the ratios from this ellipse, we can compute an estimation of the inverse of the centered ellipse as

$$C_{ece}^{-1} = \begin{bmatrix} -\frac{w^2}{4} & -\frac{w^2 B}{4A} & 0 \\ -\frac{w^2 B}{4A} & -\frac{w^2 C}{4A} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{3.14}$$

Subsequently, we use the position of the ellipse in image coordinates to get the inverse of the ellipse as

$$C_{ee}^{-1} = \begin{bmatrix} 1 & 0 & u_c \\ 0 & 1 & v_c \\ 0 & 0 & 1 \end{bmatrix} C_{ece}^{-1} \begin{bmatrix} 1 & 0 & u_c \\ 0 & 1 & v_c \\ 0 & 0 & 1 \end{bmatrix}^T , \tag{3.15}$$

where u_c and v_c are the coordinates of the ellipse in the image. We can then project back this inverse ellipse into an inverse circle using the homography of (3.3), to yield

$$C_{ec}^{-1} = H^{-1} C_{ee}^{-1} H^{-T} , \tag{3.16}$$

where this inverse circle has the form

$$C_{ec}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & -\frac{1}{r^2} \end{bmatrix} , \tag{3.17}$$

with r the radius of tree landmark we are interested in and k a constant very close to 1. This makes this 2D conic section an ellipse, that it is very close to a circle.

3.4 The Jacobian of the Tree Width Sensor Model

In a SLAM framework, the Jacobian of the sensor model is required in the update step. Since we formulated a new sensor model, we also need to find its Jacobian. This Jacobian has an extra row for the width sensor and an extra column for the radius of tree trunk, compared to the Jacobian of a sensor model which only incorporates the position of the tree trunk. Since inputs to our sensor model do not depend on each other, we can divide our Jacobian into four pieces. It can hence be written in the most general form as

$$\frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h}{\partial \mathbf{t}_r^W} & \frac{\partial h}{\partial q_r^{WC}} & \frac{\partial h}{\partial \mathbf{t}_t} & \frac{\partial h}{\partial r_t} \end{bmatrix} , \tag{3.18}$$

where $\frac{\partial h}{\partial \mathbf{t}_r^W}$, $\frac{\partial h}{\partial q_r^{WC}}$, $\frac{\partial h}{\partial \mathbf{t}_t}$ and $\frac{\partial h}{\partial r_t}$ are Jacobians of our sensor model with respect to the robot position, the robot orientation, the tree position and the tree radius, respectively. These Jacobians can be formulated as

$$\frac{\partial h}{\partial \mathbf{t}_r^W} = \frac{\partial h}{\partial iCe}\frac{\partial iCe}{\partial H}\frac{\partial H}{\partial T}\frac{\partial T}{\partial \mathbf{t}_r^W} , \tag{3.19}$$

$$\frac{\partial h}{\partial q_r^{WC}} = \frac{\partial h}{\partial iCe}\frac{\partial iCe}{\partial H}\frac{\partial H}{\partial T}\frac{\partial T}{\partial R_r^{WC}}\frac{\partial R_r^{WC}}{\partial q_r^{WC}} , \tag{3.20}$$

$$\frac{\partial h}{\partial \mathbf{t}_t} = \frac{\partial h}{\partial iCe}\frac{\partial iCe}{\partial H}\frac{\partial H}{\partial T}\frac{\partial T}{\partial \mathbf{t}_t} , \tag{3.21}$$

$$\frac{\partial h}{\partial \mathbf{t}_r^W} = \frac{\partial h}{\partial iCe}\frac{\partial iCe}{\partial r_t} . \tag{3.22}$$

Detailed derivation of this Jacobian matrix can be found in the Appendix A.2.

SLAM implementations also require the Jacobian of the inverse sensor model. We could directly calculate the Jacobian of the inverse sensor model, but since we

already have the Jacobian of the sensor model, we can take its inverse to obtain the same result. Our sensor model can be defined as

$$\begin{bmatrix} u & v & w \end{bmatrix}^T = f(\mathbf{t}_r^W, q_r^{WC}, \mathbf{t}_t, r_t) \ , \tag{3.23}$$

and inverse of it can be defined as

$$r_t = f^{-1}(\mathbf{t}_r^W, q_r^{WC}, \mathbf{t}_t, u, v, w) \ . \tag{3.24}$$

We compute the inverse of the Jacobian approximately using simplified functions. We do not need to calculate $\frac{\partial r}{\partial u}$ and $\frac{\partial r}{\partial v}$, which simplifies these formulas to

$$w = f(\mathbf{x}, r_t) \ , \tag{3.25}$$

$$r_t = f^{-1}(\mathbf{x}, w) \ , \tag{3.26}$$

where $\mathbf{x}$ is includes $\mathbf{t}_r^W$, q_r^{WC} and $\mathbf{t}_t$. With this reduction, we do not have to take pseudo inverse of the Jacobian matrix.

We take derivatives of (3.25) and (3.26), which yields

$$\begin{bmatrix} \frac{\partial w}{\partial \mathbf{x}} & \frac{\partial w}{\partial r} \end{bmatrix} = J_f \begin{bmatrix} \frac{\partial \mathbf{x}}{\partial \mathbf{x}} & \frac{\partial \mathbf{x}}{\partial r} \\ \frac{\partial r}{\partial \mathbf{x}} & \frac{\partial r}{\partial r} \end{bmatrix} \ , \tag{3.27}$$

$$\begin{bmatrix} \frac{\partial r}{\partial \mathbf{x}} & \frac{\partial r}{\partial w} \end{bmatrix} = J_f^{-1} \begin{bmatrix} \frac{\partial \mathbf{x}}{\partial \mathbf{x}} & \frac{\partial \mathbf{x}}{\partial w} \\ \frac{\partial w}{\partial \mathbf{x}} & \frac{\partial w}{\partial w} \end{bmatrix} \ . \tag{3.28}$$

We use part of (3.27), and extend it as

$$\frac{\partial w}{\partial \mathbf{x}} = J_{fx} \frac{\partial \mathbf{x}}{\partial \mathbf{x}} + J_{fr} \frac{\partial r}{\partial \mathbf{x}} \ , \tag{3.29}$$

where J_{fr} is the last element of J_f and J_{fx} is the rest. This function yields to

$$\frac{\partial r}{\partial \mathbf{x}} = -\frac{J_{fx}}{J_{fr}} \frac{\partial \mathbf{x}}{\partial \mathbf{x}} + \frac{1}{J_{fr}} \frac{\partial w}{\partial \mathbf{x}} \ , \tag{3.30}$$

which can be rewritten in matrix form as

$$\frac{\partial r}{\partial \mathbf{x}} = \begin{bmatrix} -\frac{J_{fx}}{J_{fr}} & \frac{1}{J_{fr}} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{x}}{\partial \mathbf{x}} \\ \frac{\partial w}{\partial \mathbf{x}} \end{bmatrix} \ , \tag{3.31}$$

which is part of (3.28). Hence, J_f^{-1} can be computed using

$$J_f^{-1} = \begin{bmatrix} -\frac{J_{fx}}{J_{fr}} & \frac{1}{J_{fr}} \end{bmatrix} . \tag{3.32}$$

Using simplified formula of the sensor model and its inverse makes calculations simple and we only need a fraction of the Jacobian. This estimation is close enough to use for initialization of the radius of tree landmarks.

3.5 Putting it all together

In our SLAM environment, a tree trunk landmark is parameterized in three different parametrizations from when it is first seen to the end of the SLAM run as shown in Figure 3.3. These three parametrizations are: Inverse depth, XYZ and XYZ-R.

When a landmark is first seen, it is initialized in an inverse depth parametrization [9]. This enables us to start using landmarks at the beginning of the algorithm without knowledge of their depth. In time, their depth estimation will be good enough and they can be safely turned into an XYZ parametrization as described in [8].

When a landmark uses an inverse depth parametrization, its position estimate is very weak and cannot be used to estimate the radius of the tree. But when their position estimate becomes good enough, they can be converted to an XYZ parametrization. At this point, we can also estimate the radius of the tree landmark using our inverse sensor model. Using the radius from the inverse sensor model, the landmark can be converted into what we call an XYZ-R parametrization. However, to be able to convert a landmark into XYZ-R parametrization, we also need to find the uncertainty associated with the radius, which can be computed using the Jacobian of our inverse sensor model for the tree radius.

Once a landmarks is in an XYZ-R parametrization, we can start to use the tree-width sensor. Now we have extra width information from tree landmarks,

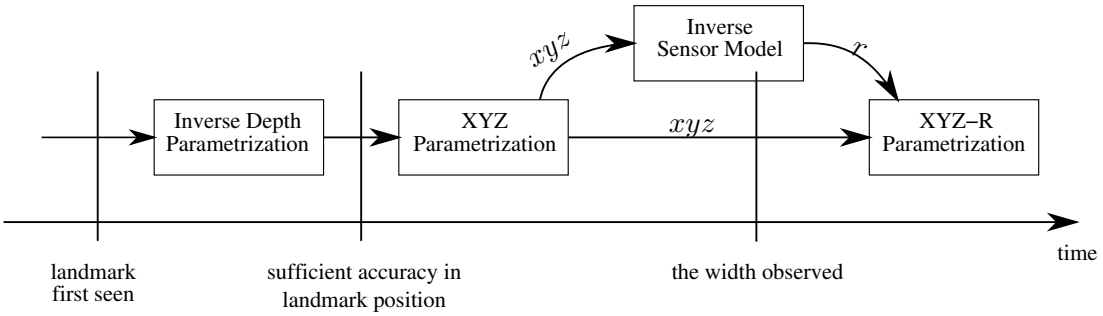

Figure 3.3: Life cycle of a landmark.

which we can use in SLAM update equations. The Jacobian of our new sensor model can similarly be used in the update equations.

Chapter 4

SLAM Performance Using Tree Widths

4.1 Experimental Procedure

For our experimental results, we use synthetic data that we generated in a simulation environment we created for that purpose. Our simulation environment includes a virtual camera moving along a user defined path while taking images of a virtual world populated with cylindrical "trees" at every time step. In Section 4.1.1, we describe the details of this simulation system.

Using the EKF SLAM of Civera [10], we perform SLAM using measurements from our simulation environment. To be able to use the tree trunk width as an extra sensory information incorporated to the EKF SLAM implementation, we added an additional sensor model and its Jacobian to the implementation.

Using ground truth locations of landmarks and camera trajectory from the simulation environment, together with the output of the EKF SLAM that gives estimated landmark locations and the camera trajectory we can evaluate the performance of our SLAM algorithm.

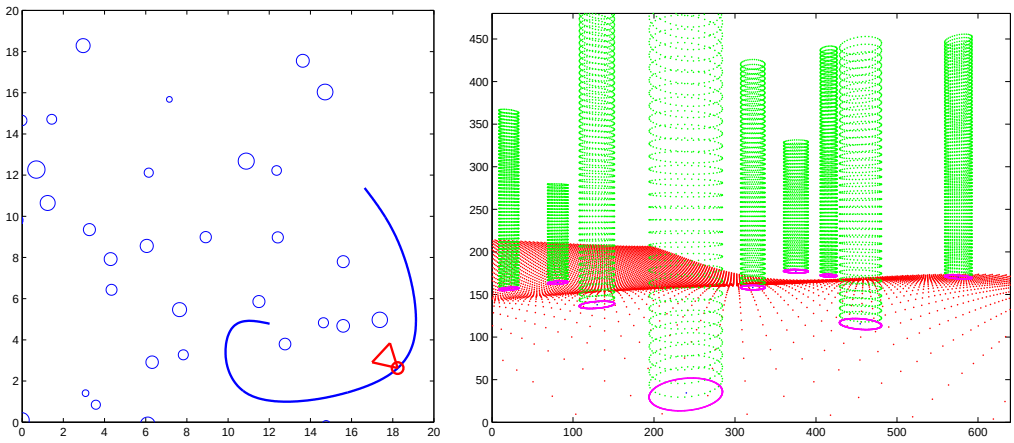

Figure 4.1: Left: Top view of tree trunk landmarks and the predefined Bezier trajectory. Right: Tree trunk landmarks from the viewpoint of the camera positioned on the circle marker in the left figure.

4.1.1 The Simulation Environment

Our simulation environment implements a virtual forest of tree trunks, generated randomly in a predefined area. We do this on a predefined grid, using Gaussian noise on tree locations. With some probability, we add a tree to a grid cell, and then add Gaussian noise on the placement of the tree. The radius of the tree also chosen randomly using a separate Gaussian distribution. As for the height of the root of the tree, we predefine a ground elevation map using simple functions. As a result, we have a randomly generated but realistic and uniformly distributed forest to use in our simulations. An example forest generated in this manner is shown in Figure 4.1.

Using the map of trees we randomly generated, we choose a trajectory through the forest. It is important that this trajectory is continuous and defined in a flexible way. To this end we select a number of points in the map and have the trajectory pass from these points, defining an associated Bezier curve. We then get samples from this Bezier curve to use as the position and orientation of the camera along the trajectory. An example trajectory is shown in Figure 4.1.

After generating random trees and defining the trajectory, we take images of the world along the trajectory. In every sample we get from the trajectory, we

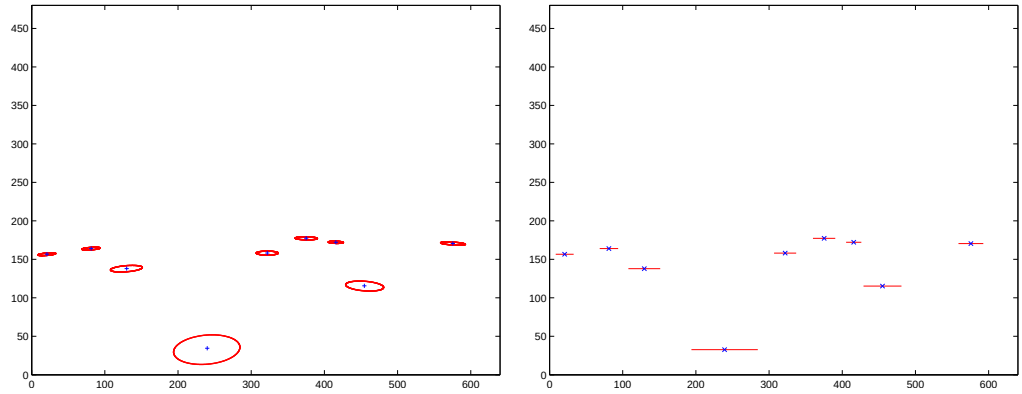

Figure 4.2: Left: Projected ellipses as tree trunk bases. Right: Center and width of tree trunk bases, obtained using our sensor model.

have a position and an orientation for camera. Our sensor model then determines which trees are visible on the image plane. We decide if a tree landmark is visible in the image plane or not using the camera properties. If the tree landmark is completely inside of the image area, then we add it into the array of measured trees which will be fed to the EKF SLAM. We also need to decide if a tree is occluded behind other trees or not. Since we know how far a tree is away from the camera, we say that a tree is occluded if it is behind another tree and they overlap in the x-axis. When we use only non-occluded trees in EKF SLAM these occluded trees are ignored. Finally, every tree landmark has a specific ID number to be used in EKF SLAM, so we do not address the data association problem. An example image and the corresponding tree locations and widths are shown in Figure 4.2.

4.1.2 Map Alignment and Error Metrics

EKF SLAM gives us a map of estimated landmark positions and an approximate trajectory. We need to compare these estimated values with the ground truth in order to be able to assess the performance of our algorithm. Unfortunately, the estimated map and the trajectory have an unknown scale, position and orientation due to the monocular nature of our framework. We need to correctly transform estimated map points and the trajectory to compute estimation errors. Using

different values of scaling, translation and rotation we will get different error values. Let the transformation be parameterized with the vector

$$v := [\mathbf{p}, R, s]^T \ , \tag{4.1}$$

where $\mathbf{p}$ is a translation, R is a rotation and s is a scale. Using v, we can define a map mismatch formula for a specific map M using the landmark error as

$$e_l^M(v) := \frac{1}{N} \sum_i \left\| T_v(\mathbf{p}_{mi}^M) - \mathbf{p}_{ri}^M \right\|^2 \ , \tag{4.2}$$

where $T_v(\mathbf{p})$ function defines a specific alignment of map using variable set v, $\mathbf{p}_{mi}^M$ is estimated landmark position and $\mathbf{p}_{ri}^M$ is real landmark position. We define $T_v(\mathbf{p}_i)$ as

$$T_v(\mathbf{p}_i) := sR(\mathbf{p}_i + \mathbf{p}) \ . \tag{4.3}$$

We need to find the transformation which minimizes map mismatch value as in

$$v_o^M := \underset{v}{argmin} \ \left(e_l^M(v) \right) \ . \tag{4.4}$$

We will use the optimum transformation to align landmarks and trajectories. For this optimization we will use Procrustes Analysis, which is a simple method which aligns two 3D point clouds where point-to-point correspondences are known [15]. We only align landmarks in our case, since we have a one-to-one point correspondence. This method follows three steps: translation, scaling and rotation. At the first step, mean values of landmarks are translated to the origin for both maps as in Figure 4.3 using

$$\mathbf{p}_i^* = \mathbf{p}_i - \bar{\mathbf{p}} \ , \tag{4.5}$$

where $\bar{\mathbf{p}}$ is the mean value of points. At the second step, root mean square distances are calculated and the maps are scaled accordingly to have same scale as in Figure 4.3. The root mean square error can be found using

$$s := \sqrt{\frac{\sum_i \left\| p_i^* \right\|^2}{k}} \ , \tag{4.6}$$

where k is the number of landmarks. Using s ground truth and estimated landmarks will be scaled to have unit scale, with each point scaled as

$$\mathbf{p}_i^{**} = \frac{p_i^*}{s} \ . \tag{4.7}$$

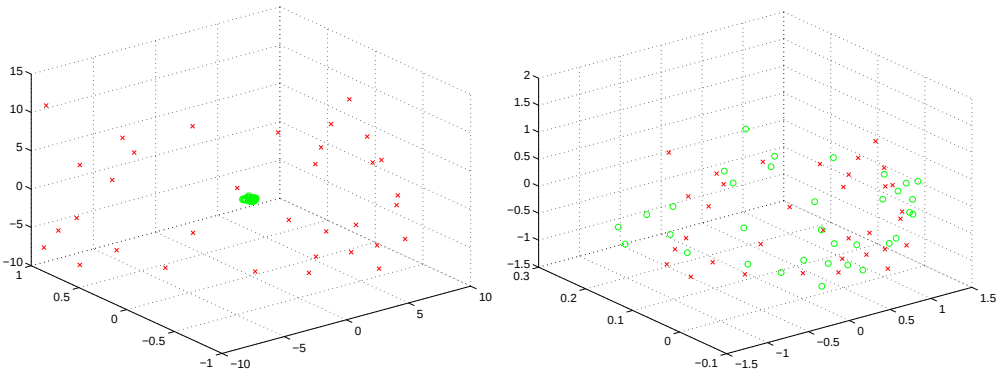

Figure 4.3: Left: The mean of the estimated landmarks and real landmarks are shifted to the origin. Right: Landmarks are scaled in order to have the same scale.

Finally, using the Kabsch Algorithm, an optimal rotation can be found. Using paired point sets in both maps, we can find a covariance matrix such as

$$A := P^T Q \ , \tag{4.8}$$

where P and Q are n-by-3 matrices representing the translated and scaled points of the estimated landmarks and the real landmarks. Calculating the SVD of the covariance matrix A gives us

$$A = VSW^T \ . \tag{4.9}$$

We also need to decide whether we need to correct our rotation matrix to ensure a right hand coordinate system

$$d = sign(det(A)) \ . \tag{4.10}$$

Finally we can get our rotation matrix as,

$$R := W \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} V^T \ . \tag{4.11}$$

Using this rotation matrix, we rotate estimated landmarks as in Figure 4.4.

Procrustes Analysis aligns the maps, but it translates and scales both estimated map and ground truth. At this point, estimated map and trajectory are aligned with the ground truth. However, ground truth does not have original

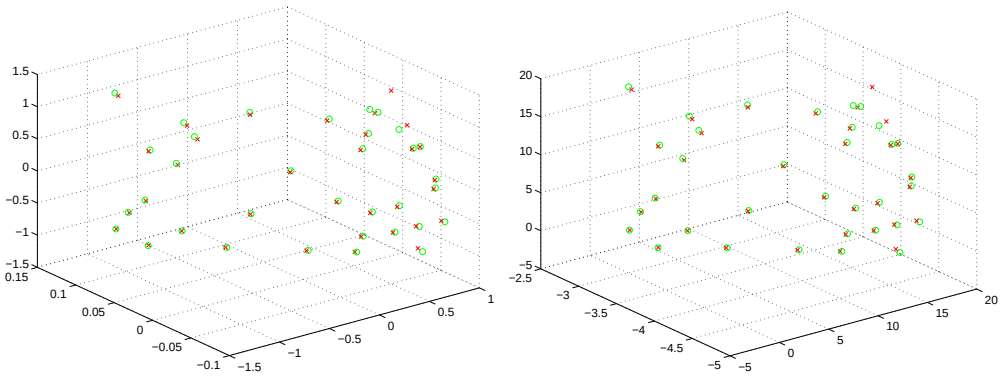

Figure 4.4: Left: Estimated landmarks are rotated, to yield an optimally aligned map. Right: Landmarks are scaled in order to have the original scale.

scale and mean position. Therefore, our maps needs to be scaled and translated, in order to regain the scale and mean position of the ground truth as in Figure 4.4.

Using aligned landmarks and trajectories, we can now compute error metrics. We have two different error metrics: Landmark error and Trajectory error. For computing the landmark error, we use the squared mean distance using pairs from real landmarks and estimated landmarks, as defined by (4.2). For the trajectory error, we take the position of the camera in every time step for both the estimated trajectory and the real trajectory; and find the mean squared distance error using pairs of positions as

$$e_t^M(v) := \frac{1}{N} \sum_i \left\| T_v(\mathbf{p}_{eti}^M) - \mathbf{p}_{rti}^M \right\|^2 \ , \tag{4.12}$$

where $\mathbf{p}_{eti}^M$ is the estimated trajectory position and $\mathbf{p}_{rti}^M$ is the real trajectory position at the time step i for map M. In both error metrics we use Euclidian distance between points.

During initial stages of EKF SLAM, landmarks may not converge to their real positions right away. Consequently, the estimated trajectory of the camera might not overlap with the actual trajectory. However, after some time, the trajectory estimate becomes closer to the real trajectory. Using an inaccurate and transient trajectory in our error analysis would give results that are not very meaningful. To prevent this, we choose a circular path and a trajectory to travel around this

path twice. At the end of the first lap, we compute the landmark error, and during the second lap, we use the estimated trajectory to find the trajectory error. This way we can get more meaningful, steady state error definitions.

4.1.3 EKF SLAM Implementation

In order to evaluate our sensor model, we need a functional SLAM environment. For this purpose, we use the EKF SLAM implementation by Civera [10]. In his work they implement a sensor model which takes information from landmarks in either inverse depth or XYZ parametrization and estimates the positions of landmarks in the image plane. This work also includes relevant Jacobians for the sensor model. An image sequence is fed into the EKF SLAM system, which is processed to find features to be used as specific landmarks.

In this thesis we adopt this implementation as a starting point. First, we need to feed the EKF SLAM our sensor data from the simulation environment, so we changed the implementation accordingly to make it read data from our image files. Using our data the EKF SLAM implementation works well using only inverse depth and XYZ parametrizations. However, when landmarks turn into the XYZ parametrization, we use our inverse sensor model to estimate landmark radius and add it to the landmark representation through its mean vector and covariance matrix. With our XYZ-R parametrization, we have a mean vector for landmarks defined as

$$\overline{p} := \begin{bmatrix} x & y & z & r \end{bmatrix}^T , \tag{4.13}$$

where with additional radius parameter, we now have four parameters instead of three. This change corresponds to an increase in time complexity about 1.8. After changing to this new parametrization, we use our own sensor model to predict these XYZ-R parameterized landmarks. To update these landmarks, we used the Jacobian of our sensor model we formulated in previous sections.

4.2 Selection of EKF Gain Matrices

Incorporating our sensor model into the EKF SLAM requires us to define how much we trust our new sensor. In EKF SLAM, we normally define gain matrices for the sensor model that show how much trust we put in the sensor model and the motion model, respectively. We use diagonal gain matrices where diagonal entries determine how much relative trust we place in measurements of the sensor. However, before we can find an optimal gain matrix for our sensor model, we need to first find an optimal gain matrix for a sensor model without width measurements from tree trunks. We will use this limited sensor model until a landmark changes into the XYZ-R parametrization. Therefore, we need to optimize gain matrix for it first.

The limited sensor model yields the position of the tree trunk landmark in the image plane. This gives two measurements, which intuitively should have the same gain, leading to one gain value overall. Gain matrix for this sensor model thus becomes

$$K_1 := \begin{bmatrix} w & 0 \\ 0 & w \end{bmatrix} . \tag{4.14}$$

We performed systematic experiments, using different values for w. Two different sets of experiments were performed. First, using all trees by ignoring occlusion, and then only using non-occluded trees. We have used the following weight settings:

$$w \in \{0.5, 1, 2, 4, 8, 16, 32, 48, 64, 80, 96, 128\} . \tag{4.15}$$

To select the optimum weight, we tested every different weight setting for 10 different maps. We defined negative average landmark error and negative average trajectory error to be able to evaluate results. Negative average landmark error is defined as

$$A_{al} := \frac{1}{n} \sum_{M=1}^{n} max(-10, -e_l^M(v_o)) , \tag{4.16}$$

and negative average trajectory error is defined as

$$A_{at} := \frac{1}{n} \sum_{M=1}^{n} max(-3, -e_t^M(v_o)) \ . \tag{4.17}$$

These error functions let us take the average of errors from different maps where we saturate errors at some certain level to have smoother results. We choose the values 10 for landmark error and 3 for trajectory error to be maximum error that can be classified as converged test runs. Diverged runs may have tremendous error values, averaging them yields meaningless results.

In Figure 4.5, we show negative average landmark error and negative average trajectory error where we ignore occlusion. In Figure 4.6, the only difference is that we use only non-occluded trees. Looking at these error values, we can clearly see that when w is smaller than 20, the negative average errors are very big, showing that SLAM diverges most of the time. But for bigger w values above 60, it seems that there was no big difference between different values. But as we can see even if the difference is small after some point, the performance of the algorithm gets worse with increasing w. From both of these observations, we can select w as 80, which gives one of the best results. After selecting w, we can proceed with evaluations for selecting the optimal gains for our new sensor model. Until we estimate the radius of the tree landmarks, we will use this gain matrix with the other sensor model which does not uses radius of tree landmarks. Then we will switch to our sensor model using the gain matrix which is optimal for it.

Our sensor model yields position and width measurements from the tree trunk landmark, so we need to define three values for each measurement. The position measurements should have the same gain, leading to two gain values overall. Our gain matrix thus becomes

$$K_2 := \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_1 & 0 \\ 0 & 0 & w_2 \end{bmatrix} \ . \tag{4.18}$$

We have performed similar experiments to the ones we done for finding the optimal gain matrix for the limited sensor model. Once again we used different

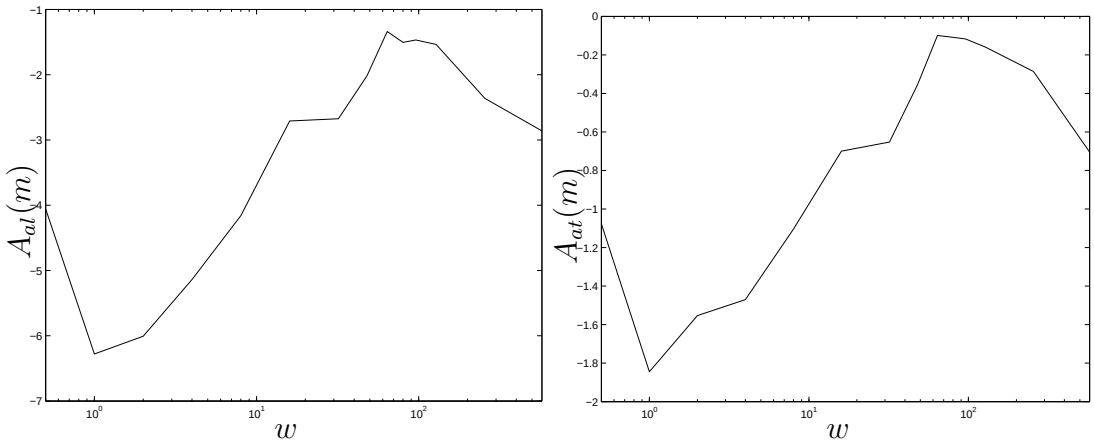

Figure 4.5: Performance analysis for other sensor model which does not use the radius of tree landmarks. In these experiments we ignore occlusion. Left: Negative average landmark error for different gain pairs for 10 different maps. Right: Negative average trajectory error.

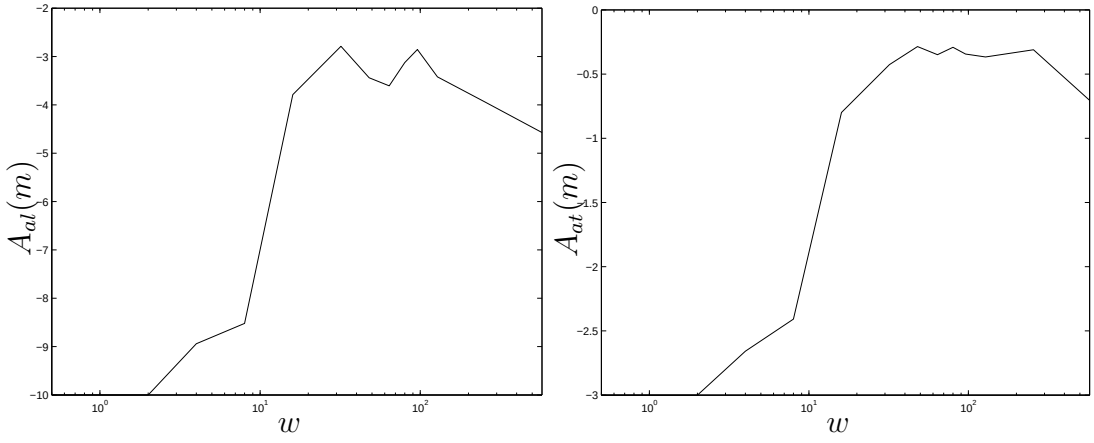

Figure 4.6: Performance analysis for other sensor model which does not use the radius of tree landmarks. In these experiments we only use non-occluded trees. Left: Negative average landmark error for different gain pairs for 10 different maps. Right: Negative average trajectory error.

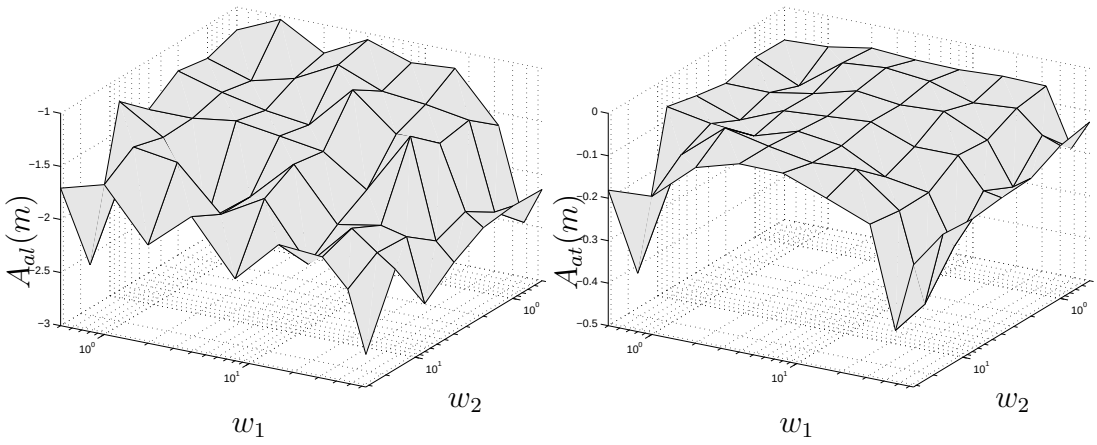

Figure 4.7: Performance analysis for our sensor model. In these experiments we ignore occlusion. Left: Negative average landmark error for different gain pairs for 10 different maps. Right: Negative average trajectory error.

weight settings for these experiments as

$$w_1 \in \{0.5, 1, 2, 4, 8, 16, 32, 48, 64\}, \tag{4.19}$$

$$w_2 \in \{0.5, 1, 2, 4, 8, 16, 32\} . \tag{4.20}$$

To select the optimum pair of weights we tested every combination of weights w_1 and w_2 for 10 different maps. In Figure 4.7, we show negative average landmark error and negative average trajectory errors with occlusions ignored. In Figure 4.8, the only difference is that we use only non-occluded trees. In Figure 4.7, when we ignore occlusion, we can see that selecting smaller values for w_1 and w_2 gives better results. When w_2 gets bigger the performance of the algorithm gets worse, but as we see w_1 is not affected much until it passes over 30. On the other hand in Figure 4.8, we can see that if there is occlusion then selecting very small values for w_1 and w_2 makes results in bad performance. Using the information from these two different set of experiments selecting w_1 16 and w_2 4 seems promising, but many other parameters could be used as well.

Figure 4.9 illustrates an example with the left and right figures generated with and without the tree-width sensor. Note that our new sensor is also capable of estimating the radii of trees.

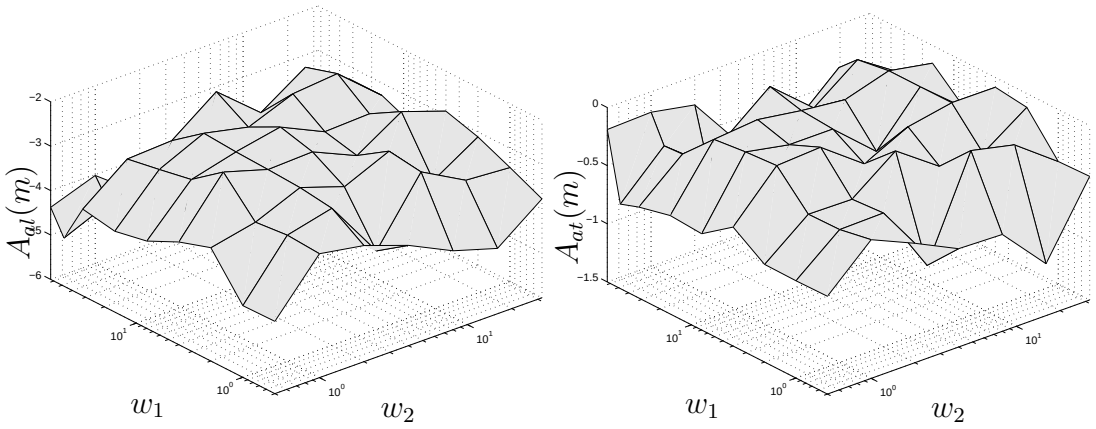

Figure 4.8: Performance analysis for our sensor model. In these experiments we only use non-occluded trees. Left: Negative average landmark error for different gain pairs for 10 different maps. Right: Negative average trajectory error.

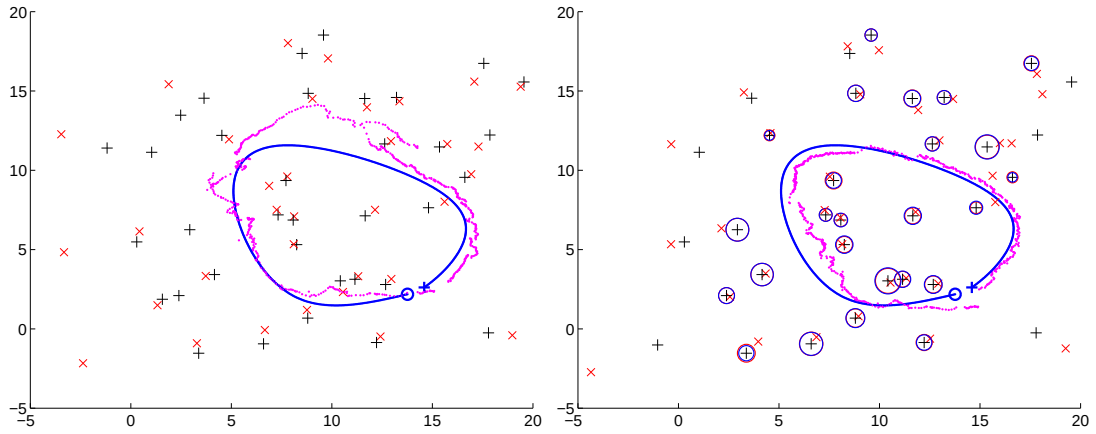

Figure 4.9: An example SLAM run without (left) and with (right) the tree-width sensor. Plus and cross signs show real and estimated landmark locations. The right figure also shows real and estimated tree radii for those landmarks that have been converted into the XYZ-R parameterization. In both figures, the jagged curves show the estimated trajectories.

4.3 Performance Using All Trees, Occlusions ignored

We evaluate SLAM performance through two sets of systematic experiments. First, we compare performance with and without the tree-width sensor under different levels of pixel noise on the image. Second, we evaluate algorithm performance for different tree densities in the forest. In both cases, we use the average landmark error

$$E_{al} := \frac{1}{n} \sum_{M=1}^{n} e_l^M(v_o) \ , \tag{4.21}$$

and the average trajectory error

$$E_{at} := \frac{1}{n} \sum_{M=1}^{n} e_t^M(v_o) \ . \tag{4.22}$$

as performance metrics, taking only convergent runs with $e_l^M(v_o) < 10$ and $e_t^M(v_o) < 3$ into account. We also report the ratio of convergent runs to the total number of runs as an additional performance metric.

4.3.1 Dependence on Noise

Figure 4.10 compares the convergence ratios with (solid) and without (dashed) the tree-width sensor considering all trees, under noise levels with standard deviations ranging from 1 pixel to 10 pixels. For each noise level, 10 different noise vectors with the same variance are used for each of the 10 maps for a total of 100 runs to ensure statistical validity. Our results show that under noise levels with standard deviation larger than 5 pixel, better convergence ratios has been seen with the proposed tree-width sensor.

Average trajectory and landmark errors associated for these experiments are shown in Figure 4.11, which also show standard error bars. Our results show that the incorporation of the tree-width as an additional sensor results in notable improvements for trajectory error. As for landmark error it does not affect much. The relatively large magnitude of these error figures results from our thresholds

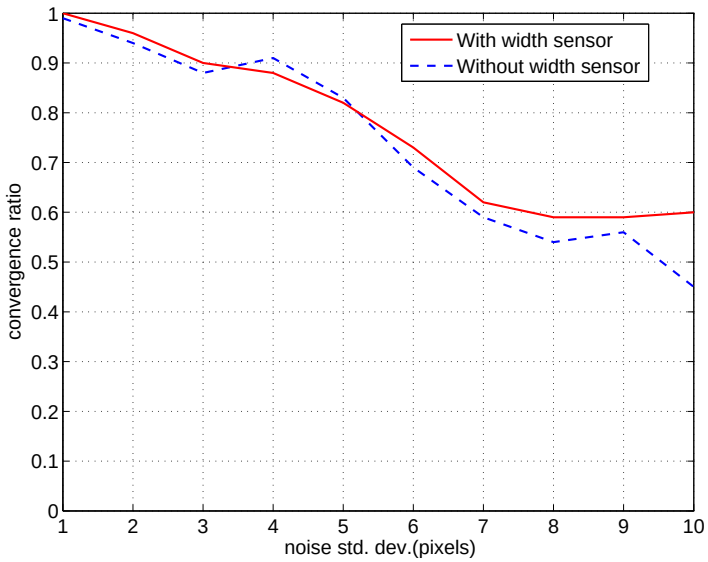

Figure 4.10: EKF convergence ratios with (solid) and without (dashed) the tree-width sensor for different pixel noise levels. Occlusions ignored, all trees are considered.

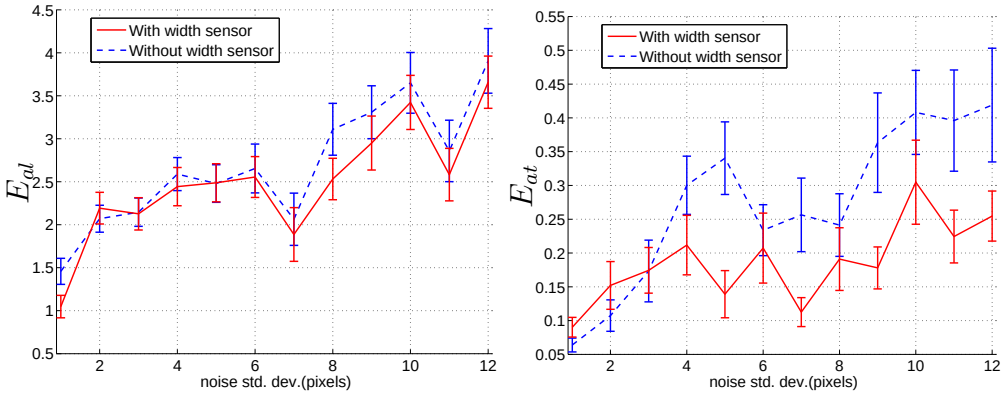

Figure 4.11: Average landmark (left) and trajectory (right) errors for SLAM with (solid) and without (dashed) the tree-width sensor for different pixel noise levels. Occlusions ignored, all trees are considered.

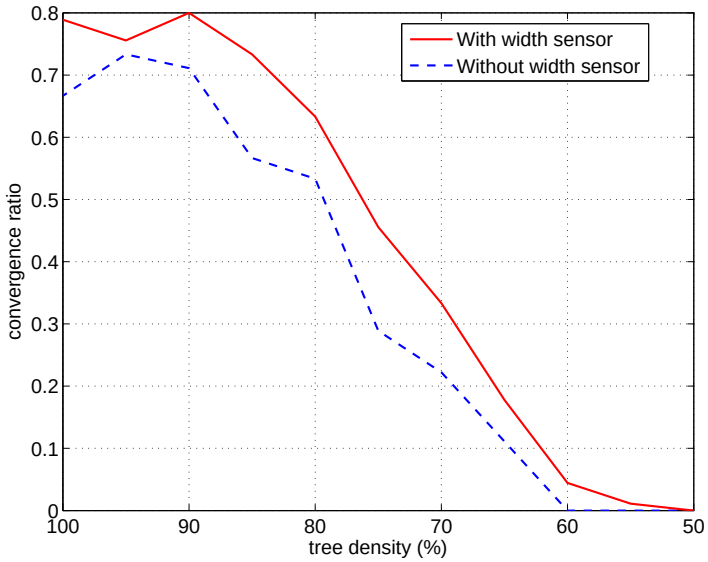

Figure 4.12: EKF convergence ratios with (solid) and without (dashed) the tree-width sensor for different tree densities. Occlusions ignored, all trees are considered.

for divergence, but this does not invalidate the relative improvements resulting from the tree-width sensor.

4.3.2 Dependence on Tree Density

Our second set of experiments consider the effect of tree density in the environment on algorithm performance. Having chosen equal tree densities for each of the 10 maps being considered, we randomly remove individual trees from each map to achieve a fraction of this original density. To achieve statistical validity, we repeat this 3 times for each lower density for each map, averaging the results across 90 runs, also using 3 different noise vectors in each case. Convergence performances are shown in Figure 4.12, establishing that the use of tree-width measurements improves filter performance at lower tree densities.

Similarly, Figure 4.13 shows average landmark and trajectory errors for the same experiments. The tree-width sensor improves SLAM accuracy in general, with erratic results below 70% tree density observed primarily due to the very low

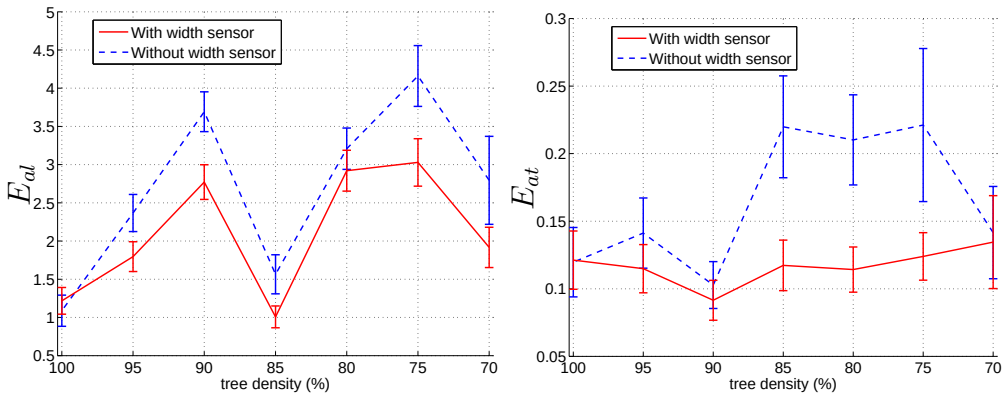

Figure 4.13: Average landmark (left) and trajectory (right) errors for SLAM with (solid) and without (dashed) the tree-width sensor for different tree densities. Occlusions ignored, all trees are considered.

convergence ratios (lower than 10%) for both with and without the new sensor. These results show that the use of tree-width measurements and the estimation of the radii associated with tree landmarks in an outdoor environment helps with convergence and accuracy issues for the use of these natural landmarks in a SLAM framework.

4.4 Performance Using Only Non-occluded Trees

We have repeated both test again, this time considering only non-occluded trees.

4.4.1 Dependence on Noise

Figure 4.14 compares the convergence ratios with (solid) and without (dashed) the tree-width sensor only considering non-occluded trees. Our results show that close to a 10% improvement on convergence ratios has been possible with the proposed tree-width sensor. Comparing with the convergence ratios in Figure 4.10, we can see that under noise levels with standard deviation larger than 2 pixel, better convergence ratios has been seen when occlusions are considered.

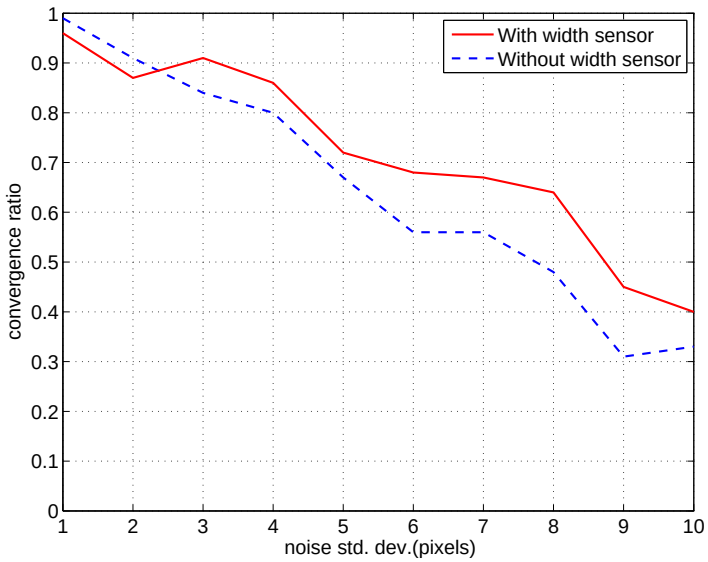

Figure 4.14: EKF convergence ratios with (solid) and without (dashed) the tree-width sensor for different pixel noise levels. Only non-occluded trees are considered.

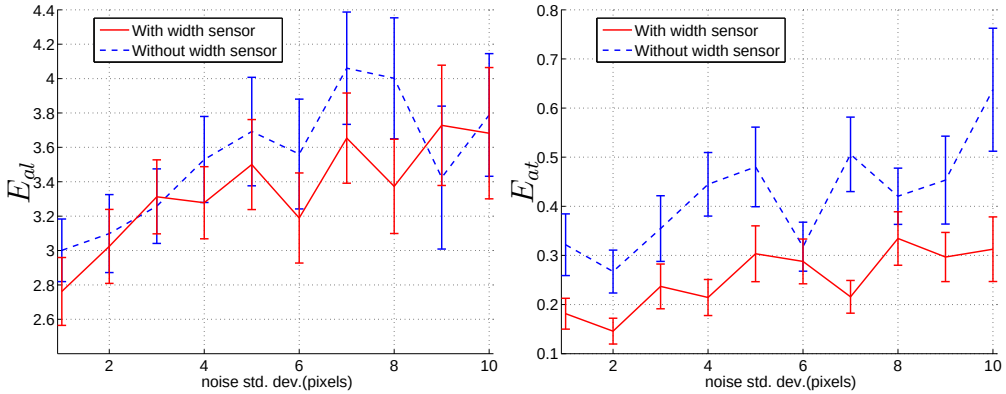

Figure 4.15: Average landmark (left) and trajectory (right) errors for SLAM with (solid) and without (dashed) the tree-width sensor for different pixel noise levels. Only non-occluded trees are considered.

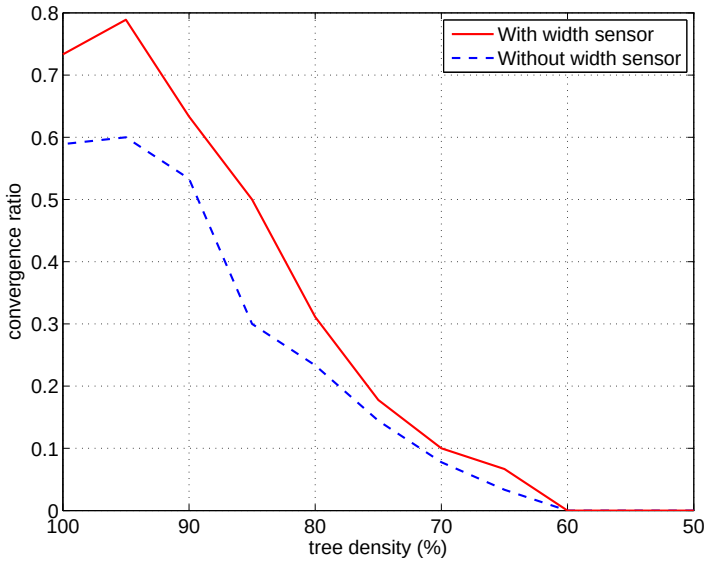

Figure 4.16: EKF convergence ratios with (solid) and without (dashed) the tree-width sensor for different tree densities. Only non-occluded trees are considered.

Average trajectory and landmark errors associated for these experiments are shown in Figure 4.15, which also show standard error bars. Our results show that the incorporation of the tree-width as an additional sensor results in notable improvements for both error metrics. Comparing with the Figure 4.11, we could see that when occlusions are considered our algorithm shows much better improvements.

4.4.2 Dependence on Tree Density

Convergence performances are shown in Figure 4.16, establishing that the use of tree-width measurements improves filter performance at lower tree densities as similar to the performances in Figure 4.12.

Similarly, Figure 4.17 shows average landmark and trajectory errors for the same experiments. We have seen similar results as in Figure 4.13.

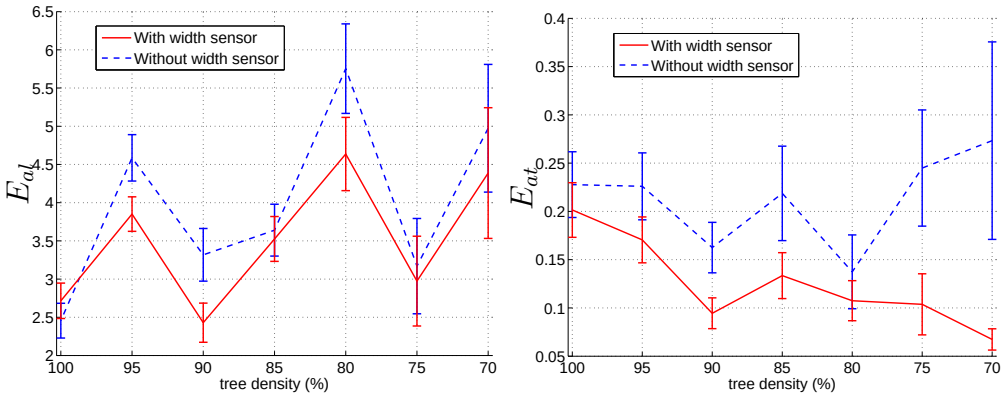

Figure 4.17: Average landmark (left) and trajectory (right) errors for SLAM with (solid) and without (dashed) the tree-width sensor for different tree densities. Only non-occluded trees are considered.

Chapter 5

Conclusion

In this thesis, we focus on using shape information from landmarks as an additional input to the SLAM. Our main contribution is to use the width of trees as an additional sensory input, by means of incorporating radius information to each tree on the map. To be able to accomplish this we derived a width-sensor model for the extra width measurement, relevant Jacobians and an inverse sensor model in order to use within a SLAM framework.

Using a working SLAM implementation, we incorporated our sensor model and inverse sensor model with the relevant Jacobians. We defined a new parametrization called XYZ-R, in order to accomplish this. To be able to test our new sensor model in a SLAM environment we created a simulation environment which generates data to be used in SLAM simulations. Then using data from our simulation environment and conducting systematic experiments, we found optimal gain matrices for the sensor model, which uses width information from tree trunks as an additional sensory reading and for the sensor model which only uses the position of landmarks. Using the optimum gain matrices for both sensor models we conducted two different sets of systematic experiments. The first experiment set shows that with increased noise, our sensor model shows better performance compared to the other sensor model. The second experiment set shows that our sensor model works much better than the other sensor model when the number of available tree landmarks decrease. With our new sensor

model, convergence ratio and localization improved greatly, and the estimation of landmark position slightly improved.

One of the limitations of our work is that, it requires sufficient certainty in landmark positions before our sensor model can be used. In future studies, if we could also incorporate the radius of trees into the inverse depth parametrization, we would be able to use extra sensory information from the very beginning. We believe this would improve the convergence rate of the SLAM environment as well as the convergence ratio.

Appendix A

Derivations

A.1 The Sensor Model Derivations

When the camera and the circle of the trunk base is on the same plane, the homography matrix become singular. To get the matrix defining the ellipse, we were applied

$$C_e = H^{-T} C_c H^{-1} \,, \tag{A.1}$$

where H can be singular. However, we can use the inverse of the ellipse by applying

$$C_e^{-1} = H C_c^{-1} H^T \,, \tag{A.2}$$

where we need inverse of the matrix defining the circle which is

$$C_c^{-1} := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -\dfrac{1}{r_t^2} \end{bmatrix} \,. \tag{A.3}$$

Using the inverse of the ellipse we need to find its center and width. We have already show how to find these using the ellipse matrix in Section 3.2. Let the generic formulation of the inverse of the ellipse be

$$C_e^{-1} := \begin{bmatrix} B & \mathbf{V} \\ \mathbf{V}^T & f \end{bmatrix} \,, \tag{A.4}$$

where B is a 2-by-2 matrix and $\mathbf{V}$ is a 2-by-1 vector. Taking block inversion of this matrix corresponds to

$$C_e = \begin{bmatrix} \left(B - \frac{1}{f}\mathbf{V}\mathbf{V}^T\right)^{-1} & -\frac{1}{f}\left(B - \frac{1}{f}\mathbf{V}\mathbf{V}^T\right)^{-1}\mathbf{V} \\ -\frac{1}{f}\mathbf{V}^T\left(B - \frac{1}{f}\mathbf{V}\mathbf{V}^T\right)^{-1} & \frac{1}{f} + \frac{1}{f^2}\mathbf{V}^T\left(B - \frac{1}{f}\mathbf{V}\mathbf{V}^T\right)^{-1}\mathbf{V} \end{bmatrix} , \quad \text{(A.5)}$$

where $-\frac{1}{f}\left(B - \frac{1}{f}\mathbf{V}\mathbf{V}^T\right)^{-1}\mathbf{V}$ and $-\frac{1}{f}\mathbf{V}^T\left(B - \frac{1}{f}\mathbf{V}\mathbf{V}^T\right)^{-1}$ are symmetric. For ease of use we will use

$$C_e = \begin{bmatrix} \alpha & \beta \\ \beta^T & \epsilon \end{bmatrix} . \quad \text{(A.6)}$$

Using ellipse matrix we can find center points of ellipse, we need to find a homography M which moves the center of ellipse to the origin. Applying this to the ellipse matrix would yield to

$$C_{ce} = M^T \begin{bmatrix} \alpha & \beta \\ \beta^T & \epsilon \end{bmatrix} M , \quad \text{(A.7)}$$

where M is a translation matrix defined as

$$M := \begin{bmatrix} I & \mathbf{u} \\ \mathbf{0} & 1 \end{bmatrix} . \quad \text{(A.8)}$$

After translation with M we have

$$C_{ce} = \begin{bmatrix} \alpha & \alpha u + \beta \\ u^T\alpha + \beta^T & u^T\alpha u + \beta^T u + u^T\beta + \epsilon \end{bmatrix} , \quad \text{(A.9)}$$

where $\alpha u + \beta$ should be 0 for the center point of C_e to be at the origin. Then the center of ellipse should be

$$u = -\alpha^{-1}\beta , \quad \text{(A.10)}$$

incorporating the correspondences from the ellipse matrix we get

$$u = -\left(B - \frac{1}{f}\mathbf{V}\mathbf{V}^T\right)\left(-\frac{1}{f}\left(B - \frac{1}{f}\mathbf{V}\mathbf{V}^T\right)^{-1}\mathbf{V}\right) , \quad \text{(A.11)}$$

and finally we found the center of the ellipse using the values from the inverse of the ellipse matrix as

$$u = \frac{\mathbf{V}}{f} . \quad \text{(A.12)}$$

Now we have the center of the ellipse, and we have the ability to transform the center of the ellipse to the origin. Inverse of such and ellipse is

$$C_{ce}^{-1} = M^{-1} C_e^{-1} M^{-T} \,, \tag{A.13}$$

where M^{-1} is

$$M^{-1} = \begin{bmatrix} I & -\mathbf{u} \\ \mathbf{0} & 1 \end{bmatrix} \,, \tag{A.14}$$

which gives

$$C_{ce}^{-1} = \begin{bmatrix} B - \mathbf{u}\mathbf{V}^T - \mathbf{u}^T\mathbf{V} + \mathbf{u}\mathbf{u}^T f & \mathbf{V} - f\mathbf{u} \\ \mathbf{V}^T - f\mathbf{u}^T & f \end{bmatrix} \,, \tag{A.15}$$

which simplifies into using (A.12)

$$C_{ce}^{-1} = \begin{bmatrix} B - \dfrac{\mathbf{V}\mathbf{V}^T}{f} & 0 \\ 0 & f \end{bmatrix} \,. \tag{A.16}$$

Using the resulting inverse ellipse we could find the width of the ellipse which has its center point on the origin. Let us define this inverse ellipse as

$$C_{ce}^{-1} = \begin{bmatrix} a & b & 0 \\ b & c & 0 \\ 0 & 0 & f \end{bmatrix} \,, \tag{A.17}$$

taking inverse of this ellipse will give us the ellipse function which we have translated to the origin,

$$C_{ce} = \begin{bmatrix} \dfrac{1}{ac - b^2} \begin{bmatrix} c & -b \\ -b & a \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \dfrac{1}{f} \end{bmatrix} \,. \tag{A.18}$$

Using (3.11) we can find the width as

$$w = 2\sqrt{\frac{-a}{f}} \,, \tag{A.19}$$

To be able to find width of the ellipse intuitively we need to translate the center of the ellipse to the origin. But we do not need to translate the inverse

of the ellipse, we can use the inverse of the ellipse directly to find width of the ellipse. Let us define inverse of the ellipse as

$$C_e^{-1} = \begin{bmatrix} A & B & D \\ B & C & E \\ D & E & F \end{bmatrix} , \tag{A.20}$$

then from (A.17) and (A.19), we can find the width of the ellipse by

$$w = 2\sqrt{\frac{-A}{F} + \frac{D^2}{F^2}} , \tag{A.21}$$

A.2 Jacobian of The Sensor Model

$\frac{\partial T}{\partial \mathbf{t}_r^W}$ is the jacobian of T with respect to the robot position. In (3.1) lets call first matrix M_R and second one M_T, then utilizing the chain rule for derivatives of matrices

$$\frac{\partial T}{\partial \mathbf{t}_r^W} = \left(M_T^T \otimes I_4\right) \frac{\partial M_R}{\partial \mathbf{t}_r^W} , \tag{A.22}$$

where $\frac{\partial M_R}{\partial \mathbf{t}_r^W}$ could be found as

$$\frac{\partial M_R}{\partial \mathbf{t}_r^W} := \begin{bmatrix} & \mathbf{0}_{12x3} & \\ -r_{11} & -r_{21} & -r_{31} \\ -r_{12} & -r_{22} & -r_{32} \\ -r_{13} & -r_{23} & -r_{33} \\ 0 & 0 & 0 \end{bmatrix} , \tag{A.23}$$

in which the elements of R_r^{WC} is used. R_r^{WC} is defined as

$$R_r^{WC} := \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{A.24}$$

$\frac{\partial T}{\partial R_r^{WC}}$ is the jacobian of T with respect to the robot orientation

$$\frac{\partial T}{\partial R_r^{WC}} = \left(M_T^T \otimes I_4\right) \frac{\partial M_R}{\partial R_r^{WC}} , \tag{A.25}$$

where $\frac{\partial M_R}{\partial R_r^{WC}}$ could be found as

$$\frac{\partial M_R}{\partial R_r^{WC}} := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -t_{r1} & -t_{r2} & -t_{r3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -t_{r1} & -t_{r2} & -t_{r3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -t_{r1} & -t_{r2} & -t_{r3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \text{(A.26)}$$

in which the elements of $\mathbf{t}_r^W$ is used. $\mathbf{t}_r^W$ is defined as

$$\mathbf{t}_r^W := \begin{bmatrix} t_{r1} & t_{r2} & t_{r3} \end{bmatrix}^T . \quad \text{(A.27)}$$

Since our implementation uses the quaternion instead of the orientation matrix, we need $\frac{\partial R_r^{WC}}{\partial q_r^{WC}}$ which is the jacobian of the orientation matrix with respect to the quaternion. It is defined as

$$\frac{\partial R_r^{WC}}{\partial q_r^{WC}} := \begin{bmatrix} r & x & -y & -z \\ z & y & x & r \\ -y & z & -r & x \\ -z & y & x & -r \\ r & -x & y & -z \\ x & r & z & y \\ y & z & r & x \\ -x & -r & z & y \\ r & -x & -y & z \end{bmatrix}, \quad \text{(A.28)}$$

in which the elements of a quaternion is used. A quaternion is defined as

$$q := \begin{bmatrix} r & x & y & z \end{bmatrix}^T . \tag{A.29}$$

$\frac{\partial T}{\partial \mathbf{t}_t}$ is the jacobian of T with respect to the tree position

$$\frac{\partial T}{\partial \mathbf{t}_t} = (I_4 \otimes M_R) \frac{\partial M_T}{\partial \mathbf{t}_t} , \tag{A.30}$$

where $\frac{\partial M_T}{\partial \mathbf{t}_t}$ could be found as

$$\frac{\partial M_T}{\partial \mathbf{t}_t} := \begin{bmatrix} & \mathbf{0}_{12x3} & \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} . \tag{A.31}$$

$\frac{\partial H}{\partial T}$ can be found as

$$\frac{\partial H}{\partial T} = (I_3 \otimes A_1) \left(A_2^T \otimes I_4 \right) , \tag{A.32}$$

where A_1 and A_2 are defined in (3.4).

$\frac{\partial iCe}{\partial H}$ can be found by taking partial derivative of (A.2)

$$\frac{\partial iCe}{\partial H} = \left(\left(C_c^{-1} H^T \right) \otimes I_3 \right) + (I_3 \otimes H) \left(I_3 \otimes C_c^{-1} \right) T_{33} , \tag{A.33}$$

where T_{33} is

$$T_{33} := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{A.34}$$

$\frac{\partial iCe}{\partial r_t}$ can be found as

$$\frac{\partial iCe}{\partial r_t} = (I_3 \otimes H)(H \otimes I_3)\frac{\partial C_c^{-1}}{\partial r_t}\,, \tag{A.35}$$

where $\frac{\partial C_c^{-1}}{\partial r_t}$ is

$$\frac{\partial C_c^{-1}}{\partial r_t} := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2}{r^3} \end{bmatrix}^T\,. \tag{A.36}$$

$\frac{\partial h}{\partial iCe}$ can be found as

$$\frac{\partial h}{\partial iCe} := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{F} & 0 & -\frac{D}{F^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{F} & -\frac{E}{F^2} \\ -\frac{1}{F}\sqrt{\frac{F^2}{D^2-AF}} & 0 & 0 & 0 & 0 & 0 & \frac{2D}{F^2}\sqrt{\frac{F^2}{D^2-AF}} & 0 & \frac{AF-2D^2}{F^3}\sqrt{\frac{F^2}{D^2-AF}} \end{bmatrix}\,, \tag{A.37}$$

in which the elements of the inverse of an ellipse is used. Inverse of an ellipse is defined as

$$C_e^{-1} = \begin{bmatrix} A & B & C \\ B & C & E \\ D & E & F \end{bmatrix}\,, \tag{A.38}$$

Bibliography

[1] S. Ahn, M. Choi, J. Choi, and W. K. Chung. Data association using visual object recognition for EKF-SLAM in home environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Vols 1-12*, pages 2588–2594, 2006.

[2] W. Ali, F. Georgsson, and T. Hellstrom. Visual tree detection for autonomous navigation in forest environment. In *IEEE Intelligent Vehicles Symposium, VOLS 1-3*, pages 1144–1149, 2008.

[3] H. J. Andersen, T. L. Dideriksen, C. Madsen, and M. B. Holte. Investigating the potential combination of GPS and scale invariant visual landmarks for robust outdoor cross-country navigation. In *VISAPP 2006: Proceedings of the First International Conference on Computer Vision Theory and Applications, Vol 2*, pages 349–356, 2006.

[4] D. C. Asmar, S. M. Abdallah, and J. S. Zelek. Vision SLAM maps: Towards richer content. In Liu, D and Wang, L and Tan, KC, editor, *Design and Control of Intelligent Robotic Systems*, volume 177, pages 303–329. Springer-Verlag Berlin, 2009.

[5] D. C. Asmar, J. S. Zelek, and S. M. Abdallah. Seeing the trees before the forest. *Computer and Robot Vision, Canadian Conference*, 0:587–593, 2005.

[6] D. C. Asmar, J. S. Zelek, and S. M. Abdallah. Tree trunks as landmarks for outdoor vision SLAM. *Computer Vision and Pattern Recognition Workshop*, 0:196, 2006.

[7] M. Buehler, R. Playter, and M. Raibert. Robots step outside. In *Proc. of the Int. Symp. on Adaptive Motion of Animals and Machines*, pages 1–4, September 2005.

[8] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth to depth conversion for monocular SLAM. In *IEEE International Conference on Robotics and Automation*, volume 1-10, pages 2778–2783, 2007.

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

[10] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point RANSAC for extended kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, Sep-Oct 2010.

[11] A. J. Davison. Real-time simultaneous localization and mapping with a single camera. *International Conference on Computer Vision*, 2003.

[12] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, JUN 2007.

[13] E. Eade. *Monocular Simultaneous Localization and Mapping*. PhD thesis, Cambridge University, 2008.

[14] P. Elinas, R. Sim, and J. J. Little. sigma SLAM: Stereo vision SLAM using the Rao-Blackwellised Particle Filter and a novel mixture proposal distribution. In *2006 IEEEInternational Conference on Robotics and Automation*, volume 1-10, pages 1564–1570, 2006.

[15] J. C. Gower and G. B. Dijksterhuis. *Procrustes Problems*. Oxford University Press, USA, 2004.

[16] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[17] S. Hirtle. Landmarks for navigation in human and robots. In M. Jefferies and W.-K. Yeap, editors, *Robotics and Cognitive Approaches to Spatial Mapping*, volume 38, pages 203–214. Springer Berlin / Heidelberg, 2008.

[18] A. Huertas, L. Matthies, and A. Rankin. Stereo-based tree traversability analysis for autonomous off-road navigation. *Applications of Computer Vision and the IEEE Workshop on Motion and Video Computing*, 1:210–217, 2005.

[19] J. Kim and S. Sukkarieh. Real-time implementation of airborne inertial-SLAM. *Robotics and Autonomous Systems*, 55(1):62 – 71, 2007.

[20] T. Krajnik, J. Faigl, V. Vonasek, K. Kosnar, M. Kulich, and L. Preucil. Simple yet stable bearing-only navigation. *Journal of Field Robotics*, 27(5):511–533, SEP-OCT 2010.

[21] R. Madhavan and H. Durrant-Whyte. Natural landmark-based autonomous vehicle navigation. *Robotics and Autonomous Systems*, 46(2):79–95, FEB 29 2004.

[22] J. Miro, G. Dissanayake, and W. Zhou. Vision-based SLAM using natural features in indoor environments. In *Intelligent Sensors, Sensor Networks & Information Processing Conference*, pages 151–156, 2005.

[23] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *IEEE Robotics and Autonomous Systems*, volume 1-3, pages 1985–1991, 2003.

[24] R. Murrieta-Cid, C. Parra, and M. Devy. Visual navigation in natural environments: From range and color data to a landmark-based model. *Autonomous Robots*, 13(2):143–168, SEP 2002.

[25] X.-D. Nguyen, B.-J. You, and S.-R. Oh. A simple landmark model for vision-based simultaneous localization and mapping. In *SICE-ICASE International Joint Conference*, volume 1-13, pages 2659–2664, 2006.

[26] J. Nieto, T. Bailey, and E. Nebot. Recursive scan-matching SLAM. *Robotics and Autonomous Systems*, 55(1):39 – 49, 2007.

[27] P. Nunez, R. Vazquez, J. C. del Toro, A. Bandera, and F. Sandoval. A curvature based method to extract natural landmarks for mobile robot navigation. In Urena, JU and Dominguez, JJG, editor, *IEEE International Symposium on Signal Processing*, pages 759–764, 2007.

[28] P. Nunez, R. Vazquez-Martin, J. C. del Toro, A. Bandera, and F. Sandoval. Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation. *Robotics and Autonomous Systems*, 56(3):247–264, MAR 31 2008.

[29] M. Rous, H. Lupschen, and K. Kraiss. Vision-based indoor scene analysis for natural landmark detection. In *IEEE International Conference on Robotics and Automation*, volume 1-4, pages 4642–4647, 2005.

[30] P. Saeedi, P. Lawrence, and D. Lowe. Vision-based 3-D trajectory tracking for unknown environments. *IEEE Transaction on Robotics*, 22(1):119–136, FEB 2006.

[31] U. Saranli, M. Buehler, and D. E. Koditschek. RHex: A simple and highly mobile robot. *International Journal of Robotics Research*, 20(7):616–631, July 2001.

[32] D. Schleicher, L. M. Bergasa, M. Ocana, R. Barea, and E. Lopez. Real-time hierarchical stereo visual slam in large-scale environments. *Robotics and Autonomous Systems*, 58(8):991–1002, AUG 31 2010.

[33] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette. A mapping and localization framework for scalable appearance-based navigation. *Computer Vision and Image Understanding*, 113(2):172–187, FEB 2009.

[34] J. Sola. Consistency of the monocular EKF-SLAM algorithm for three different landmark parametrizations. In Rakotondrabe, M and Ivan, IA, editor, *IEEE International Conference on Robotics and Automation*, pages 3513–3518, 2010.

[35] C.-H. Teng, Y. sheng Chen, and W. hsing Hsu. Tree segmentation from an image. *Machine VIsion Applications*, 2005.

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

[37] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, SEP 2006.

[38] S. Tully, H. Moon, G. Kantor, and H. Choset. Iterated filters for bearing-only SLAM. In *IEEE International Conference on Robotics and Automation*, volume 1-9, pages 1442–1448, 2008.

[39] T. Vidal-Calleja, C. Berger, J. Sola, and S. Lacroix. Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robotics and Autonomous Systems*, 2011.

[40] M. Wang, H. Tamimi, and A. Zell. Robot navigation using biosonar for natural landmark tracking. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 3–7, 2005.

[41] J. Weingarten, G. Lopes, M. Buehler, R. Groff, and D. Koditschek. Automated gait adaptation for legged robots. In *IEEE International Conference on Robotics and Automation*, volume 1- 5, pages 2153–2158, 2004.

[42] E. Yeh and D. J. Kriegman. Toward selecting and recognizing natural landmarks. In *International Conference on Intelligent Robots and Systems*, volume 1, pages 47–53, 1995.

[43] T. Yildiz. Detection of tree trunks as visual landmarks in outdoor environments. Master's thesis, Bilkent University, 2010.