ABSTRACT

Title of dissertation: CONCURRENT LOCALIZATION AND MAPPING WITH SONAR SENSORS AND CONSIDERATION OF VEHICLE MOTIONS

Hesham Ismail, 2016

Dissertation directed by: Professor Balakumar Balachandran
Department of Mechanical Engineering

Simultaneous Localization and Mapping (SLAM) is a procedure used to determine the location of a mobile vehicle in an unknown environment, while constructing a map of the unknown environment at the same time. Mobile platforms, which make use of SLAM algorithms, have industrial applications in autonomous maintenance, such as the inspection of flaws and defects in oil pipelines and storage tanks. A typical SLAM consists of four main components, namely, experimental setup (data gathering), vehicle pose estimation, feature extraction, and filtering. Feature extraction is the process of realizing significant features from the unknown environment such as corners, edges, walls, and interior features. In this work, an original feature extraction algorithm specific to distance measurements obtained through SONAR sensor data is presented. This algorithm has been constructed by combining the SONAR Salient Feature Extraction Algorithm and the Triangulation Hough Based Fusion with point-in-polygon detection. The reconstructed maps obtained through simulations and experimental data with the fusion algorithm are compared to the

maps obtained with existing feature extraction algorithms. Based on the results obtained, it is suggested that the proposed algorithm can be employed as an option for data obtained from SONAR sensors in environment, where other forms of sensing are not viable. The algorithm fusion for feature extraction requires the vehicle pose estimation as an input, which is obtained from a vehicle pose estimation model. For the vehicle pose estimation, the author uses sensor integration to estimate the pose of the mobile vehicle. Different combinations of these sensors are studied (e.g., encoder, gyroscope, or encoder and gyroscope). The different sensor fusion techniques for the pose estimation are experimentally studied and compared. The vehicle pose estimation model, which produces the least amount of error, is used to generate inputs for the feature extraction algorithm fusion. In the experimental studies, two different environmental configurations are used, one without interior features and another one with two interior features. Numerical and experimental findings are discussed. Finally, the SLAM algorithm is implemented along with the algorithms for feature extraction and vehicle pose estimation. Three different cases are experimentally studied, with the floor of the environment intentionally altered to induce slipping. Results obtained for implementations with and without SLAM are compared and discussed. The present work represents a step towards the realization of autonomous inspection platforms for performing concurrent localization and mapping in harsh environments.

CONCURRENT LOCALIZATION AND MAPPING WITH SONAR SENSORS AND CONSIDERATION OF VEHICLE MOTIONS

by

Hesham Ismail

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:
Professor Balakumar Balachandran, Chair and Advisor
Professor Amr Baz, Department of Mechanical Engineering
Associate Professor Nikhil Chopra, Department of Mechanical Engineering
Assistant Professor Jin-Oh Hahn, Department of Mechanical Engineering
Professor Roberto Celi, Department of Aerospace Engineering (Dean's Representative)

Acknowledgments

I would like to thank Professor Balakumar Balachandran for his time and support throughout this project. Also, I would like to thank my committee members for their positive feedback. I want to thank my lab mates, especially Dr. Nick Vlajic, Dr. Edmon Perkins, Dr. Tim Fitzgerald, and Ms. Rubyca Jaai for their valuable discussion and help throughout this project. I would like to thank my friends, Dr. Abdallah Al Tamimi and Dr. Saeed Tamar for the positive support and encouragement through the years. Also, I am grateful to my sponsor, Abu Dhabi National Oil Company (ADNOC), for their fellowship and support to this project. Finally, a warm and especial thanks to my family for their love and support through this long journey of my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of Simultaneous Localization and Mapping

1.1.1 Definition of SLAM and Problem Formulation

Simultaneous localization and mapping (SLAM) is a technique used to determine the location of a mobile vehicle in an unknown environment, while constructing a map of the unknown environment at the same time (Aulinas, Petillot, Salvi, and Lladó, 2008). A primary goal of SLAM is to have a truly autonomous mobile vehicle. The manner in which a SLAM algorithm is used, a mobile vehicle mounted with sensors is used to identify landmarks (or features) in the environment such as points, lines and corners as well as determine its position simultaneously. Broadly speaking, there are two kinds of SLAM approaches: i) deterministic SLAM and ii) probabilistic SLAM. In deterministic SLAM, errors are bounded because measurements from the sensors are assumed to be exact and there are no approximations. However, in probabilistic SLAM there is an associated level of uncertainty with each landmark. The probability density function in probabilistic SLAM is typically assumed to be Gaussian with a known covariance. Examples of probabilistic SLAM are Smoothing and Mapping(SAM), Extended Kalman Filter (EKF), and so on. Overall, probabilistic SLAM is more practical for real cases than deterministic SLAM (Joly and Rives,

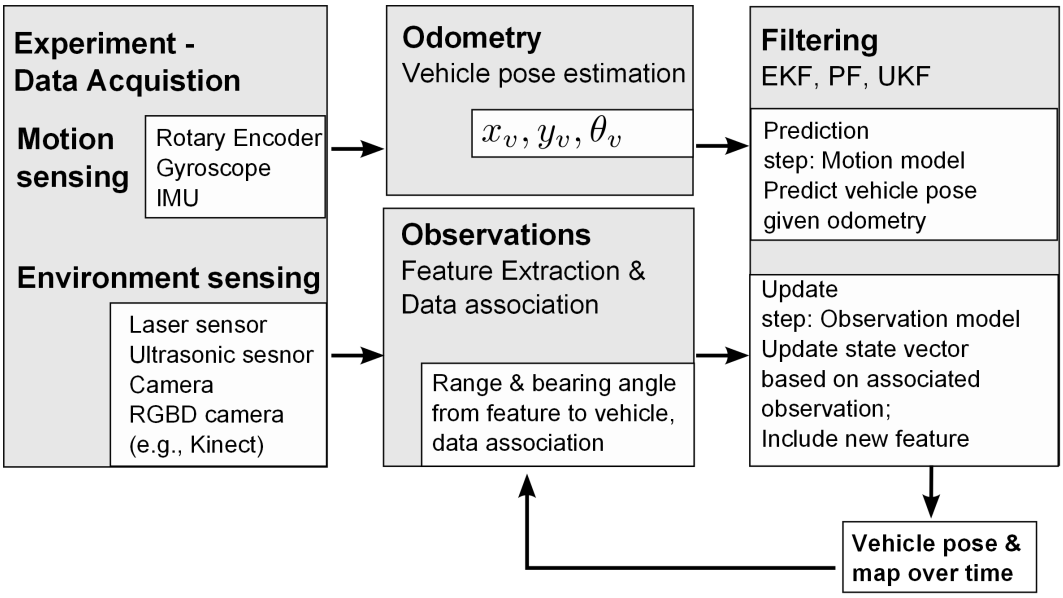


Figure 1.1: Main building blocks needed to implement SLAM.

2008). SLAM can be implemented in a variety of environments, such as indoor environments (e.g., offices, laboratories), outdoor settings (e.g., parks, sidewalks, and roads), underwater applications (e.g., water tanks, pipelines, and ocean), and industrial applications (e.g., inspections in unsafe or unknown environments). Motivation for the current study originates from the need for autonomous inspection of oil tanks and pipelines. A review paper by Durrant-Whyte and Bailey (2006) contains details about SLAM, such as its origins and open research areas (i.e., computational efficiency, data association, and loop closure).

To implement SLAM, four main parts are needed, as shown in Figure 1.1. First, an experimental setup is required with two different types of sensor: a motion sensor (e.g., rotary encoder, gyroscope,and internal Inertial Measurement Unit (IMU)) and an environment sensor (e.g., laser, ultrasonic, camera, and RGBD camera such as Kinect). Second, data gathered from the motion sensor are used to

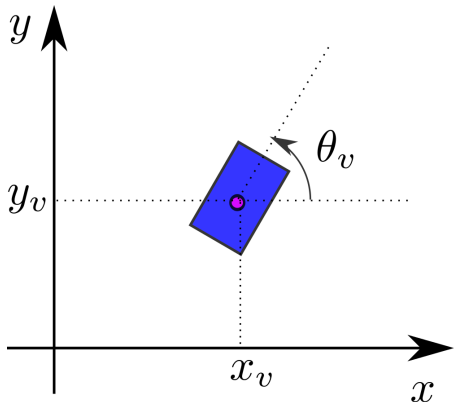


Figure 1.2: Vehicle pose in global coordinate system x and y.

estimate the vehicle pose, (x_v, y_v, θ_v), where θ_v is the heading angle and (x_v, y_v) is the mobile vehicle position in Cartesian coordinates, as shown in Figure 1.2. Third, information from the environmental sensor is used to determine the location of landmarks (or features) in the environment as well as the location of the mobile vehicle. Data association is a process used to determine if a feature has already been detected or if it is a new feature that needs to be accounted for. Data association works in the following way: if a mobile vehicle detects a feature at time k and visits the same feature at time k+10, the purpose of data association is to make sure that the mobile vehicle recognizes that these two features originated from the same object. Finally, different kinds of filters can be implemented. In practice, measurements from the vehicle contain noise (or uncertainty). Therefore, the pose of the vehicle and the landmarks are estimated by using different algorithms. The three most common filtering techniques are Kalman filter (KF), particle filter (PF), and graph-based filter. In this dissertation work, the Extended Kalman filter (EKF), which is a nonlinear version of the KF, is to be implemented in SLAM.

1.1.2 Application of Kalman Filter to SLAM

The Kalman filter (KF) is a linear optimal estimator, and it is used in the SLAM algorithm to estimate landmarks and vehicle pose. It is recursive, which means new measurements can be processed as soon as they arrive. The KF formulation is based on the assumption that the noise is Gaussian, and it tries to minimize the mean square error of the estimate parameters (Terejanu, 2008; Welch and Bishop, 2006). The process (motion) model used in KF is described in equation (1.1) and the observation model used in KF is described in equation (1.2). The variables used in equations (1.1) and (1.2) are described in Table 1.1.

$$\check{\mathbb{X}}(k+1) = A\mathbb{X}(k) + B\mathbf{u}(k+1) + \boldsymbol{\varepsilon}(k+1) \tag{1.1}$$

$$\check{Z}(k+1) = C\hat{\mathbb{X}}(k+1) + \delta(k+1) \tag{1.2}$$

The KF algorithm is presented in Algorithm 1. The KF algorithm is divided into two steps: prediction and correction. In this algorithm, steps 2 and 3 represent the prediction step, whereas steps 4, 5, and 6 represent the correction step. During the first iteration in the KF algorithm, $\mathbb{X}(k)$ and $\mathbb{P}(k)$ are estimated. In the prediction step, the state and covariance are estimated forward from k to $k+1$ based on the process model. Next, the correction step starts by computing the Kalman gain $W(k+1)$ followed by obtaining an observation (measurement) vector $Z(k+1)$ and using $W(k+1)$ and $\hat{Z}(k+1)$ to calculate *a posteriori* state estimate, $\mathbb{X}(k+1)$, and *a posteriori* covariance estimate, $\mathbb{P}(k+1)$.

If the process model or the measurement model is non-linear, the Extended

Table 1.1: Nomenclature for KF and EKF algorithms

Variable	Dimension	Description
$\mathbb{X}(k+1)$	$n \times 1$	Posteriori estimate of the state vector at step $k+1$
$\check{\mathbb{X}}(k+1)$	$n \times 1$	Actual state vector at step $k+1$
$\hat{\mathbb{X}}(k+1)$	$n \times 1$	Mean of the posterior (next) state at step $k+1$
$\hat{\mathbb{P}}(k+1)$	$n \times n$	Projected (estimate) error covariance at step $k+1$
$\mathbb{P}(k+1)$	$n \times n$	Error covariance updated at step $k+1$
$W(k+1)$	$n \times l$	Kalman gain matrix at step $k+1$
A	$n \times n$	Linear process matrix
B	$n \times m$	Linear control matrix
$\mathbf{u}(k+1)$	$m \times 1$	Control vector at step $k+1$
$\boldsymbol{\varepsilon}(k+1)$	$n \times 1$	Process noise vector at step $k+1$
$Z(k+1)$	$l \times 1$	Measurement vector at step $k+1$
$\check{Z}(k+1)$	$l \times 1$	True measurement vector at step $k+1$
$\hat{Z}(k+1)$	$l \times 1$	Observation model at step $k+1$
C	$l \times n$	Linear observation matrix
$\delta(k+1)$	$l \times 1$	Measurement noise vector at step $k+1$
$Q(k+1)$	$n \times n$	Process noise covariance matrix at step $k+1$
R	$m \times m$	Measurement noise covariance matrix
$f(\cdot)$	$n \times 1$	Process nonlinear vector function
$h(\cdot)$	$m \times 1$	Observation nonlinear vector function
$J_{vv}(k+1)$	$n \times n$	Jacobian matrix of $f(\cdot)$ with respect to $\mathbb{X}$ at step $k+1$
$J_{vu}(k+1)$	$n \times n$	Jacobian matrix of $f(\cdot)$ with respect to $\mathbf{u}$ at step $k+1$
$H(k+1)$	$l \times n$	Jacobian matrix of $h(\cdot)$ with respect to $\mathbb{X}$ at step $k+1$
k	1 x 1	Time step

Algorithm 1 Kalman Filter Algorithm developed by Kalman (1960).

Kalman filter $(\mathbb{X}(k), \mathbb{P}(k+1), \mathbf{u}(k+1), Z(k+1))$**:**

$\hat{\mathbb{X}}(k+1) = A\mathbb{X}(k) + B\mathbf{u}(k+1)$

$\hat{\mathbb{P}}(k+1) = A\mathbb{P}(k) A^T + Q(k+1)$

$W(k+1) = \hat{\mathbb{P}}(k+1) C^T (C\hat{\mathbb{P}}(k+1) C^T + R)^{-1}$

$\mathbb{X}(k+1) = \hat{\mathbb{X}}(k+1) + W(k+1) \left(Z(k+1) - C\hat{\mathbb{X}}(k+1) \right)$

$\mathbb{P}(k+1) = (I - W(k+1) C) \hat{\mathbb{P}}(k+1)$

return $\mathbb{X}(k+1), \mathbb{P}(k+1)$

Kalman filter (EKF) is used. EKF uses Taylor series expansion to linearize the non-linear system. The process (motion) model used in EKF is described in equation (1.3) and the observation model used in EKF is described in equation (1.4). The variables of equations (1.3) and (1.4) are described in Table 1.1.

$$\check{\mathbb{X}}(k+1) = f(\mathbf{u}(k+1), \mathbb{X}(k)) + \boldsymbol{\varepsilon}(k+1) \tag{1.3}$$

$$\check{Z}(k+1) = h\left(\hat{\mathbb{X}}(k+1)\right) + \delta(k+1) \tag{1.4}$$

The EKF algorithm is provided in Algorithm 2. The steps in this algorithm are similar to the steps explained previously for the KF algorithm. The only difference arises from replacing A, B and Ct matrix by $f(\cdot)$ and $h(\cdot)$ functions and their Jacobians, J_{vv}, J_{vu} and H. For more information on the KF and EKF algorithms, the reader is referred to Thrun, Burgard, and Fox (2005). There are other types of filtering used to solve the SLAM problem, for example, the particle filter and the graph-based filter. Related information can also be found in the book by Thrun *et al.* (2005).

Algorithm 2 Extended Kalman Filter Algorithm

1: **Extended Kalman filter** $(\mathbb{X}(k), \mathbb{P}(k), \mathbf{u}(k+1), Z(k+1))$**:**

2: $\hat{\mathbb{X}}(k+1) = f(\mathbf{u}(k+1), \mathbb{X}(k))$

3: $\hat{\mathbb{P}}(k+1) = J_{vv}(k+1)\,\mathbb{P}(k)\,J_{vv}^{T}(k+1) + J_{vu}(k+1)\,Q(k+1)\,J_{vu}^{T}(k+1)$

4: $W(k+1) = \hat{\mathbb{P}}(k+1)\,H^{T}(k+1)\,(H(k+1)\,\hat{\mathbb{P}}(k+1)\,H^{T}(k+1) + R)^{-1}$

5: $\mathbb{X}(k+1) = \hat{\mathbb{X}}(k+1) + W(k+1)\left(Z(k+1) - h\left(\hat{\mathbb{X}}(k+1)\right)\right)$

6: $\mathbb{P}(k+1) = (I - W(k+1)\,H(k+1))\,\hat{\mathbb{P}}(k+1)$

7: return $\mathbb{X}(k+1), \mathbb{P}(k+1)$

1.1.3 Literature Review on SLAM

P.Cheesman, J.Crowley, and H.D. Whyte were the first to investigate the SLAM problem in a probabilistic framework; see Durrant-Whyte and Bailey (2006). The three most common filtering algorithms used in SLAM are the Kalman Filter (KF), a particle filter (PF), and an Expectation Maximization based method (EM). KF was used for linear systems and assumes a Gaussian distribution for the noise. Many researchers have expanded upon the KF method by developing the Extended Kalman Filter (EKF) to handle nonlinear systems, but EKF has its limitations. The computational demands of EKF increase as the number of features in the environment increase, but it is seen to be tractable for moderately sized environments (Paz and Neira, 2006). Rearchers have developed a better solution in order to handle large maps that use Information Filters (IF) and Extended Information Filters (EIF) (Thrun and Liu, 2005). IF and EIF take advantage of the information matrix, defined as the inverse of the covariance matrix, and the fact that it is bounded. Therefore, estimation can be performed in constant time. Also, the canonical SLAM distribution is sparse (Walter, Eustice, and Leonard, 2007). Another problem is that the elements in the information matrix have relatively small values in a feature-based SLAM. To solve this problem, researchers break down these weak links, and Sparse Extended Information Filters (SEIF) are used. The problem with breaking these weak links is that some of the correlations between the landmark and robot pose is lost. Another significant research contribution has been the Exactly Sparse Extended Information Filter (ESEIF) for feature-based SLAM

and Exactly Sparse Delayed-State Filter (EDSF) for view-based SLAM. View-based SLAM uses historical robot poses instead of the features. The exact method removes the approximation assumption and instead uses an augmented strategy where every robot is linked to its previous or next pose. This allows the Information matrix to become sparse without any assumptions (Eustice, Singh, and Leonard, 2006; Walter *et al.*, 2007). In addition to single robot SLAM, multi-robot systems use SLAM algorithms, mostly with PF. Additionally, some researchers use SEIF. PF are mostly used for multi-robot systems due to the efficiency used in building small maps, its non-Gaussian approximation, and because it handles nonlinearities well (Lee and Lee, 2009; Thrun and Liu, 2005; Carlone, Ng, Du, Bona, and Indri, 2010). In using multi-robot SLAM, different challenges appear, such as coordination, data communication, and map merging.

1.2 Review of Feature Extraction using SONAR

Feature extraction, as explained previously, is an integral part of SLAM. In an environment, there can be two types of features, namely artificial features and geometric features. The artificial features are essentially well-placed beacons at known locations in the environment that will help the mobile vehicle localize itself with respect to these beacons. On the other hand, the geometric features are features that naturally exist in the environment and can be described in terms of geometric parameterization. In addition, geometric features can help a mobile vehicle reach an autonomous state, since it doesn't modify the environment. It uses features already

intrinsic to the environment (e.g., planes, corner, edges, cylinder, and so on) for localization and map building.

The gathered data from SONAR are distance, from SONAR to the landmark, and angle of the SONAR reading. The data from a SONAR scan can be analyzed by using a grid-based probabilistic model or a feature extraction model. In a grid-based probabilistic model, the environment is divided into a two-dimensional array denoted by cells. Each cell has a certainty value, which is updated every time a SONAR sensor gets a reading. Examples of grid-based probabilistic models include occupancy grids (Elfes, 1987), inference grids (Elfes, 1992), and vector field histograms (Borenstein and Koren, 1991). Feature extraction models transform SONAR data into information about the environment (e.g., planes, corners, edges, cylinders, etc.). Examples of feature-based models includes Triangulation-Based Fusion (TBF), Hough Transform (HT), Region of Constant Depth (RCD), Triangulation Hough Fusion (THF), and salient feature extraction. In the upcoming subsection, different feature extraction algorithms that are commonly used with SONAR data will be discussed.

1.2.1 Region of Constant Depth

Attaching a SONAR sensor to a servo motor allows for a full scan at a particular location in the environment. In a full scan, some of the consecutive readings have the same range value. This group of consecutive readings are known as Region of Constant Depth (RCD) (Leonard and Durrant-Whyte, 1991). As the mobile vehicle

Table 1.2: Variable used in Figure 1.3

Variable	Unit	Description
$\theta_v(k)$	rad	Orientation of the mobile vehicle at time k
$r(k)$	cm	Sonar sensor range reading at time k

moves from location k to location $k+1$, the unknown RCD are tracked. For example, RCDs which correspond to a plane will all be tangent to the plane as shown in Figure 1.3. In addition, RCDs which correspond to a corner will all intersect at a point, as shown in Figure 1.3. There are also RCDs for multiple reflections which can be distinguished from planes and corners, because they follow unpredictable trajectories.

1.2.2 Triangulation-Based Fusion

The TBF algorithm was first introduced by Wijk, Jensfelt, and Christensen (1998), and it is used to find the point features in unknown environments. Each SONAR data reading is represented by an arc, and the intersection of two SONAR arcs forms a candidate point feature (target) as shown in Figure 1.4.

To find the intersection point, (x_T, y_T), the solution for the set of equations (1.5) is computed.

$$\begin{aligned} (x_T - x_{s1})^2 + (y_T - y_{s1})^2 &= r_1^2 \\ (x_T - x_{s2})^2 + (y_T - y_{s2})^2 &= r_2^2 \end{aligned} \tag{1.5}$$

The solution for the set of equations (1.5) is given by:

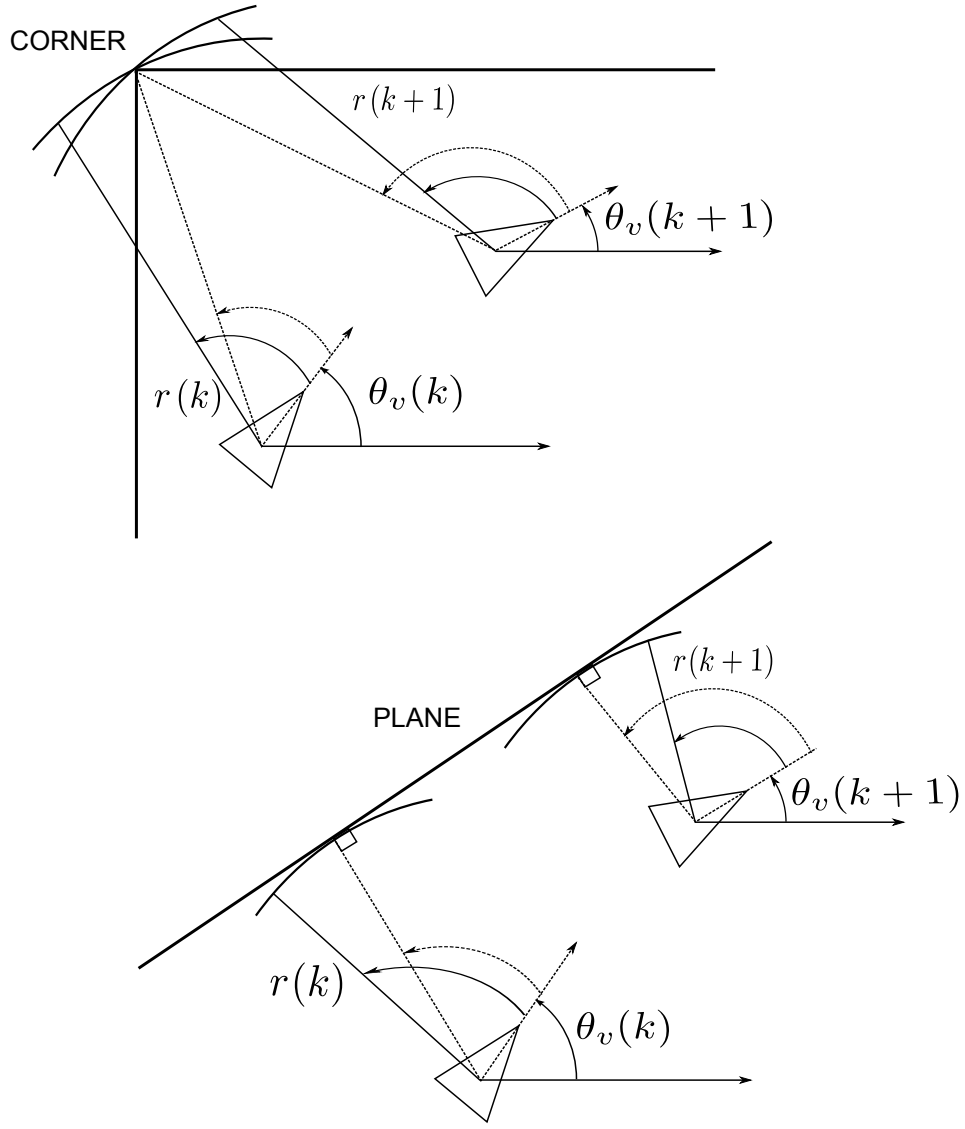


Figure 1.3: RCDs that correspond to corners and planes, as the mobile vehicle moves from location k to $k+1$.

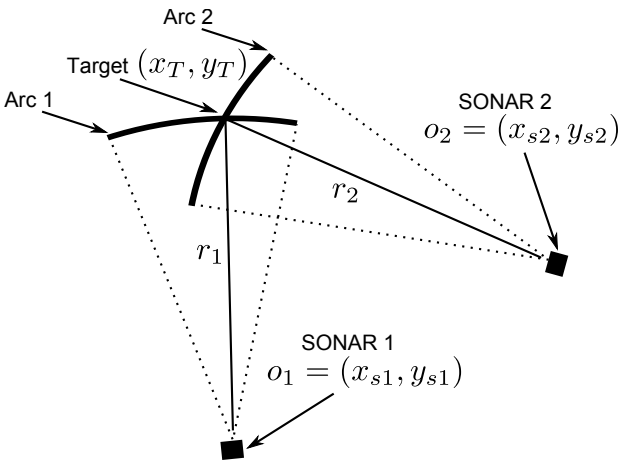


Figure 1.4: The arc intersections of SONAR data represents the point feature (target).

Table 1.3: Variables used in equation (1.5)

Variable	**Unit**	**Description**
(x_s, y_s)	(cm, cm)	SONAR sensor position
o		SONAR sensor position
(x_T, y_T)	(cm, cm)	Target or intersection point
r	cm	SONAR sensor range reading from the SONAR sensor to the target

$$y_T = y_{s1} + \frac{1}{c^2}\left(bd \pm |a|\sqrt{r_1^2 c^2 - d^2}\right)$$
$$x_T = x_{s1} \pm \sqrt{r_1^2 - (y_T - y_{s1})^2}$$

with,

$$\begin{cases} a &= x_{s1} - x_{s2} \\ b &= y_{s1} - y_{s2} \\ c &= \sqrt{a^2 + b^2} \\ d &= \frac{1}{2}\left(r_2^2 - r_1^2 - c^2\right) \end{cases}$$

The imaginary solutions are ignored, as well as those outside arc 1 and arc 2. The mean SONAR arc is 24^o. After finding the intersection, a point feature is

selected for $n_t \geq threshold\ value$. The threshold value will be varied and studied in the next chapter.

The TBF algorithm by Wijk *et al.* (1998), which is also used in the next chapter, is described in algorithm 3. The algorithm steps are explained as follows: step (1) loop over the recent scan of the SONAR sensor; step (2) set the counter for the intersection to zero; step (3) loop over all SONAR data gathered except the current point; step (4) compare the recent SONAR scan with all previous SONAR scans, and try to find an intersection point, (x_T,y_T), that satisfies equation 1.5, and also falls inside the SONAR arc range; step (5) for a successful step (4), update the target x-position recursively; step (6) for a successful step (4), update the target y-position recursively; step (7) increase the counter for intersection by 1; step (8) if the counter of the intersection is greater than a threshold value, the target position, (x_{est},y_{est}), will be saved as a point feature in step (9).

For further information about the TBF algorithm, the reader is referred to the aforementioned references.

1.2.3 Hough Transform

The Hough Transform is used to detect features such as edges, circles, ellipses, and other non-standard shapes. It was first introduced by Hough (1962) for recognizing different patterns. Here, this transform will be used to detect line features. An edge (line) can be described in many ways. In the Hough Transform, lines are expressed in polar coordinates, referred to as the "normal notion" for a line. This

Algorithm 3 TBF Algorithm developed by Wijk *et al.* (1998)

1: **for** $i = 1 \rightarrow m$ **do**

2: $n_t = 0$

3: **for** $j = (n-1) \rightarrow 1$ **do**

4: **if** (x_T, y_T) =find intersection $(x_{si}, y_{si}, r_i, \gamma_i, x_{sj}, y_{sj}, r_j, \gamma_j)$ **then**

5: $x_{est} = \dfrac{n_t x_{est} + x_t}{n_t + 1}$

6: $y_{est} = \dfrac{n_t y_{est} + y_t}{n_t + 1}$

7: $n_t + +$

8: **if** $n_t \geq threshold\ value$ **then**

9: $save\,(x_{est}, y_{est})$ as a point feature

representation is advantageous because it can properly account for vertical lines, which are not well-defined functions in Cartesian coordinates. A graphic with a depiction of a line and definitions is shown in Figure 1.5.

Thus using a normal description for a line, a vertical line can be detected. This is shown in Figure 1.5, and the equation is written in equation 1.6.

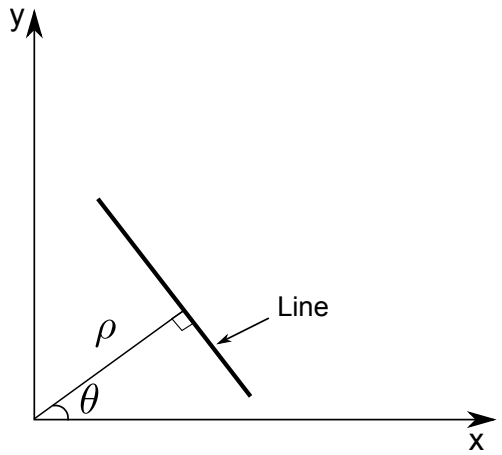


Figure 1.5: Line described by a normal line from the origin.

$$\rho = x \cos\theta + y \sin\theta \tag{1.6}$$

Here, ρ is the length of a vector that is normal to the line and passes through the origin, and θ is the orientation of ρ with respect to the x-axis. The algorithm used to perform the Hough Transform for straight lines is described in algorithm 4. For further information on the Hough Transform, the reader is referred to Hough (1962) and Tardós, Neira, Newman, and Leonard (2002).

Algorithm 4 Hough Transform algorithm for straight line

1: Load data ((x,y) values)

2: For each (x,y) value, vary θ from 0^o to 360^o and calculate ρ

3: Find maximum and minimum ρ, and maximum and minimum θ

4: Make bins for the ranges of ρ and θ

5: Place the value of ρ calculated from step 2 into the correct bin

6: Calculate the total number of objects (votes) in each bin

7: Get ρ and θ values of the top-voted bins and plot the straight lines

1.2.4 Triangulation & Hough Transform-Based Fusion

Baolong, Bo, Yongqing, Xuan, and Lei (2007) presents a novel sensor fusion scheme know as the THF algorithm. The THF algorithm is done in two steps. First, the TBF algorithm is done on raw SONAR data. Second, Cluster Inhibiting Hough

Transform (CIHT) is done on TBF data. For the SONAR setup, Baolong *et al.* (2007) used 24 fixed SONAR sensors configured in a ring, equiangularly spaced. The THF is robust to pedestrian movement near the mobile vehicle and is an effective technique for line and point detection. The THF algorithm was tested on simulation data but not on experimental data.

In the next chapter, the standard Hough Transform will be used instead of Cluster Inhibiting Hough Transform for the THF algorithm. The difference between the Hough Transform and the THF algorithm is that in Hough Transform, the raw SONAR data is used as input to the algorithm, whereas in the THF case, the processed TBF data is used as input for the Hough Transform. For more information about THF, the reader is referred to references Baolong *et al.* (2007).

1.2.5 Sonar Salient Feature

Sonar Salient Feature (Convex Saliency Circling) algorithm was introduced by Lee and Song (2010). It takes SONAR data and represents the point features as circle clouds. Figure 1.6 shows all possible hypothetical circles of two SONAR measurements at two different locations. The algorithm 5 helps select the appropriate hypothetical circle for the point features, such as corners and edges.

The footprint association model (FPA) is used to determine if two SONAR data points originate from the same target. The Convex Saliency Circling algorithm steps can be explained as follow: step (1), select the first SONAR reading from the total SONAR data set; step (2) check if the range reading, r_i, is within the

Table 1.4: Variables used in Figure 1.6 and Algorithm 5

Variable	**Unit**	**Description**
o_1	(cm, cm)	SONAR location 1
o_2	(cm, cm)	SONAR location 2
r_1	cm	SONAR range reading from location 1
r_2	cm	SONAR range reading from location 2
r_k	cm	SONAR range reading from location k
A	(cm, cm)	Center of the accepted hypothetical circle
ϕ_1	rad	Bearing from positive X-axis to the center A at location 1
ϕ_2	rad	Bearing from positive X-axis to the center A at location 2
$\gamma_{1,min}$	rad	Minimum orientation of the SONAR at location 1
$\gamma_{2,min}$	rad	Minimum orientation of the SONAR at location 2
d	cm	Distance between SONAR location 1 and SONAR location 2
β	rad	The angular uncertainty of SONAR sensor
c	(cm, cm, cm)	Hypothetical circle
m		Total number of SONAR reading from one location
n		Total number of locations which the mobile vehicle visited or from which it took scans
δ		Number used to determine the number of segments
D	cm	Distance from the SONAR to the center of the hypothetical circle
σ		Threshold value
r_a	cm	Radius of the accepted hypothetical circle
r_c	cm	Radius of the hypothetical circle
(x_c, y_c)	(cm, cm)	Center of the hypothetical circle
p_{min}	cm	Minimum threshold
p_{max}	cm	Maximum threshold

Algorithm 5 Convex Saliency Circling Algorithm developed by Lee and Song (2010)

1: **for** $i = 1 \rightarrow m \times n$ **do**

2: **if** $r_{min} < r_i < r_{max}$ **then**

3: **for** $j = i + 1 \rightarrow m \times n$ **do**

4: **if** $r_{min} < r_j < r_{max}$ **then**

5: **for** $\phi_i = \gamma_{i,min} \rightarrow \gamma_{i,min} + \beta,\ \phi_i = \phi_i + \delta$ **do**

6: FPA model { In: $(o_i, r_i, \gamma_{i,min}), (o_j, r_j, \gamma_{j,min}), \phi_i$; Out: (x_c, y_c, r_c)}

7: **if** $p_{min} < r_c < p_{max}$ **then**

8: **for** $k = j + 1 \rightarrow m \times n$ **do**

9: **if** $|D - r_c - r_k| < \sigma$ **then**

10: $C = [C; x_c\ y_c\ r_c]$

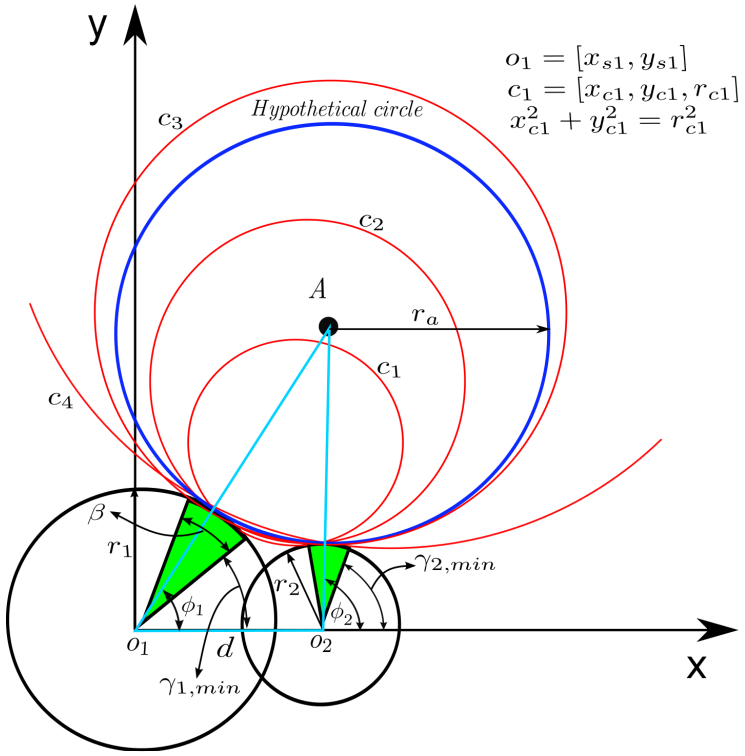


Figure 1.6: Possible hypothetical circles of two SONAR readings from two different locations.

acceptable range (r_{min}, r_{max}); step (3) select the second SONAR reading from the total SONAR data set; step (4) check if the SONAR range reading r_j is within the acceptable range (r_{min}, r_{max}); step (5) divide the SONAR arc into $\frac{\beta}{\delta} + 1$ (for example, for $\beta = 30^o$ and $\delta = 15^o$, corresponding to 2 segments); step (6) the input to the FPA model is information from SONAR taken at locations 1 and 2, and the output is hypothetical circles with center (x_c, y_c) or A and radius r_c or r_a; step (7) check if the radius of the hypothetical circle is within the threshold value; step (8) select the third SONAR data; step (9) if $|D - r_c - r_k| < \sigma$, do the next step; step (10) if all of the above conditions are satisfied, save the value of the hypothetical circle (point feature) in storage C.

1.2.6 Literature Review on Feature Extraction

A literature review of commonly used feature extraction algorithms using data from SONAR sensors is presented here. Wijk *et al.* (1998) developed the Triangulation-Based Fusion (TBF) that is used to extract point features from raw SONAR data. Wijk *et al.* (1998) demonstrated their algorithm on a mobile vehicle with 16 SONAR sensors. Wijk and Christensen (2000a) concluded that the TBF algorithm is robust in indoor environments, which include many natural landmarks. However, the TBF algorithm does not work well in corridors, because corridors are quite empty of natural landmarks, which causes the mobile vehicle to get lost because of mismatched (ghost) landmarks. This algorithm is implemented and compared to other feature extraction algorithms in Section 2.

Choi, Ahn, and Chung (2005) continued on the work of Wijk *et al.* (1998) and improved the TBF algorithm by adding the following: i) stable intersections, ii) efficient sliding window update, and iii) removal of false features on the wall. Also, they added a simple and novel line feature detection scheme. In order to detect line features, the reading from 3 adjacent SONAR sensors are gathered. If the three adjacent SONAR readings have similar range readings and the middle one has a minimum value among them, a line feature is detected. They also compared the obtained SONAR data with laser data. Laser data requires fewer processing steps in comparison to SONAR data. For the SONAR setup, they used a ring of 16 fixed SONAR sensors on the mobile vehicle.

Weiqin (2009) also extended on the work of Wijk *et al.* (1998) to include line feature extraction in addition to points features. To detect line features, Weiqin (2009) generated tangent lines for two arcs. Then, he choose one tangent line that passes through both tangent points as a line feature. He tested this algorithms on a mobile vehicle with 12 fixed SONAR sensors.

Leonard and Durrant-Whyte (1991) used SONAR sensors to achieve mobile vehicle localization. They used two different configuration of SONAR sensors. In the first configuration, the SONAR sensors were attached to a servo-motor and the servo-motor performs a full revolution to scan the environment at a particular location. The number of scans was equal to 612 measurements equiangularly spaced. In the second configuration, six SONAR sensors where mounted in a fixed configuration on top of the mobile platform. The servo-mounted SONAR gave more detailed information about the environment and data interpretation was easier compared to the fixed SONAR sensors. The disadvantage of the servo-mounted SONAR array is that it takes a longer time, to collect data in comparison to the ring of fixed SONAR sensors. Fixed SONAR sensor, have the advantage of allowing for "on-the-fly" position estimation. Another major observation of Leonard and Durrant-Whyte (1991) was that geometries such as planes, cylinders and corners were the strongest candidate for beacons. One of the limitations of the work was that an *a priori* map of the environment was needed to perform the localization of the mobile vehicle. Another limitation was that from a single SONAR scan, RCD produced from corners cannot be distinguished from RCD produced from planes.

Tardós *et al.* (2002) compared difference between a SONAR sensors and a laser sensor. For the experimental setup setup, Tardós *et al.* (2002) used 24 fixed SONAR sensors in a ring configuration, equiangularly spaced for feature extraction. They used a Hough Transform as the feature extraction algorithm to determine corners and planes. If the vote is similar for a corner and a plane feature, they used a winner-takes-all strategy to distinguish between them. Tardós *et al.* (2002) chose the Hough Transform over the TBF because it is computationally faster. Tardós *et al.* (2002) performed the experiment while people were moving around the mobile vehicle. One limitation of their work is that similar SONAR return can be achieve when the SONAR sensors attached to the mobile vehicle moves perpendicular to a wall or in a straight line toward a corner or edge. Another general limitation to the Hough Transform is that the detected line goes on forever (i.e., it does not have a starting or ending point). This limitation was not mentioned by Tardós *et al.* (2002). Yap and Shelton Yap and Shelton (2009) used the Randomized Hough Transform (RHT) to detect line features. The RHT algorithm is based on the assumption that the detected lines are orthogonal to each other. If a detected line is not perpendicular to another, then, it is removed. Yap and Shelton tested the RHT algorithm in a large environment setting.

Lee and Song (2010) developed a new feature extraction for SONAR, known as SONAR salient feature. The sonar salient feature is effective at detecting point features such as corners and edges. For the SONAR setup, Lee and Song (2010) used 12 fixed SONAR sensors in a ring configuration. One limitation to this work by Lee and Song (2010) is that they added artificial landmarks (e.g., cylindrical

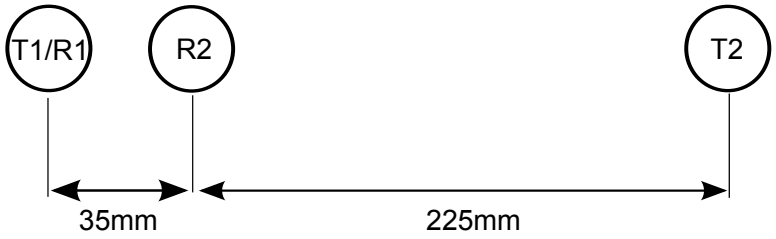


Figure 1.7: Kleeman and Kuc (1994) sensor arrangement: T1: first transmitter, R1: first receiver.

landmarks) that were placed at certain locations, since it was difficult to detect natural landmarks.

Kleeman and Kuc (1994) tried a different approach for the sonar arrangement. Kleeman and Kuc (1994) used two transmitters and two receivers rather than a fixed SONAR ring or the servo-mounted SONAR configuration. The setup is shown in Figure 1.7. Using this setup, Kleeman and Kuc (1994) achieved 0.1^o accuracy and 1 mm range accuracy. Also, targets separated by 10 mm can be distinguish from each other. A major contribution was the minimal number of sensors needed to distinguish between corners, planes, and edges. The SONAR returns from a plane and right angle corner are similar and can be hard to differentiate. An earlier work, Barshan and Kuc (1990) used the information (amplitude and range) from a SONAR return to differentiate corners from planes. Another work, Kuc and Barshan (1989) used SONAR for navigation purpose and the strategy he used is to find edges to avoid obstacles. The edges were the most difficult to detect. Therefore, for feature extraction, edges are the least favored feature to extract from the environment.

Continuing on the work of Kleeman and Kuc (1994), Fazli and Kleeman (2007) presented a novel delayed-classification algorithm that was used for feature classification while the mobile vehicle is moving. For the SONAR setup, Fazli and Kleeman

(2007) used 24 simultaneously-fired transmitters and 48 receivers. The algorithm was capable of determining geometric features within 10 cm of mobile vehicle travel. The top speed achieved by the mobile vehicle was 30 cm/s. However, at 30 cm/s, most of the point features and some of the line features were not recognized, but for the sake of navigation, the few features that were recognized were enough for the mobile vehicle to travel. Fazli and Kleeman (2007) suggested using slower speeds if more features need to be detected. In another different use of SONAR sensors, Steckel and Peremans Steckel and Peremans (2013) used biomimetic SONAR sensors to perform SLAM. However, in this effort, features in the environment were not extracted, since the focus was on the localization of a mobile vehicle navigating through a complex environment. The algorithm was tested in a corridor environment. As expected, this environment was hard to navigate because of the lack of features.

1.3 Vehicle Model

Commonly kinematic models are used to represent the vehicle motion in SLAM algorithms (Wijk *et al.*, 1998; Kuc and Barshan, 1989; Baolong *et al.*, 2007). Recently, few researchers have started to account for vehicle dynamics (Jaai, Chopra, Balachandran, and Karki, 2012) in SLAM. There are several reasons why vehicle dynamics are avoided in SLAM; notably, it is more complicated and requires more time to compute. Implementing vehicle dynamics models in SLAM can have several advantages. For example, a vehicle dynamics model can allow for more accurate

Table 1.5: Advantages and disadvantages of Ackermann steering

Ackermann Steering	
Advantages	**Disadvantages**
Lateral stability at high speed	Low maneuverability
Controllability	Explicit mechanical steering
Low power consumption	

motion estimation. Additionally, vehicle dynamics models in SLAM are more suitable for unstructured outdoor environments. For example,Jaai *et al.* (2012) studied the use of a vehicle dynamics model in a slippery environment, such as oil. Dynamic models consider forces action on the vehicle, such surface-wheel interactions, and account for acceleration of the vehicle.

The dynamic model of each vehicle is dependent upon the vehicle geometry and steering mechanism. The most commonly used steering configurations in mobile wheeled vehicles are Ackermann steering, differential steering, and skid steering. Each of these steering methods has advantages and disadvantages and are summarized in Tables 1.5, 1.6, and 1.7. For more information about the benefits of a vehicle dynamics model and different types of mobile vehicle steering, the reader is referred to Yu, Chuy, Collins Jr., and Hollis (2009). For this dissertation work, the author will focus on skid steering wheeled mobile vehicle. A skid steering wheeled mobile vehicle has non-steerable wheels, and lateral slip must happen for the vehicle to turn. This kind of vehicle steers by having the left and right wheels, rotate at different speeds.

Table 1.6: Advantages and disadvantages of differential steering

Differential Steering	
Advantages	**Disadvantages**
High maneuverability	Low traction
Simple mechanical steering	

Table 1.7: Advantages and disadvantages of skid steering

Skid Steering	
Advantages	**Disadvantages**
High maneuverability	Motion tends to be energy inefficient
Faster response	Difficult to control
Simple and robust mechanical structure	Tires wear faster
Strong traction	
Suitable for most and difficult terrains	

1.3.1 Literature Review on Vehicle Model

Anousaki and Kyriakopoulos (2007) were the first to apply skid steering kinematics to a SLAM problem. Mandow, Martinez, Morales, Blanco, Garcia-Cerezo, and Gonzalez (2007) kinematic model for skid-steering mobile vehicle was used. The parameters in the kinematic model that were used were experimentally determined. In the experiments information from the encoder and INS (Inertial Navigation System) to estimate the mobile vehicle position, and two laser sensors were used to measure features in the environment. Furthermore, Anousaki and Kyriakopoulos (2007) performed experimental work in an outdoor environment where the mobile vehicle had to move 15 meters. Anousaki and Kyriakopoulos (2007) used a covariance intersection filter, because it doesn't hold correlation between features, and therefore the matrix does not grow as large as in the case of the Extended Kalman Filter. Yi, Zhang, Song, and Jayasuriya (2007) used a similar experimental arrange-

ment and vehicle kinematics model for a four-wheeled skid steering mobile robot. To measure the slippage of the wheel, Yi *et al.* (2007) used only an encoder and a low-cost inertial measurement unit(IMU), while the validating the results using computer vision. To get the velocity from the IMU, which gives acceleration measurements, a Nonlinear Kalman Filter was applied. Yi *et al.* (2007) determined that wheel slip was reduced when the left and right wheel rotated at similar rates.

Reina, Ojeda, Milella, and Borenstein (2006) developed a novel method to measure longitudinal slippage and sinkage for mobile wheel vehicle. To measure the longitudinal slippage, Reina *et al.* (2006) used wheel encoder, IMU and electric current sensor and achieved 61% correct detection of longitudinal slip. Whereas for sinkage, Reina *et al.* (2006) attached a camera to the wheel with a field of view containing the wheel terrain interface. Another work by Reina, Ishigami, Nagatani, and Yoshida (2008), Reina *et al.* (2008) came up with a novel approach to measure the lateral slippage using a rearward facing video camera. The camera measures the pose of the trace that was produced by the wheel. Following, Hough Transform enhanced by fuzzy reasoning was done for post processing. Recent work by Reina, Ishigami, Nagatani, and Yoshida (2010) integrated longitudinal and lateral wheel-terrain slip model to improve the mobile vehicle pose estimate.

Caracciolo, De Luca, and Iannitti (1999) presented a vehicle dynamics model for a four-wheeled differential drive mobile vehicle, which is also known as the skid steering mobile vehicle. Also, Caracciolo *et al.* (1999) presented a trajectory tracking control for the mobile vehicle in an outdoor terrain. The control algorithm was tested using simulations. Several assumptions were made for the vehicle dynamics

model which are: rigid vehicle moving on a horizontal plane, vehicle speed is below 6 mph, longitudinal wheel slippage is neglected, and tire lateral force is a function of its vertical load. Also, the vehicle model ignored the effects of suspension and tire deformation. Continuing on Caracciolo *et al.* (1999), Kozłowski and Pazderski (2004) presented a vehicle model for a four-wheeled skid steering mobile robot which consists of three parts, including kinematics, dynamics, and drive subsystems. The assumptions considered by Kozłowski and Pazderski (2004) were similar to Caracciolo *et al.* (1999). The control algorithm was tested using simulations. The tracking results presented by Kozłowski and Pazderski (2004) were more robust when compared to Caracciolo *et al.* (1999). For skid-steering mobile wheeled vehicles, the assumption of no longitudinal wheel slippage valid at slow speeds and is not a realistic assumption for most cases.

Lucet, Grand, Sallé, and Bidaud (2009) presented a dynamics model of a six-wheeled skid steering mobile vehicle. Also, a sliding mode controller was used, where the interaction force between the soil and wheel can be ignored. Lucet *et al.* (2009) performed work on simulations. Lucet *et al.* (2009) used similar assumptions as Caracciolo *et al.* (1999); Kozłowski and Pazderski (2004), but considered longitudinal slip, because he was testing the mobile vehicle at higher speeds. Yu *et al.* (2009) presented a dynamics model of a skid steering wheeled vehicle and validated the simulations using experiments. Shuang, Cheung, Cheng, Lei, and Xiaozhong (2007) presented a dynamics model of a four-wheeled skid steering mobile vehicle. For the tire model, Shuang *et al.* (2007) used the semi-empirical tire model which is more accurate than the linear model, but less accurate than the magic formula model.

Shuang *et al.* (2007) avoided the magic tire formula because it requires the longest computation time of the tire models discussed. Several assumptions were made when deriving the vehicle dynamics model. 1) vehicle moves in a horizontal plane, 2) effect of suspension and tire deformation is ignored, 3) low vehicle speed, 4) vehicle has no vertical motion, 5) vehicle has no pitch or roll motion. This work will compare a dynamics mobile vehicle model to a kinematics mobile vehicle model, for a skid steering mobile vehicle while applying EKF-SLAM.

1.4 Problem of Interest

SOund NAvigation and Ranging (SONAR) is used in many mobile vehicles for measuring distances, since it is inexpensive when compared to lasers and ranging cameras. Inspection of oil pipelines and storage tanks is one application where SONAR sensors are desirable. Since oil is flammable, the use of laser sensors is avoided, for reasons of safety. Furthermore, since oil is opaque, cameras are not used for ranging purposes. Since oil storage tanks and pipelines are made off steel, digital compasses are avoided. The inspection mobile vehicle will be used inside the oil storage tank, therefore Global Positioning System (GPS) is avoided, since the GPS signal can't reach inside the oil storage tank. SONAR has a great ranging accuracy (approximately 1% error of the distance measured), but its angular uncertainty (approximately $22.5^o - 30^o$) is large (Wijk *et al.*, 1998). Other SONAR related limitations include the following: i) cross talk between two or more sensors, ii) multiple reflections, and iii) weak echoes (Weiqin, 2009).

Mobile vehicles may be used in oil storage tanks to locate and identify defects on the floor of the tank. In order for the inspection to be completely autonomous, the mobile vehicle must first locate its own position within the tank. Once the location of the mobile vehicle is known, it may use one of the non-destructive inspection schemes to determine the presence of a fault and report the fault location. The long term goal of the current work is to reach a state of fully autonomous non-destructive inspection.

In oil storage tanks, the presence of oil makes it difficult for mobile vehicles to avoid slipping. Kinematic models tend to fail in environments where slip is large, as was shown in the study of Jaai *et al.* (2012). In these types of environments, the dynamics of the mobile vehicle must be considered. During the inspection of the base of the oil storage tank, the mobile vehicle needs to move from one location to another while estimating its own location by using motion sensors, such as Inertial Measurement Unit (IMU) and encoder. Next, the use of SONAR sensors will be used to estimate the mobile vehicle location and estimate and find new landmarks (or features) around the environment. Finally, a filter such as EKF will be used to perform Simultaneously Localization And Mapping (SLAM). Oil tank and pipeline inspection are applications where considering vehicle dynamics is crucial because of slippage, incline of the environment, and other factors such as acceleration and deceleration. In addition to the development of a feature extraction algorithm using SONAR sensors, vehicle dynamics with application to SLAM will be investigated.

1.5 Objectives

The overall goal of this work is to understand feature extraction using SONAR sensors, to implement it using an Extended Kalman filter for SLAM, and also to understand how vehicle motions influence SLAM. Specific objectives include the following:

1. Develop a feature extraction algorithm that can identify points and lines in an unknown environment from SONAR data

2. Study application of feature extraction algorithm on experimental data and compare with existing algorithms used in the literature for SLAM application

3. Consider vehicle motions in the SLAM formulation, with special emphasis on vehicles which experience wheel slippage; for example, skid steering

4. Examine the proposed fusion algorithm with EKF-SLAM on different slippage environment and compared it to a no-SLAM case.

1.6 Organization of Dissertation

The rest of the dissertation is organized as follows. In Chapter 2, a comparison of commonly used feature extraction algorithms is compared and validated with simulation as well as experimental data. In Chapter 3, the proposed algorithm fusion is explained in detail. The proposed fusion algorithm is studied on simulation data for two different environments. Next, the proposed fusion algorithm is applied

to experimental data for two different environments with and without interior features. In Chapter 4, four different approaches for calculating the vehicle model are explored. All four different approaches are studied in two different environments with and without interior features. In Chapter 5, the proposed fusion algorithm from Chapter 3 and the varying signal drift approach from Chapter 4 are used to implement the SLAM algorithm. The SLAM algorithm is studied on three different environments, where the floor of the environment is altered, for example, with dry powder and with a hydrophobic coating (called NeverWet), to induce slipping. The SLAM and no-SLAM cases are then compared for different environments. Finally, Chapter 6 contains a summary of the contributions made in each of the different chapters as well as suggested directions for future work.

Chapter 2

Commonly Used Feature Extraction Algorithms

In this chapter, the TBF, Hough Transform , THF, and SONAR salient algorithm are validated in simulated square environments. These algorithms are tested on experimental SONAR data, where the mobile vehicle travels in a known rectangular environment. The commonly used algorithms in the literature (TBF, Hough Transform, THF, and SONAR salient) are compared for two different vehicle trajectories. Unlike TBF, Hough Transform, THF, and SONAR salient k-means algorithm is not a feature extraction algorithm. It is a clustering algorithm, it aims to separate data into groups and then taking the mean of it making sure the each data belong to the cluster with the nearest mean (Hartigan and Wong, 1979).

2.1 Validation of Algorithms

Before implementing the algorithms on experimental data, they are validated with simulated data that represents a square box. The simulated square box is shown in Figure 2.1. The TBF algorithm will be tested on the square box, followed by the Hough Transform, THF, and SONAR Salient feature extraction algorithm.

First, the TBF algorithm for two different threshold values of the parameter n_t, 0 and 6, is shown in Figure 2.2. It can be seen that the TBF algorithm works well with the simulated square box, since it captures all of the edges quiet well. Next,

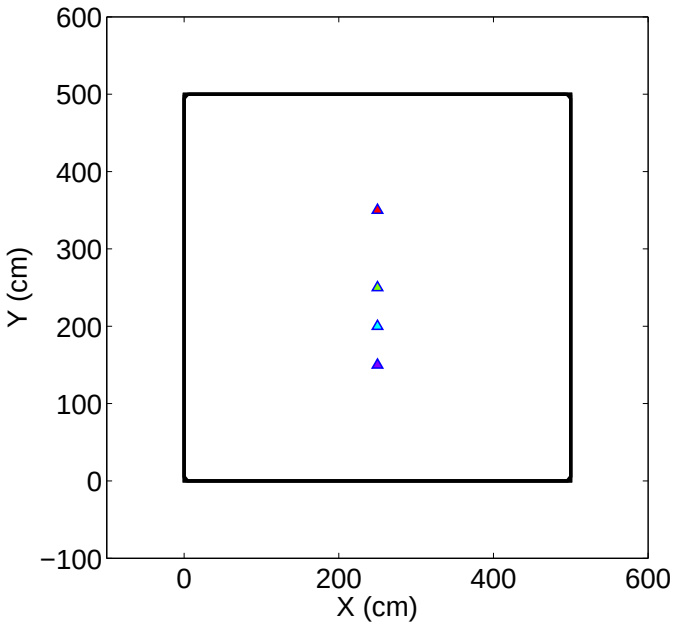
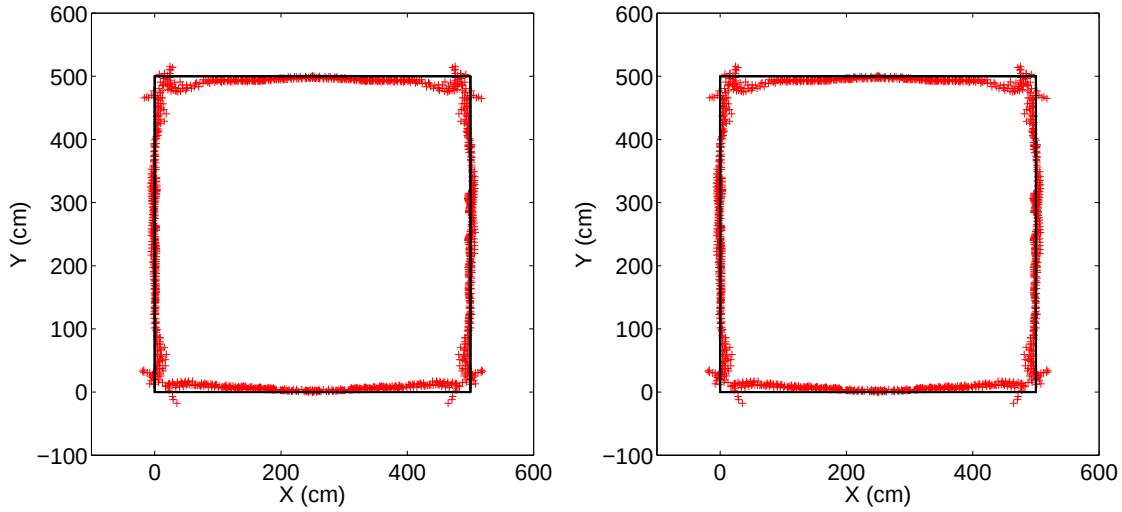


Figure 2.1: A simulated square box used to test different algorithms. The triangles mark the position of the mobile vehicle at different locations.

the Hough Transform was tested on the square box, and the results are shown in Figure 2.3a. After, the THF was tested on the processed TBF data for a threshold parameter n_t value of 6, and the THF results are shown in Figure 2.3b. Finally, the SONAR salient feature extraction algorithm was tested on the square box, and the results are shown in Figure 2.4. The main purpose of the simulation was not to compare between the algorithms, but to test if the algorithms give reasonable output. The algorithms performed as expected, and these algorithms can be used for feature detection for SLAM.

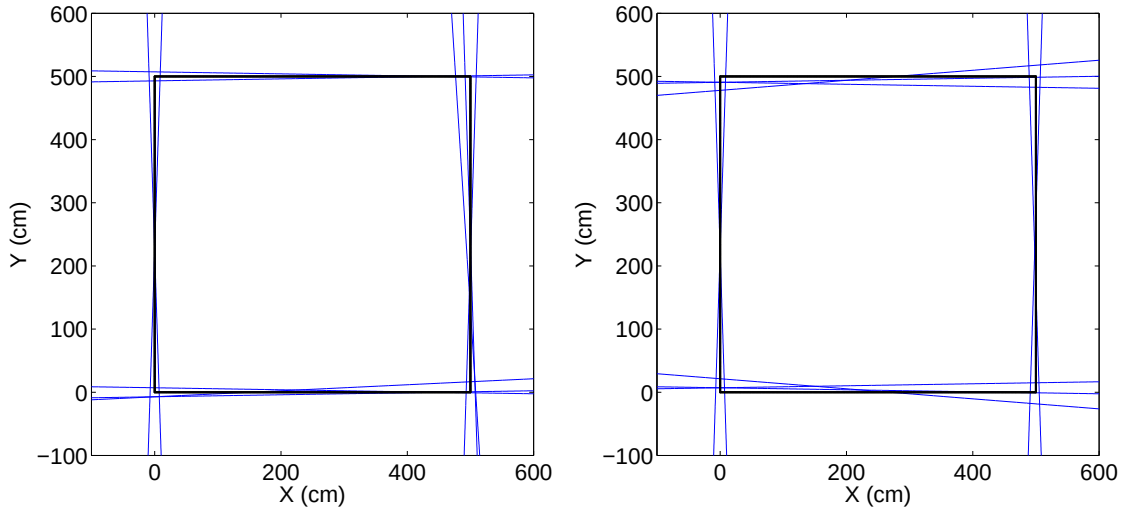
2.2 Experiment Setup

The algorithms discussed in the previous section are implemented on SONAR data obtained in an experimental arrangement. This arrangement is similar to the



(a) $n_t > 0$ (b) $n_t > 6$

Figure 2.2: TBF results for different threshold values. The actual box is superimposed over the processed TBF data for comparison purposes only.



(a) Hough Transform (b) THF

Figure 2.3: The Hough Transform and THF algorithm results for the simulated square box. The number of divisions is 200, and the number of lines is 100 for both algorithms.

configurations used previously in the group (Jaai *et al.*, 2012) for SLAM studies. In this arrangement, four walls were placed to form a closed environment, as shown in Figure 2.5, a mobile vehicle was constructed with the following components: i) two SONAR sensors, ii) servo motor, iii) Arduino micro-controller, and iv) Xbee

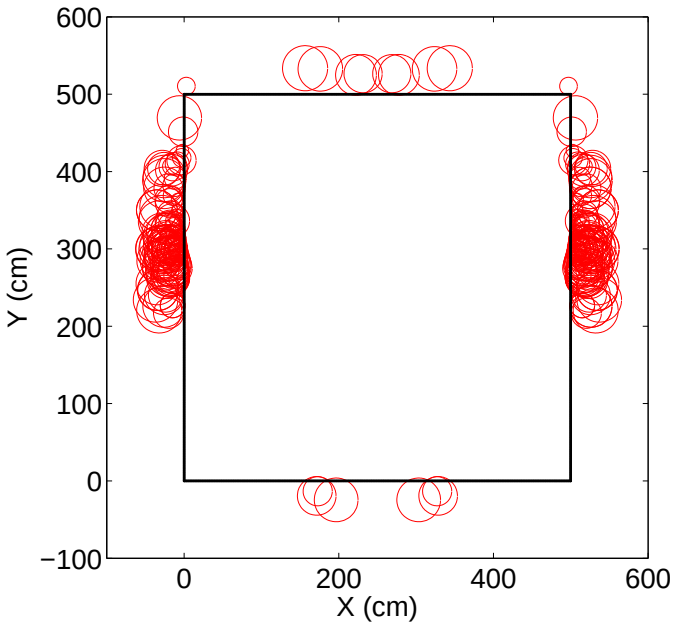


Figure 2.4: The SONAR salient feature extraction algorithm results for the simulated square box. The acceptable radius range was between 3 cm and 30 cm.

wireless communication. The two SONAR sensors are attached to a servo motor, as shown in Figure 2.6. To get a full scan, the servo motor rotates 180^o, with 1^o degree increments. Since the SONAR sensors are on opposite sides, a 180^o degree rotation of the servo motor will allow for a full scan of 360^o of the environment.

The experiments are performed with the following procedure. First, the mobile vehicle is placed inside the experimental environment. The mobile vehicle is moved to 10 different locations, and a full SONAR scan is performed at each location. The 10 locations are shown in Figure 2.7 for Case 1. The mobile vehicle moves to 15 new locations near the top right corner for Case 2. Case 2 was introduced to examine the performance of the algorithms when the mobile vehicle is near a corner.

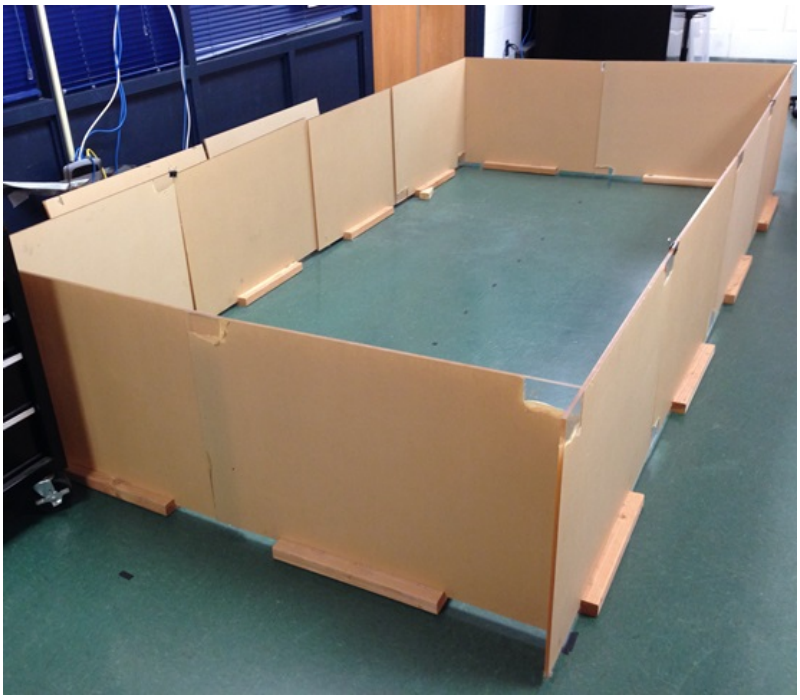


Figure 2.5: Experimental arrangement with the following dimensions: width = 170.8 cm, length = 348.7 cm, and height = 63.5 cm.

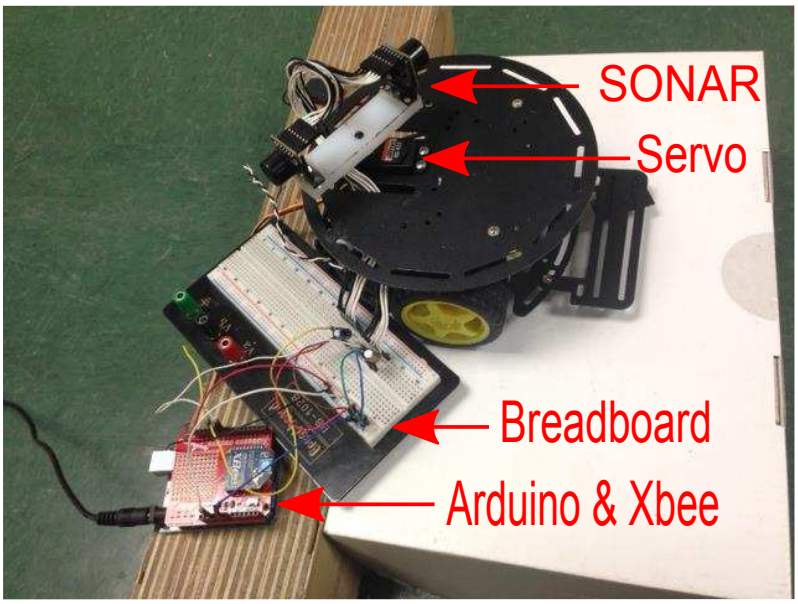
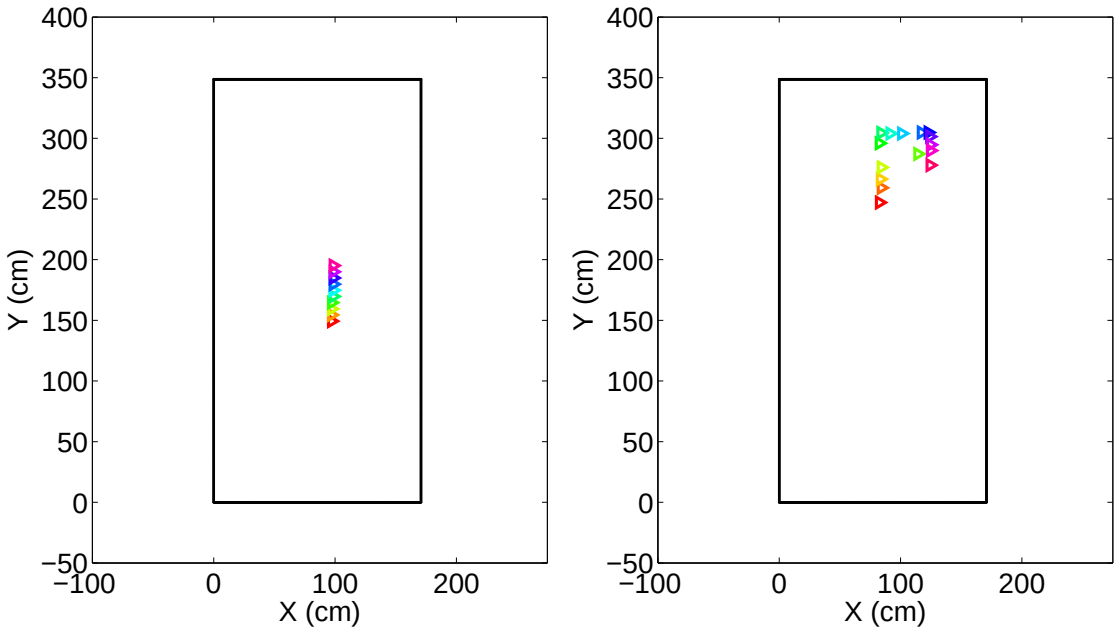


Figure 2.6: Mobile platform used in the experiments.



(a) Straight trajectory Case 1

(b) Curved trajectory Case 2

Figure 2.7: Schematic diagram of the experimental setup. The mobile vehicle is represented by triangular symbol, and its true positions have been measured for 10 different locations for Case 1 and 15 different locations for Case 2.

2.3 Experimental Results

This section is divided into two subsections, where results are shown separately for Case 1 and the other for Case 2. For each case, results from TBF, the Hough Transform with k-means clustering, THF with k-means clustering, and SONAR salient feature extraction algorithm are shown.

2.3.1 Case 1

The raw SONAR data obtained from the ten different locations are shown in Figure 2.8. Each SONAR point is represented by an arc. The arc angle is 30^o, as shown in Figure 2.9. Next, the SONAR data is filtered, accepting only data between 31 cm and 200 cm. SONAR data outside this range is considered unreliable. Since

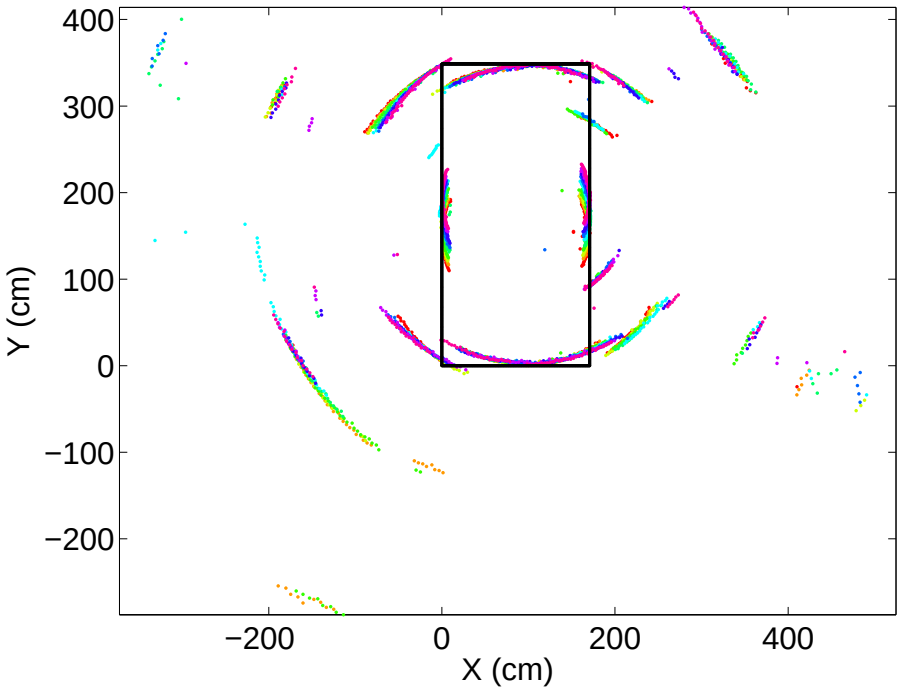


Figure 2.8: Raw SONAR data (3600 points) for the 10 different locations considered in the experiments for Case 1. The actual box is superimposed over the SONAR data for comparison only.

the SONAR device (Maxibotix HR-LV MAXSONAR EZ4) used cannot detect data below 30 cm, any value below this is rounded up to 30 cm. Through experiments conducted with (Maxibotix HR-LV MAXSONAR EZ4) SONAR it was noticed that SONAR data above 200 cm were not stable. The filtered data is shown in Figure 2.10. Just by applying the limit filter, an improvement can be seen in the SONAR data, as shown in Figure 2.10. The filtered SONAR data was sent to the TBF algorithm. The TBF results are shown in Figure 2.11. The TBF algorithm has been generated for different threshold values as shown in Figure 2.11.

To detect the line features, the standard Hough Transform (see Figure 2.12) was used. Next, the filtered data was sent to the Hough Transform algorithm and the results obtained are shown in Figure 2.13a. By using the Hough Transform,

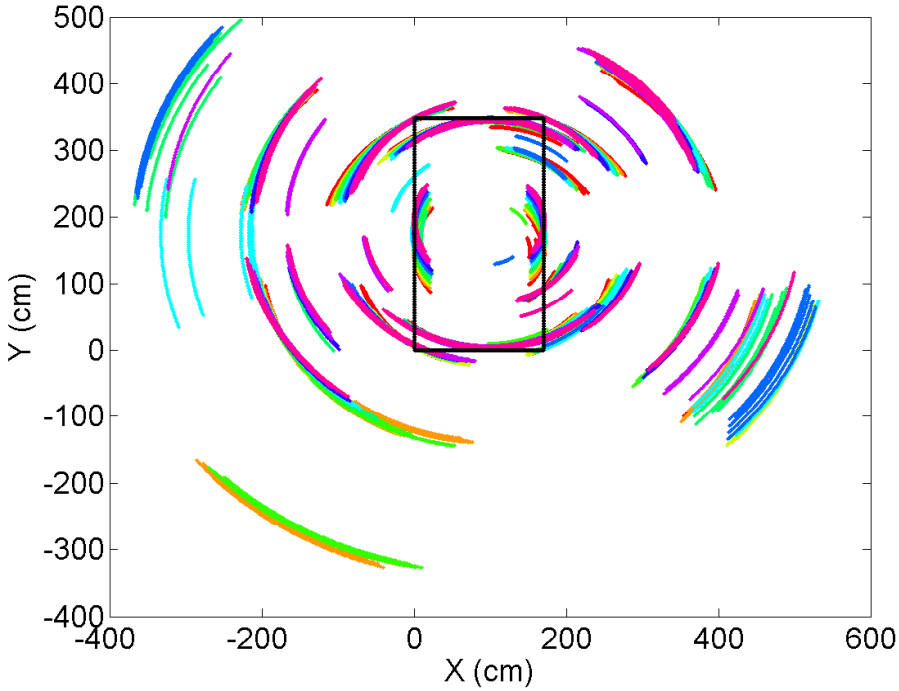


Figure 2.9: SONAR data is represented by arcs for Case 1, and the arc angle is 30^o.

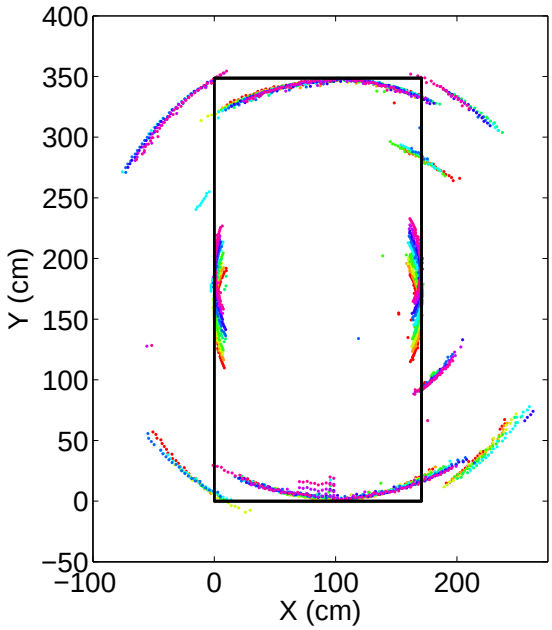
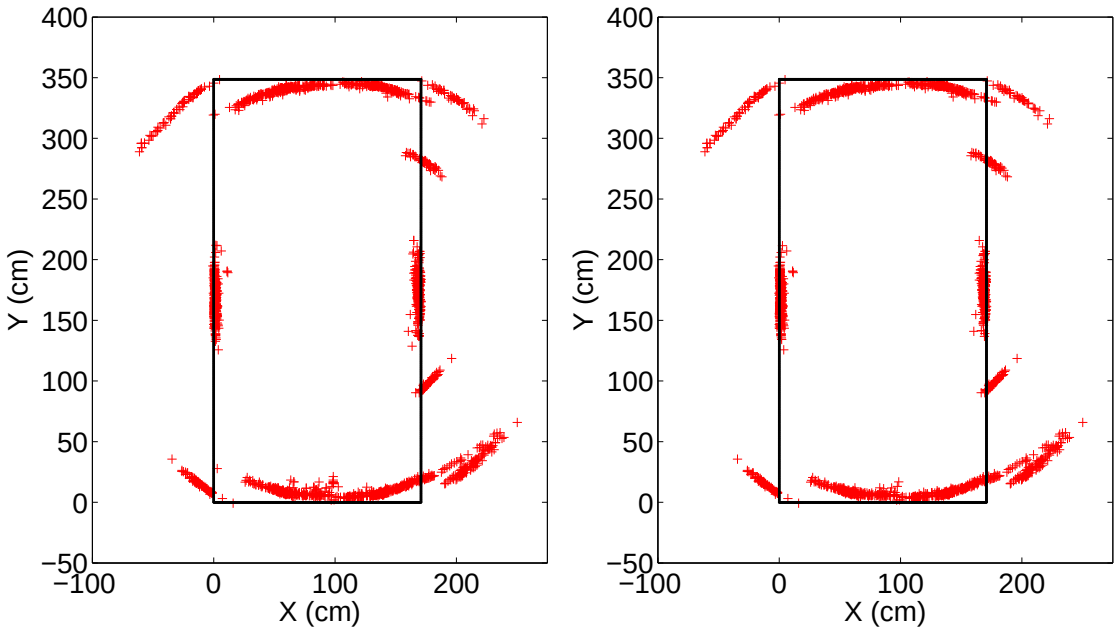
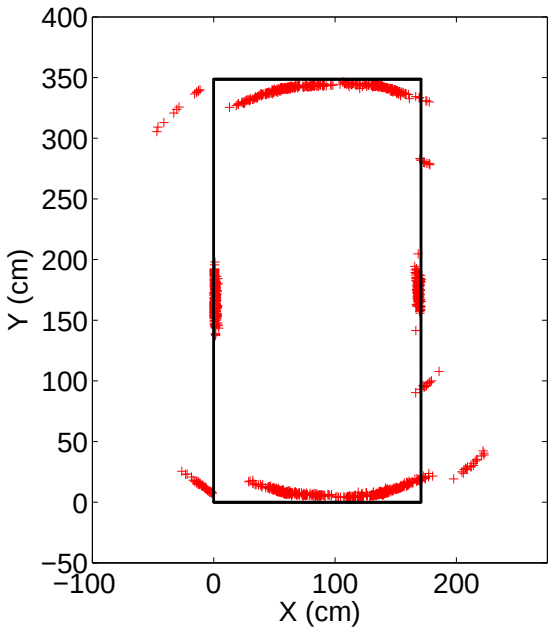


Figure 2.10: Filtered SONAR data for Case 1; accepted data is between 31 cm and 200 cm.

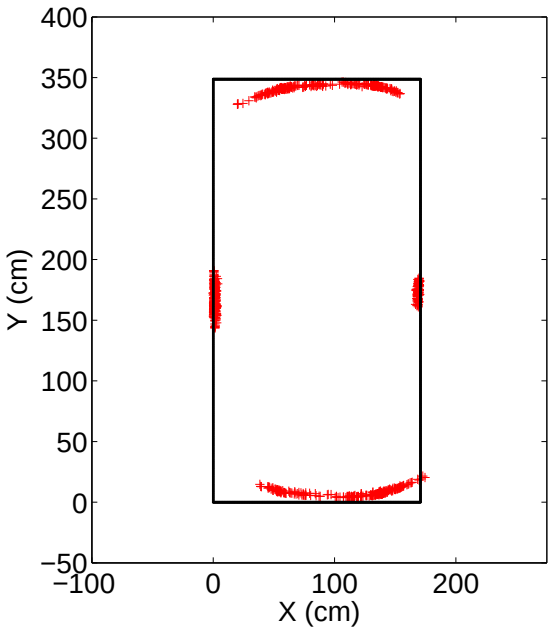


(a) $n_t > 0$

(b) $n_t > 6$



(c) $n_t > 100$



(d) $n_t > 200$

Figure 2.11: TBF results for different threshold values for Case 1.

four separate lines were detected. It is noted that a clustering algorithm can be implemented later as a form of averaging, which may lead to a better agreement of the results to the actual box. One limitation of the Hough Transform is that

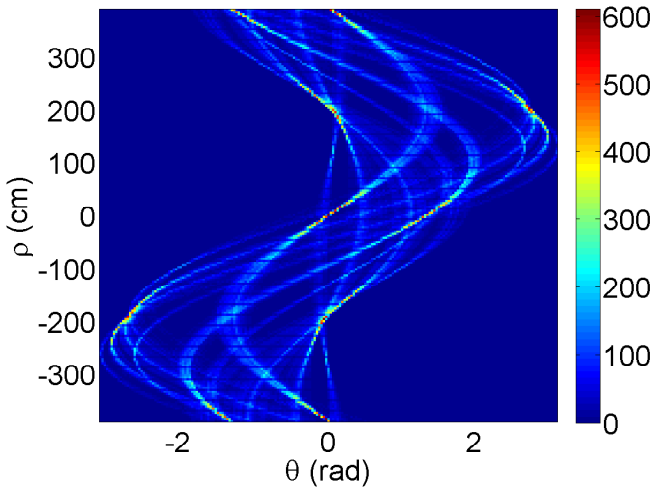
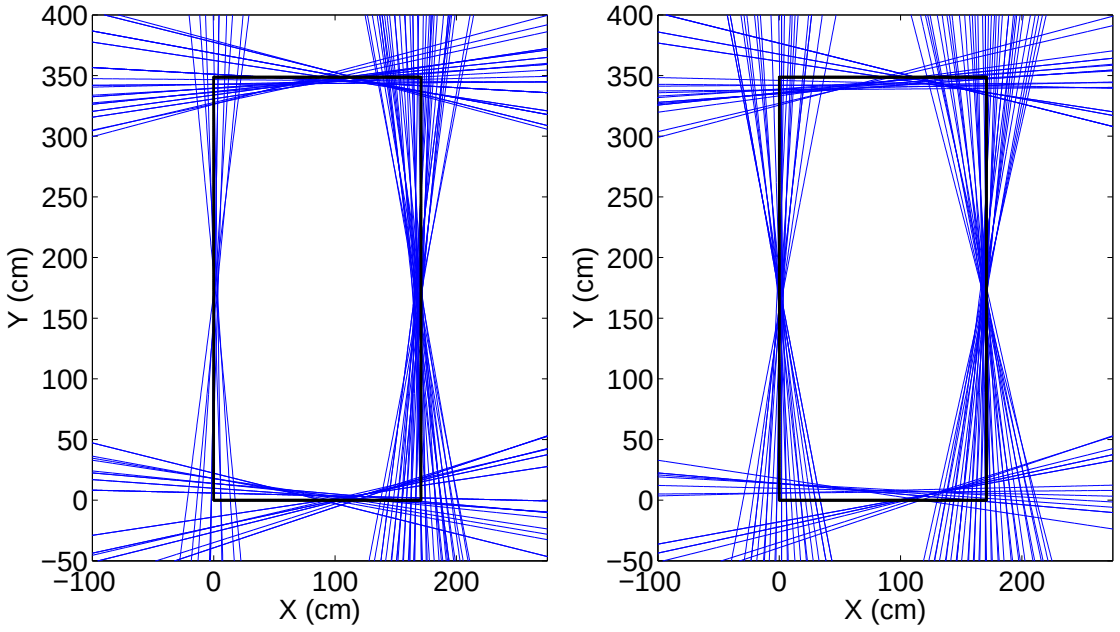


Figure 2.12: ρ and θ bins, the colors correspond to the number of votes.

the lines extend forever. In other words, one cannot tell where the line has started and where it ends. Therefore, in the next algorithm (THF), the data obtained from TBF analysis are used to enhance the Hough Transform results by determining the starting and ending points of the line.



(a) Hough Transform (b) THF

Figure 2.13: The Hough Transform and THF algorithm results for Case 1. The number of divisions is 200, and the number of lines is 100 for both algorithms.

The THF results are shown in Figure 2.13b. As mentioned, the THF implementation is composed of two steps, namely the TBF and Hough Transform. As an example, the input to the Hough Transform is TBF results obtained for $n_t > 6$, which is shown in Figure 2.11.

It is difficult to compare the results between Figure 2.13a and Figure 2.13b, before applying a clustering algorithm. However, it is expected that after implementation of the clustering algorithm, both results will be close to the actual box. Therefore, a k-means clustering algorithm was applied to the results shown in Figure 2.13a and Figure 2.13b. Before applying the k-means clustering algorithm, the results went through gating conditions. The gating conditions help the k-means clustering algorithm to cluster data correctly. The gating conditions are described in Figure 2.14. The gating conditions insure that ρ of the detected line is positive and θ of the detected line is between $(1^o + 45^o)$ and $(360^o + 45^o)$, the reason for this range is because values lower than 45^o belong to similar line group of values above 315^o if ρ value is close enough for both group.

In the subplots (a) and (c) of Figure 2.15, the top 100 voted lines from the Hough Transform and the THF algorithm are shown. This data went through the gating conditions shown in Figure 2.14 first, and then through the k-means clustering algorithm. In k-means clustering, the number of clusters is provided by the user. As future work, k-means clustering with Expectation Maximization (EM) is suggested, as this would supersede the necessity of a user-supplied number of clusters; the results are shown in subplots (b) and (d) of Figure 2.15. Also, the final results of the k-means clustering in Cartesian coordinate is shown in Figure 2.16.

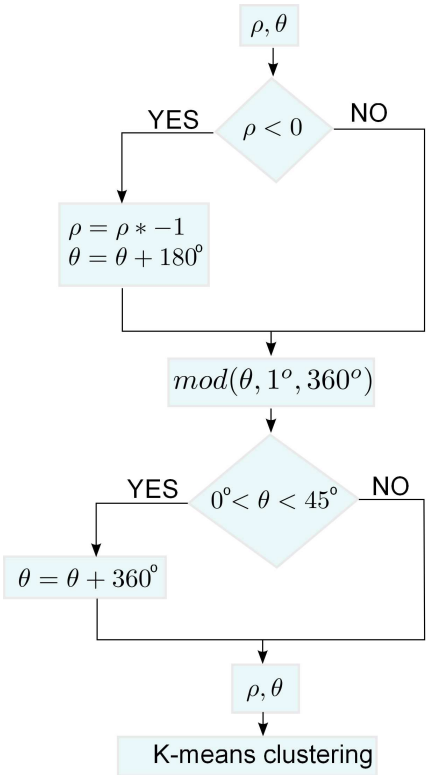


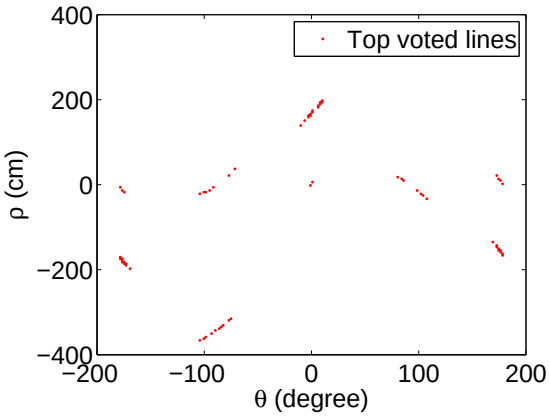
Figure 2.14: Gating condition used before the k-means clustering to ensure correct clustering.

In addition to clustering, the SONAR salient feature extraction algorithm was ran on the limited constraint data and the results are shown in Figure 2.17.

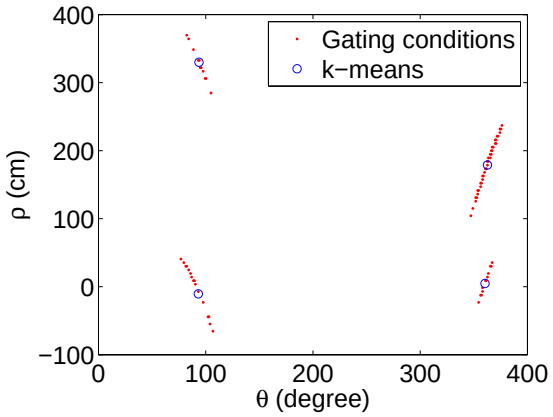
Table 2.1: Case 1: Comparison of the Hough Transform and THF

Line	**Absolute difference Hough Transform ρ(cm), θ(degree)**	**Absolute difference THF ρ(cm), θ(degree)**
Left	$4.77, 0.48$	$7.30, 1.45$
Bottom	$10.56, 3.17$	$8.88, 0.97$
Right	$8.15, 2.79$	$8.71, 2.86$
Top	$18.68, 3.75$	$11.20, 1.06$

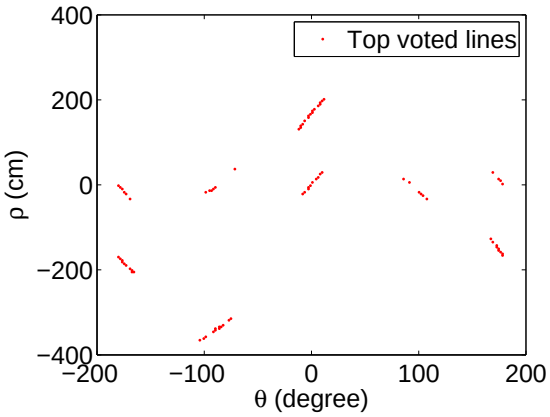
After running the k-means clustering algorithm, a comparison between Hough Transform and THF for Case 1 can be made, which is summarized in Table 2.1. Overall, both algorithms performed well for Case 1. A slight improvement in the THF algorithm can be noticed from Table 2.1. Also, in the worst case, the absolute



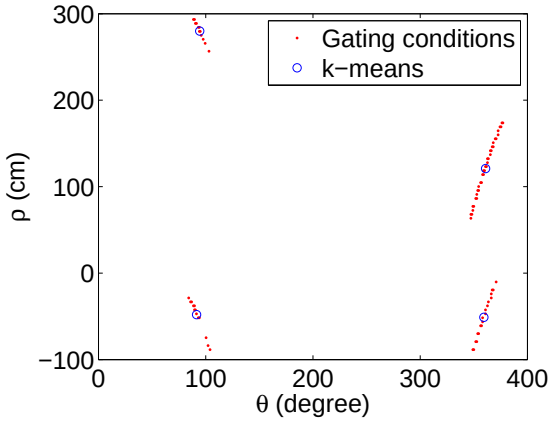
(a) Hough Transform top 100 voted lines



(b) Hough Transform after gating conditions and k-means clustering



(c) THF top 100 voted lines



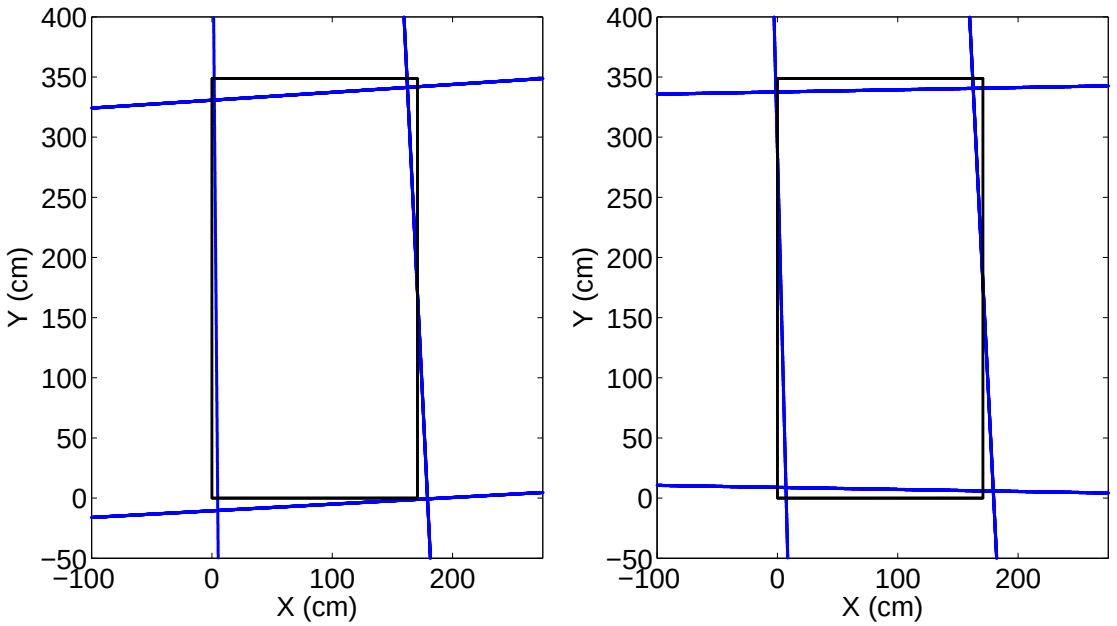
(d) THF after gating conditions and k-means clustering

Figure 2.15: Gating condition and k-means clustering results for Case 1.

difference between the determined and true value did not exceed 19 cm in radial displacement and 4^o in angle.

2.3.2 Case 2

The results obtained for Case 2 are presented in a similar way as that for Case 1. The raw SONAR data are plotted in Figure 2.18. The SONAR data are passed through a filter, and only data between 31 cm and 200 cm is accepted. The filtered results are shown in Figure 2.19. As can be seen from Figure 2.19, the lower wall



(a) Hough Transform (b) THF

Figure 2.16: K-means clustering results for Hough Transform and THF data for Case 1. The actual box was superimposed over the figure for comparison purposes.

was ignored by the filter since it was over 200 cm away. In Case 2, the goal is to test the TBF algorithm near a corner to check performance.

The TBF algorithm is studied for different threshold values and the results are shown in Figure 2.20. Similar to Case 1, as the n_t value increases, most outliers are removed. For the $n_t > 100$ and $n_t > 200$ cases, better outlier removal is seen and more of the environment is captured. As a conclusion, for the TBF algorithm, visiting more locations in the environment yields more features of the environment. The Hough Transform is carried out for Case 2. The results are shown in Figure 2.21a.

As an example, the THF results obtained for $n_t > 6$ are used as input data to the Hough Transform analysis, and the results are shown in Figure 2.21b. It is harder to compare Figure 2.21a and Figure 2.21b, therefore the k-means clustering

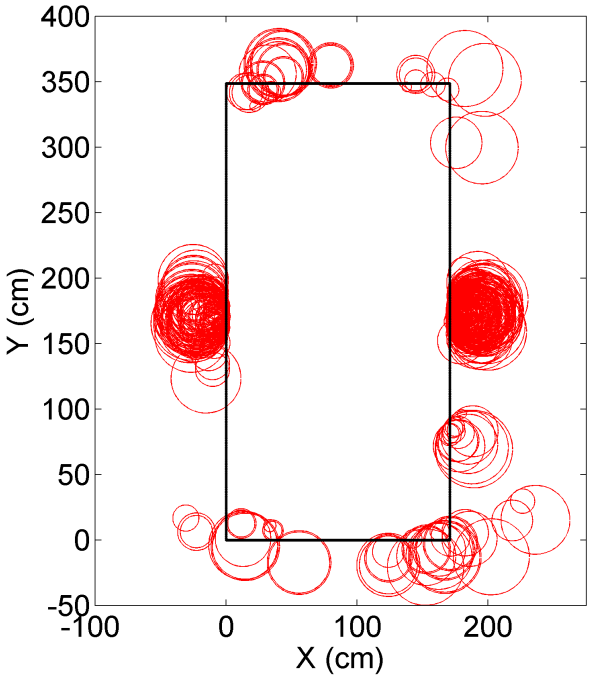


Figure 2.17: SONAR salient feature extraction results for Case 1. The acceptable radius range is between 3 cm and 30 cm.

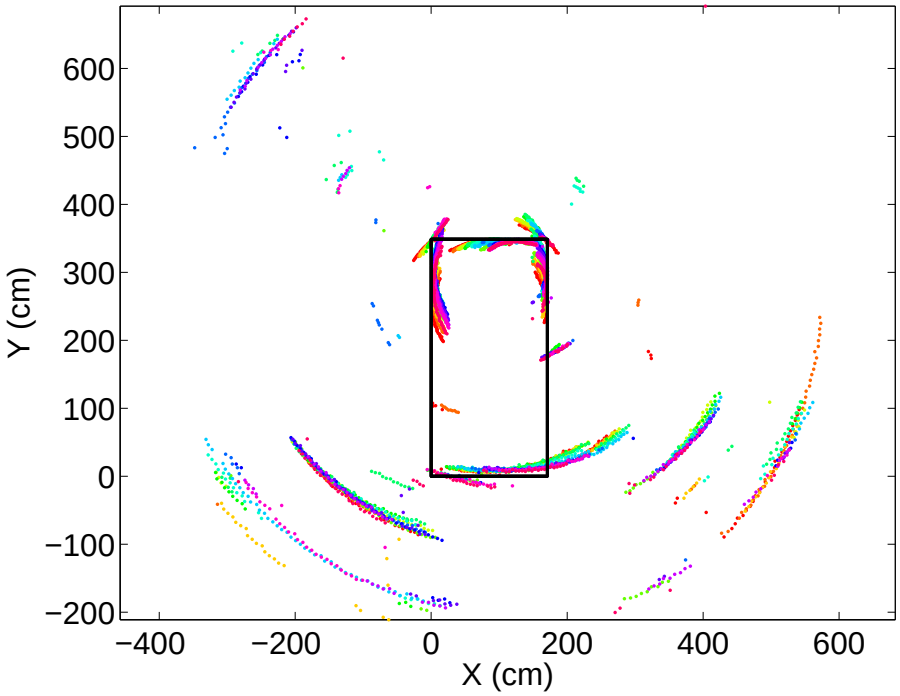


Figure 2.18: Case 2: Raw SONAR data (5400 points) for the 15 different locations considered in the experiments. The actual box is superimposed over the SONAR data for comparison purposes only.

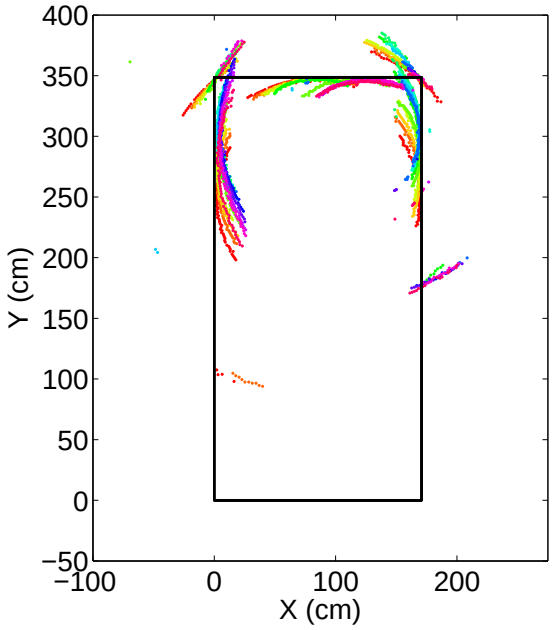


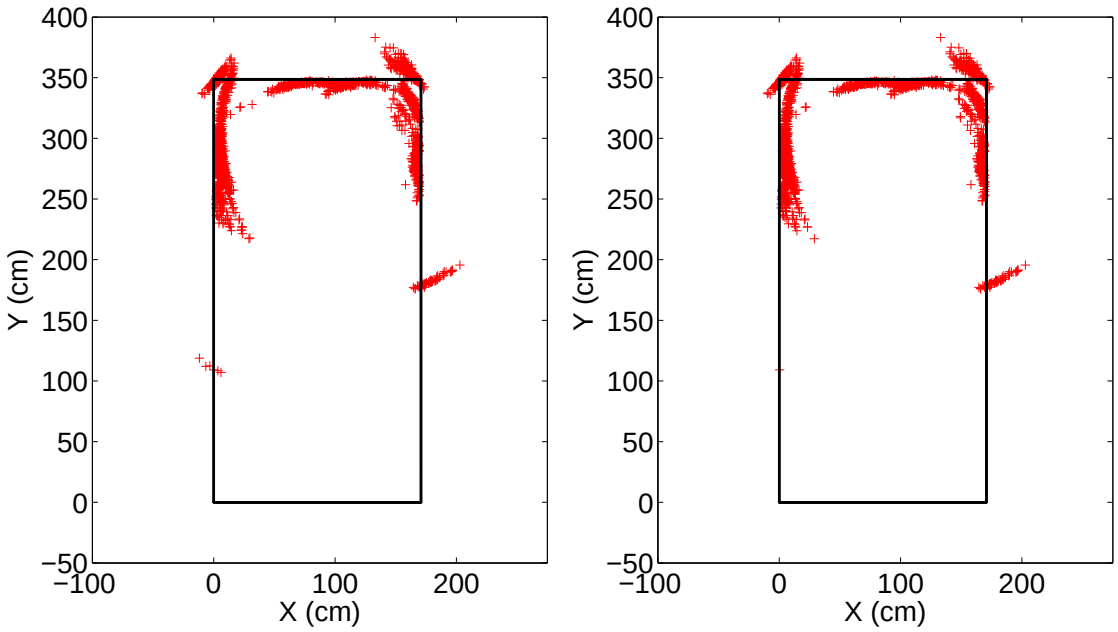
Figure 2.19: Filtered SONAR data for Case 2; accepted data is between 31 cm and 200 cm.

algorithm was used. Similar to Case 1, the 100 most voted lines for the Hough Transform and THF were sent to the gating conditions and k-means clustering algorithm, which is shown in Figure 2.22. Also, the final results of the k-means clustering in Cartesian coordinate is shown in Figure 2.23. In addition to clustering, the SONAR salient feature extraction algorithm was ran on the limited constraint data and the results are shown in Figure 2.24.

Table 2.2: Case 2: Comparison of the Hough Transform and THF

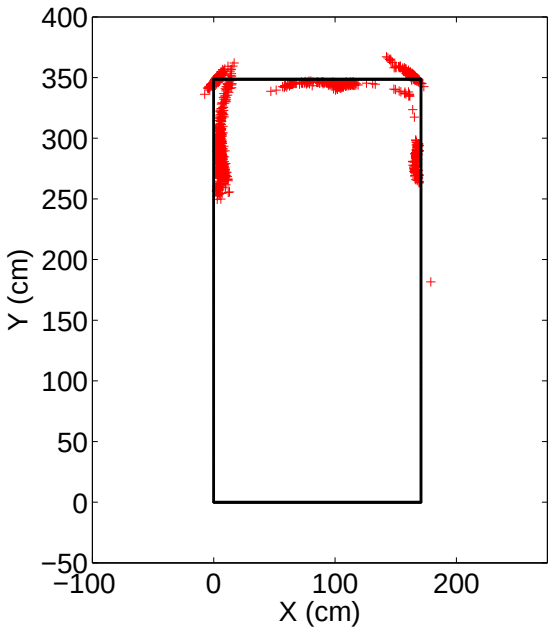
Line	**Absolute difference Hough Transform** ρ**(cm),** θ**(degree)**	**Absolute difference THF** ρ**(cm),** θ**(degree)**
Top	$11.64, 1.30$	$7.46, 0.39$
Left	$10.27, 0.82$	$11.56, 1.06$
Right	$30.23, 6.52$	$31.44, 5.60$

For Case 2, the TBF analysis worked much better when the mobile vehicle was close to a corner and more data was collected in comparison to Case 1. In order to

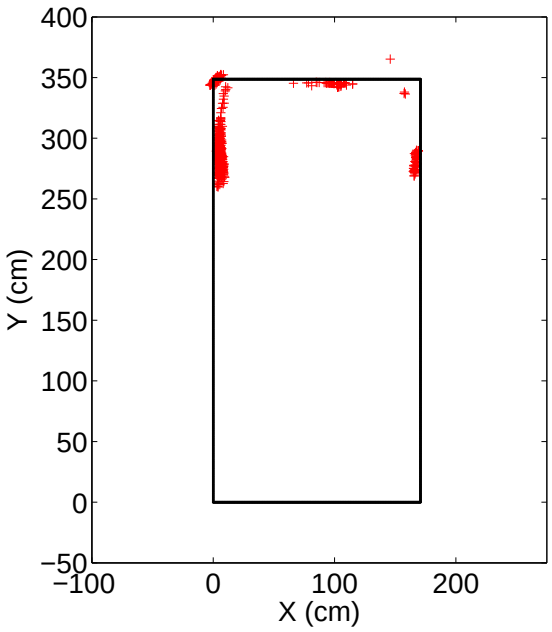


(a) $n_t > 0$

(b) $n_t > 6$



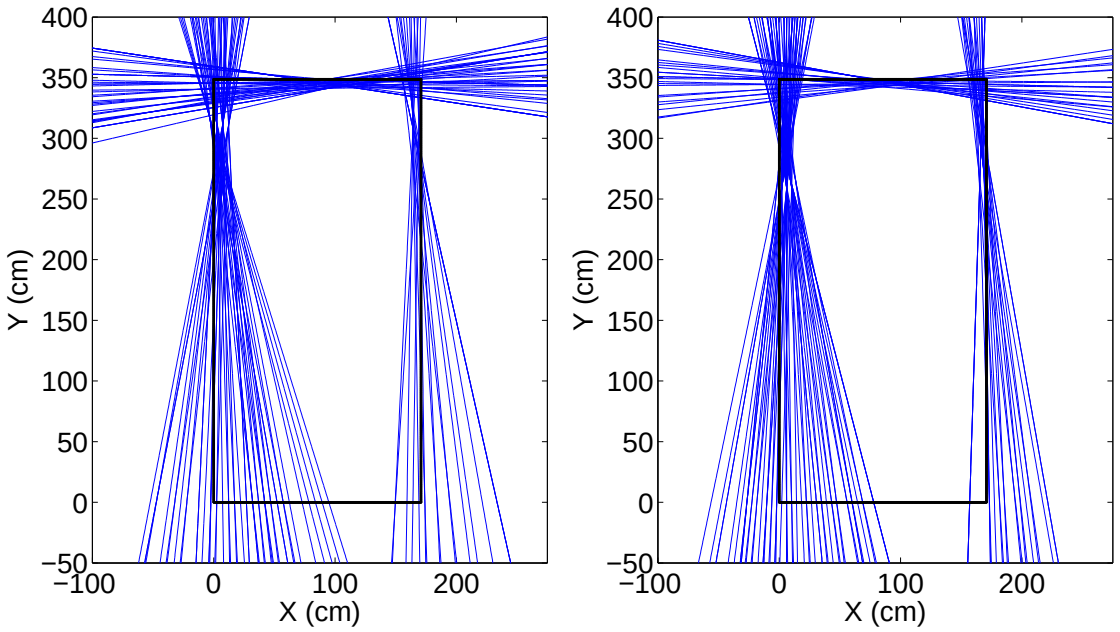
(c) $n_t > 100$



(d) $n_t > 200$

Figure 2.20: Point feature extraction by using the TBF algorithm for different threshold values. It was notice that for point (corner) features, SONAR data tend to make a 45^o angle, this could due to the reflections from the two side walls.

improve the TBF algorithm results, mobile vehicle locations should be increased; for example, TBF results capture more parts of the environment in Case 2 than Case 1,

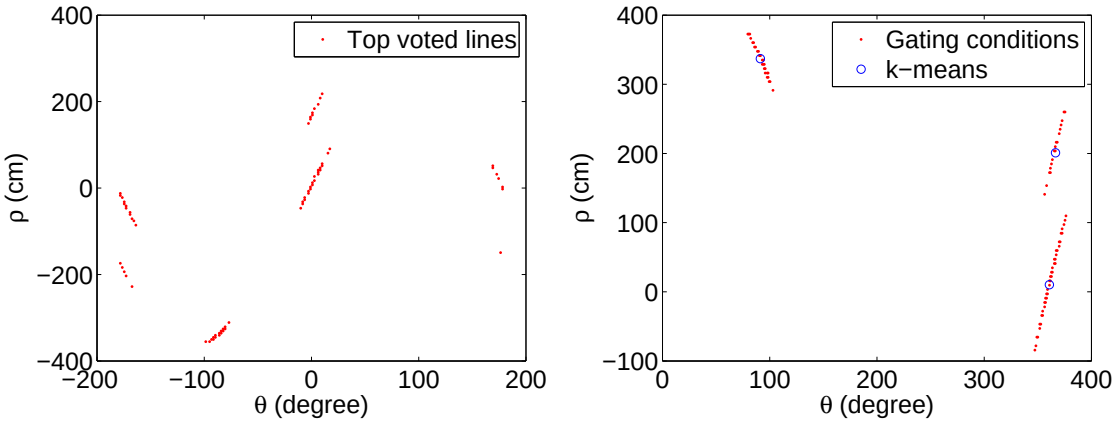


(a) Hough Transform (b) THF

Figure 2.21: The Hough Transform and THF algorithm results for Case 2. The number of divisions is 200, and the number of lines is 100 for both algorithms.

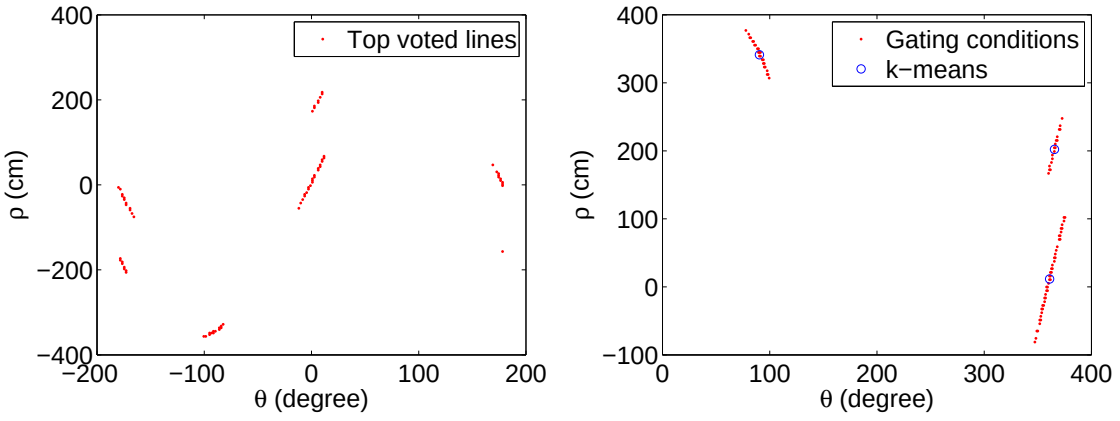
because the mobile vehicle had 5 extra locations to visit. While programming TBF algorithm and Hough Transform algorithm, it was notice that, the TBF algorithm is a slower algorithm compared to the Hough Transform algorithm. The more positions the mobile vehicle visits, the more time that the algorithm would require. Also, the computation time for the TBF analysis grows very quickly for each additional location, for example, for 6 locations it takes around 39 seconds, but for 30 locations it takes around 721 seconds. Note, each location has 360 data. The test was performed on an $i7$ CPU with 3.2 GHz processor and 16.0 GB usable RAM, also 5 cores were used when TBF was performed through the use of "parfor" command in MATLAB.

After running the k-means clustering algorithm for Case 2, the comparison between the Hough Transform and THF for Case 2 can be made and it is summarized



(a) Hough Transform of top 100 voted lines

(b) Hough Transform after gating condition and k-mean clustering

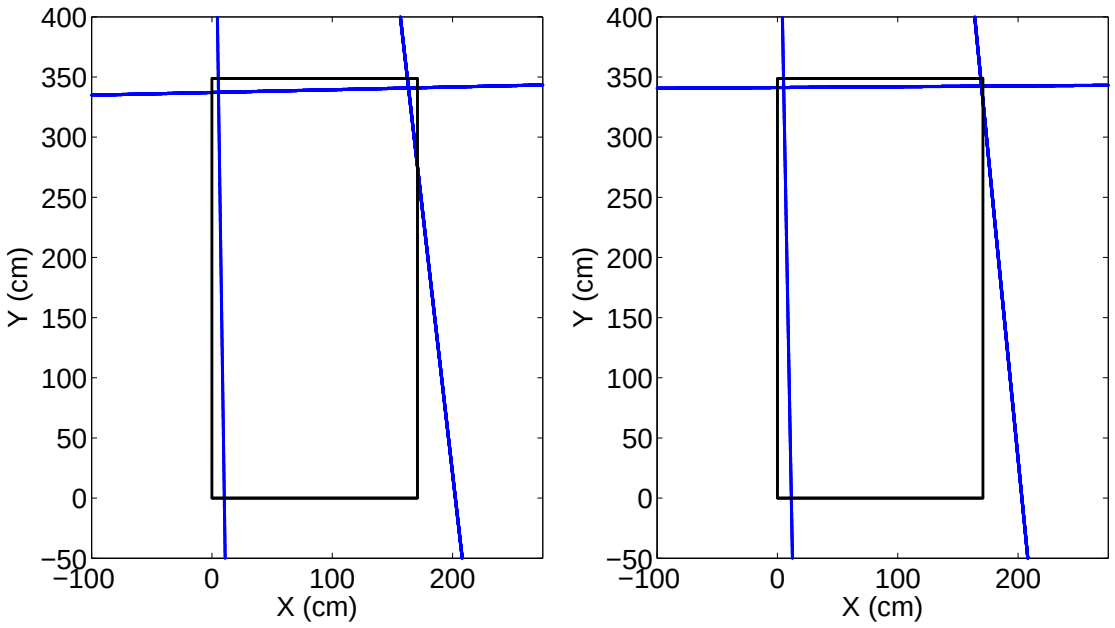


(c) THF top 100 voted lines

(d) THF after gating conditions and k-means clustering

Figure 2.22: Gating condition and k-means clustering results for Hough Transform and THF results for Case 2.

in Table 2.2. Overall, the THF algorithm performed better in general. The worst case for both algorithms (Hough Transform and THF algorithm) was founded to be for the right wall, where the ρ value for the Hough Transform was 30.23 cm, while the value from the THF algorithm was 31.44 cm. Both algorithms performed reasonably well. Again, more data were collect for Case 2, which lead to better results for both TBF and THF algorithms. Also, it was noticed that the left wall results were much better compared to the right wall. This can be explained, because the SONAR



(a) Hough Transform (b) THF

Figure 2.23: K-means clustering results on Hough Transform and THF data. The actual enclosure contour has been superimposed over the figure for comparison purposes only.

sensors were closer to the right wall, making reflection more problematic. Also, the THF is more suitable for offline SLAM, since it is slower. Whereas the Hough Transform is more suitable for online SLAM. Finally, THF gives more information, since it can produce both point and line feature extraction, whereas the Hough Transform can only distinguish line features.

2.4 Summary

In this chapter, a validation of commonly used feature extraction algorithms (TBF, Hough Transform, THF and SONAR salient) have been examined on a simulated square environment. Following that, an experimental setup was build to test and compare the performance of the commonly used feature extraction in a empty

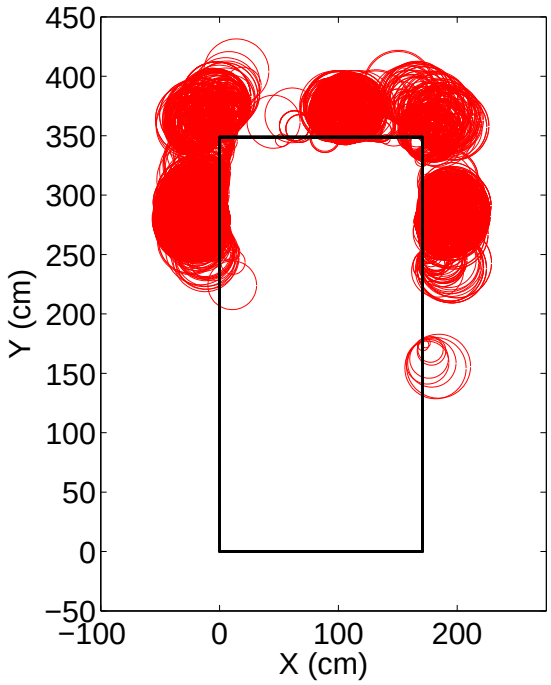


Figure 2.24: SONAR salient feature extraction results for Case 2. The acceptable radius range is between 3 cm and 30 cm.

rectangular environment for two trajectory cases. From this study, the following conclusions can be drawn:

1. The TBF algorithm is seen to work better with more data as shown in Case 2 when compared to Case 1. The TBF algorithm as well as the SONAR salient can capture more parts of the environment if the mobile vehicle stop and perform more full SONAR scans.

2. The computational time of TBF and SONAR salient algorithm increase a lot for each additional location. Each location has 360 data. For example, for 6 locations it takes around 39 seconds, but for 30 locations it takes around 721 seconds. The studies were performed by using an $i7$ CPU with 3.2 GHz processor and 16.0 GB usable RAM, also 5 cores were in used when TBF

studies were carried out through the use of "parfor" command in MATLAB.

3. The Hough transforms and the THF analyses are found to work very well in both cases. A noticeable improvement in THF results over the corresponding Hough Transform results of Case 1 was observed.

4. The TBF, Hough Transform, SONAR salient, and THF algorithms have been compared on the same SONAR data which can be a good benchmarked of the algorithms result. The benchmark of the algorithms on SONAR data was first done here in this dissertation work (Ismail and Balachandran, 2013).

5. Finally, it is expected that SLAM problems based on SONAR will benefit from the TBF algorithm, combined Hough Transform and THF algorithm and SONAR salient feature extraction algorithms.

This study was intended not only to examine if the algorithm worked in simulation and experimental environments, but also to test if different combination of these algorithm can be performed subsequently or simultaneously to realize higher level of information. This idea or step will be explored in the next chapter.

Chapter 3

Proposed Feature Extraction

In this chapter, the proposed algorithm fusion is explained. The proposed fusion algorithm is studied on two different simulated environments to test its performance and accuracy. Following, the proposed algorithm is used on experimental data obtained from the mobile vehicle presented in Chapter 2 on two different rectangular environments with and without interior features.

3.1 Proposed Algorithm Fusion

An original fusion algorithm for feature extraction is proposed in this section. In this proposed algorithm, a combination of available algorithms are used for feature extraction wherein the strength of each algorithm are exploited. This fused algorithm can be used to capture points, lines, and cylindrical features, whereas the TBF and SONAR salient algorithms are limited to point features, and Hough Transform is used best for capturing line features. In this new combined algorithm, the different features in the environment are determined and the redundant or repeated, features are removed. This algorithm is different from the one presented in the work of Yap and Shelton (2009), since orthogonal intersections of lines are not assumed, and line intersections with arbitrary angles are allowed. In addition, the present algorithm can be used to detect corner, cylindrical aspects, and lines. The

TBF algorithm is used to determine point features from the environment, and the Hough transform algorithm is used to determine line features. Furthermore, if the Hough transform is applied to TBF data, better line detection can be achieved; this is known as Triangulation Hough based fusion (THF). Also, for the new line detected, TBF data are used to draw line segments. Here, in the combined algorithm, TBF results are used again to improve the overall performance and accuracy of the feature extraction and remove redundant features. Noting that the SONAR salient algorithm is better for detecting points and cylindrical shaped object, here, it has been used to determine point and cylindrical features. It is mentioned that the inside of oil storage tanks consists of cylindrical features, which can be detected with SONAR salient algorithm. Through the use of point-in-polygon detection (PIP), the proposed algorithm can separate the data into interior and exterior (boundary) groups. PIP is used to classify points as being inside or outside of a given polygon. For further details on PIP, the reader is refereed to Hormann and Agathos (2001). For example, the Hough transform and TBF algorithm can be used to reconstruct the environment's boundaries, while the SONAR salient algorithm is used to reconstruct the interior features. The separation of groups ensures that each part of the algorithm can be used on "preferred" features. The new fusion algorithm has been studied through simulations and experiments, and the results are reported in this chapter.

In the new fusion algorithm, the TBF, Hough Transform, and THF algorithms are used in parallel with the SONAR salient algorithm. The proposed fusion algorithm scheme is shown in Figure 3.1. Since some of the steps can be carried out in

parallel, there can be a dramatic decrease in computational time. The different steps are as follows: (1) input raw SONAR data to the fusion algorithm; (2) apply limit constraints to the SONAR data and accept only SONAR range data between 50 cm and 200 cm. The SONAR sensor (model HRLV-MAXSONAR-EZ) cannot be used to detect objects (targets) that are closer than 30 cm. Therefore, values lower than 30 cm are reported as 30 cm. Also, for this sensor, it is recommended to avoid data below 50 cm because of multiple reflections that occur when the SONAR sensor is close to an object. In addition, for distance above 200 cm, since the SONAR beam divergence is large and dispersed, the sensor cannot be used to detect objects above this range; (3) process the limit constraint results with the TBF algorithm as well as the SONAR salient feature algorithm; (4) process the TBF results with Hough transform and refer to the outcome as THF results; (5) process the THF results with a k-mean clustering algorithm; (6) find the cluster lines intersection; (7) integrate the intersection points and TBF results to find line segments and determine the boundaries of the environment; (8) combine the environment boundary information with SONAR salient points by using PIP to determine the interior points; (9) remove the interior points close to the environment boundary if the minimum distance between the point and the line segments is less than the threshold value; (10) cluster the interior results by using a k-mean clustering algorithm; and (11) complete the process with the clustered interior results and the line segment results. In the proposed fusion algorithm, the user has control over several parameters, which are included in Table 3.1.

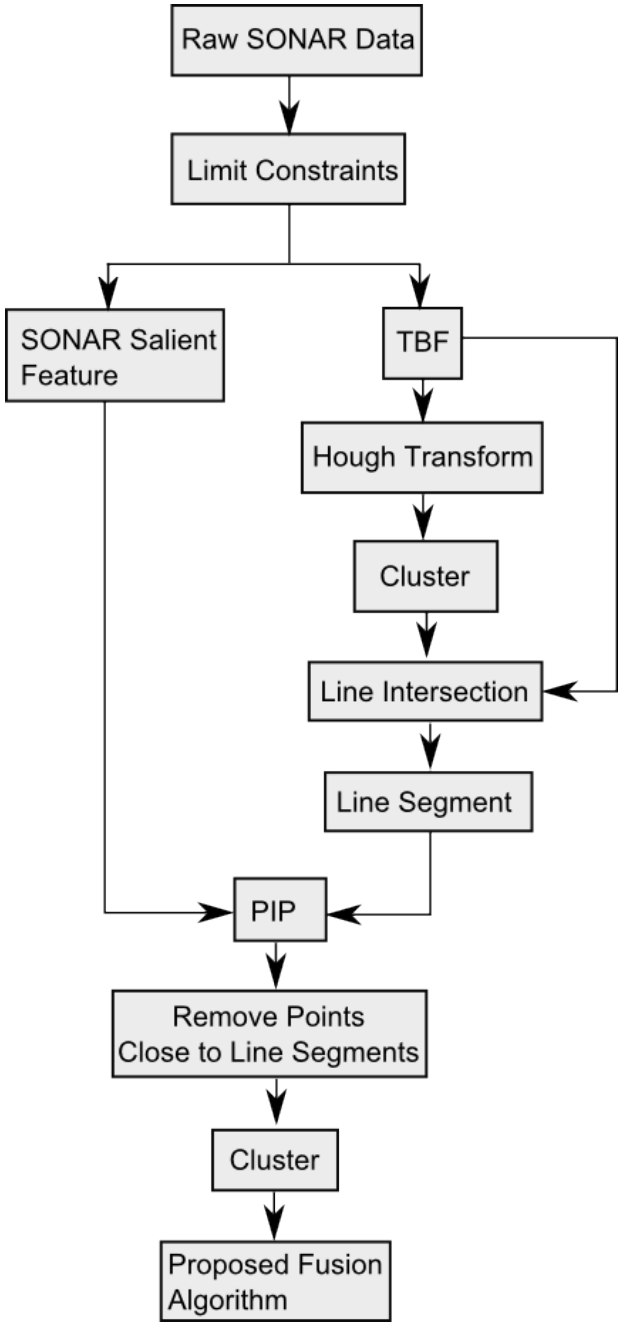


Figure 3.1: Proposed algorithm fusion developed by Ismail and Balachandran (2014, 2015a). The TBF algorithm is used for point features detection such as corners. The SONAR salient algorithm is also used for point features such as interior features. Hough Transform is used for line features such as edges.

Table 3.1: User specified parameters for the proposed fusion algorithm

Symbol	**Unit**	**Description**
r_{min}	cm	Minimum distance accepted from SONAR sensor range reading used in the limit constraint algorithm
r_{max}	cm	Maximum distance accepted from SONAR sensor range reading used in the limit constraint algorithm
n_t		Threshold value used in TBF algorithm
NOD		Number of divisions used in the Hough Transform algorithm
NOL		Number of lines used in the Hough Transform algorithm
NOCE		Number of clusters for the exterior features used in the k-mean clustering algorithm
p_{min}	cm	Minimum acceptable radius of the hypothetical circle used in the SONAR salient algorithm
p_{max}	cm	Maximum acceptable radius of the hypothetical circle used in the SONAR salient algorithm
PIL	%	Point-in-line refers to how many TBF points fall in each hypothetical line segment; if enough TBF points are available, accept the hypothetical line segment used in the line segment algorithm
NOCI		Number of clusters for the interior features used in the k-means clustering algorithm

3.2 Algorithm Simulation and Results

In this section, as a precursor to the experiments to follow, an L-Shaped closed environment and another closed environment are used to study the new fusion algorithm by using simulation data that were generated by using MATLAB software. Appropriate uncertainties are included with the raw SONAR data. The range and angular uncertainties for the chosen "HRLV-MAXSONAR-EZ" SONAR are 0.1 cm and 22.5^o, respectively. These values are obtained from the sensor data sheet. The variance of the range of SONAR sensor is equal to $(0.1 \text{ cm})^2$, and the variance of the bearing of SONAR sensor is equal to $(22.5^o)^2$. Noise was introduced to the range and bearing readings of the SONAR sensor simulations, in order to make the simulations more realistic. For example, if the SONAR range reading is 50 cm and range uncertainty is 0.1 cm, the new range value will be 50 cm $+randn() * 0.1$ cm, where $randn()$ is a MATLAB function that produces values from a normally distributed distribution with a mean 0 and a variance of 1. The SONAR bearing readings are similarly treated. In this chapter, uncertainties were not introduced to the SONAR sensor location, since, the goal is to test the performance of the proposed feature extraction algorithm for a perfect localization of the mobile vehicle.

3.2.1 Environment I: L-Shaped Environment

The L-Shaped environment shown in Figure 3.2, consists of 6 corners, 6 line segments, and one circular interior feature. A full 360^o SONAR scan with 1^o increments was performed at each of the 20 different locations inside this environ-

ment. The mobile vehicle comes to a complete stop before a full SONAR scan is performed. This is done, since the motor which rotates the SONAR array moves relatively slowly. In the envisaged future application of the oil tank floor inspections, for similar reasons, it is expected that the mobile vehicle will also stop for SONAR and other sensor measurements. In general, as more locations are visited by the mobile vehicle, better results are obtained with the proposed algorithm fusion, at the expense of increased computational time. Also, the visited locations need to be appropriately separated (i.e, SONAR readings should bot be grouped in one small area and so on). In the previous chapter, the mobile vehicle visited 15 locations in a similar environment, with the visited locations being close to one corner only. In this prior work, the corner could be detected, but no other features in the environment could be identified. Based on the prior experiments, it was ascertained that a separation distance in the range of 30 cm to 100 cm worked out well for the visited locations of the considered mobile vehicle platform. A separation distance below 30 cm increased the overall number of visits of the mobile vehicle and unnecessarily increased the computational time. On the other hand, a separation distance above 100 cm could result in the loss of important environment features, as the acceptable SONAR range data is between 50 cm and 200 cm. Through experiments and simulations for considered sensors and mobile platform, it was found that separation distances between 30 cm and 100 cm provided good results. Each mobile vehicle location in Figure 3.2 is represented by a different color and the scan from a certain location has the same color as shown in Figures 3.3a and 3.3b.

The raw SONAR data are shown in Figure 3.3a. First, the raw SONAR data

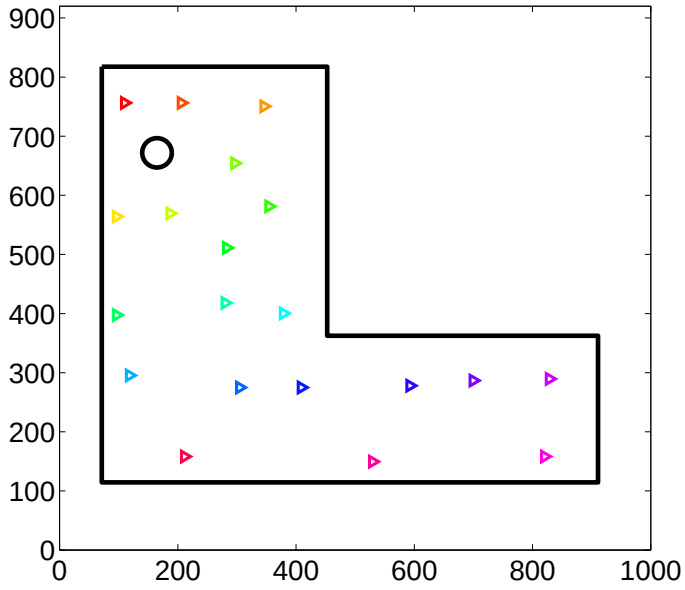


Figure 3.2: L-Shaped Environment. The triangles represent different mobile sensor platform positions. The circular landmark inside the environment has a radius of 25 cm.

are processed with the limit constraint algorithm. Data below 50 cm and above 200 cm have been removed, and the results of this processing are shown in Figure 3.3b. Next, the limit constraint data are processed with the TBF algorithm and studies are conducted for different threshold values. The threshold values have been arbitrarily selected to be 0, 6, and 50. By selecting a low threshold value for the TBF algorithm, it was ensured that no important features were removed. In the previous chapter, it was shown that by increasing the threshold to a value above 100, a considerable number of the important features were removed, whereas for a threshold below a value of 100, no important features were lost. this observation is consistent with the conclusion drawn by Wijk and Christensen (2000b); they used a low threshold value to avoid the removal of important features. In this chapter, the selected threshold values for TBF algorithm were less than or equal to 10 for the simulation and the experiments; with a 360^o SONAR scan at each location, no

important features were found to be removed. The obtained results are shown in Figure 3.4.

Next, the processed TBF data are sent for application of the Hough Transform. The resulting THF data are shown in Figure 3.5a. Subsequently, the THF data are clustered by using k-mean clustering as shown in Figure 3.5b. Also, the limit constraint data are sent to the SONAR salient algorithm. The SONAR salient algorithm results are shown in Figure 3.6. Next, the SONAR salient data are used in the PIP algorithm to sort the data into interior and exterior points. This can readily be implemented in MATLAB by using the function "inpolygon". Interior points near the boundary are removed if the distance between the point and the line segments is less than the chosen threshold value. Exterior points and interior points near a boundary are removed. Following that, the remaining interior points are clustered by using a k-mean clustering algorithm. The inner group cluster determines the interior features in the environment. Next, a line is drawn between intersection points if sufficient TBF data (85%) are present between the intersection points. Finally the line segments and the cluster of the inner group are put together. The results obtained with the new fusion algorithm are shown in Figure 3.7.

The simulation results are compared to the actual data for features such as corners, interior features and lines in Tables 3.2, 3.3, and 3.4. First, the corners' data obtained from the simulations are compared with the actual data. All corners are detected with the fusion algorithm and the maximum percentage errors in X and Y coordinate values are found to be below 5% in X and 8% in Y as shown in Table 3.2. Also, the fusion algorithm is able to detect the interior feature. The percentage

Table 3.2: Corner coordinates for L-Shaped environment and absolute percentage errors between estimates and actual values.

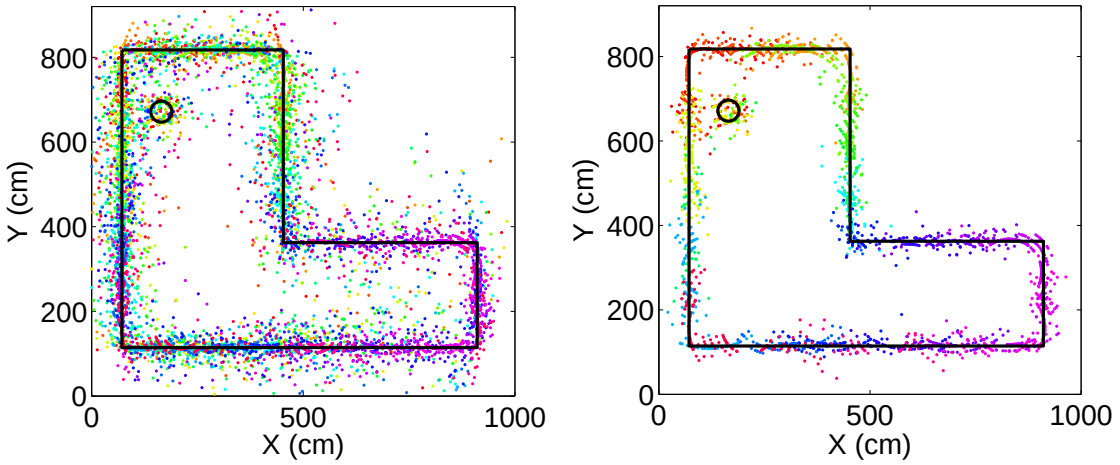
Corner #	**Actual Data** X**(cm),** Y**(cm)**	**Simulation Results** X**(cm),** Y**(cm)**	**% Error** X**(%),** Y**(%)**
1	$71.45, 817.64$	$74.62, 789.87$	$4.43, 3.40$
2	$452.74, 817.64$	$441.18, 754.27$	$2.55, 7.75$
3	$452.74, 362.47$	$449.98, 361.40$	$0.61, 0.30$
4	$910.85, 362.47$	$910.14, 357.59$	$0.08, 1.35$
5	$910.85, 114.46$	$910.16, 116.81$	$0.08, 2.05$
6	$71.45, 114.46$	$70.01, 112.23$	$2.02, 1.95$

Table 3.3: Interior feature data for L-Shaped environment and absolute percentage errors between estimates and actual values.

Interior Feature #	**Actual Data** X**(cm),** Y**(cm), Radius(cm)**	**Simulation Results** X**(cm),** Y**(cm), Radius(cm)**	**% Error** X**(%),** Y**(%), Radius%**
1	$164.71, 671.75,$ 25	$169.10, 668.76,$ 15.63	$2.66, 0.44,$ 37.48

Table 3.4: Line data for L-Shaped environment and absolute percentage errors between estimates and actual values.

Line #	**Actual Data** ρ**(cm),** θ**(degree)**	**Simulation Results** ρ**(cm),** θ**(degree)**	**% Error** ρ**(%),** θ**(%)**
1	$817.64, 90.00$	$793.17, 84.52$	$2.99, 6.09$
2	$452.74, 360.00$	$458.93, 361.39$	$1.37, 0.38$
3	$362.47, 90.00$	$366.64, 89.40$	$1.15, 0.67$
4	$910.85, 360.00$	$909.86, 360.00$	$0.11, 0.00$
5	$114.46, 90.00$	$112.53, 90.25$	$1.69, 0.27$
6	$71.45, 360.00$	$69.79, 359.68$	$2.32, 0.09$



(a) Raw SONAR data

(b) Results after application of limit constraints

Figure 3.3: The actual L-shaped environment is overlaid on top of the SONAR data for comparison purposes. After the application of limit constraints, data below 50 cm and above 200 cm have been removed.

error between simulation and actual data for the center of the interior feature in X and Y is found to be below 3%. The radius error is about 38% as shown in Table 3.3. This relatively high error in the interior feature radius is suspected to be due to the averaged data. The choice of a higher acceptable range might help alleviate this error. Finally, the errors made in line data from the simulations are found to be less than 3% and 7% in ρ and θ, respectively, as presented in Table 3.4.

3.2.2 Environment II: Second Closed Environment

The second environment consists of 8 corners, 8 line segments, and one circular interior feature. In this environment, the mobile vehicle was taken to the 25 different locations shown in Figure 3.8 and a full SONAR scan was performed at each location with 1^o angle increment. Each mobile vehicle location in Figure 3.8 is represented by a different color and the scan from a certain location has the same color as shown in

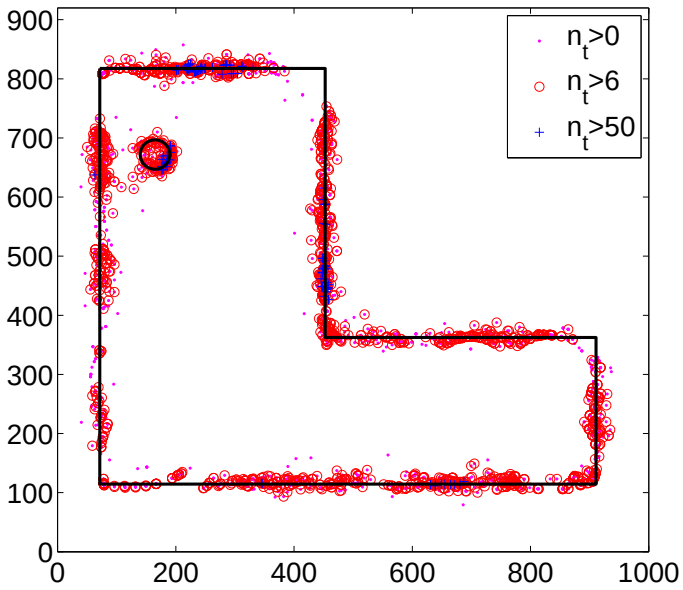
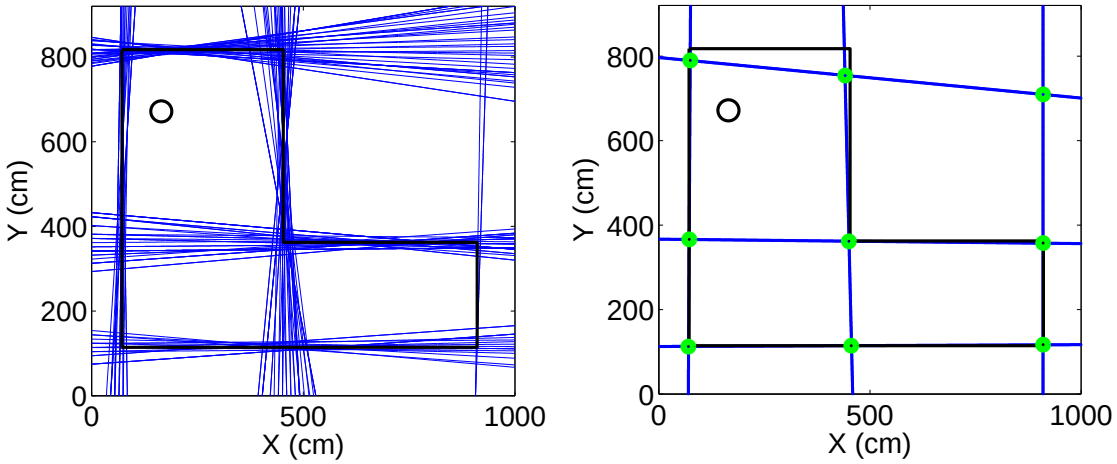


Figure 3.4: TBF results for the L-Shaped environment with different threshold values.



(a) THF results

(b) K-mean clustering results

Figure 3.5: THF results for $n_t > 0$ for the L-Shaped environment. The number of bins and lines used are 200 and 140, respectively. The intersections of the clustered lines have been calculated and marked.

Figures 3.9a and 3.9b. Feature extraction studies similar to those carried out for the first environment were performed. The associated raw SONAR data for this case are shown in Figure 3.9a. Next, the raw SONAR data are processed by using the limit constraint algorithm and the obtained results are shown in Figure 3.9b. For the

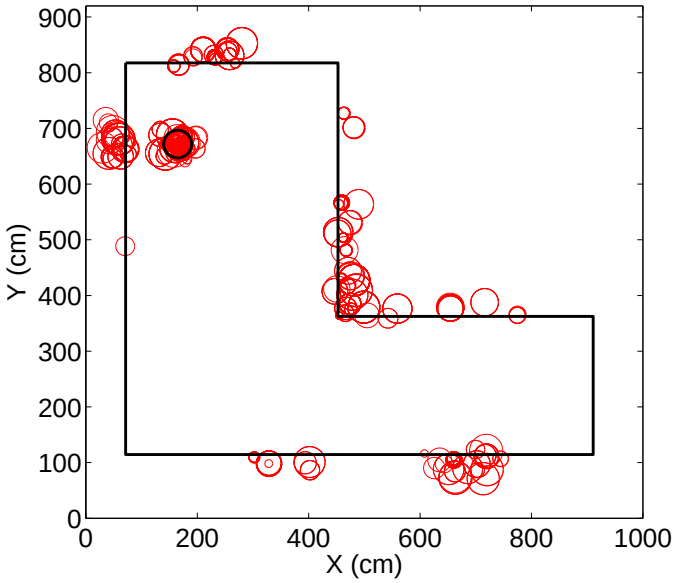


Figure 3.6: SONAR salient feature extraction result for the L-Shaped environment for $p_{min} = 3$ cm and $p_{max} = 30$ cm, before the classification.

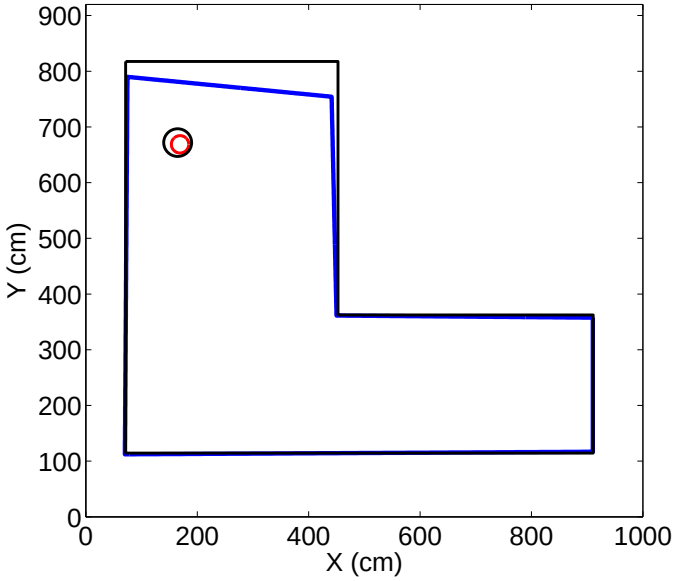


Figure 3.7: Fusion algorithm results for chosen L-shaped environment.

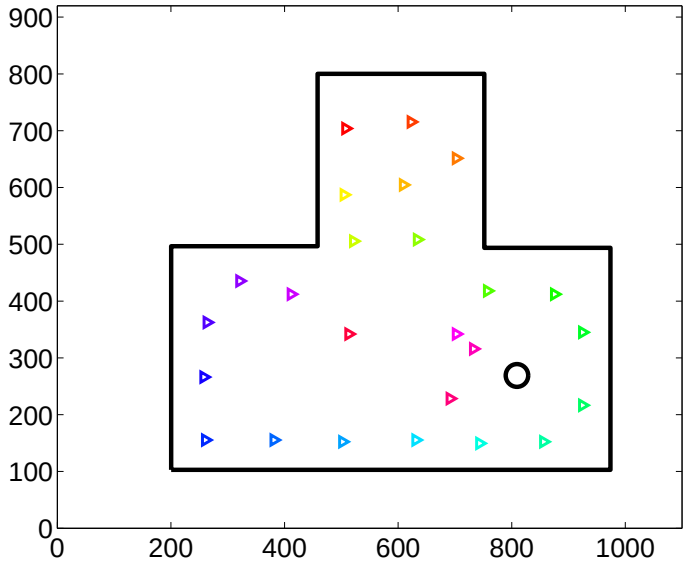


Figure 3.8: Second closed environment. The triangles represent different mobile vehicle positions. The circular landmark inside the environment has a radius of 20 cm.

limit constraint data, data below 50 cm and above 200 cm have been removed. The data obtained after application of the limit constraint algorithm are used as inputs to the TBF algorithm, with the selected threshold values being 0, 6, and 50 and the results obtained are shown in Figure 3.10. Next, the TBF results are processed by using the Hough Transform, where the number of bins and lines used are 200 and 140, and the obtained outcome is known as THF results and shown in Figure 3.11a for $n_t > 0$. As a next step, the THF results are clustered by using k-mean clustering and the intersections between the clustered lines are marked as shown in Figure 3.11b. In addition, the limit constraint results are input to the SONAR salient algorithm and the resulting outcome is shown in Figure 3.12. Finally, the results obtained with the fusion algorithm are shown in Figure 3.13.

Next, the comparisons made between the estimated features and the actual data value for the second environment considered are shown in Tables 3.5, 3.6, and

Table 3.5: Corner coordinates for the second environment and absolute percentage errors between estimates and actual values.

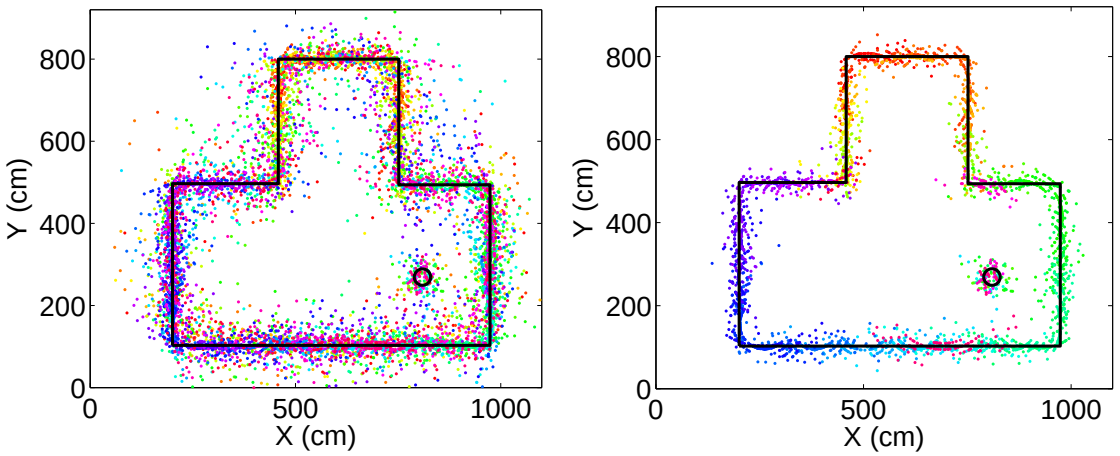
Corner #	**Actual Data** X**(cm),** Y**(cm)**	**Simulation Results** X**(cm),** Y**(cm)**	**% Error** X**(%),** Y**(%)**
1	$200.37, 102.79$	$205.59, 103.05$	$2.61, 0.26$
2	$200.37, 496.68$	$203.09, 491.67$	$1.36, 1.01$
3	$458.23, 496.68$	$452.43, 498.76$	$1.27, 0.42$
4	$458.23, 800.13$	$467.82, 798.05$	$2.09, 0.26$
5	$751.75, 800.13$	$740.91, 796.59$	$1.44, 0.44$
6	$751.75, 493.77$	$763.44, 505.35$	$1.56, 2.35$
7	$973.94, 493.77$	$977.56, 510.73$	$0.37, 3.43$
8	$973.94, 102.79$	$969.76, 100.97$	$0.43, 1.77$

Table 3.6: Interior feature data for the second environment and absolute percentage errors between estimates and actual values.

Interior Feature #	**Actual Data** X**(cm),** Y**(cm), Radius(cm)**	**Simulation Results** X**(cm),** Y**(cm), Radius(cm)**	**% Error** X**(%),** Y**(%), Radius%**
1	$809.35, 269.10,$ 20	$809.05, 265.88,$ 14.09	$0.04, 1.20,$ 29.55

Table 3.7: Line data for the second environment and absolute percentage errors between estimates and actual values.

Line #	**Actual Data** ρ**(cm),** θ**(degree)**	**Simulation Results** ρ**(cm),** θ**(degree)**	**% Error** ρ**(%),** θ**(%)**
1	$200.37, 360.00$	$206.59, 360.34$	$3.10, 0.10$
2	$751.75, 360.00$	$802.14, 364.62$	$6.70, 1.28$
3	$496.68, 90.00$	$487.10, 91.40$	$1.93, 1.56$
4	$102.79, 90.00$	$103.00, 89.89$	$0.20, 0.12$
5	$458.23, 360.00$	$427.94, 357.19$	$6.61, 0.78$
6	$800.13, 90.00$	$798.97, 89.81$	$0.14, 0.22$
7	$973.94, 360.00$	$968.10, 358.94$	$0.60, 0.29$



(a) Raw SONAR data (b) Limit constraint data

Figure 3.9: The actual closed environment is overlaid on top of the SONAR data for comparison purposes. For the limit constraint data, data below 50 cm and above 200 cm have been removed.

3.7. First, the simulation results for the corners are compared to the actual corner locations and the maximum percentage errors in the X location is found to be below 3% and in the Y location is found to be below 4% as shown in Table 3.5. When the interior feature results from the simulations are compared with the actual values for the interior feature, it is found that the maximum percentage error for the interior feature's center is below 1% for the X value and below 2% for the Y value. The error in the radius estimation is found to be below 30% as shown in Table 3.6. Finally, the simulation results for the lines are compared to the actual line values. The maximum percentage errors for the lines is found to be below 7% in ρ and below 2% in θ as presented in Table 3.7.

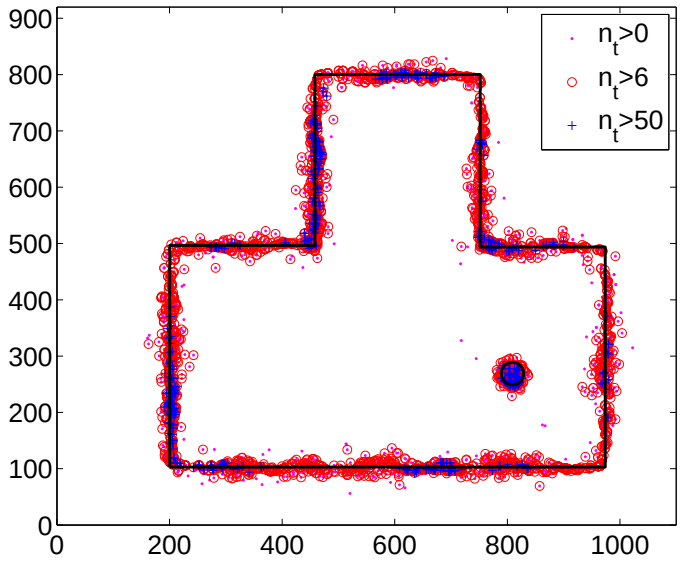
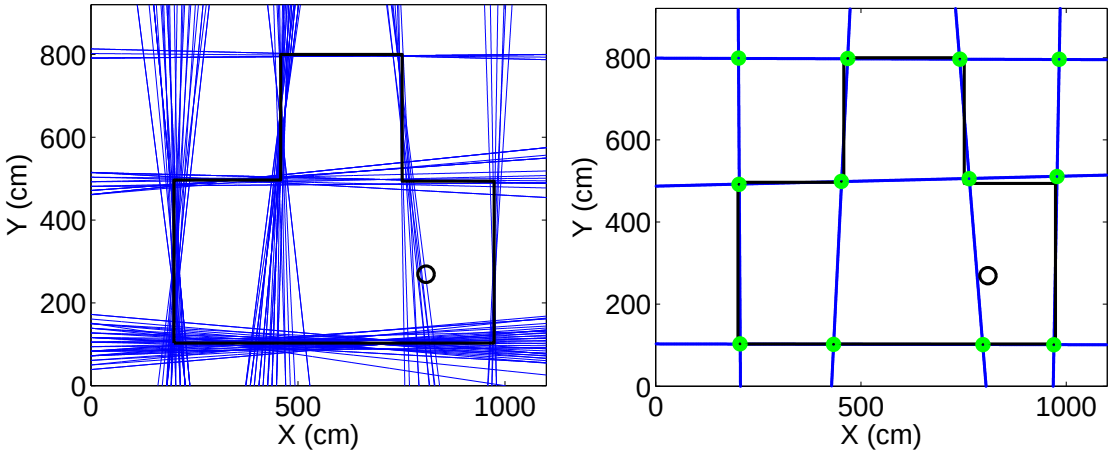


Figure 3.10: TBF results for the second environment with different threshold values.



(a) THF results

(b) Kmeans clustering results

Figure 3.11: THF results for $n_t > 0$ for the second environment. The number of bins and lines used are 200 and 140, respectively. The intersections of the clustered lines have been calculated and marked.

3.3 Experimental Arrangement

In this section, data are gathered in two different environments using the mobile vehicle described in the previous section. The two different environments are shown in Figure 3.14a and 3.14b. There are no interior features in the first closed environment. In the first environment, the mobile vehicle is moved to 24

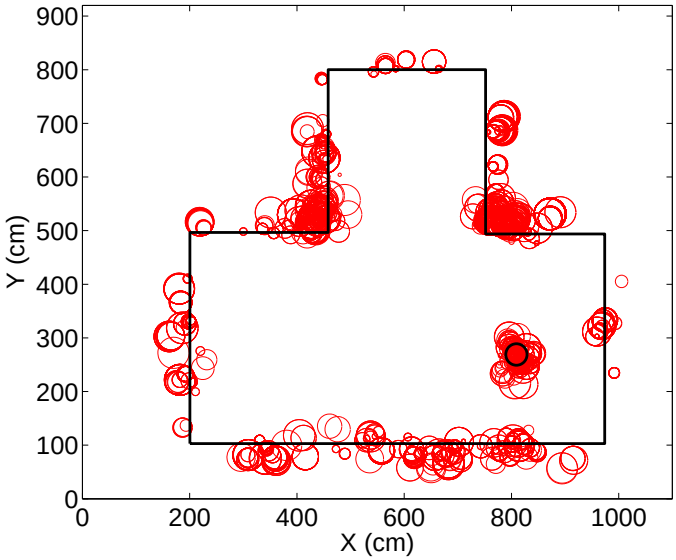


Figure 3.12: SONAR Salient feature extraction result for the second environment for $p_{min} = 3$ cm and $p_{max} = 30$ cm, before classification.

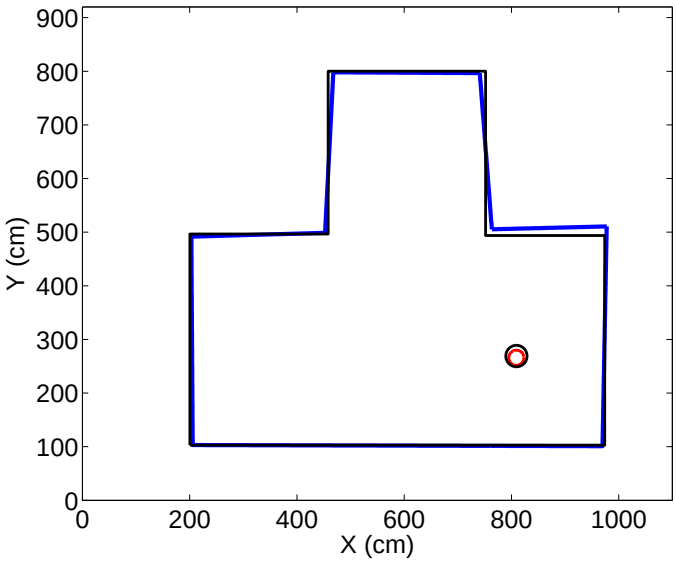


Figure 3.13: Fusion algorithm results for second environment.

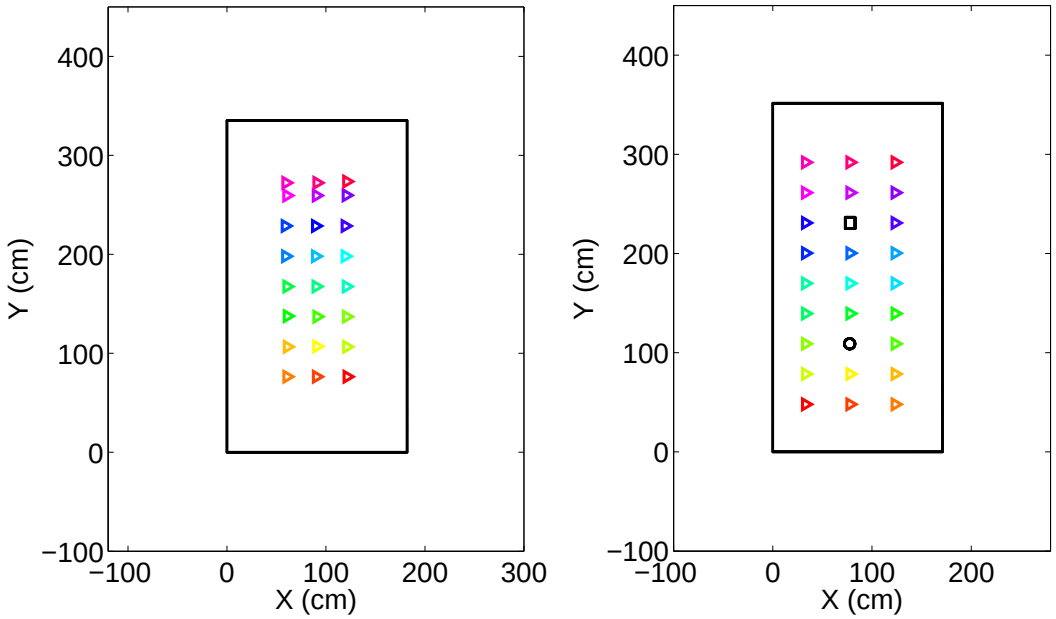
different locations, and a full scan is performed at each location. The different vehicle locations are shown in Figure 3.15a, where each true position of the vehicle has been marked by a triangle. In the second environment, the mobile vehicle is moved to 25 different locations and a full scan is performed at each location. Two interior features are included in the second environment. In Figure 3.15b, each true position of the mobile vehicle has been marked by a triangle and the interior features are also outlined.

(a) No interior feature (b) 2 interior features

Figure 3.14: Experimental arrangement with no interior feature following Ismail and Balachandran (2013) and with 2 interior features following Ismail and Balachandran (2014, 2015a).

3.4 Experimental Results

In this section, the feature extraction results for the two considered environments are presented. SONAR data are processed in both cases by using the new fusion algorithm presented in this chapter. First, the environment with no interior feature is considered.



(a) No interior feature

(b) 2 interior features

Figure 3.15: Different locations of mobile platform in the environment without and with 2 interior features. Triangles are used to mark the locations of the mobile vehicle, and the interior features are outlined with a square shape and a circle shape.

3.4.1 Environment I: Rectangular Environment with no Internal Features

The raw SONAR data is shown in Figure 3.16. As in the simulations, first, the raw SONAR data is processed by using the limit constraint algorithm, and data below 50 cm and above 200 cm are removed. The outcome is shown in Figure 3.17. In the next step, the limit constraint results are processed by using the TBF algorithm, for which the value of 10 has been selected as the threshold. For studies on the effects of this threshold value, the reader is referred to chapter 2 Ismail and Balachandran (2013). The TBF result is shown in Figure 3.18. Subsequently, the processed TBF data is input to the Hough Transform, and the resulting THF

Table 3.8: Corner coordinates for the environment with no interior features and absolute errors between estimates and actual values.

Corner #	**Actual Data** X**(cm),** Y**(cm)**	**Exp. Results** X**(cm),** Y**(cm)**	**|Actual-Exp.|** X**(cm),** Y**(cm)**
1	$182.00, 0.00$	$172.59, 3.01$	$9.41, 3.01$
2	$0.00, 0.00$	$1.00, 4.27$	$1.00, 4.27$
3	$0.00, 335.10$	$3.07, 332.14$	$3.07, 2.96$
4	$182.00, 335.10$	$177.90, 329.22$	$4.10, 5.88$

data is shown in Figure 3.19a. Next, the THF data are clustered by using k-mean clustering and the line intersections are marked as shown in Figure 3.19b. The limit constraint results are also processed by using the SONAR salient algorithm. The obtained results of this processing are shown in Figure 3.20.

Next, the SONAR salient results are separated into interior and exterior groups by using PIP analysis. In this case, as there are no interior points, the k-means clustering is not performed. After that, by using the TBF data, lines are drawn between the intersection points. This is carried out if there are sufficient TBF data (70%) between the considered intersection points. Finally, the line segments and the clusters of the inner groups are put together and the resulting outcome of the fusion algorithm is shown in Figure 3.21. Through the studies, it was found that to improve the results of the fusion algorithm, the mobile vehicle positions needed to include a number of locations, which are not close to each other, as also previously noted in this chapter. In addition, the raw SONAR data were noisy. With the aids of limit constraints and the TBF algorithm, the data quality was enhanced for processing with the Hough transform. Overall, the most significant features of the environment were extracted by using the fusion algorithm.

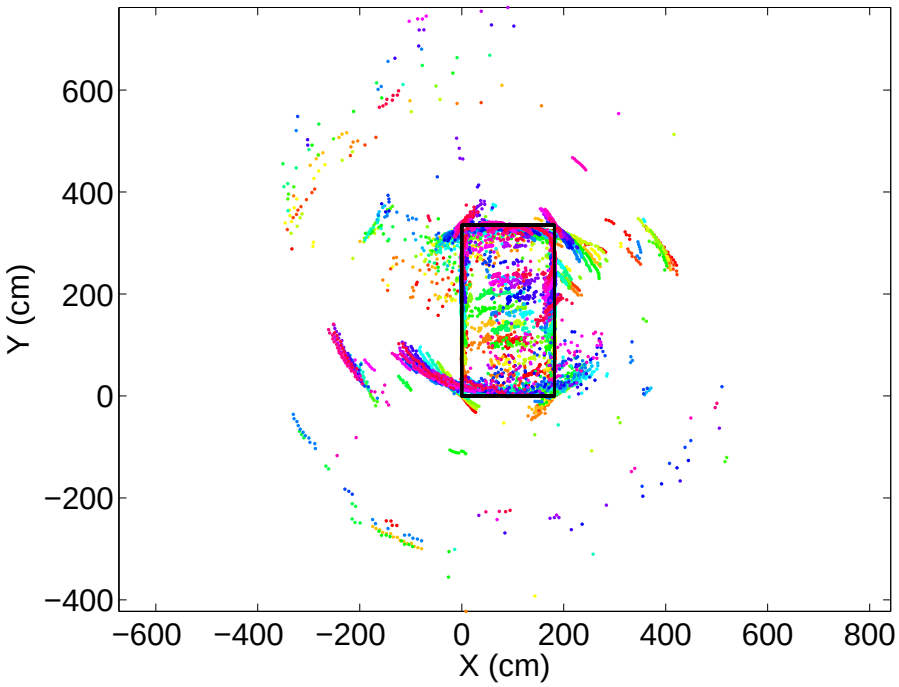


Figure 3.16: Raw SONAR data for experimental environment without interior features. The actual environment with no interior features is placed over the SONAR data for comparison purpose only.

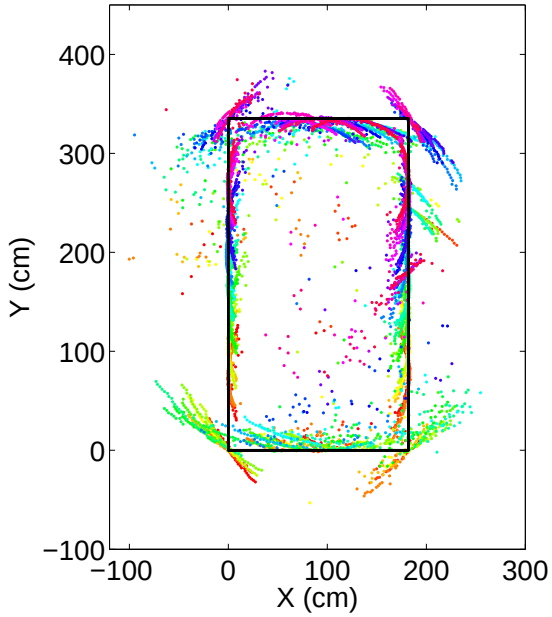


Figure 3.17: Results after application of limit constraints, data below 50 cm and above 200 cm have been removed.

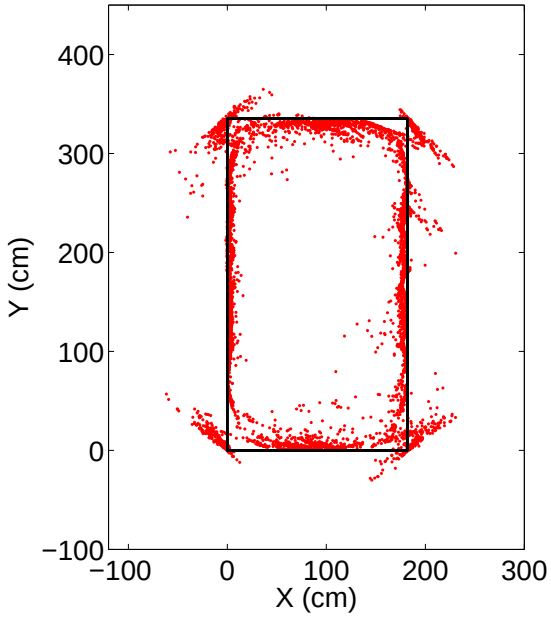
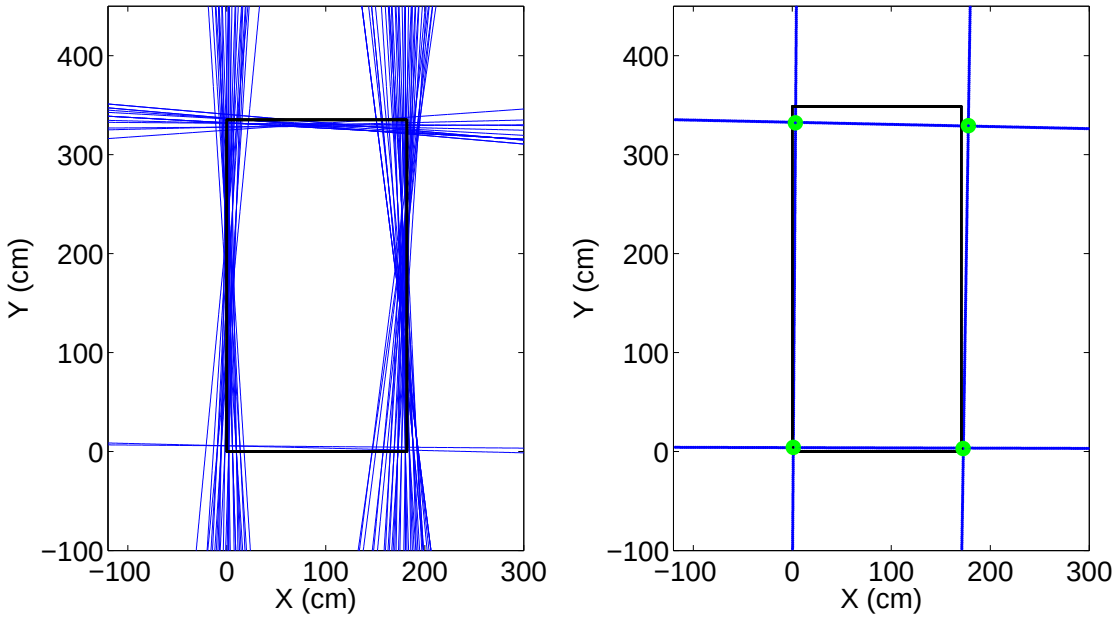


Figure 3.18: TBF results $n_t > 10$ for environment with no interior features.

Table 3.9: Line data for environment with no interior features and absolute errors between estimates and actual values.

Line #	**Actual Data** ρ**(cm),** θ**(degree)**	**Exp. Results** ρ**(cm),** θ**(degree)**	**|Actual-Exp.|** ρ**(cm),** θ**(degree)**
1	$335.10, 90.00$	$332.62, 88.76$	$2.48, 1.24$
2	$182.00, 360.00$	$172.78, 359.12$	$9.22, 0.88$
3	$0.00, 90.00$	$3.97, 89.85$	$3.97, 0.15$
4	$0.00, 360.00$	$0.99, 359.60$	$0.99, 0.40$



(a) THF results

(b) Kmeans clustering results

Figure 3.19: THF result for $n_t > 10$ for the environment with no interior features. The number of bins and lines used were 200 and 50, respectively. The intersections of the clustered lines have been calculated and marked as circles.

A comparison of the features extracted from the experimental data with the actual data is made in Tables 3.8 and 3.9. It is found that with the new fusion algorithm, one can detect all corners and the maximum absolute errors in X and Y values are below 10 cm and 6 cm, respectively. For the line comparisons, it is found that the maximum absolute errors in ρ and θ are below 10 cm and 2^o, respectively.

3.4.2 Environment II: Rectangular Environment with Internal Features

In another experimental study, the author examined the performance of the fusion algorithm to extract features for the environment with two interior features, shown in Figure 3.14b. Feature extraction studies similar to those carried out for

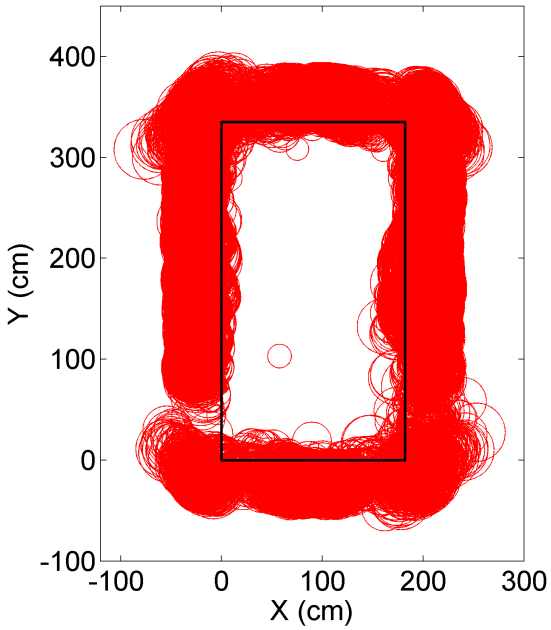


Figure 3.20: SONAR salient feature extraction result for the environment with no interior feature, accepted radius range between 3 cm and 30 cm, before the classification.

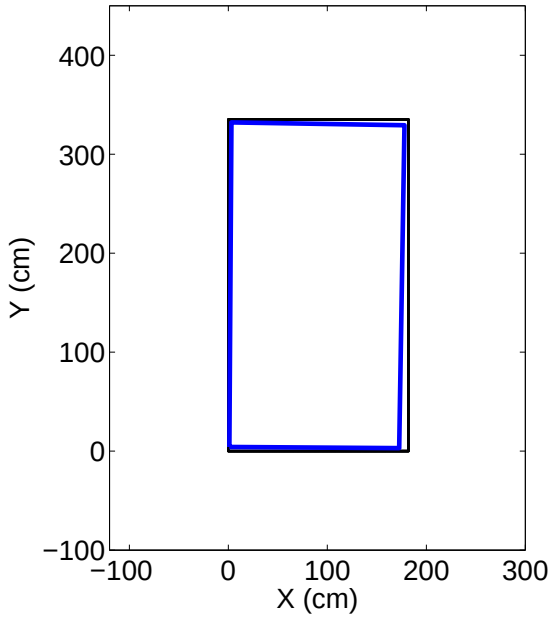


Figure 3.21: Fusion algorithm results for the environment with no interior features.

the empty environment were performed. First, the raw SONAR data for this case are shown in Figure 3.22. This raw SONAR data is next processed using the limit constraint algorithm and the results are shown in Figure 3.23. In the following steps, the data obtained after application of limit constraints are processed by using the TBF algorithm and studies were performed for different threshold values. The selected threshold value is 10. The TBF results are shown in Figure 3.24. Next, the TBF data are processed by using the Hough Transform, and the number of bins and lines used are 200 and 40, respectively as shown in Figure 3.25a. The resulting data are known as THF results. Subsequently, these results are clustered using the k-mean clustering and the determined line intersections are marked as shown in Figure 3.25b. Also, the data obtained after application of limit constraints are sent to the SONAR salient algorithm, with the accepted radius range being between 3 cm and 15 cm and the obtained results are shown in Figure 3.26. The SONAR salient results are processed by using PIP analysis to separate interior features from exterior features. After that, using the TBF data, lines are drawn between the intersection points. This is carried out if sufficient TBF data (70%) is there between the intersection points. Finally, the line segments and the clusters of the inner groups are combined. The outcome of the new fusion algorithm is presented in Figure 3.27. It is seen that the new fusion algorithm is able to capture the most significant environment features, which are the corners, lines, and two interior features.

A comparative discussion between the features extracted by using the fusion algorithm and the actual values is provided in Table 3.10, 3.11, and 3.12. For the

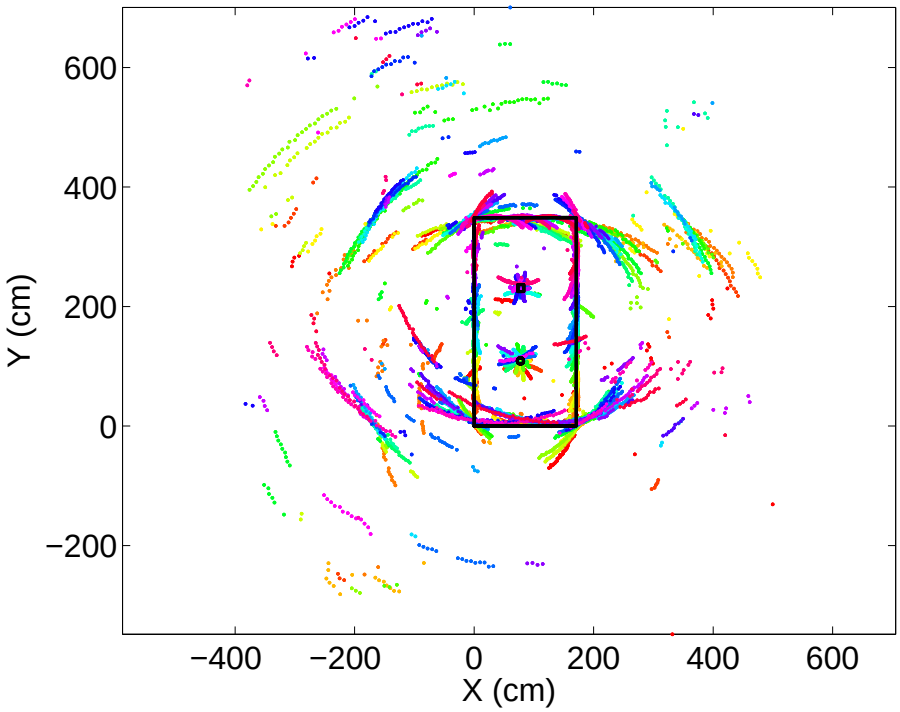


Figure 3.22: Raw SONAR data, the actual environment with two interior features is overlaid on top of SONAR data for comparison purposes, Ismail and Balachandran (2015a).

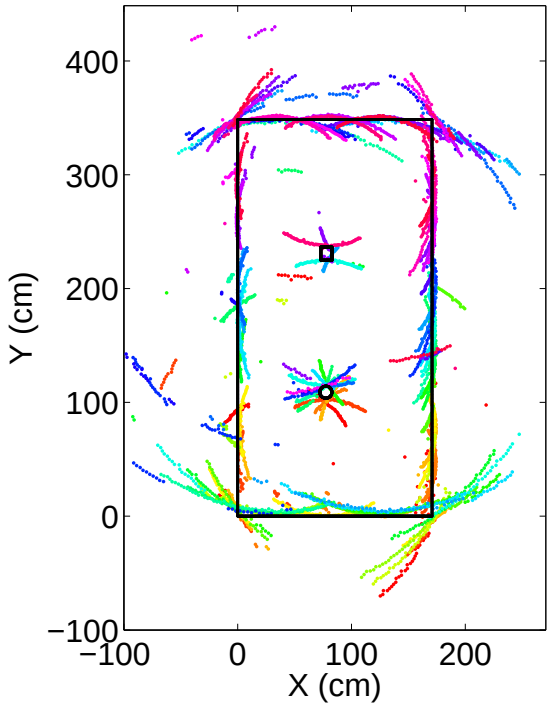


Figure 3.23: Results obtained after application of limit constraints, data below 50 cm and above 200 cm have been removed.

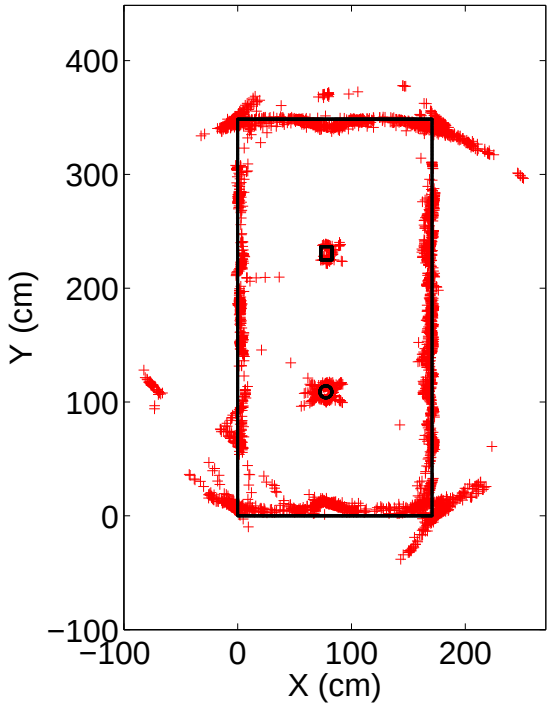


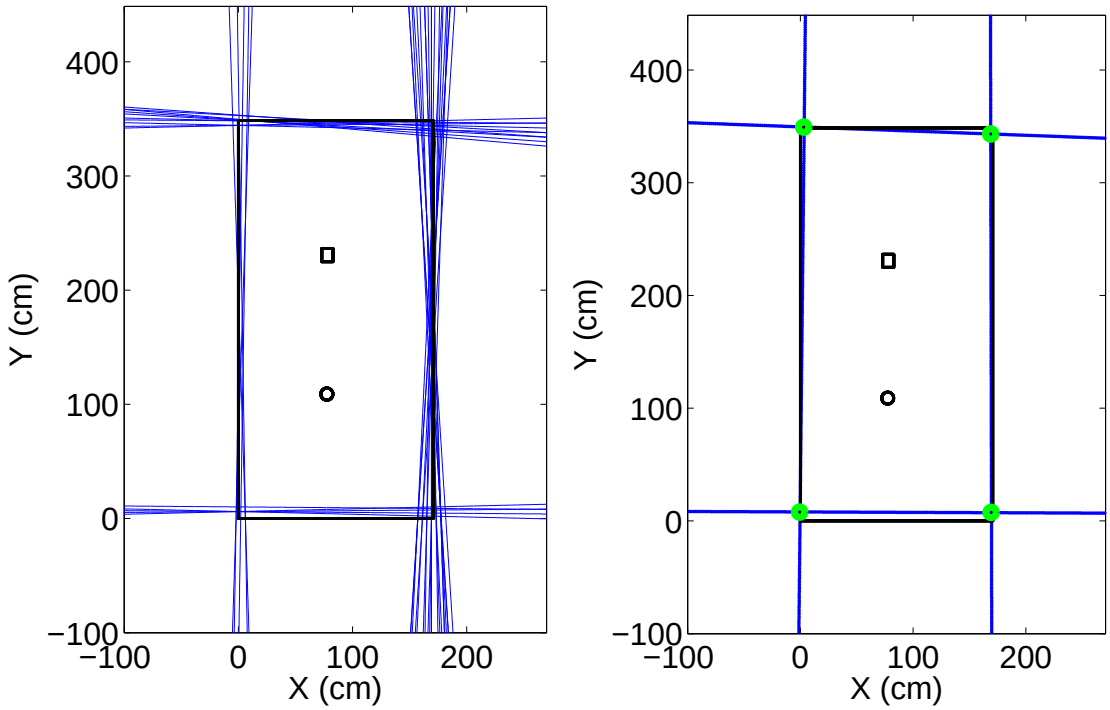
Figure 3.24: TBF results $n_t > 10$ for environment with two interior features.

Table 3.10: Corner coordinates for the environment with 2 interior features and absolute errors between estimates and actual values.

Corner #	**Actual Data** X**(cm),** Y**(cm)**	**Exp. Results** X**(cm),** Y**(cm)**	**|Actual-Exp.|** X**(cm),** Y**(cm)**
1	$171.00, 351.50$	$168.85, 343.30$	$2.15, 8.20$
2	$171.00, 0.00$	$169.18, 7.57$	$1.82, 7.57$
3	$0.00, 0.00$	$-0.65, 7.90$	$0.65, 7.90$
4	$0.00, 351.50$	$2.96, 349.41$	$2.96, 2.09$

Table 3.11: Line data for environment with 2 interior features and absolute errors between estimates and actual values.

Line #	**Actual Data** ρ**(cm),** θ**(degree)**	**Exp. Results** ρ**(cm),** θ**(degree)**	**|Actual-Exp.|** ρ**(cm),** θ**(degree)**
1	$0.00, 90.00$	$8.00, 89.77$	$8.00, 0.23$
2	$0.00, 360.00$	$-0.29, 359.40$	$0.29, 0.60$
3	$351.50, 90.00$	$349.37, 87.87$	$2.13, 2.13$
4	$171.00, 360.00$	$169.55, 360.11$	$1.45, 0.11$



(a) THF results

(b) K-mean clustering results

Figure 3.25: THF results for $n_t > 10$ for environment with two interior features. The number of bins and lines used are 200 and 40, respectively. The intersections of the clustered lines are determined and marked.

experimental environment with interior features, with the new algorithm, the author is able to detect all corners and the maximum absolute differences in X and Y values are found to be below 3 cm and 8 cm, respectively as shown in Table 3.10. For the line comparisons, the maximum absolute differences in ρ and θ are found to be below 8 cm and 3^o, respectively as shown in Table 3.11. The maximum percentage error in determining an interior feature is below 5% for the X value, below 2% for the Y value, and below 35.00% for the radius value as presented in Table 3.12. As expected, in the SONAR salient algorithm processing, although the location of the square cross-sectioned feature was picked up, the square cross-sectioned feature was picked up as a feature with circular cross-section. In addition, there was a 34.17%

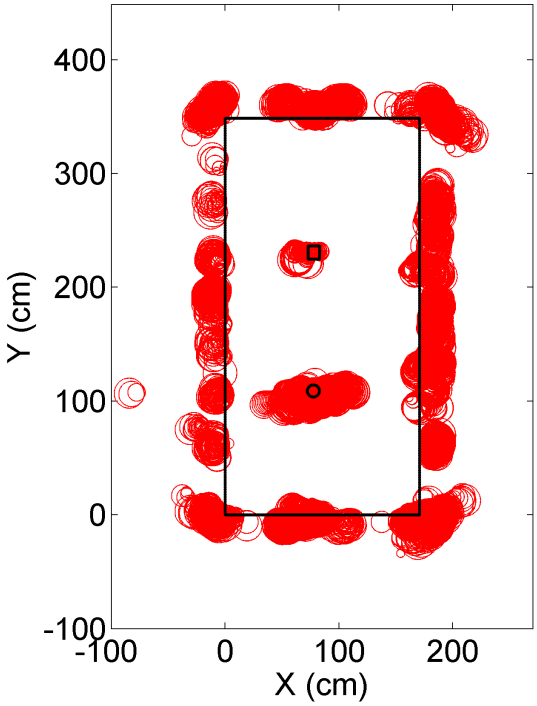


Figure 3.26: SONAR salient feature extraction result for environment with two interior features, accepted radius range between 3 cm and 15 cm, before the classification.

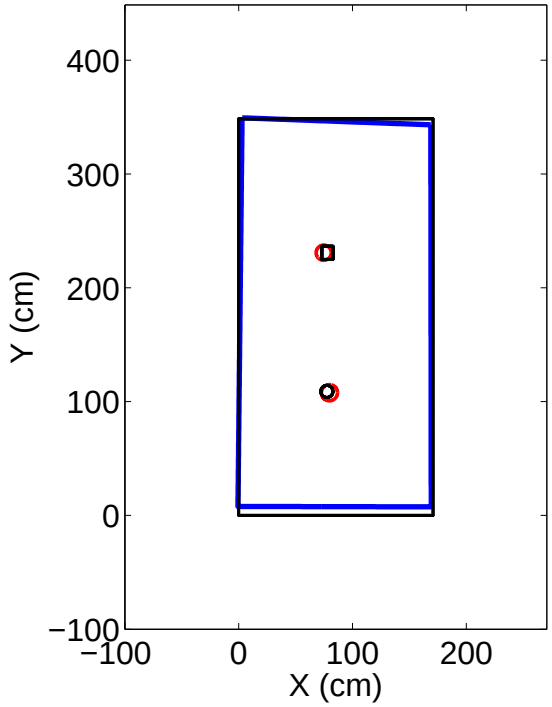


Figure 3.27: Results of fusion algorithm for experimental environment with two interior features.

Table 3.12: Interior features data for environment with 2 interior features and absolute percentage errors between estimates and actual values.

Interior Feature #	**Actual Data X(cm), Y(cm), Radius(cm)**	**Exp. Results X(cm), Y(cm), Radius(cm)**	**% Error X(%), Y(%), Radius%**
1	77.95, 230.80, N/A	74.72, 230.90, 6.67	4.15, 0.04, N/A
2	77.50, 109.00, 5.30	79.75, 107.88, 7.11	2.90, 1.03, 34.17

error in the determined radius value.

3.5 Summary

Through numerical studies and experiments, algorithms for extracting features from SONAR data have been examined. To take advantage of some of the attractive features of existing algorithms, a fusion algorithm has been developed and shown to capture all of the significant features in the environments studied. The obtained features were found to be unique with no repeated characteristics or redundancy. Through the numerical studies, it was learned that the SONAR Salient algorithm can be effective in locating interior features but considerable errors do occur in the radius values of the determined features. The experimental results also support the effectiveness of the proposed fusion algorithm for extracting features in closed environments without and with interior features. The choice of a high enough number of locations and locations not close to each other were found to be important for enhancing feature extraction quality. In the next chapters, the fusion algorithm will be used in an an Extended Kalman Filter (EKF)-SLAM scheme. This integration is expected to help the EKF-SLAM in terms of speed, as only the pertinent features

data result from the fusion algorithm are used.

Chapter 4

Vehicle Pose Estimation

Mobile vehicle pose estimation can be performed with various sensors, for example, encoder, gyroscope, accelerometer, magnometer, digital compass, and GPS sensor. A harsh environment such as the interior of an oil tank or oil pipeline limits the type of sensors one can use as previously noted. Keeping this in mind, a sensor combination of encoder and gyroscope is investigated next.

4.1 Motion Sensing

The encoder is one of the most used sensors for vehicle pose estimation (Borenstein and Feng, 1996b; Chenavier and Crowley, 1992; Dinçay, 2010; Wang, 1988; Kleeman, 1995; Goel, Roumeliotis, and Sukhatme, 1999; Tsumura, Fujiwara, Shirakawa, and Hashimoto, 1981; Borenstein and Koren, 1991; Cox, 1991). These sensors suffer from two kinds of errors, namely, systematic and non-systematic errors. The systematic errors are caused by unequal wheel diameters, measurement errors in wheel baseline and wheel radius, wheel misalignment, encoder resolution, and encoder sampling rate. The non-systematic errors are caused by wheel slippage, uneven floor, and unexpected obstacles in the mobile vehicle path (Dinçay, 2010). Most of the systematic error can be removed through calibration. Different calibration techniques include uni-directional square path, bi-directional square path

(Borenstein and Feng, 1996b), and 'figure-eight' path (Tsumura *et al.*, 1981; Borenstein and Koren, 1991; Cox, 1991). For a mobile vehicle with a zero turn radius, lateral and longitudinal slippage are introduced, which cause encoder based vehicle pose estimation to fail.

Researchers have started to integrate different sensors with encoders to improve vehicle pose estimation during special cases such as wheel slippage. After encoders, a popular sensor used in mobile vehicles is the gyroscope. This sensor, which can be used to measure the angular rotation of the mobile vehicle, is not affected by wheel slippages. However, gyroscope data suffer from a bias; researchers have came up with different error models to account for this bias (Barshan and Durrant-Whyte, 1995; Kim, Chung, Youm, and Oht, 1999; Chung, Ojeda, and Borenstein, 2001; Borenstein and Feng, 1996a; Ji, Wang, Xu, Shi, and Xia, 2006). Kim *et al.* considered a real time gyroscope error model since a gyroscope can be affected by temperature changes (Kim *et al.*, 1999). In addition to considering the error models, Chung *et al.*, calibrated the gyroscope with a known angular rate device to account for nonlinearity (Chung *et al.*, 2001). Borenstein and Feng integrated data from an encoder as well as a gyroscope and only use gyroscope data if a large change in angle between gyroscope and encoder is observed. This technique is called *gyrodometry* (Borenstein and Feng, 1996a). In another direction, Ji *et al.* carried out signal processing with gyroscope data, by using tools such as improved median filter and wavelet analysis to determine the gyroscope drift (Ji *et al.*, 2006). Zunaidi *et al.* used fuzzy logic to integrate data from gyroscope, encoder, and accelerometer sensors (Zunaidi, Kato, Nomura, and Matsui, 2006).

4.1.1 Encoder

To estimate the mobile vehicle position, encoders are attached to mobile vehicle wheels to measure their respective rotations. Since, the radius of the wheel is known, one can determine the distance each wheel travels. The travel distances of the front left and back left wheels are averaged to obtain the averaged travel distance for the left wheels. Also, the travel distances of the front and back right wheels are averaged to obtain the averaged travel distance for the right wheels. The travel distance is averaged on each side, since the control input is the same for each side. To calculate the change in travel distance and orientation of the mobile vehicle, equations (4.1) are used. To update the mobile vehicle position incrementally, equations (4.2) are used. The variables used in equations (4.1), and (4.2) are explained in Table 4.1. Equations (4.1), and (4.2) are used in most differential wheel drive mobile vehicles to estimate the mobile vehicle position using only encoders. For more details about the derivation of equations (4.1) and (4.2), the reader is referred to earlier work of Wang (1988). Some source of uncertainty which prevents equations (4.2) to model the mobile vehicle motions accurately are the following: *i*) measurement resolution of the encoder; *ii*) different wheel diameters; *iii*) different contact points of the wheel; *iv*) slippage; *v*) uneven floors, and so on. (Borenstein and Feng, 1996b; Chenavier and Crowley, 1992; Dinçay, 2010; Wang, 1988; Kleeman, 1995; Goel *et al.*, 1999; Tsumura *et al.*, 1981; Borenstein and Koren, 1991; Cox, 1991).

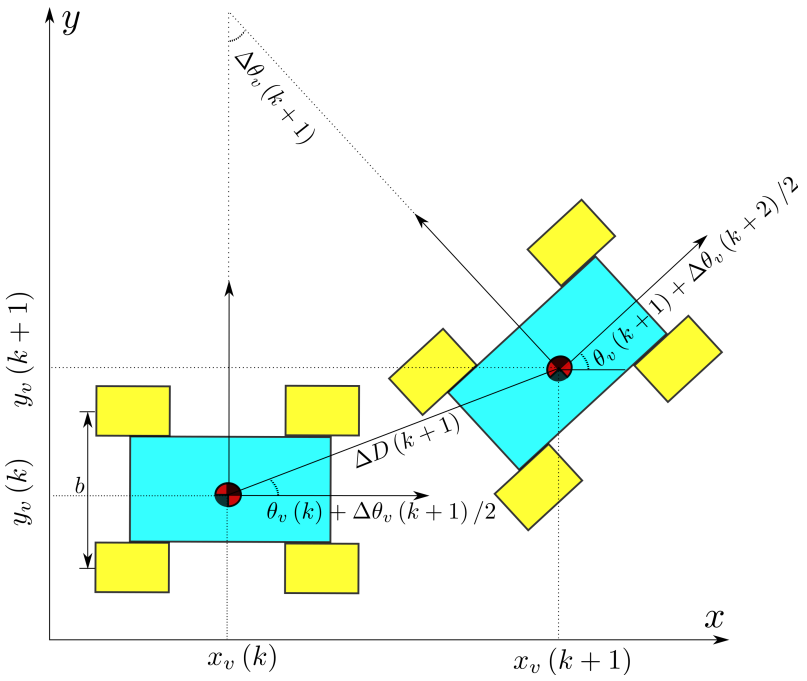


Figure 4.1: Mobile vehicle kinematics. An example of the drive system.

$$\begin{aligned} \Delta D(k+1) &= \frac{\Delta D_r(k+1) + \Delta D_l(k+1)}{2} \\ \Delta\theta_v(k+1) &= \frac{\Delta D_r(k+1) - \Delta D_l(k+1)}{b} \end{aligned} \tag{4.1}$$

$$\begin{bmatrix} x_v(k+1) \\ y_v(k+1) \\ \theta_v(k+1) \end{bmatrix} = \begin{bmatrix} x_v(k) \\ y_v(k) \\ \theta_v(k) \end{bmatrix} + \begin{bmatrix} \Delta D(k+1)\cos\left(\theta_v(k) + \dfrac{\Delta\theta_v(k+1)}{2}\right) \\ \Delta D(k+1)\sin\left(\theta_v(k) + \dfrac{\Delta\theta_v(k+1)}{2}\right) \\ \Delta\theta_v(k+1) \end{bmatrix} \tag{4.2}$$

4.1.2 Gyroscope

A gyroscope is used to measure the angular change in the mobile vehicle position. However, as mentioned earlier, this sensor suffers from the drift factor issue. Researchers have come up with different error models to estimate this drift. Barshan and Durrant-Whyte used an error model described by equation (4.3). They kept

Table 4.1: Nomenclature of encoder and gyroscope equations

Variable	**Unit**	**Description**
$\Delta D\,(k+1)$	cm	Change in distance of the center of the mobile vehicle at step $k+1$
$\Delta D_r\,(k+1)$	cm	Change in distance of the right wheel at step $k+1$
$\Delta D_l\,(k+1)$	cm	Change in distance of the left wheel at step $k+1$
$\Delta\theta_v\,(k+1)$	rad	Change in orientation of the mobile vehicle at step $k+1$
b	cm	Distance between left and right wheels
$x_v\,(k)$	cm	Mobile vehicle x position at step k
$y_v\,(k)$	cm	Mobile vehicle y position at step k
$\theta_v\,(k)$	rad	Mobile vehicle orientation at step k
$x_v\,(k+1)$	cm	Mobile vehicle x position at step $k+1$
$y_v\,(k+1)$	cm	Mobile vehicle y position at step $k+1$
$\theta_v\,(k+1)$	rad	Mobile vehicle orientation at step $k+1$
R	cm	Mobile vehicle wheel radius
C_1	rad/s	Gyroscope error model parameter
C_2	rad/s	Gyroscope error model parameter
T	s	Gyroscope error model parameter
m		Number of stops performed by the mobile vehicle
D	rad/s	Drift constant
dt	s	Time step
$\omega\,(k+1)$	rad/s	Gyroscope reading around z-axis at step $k+1$
$\Omega\,(k+1)$	rad/s	Gyroscope reading around z-axis at step $k+1$ after accounting for drift value
ϵ_{vd}	rad/s	Varying drift value
ϵ_{sd}	rad/s	Constant drift value
t	s	Time

the mobile vehicle stationary for 12 hours to collect gyroscope data, and applied the curve fit given by equation (4.3) to the gyroscope data (Barshan and Durrant-Whyte, 1995).

$$\epsilon_{cd}(t) = C_1 \left(1 - \exp^{-\frac{t}{T}} \right) + C_2 \tag{4.3}$$

In a different approach, the temperature value is considered. Since, the drift value of the gyroscope changes with temperature (Kim *et al.*, 1999). Hence, the drift value calculation is reset whenever the mobile vehicle comes to a complete stop, as shown in equation (4.4). In another gyroscope model, known as *gyrodometry* , the gyroscope data are used only when the difference between gyroscope data and encoder data is substantial (Borenstein and Feng, 1996a).

$$\epsilon_{vd}(m) = D(m) \tag{4.4}$$

The vehicle pose estimation is generated by using four different approaches. The first approach is used only with encoder data to estimate vehicle pose and orientation as given by equations (4.1), and (4.2). In the second, third, and fourth approaches both encoders and gyroscope sensors are used. The gyroscope is used for orientation calculation in the second approach, with equation (4.3) being used to take the effect of the drift into account. The encoder is used for the mobile vehicle speed calculation. This approach is known as the *constant drift approach*. The overall system for the constant drift approach is given by equations (4.5), and (4.7). In the third approach, the gyroscope is used for orientation calculation, with

equation (4.4) being used to correct for the drift. The drift value is not constant and is calculated every time the mobile vehicle comes to a complete halt. In this approach, the encoder is used to measure the speed of the mobile vehicle only and not used to determine the orientation. This approach is known as the *varying drift approach*. The overall system for the varying drift approach is given by equations (4.6), and (4.7). The fourth and last approach is known as *gyrodometry*. The determination of orientation is shared between encoder and gyroscope depending on the set threshold value. If the difference between the orientations determined by the encoder and gyroscope is large, the gyroscope data are used for the orientation measurement. The encoder is used for speed calculations of the mobile vehicle. In summary, the encoder is always used to calculate the speed of the mobile vehicle but the orientation value is determined differently in the four approaches. The author intends to study all four different approaches for vehicle pose estimation and select the one with lowest error before applying the feature extraction algorithm. The results obtained with the different approaches will be presented and comparative discussions on vehicle pose estimation will be made.

$$\Omega\left(k+1\right)=\omega\left(k+1\right)-\epsilon_{cd}\left(t\right) \tag{4.5}$$

$$\Omega\left(k+1\right)=\omega\left(k+1\right)-\epsilon_{vd}\left(m\right) \tag{4.6}$$

$$\begin{bmatrix} x_v\left(k+1\right) \\ y_v\left(k+1\right) \\ \theta_v\left(k+1\right) \end{bmatrix} = \begin{bmatrix} x_v\left(k\right) \\ y_v\left(k\right) \\ \theta_v\left(k\right) \end{bmatrix} + \begin{bmatrix} \Delta D\left(k+1\right)\cos\left(\theta_v\left(k\right)+\Omega\left(k+1\right)dt\right) \\ \Delta D\left(k+1\right)\sin\left(\theta_v\left(k\right)+\Omega\left(k+1\right)dt\right) \\ \Omega\left(k+1\right)dt \end{bmatrix} \tag{4.7}$$

4.2 Experimental Arrangement

The mobile vehicle setup consists of the following components and features: *i*) two SONAR sensors, *ii*) servo motor, *iii*) four DC motors, *iv*) four encoders, *v*) Arduino uno and Arduino mega units, *vi*) two $9V$ batteries, and one $12V$ battery, *vii*) Inertial Measurement Unit (IMU) sensor (Adafruit 9-DOF IMU Breakout - L3GD20H + LSM303) for tri-axial accelerometer, gyroscope, and magnetometer measurement, *viii*) three battery switches, *ix*) Xbee wireless communication, and *x*) SD card for storing encoder, IMU sensor, and SONAR data. The mobile vehicle configuration is shown in Figure 4.2 and the sensor specifications are given in Table 4.2.. The environment boundaries (walls) used in the current experiments are similar to the ones used in the group for SLAM studies and these boundaries have been constructed by using plexiglass material (Ismail and Balachandran, 2014; Jaai *et al.*, 2012). Two experimental configurations have been used, one with 2 interior features and another without interior features, as shown in Figure 4.3. Essentially, the environment with no interior features is easier to study, since it has fewer landmarks. On the other hand, the environment with 2 interior feature is little bit more realistic, since the interior of an oil storage tank contains features that can be used as landmarks and can aid in mobile vehicle navigation.

Table 4.2: Sensor specifications used in the experimental setup

Name	Description
SONAR	Name = 'HRLV-MaxSONAR-EZ4'; Resolution = 1 mm; Reading rate = 10 Hz; Operational Hz = 42 kHz
IMU	Name = 'Adafruit 9-DOF IMU Breakout - L3GD20H + LSM303'; Resolution = $\pm 2000^o/s$; Reading rate = 100 Hz individual use; Reading rate = 50 Hz integrated with other sensors ; The z-axis gyroscope was only used from the IMU
Encoder	Name = 'Quadrature magnetic encoder'; Resolution = 8400 cpr; Our Resolution = 2100 cpr because of Arduino Mega limitations, where cpr stands for counts per revolution; Reading rate = 50 Hz

In the first set of experiments, the mobile vehicle is placed inside the closed rectangular environment with 2 interior features. The mobile vehicle is taken to a new location in about 15 to 30 cm increments and over this travel, data is collected from the 4 encoders and the IMU sensor. After each travel increment, the mobile vehicle is brought to a complete stop and SONAR data are collected from two SONAR sensors attached to the servo motor. The servo motor is rotated 180^o in 1^o increments to perform a full 360^o scan since the two SONAR sensors are attached opposite of each other on the servo motor. Since the vehicle is stationary for the SONAR scan, no extraneous errors are added due to vehicle motion. After completing the SONAR scan, the mobile vehicle is kept stationary for 4 seconds. Then, the process is repeated 26 times to collect more information about the environment. For the environmental setup with 2 interior features, the estimated 26 mobile vehicle locations are shown in Figure 4.4. In the second set of experiments, the mobile vehicle is placed insider the closed rectangular environment with no interior features

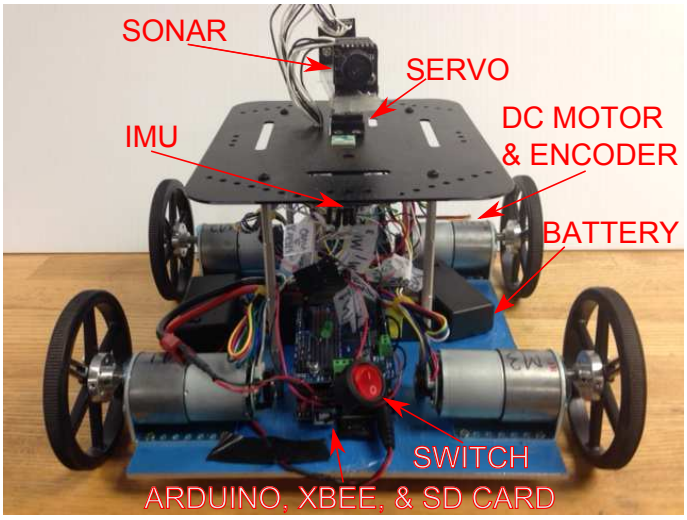
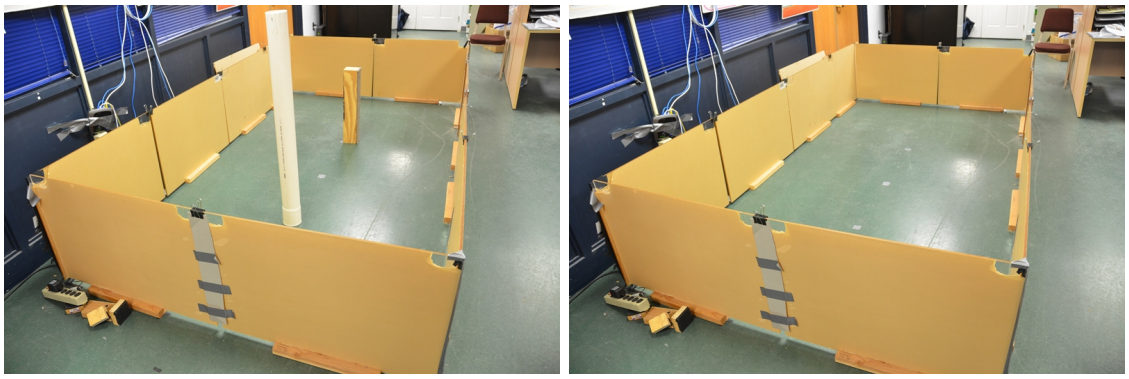


Figure 4.2: Mobile vehicle setup used in experiments.

and the same approach is followed. The estimated 26 mobile vehicle locations for the environment with no interior features are shown in Figure 4.5.



(a) arrangement with 2 interior features (b) arrangement with no interior features

Figure 4.3: Experimental arrangements with and without interior features. The dimension of the environment is as follows: width = 182.0 cm, length 335.1 cm, and height = 63.5 cm.

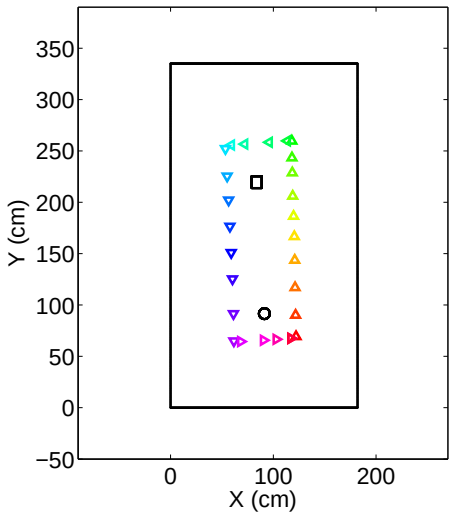


Figure 4.4: Mobile vehicle pose estimation locations for the environment with 2 interior features. The triangles mark the mobile vehicle locations where the SONAR scans take place (Ismail and Balachandran, 2015b).

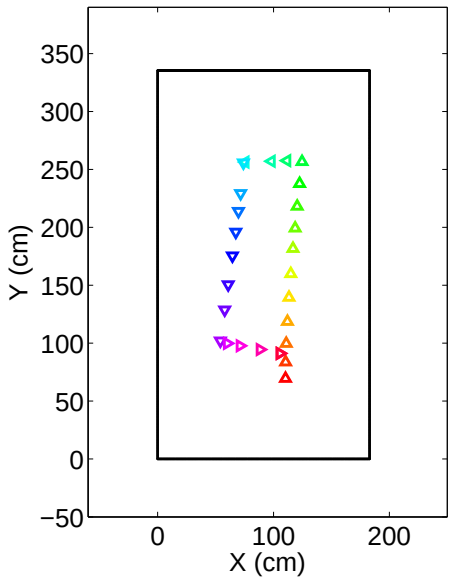


Figure 4.5: Mobile vehicle pose estimation for the environment with no interior features. The triangles mark the mobile vehicle locations where the SONAR scans take place (Ismail and Balachandran, 2015b).

4.3 Results

The results are presented for two different environment arrangement: *i*) two interior features and *ii*) no interior features. First, the result for the environment with two interior features are presented.

4.3.1 Case with Interior Features

Before conducting the vehicle pose estimate, the drift value for the gyroscope sensor was calculated. To realize this, the mobile vehicle was kept stationary and gyroscope data were collected for approximately 2 and a half hours. The result is shown in Figure 4.6. The gyroscope data were examined by using equation (4.3). The fitted error model parameters determined are $C_1 = -0.006167 rad/s$, $C_2 = -0.000463 rad/s$, and $T = 0.1576 s$. It was noticed that the error model output approaches $C_1 + C_2$ after less than 1 second. Therefore, the drift value of the gyroscope is considered constant with a value of $C_1 + C_2 = -0.00663 rad/s$. Another approach for calculating the drift is based on calculating the average value whenever the mobile vehicle come to a complete stop. This approach is more suitable because ambient temperature and other external factors effect the gyroscope performance. The results obtained from the constant drift approach are presented along with those obtained from the varying drift approach in Figure 4.7.

The mobile vehicle is driven in the environment with 2 interior features and is stopped 26 times to perform full SONAR scans at each location. The four different approaches, namely, the constant drift, varying drift, gyrodometry, and encoder only

Table 4.3: Comparisons of vehicle pose estimation results from four different approaches for the environment with 2 interior features. The absolute differences between the estimated final position and the actual final position are shown.

Approaches	**Absolute Diff. x (cm)**	**Absolute Diff. y (cm)**
Varying Drift	0.43	4.99
Constant Drift	17.12	10.34
Gyrodometry	15.62	8.61
Encoder Only	81.54	207.20

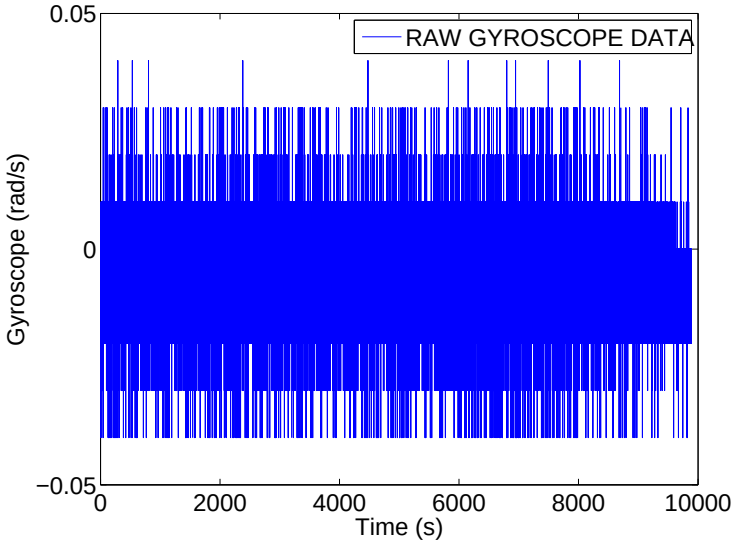


Figure 4.6: Gyroscope data collected from a stationary mobile vehicle.

approaches are tested and compared for vehicle pose estimation . The results are shown in Figure 4.8. The absolute differences between the estimate final position and the actual final position are calculated and presented in Table 4.3. It is seen that the results obtained with the varying drift approach have the least amount of absolute difference between the final estimate and absolute position. Therefore, the vehicle pose estimation from the the varying drift approach is used as input for feature extraction.

The environment with two interior features is shown in Figure 4.3a and the

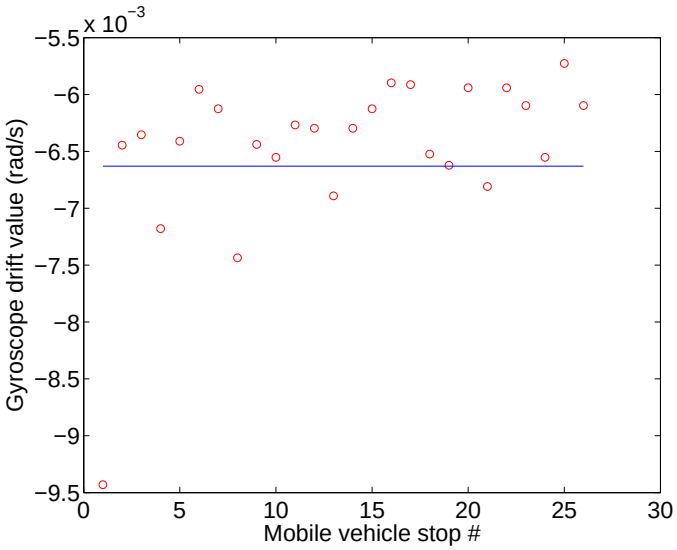


Figure 4.7: Gyroscope drift values. The blue line is the value determined from the constant drift approach. The red circles are results obtained from the varying drift approach for the studies with 2 interior features.

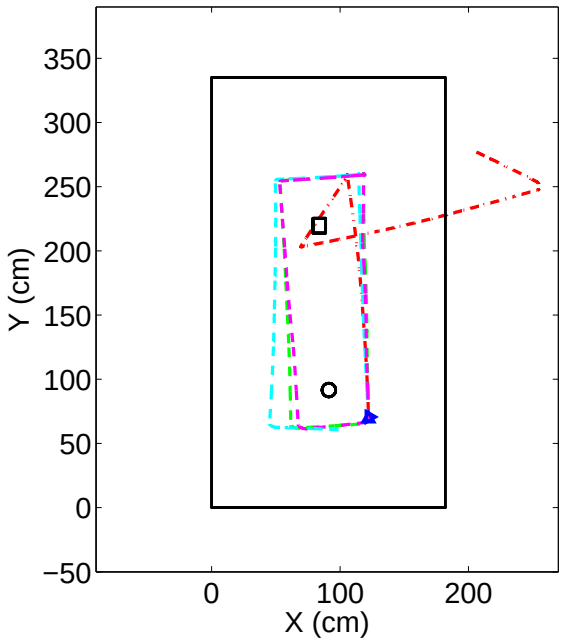


Figure 4.8: Mobile vehicle pose estimation with the four different approaches for the environment with 2 interior features. The red dashed lines are results from the encoder only approach, the green dashed lines are results from the varying drift approach, the cyan dashed lines are results from the constant drift approach, and finally, the magenta colored lines are results from the gyrodometry approach with threshold value of $0.0033 rad/s$. The blue triangle pointing upwards marks the true starting point, and the blue triangle pointing to the right marks the true ending point.

mobile vehicle pose information for the varying drift approach is shown in Figure 4.4. After considering the best vehicle pose estimation, the raw SONAR scan results are shown in Figure 4.9. Next, the raw SONAR data is processed with the limit constraints wherein data below 50 cm and above 200 cm are removed. The results are shown in Figure 4.10. Subsequently, the limit constraint results are processed with the TBF algorithm for which the selected threshold value was chosen as 10. The TBF results are shown in Figure 4.11. Also, in parallel, the limit constraint processed data are input to the SONAR salient algorithm, with the acceptable radius range values being 3 cm and 20 cm. The SONAR salient algorithm processed results are shown in Figure 4.12. Next, the TBF results are processed with the Hough Transform algorithm and the results are known as THF results. These results are shown in Figure 4.13. After that, the THF results are clustered using k-mean clustering and the line intersections are marked with circles. The k-mean clustering results are shown in Figure 4.14. Finally, the k-mean clustering results, results, and SONAR salient results are integrated together to produce the final plot, which is the outcome of the fusion algorithm. The final results are shown in Figure 4.15.

4.3.2 Case with no Interior Features

A similar procedure is followed for the case with no interior features. The mobile vehicle is driven inside the environment with no interior features and full SONAR scans are taken at 26 different locations. The environment is shown in Figure 4.3b and the mobile vehicle pose information is shown in Figure 4.5. Before

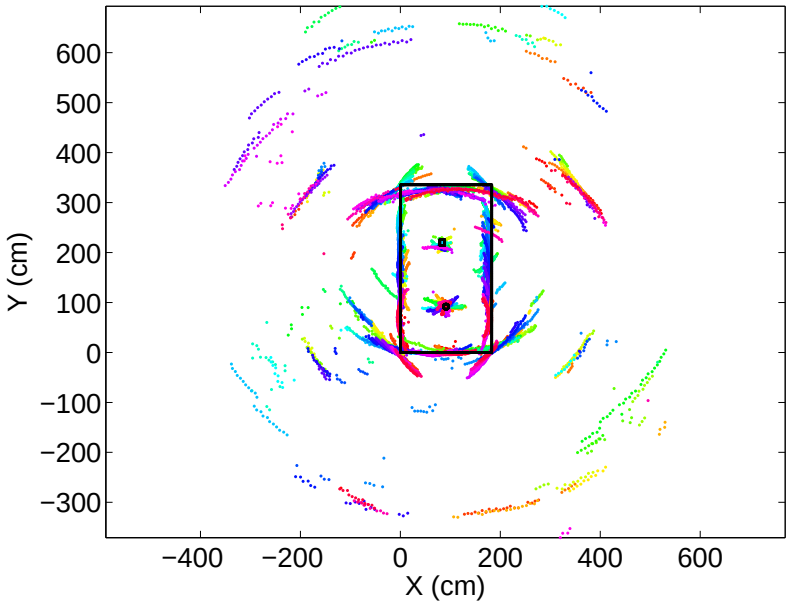


Figure 4.9: Environment with 2 interior features: raw SONAR scan results.

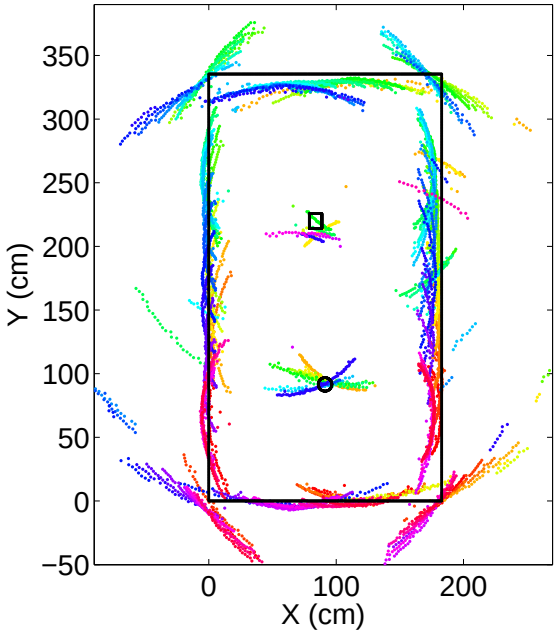


Figure 4.10: Environment with 2 interior features: limit constraint processed results.

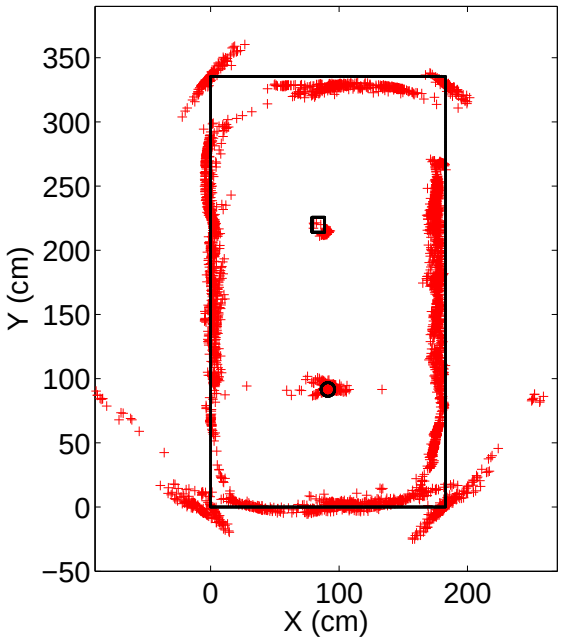


Figure 4.11: Environment with 2 interior features: TBF results.

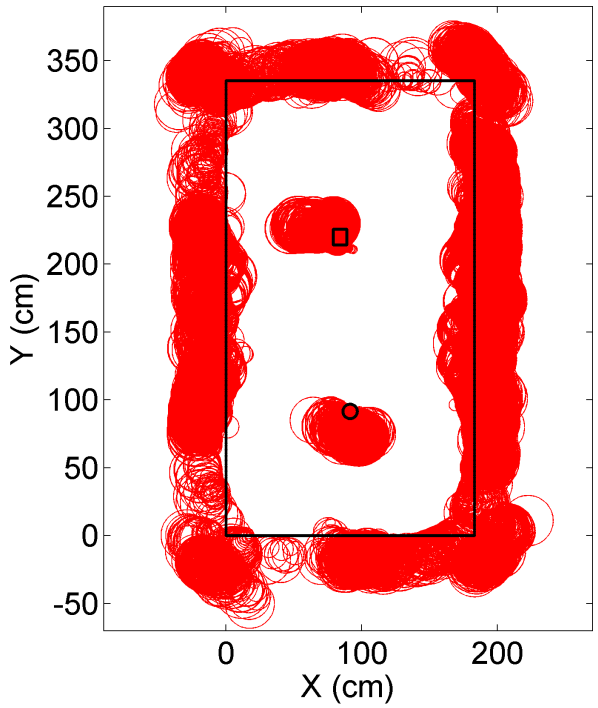


Figure 4.12: Environment with 2 interior features: SONAR salient algorithm processed results, before classification.

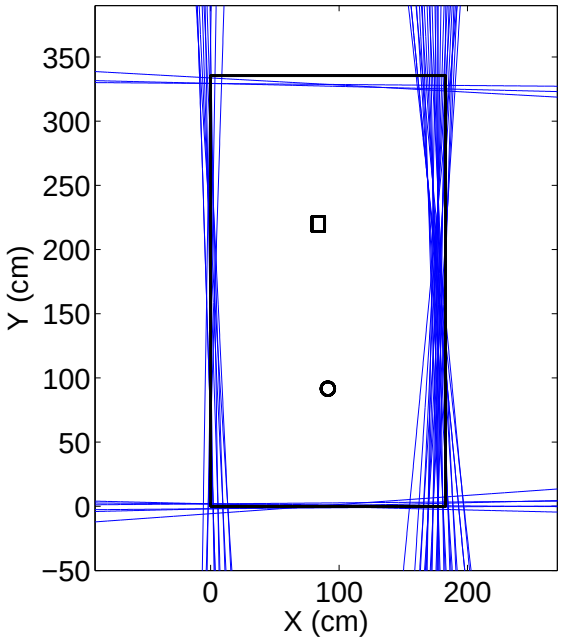


Figure 4.13: Environment with 2 interior features: THF results, where the number of bins were 200 and the number of lines were 120.

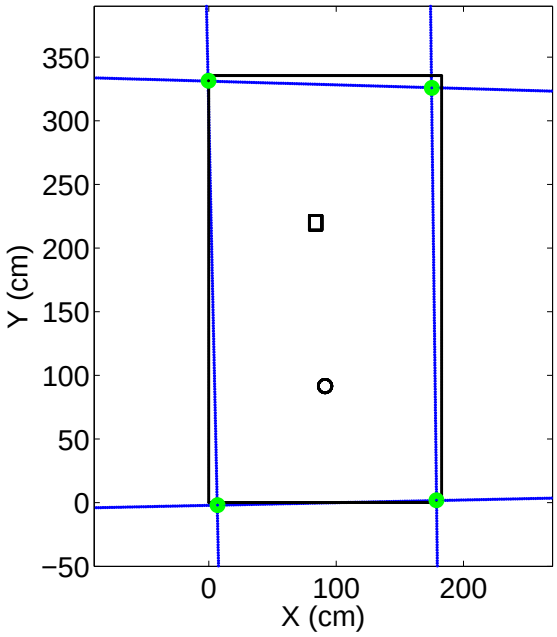


Figure 4.14: Environment with 2 interior features: k-mean clustering results.

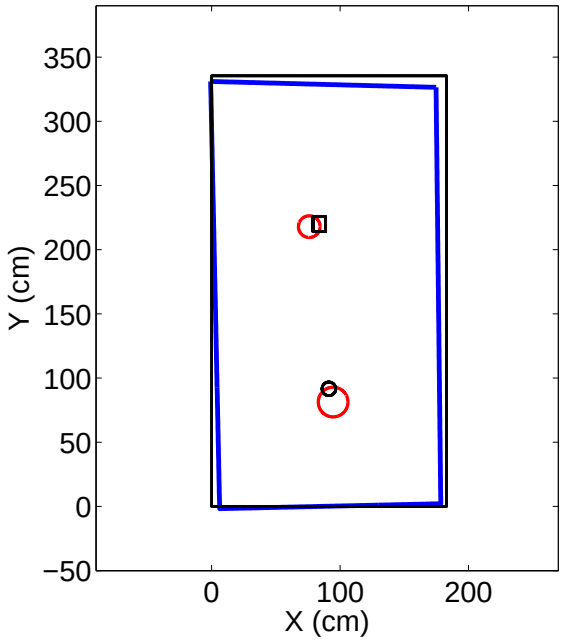


Figure 4.15: Environment with 2 interior features: fusion algorithm results with the varying drift approach.

conducting the vehicle pose estimation, the drift values are calculated. For the constant drift analysis, no additional calculations are needed, since similar results are used to the previous case with 2 interior features. For the varying drift analysis, the mean value of the gyroscope data is calculated whenever the mobile vehicle comes to a complete stop during the experiment. Related results are shown in Figure 4.16. Next, the vehicle pose estimation results obtained by using the four different approaches, namely, the constant drift, varying drift, gyrodometry, and encoder only approaches, are shown in Figure 4.17. The results obtained for the vehicle pose are compared, as shown in Table 4.4. Based on these results, the information obtained with the varying drift and constant drift approaches sent for processing with the fusion algorithm. The final results obtained with the varying drift approach are shown in Figure 4.18, and the final results obtained with the

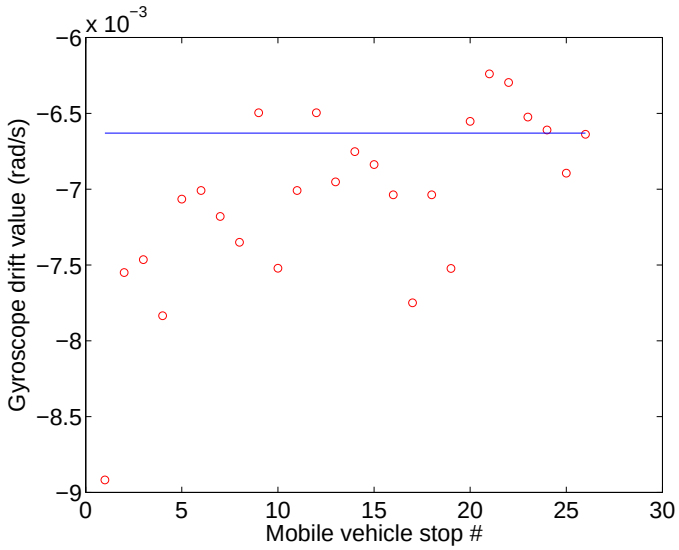


Figure 4.16: Gyroscope drift values. The blue line is the value determined from the constant drift approach. The red circles are results obtained from the varying drift approach for the studies with no interior features.

Table 4.4: Comparisons of vehicle pose estimation results from four different approaches for the environment with no interior features. The absolute differences between the estimated final position and the actual final position are shown.

Approaches	**Absolute Diff. x (cm)**	**Absolute Diff. y (cm)**
Varying Drift	16.42	4.57
Constant Drift	0.58	1.75
Gyrodometry	52.33	3.12
Encoder Only	68.73	163.08

constant drift approach are shown in Figure 4.19.

From Figure 4.18, and 4.19 it can be seen that the varying drift method produced the better results. However, the constant drift approach yielded a better estimate of the final position. One cannot draw conclusions just on the basis of a comparison of final positions alone. Both the final positions as well as the proposed feature extraction results and related accuracy need to be taken into account.

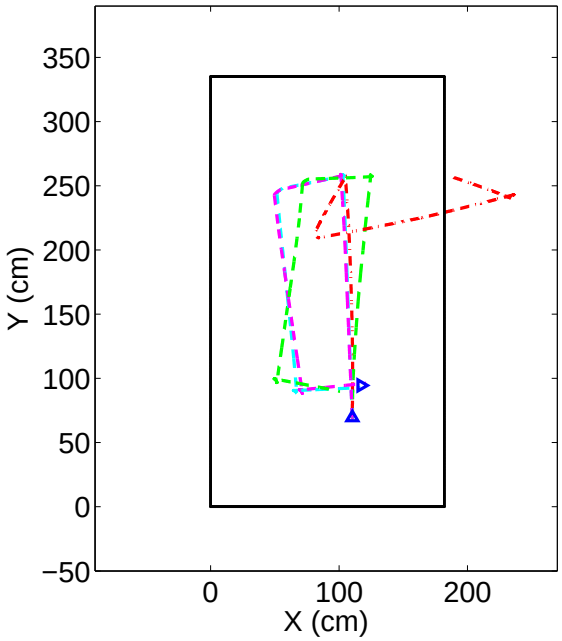


Figure 4.17: Mobile vehicle pose estimation with the four different approach for the environment with no interior features. The red dashed lines represent results obtained from the encoder only approach, the green dashed lines represent results obtained from the varying drift approach, the cyan dashed lines represent results obtained from the constant drift approach, and finally, the magenta colored represent results obtained from the gyrodometry approach with threshold value of $0.0033 rad/s$. The blue triangle pointing upwards mark the true starting point, and the blue triangle pointing to the right marks the true ending point.

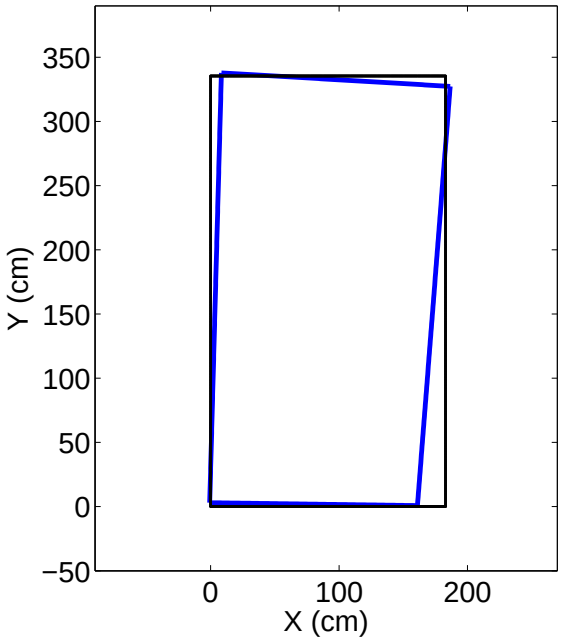


Figure 4.18: Environment with no interior features: fusion algorithm results with the varying drift approach.

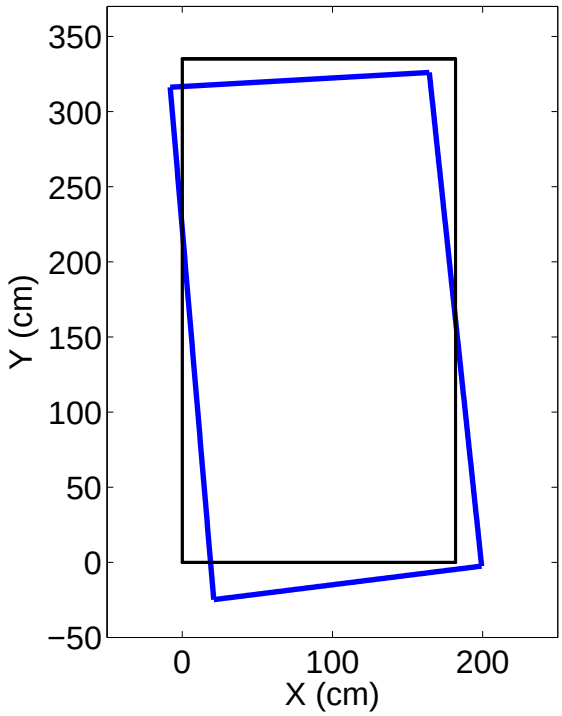


Figure 4.19: Environment with no interior features: fusion algorithm results with the constant drift approach.

4.4 Summary

In this work, the author has studied four different approaches, namely, constant drift, varying drift, gyrodometry and encoder only approaches for vehicle pose estimation. It was observed that for the environment with 2 interior features, the varying drift approach for vehicle pose estimate provided the most accurate final position. The vehicle pose estimate data obtained with the varying drift approach was processed with the fusion algorithm and the environment captured was noted to have all unique features (4 walls and 2 interior features). For the environment with no interior features, the constant drift approach yielded a better final pose estimate than the varying drift approach. Therefore, both position estimates were processed with the fusion algorithm and it was seen that better results are obtained with the

varying drift approach. Overall, the varying drift approach is noted to provide better position estimation than the other approaches because the ambient temperature and other external factors effect the gyroscope reading. The gyrodometry results can be further improved by selecting a better threshold value. In the next chapter, the current work will serve as a platform for SLAM applications.

Chapter 5

SLAM Studies

In this chapter, SLAM implementation is examined with experimental data. First, the vehicle model, covariance of the control input, output of the feature extraction algorithm, observation model, data association, Extended Kalman Filter (EKF), and augmentation matrices are discussed. Next, an experimental setup using a mobile vehicle, which consists of a SONAR sensor, gyroscope, and encoder sensor is studied in an environment using EKF-SLAM. The experiments are conducted in conditions with and without slippage.

There are two types of sensors, a sensor whose error accumulates as the mobile vehicle moves and/or as time progresses, and a sensor whose error does not accumulate. Encoders, gyroscopes, and accelerometers are examples of the first type of sensor. Lasers, SONAR sensors, compasses, and Global Positioning Systems (GPS) are examples of the second type of sensor, known as absolute sensors. An absolute sensor does not accumulate error over time. Through the use of EKF-SLAM, the accumulated errors from the encoder and gyroscope can be reduced, or even bounded, by integrating the absolute error from an absolute sensor with this method. Researchers have used encoders and gyroscopes with different absolute sensors, such as cameras, lasers, compasses, GPSs, and SONAR sensors. Previously in Chapter 1, the reasons why several absolute sensors are avoided are explained in detail.

5.1 EKF-SLAM Operation

In the EKF-SLAM operation, the data is augmented into a vector known as the system state vector, $\mathbb{X}$, and the system covariance matrix, $\mathbb{P}$. The system state vector, which is an augmented vector, contains the mobile vehicle position as well as the feature locations. The system covariance matrix, an augmented matrix, contains the covariance between mobile vehicle position and the feature location. The system state vector as well as the system covariance matrix at step k are define in equations (5.1) and (5.1) and described in Table 5.1.

$$\mathbb{X}(k) = \left[\mathbf{x_v}(k) \;\middle|\; \mathbf{x_f}(k,1) \ldots \mathbf{x_f}(k,j) \ldots \mathbf{x_f}(k,m)\right]^T \tag{5.1}$$

where,

$\mathbf{x_v}(k) = [x_v(k) \quad y_v(k) \quad \theta_v(k)]$

$\mathbf{x_f}(k,j) = [x_f(k,j) \quad y_f(k,j)]$

$$\mathbb{P}(k) = \begin{bmatrix} P_{vv}(k) & P_{vf}(k) \\ P_{fv}(k) & P_{ff}(k) \end{bmatrix} \tag{5.2}$$

For the discussion in this chapter, bold letter denote vectors, a subscript of v denotes "with respect to the vehicle", a subscript of u denotes "with respect to the controller", a subscript of f denotes "with respect to the feature", double struck font denotes augmented vectors or matrices. Also, hats denote predicted values, while upside down hats refer to actual values.

Table 5.1: Variable description for the vehicle model section

Symbol	Unit	Description
k		Step number or time for the experiments of default, dry powder, and hydrophobic coating cases
$\mathbb{X}(k)$		Augmented vector that contains mobile vehicle position as well as feature locations at step k
$\mathbf{x_v}(k)$		Mobile vehicle position at step k
$x_v(k)$	cm	Mobile vehicle x-position at step k
$y_v(k)$	cm	Mobile vehicle y-position at step k
$\theta_v(k)$	rad	Mobile vehicle orientation at step k
$\mathbf{x_f}(k,1)$		First feature location at step k
$\mathbf{x_f}(k,j)$		j^{th} feature location at step k
$\mathbf{x_f}(k,m)$		Last feature location at step k
$x_f(k,j)$	cm	j^{th} feature x-location at step k
$y_f(k,j)$	cm	j^{th} feature y-location at step k
$\mathbb{P}(k)$		Augmented matrix that contains the covariance of mobile vehicle position and feature locations at step k
$P_{vv}(k)$		Covarinace matrix of mobile vehicle position with respect to mobile vehicle position at step k
$P_{vf}(k)$		Covarinace matrix of mobile vehicle position with respect to feature locations at step k
$P_{fv}(k)$		Covarinace matrix of feature locations with respect to mobile vehicle position at step k
$P_{ff}(k)$		Covarinace matrix of feature locations with respect to feature locations at step k
$\mathbf{u}(k+1)$		Control input vector at step $k+1$
$V_R(k+1)$	cm/s	Average velocity of top and bottom right wheels at step $k+1$
$V_L(k+1)$	cm/s	Average velocity of top and bottom left wheels at step $k+1$
$V(k+1)$	cm/s	Average velocity of the mobile vehicle at step $k+1$
$\Omega(k+1)$	rad/s	Angular velocity with respect to the z-axis after accounting for the varying drift factor at step $k+1$; for more details on the varying drift factor, a reader is referred to Chapter 4
$\hat{\mathbf{x}}_\mathbf{v}(k+1)$		Prediction of mobile vehicle position at step $k+1$
dt	s	Time step
$\hat{P}_{vv}(k+1)$		Prediction of covarinace matrix of mobile vehicle position with respect to mobile vehicle position at step $k+1$
$J_{vv}(k+1)$		Jacobian of mobile vehicle position with respect to mobile vehicle position at step $k+1$
$J_{vu}(k+1)$		Jacobian of mobile vehicle position with respect to mobile vehicle control inputs at step $k+1$
$Q(k+1)$		Covariance of the control inputs

5.1.1 Vehicle Model

The vehicle model, which uses data from the encoder and gyroscope, was defined previously in Chapter 4 and is shown below in equation (5.3), in order to make the derivation of the Jacobian easier to follow. The mobile vehicle position at step k is given by $\mathbf{x_v}(k) = [x_v(k) \quad y_v(k) \quad \theta_v(k)]^T$, and the control of the mobile vehicle at step $k+1$ is given by $\mathbf{u}(k+1) = [V_R(k+1) \;\; V_L(k+1) \quad \Omega(k+1)]^T$. The control input, $\mathbf{u}(k+1)$, is used to predict the vehicle position, $\hat{\mathbf{x}}_\mathbf{v}(k+1)$, at step $k+1$.

$$\begin{aligned} \hat{\mathbf{x}}_\mathbf{v}(k+1) &= f(\mathbf{x_v}(k), \mathbf{u}(k+1)) \\ &= \begin{bmatrix} x_v(k) + V(k+1)\,dt\cos(\theta_v(k) + \Omega(k+1)\,dt) \\ y_v(k) + V(k+1)\,dt\sin(\theta_v(k) + \Omega(k+1)\,dt) \\ \theta_v(k) + \Omega(k+1)\,dt \end{bmatrix} \end{aligned} \tag{5.3}$$

For further details about the variables used in equation (5.3), the reader is referred to Table 5.1. After predicting the mobile vehicle state at step $k+1$, the predicted covariance matrix of the mobile vehicle position with respect to the mobile vehicle position, $\hat{P}_{vv}(k+1)$, is calculated at step $k+1$. To calculate $\hat{P}_{vv}(k+1)$, two Jacobian matrices are calculated from the vehicle model equation which are the vehicle Jacobian matrix, $J_{vv}(k+1)$, and the control Jacobian matrix, $J_{vu}(k+1)$.

The vehicle Jacobian is defined in equation (5.4).

$$J_{vv}(k+1) = \frac{\partial f(\mathbf{x_v}(k), \mathbf{u}(k+1))}{\partial \mathbf{x_v}(k)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_v(k)} & \frac{\partial f_1}{\partial y_v(k)} & \frac{\partial f_1}{\partial \theta_v(k)} \\ \frac{\partial f_2}{\partial x_v(k)} & \frac{\partial f_2}{\partial y_v(k)} & \frac{\partial f_2}{\partial \theta_v(k)} \\ \frac{\partial f_3}{\partial x_v(k)} & \frac{\partial f_3}{\partial y_v(k)} & \frac{\partial f_3}{\partial \theta_v(k)} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & -V(k+1)\,dts \\ 0 & 1 & V(k+1)\,dtc \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

The control Jacobian is defined in equation (5.5).

$$J_{vu}(k+1) = \frac{\partial f(\mathbf{x_v}(k), \mathbf{u}(k+1))}{\partial \mathbf{u}(k+1)} = \begin{bmatrix} \frac{\partial f_1}{\partial V_R(k+1)} & \frac{\partial f_1}{\partial V_L(k+1)} & \frac{\partial f_1}{\partial \Omega(k+1)} \\ \frac{\partial f_2}{\partial V_R(k+1)} & \frac{\partial f_2}{\partial V_L(k+1)} & \frac{\partial f_2}{\partial \Omega(k+1)} \\ \frac{\partial f_3}{\partial V_R(k+1)} & \frac{\partial f_3}{\partial V_L(k+1)} & \frac{\partial f_3}{\partial \Omega(k+1)} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{dt}{2}c & \frac{dt}{2}c & -V(k+1)\,dt^2 s \\ \frac{dt}{2}s & \frac{dt}{2}s & V(k+1)\,dt^2 c \\ 0 & 0 & dt \end{bmatrix} \tag{5.5}$$

where,

$$V(k+1) = \frac{V_L(k+1) + V_R(k+1)}{2}$$

$$c = \cos(\theta_v(k) + \Omega(k+1)\,dt)$$

$$s = \sin(\theta_v(k) + \Omega(k+1)\,dt)$$

The covariance of the control inputs of the mobile vehicle is defined in equation (5.6). The calculation of the values used for the covariance of the control input is explained in the next section.

$$Q\left(k+1\right)=\begin{bmatrix}\sigma_{VR}^{2}\left(k+1\right) & 0 & 0\\ 0 & \sigma_{VL}^{2}\left(k+1\right) & 0\\ 0 & 0 & \sigma_{\Omega}^{2}\left(k+1\right)\end{bmatrix} \tag{5.6}$$

The predicted covariance matrix of the mobile vehicle position with respect to mobile vehicle position at step $k+1$, $\hat{P}_{vv}\left(k+1\right)$, can be calculated through the use of equations (5.4) - (5.6), as can be seen from equation (5.7).

$$\hat{P}_{vv}\left(k+1\right)=J_{vv}\left(k+1\right)P_{vv}\left(k\right)J_{vv}\left(k+1\right)^{T}+J_{vu}\left(k+1\right)Q\left(k+1\right)J_{vu}\left(k+1\right)^{T} \tag{5.7}$$

For more information on the variables used in equations (5.4) - (5.7), one is referred to Table 5.1.

5.1.2 Covariance of the Control Input

In this section, the covariance of control input will be determined experimentally. Several tests are conducted to get the covariance values. The straight line test is used to get the variance of the left and right encoder, whereas the zero turn radius test is used to get the variance of the gyroscope. For each test, the mobile vehicle was put through the specific test 55 times and the true values are measured by hand with a measuring tape for the straight line test and with a digital protractor for the

zero-turn radius test.

5.1.2.1 Variance of the encoder

The straight line test is used to find the variance of the top left encoder. The mobile vehicle is given a command to go in a straight line (y-axis) with a constant speed (half of the top speed) for 9 seconds. After that, the true value is measured using a measuring tape. The straight line test is repeated 55 times and the true values are tabulated in Table A.1. Even though the mobile vehicle drifted slightly along the x-axis while doing the straight line test, only y-values were considered when calculating the variance of the top left encoder, as the distance traveled in the x-direction was negligible when compared with the distance in the y-direction. This assumption is used to reduce the complexity of determining the variance of the top left encoder. Later, a factor is added to account for additional uncertainty, for example, the drift along the x-axis or slippage from powder or dust.

The derivation of the variance of the encoder is presented next. First, the offsets between the true values and the estimated measurements are calculated, as shown in equation (5.8).

$$\Delta y_v(i) = \check{y}_v(i) - \hat{y}_v(i) \tag{5.8}$$

Next, the mean and the variance are determined for the offset values. Following this, two factors are calculated which are defined in equations (5.9) and (5.10), which are the correction factor and the variance factor for the encoder.

Table 5.2: Variable description for the encoder variance

Symbol	Unit	Description
i		Experimental number of the straight line tests or zero-turn radius tests
$\hat{y}_v\,(i)$	cm	The estimated y-position of the mobile vehicle estimated using encoder counts for the i^{th} experiment
$\hat{\mathbf{y}}_\mathbf{v}$		The vector of estimated y-positions of the mobile vehicle estimated using encoder counts
$\check{y}_v\,(i)$	cm	The true y-position of the mobile vehicle measured using measuring tape for the i^{th} experiment
$\Delta y_v\,(i)$	cm	The offset between the true y-position and the estimated y-position of the mobile vehicle for the i^{th} experiment
$\Delta\mathbf{y}_\mathbf{v}$		The offset vector between the true y-position and the estimated y-position of the mobile vehicle
k_{enc}	cm^2/cm	Variance factor which is used in variance calculation for the encoder
C_{enc}	cm/cm	Correction factor which is used in correcting the offset of the encoder reading
$\Delta D_{lo}\,(k+1)$	cm	Travel distance for the change in encoder reading before applying the correction factor to it, for the left wheels at step $k+1$ (i.e., "old" value)
$\Delta D_{ln}\,(k+1)$	cm	Travel distance for the change in encoder reading after applying the correction factor to it, for the left wheels at step $k+1$ (i.e., "new" value)
$\Delta D_{ro}\,(k+1)$	cm	Travel distance for the change in encoder reading before applying the correction factor to it, for the right wheels at step $k+1$ (i.e., "old" value)
$\Delta D_{rn}\,(k+1)$	cm	Travel distance for the change in encoder reading after applying the correction factor to it, for the right wheels at step $k+1$ (i.e., "new" value)
$\sigma^2_{VL}\,(k+1)$	cm^2	Variance of the left encoder at step $k+1$
$\sigma^2_{VR}\,(k+1)$	cm^2	Variance of the right encoder at step $k+1$

$$k_{enc} = \frac{var\left(\Delta \mathbf{y_v}\right)}{mean\left(\hat{\mathbf{y}}_\mathbf{v}\right)} \tag{5.9}$$

$$C_{enc} = \frac{mean\left(\Delta \mathbf{y_v}\right)}{mean\left(\hat{\mathbf{y}}_\mathbf{v}\right)} \tag{5.10}$$

In these equations, $(\mathbf{y_v})$ refers to the vector whose components are the 55 experimental offsets. After that, the correction factor is used to determine the new change in distance for each of the left and right wheels, as defined in equation (5.11).

$$\begin{aligned} \Delta D_{ln}\left(k+1\right) &= (1 - C_{enc})\Delta D_{lo}\left(k+1\right) \\ \Delta D_{rn}\left(k+1\right) &= (1 - C_{enc})\Delta D_{ro}\left(k+1\right) \end{aligned} \tag{5.11}$$

Finally, the variances for the left and right encoder readings are calculated from the variance factor, as defined in equation (5.12). The variables in equations (5.8) - (5.11) are defined in Table 5.2.

$$\begin{aligned} \sigma^2_{VL}\left(k+1\right) &= k_{enc}\Delta D_{ln}\left(k+1\right) \\ \sigma^2_{VR}\left(k+1\right) &= k_{enc}\Delta D_{rn}\left(k+1\right) \end{aligned} \tag{5.12}$$

5.1.2.2 Variance of the gyroscope

The zero-turn radius test is used to find the variance of the gyroscope along the z-axis. The mobile vehicle is given a command to turn the left wheels forward and the right wheels backward with a constant speed (half of the top speed) for 6 seconds. This causes the mobile vehicle to turn in a particular way, known as the zero-turn radius. After that, the true value was measured by using a digital

protractor. The zero-turn radius test is repeated 55 times and the true values are tabulated in Table A.2.

By using a similar approach as the straight line test, the variance of the gyroscope in the z-direction is calculated next. As explained in Chapter 4, a drift is present in gyroscope data. To account for the drift value, the mobile vehicle was held stationary for 4 seconds, approximately, before doing the zero-turn radius, where the gyroscope data for the stationary position are averaged and subsequently subtracted from the gyroscope data when the mobile vehicle starts to move. For more detail about calculating the varying drift value, the reader is referred to Chapter 4. After accounting for the drift value, the gyroscope data are multiplied by time and summed to estimate the total angle of the mobile vehicle after performing the zero-turn radius, as shown in equation (5.13).

$$\hat{\theta}_v(i) = \sum_{p=1}^{p_t} (\omega(p) - \epsilon_{vd})\, dt \tag{5.13}$$

Next, the offsets between the true angle values and the estimate angle measurements are calculated and shown in equation (5.14).

$$\Delta\theta_v(i) = \check{\theta}_v(i) - \hat{\theta}_v(i) \tag{5.14}$$

Subsequently, the mean and the variance are determined for the offset values. Following this, the correction factor and the variance factor for the gyroscope are calculated, as defined in equations (5.15) and (5.16).

Table 5.3: Variable description for the gyroscope variance

Symbol	Unit	Description
p		Step number or time for the zero-turn radius test
p_t		Total number of step numbers
$\omega\,(p)$	rad/s	Angular velocity from the inertial measurement units (IMU) sensor around the z-axis at step p
ϵ_{vd}	rad/s	Varying drift value
$\hat{\theta}_v\,(i)$	rad	The estimated orientation of the mobile vehicle estimated using IMU sensor for the i^{th} experiment
$\hat{\boldsymbol{\theta}}_{\mathbf{v}}$		The estimated orientation vector of the mobile vehicle estimated using IMU sensor
$\check{\theta}_v\,(i)$	rad	The true orientation of the mobile vehicle measured using digital protractor for the i^{th} experiment
$\Delta\theta_v\,(i)$	rad	The offset between the true and the estimated orientation of the mobile vehicle for the i^{th} experiment
$\Delta\boldsymbol{\theta}_{\mathbf{v}}$		The offset vector between the true and the estimated orientations of the mobile vehicle
k_{gyro}	rad^2/rad	Variance factor which is used in variance calculation of the gyroscope
C_{gyro}	rad/rad	Correction factor which is used in correcting the offset in the gyroscope reading
$\Omega_o\,(k+1)$	rad/s	Angular velocity around the z-axis from the IMU sensor after accounting for the varying drift value at step $k+1$
Ω_n	rad/s	Angular velocity around the z-axis from the IMU sensor after applying the correction factor at step $k+1$
$\sigma^2_{\Omega}\,(k+1)$	rad^2/s^2	Variance of the gyroscope at step $k+1$

$$k_{gyro} = \frac{var\left(\Delta\boldsymbol{\theta}_{\mathbf{v}}\right)}{mean\left(\hat{\boldsymbol{\theta}}_{\mathbf{v}}\right)} \tag{5.15}$$

$$C_{gyro} = \frac{mean\left(\Delta\boldsymbol{\theta}_{\mathbf{v}}\right)}{mean\left(\hat{\boldsymbol{\theta}}_{\mathbf{v}}\right)} \tag{5.16}$$

After that, the correction factor is used to determine the new change in angular velocity after applying the correction factor, as defined in equation (5.17).

$$\Omega_n\left(k+1\right) = \left(1 - C_{gyro}\right)\Omega_o\left(k+1\right) \tag{5.17}$$

Finally, the variance factor is used to determine the variance for the gyroscope data around the z-axis, after the correction as defined in equation (5.18). The variables in equations (5.13) - (5.17) are listed in Table 5.3.

$$\sigma_{\Omega}^2\left(k+1\right) = \frac{k_{gyro}\Omega_n\left(k+1\right)}{dt} \tag{5.18}$$

In the covariance of the control input section, three variances are calculated, namely, the variance of the left encoder, right encoder, and gyroscope. The variances of the left and right encoders are dependent on the travel distance. For example, the further the vehicle travels, the left and right encoders' variances increase as a function of vehicle travel distance. The same thing is true for the variance of the gyroscope; the more the mobile vehicle turns, the variance of the gyroscope will increase as the vehicle turns. This is also true for any sensor that is not absolute, as discussed previously.

Table 5.4: Variable description for the output of the proposed feature extraction

Symbol	Unit	Description
$Z(k+1)$		Features from the proposed feature extraction at step $k+1$
$\mathbf{z}(k+1,1)$		First point features from the proposed feature extraction at step $k+1$
$\mathbf{z}(k+1,q)$		q^{th} point features from the proposed feature extraction at step $k+1$
$\mathbf{z}(k+1,n)$		Last point features from the proposed feature extraction at step $k+1$
q		Current point feature
n		Total number of point features
$z_\rho(k+1,q)$	cm	The distance between the mobile vehicle and the q^{th} point feature at step $k+1$ from the proposed feature extraction
$z_\theta(k+1,q)$	rad	The angle between the mobile vehicle and the q^{th} point feature at step $k+1$ from the proposed feature extraction

5.1.3 Proposed Feature Extraction

The proposed feature extraction output are points and line features. The point features from the proposed feature extraction are defined in equation (5.19) and described in Table 5.4. The point feature is defined as the distance and angle from the mobile vehicle position, instead of Cartesian coordinates since the data from the SONAR sensor are defined in term of distance and angle.

$$\begin{aligned} Z(k+1) &= \begin{bmatrix} \mathbf{z}(k+1,1) \dots \mathbf{z}(k+1,q) \dots \mathbf{z}(k+1,n) \end{bmatrix} \\ &= \begin{bmatrix} z_\rho(k+1,1) \dots z_\rho(k+1,q) \dots z_\rho(k+1,n) \\ z_\theta(k+1,1) \dots z_\theta(k+1,q) \dots z_\theta(k+1,n) \end{bmatrix} \end{aligned} \tag{5.19}$$

5.1.4 Observation Model

The observation model for the SONAR sensor is derived next. The observation model is derived for a point feature, such as a corner or an interior feature. First,

the distance from the mobile vehicle to the point feature is determined by using the Euclidean distance equation, which is given in equation (5.20).

$$\begin{aligned} d_{obs}\left(k+1,j\right) &= h_1\left(\mathbf{x_f}\left(k+1,j\right),\hat{\mathbf{x}}_\mathbf{v}\left(k+1\right)\right) \\ &= \sqrt{\left(x_f\left(k+1,j\right)-\hat{x}_v\left(k+1\right)\right)^2+\left(y_f\left(k+1,j\right)-\hat{y}_v\left(k+1\right)\right)^2} \end{aligned} \tag{5.20}$$

Next, the angle between the mobile vehicle and the point feature is determined by the angle between two points and the x-axis subtracted by the mobile vehicle orientation which is given by equation (5.21).

$$\begin{aligned} \theta_{obs}\left(k+1,j\right) &= h_2\left(\mathbf{x_f}\left(k+1,j\right),\hat{\mathbf{x}}_\mathbf{v}\left(k+1\right)\right) \\ &= \tan^{-1}\left(\frac{y_f\left(k+1,j\right)-\hat{y}_v\left(k+1\right)}{x_f\left(k+1,j\right)-\hat{x}_v\left(k+1\right)}\right)-\hat{\theta}_v\left(k+1\right) \end{aligned} \tag{5.21}$$

The observed distance and angle are stored in a vector, $\hat{z}\left(k+1,j\right)$, which is defined in equation (5.22).

$$\hat{\mathbf{z}}\left(k+1,j\right)=h\left(\mathbf{x_f}\left(k+1,j\right),\hat{\mathbf{x}}_\mathbf{v}\left(k+1\right)\right)=\begin{bmatrix} d_{obs}\left(k+1,j\right) \\ \theta_{obs}\left(k+1,j\right) \end{bmatrix} \tag{5.22}$$

After defining the distance and the angle between the mobile vehicle and the point features, the Jacobian is determined for the equations (5.20) and (5.21) with respect to the mobile vehicle position and point feature location, as shown in equation (5.23).

Table 5.5: Variable description for the observation model and data association sections

Symbol	Unit	Description
$\hat{x}_v\,(k+1)$	cm	Prediction of mobile vehicle x-position at step $k+1$
$\hat{y}_v\,(k+1)$	cm	Prediction of mobile vehicle y-position at step $k+1$
$\hat{\theta}_v\,(k+1)$	cm	Prediction of mobile vehicle orientation at step $k+1$
$d_{obs}\,(k+1,j)$	cm	Distance between the mobile vehicle and previously observed j^{th} point feature at step $k+1$
$\theta_{obs}\,(k+1,j)$	rad	Angle between the mobile vehicle and previously observed j^{th} point feature with respect to the x-axis minus the vehicle orientation
$\hat{\mathbf{z}}\,(k+1,j)$		Previously observed j^{th} point feature at step $k+1$
$H\,(k+1,j)$		Jacobian of the observation model at step $k+1$ for the j^{th} point features
$\mathbf{v}\,(k+1,j,q)$		Observation invocation at step $k+1$, the difference between j^{th} point feature previously observed and q^{th} point feature from the proposed feature extraction at step $k+1$
$S\,(k+1,j)$		Covariance of the observation invocation at step $k+1$ for the j^{th} point feature
$d_{mah}\,(k+1,j,q)$		Mahalanobis distance used in the nearest neighbour algorithm of the data association at step $k+1$ for the previously observed j^{th} point feature and q^{th} point feature from the proposed feature extraction
R		Covariance matrix of the measurement of the SONAR sensor
$\sigma^2_{s\rho}$	cm^2	Variance of the range reading from the SONAR sensor
$\sigma^2_{s\theta}$	rad^2	Variance of the angle from the SONAR sensor
$W\,(k+1,j)$		Kalman gain at step $k+1$ for the j^{th} point feature
$\hat{\mathbb{X}}\,(k+1)$		Prediction of the augmented vector of mobile vehicle position and feature locations at step $k+1$
$\mathbb{X}\,(k+1)$		Correction for the augmented vector of mobile vehicle position and feature locations after applying Kalman gain to it, at step $k+1$
$\hat{\mathbb{P}}\,(k+1)$		Prediction for the augmented matrix of covariance of mobile vehicle position and feature locations at step $k+1$
$\mathbb{P}\,(k+1)$		Correction for the augmented matrix of covariance of mobile vehicle position and feature locations after applying the Kalman gain to it, at step $k+1$

$$H(k+1,j) = \left[\begin{array}{ccc|cc} \frac{\partial h_1}{\partial \hat{x}_v(k+1)} & \frac{\partial h_1}{\partial \hat{y}_v(k+1)} & \frac{\partial h_1}{\partial \hat{\theta}_v(k+1)} & 0 & 0 \quad \dots \\ \frac{\partial h_2}{\partial \hat{x}_v(k+1)} & \frac{\partial h_2}{\partial \hat{y}_v(k+1)} & \frac{\partial h_2}{\partial \hat{\theta}_v(k+1)} & 0 & 0 \quad \dots \end{array}\right.$$
$$\left.\begin{array}{cccccccc} 0 & 0 & \frac{\partial h_1}{\partial x_f(k+1,j)} & \frac{\partial h_1}{\partial y_f(k+1,j)} & 0 & 0 & \dots & 0 \quad 0 \\ 0 & 0 & \frac{\partial h_2}{\partial x_f(k+1,j)} & \frac{\partial h_2}{\partial y_f(k+1,j)} & 0 & 0 & \dots & 0 \quad 0 \end{array}\right]$$

$$H(k+1,j) = \left[\begin{array}{ccc} -\frac{x_f(k+1,j)-\hat{x}_v(k+1)}{d_{obs}(k+1,j)} & -\frac{y_f(k+1,j)-\hat{y}_v(k+1)}{d_{obs}(k+1,j)} & 0 \\ \frac{y_f(k+1,j)-\hat{y}_v(k+1)}{d^2_{obs}(k+1,j)} & -\frac{x_f(k+1,j)-\hat{x}_v(k+1)}{d^2_{obs}(k+1,j)} & -1 \end{array}\right.$$
$$\begin{array}{cccccc} 0 & 0 & \dots & 0 & 0 & \frac{x_f(k+1,j)-\hat{x}_v(k+1)}{d_{obs}(k+1,j)} \\ 0 & 0 & \dots & 0 & 0 & -\frac{y_f(k+1,j)-\hat{y}_v(k+1)}{d^2_{obs}(k+1,1)} \end{array}$$
$$\left.\begin{array}{cccccc} \frac{y_f(k+1,j)-\hat{y}_v(k+1)}{d_{obs}(k+1,j)} & 0 & 0 & \dots & 0 & 0 \\ \frac{x_f(k+1,j)-\hat{x}_v(k+1)}{d^2_{obs}(k+1,j)} & 0 & 0 & \dots & 0 & 0 \end{array}\right] \quad (5.23)$$

The Jacobian of the observation model will be used next in the data association section. The variables used in equations (5.20) - (5.23) are listed in Table 5.5.

5.1.5 Data Association

Data association is a technique that is used to assign observed features to a previously observed feature or to classify them as new features. Some common problems with data association are as follows: *i*) assignment of an observed feature to an incorrect observed feature, which can cause the SLAM algorithm to fail; *ii*) observation of a new feature once and never observing it again; *iii*) and not observing

a particular feature at every time step. These challenges make data association more difficult, because incorrect assignment of features can cause the SLAM algorithm to fail. There are different types of data association algorithms, such as the following: *i*) known data association, where the features are known before SLAM is applied; *ii*) and unknown data association, where the features are unknown and are determined during the SLAM implementation. In this section, the unknown data association method known as "nearest neighbour", is explained next.

The unknown data association algorithm known as "nearest neighbour", consists of four different steps. These steps are as follows: *i*) loop over all unique features that have been obtained through the proposed feature extraction (for more details on the proposed feature extraction, the reader is referred to Chapter 3); *ii*) associate each feature to the closest associate, if it has been seen more than N times, where N is the threshold value; *iii*) and if it is less than N times, check if the feature is within the first validation gate, maximum distance for association, and increment the vote by 1 if it is within the first validation gate. Validation gate is an area used to check the measurement from the proposed feature extraction with the prediction from the observation model and remove unlikely measurement to keep track of the good measurement; *iv*) if the feature fails the first validation gate, assign the feature as a new feature, if it is below the second validation gate, the minimum distance for creation of new feature. The second validation gate is usually the size of the test environment or the expected range of the sensor, and it is much bigger than the first validation gate.

In the "nearest neighbour" algorithm, the Mahalanobis distance is used, which

is the distance between a point and a distribution. The Mahalanobis distance has the following characteristic: *i*) it considers that the variances in each direction to be different; *ii*) it considers covariance between variables; and *iii*) it is used to compare with the first and second validation gates. The Mahalanobis distance is defined in equations (5.24).

$$\begin{aligned} \mathbf{v}\left(k+1,j,q\right) &= \mathbf{z}\left(k+1,q\right) - \hat{\mathbf{z}}\left(k+1,j\right) \\ S\left(k+1,j\right) &= H\left(k+1,j\right)\mathbb{P}_{k+1}H^{T}\left(k+1,j\right) + R \\ d^{2}_{mah}\left(k+1,j,q\right) &= \mathbf{v}^{\ T}\left(k+1,j,q\right)S^{-1}\left(k+1,j\right)\mathbf{v}\left(k+1,j,q\right) \\ R &= \begin{bmatrix} \sigma^{2}_{s\rho} & 0 \\ 0 & \sigma^{2}_{s\theta} \end{bmatrix} \end{aligned} \tag{5.24}$$

For more information about the Mahalanobis distance, the reader is referred to Mahalanobis (1936); De Maesschalck, Jouan-Rimbaud, and Massart (2000). Additionally, information about the variables used in equations (5.24) may be found in Table 5.5.

5.1.6 EKF-SLAM

In this section, all of the pieces of the EKF-SLAM algorithm will be fused together, in order to present a full picture of how each part is integrated into EKF-SLAM. First, the vehicle model is needed to estimate the mobile vehicle position at step $k+1$. Through the use of the variance of the encoder and gyroscope, the control input covariance for the mobile vehicle can be determined at step $k+1$. After the mobile vehicle has at least 6 full SONAR scans, the proposed feature extraction is

used to determine unique features of the environment. Analysis of feature extraction through SONAR data is different than that used for laser data, because analysis of laser data can be performed after only 1 full scan. The reason behind this is that lasers have lower angular uncertainty, which is approximately 1^o, whereas SONAR sensors have angular uncertainty in the range between $22.5^o - 30^o$. After finding these unique features from the proposed feature extraction, data association is used to assign the feature to pre-existing features, a new features, or even delete the feature. The observation model is used within the data association. Next, the EKF will be applied (for more detail about EKF, the reader is referred to Chapter 1). The Kalman gain is given by equation (5.25).

$$W\left(k+1,j\right) \;=\; \hat{\mathbb{P}}\left(k+1\right) H^T\left(k+1,j\right) S^{-1}\left(k+1,j\right) \tag{5.25}$$

The Kalman gain is used with the predicted state vector, as well as with the predicted covariance matrix, to update the system state vector and covariance as given in equations (5.26) and (5.27). For the explanation of the variables used in equations (5.25) - (5.27), the reader is referred to Table 5.5.

$$\mathbb{X}\left(k+1\right) \;=\; \hat{\mathbb{X}}\left(k+1\right) + W\left(k+1,j\right)\mathbf{v}\left(k+1,j,q\right) \tag{5.26}$$

$$\mathbb{P}\left(k+1\right) \;=\; \hat{\mathbb{P}}\left(k+1\right) - W\left(k+1,j\right) S\left(k+1,j\right) W\left(k+1,j\right)^T \tag{5.27}$$

Finally, the data are augmented into $\mathbb{X}\left(k+1\right)$, the augmented vector which contains the mobile vehicle position and the feature locations at step $k+1$, and $\mathbb{P}\left(k+1\right)$, the augmented covariance matrix which contain the covariance of the

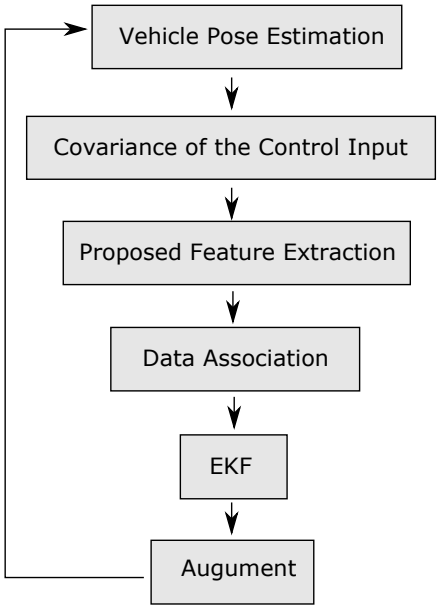


Figure 5.1: Flowchart of EKF-SLAM.

mobile vehicle position and the feature locations. The flowchart of EKF-SLAM is given in Figure 5.1.

5.1.7 Augmentation

In the previous section, the augmentation was mentioned but not described in detail. In this section, the augmentation process is explored further. After the EKF-SLAM update step, the new features that are accepted from the data association, $\mathbf{z}(k+1, q)$, are augmented into $\mathbb{X}(k+1)$ and $\mathbb{P}(k+1)$ at step $k+1$. The new feature positions are given by equation (5.28), which will be augmented to the end of $\mathbb{X}$.

Table 5.6: Variable description for the augment section

Symbol	Unit	Description
$\mathbf{x_{fn}}(k+1,q)$		The q^{th} new feature accepted from the data association at step $k+1$ which will be augmented into $\mathbb{X}$
$x_{fn}(k+1,q)$	cm	The q^{th} new feature x-location at step $k+1$
$y_{fn}(k+1,q)$	cm	The q^{th} new feature y-location at step $k+1$
$J_{fv}(k+1,q)$		Jacobian of the q^{th} new feature accepted from the data association with respect to mobile vehicle position at step $k+1$
$J_{ff}(k+1,q)$		Jacobian of the q^{th} new feature accepted from the data association with respect to the q^{th} new feature location at step $k+1$

$$\begin{aligned}\mathbf{x_{fn}}(k+1,q) &= g\left(\mathbf{x_v}(k+1),\mathbf{z}(k+1,q)\right)\\ &= \begin{bmatrix} x_v(k+1)+z_\rho(k+1,q)\cos\left(\theta_v(k+1)+z_\theta(k+1,q)\right)\\ y_v(k+1)+z_\rho(k+1,q)\sin\left(\theta_v(k+1)+z_\theta(k+1,q)\right)\end{bmatrix}\end{aligned} \tag{5.28}$$

To calculate the covariance of the new feature location with respect to the mobile vehicle position the Jacobian is calculated. The Jacobian of the new feature location with respect to the vehicle position is given in equation (5.29).

$$\begin{aligned}J_{fv}(k+1,q) &= \frac{\partial g\left(\mathbf{x_v}(k+1),\mathbf{z}(k+1,q)\right)}{\partial \mathbf{x_v}(k+1)}\\ &= \begin{bmatrix} \frac{\partial g_1}{\partial x_v(k+1)} & \frac{\partial g_1}{\partial y_v(k+1)} & \frac{\partial g_1}{\partial \theta_v(k+1)}\\ \frac{\partial g_2}{\partial x_v(k+1)} & \frac{\partial g_2}{\partial y_v(k+1)} & \frac{\partial g_2}{\partial \theta_v(k+1)}\end{bmatrix}\\ &= \begin{bmatrix} 1 & 0 & -z_\rho(k+1,q)\sin\left(\theta_v(k+1)+z_\theta(k+1,q)\right)\\ 0 & 1 & z_\rho(k+1,q)\cos\left(\theta_v(k+1)+z_\theta(k+1,q)\right)\end{bmatrix}\end{aligned} \tag{5.29}$$

Also, the Jacobian of the new feature location with respect to the new feature location is given by equation (5.30).

$$
\begin{aligned}
J_{ff}\left(k+1,q\right) &= \frac{\partial g\left(\mathbf{x_v}\left(k+1\right),\mathbf{z}\left(k+1,q\right)\right)}{\partial \mathbf{x_{fn}}\left(k+1,q\right)} \\
&= \begin{bmatrix} \dfrac{\partial g_1}{\partial x_{fn}\left(k+1,q\right)} & \dfrac{\partial g_1}{\partial y_{fn}\left(k+1,q\right)} \\ \dfrac{\partial g_2}{\partial x_{fn}\left(k+1,q\right)} & \dfrac{\partial g_2}{\partial y_{fn}\left(k+1,q\right)} \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
J_{ff}\left(k+1,q\right) = \begin{bmatrix} \cos\left(\theta_v\left(k+1\right)+z_\theta\left(k+1,q\right)\right) \\ \sin\left(\theta_v\left(k+1\right)+z_\theta\left(k+1,q\right)\right) \\ -z_\rho\left(k+1,q\right)\sin\left(\theta_v\left(k+1\right)+z_\theta\left(k+1,q\right)\right) \\ z_\rho\left(k+1,q\right)\cos\left(\theta_v\left(k+1\right)+z_\theta\left(k+1,q\right)\right) \end{bmatrix}
\end{aligned} \tag{5.30}
$$

Next, using the Jacobian calculated earlier in this section, the covariance for the new feature location with respect to mobile vehicle position is calculated in equation (5.31).

$$
P_{fv}\left(k+1\right) = J_{fv}\left(k+1,q\right)P_{vv}\left(k+1\right) \tag{5.31}
$$

The covariance of the mobile vehicle position with respect to the new feature location is given by equation (5.32), which is the transpose of equation (5.31).

$$
P_{vf}\left(k+1\right) = P_{fv}^{T}\left(k+1\right) \tag{5.32}
$$

Finally, the covariance of the new feature location with respect to the new feature location is given by equation (5.33).

$$P_{ff}(k+1) = J_{fv}(k+1,q)\,P_{vv}(k+1)\,J_{fv}^{T}(k+1,q) + J_{ff}(k+1,q)\,RJ_{ff}^{T}(k+1,q) \quad (5.33)$$

Equations (5.31) - (5.33) will be augmented to the covariance matrix, $\mathbb{P}(k+1)$ at step $k+1$. For the descriptions of the variables used in the equations from this section, reader are referred to Table 5.6.

5.2 Experimental Setup

The mobile vehicle setup is similar to the one used in Chapter 4, which consists of the following: *i*) two SONAR sensors, *ii*) servo motor, *iii*) four DC motors, *iv*) four encoders, *v*) Arduino uno and Arduino mega units, *vi*) two 12V batteries, *vii*) Inertial Measurement Unit (IMU) sensor (Adafruit 9-DOF IMU Breakout - L3GD20H + LSM303) for tri-axial accelerometer, gyroscope, and magnetometer measurement, *viii*) two battery switches, *ix*) Xbee wireless communication, *x*) SD card for storing encoder, IMU sensor, and SONAR data, and *xi*) case with 2 pivoted windows. The mobile vehicle went through 3 minor modifications which consisted of the following: *i*) replacement of the two $9V$ batteries with one $12V$ battery (allowing for longer test times, above 2 hours), *ii*) reduction of the number of switches from 3 switches to 2 switches, and finally *iii*) addition of a cover with two pivoted windows to the mobile vehicle (the cover was added to keep the electronics safe from powder and liquids which were used to induce slipping). The new mobile vehicle configuration is shown in Figure 5.2.

The environmental setup consist of four walls, which are constructed from plexiglass material as shown in Figure 5.3. Three different environmental setups were tested in this chapter. First, an environment with no alteration to the floor. Second, an environment where a portion of the floor was altered by adding dry powder to it. Finally, an environment where a portion of the floor was altered by adding a special liquid know as NeverWet. The dry powder and NeverWet liquid caused the mobile vehicle to slip. The three different environmental setups are shown in Figure 5.3.

For the three different environmental setups, the mobile vehicle is placed inside the closed rectangular environment. The mobile vehicle is taken to a new location in approximately 15 to 30 cm increments, and during this travel, data is collected from the 4 encoders and the IMU sensor. After each travel increment, the mobile vehicle is brought to a complete stop and SONAR data are collected from two SONAR sensors attached to the servo motor. The servo motor is rotated 180^o in 1^o increments to perform a full 360^o scan, since the two SONAR sensors are attached to scan in opposite directions. After completing the SONAR scan, the mobile vehicle is kept stationary for 4 seconds. The reason for the 4 seconds pause after completing the SONAR scan is to reset the drift value of the gyroscope; for more details about the drift value of the gyroscope, readers are referred to Chapter 4. Then, the process is repeated a certain number of times to collect more data from the environment.

After performing the test three times with the different environmental setups, the data are analyzed by using MATLAB for offline analysis of the EKF-SLAM.

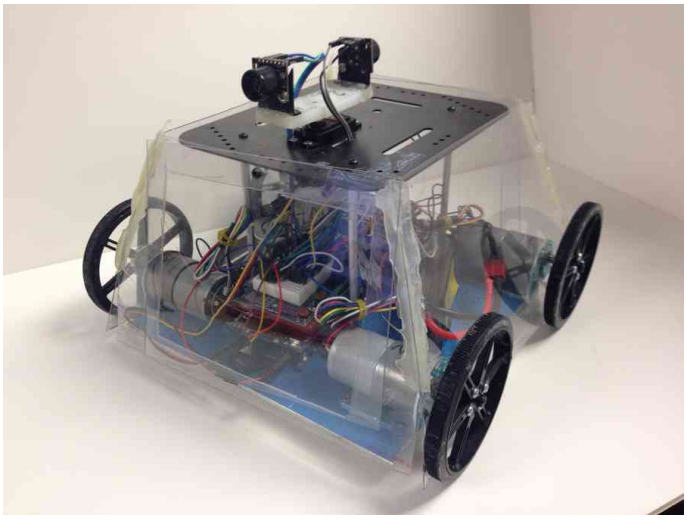


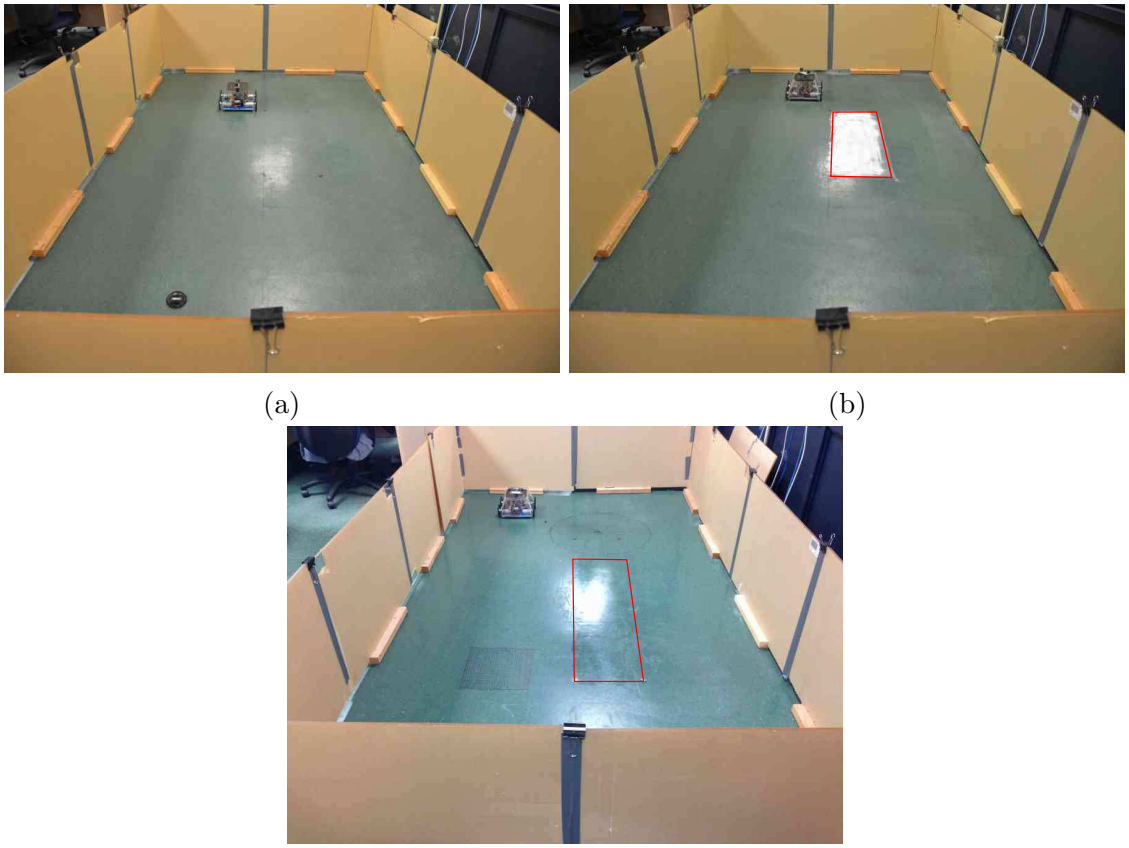
Figure 5.2: Mobile vehicle setup used in the experiments.

5.3 Experimental Results

In this section, the results obtained for three different environmental setups will be presented. First, the environment with no alteration to the floor is examined; this is known as the default case. Next, the environment with an addition of dry powder (talcum powder) to a portion of the floor is examined; this is known as the dry powder case. Finally, the environment with addition of hydrophobic coating (NeverWet liquid) to a portion of the floor is examined; this is known as the hydrophobic coating case. The dry powder and hydrophobic coating were added to induce slipping.

5.3.1 Default Case

The default case is the case where the floor of the environment is not altered, as shown in Figure 5.3a. Basically, the same raw data from Chapter 4 for the environment with no interior features is used here. In Chapter 4, the focus was on



(a) (b)

(c)

Figure 5.3: Experimental arrangements with and without alteration to floor. a) Default case: no alteration to floor; b) Dry powder case: addition of powder to a portion of the floor; c) Hydrophobic coating case: addition of NeverWet liquid to a portion of the floor. The dimension of the environment is as follows: width = 182.0 cm, length 335.1 cm, and height = 63.5 cm.

the vehicle pose estimation and feature extraction, but without the use of the SLAM algorithm. In this chapter, the use of the SLAM algorithm will be examined and will be compared to the case where SLAM was not used. During SLAM analysis, 3 different cases are introduced where the value of standard deviation of SONAR sensor range, $\sigma_{s\rho}$, is changed, for example, 5, 10, and 15 cm. The standard deviation of SONAR sensor angle, $\sigma_{s\theta}$, was held constant at 4^o. All the 3 cases for SLAM for the default environment are compared to the case without SLAM.

To understand the process as to how the EKF-SLAM algorithm worked, Figure B.1, B.2, and B.3 are used to detail the steps when the EKF-SLAM updates were carried out. First, the mobile vehicle is moved and is used to take full SONAR scan in 6 new positions. Next, the proposed feature extraction procedure is used with the SONAR data in order to extract point or line features. After, the mobile vehicle is moved to a new position, where a full SONAR scan is to be performed. After that, the proposed feature extraction is carried out with all SONAR data except the first one, and point and line features are extracted. Next, EKF-SLAM is applied. After that, the mobile vehicle is moved to a new position where a full SONAR scan is carried out. After that, the proposed feature extraction is run on all SONAR data except the first 2 full SONAR scans data. Then, the EKF-SLAM is applied on the data. Finally, the process is repeated N times, to capture the most unique features from the environment and update mobile vehicle positions and feature locations through the use of EKF-SLAM. Also, in Figure 5.4, the results obtained during the first, middle, and final steps of EKF-SLAM implementation are shown. For example, in Figure 5.4, for the data associated with labels 11 through 16, the mobile vehicle is brought to a stop at each of the locations numbered 11 to 16 and a total of 6 full scans are taken. The raw SONAR data are sent to the proposed feature extraction algorithm and the results obtained are shown in Figure 5.4 with the qualifier "processed". For the results corresponding to locations between 1 through 26 the plots are given in Appendix B as Figures B.1, B.2, and B.3.

After updating the mobile vehicle positions through EKF-SLAM, the raw

SONAR data for the 3 different SLAM cases as well as the case without SLAM are shown in Figure 5.5. The mobile vehicle paths for the 3 SLAM cases have been adjusted and the raw SONAR data can be seen to be more confined to the environment. The processed feature extraction data for the 3 different SLAM cases as well as the case without SLAM are shown in Figure 5.6. To compare the SLAM and no-SLAM cases, the absolute difference between calculated corners and true corners as well as absolute difference between calculated lines and true lines are determined and summarized in Tables C.1 and C.2. In addition, a graphical representation of the table data is provided in Figure 5.7. The SLAM case for $\sigma_{s\rho} = 15$ cm showed a minor improvement over the other 2 SLAM cases, as well as over the no-SLAM case, when compared to the point features from Figure 5.7. It was noticed from Figure 5.7 that the SLAM cases when $\sigma_{s\rho} = 15$ cm and $\sigma_{s\rho} = 10$ cm performed better than the other SLAM case, when $\sigma_{s\rho} = 5$ cm as well as over the no-SLAM case. In conclusion, for the default case, the case with SLAM performed slightly better than the no-SLAM case. This could be traced back to the proposed feature extraction algorithm, which is robust enough to reject false features due to the large data gathered from the SONAR sensor and the large threshold value used in the different part of the proposed feature extraction. For further details about the proposed feature extraction, readers are referred to Chapter 3.

5.3.2 Dry Powder Case

The dry powder case is the case where the floor of the environment is altered by adding powder (talcum powder) to a portion of the floor, as shown in Figure 5.3b. The results are compared for the SLAM and no-SLAM cases. Similar to the default case, $\sigma_{s\rho}$ values are changed, with values considering being 5, 10, and 15 cm. However, the $\sigma_{s\theta}$ value is held constant, 4^o, for the SLAM analysis. Results from all three SLAM cases for the dry powder environment were compared to those obtained for the no-SLAM case.

As in the default case, the EKF-SLAM update steps for $\sigma_{s\rho} = 5$ cm are shown in Figure 5.8. In Figure 5.8, the author shows the first, middle, and last steps of the EKF-SLAM implementation. The raw SONAR data for the 3 different SLAM cases as well as that obtained for the no-SLAM case is shown in Figure 5.9; the mobile vehicle paths for the 3 SLAM cases were adjusted and the raw SONAR data can be seen to be more confined to the environment for one of the SLAM cases, when $\sigma_{s\rho} = 15$ cm. The results for the other 2 SLAM cases started to deviate from the true position and the raw SONAR data can be seen to be outside the environment at the end of the experiment as shown in Figure 5.9. The no-SLAM case performed better than 2 of the SLAM cases, when $\sigma_{s\rho} = 5$ cm and $\sigma_{s\rho} = 10$ cm, but the final SLAM case, $\sigma_{s\rho} = 15$ cm, performed the best. The processed feature extraction data for the 3 different SLAM cases as well as that for the no-SLAM case are shown in Figure 5.10.

To compare SLAM and no-SLAM cases, the absolute difference between calcu-

lated corners and true corners as well as the absolute difference between calculated lines and true lines are determined and summarized in Tables C.3 and C.4. In addition, a graphical representation of the table data is provided in Figure 5.11. All 3 SLAM cases had better results than the no-SLAM case. The SLAM case when $\sigma_{s\rho} = 15$ cm performed the best against the other two SLAM cases as well as against the no-SLAM case. Also, it is noticed from Figure 5.11 that the absolute difference for the SLAM cases when $\sigma_{s\rho} = 15$ cm had θ_{line} below 3.1^o, whereas the no-SLAM case had θ_{line} below 6^o. Overall, even though the SLAM case performed much better the no-SLAM case, the no-SLAM case gave an acceptable result through the use of the proposed feature extraction.

5.3.3 Hydrophobic Coating Case

The hydrophobic coating case is the case where the floor of the environment is altered by adding NeverWet liquid to a portion of the floor, as shown in Figure 5.3c. The results are compared for SLAM and no-SLAM cases. During SLAM analysis, the 3 different cases are introduced where the value of $\sigma_{s\rho}$ is changed, for example, 5, 10, and 15 cm, but the $\sigma_{s\theta}$ value is held constant, 4^o. Results obtained for all the 3 SLAM cases with the hydrophobic coating environment are compared to the results obtained for the no-SLAM case.

As in the default and dry powder cases, the EKF-SLAM update steps for $\sigma_{s\rho} = 5$ cm are shown in Figure 5.12. In Figure 5.12, the author shows the first, middle, and last steps of the EKF-SLAM implementation. The raw SONAR data

for the 3 different SLAM cases as well as the no-SLAM case are shown in Figure 5.13; the mobile vehicle paths for the 3 SLAM cases are adjusted and the raw SONAR data can be seen to be more confined to the environment for two of the SLAM cases when $\sigma_{s\rho} = 15$ cm and when $\sigma_{s\rho} = 10$ cm. The result for the other SLAM case started to deviate from the true position and the raw SONAR data can be seen to be outside the environment at the end of the experiment. The no-SLAM case performed better than one of the SLAM cases; that is when $\sigma_{s\rho} = 5$ cm, but the final SLAM case $\sigma_{s\rho} = 15$ cm was noted to performed the best. The proposed feature extraction data for the 3 different SLAM cases as well as the no-SLAM case are shown in Figure 5.14.

To compare SLAM and no-SLAM cases, the absolute difference between calculated corners and true corners as well as the absolute difference between calculated lines and true lines are determined and summarized in Tables C.5 and C.6. In addition, a graphical representation of the table data is provided in Figure 5.15. All 3 SLAM cases had better results than the no-SLAM case. The SLAM case when $\sigma_{s\rho} = 15$ cm performed the best against the other two SLAM cases as well as against the no-SLAM case. Also, it is noticed from Figure 5.15 that the SLAM cases when $\sigma_{s\rho} = 15$ cm had major improvements; for example, the highest absolute difference for line feature θ_{line} for SLAM case was below 3^o, whereas for the no-SLAM case the highest absolute difference was below 6^o. Also, the absolute difference for the SLAM case for ρ_{line} was below 15 cm whereas for the no-SLAM case was below 35 cm. Overall, the dry powder and hydrophobic coating cases induced slippage which challenged the no-SLAM implementation, but the SLAM case was able to perform

and maintain a good map representation as well as good vehicle path correction.

5.4 Summary

The mobile vehicle travelled in three different environmental configurations *i*) default, *ii*) dry powder, and *iii*) Hydrophobic coating. For each environmental configuration, the SLAM algorithm for 3 different cases is studied and the respective performance is compared to that from a no-SLAM case. The SLAM case with $\sigma_{s\rho} = 15$ cm and $\sigma_{s\theta} = 4^o$ was noted to perform the best compared to the other two SLAM cases as well as the no-SLAM case for all 3 different environments studied in this chapter. The improvement of the SLAM case when $\sigma_{s\rho} = 15$ and $\sigma_{s\theta} = 4^o$ was noticed in the corner as well as in the determined line which are the results of the proposed feature extraction. Basically, this study determined a good starting point for values to select for $\sigma_{s\rho} = 4^o$ and $\sigma_{s\theta}$ to get a good environmental estimate using the proposed feature extraction algorithm. The value of $\sigma_{s\rho}$ and $\sigma_{s\theta}$ are different than the values from $\sigma_{s\rho}$ and $\sigma_{s\theta}$ from sensor data sheets. The algorithm goes through steps to improve the uncertainty in SONAR data. This improvement can be noticed through the selection of a smaller value of $\sigma_{s\theta} = 4^o$ rather than the normal $\sigma_{s\theta}$ from the SONAR sensor data sheet, which is $\sigma_{s\theta} = 22.5^o - 30^o$. Another point to note; the proposed feature extraction algorithm makes use of the TBF algorithm with a high enough threshold value, $n_t = 50$, so that most of the outliers can be removed, for instance, if a poor feature extraction was determined from the feature extraction algorithm. Through the use of a TBF step, less emphasis or total rejection can be

noticed from the TBF algorithm results.

Through this study, the use of EKF-SLAM with the proposed feature extraction has been performed successfully. The mobile vehicle path was adjusted and the SONAR data was found to be more confined to the environment, as shown in Figure 5.5, 5.9, and 5.13. The proposed feature extraction approach reduced the angular uncertainty in SONAR as seen by selection of a lower angular uncertainty in the $\sigma_{s\theta}$ value with the SLAM algorithm.

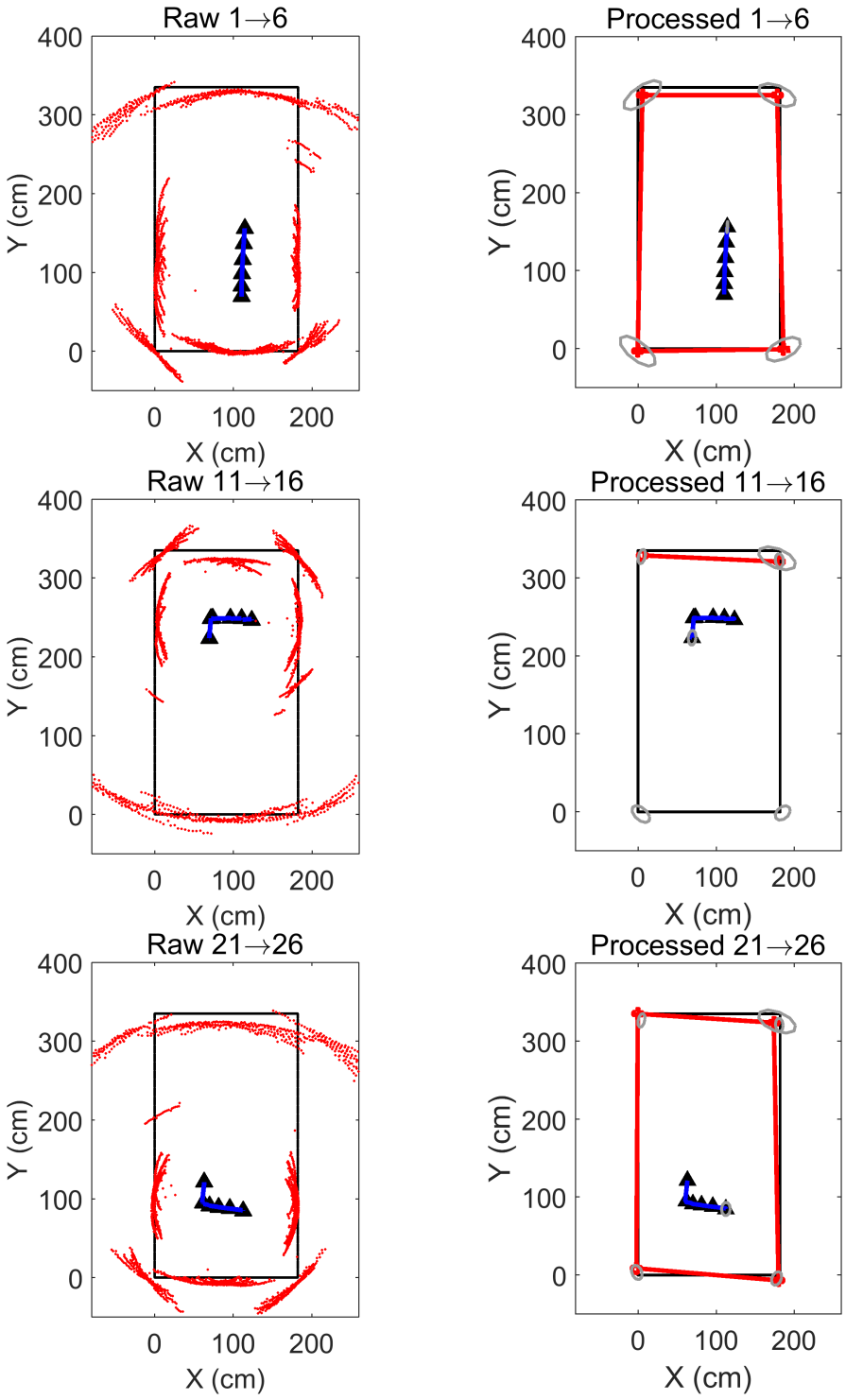


Figure 5.4: Default case: Raw and processed results for the default environment case with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$ for 3 different stops $1-6$, $11-16$, and $21-26$. The covariance of the current mobile vehicle position and the covariance of the feature locations are represented by gray ellipses; the mobile vehicle positions where the SONAR scans occurred are represented by black triangles; the mobile vehicle path is represented by dark blue lines; raw SONAR data are represented by red points. The rectangular environment contour is superimposed for comparison purposes only.

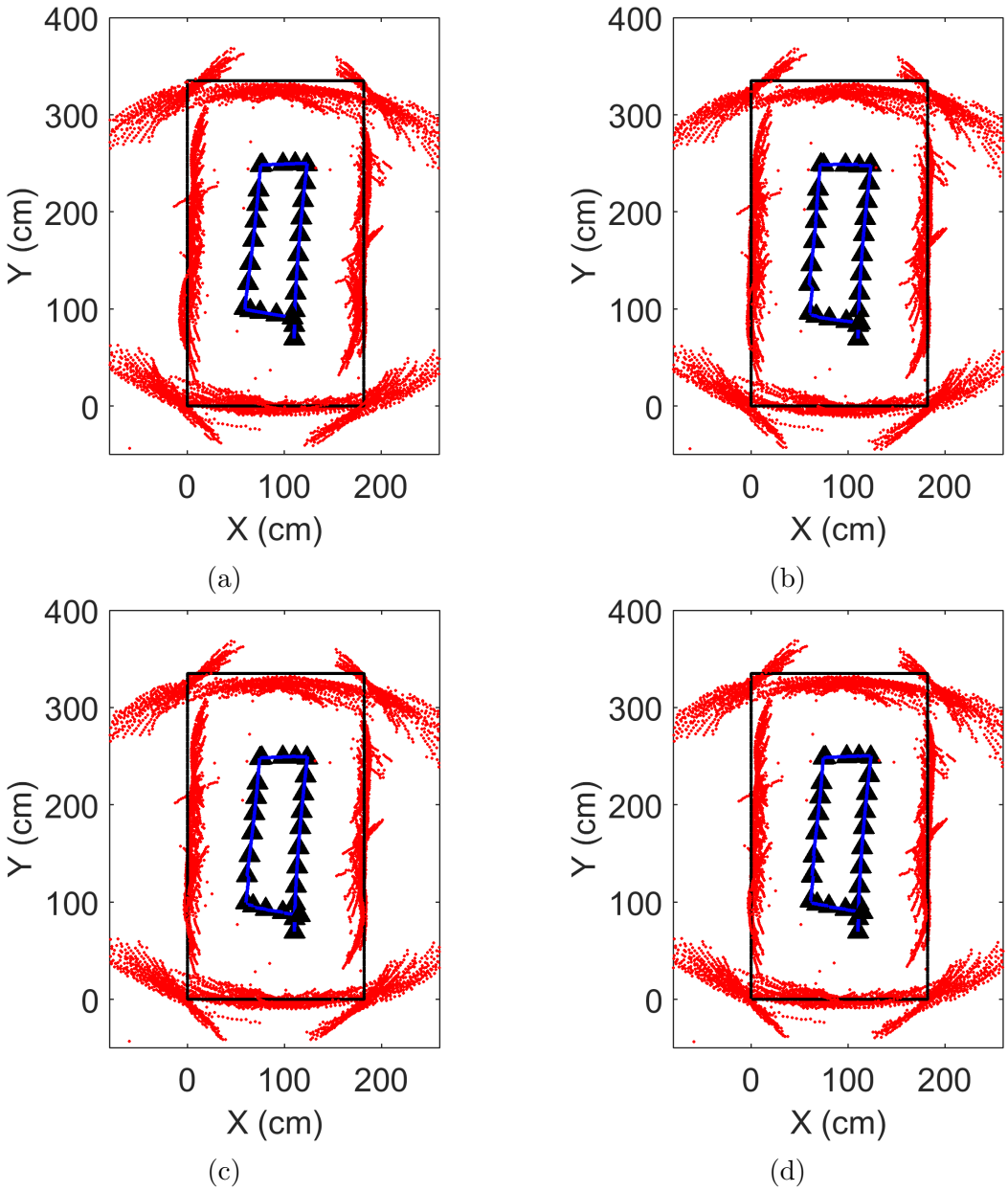


Figure 5.5: Default case: Raw SONAR data for no-SLAM and SLAM cases. The mobile vehicle positions where the SONAR scans occurred are represented by black triangles; the mobile vehicle path is represented by dark blue lines; raw SONAR data are represented by red points. The rectangular environment contour is superimposed for comparison purposes only. a) No-SLAM, b) SLAM, with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$, c) SLAM, with $\sigma_{s\rho} = 10$ cm and $\sigma_{s\theta} = 4^o$, d) SLAM, with $\sigma_{s\rho} = 15$ cm and $\sigma_{s\theta} = 4^o$.

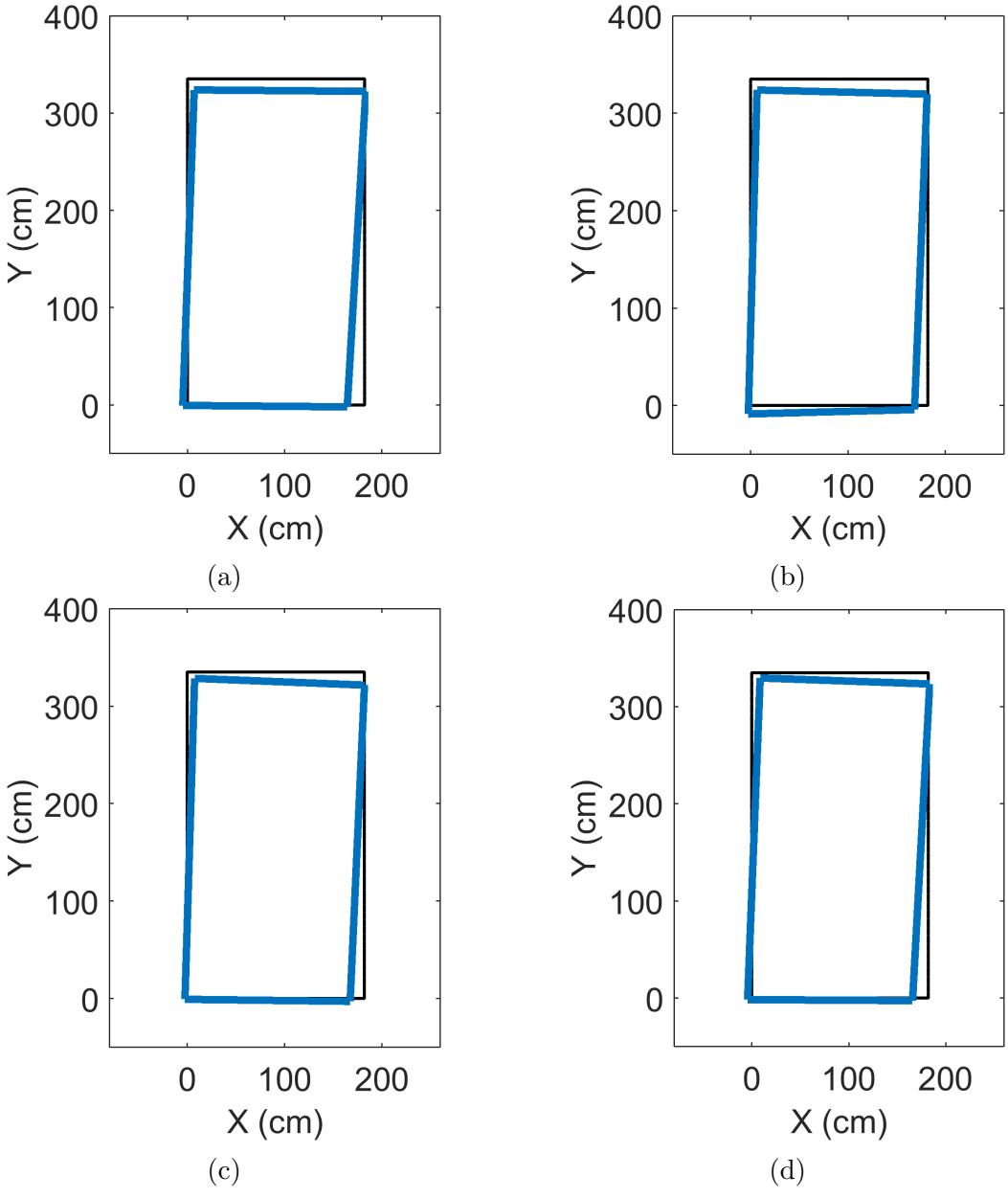


Figure 5.6: Default case: Processed feature extraction results for no-SLAM and SLAM cases. The estimated environment is represented by dark blue lines, and the rectangular environment contour is superimposed for comparison purposes only. a) No-SLAM, b) SLAM, with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$, c) SLAM, with $\sigma_{s\rho} = 10$ cm and $\sigma_{s\theta} = 4^o$, d) SLAM, with $\sigma_{s\rho} = 15$ cm and $\sigma_{s\theta} = 4^o$.

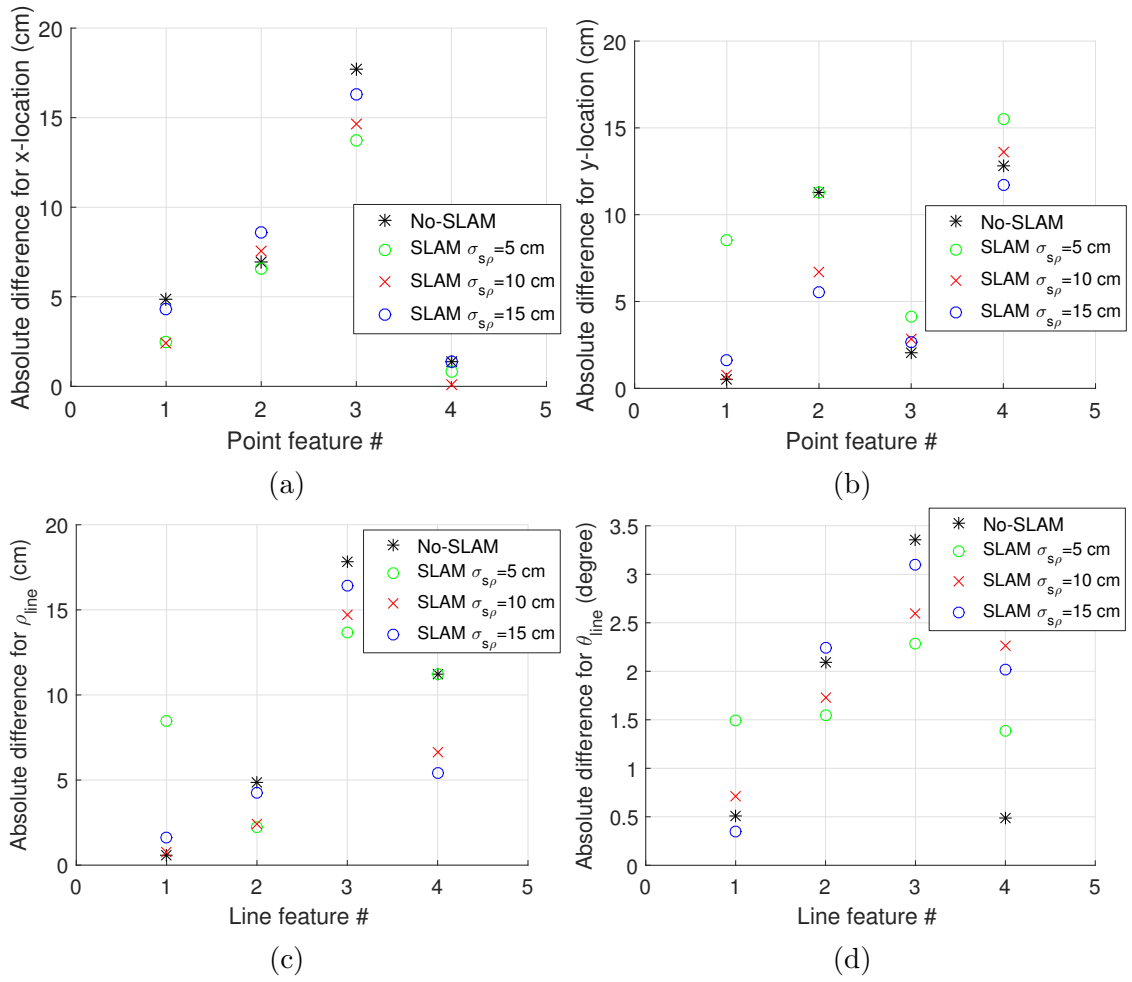


Figure 5.7: Default case: The absolute difference between calculated corners and true corners, as well as the absolute difference between calculated lines and true lines, are determined for SLAM and no-SLAM cases. a) The absolute difference between corner features in the x-direction with and without SLAM compared to the true values, b) The absolute difference between corner features in the y-direction with and without SLAM compared to the true values, c) The absolute difference between line features, ρ_{line}, with and without SLAM compared to the true values, and d) The absolute difference between line features, θ_{line}, with and without SLAM compared to the true values.

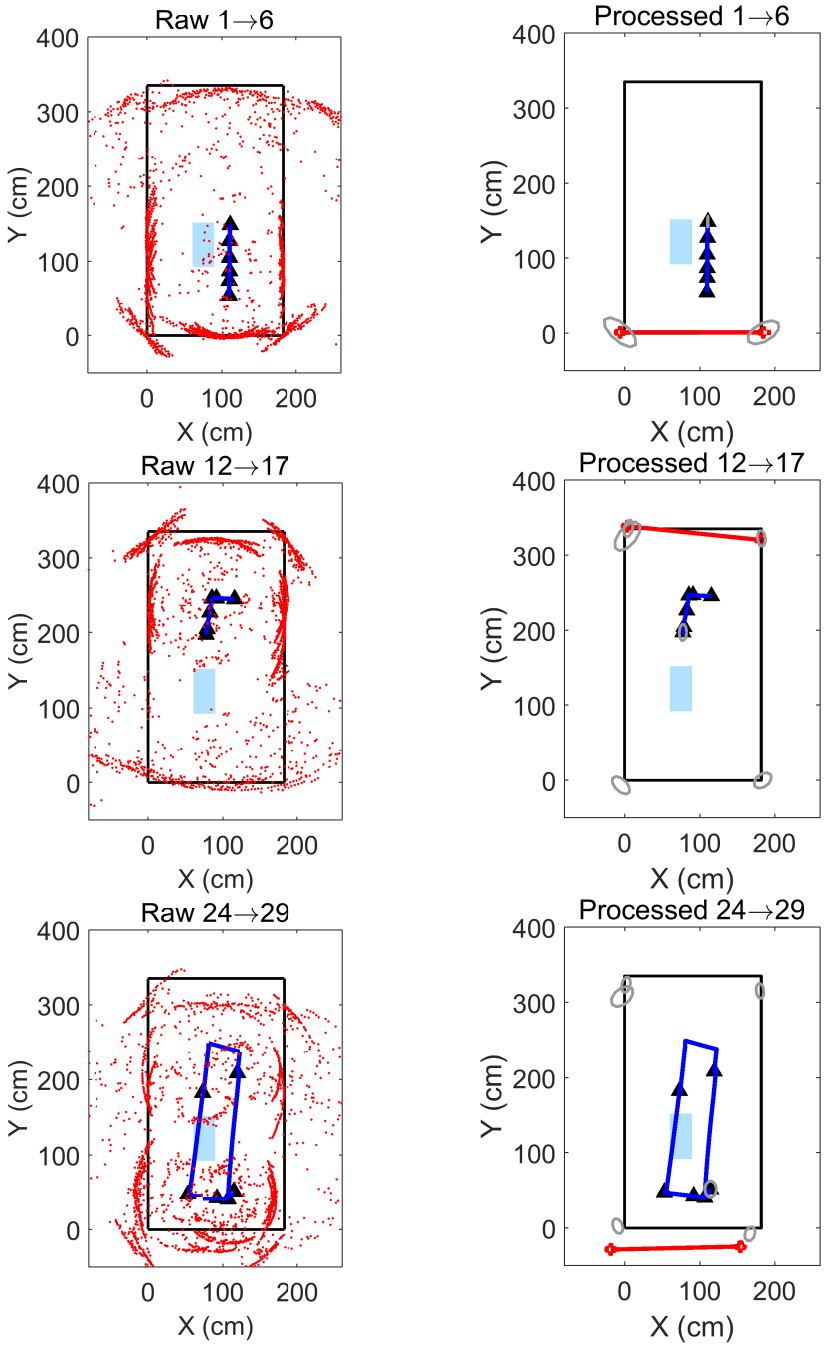


Figure 5.8: Dry powder case: Raw and processed results for the dry powder environment case for $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$ for 3 different stop ranges, specifically $1-6$, $12-17$, and $24-29$. The covariance of the current mobile vehicle position and the covariance of the feature locations are represented by gray ellipses; SONAR data are represented by red dots; the mobile vehicle positions where the full SONAR scans occurred are represented by black triangles; the path of the mobile vehicle is represented by dark blue lines. The slippage area (dry powder), represented by a light blue rectangle, and the rectangular environment, represented by black lines, are superimposed over the data for comparison purposes only.

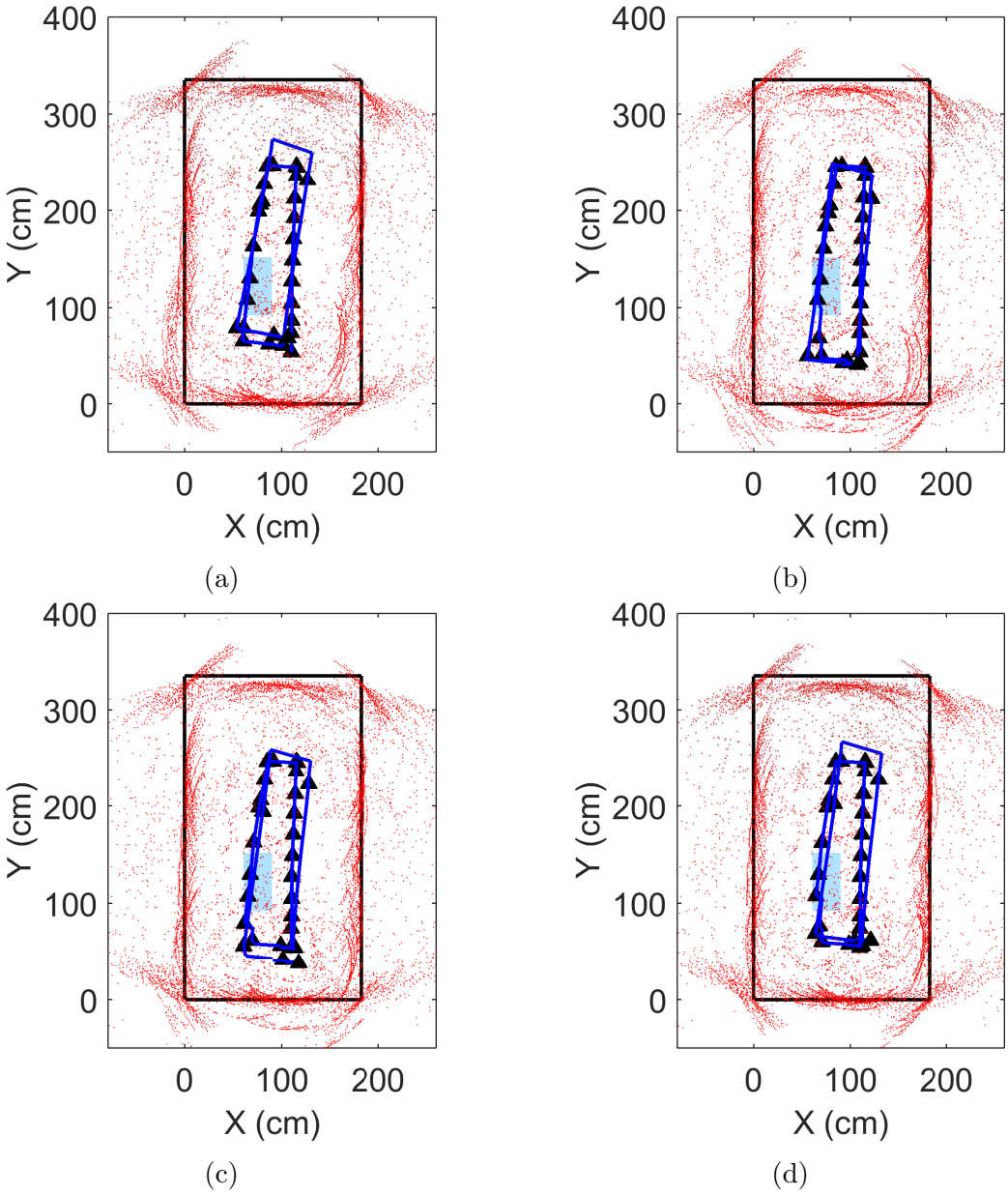


Figure 5.9: Dry powder case: Raw SONAR data for no-SLAM and SLAM cases for the dry powder environment. The mobile vehicle positions where the SONAR scans occurred are represented by black triangles; the mobile vehicle path is represented by dark blue lines; raw SONAR data are represented by red points. The slippage area (dry powder), represented by a light blue rectangle, and the rectangular environment, represented by black lines, are superimposed over the data for comparison purposes only. a) No-SLAM, b) SLAM, with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$, c) SLAM, with $\sigma_{s\rho} = 10$ cm and $\sigma_{s\theta} = 4^o$, d) SLAM, with $\sigma_{s\rho} = 15$ cm and $\sigma_{s\theta} = 4^o$.

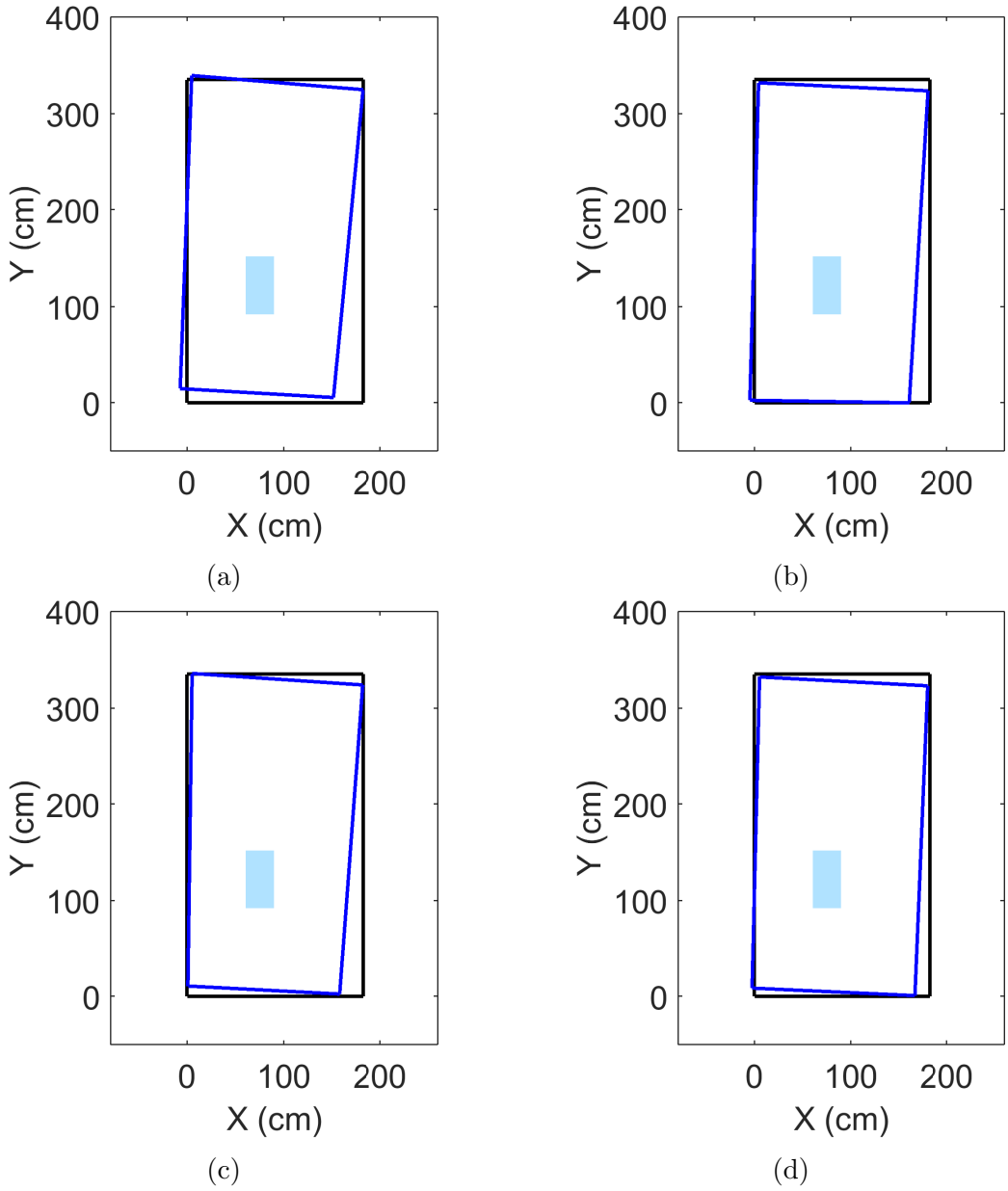


Figure 5.10: Dry powder case: Processed feature extraction results for no-SLAM and SLAM cases. The estimated environment is represented by dark blue lines. The slippage area (dry powder), represented by a light blue rectangle, and the rectangular environment, represented by black lines, are superimposed over the data for comparison purposes only. a) No-SLAM, b) SLAM, with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$, c) SLAM, with $\sigma_{s\rho} = 10$ cm and $\sigma_{s\theta} = 4^o$, d) SLAM, with $\sigma_{s\rho} = 15$ cm and $\sigma_{s\theta} = 4^o$.

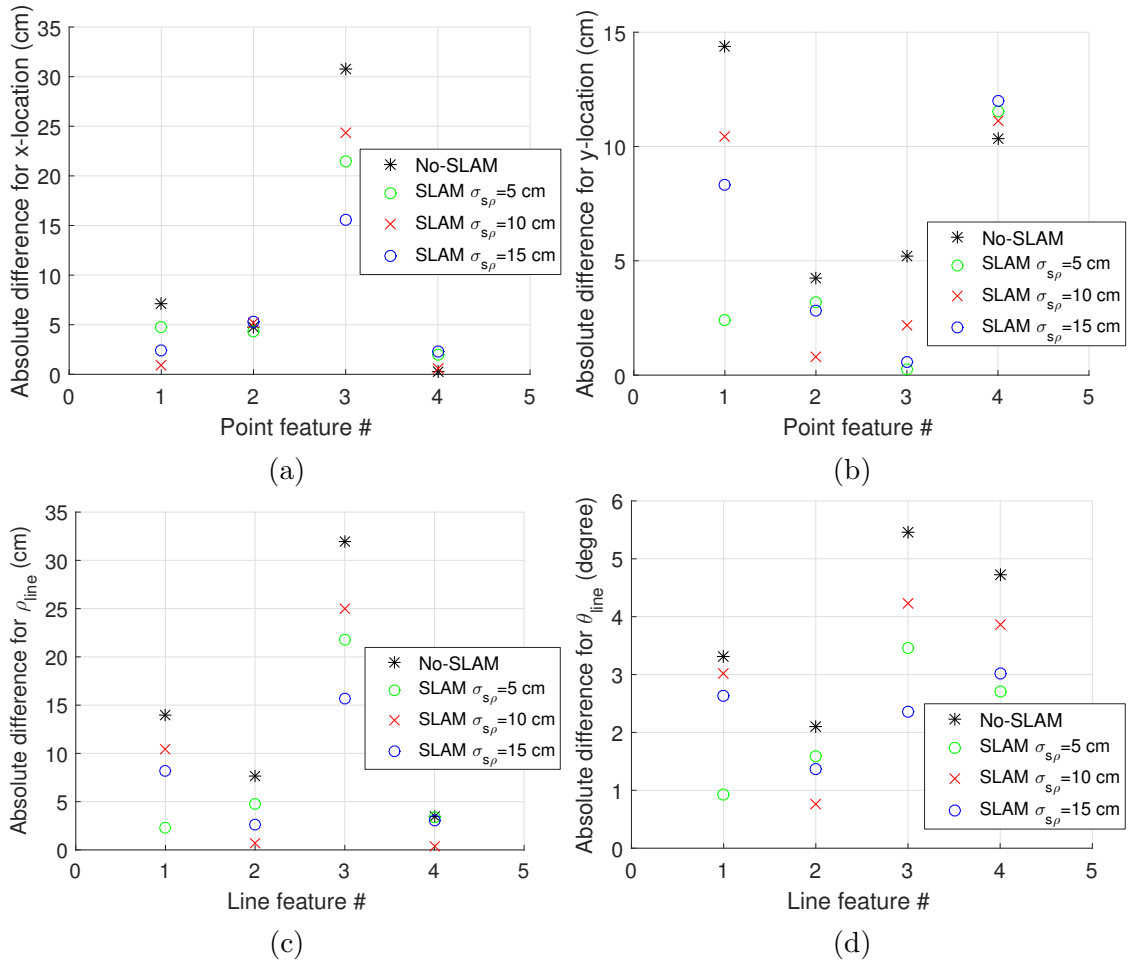


Figure 5.11: Dry powder case: The absolute difference between calculated corners and true corners, as well as the absolute difference between calculated lines and true lines, are determined for SLAM and no-SLAM cases. a) The absolute difference between corner features in the x-direction with and without SLAM compared to the true values, b) The absolute difference between corner features in the y-direction with and without SLAM compared to the true values, c) The absolute difference between line features, ρ_{line}, with and without SLAM compared to the true values, and d) The absolute difference between line features, θ_{line}, with and without SLAM compared to the true values.

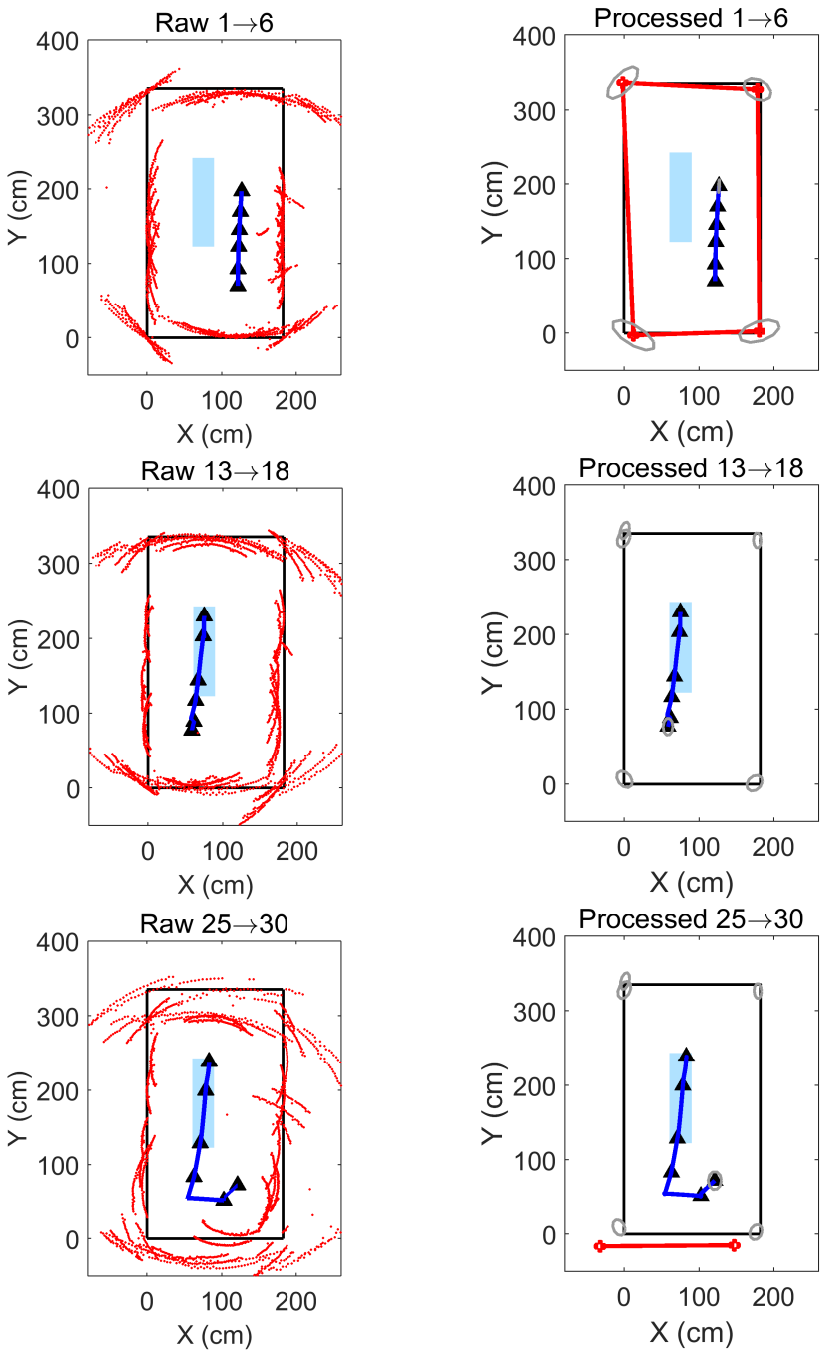


Figure 5.12: Hydrophobic coating case: Raw SONAR data for no-SLAM and SLAM cases for the hydrophobic coating environment. The covariance of the current mobile vehicle position and the covariance of the feature locations are represented by gray ellipses; the mobile vehicle positions where the SONAR scans occurred are represented by black triangles; the mobile vehicle path is represented by dark blue lines; raw SONAR data are represented by red points. The slippage area (hydrophobic coating), represented by a light blue rectangle, and the rectangular environment, represented by black lines, are superimposed over the data for comparison purposes only. a) No-SLAM, b) SLAM, with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$, c) SLAM, with $\sigma_{s\rho} = 10$ cm and $\sigma_{s\theta} = 4^o$, d) SLAM, with $\sigma_{s\rho} = 15$ cm and $\sigma_{s\theta} = 4^o$.

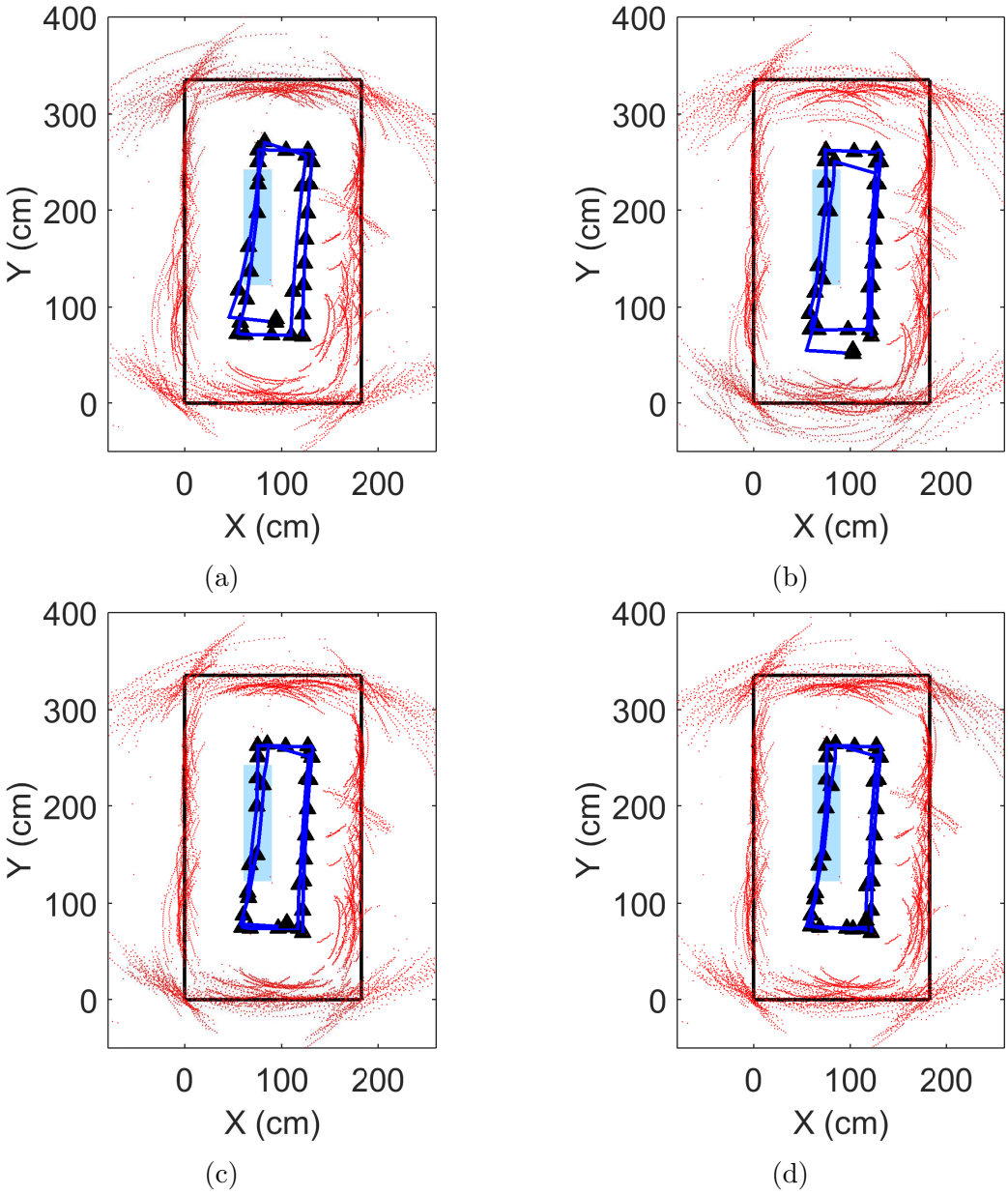


Figure 5.13: Hydrophobic coating case: Raw SONAR data for no-SLAM and SLAM cases for the hydrophobic coating environment. The mobile vehicle positions where the SONAR scans occurred are represented by black triangles; the mobile vehicle path is represented by dark blue lines; raw SONAR data are represented by red points. The slippage area (hydrophobic coating), represented by a light blue rectangle, and the rectangular environment, represented by black lines, are superimposed over the data for comparison purposes only. a) No-SLAM, b) SLAM, with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$, c) SLAM, with $\sigma_{s\rho} = 10$ cm and $\sigma_{s\theta} = 4^o$, d) SLAM, with $\sigma_{s\rho} = 15$ cm and $\sigma_{s\theta} = 4^o$.

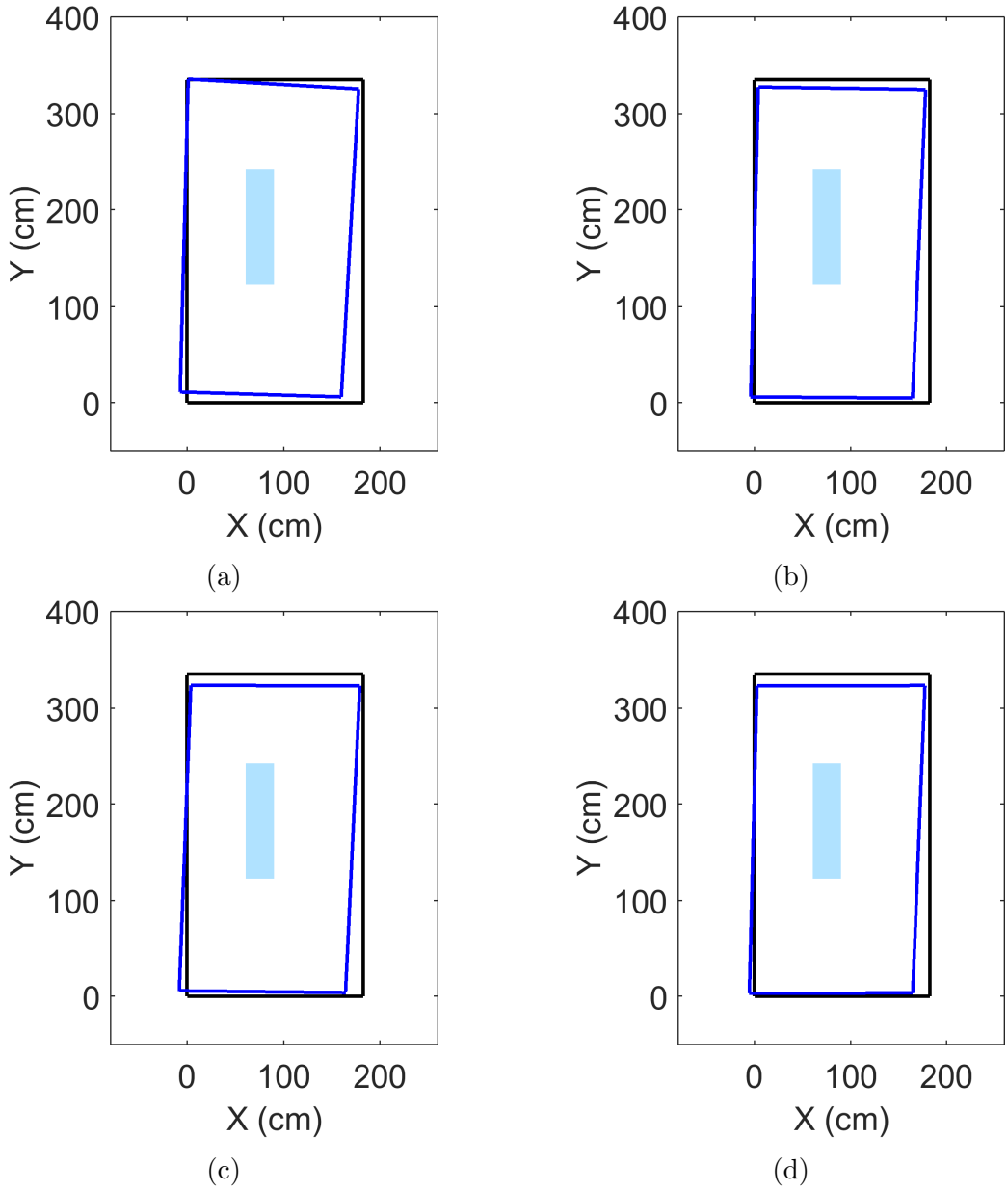


Figure 5.14: Hydrophobic coating case: Processed feature extraction results for no-SLAM and SLAM cases. The estimated environment is represented by dark blue lines. The slippage area (hydrophobic coating), represented by a light blue rectangle, and the rectangular environment, represented by black lines, are superimposed over the data for comparison purposes only. a) No-SLAM, b) SLAM, with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$, c) SLAM, with $\sigma_{s\rho} = 10$ cm and $\sigma_{s\theta} = 4^o$, d) SLAM, with $\sigma_{s\rho} = 15$ cm and $\sigma_{s\theta} = 4^o$.

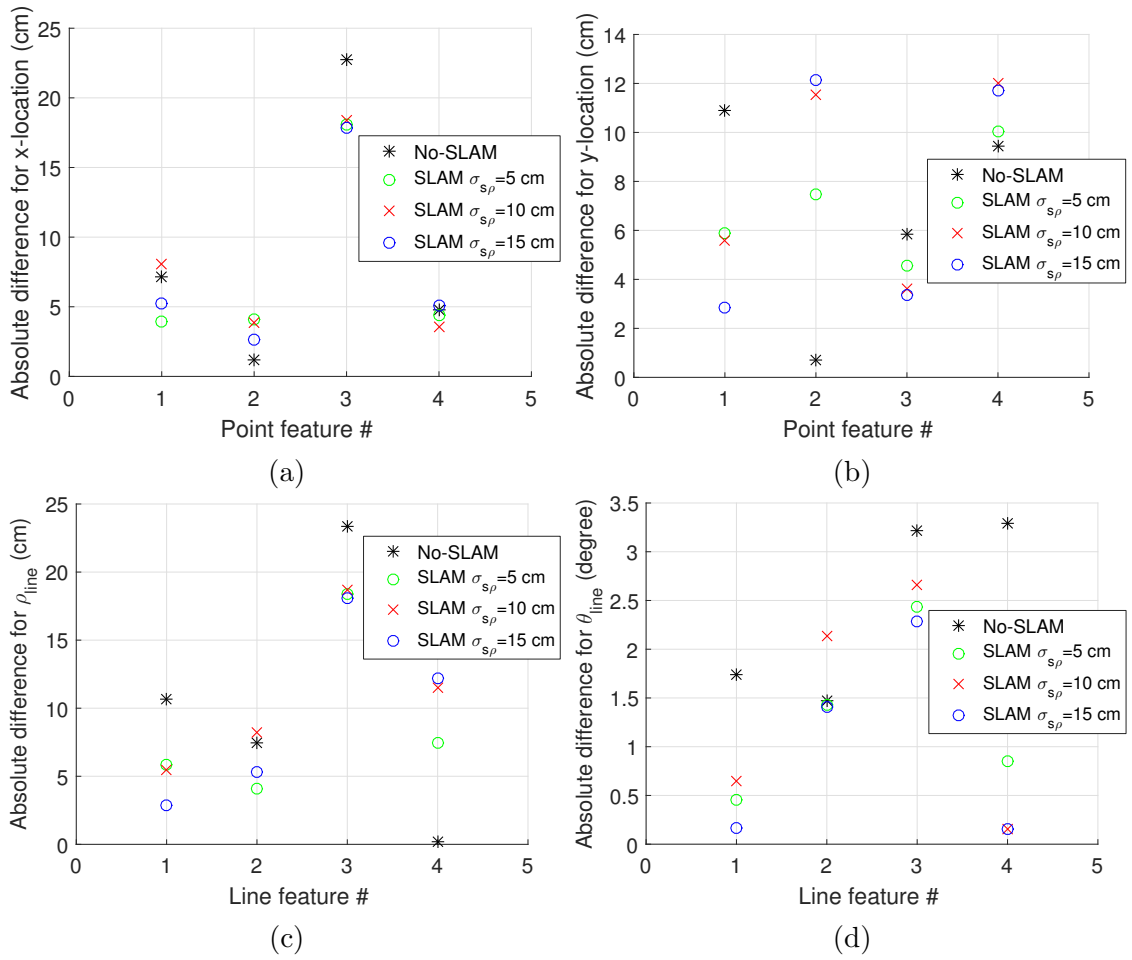


Figure 5.15: Hydrophobic coating case: The absolute difference between calculated corners and true corners, as well as the absolute difference between calculated lines and true lines, are determined for SLAM and no-SLAM cases. a) The absolute difference between corner features in the x-direction with and without SLAM compared to the true values, b) The absolute difference between corner features in the y-direction with and without SLAM compared to the true values, c) The absolute difference between line features, ρ_{line}, with and without SLAM compared to the true values, and d) The absolute difference between line features, θ_{line}, with and without SLAM compared to the true values.

Chapter 6

Summary and Recommendations for Future Work

6.1 Summary

Simultaneous Localization and Mapping (SLAM) can be used to map an unknown environment, and at the same time, find the position of the mobile vehicle. This procedure consists of motion sensing and environment sensing; for the motion sensing, a vehicle model is used to predict the mobile vehicle positions and for the environment sensing, feature extraction and data association are used to find features of the environment. Finally, an EKF filter is used to couple the motion sensing to the environment sensing. This is looped over data to keep the motion sensing error bounded through the use of environment sensing. SLAM has been applied to different environments, for example, outdoor, indoor, underwater, aerial, and outer space environments. Different environments limit the type of sensors that one can use. The environment of interest for this dissertation is the environment inside an oil storage tank or a pipeline. The oil storage tank and pipeline environments are harsh environments, which limit the type of sensors that can be used, for example, laser sensors need to be avoided for safety precautions, global positioning systems (GPSs) need to be avoided since a GPS signal cannot penetrate oil storage tank walls or pipeline walls, and digital compasses are not preferable because of the ferrous material used to build oil tanks and pipelines, digital cameras cannot be used

since oil is opaque. These limitations necessitate the use of SONAR sensors inside the harsh environment of interest to this work.

The SONAR sensor has been used to extract the map of the environment. The main problems with SONAR sensors are the high angular uncertainty, which ranges between 22.5^o and 30^o, cross talk between SONAR sensors, and multiple reflections which occur whenever the mobile vehicle is close to an obstacle or whenever the SONAR signal hits an obstacle at a high angle. These problems make the mapping of the environment with a SONAR sensor challenging. Researchers have investigated different algorithms to deal with the SONAR sensor challenges, such as Triangulation Based Fusion (TBF), Hough Transform (HT), Triangulation Hough based Fusion (THF), and SONAR salient (SS). As an initial step, the author investigated the commonly used feature extraction techniques in the literature for a SONAR sensor through simulations and experiments. The simulation environment was an empty square while the experimental environment was an empty rectangular. The reason behind investigating these commonly used algorithms was to understand the strengths and the weaknesses of the algorithms and to compare them in terms of accuracy and computational speed.

It was noted that TBF and SS, which are point feature extraction algorithms, capture more parts of the environment as the mobile vehicle visits more locations and performs full SONAR scans. The computational speeds of TBF and SS become slower with each additional SONAR scan; for example, 6 full SONAR scans take approximately 39 seconds, whereas 30 full SONAR scans take approximately 721 seconds to compute. As can be seen, adding 1 more full SONAR scan can increase

the computational time of TBF and SS algorithms significantly. On the other hand, the HT and THF algorithms, which are line feature extraction algorithms, were found to perform adequately with a quick computational time. Also, a noticeable improvement in THF results over HT results was noticed. The reason behind the improvement is as follows. The THF algorithm uses processed data from the TBF algorithm rather than raw SONAR data as in the case of the HT algorithm. All the commonly used algorithms (TBF, THF, HT, and SS) are feasible feature extraction algorithms, but each has its own advantages and disadvantages like computational time and map density. Also, the features that are found from these algorithms are repeated. The TBF, HT, SS, and THF algorithms have been compared on the same SONAR data, which can be a good benchmarker for the algorithms results. It can be difficult to compare the benefits of algorithms when they are used on different data sets. To aid in this comparison, this dissertation performed a benchmark on several algorithms, which had not been directly compared previously. As explained earlier, SLAM consists of different parts and selecting a good feature extraction can improve SLAM performance.

A new feature extraction algorithm was developed for use with SONAR sensor data. This new feature extraction algorithm fuses other extraction algorithms, but it performs better than its composite parts. Post-processing of SONAR data is very important, because of the noisy nature of SONAR data (as compared to laser data). The proposed fusion feature extraction algorithm was tested through numerical studies and physical experiments. This fusion feature extraction algorithm takes advantage of the attractive features of its component algorithms (TBF, HT, THF,

and SS), while performing better than any of these individual algorithms separately. The proposed fusion algorithm doesn't have repeated features, and it is able to capture unique features. In addition, it takes advantage of TBF to work on corner features, as well as SS to work on interior features, and both of them run in parallel to save time. Through the study, it was found that the SS algorithm can be effective in locating interior features but considerable errors occur in the radius value of the determined features. The proposed feature extraction was able to extract features from the tested environment that contained interior features. In addition, it was noticed that a high enough number of locations (which are properly spaced from each other) was found to be important for enhancing feature extraction.

After selecting a good feature extraction algorithm for the SLAM algorithm, vehicle modeling and motion sensing were explored. For the motion sensing, the encoder and gyroscope were used together to improve the motion sensing results. The reason for considering a gyroscope is that a gyroscope is not affected by the lateral wheel slippage, which is common in a skid-steering mobile vehicle. In addition, one of the future applications is to use a mobile vehicle to inspect the base of an oil storage tank, where slippage is likely to happen. Different combinations of gyroscopes and encoders have been considered in this dissertation work, for example, encoder only, constant drift value from the gyroscope, and gyrodometry, which are commonly used in the literature for motion model estimation. A new vehicle pose estimation model is introduced in this dissertation work which will be known as the varying drift approach. The selection of a good motion model is important, because as explained earlier, improving any part of SLAM has the possibility of im-

proving the overall performance of SLAM. Experiments were done to test different motion models and the varying drift approach was found to give the best results. In the varying drift approach, one basically resets the gyroscope drift value every time the mobile vehicle comes to a complete stop. The drift in the gyroscope reading is caused by temperature changes as well as other unknown factors. Resetting the drift value whenever the mobile vehicle stops results in a better vehicle model. Also, for future needs, while inspecting the base of an oil storage tank, the mobile vehicle has to come to a complete stop. Utilizing this stop, the gyroscope drift value can be calculated quite often, which would improve the overall results as explained previously.

By implementing the feature extraction fusion algorithm and using the varying drift approach, both of which were proposed in this dissertation, EKF-SLAM can be implemented to further improve the results. This implementation of SLAM produces a better representation of the environment and also further localizes the mobile vehicle. To test the EKF-SLAM scheme, slippage was induced into the environment, for example, by adding dry powder or hydrophobic liquid to the environment. This novel implementation of EKF-SLAM in a slippage-induced environment, which used an encoder, a gyroscope, and a SONAR sensor, was included in this dissertation work. The mobile vehicle was run in this environment, where motion and environment sensing was used to gather data. The gathered data were processed with and without SLAM. For the cases with dry powder or hydrophobic coating on the floor, the results show that with SLAM, the performance was better for feature extraction than without it. Whenever the mobile vehicle entered the slippage area,

the no-SLAM case is unable to predict or correct the vehicle motion, which led to features being outside the environment, whereas while performing SLAM, the mobile vehicle path was corrected which led to features being more confined to the actual environment. Through the use of the dissertation's feature extraction, the angular uncertainty in the SONAR sensor was reduced, which led the EKF-SLAM scheme to rely more on the environment sensing than the motion sensing. The reason why the proposed feature algorithm had lower angular uncertainty can be explained as follows. The TBF algorithm had a high threshold value which caused most of the outliers to be removed or less emphasis was placed on them.

Overall, the EKF-SLAM scheme was successfully implemented with the SONAR data with a slippage environment and with interior features environment. Although the SONAR data had a high angular uncertainty, through the use of the novel feature extraction introduced here, the angular uncertainty in the features were reduced. In addition, the proposed feature extraction was able to produce unique and unrepeated features. Also, varying drift approach, which was introduced in this dissertation work, helped improve the vehicle model, which also improved the mobile vehicle path. Finally, dry powder and hydrophobic liquid were added to the environment to induce slippage and for these cases, it was found that a combination of the proposed feature extraction, modified vehicle model, and the EKF-SLAM scheme were able to correct the vehicle path and keep the SONAR data better correlated with the environment.

6.2 Recommendations for Future Work

As a future study, the EKF-SLAM scheme can be tested with different modifications that might improve the overall performance even more. The different modifications are listed below:

1. The commonly available SONAR sensor provides range readings in terms of flight time. The user can convert the flight time with the speed of the medium to calculate the distance. Some researchers (Kleeman, 1995) have developed their own SONAR sensors. These SONAR sensors can be used to not only measure the range but also the amplitude of the return of the SONAR signal. This allows another layer of filtering to be performed, in order to remove unwanted or weak SONAR amplitudes. As a suggestion for future work, an implementation of the work developed in this dissertation with this improved SONAR sensor could allow for even more improvements.

2. The proposed feature extraction scheme determined the interior and exterior features of a closed environment. The locations of the interior features were determined successfully but with significant error with respect to the radius of the interior features. Increasing the value of acceptable radius for the SS algorithm can help improve the result.

3. The k-mean clustering scheme can be replaced by Expectation Maximization (EM) clustering. K-mean clustering requires information about the number of features in the environment. Replacing k-mean clustering with EM clustering

can ensure a more autonomous mobile vehicle.

4. A micro-electromechanical system (MEMS) gyroscope and a magnetic encoder were tested in this dissertation. As a recommendation for future work, different types of gyroscopes and encoders should be tested and compared to other sensors, such as a fiber optic gyroscope (FOG), capacitive encoder, and so on.

5. The use of a tracked wheel with similar environmental setups could be tested and compared to a four wheeled vehicle.

6. The bottom of an oil storage tank is inclined at an angle less than 10^o. As a future task, the mobile vehicle will be tested on an inclined surface and ordered to perform EKF-SLAM.

7. The use of a different vehicle model that uses more sensor inputs could improve the SLAM implementation to the level needed for a real-world application in an oil storage tank.

Appendix A

Covariance of Control Input

A.1 True values for straight line tests

In this section, the true values for the straight line are presented in Table A.1. The true values are used with the estimate value to calculate the variance of the wheel encoder. All the true measurements were measured using a tape measurement, and were measured for the front left wheel. Also, $(190 - 16.5)$ cm should be added to the Y-Values in Table A.1 to transform the Y-Values to the global coordinate system.

A.2 True values for zero-turn radius tests

In this section, the true values for the zero-turn radius are presented in Table A.2. The true values are used with the estimate value to calculate the variance of the gyroscope sensor. All the true measurements were measured using a digital protractor. Also, $360^o -$True Angle should be used to get the actual true angle position from the zero-turn radius test.

Table A.1: True values from the straight line tests

Test No	**X-Value (cm)**	**Y-Value (cm)**	**Test No**	**X-Value (cm)**	**Y-Value (cm)**
1	7.7	14.7	29	5.6	18.6
2	6.9	15.9	30	1.9	18.6
3	8.5	16.7	31	1.9	18.8
4	7.3	17.2	32	2.2	18.8
5	7.4	17.5	33	2.0	19.0
6	6.8	18.0	34	1.9	18.9
7	6.7	18.0	35	1.6	18.9
8	4.5	14.0	36	3.8	19.8
9	5.5	15.8	37	4.9	19.9
10	5.3	16.2	38	5.0	20.0
11	3.9	16.5	39	4.3	20.0
12	5.1	17.0	40	3.8	20.2
13	3.1	17.2	41	4.6	20.2
14	7.0	12.5	42	3.5	20.0
15	4.8	17.5	43	4.7	20.1
16	4.0	17.4	44	3.3	20.2
17	1.7	18.1	45	4.1	20.1
18	7.7	16.8	46	4.4	20.2
19	7.5	16.9	47	3.2	20.1
20	8.4	17.5	48	1.9	19.1
21	7.9	17.2	49	1.4	19.0
22	6.8	18.0	50	1.7	19.0
23	5.7	18.2	51	4.8	20.5
24	7.7	18.1	52	1.8	19.0
25	1.5	18.3	53	1.9	19.1
26	5.0	18.5	54	2.3	19.1
27	5.0	18.8	55	1.8	19.1
28	5.0	18.2			

Table A.2: True values from the zero-turn radius tests

Test No	**True Angle (Degree)**	**Test No**	**True Angle (Degree)**
1	36.40	29	25.45
2	32.75	30	24.40
3	33.60	31	25.20
4	29.25	32	24.40
5	27.75	33	22.20
6	38.85	34	23.30
7	33.70	35	23.20
8	37.10	36	23.00
9	34.50	37	28.10
10	30.60	38	26.45
11	32.00	39	26.35
12	28.60	40	23.50
13	28.35	41	24.35
14	28.10	42	26.00
15	26.85	43	24.70
16	25.85	44	21.00
17	25.80	45	23.10
18	25.20	46	28.50
19	24.20	47	26.55
20	25.90	48	26.35
21	23.85	49	27.70
22	33.20	50	23.90
23	30.75	51	24.50
24	31.70	52	23.30
25	30.05	53	23.50
26	27.35	54	26.10
27	27.70	55	20.40
28	26.95		

Appendix B

Default Case: Raw and Proposed Plots

In Appendix B, the raw SONAR data and the processed feature extraction results of the default case environment, with $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$, are provided.

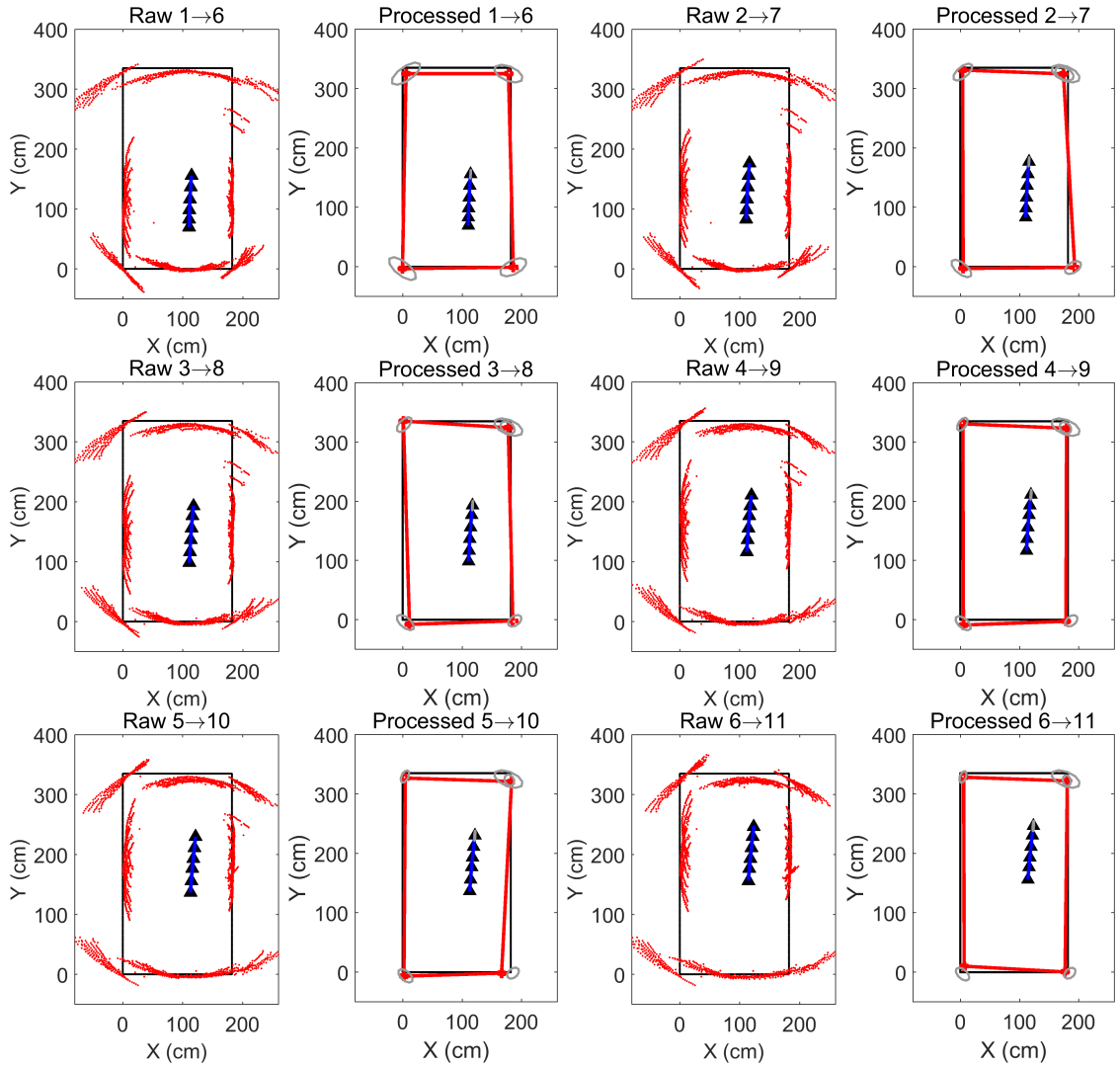


Figure B.1: Raw SONAR data and processed feature extraction results for the default case environment for $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$ for mobile vehicle stops between 1 and 11. For instance, raw 1 through 6 show all the SONAR scans that happened between stops 1 through 6, totalling 6 full SONAR scans. The raw SONAR scan are sent to the feature extraction algorithm and the results are shown in processed 1 through 6.

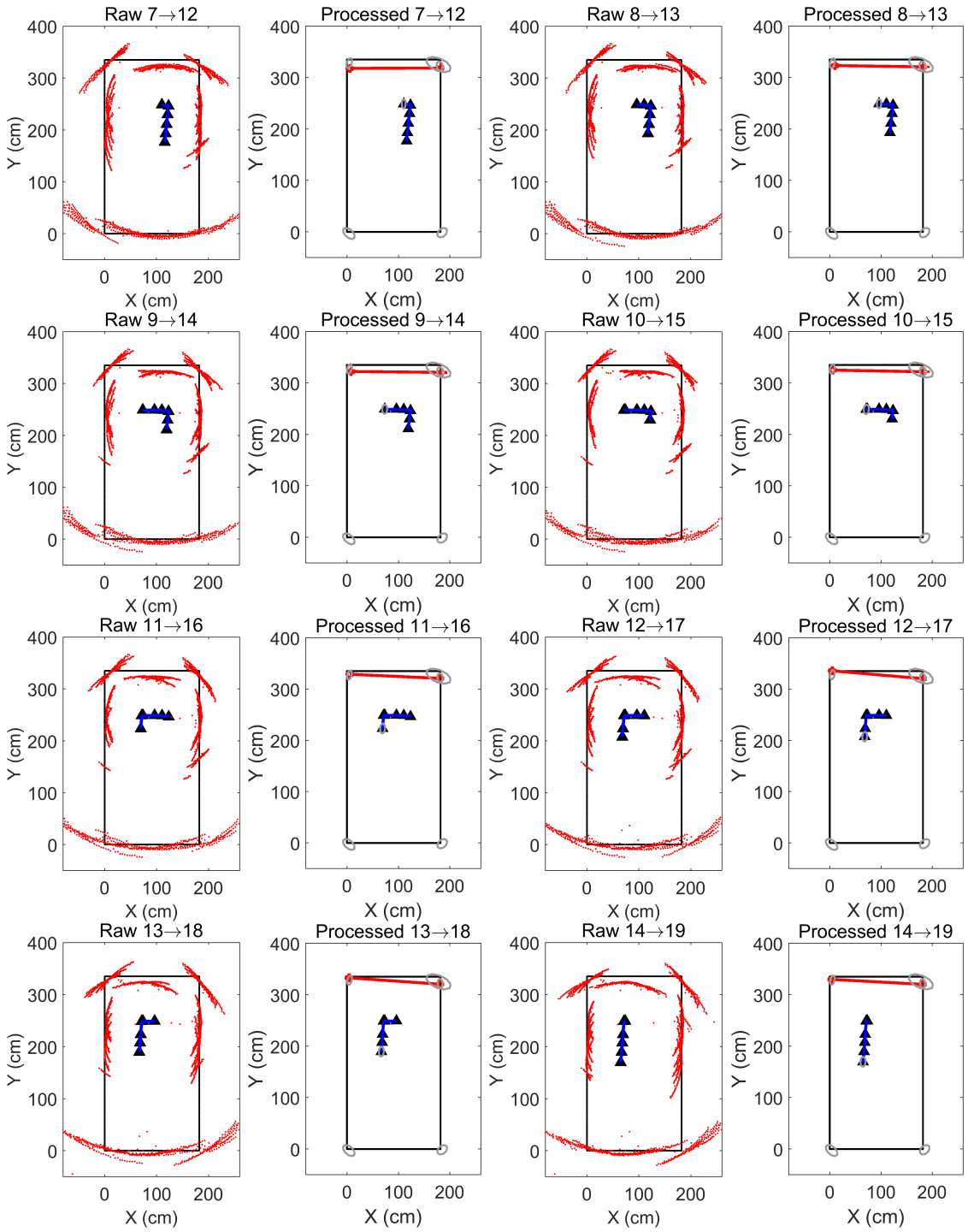


Figure B.2: Raw SONAR data and processed feature extraction results for the default case environment for $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$ for mobile vehicle stops between 7 and 19.

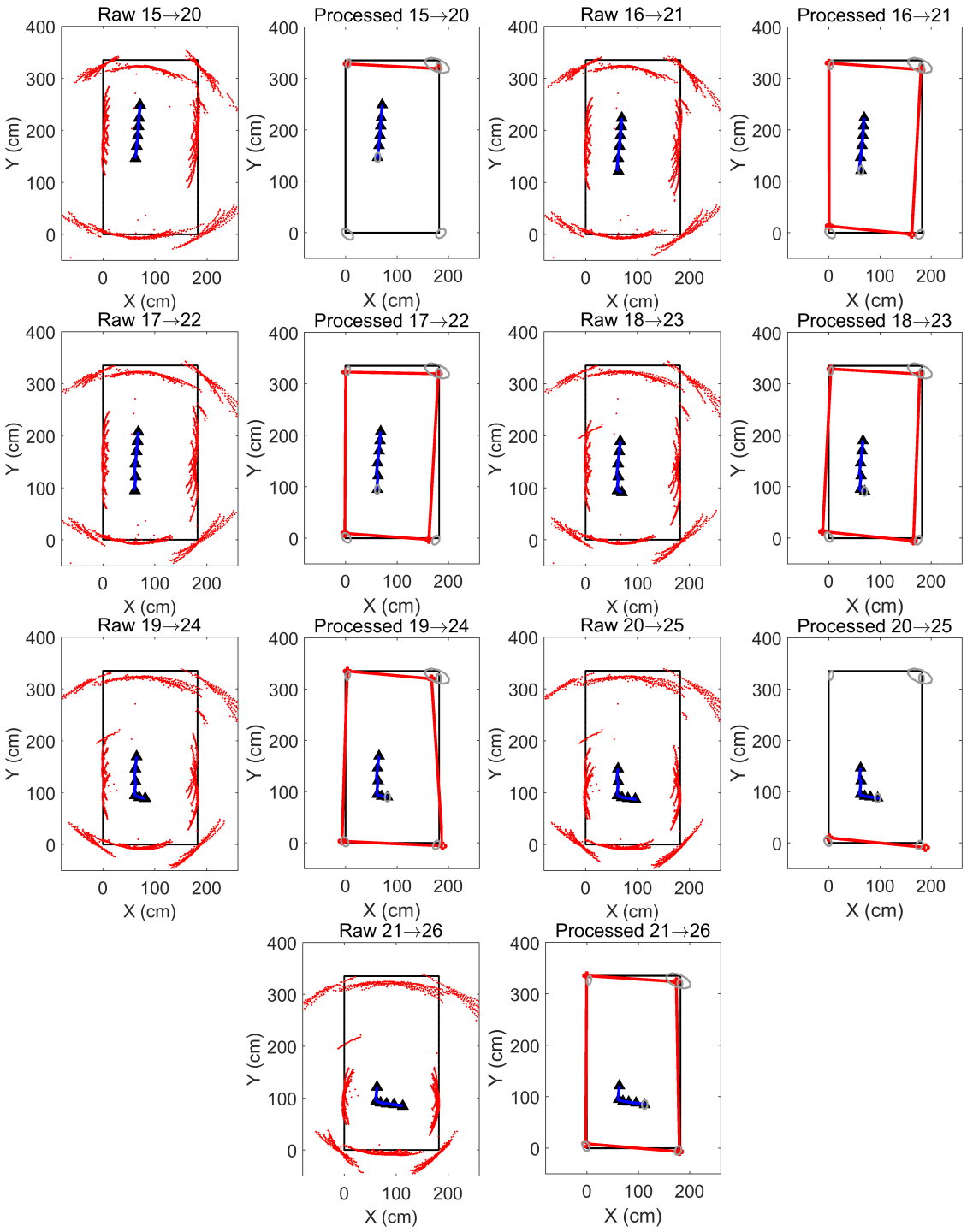


Figure B.3: Raw SONAR data and processed feature extraction results for the default case environment for $\sigma_{s\rho} = 5$ cm and $\sigma_{s\theta} = 4^o$ for mobile vehicle stops between 15 and 26.

Appendix C

Table: Results

In Appendix C, the estimated positions of the corners and walls from the sensor data, the actual positions of the corners and walls, and the absolute difference between them is listed for the default, the dry powder, and the hydrophobic coating cases.

C.1 Default Case

Table C.1: Default case: corner data for environment with no alteration to floor plan and absolute errors between estimates and actual values.

Corner #	**Actual Data** X**(cm),** Y**(cm)**	**Exp. Results** X**(cm),** Y**(cm)**	**|Actual-Exp.|** X**(cm),** Y**(cm)**
NO SLAM Case:			
1	$0.00, 0.00$	$-4.88, -0.55$	$4.88, 0.55$
2	$0.00, 335.10$	$6.94, 323.83$	$6.94, 11.27$
3	$182.00, 0.00$	$164.32, -2.05$	$17.68, 2.05$
4	$182.00, 335.10$	$183.36, 322.30$	$1.36, 12.80$
SLAM Case $\sigma_{s\rho} =$ **5 cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$-2.45, -8.55$	$2.45, 8.55$
2	$0.00, 335.10$	$6.55, 323.80$	$6.55, 11.30$
3	$182.00, 0.00$	$168.27, -4.11$	$13.73, 4.11$
4	$182.00, 335.10$	$181.16, 319.57$	$0.84, 15.53$
SLAM Case $\sigma_{s\rho} =$ **10 cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$-2.41, -0.75$	$2.41, 0.75$
2	$0.00, 335.10$	$7.53, 328.40$	$7.53, 6.70$
3	$182.00, 0.00$	$167.37, -2.84$	$14.63, 2.84$
4	$182.00, 335.10$	$182.07, 321.50$	$0.07, 13.60$
SLAM Case $\sigma_{s\rho} =$ **15 cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$-4.33, -1.62$	$4.33, 1.62$
2	$0.00, 335.10$	$8.61, 329.56$	$8.61, 5.54$
3	$182.00, 0.00$	$165.70, -2.65$	$16.30, 2.65$
4	$182.00, 335.10$	$183.37, 323.40$	$1.37, 11.70$

Table C.2: Default case: line data for environment with no alteration to floor plan and absolute errors between estimates and actual values.

Line #	**Actual Data** ρ**(cm),** θ**(degree)**	**Exp. Results** ρ**(cm),** θ**(degree)**	**|Actual-Exp.|** ρ**(cm),** θ**(degree)**
NO SLAM Case:			
1	$0.00, 90.00$	$-0.59, 89.49$	$0.59, 0.51$
2	$0.00, 360.00$	$-4.86, 357.91$	$4.86, 2.09$
3	$182.00, 360.00$	$164.16, 356.64$	$17.84, 3.36$
4	$335.10, 90.00$	$323.88, 89.50$	$11.22, 0.49$
SLAM Case $\sigma_{s\rho} = 5$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$-8.48, 91.49$	$8.48, 1.49$
2	$0.00, 360.00$	$-2.22, 358.45$	$2.22, 1.55$
3	$182.00, 360.00$	$168.30, 357.72$	$13.70, 2.28$
4	$335.10, 90.00$	$323.86, 88.61$	$11.24, 1.39$
SLAM Case $\sigma_{s\rho} = 15$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$-0.78, 89.29$	$0.78, 0.71$
2	$0.00, 360.00$	$-2.39, 358.27$	$2.39, 1.73$
3	$182.00, 360.00$	$167.32, 357.40$	$14.68, 2.60$
4	$335.10, 90.00$	$328.44, 87.74$	$6.66, 2.26$
SLAM Case $\sigma_{s\rho} = 15$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$-1.65, 89.65$	$1.65, 0.35$
2	$0.00, 360.00$	$-4.26, 357.76$	$4.26, 2.24$
3	$182.00, 360.00$	$165.60, 356.90$	$16.40, 3.10$
4	$335.10, 90.00$	$329.66, 87.98$	$5.44, 2.02$

C.2 Dry Powder Case

Table C.3: Dry powder case: corner coordinates for the environment with dry powder and absolute errors between estimates and actual values.

Corner #	**Actual Data** X**(cm),** Y**(cm)**	**Exp. Results** X**(cm),** Y**(cm)**	**|Actual-Exp.|** X**(cm),** Y**(cm)**
NO SLAM Case:			
1	$0.00, 0.00$	$-7.10, 14.40$	$7.10, 14.40$
2	$0.00, 335.10$	$4.81, 339.36$	$4.81, 4.26$
3	$182.00, 0.00$	$151.22, 5.22$	$30.78, 5.22$
4	$182.00, 335.10$	$181.78, 324.76$	$0.22, 10.34$
SLAM Case $\sigma_{s\rho} = 5$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$-4.75, 2.40$	$4.75, 2.40$
2	$0.00, 335.10$	$4.31, 331.89$	$4.31, 3.21$
3	$182.00, 0.00$	$160.52, -0.26$	$21.48, 0.26$
4	$182.00, 335.10$	$180.03, 323.56$	$1.97, 11.54$
SLAM Case $\sigma_{s\rho} = 10$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$0.86, 10.45$	$0.86, 10.45$
2	$0.00, 335.10$	$5.17, 335.89$	$5.17, 0.79$
3	$182.00, 0.00$	$157.64, 2.17$	$24.36, 2.17$
4	$182.00, 335.10$	$181.45, 323.99$	$0.55, 11.11$
SLAM Case $\sigma_{s\rho} = 15$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$-2.44, 8.32$	$2.44, 8.32$
2	$0.00, 335.10$	$5.32, 332.26$	$5.32, 2.84$
3	$182.00, 0.00$	$166.46, 0.57$	$15.54, 0.57$
4	$182.00, 335.10$	$179.70, 323.10$	$2.30, 12.00$

Table C.4: Dry powder case: line data for environment with dry powder and absolute errors between estimates and actual values.

Line #	**Actual Data** ρ**(cm),** θ**(degree)**	**Exp. Results** ρ**(cm),** θ**(degree)**	**|Actual-Exp.|** ρ**(cm),** θ**(degree)**
NO SLAM Case:			
1	$0.00, 90.00$	$13.96, 86.68$	$13.96, 3.32$
2	$0.00, 360.00$	$-7.62, 357.90$	$7.62, 2.10$
3	$182.00, 360.00$	$150.03, 354.54$	$31.97, 5.46$
4	$335.10, 90.00$	$338.61, 85.28$	$3.51, 4.72$
SLAM Case $\sigma_{s\rho} = 5$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$2.32, 89.08$	$2.32, 0.92$
2	$0.00, 360.00$	$-4.81, 358.42$	$4.81, 1.58$
3	$182.00, 360.00$	$160.24, 356.55$	$21.76, 3.45$
4	$335.10, 90.00$	$331.72, 87.29$	$3.38, 2.71$
SLAM Case $\sigma_{s\rho} = 10$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$10.48, 86.98$	$10.48, 3.02$
2	$0.00, 360.00$	$0.72, 359.24$	$0.72, 0.76$
3	$182.00, 360.00$	$157.05, 355.77$	$24.95, 4.23$
4	$335.10, 90.00$	$335.47, 86.14$	$0.37, 3.86$
SLAM Case $\sigma_{s\rho} = 15$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$8.20, 87.37$	$8.20, 2.63$
2	$0.00, 360.00$	$-2.64, 358.63$	$2.64, 1.37$
3	$182.00, 360.00$	$166.29, 357.65$	$15.71, 2.35$
4	$335.10, 90.00$	$332.08, 86.99$	$3.02, 3.01$

C.3 Hydrophobic Coating Case

Table C.5: Hydrophobic coating case: corner coordinates for the environment with NeverWet liquid and absolute errors between estimates and actual values.

Corner #	**Actual Data** X**(cm),** Y**(cm)**	**Exp. Results** X**(cm),** Y**(cm)**	**|Actual-Exp.|** X**(cm),** Y**(cm)**
NO SLAM Case:			
1	$0.00, 0.00$	$-7.17, 10.91$	$7.17, 10.91$
2	$0.00, 335.10$	$1.17, 335.80$	$1.17, 0.70$
3	$182.00, 0.00$	$159.25, 5.85$	$22.75, 5.85$
4	$182.00, 335.10$	$177.22, 325.67$	$4.78, 9.43$
SLAM Case $\sigma_{s\rho} = 5$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$-3.92, 5.88$	$3.92, 5.88$
2	$0.00, 335.10$	$4.10, 327.62$	$4.10, 7.48$
3	$182.00, 0.00$	$163.94, 4.55$	$18.06, 4.55$
4	$182.00, 335.10$	$177.59, 325.04$	$4.41, 10.06$
SLAM Case $\sigma_{s\rho} = 10$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$-8.03, 5.58$	$8.03, 5.58$
2	$0.00, 335.10$	$3.86, 323.55$	$3.86, 11.55$
3	$182.00, 0.00$	$163.65, 3.62$	$18.35, 3.62$
4	$182.00, 335.10$	$178.47, 323.07$	$3.53, 12.03$
SLAM Case $\sigma_{s\rho} = 15$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 0.00$	$-5.26, 2.86$	$5.26, 2.86$
2	$0.00, 335.10$	$2.60, 322.95$	$2.60, 12.15$
3	$182.00, 0.00$	$164.15, 3.35$	$17.85, 3.35$
4	$182.00, 335.10$	$176.88, 323.41$	$5.12, 11.69$

Table C.6: Hydrophobic coating case: line data for environment with NeverWet liquid and absolute errors between estimates and actual values.

Line #	**Actual Data** ρ**(cm),** θ**(degree)**	**Exp. Results** ρ**(cm),** θ**(degree)**	**|Actual-Exp.|** ρ**(cm),** θ**(degree)**
NO SLAM Case:			
1	$0.00, 90.00$	$10.69, 88.26$	$10.69, 1.74$
2	$0.00, 360.00$	$-7.45, 358.53$	$7.45, 1.47$
3	$182.00, 360.00$	$158.67, 356.78$	$23.33, 3.22$
4	$335.10, 90.00$	$335.31, 86.71$	$0.21, 3.29$
SLAM Case $\sigma_{s\rho} = 5$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$5.85, 89.55$	$5.85, 0.45$
2	$0.00, 360.00$	$-4.06, 358.57$	$4.06, 1.43$
3	$182.00, 360.00$	$163.60, 357.56$	$18.40, 2.44$
4	$335.10, 90.00$	$327.65, 89.15$	$7.45, 0.85$
SLAM Case $\sigma_{s\rho} = 10$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$5.48, 89.35$	$5.48, 0.65$
2	$0.00, 360.00$	$-8.23, 357.86$	$8.23, 2.14$
3	$182.00, 360.00$	$163.31, 357.34$	$18.69, 2.66$
4	$335.10, 90.00$	$323.56, 89.84$	$11.54, 0.16$
SLAM Case $\sigma_{s\rho} = 15$ **cm and** $\sigma_{s\theta} = 4^o$**:**			
1	$0.00, 90.00$	$2.88, 90.17$	$2.88, 0.17$
2	$0.00, 360.00$	$-5.33, 358.59$	$5.33, 1.41$
3	$182.00, 360.00$	$163.89, 357.72$	$18.11, 2.28$
4	$335.10, 90.00$	$322.94, 90.15$	$12.16, 0.15$

Bibliography

Anousaki, G.C. and Kyriakopoulos, K.J. (2007). Simultaneous localization and map building of skid-steered robots. *Robotics & Automation Magazine, IEEE* **14**(1), 79–89.

Aulinas, J., Petillot, Y.R., Salvi, J., and Lladó, X. (2008). The slam problem: a survey. In *CCIA*, 363–371. Citeseer.

Baolong, L., Bo, H., Yongqing, W., Xuan, Z., and Lei, G. (2007). Triangulation & hough transform based fusion of sonar data for mobile robotics. In *2nd IEEE Conference on Industrial Electronics and Applications ICIEA*, 1165–1170. IEEE.

Barshan, B. and Durrant-Whyte, H.F. (1995). Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation* **11**(3), 328–342. doi:10.1109/70.388775.

Barshan, B. and Kuc, R. (1990). Differentiating sonar reflections from corners and planes by employing an intelligent sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**(6), 560–569.

Borenstein, J. and Feng, L. (1996a). Gyrodometry: A new method for combining data from gyros and odometry in mobile robots. In *IEEE International Conference on Robotics and Automation*, volume 1, 423–428. IEEE.

Borenstein, J. and Feng, L. (1996b). Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation* **12**(6), 869–880.

Borenstein, J. and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation* **7**(3), 278–288.

Caracciolo, L., De Luca, A., and Iannitti, S. (1999). Trajectory tracking control of a four-wheel differentially driven mobile robot. In *IEEE International Conference on Robotics and Automation*, volume 4, 2632–2638. IEEE.

Carlone, L., Ng, M.K., Du, J., Bona, B., and Indri, M. (2010). Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication. In *IEEE International Conference on Robotics and Automation (ICRA)*, 243–249. IEEE.

Chenavier, F. and Crowley, J.L. (1992). Position estimation for a mobile robot using vision and odometry. In *IEEE International Conference on Robotics and Automation,*, 2588–2593. IEEE. doi:10.1109/ROBOT.1992.220052.

Choi, J., Ahn, S., and Chung, W.K. (2005). Robust sonar feature detection for the slam of mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, 3415–3420. IEEE.

Chung, H., Ojeda, L., and Borenstein, J. (2001). Accurate mobile robot dead-reckoning with a precision-calibrated fiber-optic gyroscope. *IEEE Transactions on Robotics and Automation* **17**(1), 80–84.

Cox, I.J. (1991). Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation* **7**(2), 193–204.

De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D.L. (2000). The mahalanobis distance. *Chemometrics and intelligent laboratory systems* **50**(1), 1–18.

Dinçay, B. (2010). Gps/optical encoder based navigation methods for dspic micro-controled mobile vehicle .

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE* **13**(2), 99–110. doi:10.1109/MRA.2006.1638022.

Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation* **3**(3), 249–265.

Elfes, A. (1992). Dynamic control of robot perception using multi-property inference grids. In *IEEE International Conference on Robotics and Automation*, 2561–2567. IEEE.

Eustice, R.M., Singh, H., and Leonard, J.J. (2006). Exactly sparse delayed-state filters for view-based slam. *IEEE Transactions on Robotics* **22**(6), 1100–1114.

Fazli, S. and Kleeman, L. (2007). Simultaneous landmark classification, localization and map building for an advanced sonar ring. *Robotica* **25**(03), 283–296.

Goel, P., Roumeliotis, S.I., and Sukhatme, G. (1999). Robust localization using relative and absolute position estimates. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, volume 2, 1134–1140. IEEE.

Hartigan, J.A. and Wong, M.A. (1979). Algorithm as 136: A k-means clustering algorithm. *Applied statistics* 100–108.

Hormann, K. and Agathos, A. (2001). The point in polygon problem for arbitrary polygons. *Computational Geometry* **20**(3), 131–144. doi:10.1016/S0925-7721(01)00012-8.

Hough, P.V. (1962). Method and means for recognizing complex patterns. *US Patent, 3069654* .

Ismail, H. and Balachandran, B. (2013). A comparison of feature extraction algorithms based on sonar sensor data. In *ASME 2013 International Mechanical Engineering Congress and Exposition*, V04AT04A024–V04AT04A024. American Society of Mechanical Engineers. doi:10.1115/IMECE2013-62989.

Ismail, H. and Balachandran, B. (2014). Feature extraction algorithm fusion for sonar sensor data based environment mapping. In *ASME 2014 International Mechanical Engineering Congress and Exposition*, V04AT04A025–V04AT04A025. American Society of Mechanical Engineers. doi:10.1115/IMECE2014-37116.

Ismail, H. and Balachandran, B. (2015a). Algorithm fusion for feature extraction and map construction from sonar data. *IEEE Sensors Journal* doi:10.1109/JSEN.2015.2456900.

Ismail, H. and Balachandran, B. (2015b). Vehicle pose estimation and sonar sensor based mapping. In *ASME 2015 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers.

Jaai, R., Chopra, N., Balachandran, B., and Karki, H. (2012). Simultaneous localization and mapping with consideration of robot system dynamics. In *SPIE Smart Structures and Materials+ Nondestructive Evaluation and Health Monitoring*, 83451F–83451F. International Society for Optics and Photonics.

Ji, X., Wang, S., Xu, Y., Shi, Q., and Xia, D. (2006). Application of the digital signal procession in the mems gyroscope de-drift. In *1st IEEE International Conference on Nano/Micro Engineered and Molecular Systems*, 218–221. IEEE. doi:10.1109/NEMS.2006.334690.

Joly, C. and Rives, P. (2008). Bearing-only slam: comparison between probabilistic and deterministic methods .

Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering* **82**(1), 35–45.

Kim, M.C., Chung, W.K., Youm, Y., and Oht, S.R. (1999). Position estimation of a car-like mobile robot using a gyroscope and disturbance conditions. *Power* **12**, 15mA.

Kleeman, L. (1995). Odometry error covariance estimation for two wheel robot vehicles. *Intelligent Robotics Research Centre, Department of Electrical and Computer Systems Engineering, Monash University, Tech. Rep. MECSE-95-1* .

Kleeman, L. and Kuc, R. (1994). An optimal sonar array for target localization and classification. In *IEEE International Conference on Robotics and Automation*, 3130–3135. IEEE.

Kozłowski, K. and Pazderski, D. (2004). Modeling and control of a 4-wheel skid-steering mobile robot. *Int. J. Appl. Math. Comput. Sci* **14**(4), 477–496.

Kuc, R. and Barshan, B. (1989). Navigating vehicles through an unstructured environment with sonar. In *IEEE International Conference on Robotics and Automation*, 1422–1426. IEEE.

Lee, H.S. and Lee, K.M. (2009). Multi-robot slam using ceiling vision. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, 912–917. IEEE.

Lee, S.J. and Song, J.B. (2010). A new sonar salient feature structure for ekf-based slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5966–5971.

Leonard, J.J. and Durrant-Whyte, H.F. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation* **7**(3), 376–382. doi:10.1109/70.88147.

Lucet, E., Grand, C., Sallé, D., and Bidaud, P. (2009). Dynamic control of the 6wd skid-steering mobile robot roburoc6 using sliding mode technique. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 11–15.

Mahalanobis, P.C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)* **2**, 49–55.

Mandow, A., Martinez, J.L., Morales, J., Blanco, J.L., Garcia-Cerezo, A., and Gonzalez, J. (2007). Experimental kinematics for wheeled skid-steer mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1222–1227. IEEE.

Paz, L.M. and Neira, J. (2006). Optimal local map size for ekf-based slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5019–5025. IEEE.

Reina, G., Ishigami, G., Nagatani, K., and Yoshida, K. (2008). Vision-based estimation of slip angle for mobile robots and planetary rovers. In *IEEE International Conference on Robotics and Automation (ICRA)*, 486–491. IEEE.

Reina, G., Ishigami, G., Nagatani, K., and Yoshida, K. (2010). Odometry correction using visual slip angle estimation for planetary exploration rovers. *Advanced Robotics* **24**(3), 359–385.

Reina, G., Ojeda, L., Milella, A., and Borenstein, J. (2006). Wheel slippage and sinkage detection for planetary rovers. *IEEE/ASME Transactions on Mechatronics* **11**(2), 185–195.

Shuang, G., Cheung, N.C., Cheng, E.K., Lei, D., and Xiaozhong, L. (2007). Skid steering in 4-wheel-drive electric vehicle. In *7th International Conference on Power Electronics and Drive Systems*, 1548–1553. IEEE.

Steckel, J. and Peremans, H. (2013). Batslam: simultaneous localization and mapping using biomimetic sonar. *PloS one* **8**(1), e54076.

Tardós, J.D., Neira, J., Newman, P.M., and Leonard, J.J. (2002). Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research* **21**(4), 311–330.

Terejanu, G.A. (2008). Extended kalman filter tutorial. *Online]. Disponible: http://users. ices. utexas. edu/˜ terejanu/files/tutorialEKF. pdf* .

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Thrun, S. and Liu, Y. (2005). Multi-robot slam with sparse extended information filers. In *The Eleventh International Symposium on Robotics Research*, 254–266. Springer.

Tsumura, T., Fujiwara, N., Shirakawa, T., and Hashimoto, M. (1981). An experimental system for automatic guidance of roboted vehicle following the route stored in memory. In *Proc. of the 11th Int. Symp. on Industrial Robots*, 18–193.

Walter, M.R., Eustice, R.M., and Leonard, J.J. (2007). Exactly sparse extended information filters for feature-based slam. *The International Journal of Robotics Research* **26**(4), 335–359.

Wang, C.M. (1988). Location estimation and uncertainty analysis for mobile robots. In *IEEE International Conference on Robotics and Automation*, 1231–1235. IEEE. doi:10.1109/ROBOT.1988.12229.

Weiqin, Z. (2009). Sonar features extraction algorithm for a mobile robot. In *Third International Symposium on Intelligent Information Technology Application (IITA)*, volume 3, 689–692. IEEE.

Welch, G. and Bishop, G. (2006). An introduction to the kalman filter. *University of North Carolina: Chapel Hill, North Carolina, US* .

Wijk, O. and Christensen, H.I. (2000a). Localization and navigation of a mobile robot using natural point landmarks extracted from sonar data. *Robotics and Autonomous Systems* **31**(1), 31–42.

Wijk, O. and Christensen, H.I. (2000b). Triangulation-based fusion of sonar data with application in robot pose tracking. *IEEE Transactions on Robotics and Automation* **16**(6), 740–752.

Wijk, O., Jensfelt, P., and Christensen, H.I. (1998). Triangulation based fusion of ultrasonic sensor data. In *IEEE International Conference on Robotics and Automation*, volume 4, 3419–3424. IEEE.

Yap, T.N. and Shelton, C.R. (2009). Slam in large indoor environments with low-cost, noisy, and sparse sonars. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1395–1401. IEEE.

Yi, J., Zhang, J., Song, D., and Jayasuriya, S. (2007). Imu-based localization and slip estimation for skid-steered mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2845–2850. IEEE.

Yu, W., Chuy, O., Collins Jr., E.G., and Hollis, P. (2009). Dynamic modeling of a skid-steered wheeled vehicle with experimental verification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4212–4219. IEEE.

Zunaidi, I., Kato, N., Nomura, Y., and Matsui, H. (2006). Positioning system for 4-wheel mobile robot: encoder, gyro and accelerometer data fusion with error model method. *CMU. Journal* **5**(1), 1.