DISTRIBUTED MULTI-ROBOT COOPERATIVE LOCALIZATION USING BAYESIAN FUSION ON THE SPECIAL EUCLIDEAN GROUP

by

Xiao Li

A thesis submitted to Johns Hopkins University in conformity with the requirements for the degree of Master of Science in Engineering in Robotics

Baltimore, Maryland

May 2014

Abstract

This thesis presents a new distributed cooperative localization technique using a second order sensor fusion method developed for the Special Euclidean group. Uncertainties in the robot pose, sensor measurements and landmark positions (neighboring robots in this case) are modeled as Gaussian distributions in exponential coordinates. This proves to be a better fit for posterior distributions resulting from the motion of nonholonomic kinematic systems with stochastic noise (compared to standard Gaussians in Cartesian coordinates). We provide a recursive closed-form solution to the multi-sensor fusion problem that can be used to incorporate a large number of sensor measurements into the localization routine and can be implemented in real time. The technique can be used for nonlinear sensor models without the need for further simplifications given that the required relative pose and orientation information can be provided, and it is scalable in that the computational complexity does not increase with the size of the robot team and increases linearly with the number of measurements taken from nearby robots. The proposed approach is validated with simulation first conceptually in Matlab then more realistically in the robotics simulator ROS/Gazebo. It is also compared with one of the current state of the art methods (distributed EKF) and shows promising results.

Acknowledgements

I would like to thank my advisor Professor Gregory S. Chirikjian for his tremendous help and support. The work presented in this thesis wouldn't have seen such progress if it isn't for his theoretical guidance and careful editing. I also want to thank my parents for standing by me from the start and supporting my interest and passion in the field of robotics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Multi-Robot Systems and Swarm Robotics

The field of Robotics and Automation is witnessing its drastic ascendency in terms of its significance in industrial and military applications as well as its growing importance in people's everyday lives. Multi-robot systems (sometimes also known as multi-agent systems) is a branch of robotics that deals with the collaboration among teams of homogenous/heterogenous robots in accomplishing certain tasks.

The authors of [1] presented the development of a group of robot modules that can perform tasks collectively and has the ability to rigidly connect to each other for navigation in complex environments. (for example in traversing a gap that's wider in length than any individual module). [2] presented testing of dispersion algorithms that can guide a swarm to efficiently cover a given space. And [3] showed in algorithms and hardware the mapping of an environment using multiple robots to reduce the time required to finish the task compared to a single robot.

Biological swarms (social animals) and their collective behaviors serves as the initial inspiration of research in swarm robotics. The main advantages of using multiple modularized robots to complete a certain task is threefold and are listed as follows:

- Scalability - A well designed swarm algorithm should be scalable in that the computational complexity should not increase or increases only a little with the number of robots in the swarm so that it shouldn't make a huge difference whether it's 100 or 1000 robots performing a task. This advantage/requirement is sometimes also phrased as the algorithm being fully distributed.

- Robustness - A swarm of modularized robot can effectively cope with subsystem failures (loss of a reasonable number of team members). This advantage is promoted by a high level of redundancy and the absence of a leader.
- Flexibility - The same team of robots can be reconfigured to perform different tasks and adapt to different environments [4].

The field of swarm robotics can be further subdivided into five categories, namely centralized vs decentralized communication, formation and control, collaborative mapping and localization, collaborative manipulation, collaborative path planning and object avoidance [5]. Each category is in its own an active fields of study. As with the case for single mobile robots, localization is one of the foremost problems to be solved for multi-robot systems. Since information can be shared among team members, a robot team has the potential to perform the task with higher efficiency and fewer resources [6]. Therefore this thesis focuses on the development of an efficient real-time distributed collaborative localization technique. The following sections will discuss more about multi-robot localization and put forth the main contributions of the technique developed, and lastly provide an outline of the entire thesis.

1.2 Multi-Robot Localization

The path to true autonomy starts with robots knowing where they are in a given workspace. Such a problem is known as robot localization. According to [7], the localization problem can be categorized into two subproblems: position tracking (local localization) which aims to compensate for small dead reckoning errors using sensor feedback, this approach is local in that the initial pose is assumed known. And global localization in which the robot ”figures out” its position given no knowledge of its initial pose. A tremendous amount of effort has been devoted to effectively and efficiently solve the localization problem and the field has seen major advancements in the establishment of highly practical and easy to implement algorithms with the EKF (Extended Kalman Filter) and PF (Particle Filter) based approaches as the most widely accepted solutions to the problem. However, the majority of existing approaches are tailored to localized a single robot. The field of multi-robot localization remains relatively fresh and to be explored[6].

Performing the localization task with multiple robots possess the advantage of information sharing. Robots within a team can exchange information with other members so to increase the accuracy and reduce uncertainty in their own estimate. This advantage is shown both in simulation and in experiment in [6] by letting two robots explore an indoor environment both executing their own single robot localization scheme, but the proposed collaborative fusion algorithm is used when two come into each other's detection range and results show that such an algorithmic reinforcement has the effect of significantly reducing the ambiguities existing in the original estimates. A collaborative architecture as such can effectively reduce the hardware cost of the entire team in that as long as at least one robot has a good knowledge of its location other team members can use this information along with relative measurements to infer their own position and reduce estimation errors.

1.3 Research Objectives and Contributions

Given many of the existing approaches to the multi-robot localization problem consider only uncertainties in the current robot's pose estimate and sensor measurement, the goal of this thesis is to explore cooperative localization in a more generalized setting where uncertainty in the sources of relative measurements (neighboring robots' pose estimates) are also considered. The distributed localization approach proposed in this thesis makes an effort to providing recursive closed-form expressions for real time cooperative sensor fusion used for pose updates of robots within a team. This work extends the method presented in [8] in that [8] considers cooperative localization with only one exact noise-free measurement (relative to a neighboring robot) whereas the technique proposed can taken into account any number of relative measurements while also considering sensor noise. This method is developed under the framework of exponential coordinates for Lie groups which gives this exotic sounding methodology a down-to-earth benefit: Gaussian distribution in Cartesian coordinates possesses elliptical probability density contours and the banana-shaped distribution resulting from incremental motions of a stochastic differential-kinematic system (i.e., a probabilistic model of mobile robots with nonholonomic kinematic constraints) is better represented by a Gaussian in exponential coordinates which produce a more conformable density contour (refer to figure 5.2). This underlying framework allows the proposed algorithm to tolerate higher errors without worrying about collapse of the normality assumption as uncertainty grows. Unlike most existing cooperative localization schemes that consider only uncertainty in the pose of the robot to be estimated and measurement noise, the presented method has also taken into account the uncertainty in the pose of nearby robots from which relative measurements are taken, making it a more realistic and dynamical localization technique. This approach is second order in its expansion of the Gaussians that describes the pose and measurement distributions using the *Baker-Campbell-Hausdorff (BCH) formula* [9], no simplifications are made regarding the system kinematics, thus preserving the full nonlinear characteristics of the original system. Lastly, the form of sensor measurement in this method is kept generic without assuming the type of sensor or any underlying characteristics given the Gaussian-in-exponential-coordinate model can be applied.

1.4 Outline

The remainder of this thesis is outlined as follows. **Chapter 2** introduces in more detail two most popular classes of localization techniques, their variations and applications in the multi-robot context and the pros and cons of each. **Chapter 3** introduces the mathematical foundation on which the proposed approach is based, namely the basics of Matrix Lie Groups and Exponential Mapping. **Chapter 4** provides a detailed derivation of the proposed technique. **Chapter 5** describes the experimental setup in simulation. The Robot Operating System (ROS) is chosen to be the experimental platform for its extensive community support and a wide range of established tools and libraries. The simulation platform Gazebo runs as a plugin of ROS and can be conveniently substituted by appropriately configured hardware, which is yet another reason for the choice of this control platform. **Chapter 6** presents the experimental results along with the discussion of their significance and **Chapter 7** concludes this thesis.

Chapter 2

Related Research

The problem of cooperative localization has been tackled with a wide variety of approaches over the years. And similar to single robot localization, a large portion of the existing algorithms can be consider variations of two main categories. The first family of algorithms make use of recursive Gaussian filters. Distributed versions of the Kalman Filter are proposed in [10] [11] to solve the cooperative localization problem. The Extended Kalman Filter (EKF) is utilized in [12] while also providing analytical expressions for the upper bound of the estimate uncertainty. In [13] the EKF is also used, but the algorithm is reinforced with an entropic criterion to select optimal measurements that reduce the global uncertainty. The advantage of using recursive Bayesian filters to fuse information lies in they are incremental in nature, which makes them applicable to real time estimation. Closed-form expressions for state estimation and update also facilitate computational speed. However these types of algorithms deal only with Gaussian noise which may not be the case for some real systems. And EKF linearizes the system dynamics around the state of estimate which is prone to failure when errors grow.

The second family of algorithms are built upon sampling-based nonparametric filters. Monte Carlo Localization methods are used in [14] to estimate the pose of each member robot while using grid cells to describe the entire particle set. A global collaborative localization algorithm is presented in [6] that also builds upon sample-based Markov localization. In addition, [15][16][17] have all approached the problem with different variations of the Particle Filter and have also applied their algorithm in the SLAM (Simultaneous Localization and Mapping) context. Further experimental validation is provided in [15] and [17]. Grid-based and sampling-based Markov localization techniques usually address the problem globally and can be improved via carefully designed resampling processes to counteract localization failures. They can also be used to accommodate non-Gaussian noise models. However like all sampling-based approaches, a large number of grids/samples are usually needed to acquire reasonable outcomes and the computational cost grows dramatically with the dimension of the problem. The following subsections present in detail one representative technique from each of the above categories. While variations and improvements to these algorithms do exist, the core benefits as well as shortcomings are preserved. A table comparing the two families of methods is provided at the end of the chapter.

2.1 Multi-Robot Localization Using Distributed Extended Kalman Filter

This section describes in detail a distributed localization technique based on the Extend Kalman Filter (EKF) ([11][18][19]). The problem setting is one where a group of robots are navigating a planar open space and the state vector to be estimated is $\vec{x} = [x, y, \theta]^T$. For the sake of description we assume robot i is the robot to be localized and robots $1, ..., k, ..., n$ are its n neighbors. At time t_k robot i detects robot k and measures their relative pose. The nonlinear system dynamics for each robot is of form

$$\vec{x}(t_{k+1}) = f(\vec{x}(t_k), \vec{u}(t_k)) \tag{2.1}$$

where $\vec{x}$ is the state vector to be estimated, $\vec{u}$ is the system input. Here we let the input to the system be proprioceptive sensor measurements (encoder measured velocities, etc). The estimation system dynamics is

$$\hat{\vec{x}}(t_{k+1}) = g(\hat{\vec{x}}(t_k), \vec{u}_m(t_k)) \tag{2.2}$$

If we assume errors occur in the inputs (i.e. $\vec{u} = \vec{u}_m + \vec{w}$ where $\vec{w}$ is the process noise vector) and are zero mean white noises with covariance $\Sigma_w = E[\vec{w}\vec{w}^T]$ and let $\tilde{\vec{x}} = \vec{x} - \hat{\vec{x}}$, the linearized error-state propagation equation takes the general form

$$\tilde{\vec{x}}(t_{k+1}) = \Phi(t_{k+1}, t_k)\tilde{\vec{x}}(t_k) + G(t_k)\vec{w}(t_k) \tag{2.3}$$

The algorithm then consists of four steps introduced as follows.

Prediction

Since initially the system of the multi-robot team is fully decoupled, the propagation of the state mean and covariance can be done locally by each robot. The estimated system dynamics is used for the state propagation step

$$\boxed{\hat{\vec{x}}_i(t_{k+1}^-) = g_i(\hat{\vec{x}}_i(t_k), \vec{u}_{im}(t_k))} \tag{2.4}$$

Define the system noise covariance by $Q_{di} = G_i(t_k)\Sigma_{iw}G_i(t_k)^T$, the propagation of the state covariance then follows

$$\boxed{P_i(t_{k+1}^-) = \Phi_i(t_{k+1}, t_k)P_i(t_k^+)\Phi_i^T(t_{k+1}, t_k) + Q_{di}(t_k)} \tag{2.5}$$

If no measurement is made, (2.4) and (2.5) can be used recursively to propagate the state of the system which is also called dead reckoning. And the covariance of the entire multi-robot system would be a block diagonal matrix with P_is on its diagonal.

Update

Now suppose robot i detects robot k and a relative measurement is made by its exteroceptive sensor (laser, sonar, etc)

$$\vec{z}_{ik}(t_{k+1}) = \begin{bmatrix} {}^i x_k(t_{k+1}) \\ {}^i y_k(t_{k+1}) \\ {}^i \theta_k(t_{k+1}) \end{bmatrix} + \vec{n}_{ik}(t_{k+1}) \tag{2.6}$$

where $\vec{n}_{ik}$ is the measurement noise with zero mean and covariance $R_{ik} = E[\vec{n}_{ik}\vec{n}_{ik}^T]$. Let the estimated measurement model be

$$\hat{\vec{z}}_{ik}(t_{k+1}) = \begin{bmatrix} C^T(\hat{\theta}_i)\left(\begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix} - \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix}\right) \\ \hat{\theta}_k - \hat{\theta}_i \end{bmatrix} \tag{2.7}$$

where $C(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$. Now define the measurement error $\tilde{\vec{z}} = \vec{z} - \hat{\vec{z}}$, the linearized measurement error model is then given by

$$\tilde{\vec{z}}_{ik} = -\Gamma_i^T \tilde{H}_{ik} \tilde{\vec{x}}(t_{k+1}) + \vec{n}_{ik}(t_{k+1}) \tag{2.8}$$

where

$$\begin{aligned}
\tilde{H}_{ik} &= \begin{bmatrix} I_{2x2} & J(\hat{p}_k - \hat{p}_i) \\ \vec{0}_{1x2} & 1 \end{bmatrix} \\
\Gamma_i &= \begin{bmatrix} C(\hat{\theta}_i) & \vec{0}_{2x1} \\ \vec{0}_{1x2} & 1 \end{bmatrix} \\
J &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \\
\hat{p}_i &= [\hat{x}_i(t_{k+1}^-) \;\; \hat{y}_i(t_{k+1}^-)], \hat{\theta}_i = \hat{\theta}_i(t_{k+1}^-)
\end{aligned} \tag{2.9}$$

Define

$$\begin{aligned}
H_{ik} &= \Gamma_{ik}^T \begin{bmatrix} -\tilde{H}_{ik} & I_{3x3} & 0_{3x3} \end{bmatrix} \\
S_{ik}(t_{k+1}) &= \Gamma_i^T \tilde{S}_{ik} \Gamma_i \\
\tilde{S}_{ik} &= \tilde{H}_{ik} P_i(t_{k+1}^-) \tilde{H}_{ik}^T + P_k(t_{k+1}^-) + \tilde{R}_{ik} \\
\tilde{R}_{ik} &= \Gamma_i R_{ik} \Gamma_i^T
\end{aligned} \tag{2.10}$$

The state update equation for robot i becomes

$$\boxed{\hat{\vec{x}}_i(t_{k+1}^+) = (I - P_i \tilde{H}_i^T \tilde{S}_{ik}^{-1}) \hat{\vec{x}}_i(t_{k+1}^-) + P_i \tilde{H}_i^T \tilde{S}_{ik}^{-1} (\hat{\vec{x}}_k(t_{k+1}^-) - \Gamma_i \vec{z}_{ik})} \tag{2.11}$$

For the state covariance update

$$\boxed{P_i(t_{t+1}^+) = P_i(t_{t+1}^-) - P_i(t_{t+1}^-) \tilde{H}_{ik}^T \tilde{S}_{ik}^{-1} \tilde{H}_{ik} P_i(t_{t+1}^-)} \tag{2.12}$$

2.2 Multi-Robot Monte Carlo Localization

Like all probabilistic localization methods, Monte Carlo Localization (MCL) also represents the robot positions as a distribution function (also called belief functions). However unlike parametric filters (KF, EKF described above), MCL relies on sampling-based representations where posterior beliefs are given by a set of weighted random samples called particles denoted by $S_i = \{s_i | i = 1, ..., K\}$ where $s_i = (p_i, w_i)$ with $p_i = (x_i, y_i, \theta_i)$ being the states to be estimated and $w_i > 0$ being a positive weighting factor restricting $\sum_{i=1}^{k} w_i = 1$ analogous to a discrete probability. Perhaps one of the most well known multi-robot MCL method is presented by [6]. The core of the method lies in that instead of maintaining a single belief over all robot poses, a factorial representation approximate is used to distribute the algorithm local to each robot. The factorial representation assumes that the distribution of the robot's belief (denoted by L) is the product of its N marginals, i,e,

$$P(L_1(t), ..., L_N(t)|z(t)) = P(L_1(t)|z(t)) \times ... \times P(L_N(t)|z(t)) \tag{2.13}$$

where $z(t)$ represents the measurements may it be interoceptive or exteroceptive sensor measurements. when $z(t)$ is an odometry measurement or measurement of the environment, the localization follows that of a particle filter. However when $z(t)$ represents a detection (here we also assume robot i detects robot k and and $z(t)$ provides the location of robot i relative to robot k), we have the incremental update equation

$$Bel_i(p) = Bel_i(p) \int_G P(L_i = p | L_k = p', z) Bel_k(p') \, \mathrm{d}p' \tag{2.14}$$

where $\int_G P(L_i = p | L_k = p', z) Bel_k(p') \, \mathrm{d}p'$ is robot k's estimate of where robot i is. And if r_m is reverted to provide the location of k relative to i, simple modifications can be made to (2.14) for robot k to estimate robot i's position.

To complete the algorithm, a flowchart is provided in figure 2.1 to illustrate the main steps [6]. The odometry update step correspond to the motion propagation step in the EKF method where the robot's velocity motion model is used to propagate it's belief using the odometry reading as inputs. p_{old} is the robot's position estimate before the input is applied. Also it is important to notice that (2.14) requires the multiplication of two probability densities functions (pdf) which is not straightforward since the the distributions are represented by a set of particles and hence difficult to find a correspondence between particles. [6] approached this problem with by transforming the particle sets into piecewise constant density functions using density trees (see [20],[21],[22],[23],[24] for details on density trees) .

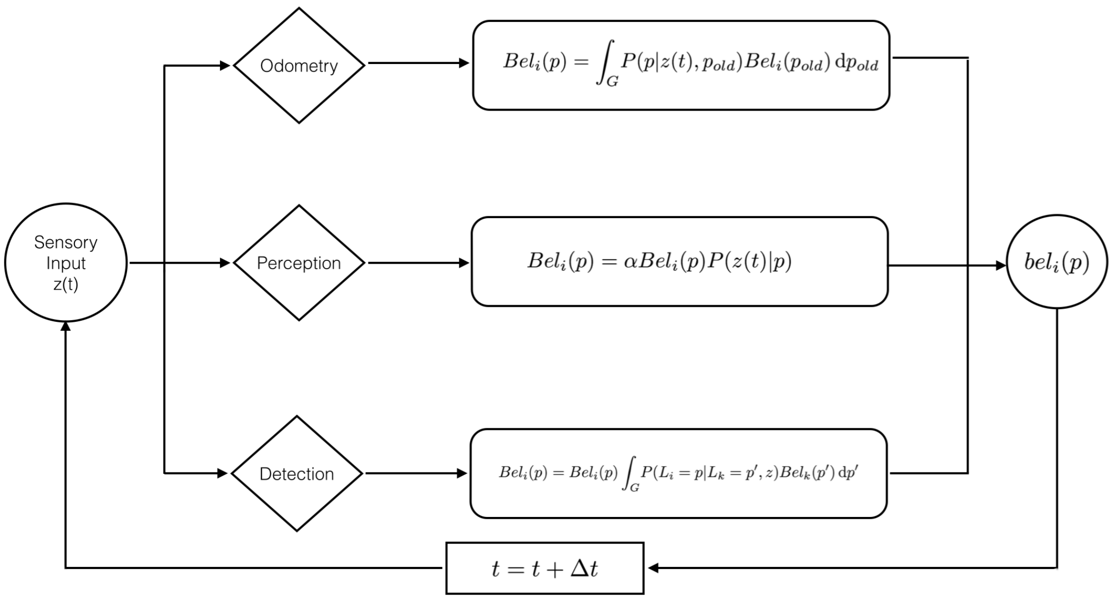

Sensory Input z(t)
Odometry
Perception
Detection
$Bel_i(p) = \int_G P(p|z(t), p_{old}) Bel_i(p_{old})\, \mathrm{d}p_{old}$
$Bel_i(p) = \alpha Bel_i(p) P(z(t)|p)$
$Bel_i(p) = Bel_i(p) \int_G P(L_i = p|L_k = p', z) Bel_k(p')\, \mathrm{d}p'$
$bel_i(p)$
$t = t + \Delta t$

FIGURE 2.1: Multi-Robot Monte Carlo Localization

Upon approximating the density in (2.14) using particles, the resulting sample set is transformed into a density tree. The density values are multiplied to each particle (within the constant region defined for that density value) of robot i . these density values replace the $Bel_i(p)$ term on the right hand side of (2.14) and results in a updated posterior distribution for robot i reflecting the detection of robot k according to

$$Bel_i(p) = \rho_{tree} \int_G P(L_i = p|L_k = p', z) Bel_k(p')\, \mathrm{d}p' \tag{2.15}$$

This chapter ends with a comparison of the two methods introduced namely the distributed EKF and MCL.

Although just two methods, they represent the two main categories of localization techniques that currently dominates the field. Both possesses their own pros and cons and the choice of which depend heavily on the type of applications they are desired for and the two approaches can potentially be combined to yield superior outcomes.

Table 2.1: Comparison Between Distributed EKF And MCL

	Distributed EKF	**Multi-Robot MCL**
Restrictions on Error Distribution	Requires Gaussian process and measurement error	Nonparametric particle representation of posterior belief, no assumptions on noise distribution
Global Localization	No	Yes
State Recovery	No	Possible given a well designed resampling process
Localization Accuarcy	Accurate when error is small	Depends on the number of particles used
Computational Cost	Small due to the closed form propagation and update equations	Depends on the number of particles. Can increase dramatically with the dimension of the state space
Ease of Implementation	Simple	Can be involved
Robustness	Prone to error due to linearization	Quite resistant to errors given multiple beliefs are maintained simultaneously
Process Multiple Detections Simultaneously	No	No
Complexity Relative To Team Size	Fully distributed. Complexity independent of team size	Complexity independent of team size

Chapter 3

Mathematical Background for The Group $SE(n)$ and Exponential Mapping

3.1 The Special Euclidean Group and Exponential Coordinates

3.1.1 The Special Euclidean Motion Group

The proposed technique is largely based on the notion of groups and its parametrization so it will be necessary to properly define the concept of groups. According to [9], a *group* is defined as a pair $(G, \circ)$ consisting of a set G and a binary operator $\circ$ such that operations are closed under the operator $\circ$, associative under this operator for all elements of G, meanwhile there also exists an identity element e such that for all elements $g \in G$, $g \circ e = e \circ g = g$ and for each $g \in G$ there exist an inverse element g^{-1} such that $g \circ g^{-1} = g^{-1} \circ g = e$. For engineering applications, the group of most interest is the Special Euclidean (SE) Group since rigid body motions are elements of the SE group and can be represented by the set of homogeneous matrices defined as

$$SE(n) = \left\{ \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \middle| R \in SO(n), \mathbf{t} \in \Re^n \right\} \tag{3.1}$$

where $SO(n)$ is the Special Orthogonal group representing rotations that can be presented as $n \times n$ rotation matrices and $\Re^n$ is the n dimensional vector space representing translations. Since we are only interested in rigid body motion, n just takes only two values particularly $n = 2$ for planar motion and $n = 3$ for 6 dimensional space motion. Going back to the original definition of groups, the Special Euclidean Group is also a form of the *Matrix Lie Group* whereby each of its elements is a real-valued matrix that defines a differentiable manifold and the binary operation is simply the matrix multiplication.

Now we introduce the concept of *Lie Algebra.* Again following [9], elements of a matrix Lie group can be are written as $g(t) = e^X$ for $X \in \mathcal{G}$ where the set $\mathcal{G}$ is the matrix Lie Algebra of G. The Lie Algebra for $SE(2)$ (denoted by $se(2)$) can be represented by the linear combination of a set of basis

$$E_1^{se(2)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad E_2^{se(2)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad E_3^{se(2)} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The exponential coordinate for an element of $SE(2)$ can be defined as $\mathbf{x}^{se(2)} = [v_1, v_2, \alpha]^T$ and under this definition an element of the the Lie algebra $se(2)$ can be written as a 3×3 matrix

$$X^{se(2)} = \hat{\mathbf{x}}^{se(2)} = \begin{bmatrix} 0 & -\alpha & v_1 \\ \alpha & 0 & v_2 \\ 0 & 0 & 0 \end{bmatrix} = \sum_{i=1}^{3} E_i^{se(2)} x_i^{se(2)}, \text{ and } X^{se(2)\vee} = \mathbf{x}^{se(2)} \tag{3.2}$$

For the case of $SE(3)$, the basis elements of the Lie algebra $se(3)$ are a set of 4×4 matrices defined by

$$E_1^{se(3)} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, E_2^{se(3)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, E_3^{se(3)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$E_4^{se(3)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, E_5^{se(3)} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, E_6^{se(3)} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

The corresponding exponential coordinate for an element of $SE(3)$ is then $\mathbf{x}^{se(3)} = [v_1, v_2, v_3, \alpha, \beta, \gamma]^T$ and an element of $se(3)$ is

$$X^{se(3)} = \hat{\mathbf{x}}^{se(3)} = \begin{bmatrix} 0 & -\gamma & \beta & v_1 \\ \gamma & 0 & -\alpha & v_2 \\ -\beta & \alpha & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \sum_{i=1}^{6} E_i^{se(3)} x_i^{se(3)}, \text{ and } X^{se(3)\vee} = \mathbf{x}^{se(3)} \tag{3.3}$$

It can be observed that taking the matrix exponential of the basis elements result in the homogeneous matrix of the corresponding motion. As an example

$$\exp(\theta E_1^{se(3)}) = \begin{bmatrix} 1 & 0 & 0 & \theta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

which corresponds to translation in the current x direction by an amount θ. And

$$\exp(\theta E_4^{se(3)}) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

which is rotation about the current x axis by θ. As mentioned before, the Special Euclidean group and its Lie algebra is related by

$$g = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} = \exp(X) \tag{3.6}$$

For $SE(2)$, the explicit relation is

$$\begin{aligned} R &= \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \\ \mathbf{t} &= \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} [v_2(\cos(\alpha) - 1) + v_1 \sin(\alpha)]/\alpha \\ [v_1(1 - \cos(\alpha)) + v_2 \sin(\alpha)]/\alpha \end{bmatrix} \end{aligned} \tag{3.7}$$

For $SE(3)$, first let $\mathbf{v} = [v_1, v_2, v_3]^T$, $\boldsymbol{\omega} = [\alpha, \beta, \gamma]^T$ and $\mathbf{skew}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix}$ be the skew symmetric matrix of vector $\boldsymbol{\omega}$. Thereby the explicit relation between $SE(3)$ and $se(3)$ is

$$\begin{aligned} R &= e^{\mathbf{skew}(\boldsymbol{\omega})} \\ \mathbf{t} &= \frac{(I - R)(\boldsymbol{\omega} \times \mathbf{v}) + \boldsymbol{\omega}\boldsymbol{\omega}^T \mathbf{v}}{\|\boldsymbol{\omega}\|^2} \end{aligned} \tag{3.8}$$

3.1.2 Adjoint Matrices

The *adjoint operator* $Ad(g)$ and $ad(X)$ are two important concepts to the derivations that follow so their definitions as well as relevant properties will be introduced in this section. To define the adjoints, we need to first define the inner product and Lie bracket operations for Lie algebras. According to [9], an *inner product* between arbitrary elements of the Lie algebra $Y = \sum_i y_i E_i$ and $Z = \sum_i z_i E_i$ can be defined such that

$$(Y, Z) = \sum_i^n y_i z_i \tag{3.9}$$

given $(E_i, E_j) = \delta_{ij}$ where δ_{ij} is the Dirac Delta Function. In addition, the *Lie Bracket* of Y, Z is defined as

$$[Y, Z] = YZ - ZY \tag{3.10}$$

With the above definitions in place, for $X, Y \in \mathcal{G}$ and $g \in G$, the adjoint operators are then

$$\begin{aligned} Ad(g)X &= \frac{d}{dt}(g \circ e^{tX} \circ g^{-1})|_{t=0} = \frac{d}{dt} exp(tgXg^{-1})|_{t=0} = gXg^{-1} \\ ad(X)Y &= \frac{d}{dt}(Ad(e^{tX})Y)|_{t=0} \end{aligned} \tag{3.11}$$

Since the adjoint operators are both linear operators, they can both be written as matrices that represent linear mapping. We call these matrices *adjoint matrices* and define them as (in component form)

$$\begin{aligned} [Ad(g)]_{ij} &= (E_i, Ad(g)E_j) = (E_i, gE_jg^{-1}) \\ [ad(X)]_{ij} &= (E_i, ad(X)E_j) = (E_i, [X, E_j]) \end{aligned} \tag{3.12}$$

And their matrix form definition is

$$\begin{aligned} [Ad(g)] &= [(gE_1g^{-1})^\vee, ..., (gE_ng^{-1})^\vee] \\ [ad(X)] &= [[X, E_1]^\vee, ..., [X, E_n]^\vee] \end{aligned} \tag{3.13}$$

Some important properties of the adjoint matrices that are used in the following calculations are listed as follow:

1. $Ad(\exp(X)) = \exp(ad(X))$
2. $ad(X)X^\vee = 0$
3. $ad(X)Y = XY - YX = [X, Y]$
4. $ad(X)Y^\vee = [X, Y]^\vee$
5. $ad([X, Y]) = ad(X)ad(Y) - ad(Y)ad(X)$
6. $ad(X)Y^\vee = -ad(Y)X^\vee$
7. $Ad(g_1)Ad(g_2)X = g_1(g_2Xg_2^{-1})g_1^{-1} = (g_1g_2)X(g_1g_2)^{-1} = Ad(g_1g_2)X$
8. $\log^\vee(g \circ e^X \circ g^{-1}) = Ad(g)\log^\vee(e^X)$

For $SE(2)$, the explicit form of the adjoint matrices are

$$Ad(g) = \begin{bmatrix} R & M\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \Re^{3\times3}, \quad ad(g) = \begin{bmatrix} -\alpha M & M\mathbf{v} \\ \mathbf{0}^T & 0 \end{bmatrix} \in \Re^{3\times3} \tag{3.14}$$

where $M = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$, R and $\mathbf{t}$ defined by (3.6). $(\mathbf{v}, \alpha) = (v_1, v_2.\alpha)$ are the exponential coordinates of $SE(2)$.

For $SE(3)$ this becomes

$$Ad(g) = \begin{bmatrix} \mathbf{0} & R \\ R & TR \end{bmatrix} \in \Re^{6\times6}, \quad ad(g) = \begin{bmatrix} \mathbf{0} & \Omega \\ \Omega & V \end{bmatrix} \in \Re^{6\times6} \tag{3.15}$$

where $\mathbf{v} = (v_1, v_2, v_3)^T, \boldsymbol{\omega} = (\alpha, \beta, \gamma)^T$ and $V = \hat{\mathbf{v}}, \Omega = \hat{\boldsymbol{\omega}}, T = \hat{\mathbf{t}}$.

3.1.3 The Baker-Campbell-Hausdorff Formula

The *Baker-Campbell-Hausdorf (BCH) formula* [9] serves as the core of the second order estimation of Gaussian convolutions (described in more detail in the next section). In essence, the BCH formula establishes a relationship between the Lie Bracket (defined in (3.10)) and the matrix exponential. Let $X, Y \in \mathcal{G}$ and define $Z(X,Y) = \log(e^X e^Y)$, the BCH formula then takes the form

$$Z(X,Y) = X+Y+\frac{1}{2}[X,Y]+\frac{1}{12}([X,[X,Y]]+[Y,[Y,X]])+\frac{1}{48}([Y,[X,[Y,X]]]+[X,[Y,[Y,X]]])+\cdots \tag{3.16}$$

This can be verified by expanding e^X, e^Y using matrix exponential Taylor series $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$ and substitute into the Taylor series for matrix logarithm

$$\log(e^X e^Y) = \log(I + (e^X e^Y - I)) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{(e^X e^Y - I)^k}{k} \tag{3.17}$$

Applying the $\vee$ operator to equation (3.16) results in

$$\begin{aligned}\mathbf{z} =&\mathbf{x}+\mathbf{y}+\frac{1}{2}ad(X)\mathbf{y}+\frac{1}{12}(ad(X)ad(X)\mathbf{y}+ad(Y)ad(Y)\mathbf{x})\\ &+\frac{1}{48}(ad(Y)ad(X)ad(Y)\mathbf{x}+ad(X)ad(Y)ad(Y)\mathbf{x})+\cdots\end{aligned} \tag{3.18}$$

Equations (3.16) and (3.18) are powerful tools that can be used in the group context similar to the Taylor Series and produce useful linearized or quadratic estimation results.

3.2 Gaussians on $SE(n)$ and Second Order Convolution Theory

A Gaussian on the $SE(n)$ is defined by

$$f(g;\mu,\Sigma)=\frac{1}{C(\Sigma)}\exp\left\{-\frac{1}{2}[\log^{\vee}(\mu^{-1}g)]^{T}\Sigma^{-1}[\log^{\vee}(\mu^{-1}g)]\right\} \tag{3.19}$$

where $\mu, g\in SE(n)$, $C(\Sigma)\approx(2\pi)^{\frac{d}{2}}\|\det(\Sigma)\|^{\frac{1}{2}}$ (d is the degree of freedom of the space defined by $SE(n)$ where $d=6$ for $SE(3)$ and $d=3$ for $SE(2)$) is the normalizing factor when $\|\Sigma\|$ is small.

For a domain of integration $G=SE(n)$, the mean of the above Gaussian is defined by

$$\int_G \log^{\vee}(\mu^{-1}g)f(g)\,\mathrm{d}g \tag{3.20}$$

and the covariance given by

$$\Sigma=\int_G[\log^{\vee}(\mu^{-1}g)][\log^{\vee}(\mu^{-1}g)]^{T}f(g)\,\mathrm{d}g \tag{3.21}$$

Given two Gaussians $f_1(g)=f(g;\mu_1,\Sigma_1)$ and $f_2(g)=f(g;\mu_2,\Sigma_2)$ in the form of (3.19), their convolution is defined as

$$\begin{aligned}(f_1 * f_2)(g) &= \int_G f_1(h) f_2(h^{-1}g)\,\mathrm{d}h \\ &= \int_G \rho_1(\mu_1^{-1}h)\rho_2(\mu_2^{-1}h^{-1}g)\,\mathrm{d}h\end{aligned}$$

With $\rho_i(g) = f(g; e, \Sigma_i)$ being Gaussian centered at the identity. It is proven (refer to [25]) that the convolution $(f_1 * f_2)(g)$ results (to the second order) in a Gaussian with mean

$$\mu_{1*2} = \mu_1\mu_2 \tag{3.22}$$

and covariance

$$\Sigma_{1*2} = A + B + F(A, B) \tag{3.23}$$

with the terms A, B and F defined by

$$A = Ad(\mu_2^{-1})\Sigma_1 Ad(\mu_2^{-1})^T \tag{3.24}$$

$$B = \Sigma_2 \tag{3.25}$$

where

$$\begin{aligned}F(A, B) = &\frac{1}{4}\sum_{i,j=1}^{d} ad(E_i) B ad(E_j)^T A_{ij} \\ &+ \frac{1}{12}\left\{[\sum_{i,j=1}^{d} A''_{ij}]B + B^T[\sum_{i,j=1}^{d} A''_{ij}]^T\right\} \\ &+ \frac{1}{12}\left\{[\sum_{i,j=1}^{d} B''_{ij}]A + A^T[\sum_{i,j=1}^{d} B''_{ij}]^T\right\}\end{aligned} \tag{3.26}$$

where

$$A''_{ij} = ad(E_i)ad(E_j)A_{ij} \tag{3.27}$$

$$B''_{ij} = ad(E_i)ad(E_j)B_{ij} \tag{3.28}$$

The above results are usually used for $SE(2)$ and $SE(3)$ where the basis elements E_i as well as Ad and ad matrices as defined in the previous section.

Chapter 4

Derivation of Second Order Bayesian Sensor Fusion on the SE Group

This chapter presents a detailed derivation of proposed technique. Again the technique focuses on fusing the relative measurements of neighboring robots and their pose information to reduce the estimation uncertainty of the current robot. A probabilistic approach is adapted where uncertainties in the robot positions and sensor measurements are modeled by Gaussians. In addition, the technique is developed under exponential coordinates for the reason that the motion of a stochastic system with differential constraints is modeled more accurately with Gaussians in exponential coordinates than that in Cartesian coordinates. The theory will first be developed for a system of two robots (which builds on [8] by taking sensor noise into consideration) and be extended to the multi-robot scenario.

4.1 Localization for A Robot Pair

The problem is given by two mobile robots i and j moving in the field whose position priors are provided by two Gaussians $f(a_i^{-1}g_i; \mu_i, \Sigma_i)$ and $f(a_j^{-1}g_j; \mu_j, \Sigma_j)$. For the planar case, $a_i, a_j \in SE(2)$ are the known initial positions of the robots relative to the world frame at $t = 0$. At time t, $\mu_i, \mu_j \in SE(2)$ and $\Sigma_i, \Sigma_j \in R^{3\times 3}$ are the means (defined relative to the initial frames a_i, a_j)and covariances obtained from a previous

prediction step which we'll also assume to be known. In addition, a sensor measurement of robot j relative to i is also obtained at time t and is given by the homogeneous matrix $m_{ij} \in SE(2)$. Since we assume the sensor has Gaussian noise, its distribution is then characterized by a Gaussian of the form $M_{ij}(g_i, g_j) = f(g_j; g_i m_{ij}, \Sigma_m)$ which says that according to the sensor, the position of robot j with respective to robot i has a mean of $g_i m_{ij}$ and covariance of Σ_m.
The goal is then to calculate a posterior for the position of robot i using the sensor measurement to update the its prior. Because the sensor provides a relative measurement, we first formulate the joint prior of robot i and j making the assumption that the priors are independent of each other, giving

$$p_{ij}(g_i, g_j) = f(a_i^{-1} g_i; \mu_i; \Sigma_i) f(a_j^{-1} g_j; \mu_j; \Sigma_j). \tag{4.1}$$

Then according to Bayes' Theorem, the joint posterior is given by

$$\overline{p}_{ij} = \eta_1 p_{ij} M_{ij} \tag{4.2}$$

where η_1 is a constant normalizing factor. Similar normalizing factors result in all fusion processes that follow and will be denoted by η_i. To save space in the derivations, we will denote $\rho_i(\mu_i^{-1} g_i) = f(g_i; \mu_i; \Sigma_i)$ and the rest follows where $\rho_i(g)$ is a Gaussian with mean at the identity. The marginal distribution of the joint posterior for robot i is then

$$\begin{aligned}
\overline{p_i}(g_i) &= f(g_i; \overline{\mu}_i, \overline{\Sigma}_i) \\
&= \eta_2 \int_G p_{ij}(g_i, g_j) M_{ij}(g_i, g_j)\, \mathrm{d}g_j \\
&= \eta_2 \int_G \rho_i(\mu_i^{-1} a_i^{-1} g_i) \rho_j(\mu_j^{-1} a_j^{-1} g_j) \rho_m(m_{ij}^{-1} g_i^{-1} g_j)\, \mathrm{d}g_j \\
&= \eta_2 \rho_i(\mu_i^{-1} a_i^{-1} g_i) \int_G \rho_j(\mu_j^{-1} a_j^{-1} g_j) \rho_m(m_{ij}^{-1} g_i^{-1} g_j)\, \mathrm{d}g_j
\end{aligned} \tag{4.3}$$

The goal is to find closed-form expressions for $\overline{\mu}_i$ and $\overline{\Sigma}_i$. Since ρ_m is symmetric around the mean, we have $\rho_m(m_{ij}^{-1}g_i^{-1}g_j) = \rho_m(g_j^{-1}g_i m_{ij})$. Letting $g' = g_i m_{ij}$, (4.3) becomes

$$\begin{aligned}\overline{p_i}(g_i) &= \eta_2 \rho_i(\mu_i^{-1}a_i^{-1}g_i)\int_G \rho_j(\mu_j^{-1}a_j^{-1}g_j)\rho_m(g_j^{-1}g_i m_{ij})\,\mathrm{d}g_j \\ &= \eta_2 \rho_i(\mu_i^{-1}a_i^{-1}g_i)\int_G \rho_j(\mu_j^{-1}a_j^{-1}g_j)\rho_m(e^{-1}g_j^{-1}g')\,\mathrm{d}g_j\end{aligned} \tag{4.4}$$

with $e \in SE(2)$ been the identity element of SE(2). According to the definition of convolution in Chapter 3, the integral in (4.4) defines a convolution $(f_1 * f_2)(g')$ where $f_1(g') = f(g'; a_j\mu_j, \Sigma_j)$ and $f_2(g') = f(g'; e, \Sigma_m)$. Let $f_{1*2}(g'; \mu_{1*2}, \Sigma_{1*2}) = (f_1 * f_2)(g')$, then (3.22)-(3.28) can be used to calculate the closed-form expressions of μ_{1*2} (which equals to $a_j\mu_j$) and Σ_{1*2}. With the integral taken care of, equation (4.4) becomes

$$\begin{aligned}\overline{p_i}(g_i) &= f(g_i; \overline{\mu}_i, \overline{\Sigma}_i) \\ &= \eta_2 f(\mu_i^{-1}a_i^{-1}g_i; e, \Sigma_i) f(g_i m_{ij}; a_j\mu_j, \Sigma_{1*2})\end{aligned} \tag{4.5}$$

For a posterior of robot i formulated in the form of (4.5), the fusion technique developed in [8] can be used to derive the closed-form expressions for $\overline{\mu}_i$ and $\overline{\Sigma}_i$.

4.2 Localization for Multi-Robot Team

Now we are ready to extend the technique to multi-robot localization. Similar to the previous subsection, the posterior of robot i is what we are trying to estimate, but instead of taking measurement from a single neighboring robot, multiple measurements are taken from however many neighboring robots that are in the sensing range (for derivation purposes we label the neighboring robots as $1, 2, ..., n$). Following a similar approach we have the joint prior

$$\begin{aligned}p_{i,1,\ldots,n} &= f(a_i^{-1}g_i; \mu_i; \Sigma_i) f(a_1^{-1}g_1; \mu_1; \Sigma_1)...f(a_n^{-1}g_n; \mu_n; \Sigma_n) \\ &= \rho_i(\mu_i^{-1}a_i^{-1}g_i)\rho_1(\mu_1^{-1}a_1^{-1}g_1)...\rho_n(\mu_n^{-1}a_n^{-1}g_n)\end{aligned} \tag{4.6}$$

Let $M_{in} = f(g_n; g_i m_{in}, \Sigma_{in})$ be the distribution of the sensor measurement of robot n relative to robot i and assume independence among all the measurements, we have joint measurement distribution

$$M_{i,1,...,n} = M_{i1}M_{i2}...M_{in} \tag{4.7}$$

To further save space, we will write in short $\rho_i = \rho_i(\mu_i^{-1}a_i^{-1}g_i)$ as the position priors and $\rho_{in} = \rho_{in}(m_{in}^{-1}g_i^{-1}g_n) = M_{in}$ as the measurement distributions. We will further define $g'_{in} = g_i m_{in}$. The posterior for robot i is then

$$\begin{aligned}\overline{p_i}(g_i) &= f(g_i; \overline{\mu}_i, \overline{\Sigma}_i)\\ &= \eta_3 \int_G \int_G ... \int_G p_{i,1,...,n} M_{i,1,...,n}\,\mathrm{d}g_1\,\mathrm{d}g_2 ... \,\mathrm{d}g_n\\ &= \eta_3 \rho_i \left(\int_G \rho_1 \rho_{i1}\,\mathrm{d}g_1\right)\left(\int_G \rho_2\rho_{i2}\,\mathrm{d}g_2\right) ... \left(\int_G \rho_n \rho_{in}\,\mathrm{d}g_n\right)\end{aligned} \tag{4.8}$$

Let $f_n(g'_{in}) = f(g'_{in}; a_n\mu_n, \Sigma_n)$ and $f_{in}(g'_{in}) = f(g'_{in}; e, \Sigma_{in})$, then (4.8) becomes

$$\begin{aligned}\overline{p_i}(g_i) &= f(g_i; \overline{\mu}_i, \overline{\Sigma}_i)\\ &= \eta_3 \rho_i(\mu_i^{-1}a_i^{-1}g_i)(f_1 * f_{i1})(g'_{i1})(f_2 * f_{i2})(g'_{i2})...(f_n * f_{in})(g'_{in})\end{aligned} \tag{4.9}$$

Calculating the convolutions using equations (3.22)-(3.28), we finally arrive at

$$\boxed{\begin{aligned}\overline{p_i}(g_i) &= f(g_i; \overline{\mu}_i, \overline{\Sigma}_i)\\ &= \eta_3 f(\mu_i^{-1}a_i^{-1}g_i; e, \Sigma_i) f(g_i m_{i1}; a_1\mu_1, \Sigma_{1*i1}) \times ...\\ &\qquad \times f(g_i m_{in}; a_n\mu_n, \Sigma_{n*in})\end{aligned}} \tag{4.10}$$

An extension of the method provided by [8] (which fuses only one measurement) gives the equations to calculate $\overline{\mu}_i$ and $\overline{\Sigma}_i$, and is presented as follows:

For neighboring robots $1, ..., k, ..., n$

1. Define $q_k = m_{ik}\mu_k^{-1}a_k^{-1}a_i\mu_i$

2. Define $\exp(\hat{x}_k) = q_k$.
3. Define $\Gamma_k = (I + \frac{1}{2}ad(\hat{x}_k))$
4. Define $S_i = \Gamma_i^T \Sigma_i^{-1} \Gamma_i$
5. Define $S_k = \Gamma_m^T Ad^{-T}(m_{ik}) \Sigma_{k*ik}^{-1} Ad^{-1}(m_{ik}) \Gamma_k$
6. $\overline{S}' = S_i + \sum\limits_{k=1}^{n} S_k$
7. $\overline{x}' = \bar{S}'^{-1} \sum\limits_{k=1}^{n} S_k x_k$

With the above definitions, the posterior distribution for robot i can be calculated by

$$\boxed{\begin{aligned} \overline{\Sigma}_i = &\quad \bar{\Gamma}' \bar{S}'^{-1} \bar{\Gamma}'^T \\ \overline{\mu}_i = &\quad \mu_i \exp(-\hat{\bar{x}}') \end{aligned}} \tag{4.11}$$

with the operator $\wedge$ and $\vee$ defined in Chapter 3.

4.3 A Complete Distributed Localization Algorithm Using Bayesian Filter In Exponential Coordinates

The fusion technique introduced above defines the state update step for the proposed localization method. However like all Bayesian Filters a complete recursive filter for state estimation consists of a state prediction step as well as a state update step. This section serves to provide the proposed algorithm in such a form.

Similar to the above setting, suppose at time t_k robot i is the robot to be localized, robots $1, ..., k, ..., n$ are its n neighbors. Their means are $\mu_i(t_k), \mu_k(t_k)$ and covariances $\Sigma_i(t_k), \Sigma_k(t_k)$ respectively. Let the stochastic differential equation (SDE) governing the motion of the robots be of the form

$$(g^{-1}\dot{g})^{\vee} dt = \mathbf{h} dt + H d\mathbf{w} \tag{4.12}$$

When $g \approx I$ and for a sampling time Δt a constant command u is given to the system resulting in motion of the system from t_k to t_{k+1}, the distributed localization scheme that estimates the location of robot i at time t_{k+1} follows two steps (letting $\Delta t = t_{k+1} - t_k$).

Prediction

$$\boxed{\begin{aligned}&\mu_i(\Delta t) = \exp(\int_0^{\Delta t} \hat{\boldsymbol{h}}_i \, \mathrm{d}\tau)\\&\Sigma_i(\Delta t) = \int_0^{\Delta t} Ad(\mu_i^{-1}(\tau))H_i H_i^T Ad^T(\mu_i^{-1}(\tau))\, \mathrm{d}\tau\\&\mu_i(t_{k+1}^-) = \mu_i(t_k) \circ \mu_i(\Delta t)\\&\Sigma_i(t_{k+1}^-) = A_i(t_k) + B_i(t_k) + F(A_i(t_k), B_i(t_k))\end{aligned}} \tag{4.13}$$

where

$$\begin{aligned}&A_i(t_k) = Ad(\mu_i(\Delta t)^{-1})\Sigma_i(t_k^+)Ad(\mu_i(\Delta t)^{-1})^T\\&B_i(t_k) = \Sigma_i(\Delta t)\\&A_i(t_k)_{ij}^{''} = ad(E_i)ad(E_j)A_i(t_k)_{ij}\\&B_i(t_k)_{ij}^{''} = ad(E_i)ad(E_j)B_i(t_k)_{ij}\end{aligned} \tag{4.14}$$

$$\begin{aligned}F_i(A_i(t_k), B_i(t_k)) = &\frac{1}{4}\sum_{i,j=1}^{d} ad(E_i)B_i(t_k)ad(E_j)^T A_i(t_k)_{ij}\\&+ \frac{1}{12}\left\{[\sum_{i,j=1}^{d} A_i(t_k)_{ij}^{''}]B_i(t) + B_i(t)^T[\sum_{i,j=1}^{d} A_i(t)_{ij}^{''}]^T\right\}\\&+ \frac{1}{12}\left\{[\sum_{i,j=1}^{d} B_i(t_k)_{ij}^{''}]A_i(t_k) + A_i(t_k)^T[\sum_{i,j=1}^{d} B_i(t_k)_{ij}^{''}]^T\right\}\end{aligned} \tag{4.15}$$

In the above equations, $\mu_i(\Delta t) and \Sigma_i(\Delta t)$ defines the incremental distribution resulting solely from the input given in the Δt time frame which location is given with respective to $\mu_i(t_k)$ not the fixed world frame. In order to take into account the uncertainties already present at time t given by $\Sigma_i(t_k)$, the distribution at time t_k is convolved with the incremental distribution resulting in the predicted distribution given by $\mu_i(t_{k+1}^-)^-, \Sigma_i(t_{k+1}^-)$.

Update

Now to incorporate the relative measurements, for each of the neighboring robots $1, ..., k, ..., n$, obtain the measurement distribution $m_{ik}(t), \Sigma_{ik}(t)$, then

$$\begin{aligned} A_{ik}(t_k) &= Ad(m_{ik}(t_k)^{-1})\Sigma_k(t_k)Ad(m_{ik}(t_k)^{-1})^T \\ B_{ik}(t_k) &= \Sigma_{ik}(t_k) \\ A_{ik}(t_k)''_{ij} &= ad(E_i)ad(E_j)A_{ik}(t_k)_{ij} \\ B_{ik}(t_k)''_{ij} &= ad(E_i)ad(E_j)B_{ik}(t_k)_{ij} \end{aligned} \tag{4.16}$$

$$\begin{aligned} F_{ik}(A_{ik}(t_k), B_{ik}(t_k)) = & \frac{1}{4}\sum_{i,j=1}^{d} ad(E_i)B_i(t_k)ad(E_j)^T A_{ik}(t_k)_{ij} \\ & + \frac{1}{12}\left\{[\sum_{i,j=1}^{d} A_{ik}(t_k)''_{ij}]B_{ik}(t_k) + B_{ik}(t_k)^T[\sum_{i,j=1}^{d} A_{ik}(t_k)''_{ij}]^T\right\} \\ & + \frac{1}{12}\left\{[\sum_{i,j=1}^{d} B_{ik}(t_k)''_{ij}]A_{ik}(t_k) + A_{ik}(t_k)^T[\sum_{i,j=1}^{d} B_{ik}(t_k)''_{ij}]^T\right\} \end{aligned} \tag{4.17}$$

$$\Sigma_{k*ik}(t(t_k)) = A_{ik}(t_k) + B_{ik}(t_k) + F(A_{ik}(t_k), B_{ik}(t_k)) \tag{4.18}$$

1. Define $q_k(t_k) = m_{ik}(t_k)\mu_k(t_k)^{-1}a_k^{-1}a_i\mu_i(t_{k+1}^-)$
2. Define $\exp(\hat{x}_k(t_k)) = q_k(t_k)$.
3. Define $\Gamma_k(t_k) = (I + \frac{1}{2}ad(\hat{x}_k(t_k)))$
4. Define $S_i(t_k) = \Gamma_i(t_k)^T[\Sigma_i(t_{k+1}^-)]^{-1}\Gamma_i(t_k)$
5. Define $S_k(t_k) = \Gamma_m(t_k)^T Ad^{-T}(m_{ik}(t_k))\Sigma_{k*ik}(t_k)^{-1}Ad^{-1}(m_{ik}(t_k))\Gamma_k(t_k)$
6. $\overline{S}'(t_k) = S_i(t_k) + \sum_{k=1}^{n} S_k(t_k)$
7. $\overline{x}'(t_k) = \bar{S}'(t_k)^{-1}\sum_{k=1}^{n} S_k(t_k)x_k(t_k)$

$$\boxed{\begin{aligned} \Sigma_i(t_{k+1}^+) &= \bar{\Gamma}'(t_k)\bar{S}'(t_k)^{-1}\bar{\Gamma}'(t)^{t_k} \\ \mu_i(t_{k+1}^+) &= \mu_i(t_{k+1}^-)\exp(-\hat{\bar{x}}'(t_k)) \end{aligned}} \tag{4.19}$$

Chapter 5

Simulation And Discussions

A simulation experiment is setup to verify the proposed method. The simulation comes in two stages, first a Matlab simulation is used to provide a proof-of-concept and initial state validation. Then a more sophisticated simulation is implemented in the Robot Operating System/Gazebo framework where a 3D simulated world is powered by the Open Dynamics Engine(ODE) and various sensor models that generates realistic sensor feedbacks and plausible interactions between robots. The following sections serves to introduction details on the experimental setups as well as provide results from the simulation and discussions.

5.1 Matlab Simulation And Results

This section provides a verification for the proposed technique in a Matlab simulated environment. A team of two-wheeled differential drive robots are moving in the field. The given inputs are such that all robots move along a straight line or a circular arc. However, due to the stochastic nature of the systems, errors accumulate over time such that odometry or dynamics alone is insufficient in estimating the robot poses. The results from the previous sections can therefore be used to update the robots' knowledge of their poses with the help of measuring their positions relative to neighboring robots.

Figure(5.1) depicts a simple model of the two-wheeled differential drive robot which is very useful in modeling segway-like mobile bases and various multi-robot experimental

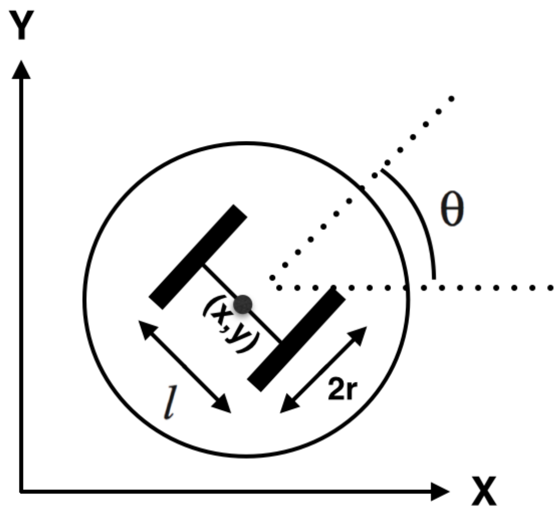


Figure 5.1: Simple model for a two-wheeled differential drive mobile system

platforms (E-pucks, iRobot create, Khepera, etc). According to [8], the kinematics of such a mobile robot can be characterized by the stochastic differential equation

$$(g^{-1}\dot{g})^{\vee}dt = \begin{bmatrix} \frac{r}{2}(\omega_1+\omega_2) \\ 0 \\ \frac{r}{2}(\omega_1-\omega_2) \end{bmatrix} dt + \sqrt{D} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{l} & -\frac{r}{l} \end{bmatrix} \begin{bmatrix} dw_1 \\ dw_2 \end{bmatrix} \tag{5.1}$$

where $g \in SE(2)$ is the homogenous matrix representing the pose of the robot, r is the wheel radius, l is the axle length, ω_1, ω_2 are the wheel angular velocities, dw_i are unit strength Wiener processes and D is a noise coefficient. This stochastic differential system can be simulated using the the Euler-Maruyama Method described in [26]. (5.1) can be written in short as

$$(g^{-1}\dot{g})^{\vee}dt = \mathbf{h}dt + \mathbf{H}d\mathbf{w} \tag{5.2}$$

when g is close to the identity, given an input pair $[\omega_1, \omega_2]^T$, the mean and covariance of system (5.1) can be estimated by

$$\mu(t) = \exp\left(\int_0^t \hat{\mathbf{h}}\, \mathrm{d}\tau\right) \tag{5.3}$$

$$\Sigma(t) = \int_0^t Ad(\mu^{-1}(\tau))\mathbf{H}\mathbf{H}^T Ad^T(\mu^{-1}(\tau))\,\mathrm{d}\tau \tag{5.4}$$

For simple motions like straight-line motion ($\omega_1 = \omega_2$) equations (5.3)(5.4) can be evaluated analytically as

$$\mu(t)_{st} = \begin{bmatrix} 1 & 0 & r\omega t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

$$\Sigma(t)_{st} = \begin{bmatrix} \frac{1}{2}Dr^2t & 0 & 0 \\ 0 & \frac{2D\omega^2r^4t^3}{3l^2} & \frac{D\omega r^3t^2}{l^2} \\ 0 & \frac{D\omega r^3t^2}{l^2} & \frac{2Dr^2t}{l^2} \end{bmatrix} \tag{5.6}$$

The same can be done with circular motion of constant curvature

$$\mu(t)_{cir} = \begin{bmatrix} \cos(\dot{\alpha}t) & -\sin(\dot{\alpha}t) & a\sin(\dot{\alpha}t) \\ \sin(\dot{\alpha}t) & \cos(\dot{\alpha}t) & a(1-\cos(\dot{\alpha}t)) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.7}$$

$$\Sigma(t)_{cir} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \tag{5.8}$$

where

$$\begin{aligned} \sigma_{11} &= \frac{c}{8}[(4a^2+l^2)(2\dot{\alpha}t+\sin(2\dot{\alpha}t)) + 16a^2(\dot{\alpha}t - \sin(2\dot{\alpha}t))] \\ \sigma_{12} = \sigma_{21} &= \frac{-c}{2}[4a^2(-1+\cos(2\dot{\alpha}t)) + l^2]\sin(\dot{\alpha}t/2)^2 \\ \sigma_{13} = \sigma_{31} &= 2ca(\dot{\alpha}t - \sin(2\dot{\alpha}t)) \\ \sigma_{23} = \sigma_{32} &= -2ca(-1+\cos(\dot{\alpha}t)) \\ \sigma_{33} &= 2c\dot{\alpha} \\ c &= \frac{Dr^2}{l^2\dot{\alpha}} \end{aligned} \tag{5.9}$$

With the pose priors calculated with (5.5)-(5.8), equations (4.16)-(4.19) are then applied with sensor measurements to update the priors. For arbitrary inputs (ω_1, ω_2) an approximation will be applied to (5.3)(5.4) which will be discussed in the next section.

This example simulates localization of a robot team in straight and circular motion. In the set-up of this simulation, the model based parameters are set as $r = 0.033, l = 0.2$. The simulation parameters for straight-line motion are $D = 5, v = 0.5, T = 1.3$, $\omega_1 = \omega_2 = \frac{v}{r}$ and $T = 2$, $\omega_1 = 26.166, \omega_2 = 21.408$ (for circular motion). The explicit expressions of (24)(25) for these types of motion can be found in [8]. The true robot motions (equation (1.5)) are simulated 500 times using the Euler-Maruyama Method [26] and the end position of each trial is plotted in the following figures. It can be observed that the posterior such a stochastic differential system (SDE) results in a banana shaped distribution as is also discussed in [7].

In this simulation, all four robots are given the command to travel in a straight line for 1.3 seconds at 0.5 m/s or along an arc of constant curvature of $1m$ at 45 deg/s for 1 second. The blue dashed lines in the figures represent the desired path of travel with the blue points at the two ends representing initial to final position. However due to process noise each robot will eventually end up somewhere near the desired goal and our objective is to estimate their true position along with a quantification of our confidence of this estimate. Specifically for this example, the true pose of the middle robot (cyan colored) is what we're trying to estimate which we'll call robot i, while the neighboring robots (yellow) are members of this team where relative measurements are obtained from. Among all the sampled end positions, one position for each robot is chosen as the true end pose (red point) and this is used to generate the mean of the measurement distribution m_{in}.

As the first step, the prior mean and covariance of robot i is calculated using equations (5.5)-(5.9) and plotted in figure 5.2 and 5.3, the resultant prediction aligns perfectly with the desire path (blue dash line), and the error "ellipse" marginalized over the heading angle is also plotted from the calculated covariance (magenta loop). Since this error "ellipse" is a contour of the resultant distribution, It can be observed that a Gaussian distribution under exponential coordinates is a much better fit for characterizing the uncertainties in an SDE of this kind than that under Cartesian coordinates. It is obvious that this prediction gives the same resultant distribution regardless of the true position and is only effected by the system dynamics and input commands. Therefore the next step is to update this prediction with measurements relative to neighboring robots.

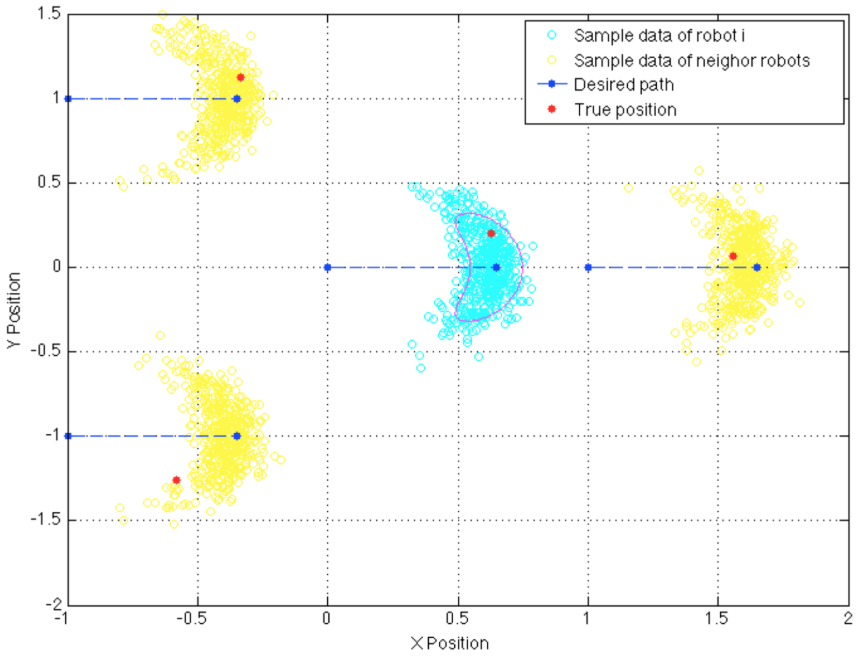


FIGURE 5.2: Localization with only the prediction model (straight-line Motion)

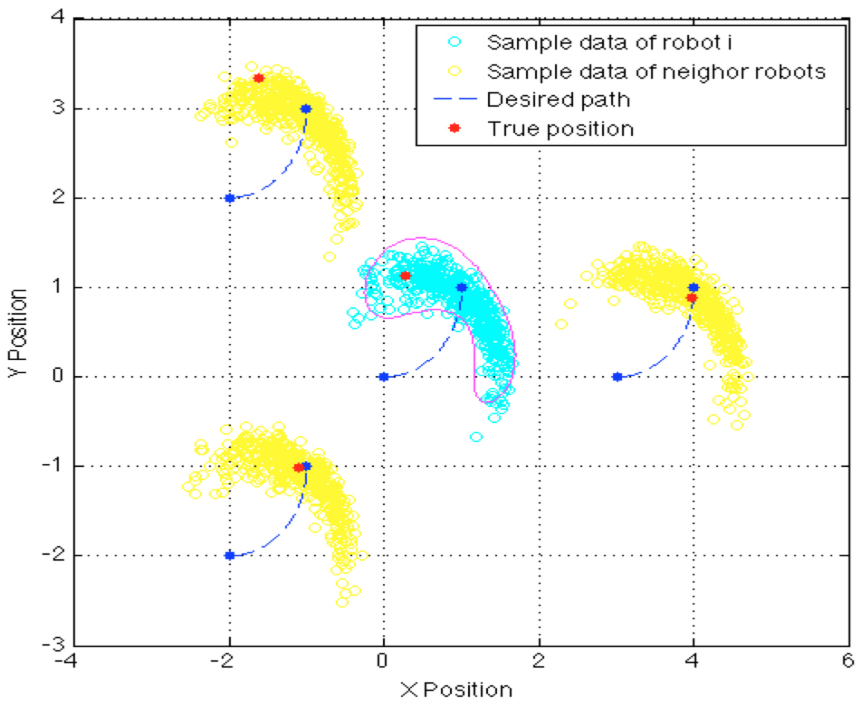


FIGURE 5.3: Localization with only the prediction model (Circular Motion)

It is assumed that robot i can exchange information with its neighbor when they come into its sensing range, which means when a relative measurement is taken of neighbor j relative to robot i, the belief (mean and covariance in this case) that j holds for its current position can be communicated to i so that i can make use of this information in its localization process. In this example, this belief (μ_j, Σ_j) for each neighboring robot j is taken to be the pose prediction calculated from (5.5)(5.6) but in reality this can very well be the posterior from their own localization results. The covariance of the measurement distribution is chosen to be

$$\Sigma_m = \begin{bmatrix} 0.01 & 0.02 & 0.001 \\ 0.02 & 0.25 & 0.015 \\ 0.002 & 0.0025 & 0.15 \end{bmatrix}$$

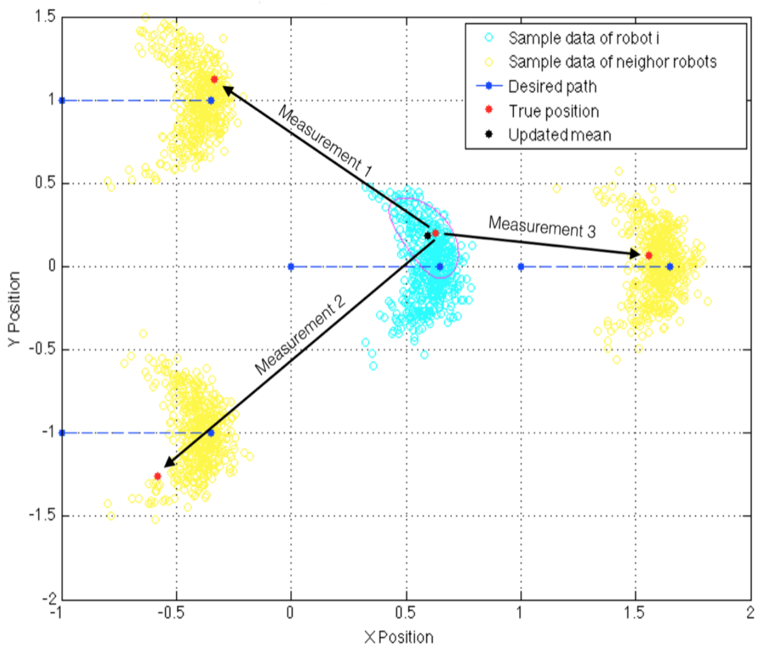


Figure 5.4: Pose update after sensor measurement and fusion (straight-line Motion)

Figure 5.4 and 5.5 shows the updated posterior of robot i calculated from fusing the relative measurements taken from its three neighbors. The result indicates a more accurate position mean (black dot) and a shrinked error "ellipse" representing higher confidence in the estimate. Since this is a distributed localization technique aimed to be implemented on the embedded processor of each individual robot, the procedure is demonstrated only for one robot and the same goes for all other.

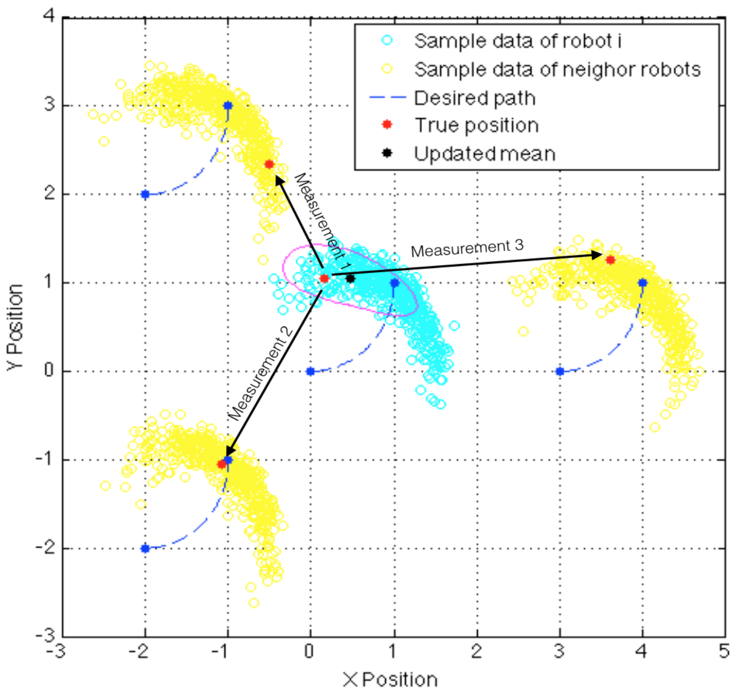


FIGURE 5.5: Pose update after sensor measurement and fusion (circular Motion)

In addition, it is tempting to figure out how the accuracy of the calculated posterior is related to the number of neighbor robot that sensor measurements can be taken from. Figure 5.6 an 5.7 shows a simulated scenario used to study this relationship. In this study, the localization process is implement for robot i first with measurement taken only from neighbor 1, then from neighbor 1 and 2 and so on. Each time the localization process is repeated 60 times and the deviation is defined as

$$deviation = \frac{\|\mathbf{x}_{true} - \mathbf{x}_{estimate}\|_2}{\|\mathbf{x}_{true}\|_2} \tag{5.10}$$

where $\mathbf{x}_{estimate} = \bar{\mu}_i$ is the mean of the updated robot position.

TABLE 5.1: Numerical Results for Correlation Study (Straight-line Motion)

Num. of Measurements	**1**	**2**	**3**	**4**	**5**	**6**
Mean Sampled Deviation	0.3264	0.2823	0.2193	0.1926	0.1985	0.1798
Frobenius norm of Sample Covariance	0.095	0.1554	0.2027	0.2413	0.2721	0.2982

Figure 5.8 and 5.9 shows boxplots of the deviations defined by (5.10) and the mean of sampled deviations along with the Frobenius norm of the average covariance matrices are tabulated in table 5.1 and 5.2. It is observed that as the number of measurements

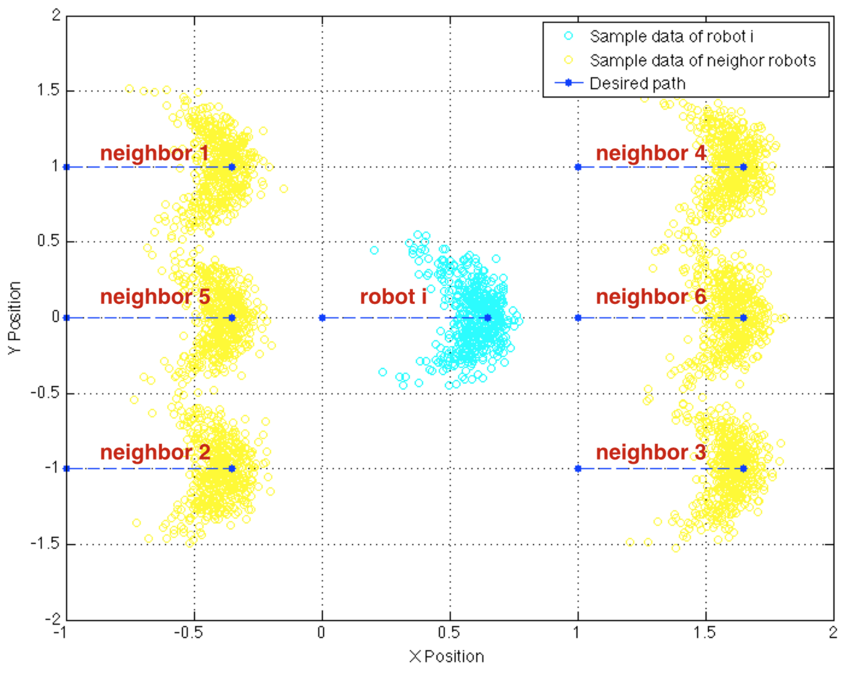


FIGURE 5.6: Simulation setup for the study of relationship between localization accuracy and number of measurements taken (straight-line motion)

TABLE 5.2: Numerical Results for Correlation Study (Circular Motion)

Num. of Measurements	**1**	**2**	**3**	**4**	**5**	**6**
Mean Sampled Deviation	0.2234	0.1585	0.1606	0.1520	0.1697	0.1461
Frobenius norm of Sample Covariance	0.2577	0.4197	0.5457	0.6648	0.7561	0.8371

incorporated in the localization scheme increase, the the accuracy of the resultant pose estimation increase, however this increase in accuracy plateaus after round three measurements. Due to the higher extent of diffusion in circular motion, there is a larger probability that the true position ends up far away from the predicted one which to a certain degree violates the prerequisite for the BCH expansion resulting in higher magnitude of the covariance matrix and fluctuation in the sampled mean deviation. Also the magnitude of the updated covariance increases with the number of measurements used as a result of the additive effect of the uncertainties in the sensor measurement as well as that in the beliefs of neighboring robots. Therefore there is tradeoff between accuracy, confidence and computational cost. This finding on a level tells us that there may exist an optimal number of measurements that can be used to achieve the best results which can be part of the possible future work.

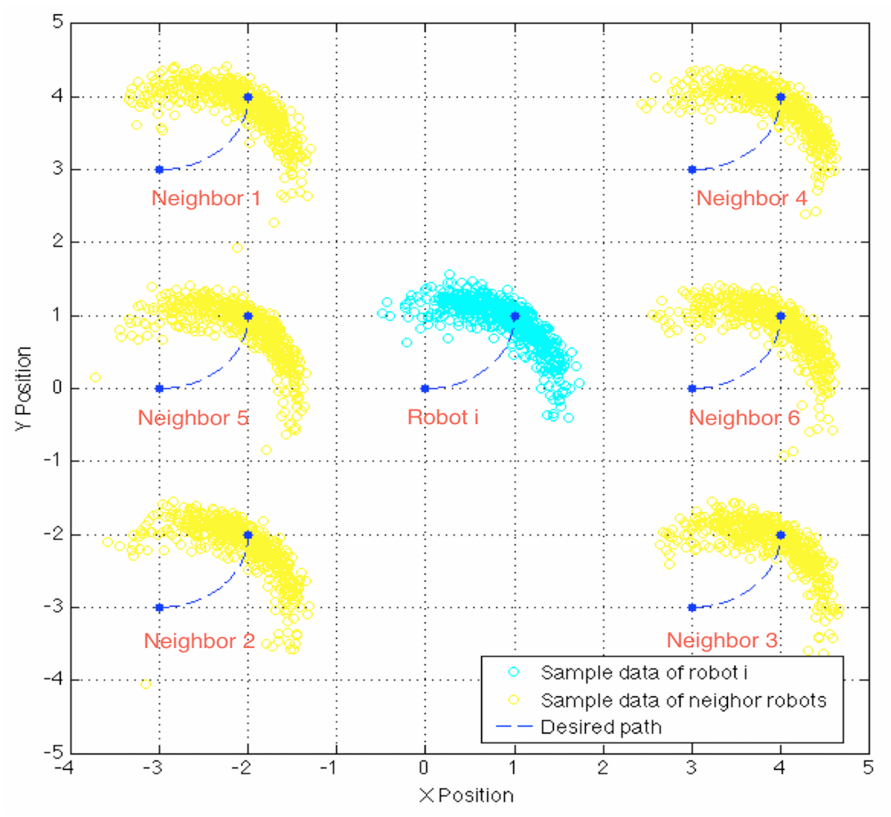


FIGURE 5.7: Simulation setup for the study of relationship between localization accuracy and number of measurements taken (circular motion)

5.2 Simulation on ROS/Gazebo

In comparison to Matlab, ROS/Gazebo provides a more realistic framework for robotic simulation. Figure below briefly illustrates the relationship between ROS and Gazebo

As can be observed from the figure, ROS serves as the interface that connects users to the application frontend. And Gazebo is the simulation backend to ROS that provides models of the physical world as well as the device of interest. Typically ROS can not tell the difference between the real world and the simulated world as long as both provide the necessary information that ROS requires for its calculation. Therefore with a carefully constructed simulation, the user's control code can be migrated with little to no modification to the real robot which establishes the main advantage of this framework. The following subsections are dedicated to present the simulation setup and results of the proposed localization technique.

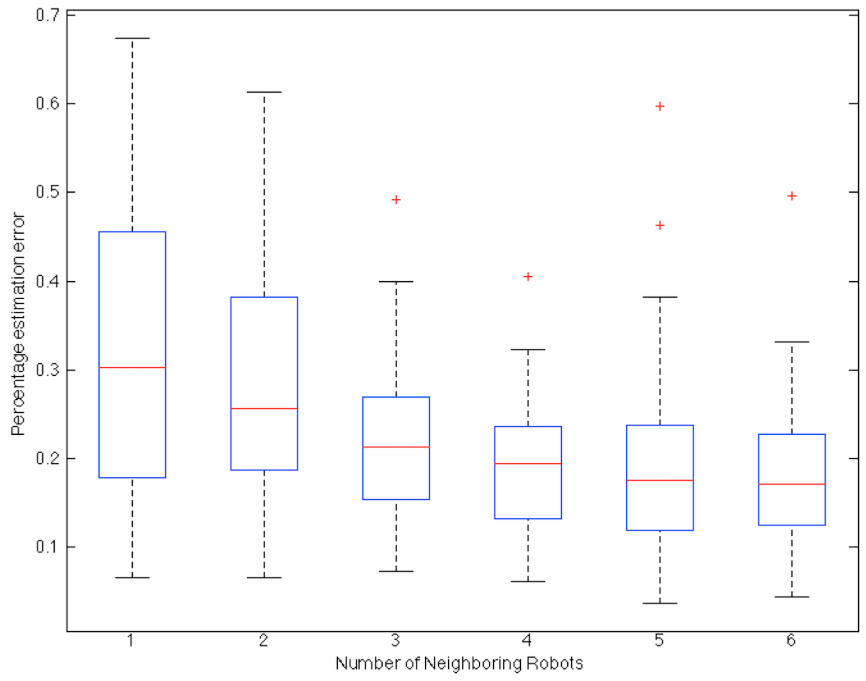


FIGURE 5.8: Boxplot for Correlation study (pictures/straight-line Motion)

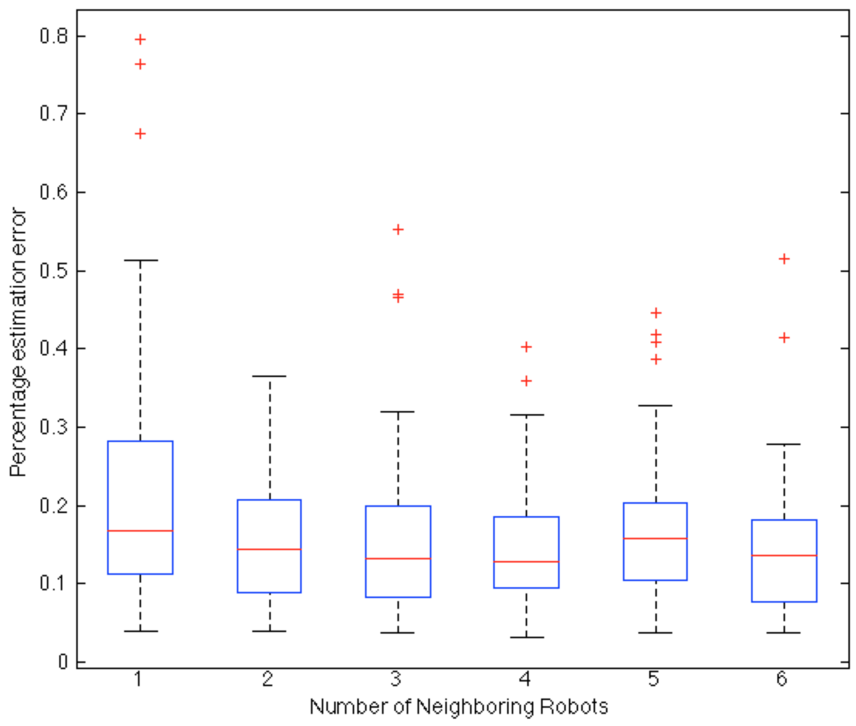


FIGURE 5.9: Boxplot for Correlation study (circular Motion)

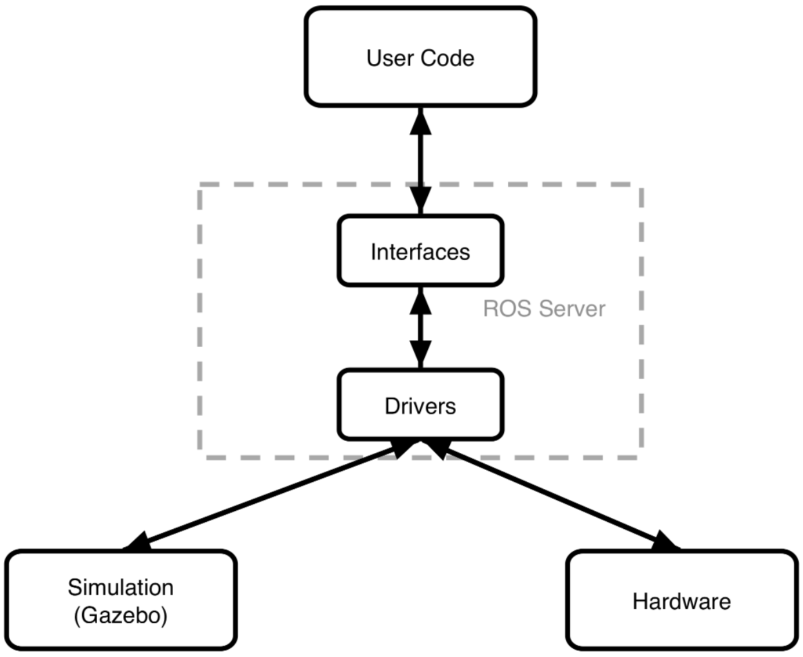
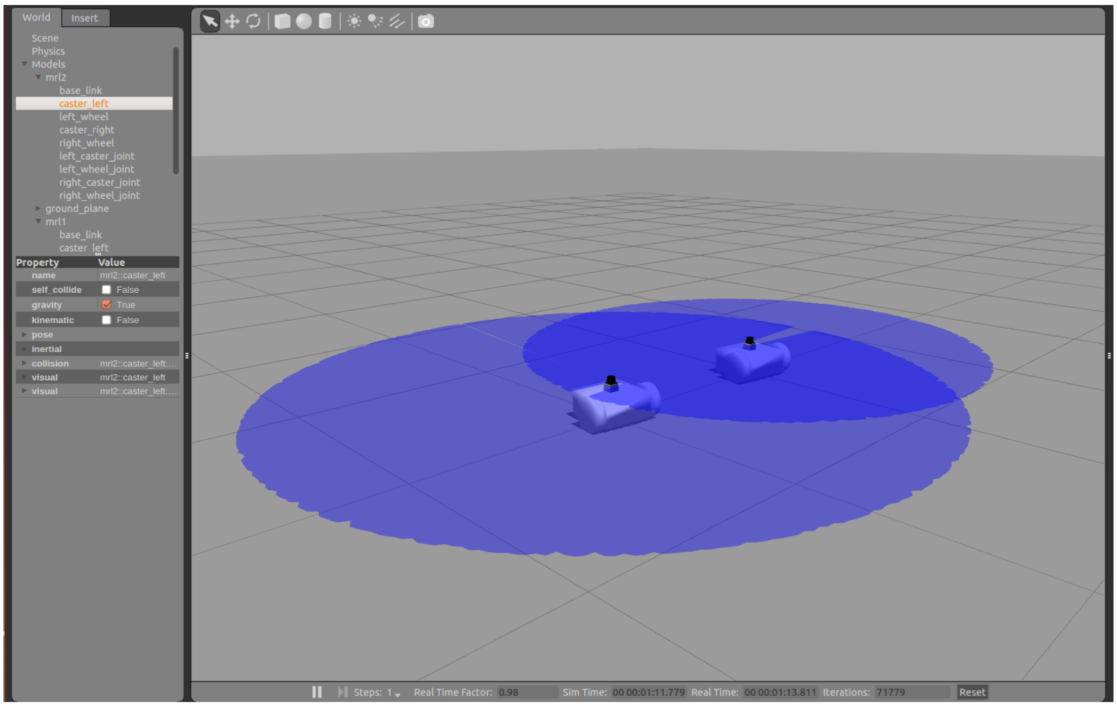


Figure 5.10: ROS Gazebo Relationship

5.2.1 Simulation Setup

The experiment simulates two two-wheel mobile bases each equipped with joint effort controllers, joint velocity encoder and a Hokuyo laser sensor (as shown in figure 5.11 below). The input joint efforts are controlled through a PID controller with $k_p = 100, k_i = 0.01, k_d = 10$ and updated at 15 hz. The Hokuyo laser updates its measurements at 40 hz with a maximum detection range of 1.5 m (0.01 m resolution) and $-\pi$ to π (1 degree resolution). The measurement noise is assumed to be normally distributed with zero mean and variable variances. The variance of the measurement noise will be adjusted to validate the performance of the proposed method (results shown in the next section).

Since the proposed method is fully distributed, the localizer along with the motion planner can be designed so that the same piece of code can be attached to any robot with no further adjustments necessary. A schematic of the simulation system is provide in figure 5.12 below. It can be observed from the schematic that the localizer constantly broadcasts it's estimate of pose and covariance so other robots can use this information alongside the relative measurement to update their own estimates. When the laser senses that another team member is within detection range, it provides its host



Property	Value
name	mrl2::caster_left
self_collide	False
gravity	[✓ True]
kinematic	False
pose	
inertial	
collision	mrl2::caster_left:...
visual	mrl2::caster_left
visual	mrl2::caster_left:...

Steps: 1 Real Time Factor: 0.98 Sim Time: 00 00:01:11.779 Real Time: 00 00:01:13.811 Iterations: 71779 Reset

FIGURE 5.11: Robots In the Simulated World

robot with the necessary measurements (position and orientation of neighbor in the host robot's local frame), in addition the host robot "asks" the sensed neighbor for its current estimate of its pose and covariance. The localizer then uses this information to update the robot's location. Figure 5.13 shows the information flow of the algorithm provided by ROS. *mrl1* and *mrl2* represent the two robots and their embedded controllers, */mrl/robot_state_publisher* is the model for joint encoders that provide the joint velocities. */mrlPos* represents the global feedback system that provides the ground truth for the pose of the robot. */mrl_control* is the interface that supports communication between the controller/localizer and the simulation world.

Earlier in this chapter we showed that equations (5.3)(5.3) can be calculated in close-form for straight-line and circular motion ((5.4)-(5.9)). However in order to perform this simulation continuously, we need a set propagation equations that takes arbitrary input (ω_1, ω_2) and produce the mean and covariance that represents the system's incremental motion under this set of constant inputs for the defined time interval. Suppose system (5.1) travels from t_k to t_{k+1} under inputs (ω_1, ω_2).since the proposed localization technique is of second order, we approximated the matrix exponential in (5.3) to the second order

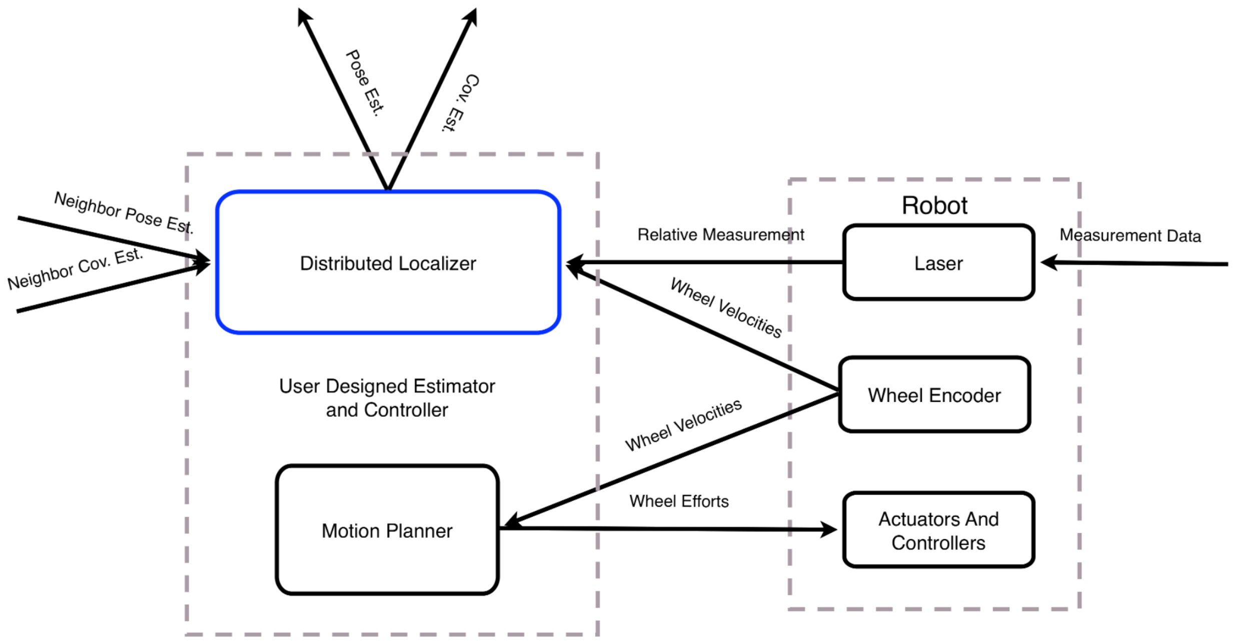


FIGURE 5.12: Algorithm Schematic

$$
\begin{aligned}
\mu(\Delta t) &= \exp(\int_0^{\Delta t} \hat{\boldsymbol{h}}\, \mathrm{d}\tau) \\
&= \exp\left(\begin{bmatrix} 0 & -\frac{r}{2}(\omega_1-\omega_2)\Delta t & \frac{r}{2}(\omega_1+\omega_2)\Delta t \\ \frac{r}{2}(\omega_1-\omega_2)\Delta t & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) \\
&\approx I + \begin{bmatrix} 0 & -\frac{r}{2}(\omega_1-\omega_2)\Delta t & \frac{r}{2}(\omega_1+\omega_2)\Delta t \\ \frac{r}{2}(\omega_1-\omega_2)\Delta t & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
&\quad + \frac{1}{2}\begin{bmatrix} 0 & -\frac{r}{2}(\omega_1-\omega_2)\Delta t & \frac{r}{2}(\omega_1+\omega_2)\Delta t \\ \frac{r}{2}(\omega_1-\omega_2)\Delta t & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}^2
\end{aligned}
\tag{5.11}
$$

With the expression of $\mu(\Delta t)$ given, the integrand of (5.4) can also be written in close-form, let $f(t) = Ad(\mu^{-1}(t))\mathbf{H}\mathbf{H}^T Ad^T(\mu^{-1})(t)$, (5.4) can be approximate by the 3/8 Simpson's rule

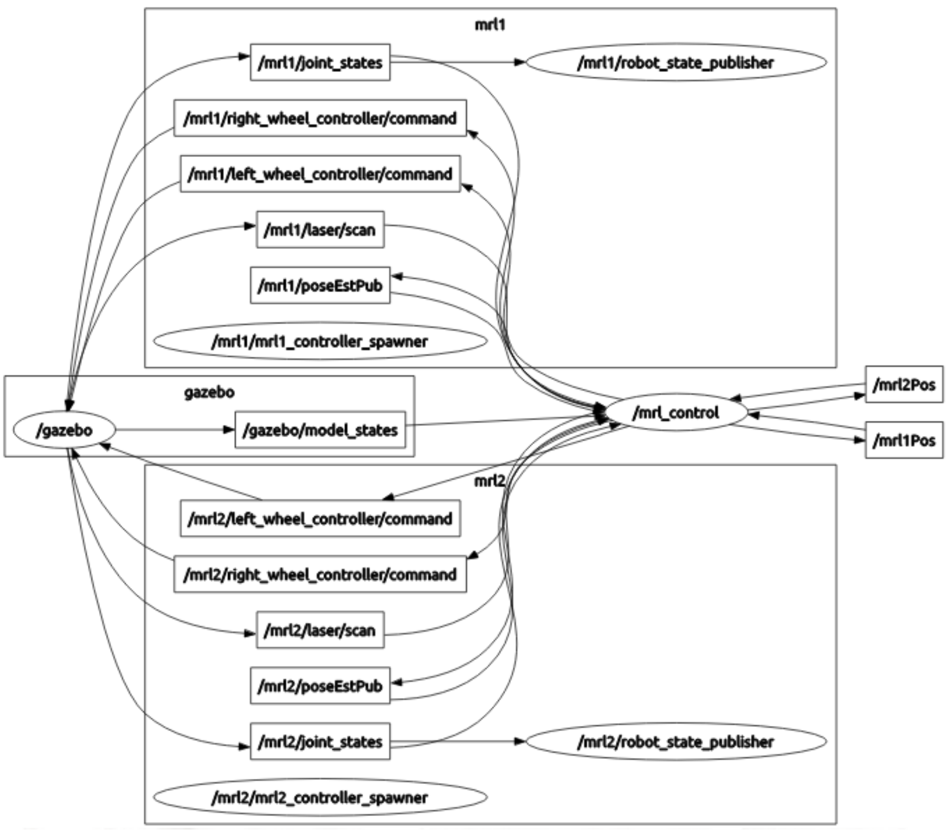

mrl1
/mrl1/joint_states
/mrl1/robot_state_publisher
/mrl1/right_wheel_controller/command
/mrl1/left_wheel_controller/command
/mrl1/laser/scan
/mrl1/poseEstPub
/mrl1/mrl1_controller_spawner
gazebo
/gazebo
/gazebo/model_states
/mrl_control
/mrl2Pos
/mrl1Pos
mrl2
/mrl2/left_wheel_controller/command
/mrl2/right_wheel_controller/command
/mrl2/laser/scan
/mrl2/poseEstPub
/mrl2/joint_states
/mrl2/robot_state_publisher
/mrl2/mrl2_controller_spawner

FIGURE 5.13: Algorithm Information Flow

$$\begin{aligned}\Sigma(\Delta t) &= \int_0^{\Delta t} f(\tau)\,\mathrm{d}\tau \\ &\approx \frac{\Delta t}{8}[f(0)+3f(\frac{\Delta t}{3})+3f(\frac{2\Delta t}{3})+f(\Delta t)]\end{aligned} \tag{5.12}$$

replace the first two equations in (4.13) with (5.11)(5.12), the proposed localization method can then be implemented continuously. Note that since the integral in (5.3) can be evaluated analytically, one can also use the matrix exponential function that comes with the programming language of choice to calculate $\mu(\Delta t)$ for higher accuracy. It is expected that the result will not deviate much from the second order approximation for small Δt.

In addition to simulating the technique presented, since the Distributed EKF method discussed in chapter 2([11]) is very close in nature with the proposed method (both incremental local distributed localization techniques that require Gaussian noise distributions), it is also tested in the same simulation framework serving as a comparison. This comparison method is implement to the extent that it uses the same set of inputs to the localizer as the proposed method in order to pose a fair comparison. A general planar dynamics equation is used as the state propagation equation (2.4) given by

$$\begin{bmatrix}\hat{x}(t_{k+1})\\ \hat{y}(t_{k+1})\\ \hat{\theta}(t_{k+1})\end{bmatrix} = \begin{bmatrix}\hat{x}(t_k)\\ \hat{y}(t)\\ \hat{\theta}(t_k)\end{bmatrix} + \begin{bmatrix}V_m\cos(\theta(t_k))\\ V_m\sin(\theta(t_k))\\ \Omega_m\end{bmatrix}\Delta t \tag{5.13}$$

where (V_m, Ω_m) are the linear and angular velocities of the robot provided by the odometry readings. Since the robot used has only wheel joint encoders, this set of inputs are transformed to the encoder readings by

$$\begin{aligned} V_m &= \frac{\omega_1+\omega_2}{2}r \\ \Omega_m &= \frac{\omega_1-\omega_2}{l}r \end{aligned} \tag{5.14}$$

After linearization, the state evolution matrix in (2.3) becomes

$$\Phi(t_{k+1}, t_k) = \begin{bmatrix}1 & 0 & -V_m\Delta t\sin(\hat{\theta}_k)\\ 0 & 1 & V_m\Delta t\cos(\hat{\theta}_k)\\ 0 & 0 & 1\end{bmatrix} \tag{5.15}$$

and the error matrix becomes

$$G(t_{k+1}, t_k) = \begin{bmatrix}\Delta t\cos(\hat{\theta}_k) & 0\\ \Delta t\sin(\hat{\theta}_k) & 0\\ 0 & \Delta t\end{bmatrix} \tag{5.16}$$

Equations (2.4)-(2.12) can hence be used for the Distributed EKF Localizer. The following subsection presents the simulation results along with a discussion of their significance.

5.2.2 Results And Discussion

Results for the ROS/Gazebo Simulation will be presented in this section. Robot 1 starts at known initial pose $[1, 1, 30°]^T$ and Robot 2 at $[1, 2, 30°]^T$. The robots receive actuator effort command from the motion planner, in this case moving straight at 0.4 m/s for 15 sec then in a circular arc. Two robots continuously take measurements from each other and uses this information to update their estimated pose. The experiment runs for around 90 sec. Figure 5.14 below shows the true trajectory of Robot 1 (thick blue line) and the estimated trajectory from the proposed method (red), distributed EKF (black) and dead reckoning (magenta).

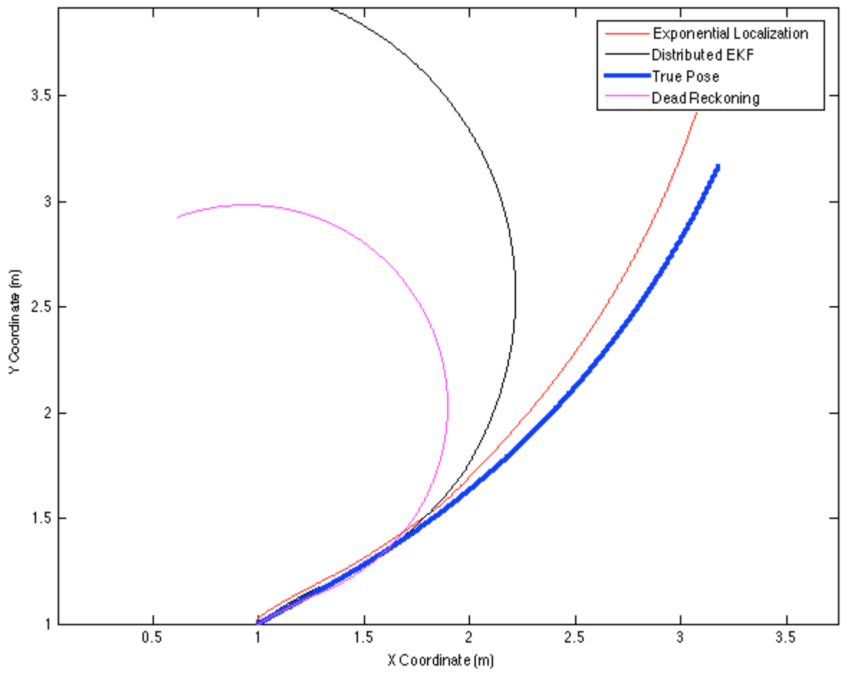


FIGURE 5.14: Localization Comparison Results for Robot 1 (with measurement noise $std = 0.01$)

Defining the error as

$$Error_{est} = \|\vec{x}_{est} - \vec{x}_{true}\|_2 \tag{5.17}$$

with $\vec{x} = [x, y]^T$. Figure 5.15 below illustrates the growth of the localization error with time. We can observe that Exponential Localization method can most effectively resist estimation error followed by the Distributed EKF and then Dead Reckoning. All estimations here are expected to diverge given there's no process that serves to reduce error and errors from process and measurement noise are anticipated to accumulate. The sensor fusion and estimate update processes are there to help suppress the rate of this divergence and the above results show that the proposed localization technique is able to accomplish this task effectively.

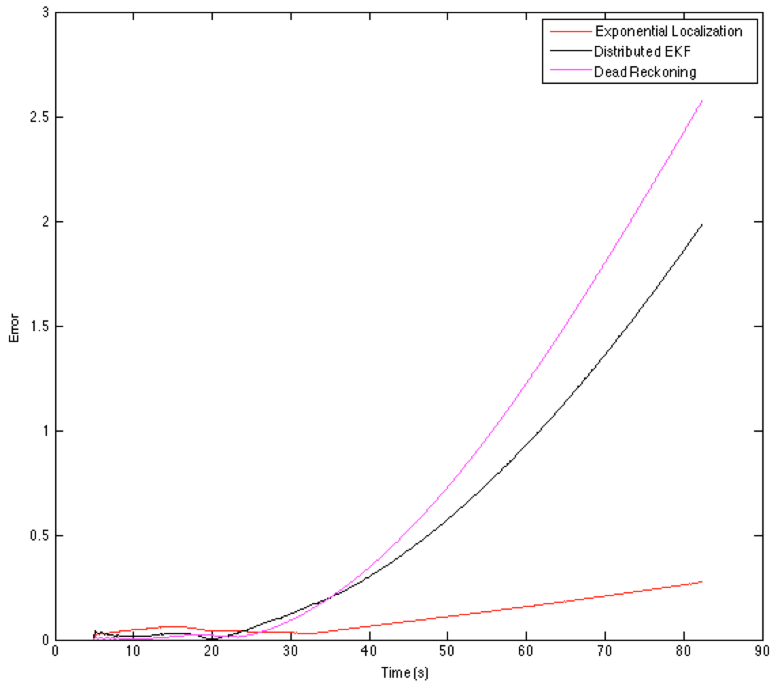


FIGURE 5.15: Localization Error Comparison Results for Robot 1 (with measurement noise $std = 0.01$)

Given the above outcomes, it is also desirable to study the effect of measurement noise on the localization processes. The noise parameters used here are based on the published specs for the Hokuyo Laser. A mean of 0.0 m and standard deviation of 0.01 m will put 99.7% of samples within 0.03 m of the true reading. Figure 5.11 shows a visualization of the laser measurements with the standard deviation (std) equals 0.01. Figure 5.16 below shows the same scenario only std equals 0.05. The edges of the measurement perimeter are blurred reflecting the fluctuation of the measurements under noise.

Same simulation process is repeated under this noise level and results are given by figure 5.17 and figure 5.18. It can be seen that the same trend is preserved with the Exponential Localization method having the lowest divergence rate and then Distributed EKF and Dead Reckoning. However the overall localization error and divergence rate increased under the increased measurement noise.

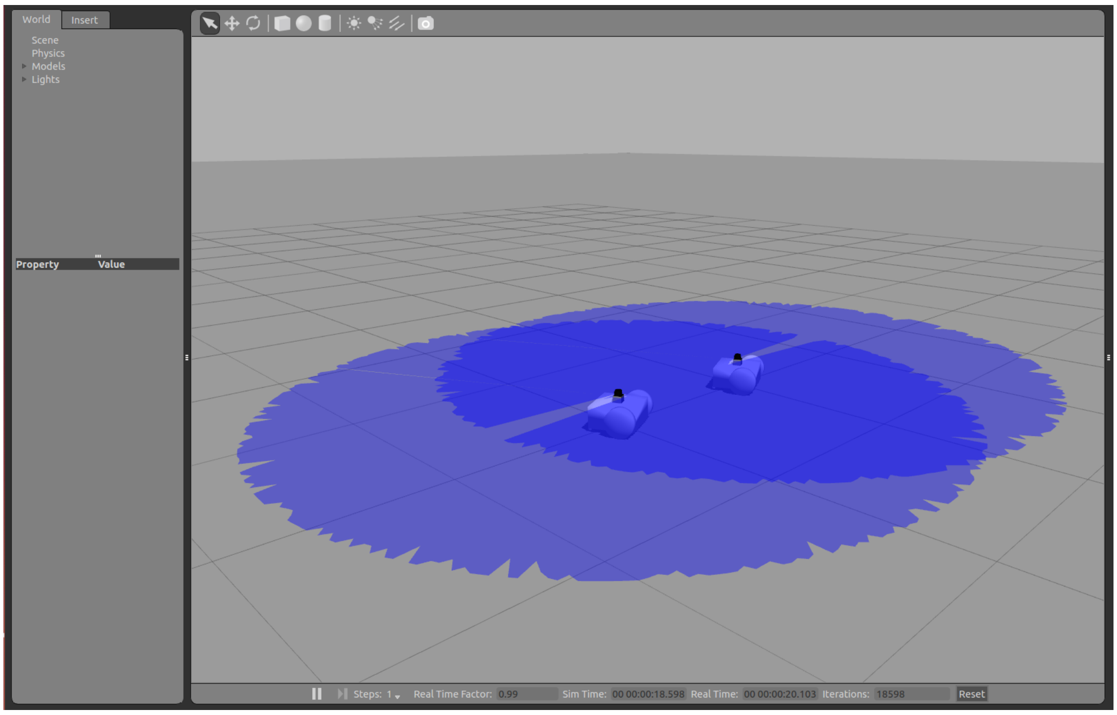


FIGURE 5.16: Laser with Zero Mean and 0.05 Standard Deviation Gaussian Noise

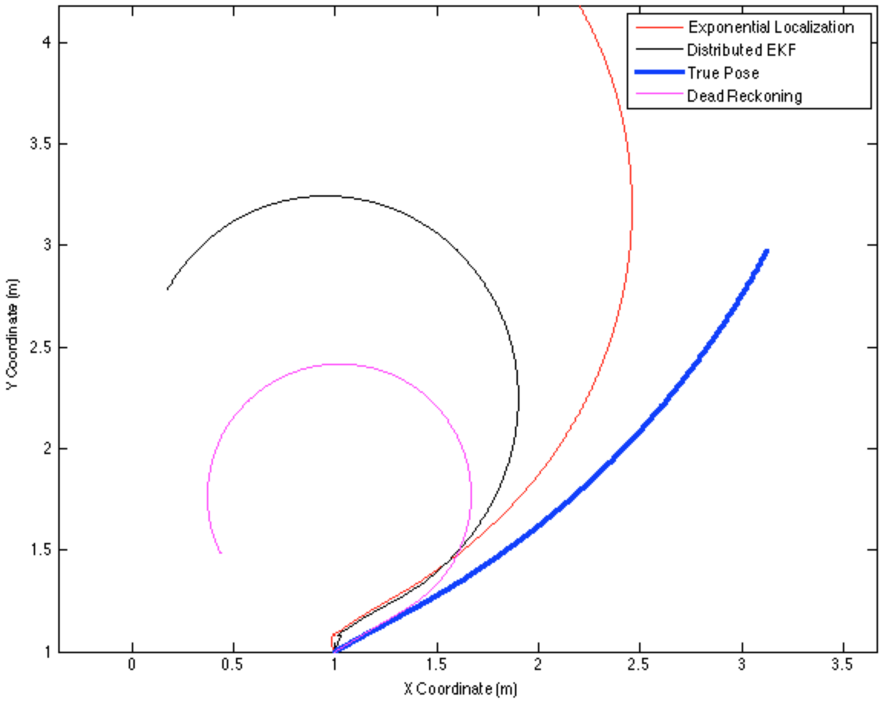


FIGURE 5.17: Localization Comparison Results for Robot 1 (with measurement noise $std = 0.05$)

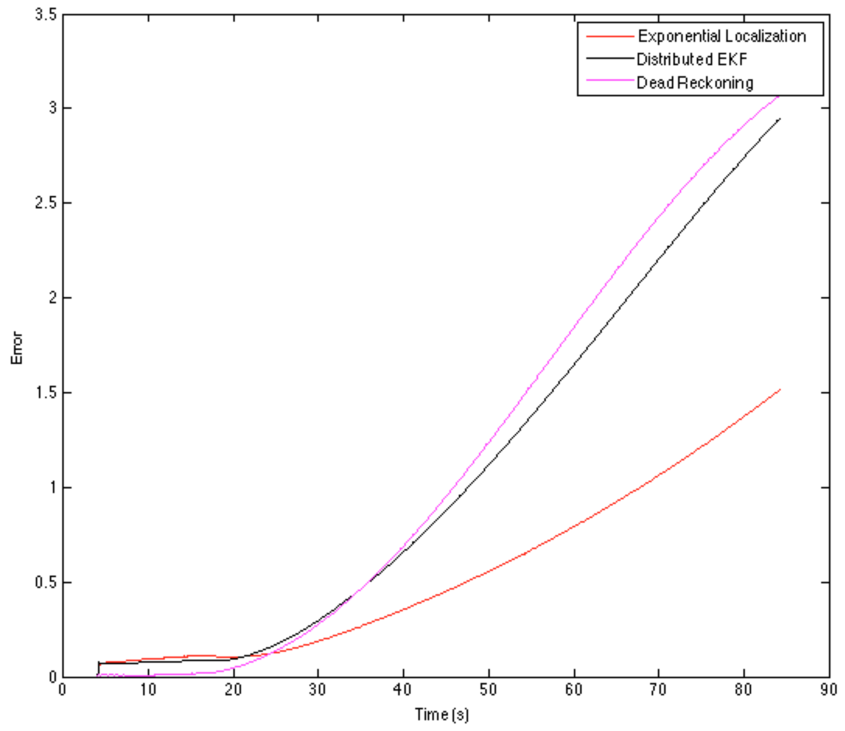
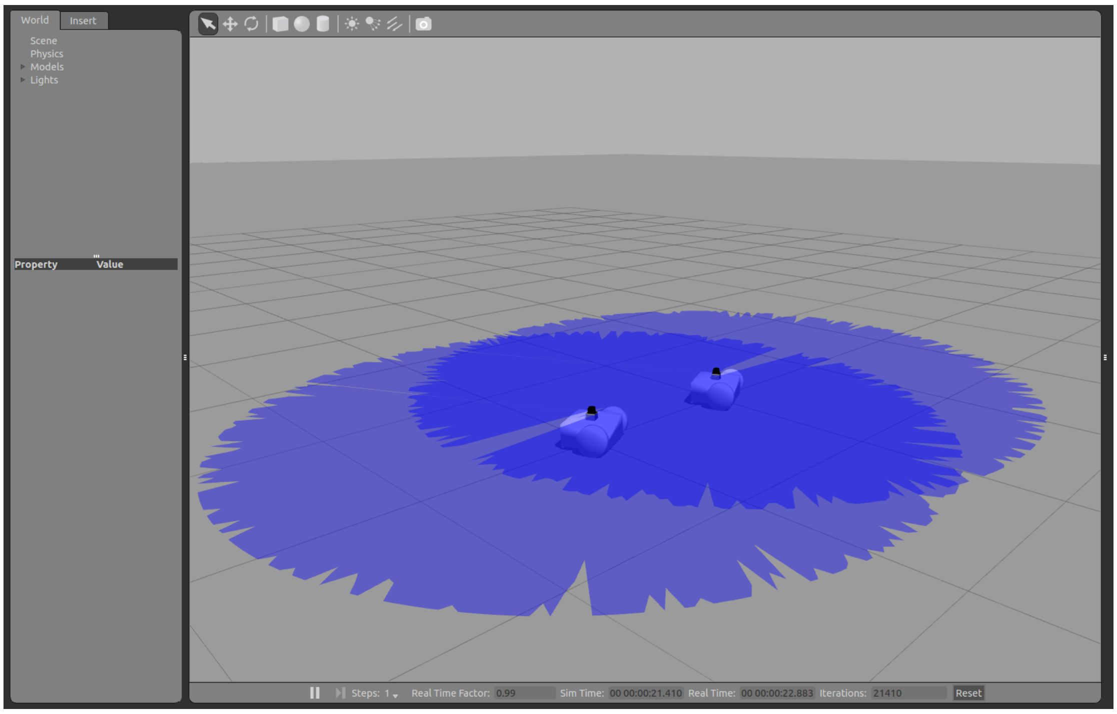


FIGURE 5.18: Localization Error Comparison Results for Robot 1 (with measurement noise $std = 0.05$)

The same process is repeated for the case when measurement standard deviation equals 0.1 and results are shown in figure 5.19, 5.20, 5.21. And similar to the case above, the general trend is agin preserved while the overall divergence for the estimations is accelerated. Dead reckoning (DR) error stayed on relatively the same level for the reason that DR takes only the wheel encoder readings as input and hence gets effected only by process noise and not measurement noise.

The true value of any cooperative localization method lies in that as long as one robot in the team has a means of reducing the localization error (through GPS, better hardware, etc), this information can be shared among all team members and be used to reduce the localization error of each. Figure 5.22 simulates such a scenario, here instead of taking the estimated pose and covariance of the neighboring robot (robot 2) as input to the localizer of robot 1, the position ground truth of robot 2 is passed in simulating the case when only robot 2 has the ability to obtain accurate pose information and robot 1 has to make use of this information to reduce its localization error algorithmically. The simulation is run for around 500 *sec* and results are shown in Figure 5.22 and 5.23. We can see that comparing to the previous cases, the error in this scenario is effectively



Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:00:21.410 Real Time: 00 00:00:22.883 Iterations: 21410

FIGURE 5.19: Laser with Zero Mean and 0.1 Standard Deviation Gaussian Noise

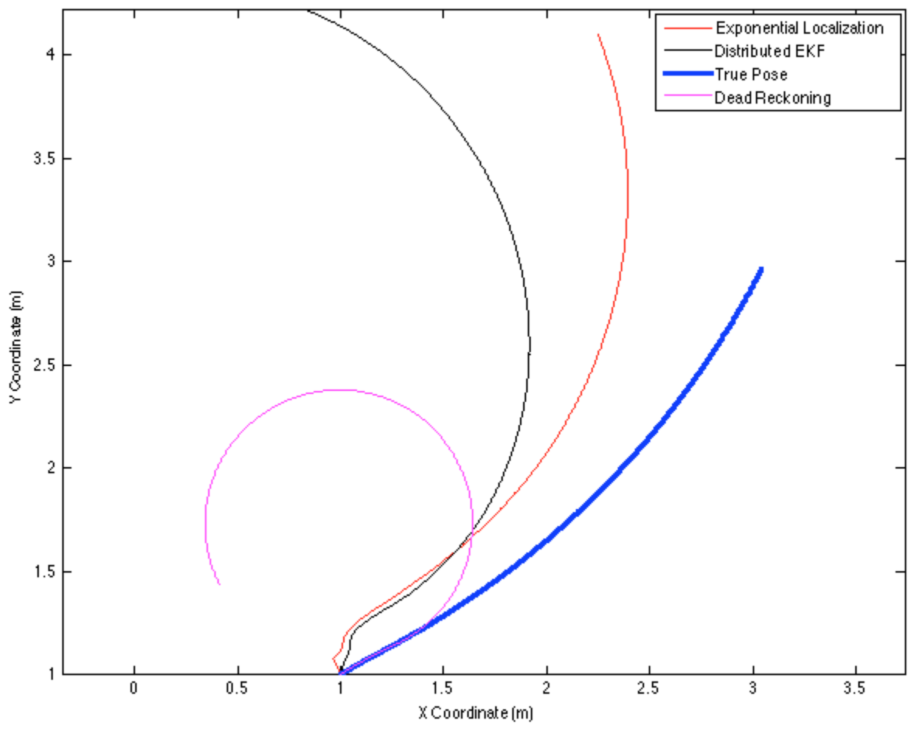


FIGURE 5.20: Localization Comparison Results for Robot 1 (with measurement noise $std = 0.1$)

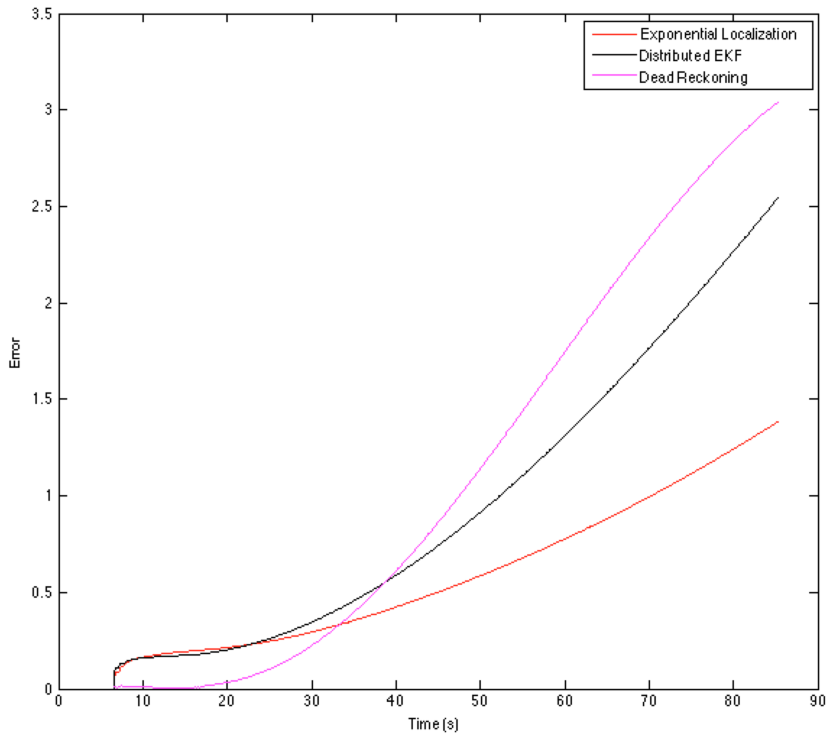


FIGURE 5.21: Localization Error Comparison Results for Robot 1 (with measurement noise $std = 0.1$)

bounded. The estimated trajectory resembles the true trajectory to a reasonable degree. Discrepancies exist due to sensor calibration (for example choice of laser scan to represent relative measurement) as well as parameter tuning (choice of initial state covariance, process noise and measurement noise covariance). This error is expect to decrease with further fine tuning of the system.

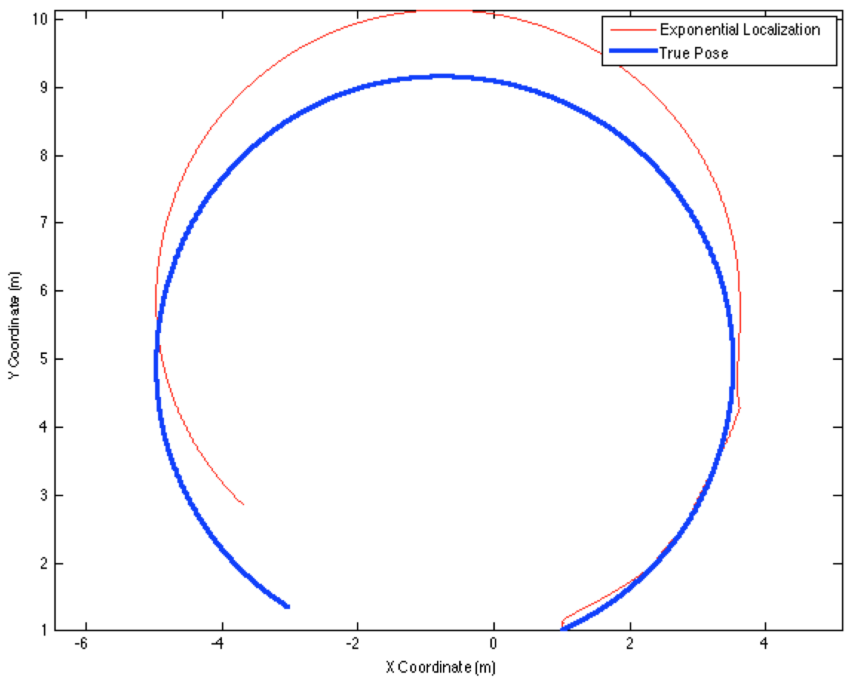


FIGURE 5.22: Localization Results for Robot 1 with Ground Truth Neighbor Position (measurement noise *std* = 0.05)

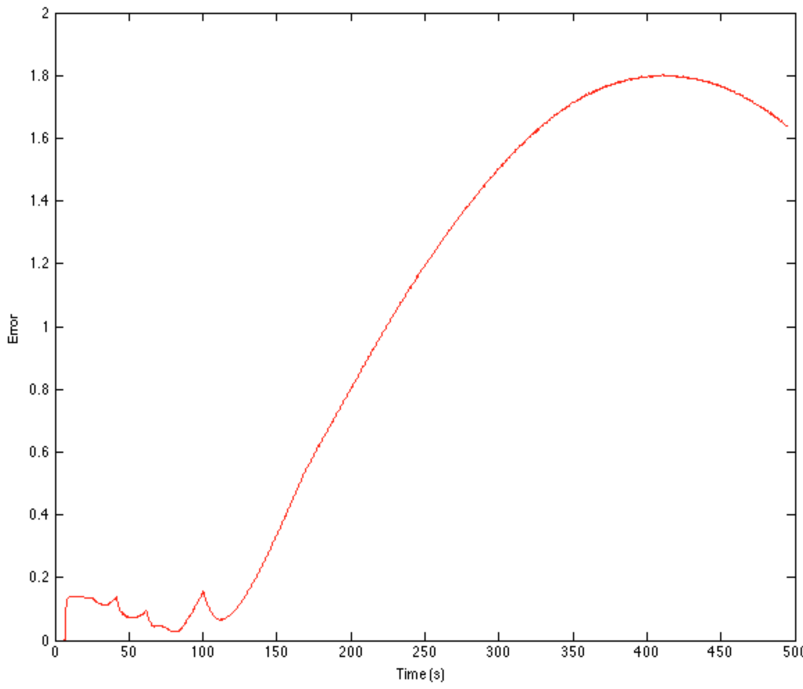


FIGURE 5.23: Localization Error for Robot 1 with Ground Truth Neighbor Position (measurement noise *std* = 0.05)

Chapter 6

Conclusion

This thesis proposed a distributed cooperative localization technique that can incorporate multiple sensor measurements to achieve higher estimation accuracy. Robots in a team can take measurements and exchange information among each other to update their knowledge of the current position. Simulation is used to validate the performance of the approach in both Matlab and ROS/Gazebo. Results from the Matlab simulation show a good localization accuracy of the presented approach and an increase in this accuracy with the number of measurements taken. The distributed EKF method is also simulated as a comparison in ROS/Gazebo, and outcomes show that the proposed exponential localization scheme yields superior results to the distributed EKF method under varied measurement noise levels. The proposed technique is distributed in that each robot can perform this localization process without the help of a centralized processor, and is scalable for the computation time does not increase as the robot team enlarges and increases only linearly with the number of measurements taken. The generality of this scheme lies in the fact that uncertainties in the belief of the current robot, all neighboring robots and sensor measurements have all been considered which yields a more realistic result. Unlike sampling-based approaches, the proposed approach provides closed-form expressions which significantly increases computational efficiency. Most existing cooperative localization schemes possess a subset of the the above attributes but rarely all. Lastly, this technique is of second order in its estimation of an updated posterior which is expected to be more accurate and reliable than first order methods.

The limitation of this method exists at its dependency on Gaussian noises which is shown applicable in many cases such as the success of Extend Kalman Filter but is not the most accurate model to represent random noises. Moreover, this is a local technique in that it depends on known initial poses and does not recover from localization failures (defined by [7]). At its current state, this approach does not possess the ability to serve as the sole localization scheme to localize a team of robots in that as errors accumulate in the beliefs of neighboring robots, erroneous information will be given to the current robot that leads to localization failures. However, this technique is local and prone to error accumulation only when none of the member robots have a reasonable estimate of their positions, as long as one robot possesses a good knowledge of its current pose (via more accurate sensors or sophisticated but computationally expensive algorithms) then this information can be used to drastically reduce the uncertainty of the entire team which introduces a level of robustness to this technique and can also significantly reduce hardware and computational cost of the team. This scenario is also simulated and results show that the proposed approach can effectively reduce the localization error with the help of "advanced neighbors". Table 6.1 shows a comparison of the proposed method with the state of the arts presented in chapter 2.

Table 6.1: Final Comparison

	Distibuted EKF	**Multi-Robot MCL**	**Exponential Localization**
Restrictions on ErrorDistribution	Requires Gaussian process and measurement error	Nonparametric particle representation of posterior belief, no assumptions on noise distribution	Gaussian
Global Localization	No	Yes	No
State Recovery	No	Possible given a well designed resampling process	No
Localization Accuarcy	Accurate when error is small	Depends on the number of particles used	Accurate within an error range
Computational Cost	Small due to the closed form propagation and update equations	Depends on the number of particles. Can increase dramatically with the dimension of the state space	Small due to closed form equations
Ease of Implementation	Simple	Can be involved	Simple
Robustness	Prone to error due to linearization	Quite resistant to errors given multiple beliefs are maintained simultaneously	Less susceptible to errors given the well conformity to the motion model
Process Multiple Detections	No	No	Yes
Complexity Relative To Team Size	Fully distributed. Complexity independent of team size	Complexity independent of team size	Linear to the Number of measurements processed

As mention in chapter 5, it will desirable to find the optimal number of measurements to fuse that would yield the best results in terms of localization accuracy and computation time. The accuracy of the exponential localization method is expect to see a great increase compared to results shown previously if the algorithm parameters (initial pose covariance, process and measurement noise covariances, etc) can be find tuned. Establishing a systematic way of tuning these parameters can be a topic of its own. It is also incredibly beneficial if the proposed method can be combined with sampling based approaches for their global localization and state recovery abilities. Lastly, experiments on hardware are required to fully establish the advantage of the proposed scheme. Overall this thesis has provided an alternative distributed cooperative localization technique in the domain of *Lie Group* and *Exponential Coordinates* and has validated in simulation the potential of this technique as the next state of the art.

Appendix A

Calculation of Second Order Gaussian Convolution

This derivation is based on [25]. Let $f_1(g) = f(g; \mu_1, \Sigma_1) = \rho_1(\mu_1^{-1} g)$ and $f_2(g) = f(g; \mu_2, \Sigma_2) = \rho_2(\mu_2^{-1} g)$ with ρ_1, ρ_2 Gaussians centered at the identity. The convolution of two Gaussians is given by

$$\begin{aligned} f_{1*2}(g; \mu_{1*2}, \Sigma_{1*2}) &= (f_1 * f_2)(g) \\ &= \int_G \rho_1(\mu_1^{-1} h) \rho_2(\mu_2^{-1} h^{-1} g) \, \mathrm{d}h \end{aligned} \tag{A.1}$$

Following Lemma 1 in [25], the resulting mean is

$$\mu_{1*2} = \mu_1 \mu_2 \tag{A.2}$$

Lemma 1 in [25] also proves that for ρ_1, ρ_2 centered at the identity

$$\begin{aligned} &\int_G \log(g)(\rho_1 * \rho_2)(g) \, \mathrm{d}g \\ &= \int_G \int_G \log(g) \rho_1(h) \rho_2(h^{-1} g) \, \mathrm{d}g \, \mathrm{d}h = 0 \end{aligned} \tag{A.3}$$

Let $c_1 = \mu_2 h$, $c_2 = \mu_1^{-1}\mu_2^{-1}c_1$, $c_3 = \mu_1^{-1}\mu_2^{-1}g$ the definition of mean described by (3.20) gives

$$\begin{aligned}
&\int_G \log(\mu_{1*2}^{-1}g)(f_1 * f_2)(g)\,\mathrm{d}g \\
&= \int_G\int_G \log(\mu_{1*2}^{-1}g)\rho_1(\mu_1^{-1}h)\rho_2(\mu_2^{-1}h^{-1}g)\,\mathrm{d}g\,\mathrm{d}h \\
&= \int_G\int_G \log(\mu_{1*2}^{-1}g)\rho_1(\mu_1^{-1}\mu_2^{-1}c_1)\rho_2(c_1^{-1}g)\,\mathrm{d}g\,\mathrm{d}c_1 \\
&= \int_G\int_G \log(\mu_{1*2}^{-1}g)\rho_1(c_2)\rho_2(c_2^{-1}\mu_1^{-1}\mu_2^{-1}g)\,\mathrm{d}g\,\mathrm{d}c_2 \\
&= \int_G\int_G \log(\mu_{1*2}^{-1}\mu_2\mu_1 c_3)\rho_1(c_2)\rho_2(c_2^{-1}c_3)\,\mathrm{d}c_3\,\mathrm{d}c_2
\end{aligned} \tag{A.4}$$

If we define $k = \mu_1^{-1}h, q = (\mu_1\mu_2)^{-1}g, q' = \mu_2^{-1}k^{-1}\mu_2 q$, Equation (21) of [25] gives the covariance of the convolution in the form

$$\Sigma_{1*2} = \int_G\int_G [\log(\mu_2^{-1}k\mu_2 q')^\vee][\log(\mu_2^{-1}k\mu_2 q')^\vee]^T \rho_1(k)\rho_2(q')\,\mathrm{d}k\,\mathrm{d}q' \tag{A.5}$$

Further let $X = \log(\mu_2^{-1}k\mu_2), Y = \log(q^{'})$ (and therefore $x = Ad(\mu_2^{-1})\log(k)^\vee, y = \log(q^{'})^\vee$), equation (A.5) becomes

$$\Sigma_{1*2} = \int_G\int_G [\log(e^X e^Y)^\vee][\log(e^X e^Y)^\vee]^T \rho_1(k)\rho_2(q')\,\mathrm{d}k\,\mathrm{d}q' \tag{A.6}$$

Expand $\log(e^X e^Y)^\vee$ using the BCH equation to the second order and retain the even terms (since odd terms integrate to zero) gives

$$\begin{aligned}\Sigma_{1*2} &= \int_G\int_G (xx^T + yy^T)\rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'} \\ &+ \frac{1}{4}\int_G\int_G [ad(X)yy^T ad(X)^T]\rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'} \\ &+ \frac{1}{12}\int_G\int_G [ad(X)ad(X)yy^T + (ad(X)ad(X)yy^T)^T]... \\ &\qquad\qquad \rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'} \\ &+ \frac{1}{12}\int_G\int_G [ad(Y)ad(Y)xx^T + (ad(Y)ad(Y)xx^T)^T]... \\ &\qquad\qquad \rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'}\end{aligned} \tag{A.7}$$

To Calculate The First Double Integral Term in (A.7)

Given $x = Ad(\mu_2^{-1})\log(k)^\vee$ and $y = \log(q^{'})^\vee$

$$\begin{aligned}&\int_G\int_G (xx^T + yy^T)\rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'} \\ &= \int_G xx^T\rho_1(k)\,\mathrm{d}k + \int_G yy^T\rho_2(q^{'})\,\mathrm{d}q^{'} \\ &= Ad(\mu_2^{-1})(\int_G [\log(k)^\vee][\log(k)^\vee]^T\rho_1(k)\,\mathrm{d}k)Ad(\mu_2^{-1})^T \\ &\quad + \int_G [\log(q^{'})^\vee][\log(q^{'})^\vee]^T\rho_2(q^{'})\,\mathrm{d}q^{'} \\ &= Ad(\mu_2^{-1})\Sigma_1 Ad(\mu_2^{-1})^T + \Sigma_2\end{aligned} \tag{A.8}$$

And we define $A = Ad(\mu_2^{-1})\Sigma_1 Ad(\mu_2^{-1})^T$, $B = \Sigma_2$

To Calculate The Second Double Integral Term in (31)

Given $ad(X) = ad(\sum_{i=1}^{d} x_i E_i) = \sum_{i=1}^{d} x_i ad(E_i)$, where E_i are the basis elements for SE(3), and the fact that

$$x_i x_j = e_i^T Ad(\mu_2^{-1})[\log(k)^\vee][\log(k)^\vee]^T Ad(\mu_2^{-1})^T e_j \tag{A.9}$$

then

$$\begin{aligned}&\int_G x_i x_j \rho_1(k)\,\mathrm{d}k\\&= e_i^T[\int_G Ad(\mu_2^{-1})[\log(k)^\vee][\log(k)^\vee]^T Ad(\mu_2^{-1})^T\,\mathrm{d}k]e_j\\&= A_{ij}\end{aligned} \tag{A.10}$$

The second double integral in (A.7) is

$$\begin{aligned}&\int_G\int_G [ad(X)yy^T ad(X)^T]\rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'}\\&= \int_G ad(X)Bad(X)^T\rho_1(k)\,\mathrm{d}k\\&= \int_G [\sum_{i=1}^{d} x_i ad(E_i)]B[\sum_{j=1}^{d} x_j ad(E_j)]^T\rho_1(k)\,\mathrm{d}k\\&= \sum_{i,j=1}^{d} ad(E_i)Bad(E_j)^T\int_G x_i x_j\rho_1(k)\,\mathrm{d}k\\&= \sum_{i,j=1}^{d} ad(E_i)Bad(E_j)^T A_{ij}\end{aligned} \tag{A.11}$$

<ins>To Calculate The Third, Fourth Double Integral Term in (A.7)</ins>

It is defined that $y = \log(q^{'})$, therefore $y_i y_j = e_i^T[\log(q^{'})][\log(q^{'})]^T e_j$ and

$$\int_G y_i y_j \rho_1(q^{'})\,\mathrm{d}q^{'} = B_{ij} \tag{A.12}$$

With the above conclusion, we can perform the following integration

$$\begin{aligned}
&\int_G\int_G ad(X)ad(X)yy^T\rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'} \\
&= \int_G ad(X)ad(X)\rho_1(k)\,\mathrm{d}k\int_G yy^T\rho_2(q^{'})\,\mathrm{d}q^{'} \\
&= [\sum_{i,j=1}^{d} ad(E_i)ad(E_j)A_{ij}]B
\end{aligned} \tag{A.13}$$

Applying the same reasoning, the third integration term in (A.7) results in

$$\begin{aligned}
&\frac{1}{12}\int_G\int_G [ad(X)ad(X)yy^T + (ad(X)ad(X)yy^T)^T]\rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'} \\
&= \frac{1}{12}\left\{[\sum_{i,j=1}^{d} ad(E_i)ad(E_j)A_{ij}]B + B^T[\sum_{i,j=1}^{d} ad(E_i)ad(E_j)A_{ij}]^T\right\}
\end{aligned} \tag{A.14}$$

and the forth integration term results in

$$\begin{aligned}
&\frac{1}{12}\int_G\int_G [ad(Y)ad(Y)xx^T + (ad(Y)ad(Y)xx^T)^T]\rho_1(k)\rho_2(q^{'})\,\mathrm{d}k\,\mathrm{d}q^{'} \\
&= \frac{1}{12}\left\{[\sum_{i,j=1}^{d} ad(E_i)ad(E_j)B_{ij}]A + A^T[\sum_{i,j=1}^{d} ad(E_i)ad(E_i)B_{ij}]^T\right\}
\end{aligned} \tag{A.15}$$

Combining the the above terms yield equations (3.22)-(3.28)

Bibliography

[1] Mondada F. Pettinaro G.C. Guignard A. Kwee I. Floreano D. Swarm-bot: a new distributed robotic concept. *Autonomous Robots*, 2004.

[2] McLurkin J. Smith J. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. *Distributed Autonomous Robotic Systems*, 2007.

[3] Rothermich J. Ecemis M. Gaudiano P. Distributed localization and mapping with a robotic swarm distributed localization and mapping with a robotic swarm distributed localization and mapping with a robotic swarm. In *SAB'04 Proceedings of the 2004 international conference on Swarm Robotics*, pages 58–69, 2004.

[4] Brambilla M. Ferrante E. Birattari M. Dorigo M. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intell*, 2013.

[5] Miner D. Swarm robotics algorithms: A survey. Technical report, MAPLE Lab, University of Maryland, 2007.

[6] Fox D. Burgard W. Kruppa H. Thrun S. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):3250344, 2000.

[7] Thrun S. Burgard W. Fox D. *Probabilistic Robotics.* The MIT Press, 2006.

[8] Long A. Wolfe K. Mashner M. Chirikjian G. The banana distribution is gaussian: A localization study with exponential coordinates. In *Robotics: Science and Systems*, 2012.

[9] Chirikjian G. *Stochastic Models, Information Theory, and Lie Groups, Volume 2.* Springer Science+Business Media, 2012.

[10] Barooah P. Russell J. Hespanha J. Approximate distributed kalman filtering for cooperative multi-agent localization. In *Distributed Computing in Sensor Systems*, volume 6131, pages 102–115, 2010.

[11] Roumeliotis S. Bekey G. Distributed multirobot localization. *IEEE Transactions on Robotics and Automation*, 18(5):781–795, 2002.

[12] Mourikis A.I. Roumeliotis S.I. Performance analysis of multirobot cooperative localization. *IEEE Transactions on Robotics*, 22(4):666–681, 2006.

[13] Caglioti V. Citterio A. Fossati A. Cooperative, distributed localization in multi-robot systems: a minimum-entropy approach. In *Distributed Intelligent Systems: Collective Intelligence and Its Applications*, pages 20–30, 2006.

[14] Liu J. Yuan K. Zou W. Yang Q. Monte carlo multi-robot localization based on grid cells and characteristic particles. In *nternational Conference on Advanced Intelligent Mechatronics*, pages 510–515, 2005.

[15] Marjovi A. Marques L. Penders J. Guardians robot swarm exploration and firefighter assistance. In *IROS*, 2009.

[16] Howard A. Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research*, 2006.

[17] Carlone L. Kaouk M. Du J. Bona B. Indri M. Simultaneous localization and mapping using rao-blackwellized particle filters in multi robot systems. *J Intell Robot Syst*, 63(283-307), 2011.

[18] Roumeliotis S.I. Bekey G.A. Collective localization: A distributed kalman filter approach to localization of groups of mobile robots. In *Preceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[19] Mourikis A.I. Roumeliotis S.I. Performance analysis of multirobot cooperative localization. *IEEE Transactions on Robotics*, 2006.

[20] Koller D. and Fratkina R. Using leaning for approximation in stochastic processes. In *Proc. of the Internation Conference on Machine Leaning*, 1998.

[21] Moore A.W. Schneider J. and Deng K. Efficient locally weighted polynomial regression predictions. In *Proc. of the International Conference on Machine Learning (ICML)*, 1997.

[22] Omohundro S.M. Efficient algorithms with neural network behavior. *Journal of Complex Systems*, 1987.

[23] Omohundro S.M. Bumptrees for efficient function,constraint,and classification learning. *Advances in Neural Information Processing Systems 3*, 1991.

[24] Thrun S. Langford J. Fox D. Monte carlo hidden markov models: Learning non-parametric models of partially observable stochastic processes. In *Proc. of the International Conference on Machine Learning (ICML)*, 1999.

[25] Wang Y. Chirikjian G. Nonparametric second-order theory of error propagation on motion groups. *The International Journal of Robotics Research*, 27(11-12):1258–1273, 2008.

[26] Higham H. An algorithmic introduction to numerical simulation of stochastic differential equations. In *SIAM review*, pages 525–546, 2001.

Biographical Statement

Xiao Li was born April 11,1989 in Yongzhou, China. He did his undergraduate work at the University of British Columbia, BC, Canada. He received his Bachelor of Engineering in Mechanical Engineering with distinction in 2012. After that he started his graduate work at Johns Hopkins University, Baltimore in the field of robotics and obtained his Master of Science in Engineering in Robotics in 2014.