MULTI-ROBOT SLAM USING PARTICLE FILTER

by

Raj U. Gupta

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2019

ABSTRACT

RAJ U. GUPTA. Multi-Robot SLAM using particle filter. (Under the direction of DR. JAMES M. CONRAD)

Simultaneous Localization and Mapping (SLAM) is the conventional chicken and egg problem where a robot has to map the environment as well as localize itself in the newly created map simultaneously. One of the most used approaches to solving this problem is probabilistic robotics. Some of the most common SLAM solutions using probabilistic methods are Extended Kalman Filter, Sparse Extended Information Filter, and Particle Filter. Since single-robot SLAM is mostly a solved problem, researchers are focusing on multi-robot SLAM to improve the efficiency and speed of map exploration. Multi-robot SLAM can be used in tasks where collaboration can improve performance and create a more accurate map. Applications of multi-robot SLAM includes fire fighting in urban and forest areas, rescue and cleaning operations and underwater and space exploration. Most of the published multi-robot SLAM articles use robots equipped with a sensor to detect range and bearing. Range-only SLAM faces issues because it lacks bearing knowledge, which makes it difficult for the robot to create a transformation matrix needed for the robot to merge maps obtained from other robots. This research explores the problem of multi-robot SLAM using range sensors. A Received Signal Strength Indicator (RSSI) sensor can be used to detect the landmark and another robot. RSSI is a measure of power received from radio signals. Radio beacons can be used as landmarks which helps to remove the data association problem for landmarks as each radio beacons have unique media access control (MAC) address in the network. This thesis proposed an approach to multi-robot range-only SLAM using full particle method.

ACKNOWLEDGEMENTS

I would like to thank Dr. James M. Conrad for all his help and guidance during my masters.

I would also like to thank Dr. Robert Cox for all his help during my graduation and thesis work

I would also like to thank Dr. Srinivas Akella for all his support and offering individual study under him which helped me in getting knowledge of SLAM and explore my interest.

I also want to thank Dr. Samuel Shue, David Grabowsky and Sunny Arokia who helped me to set my research standard and also helped me with technical problems.

Finally, I would thank my parents and family for their emotional and monetary support all over during my graduation. This would have been impossible without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
JCBB	Join Compatibility Branch and Bound
MAC	Media Access Control
NN	Nearest neighbour
PF	Particle Filter
RBPF	Rao-Blackwelized Particle filter
RF	Radio Frequency
RO-SLAM	Range Only Simultaneous Localization and Mapping.
RSSI	Received Signal Strength Indicator
SEIF	Sparse Extended Information Filter
SLAM	Simultaneous Localization and Mapping.
UKF	Unscented Kalman Filter
WSN	Wireless Sensor Network

CHAPTER 1: INTRODUCTION

One of the most important applications of robot navigation is path planning. Path planning algorithms are used to move a robot from one point to another in an effective manner. In order to move a robot from one point to another, it is also necessary for the robot to localize and find the exact location of itself on a global map. Robots cannot rely on odometry data only for localization since odometry is noisy and can lead to a false location of the robot. Hence, in order to get the exact location of the robot, various localization methods have been proposed and are already in use in industry.

Probabilistic robotics is one of the widely used approaches for the localization of robots. Extended Kalman Filter (EKF) [1], Sparse Extended Index Filter (SEIF) [2] and particle filter [3] are few of the famous approaches in probabilistic robotics [4]. These approaches find the position of a robot with some mean value and covariance along with the mean value. However, various changes are made in the environment for the practical application of these methods. For examples, magnetic strips or visual markers or barcodes [5] are used around robots for localization. The barcode, for example, can help in finding the exact location of the robot in the map if the location of the barcode is known on the map. These applications are possible in the industry but not in the places where a map of the environment is not known. In that case, a robot has to map an environment first and then use the map in the future to perform a task. One of the most frequently used approaches to solving this problem is Simultaneous Localization and mapping (SLAM).

SLAM is the problem of creating a map of the environment along with localizing itself in the created map. Robots use measurements from some unique feature in an environment (known as a landmark) to create a map of an environment. Landmarks can be any unique feature like a wall, a table, a tree or any unique pattern in the environment. Landmarks are used not only to create a map of an environment but also to localize a robot. Odometry data received from a robot is used to *predict* the new location of a robot while measurement data is used to *update* the location of a robot. In order to map the environment, odometry data is used to predict the new location of a landmark and measurement data is used to update the location of a landmark. There are many articles on SLAM using a single robot [6]. However, some tasks require more than one robot. For example, Amazon robotics uses multiple Kiva robots to increase the speed of product delivery. In the same way coordination between robots can also be found helpful in exploring and creating a map of an unknown environment. This is known as multi-robot SLAM.

While implementing SLAM using a single robot is a difficult task, adding multiple robots to this concept creates more complexity to the implementation. The missions can be accomplished with multi-robot SLAM in a faster, however, it comes with various problems and complexity of implementation. SLAM algorithms using multiple robots are generally found to be more complex for practical implementation. It is also necessary for robots to have a common way of communication in order for multi-robot SLAM to be implemented in an efficient manner. There are many applications of multi-robot SLAM such as search and rescue, intruder detection, disaster management, forest cleanliness, and surveillance. Multi-robot SLAM requires robots to coordinate with each other to explore the environment efficiently.

Researchers have proposed different approaches to multi-robot SLAM using probabilistic methods [7]. Generally, sensors such as cameras are used for SLAM which gives range and bearing of landmarks and another robot. However, sensors like sonar, proximity and radio beacons provide only range data. SLAM implementations using range sensors are known as Range-only (RO) SLAM [8]. While there are many articles on multi-robot SLAM, very few researches have considered extending RO-SLAM to multi-robot level. RO-SLAM does not have the benefit of obtaining the bearing of landmarks or another robot, hence it becomes hard to find the relative distance and bearing between robots. This creates a problem of finding the transformation matrix for robots to merge the map obtained from another robot.

1.1 Radio Beacons

Radio beacons are omnidirectional. It emits signal whose strength reduces with distance in free space. There are different models to describe signal attenuation over distance. The general propagation model is the log-distance path loss model [9]. The logarithmic attenuation model is provided by log-distance path loss model which can be tuned to nearly any environment. The RSSI (in dBm) is given as:

$$RSSI = 10nlog_{10}d + A \tag{1.1}$$

where n, d and A are path-loss exponents, transmission distance and reference value which is RSSI at 1 meter from transmitter respectively. The distance between receiver and transmitter can be found using equation 1.1 as :

$$d = 10^{\frac{RSSI-A}{10n}} \tag{1.2}$$

The path loss exponent, n can be found for the environment by recording RSSI value at some known distance from the receiver.

1.2 Motivation

Most of the multi-robot SLAM solutions are achieved using sensors which can give both distances and bearing measurements [7]. Very few articles are published on the multi-robot range only SLAM. Dali in [10] proposed unique approach to multi-robot range only SLAM. However, Dali assumed that all robots have an inertial measurement unit (IMU) angle of magnetic north. This assumption helped Dali to solve the problem of obtaining the orientation of another robot during a rendezvous event. Practical implementation of multi-robot SLAM might not have pre-knowledge of IMU angle for all robots.

Since range sensors do not provide information about bearing, it becomes hard to detect the relative position of another robot. Relative distance is important information for merging maps created by robots. Although range sensors have disadvantages that it lacks bearing measurement, few range sensors like radio beacons come with advantages which make it more likely to be used in SLAM.

Most of the proposed multi-robot SLAM implementations have a high computational cost. This computational cost is directly proportional to the number of landmarks. Radio beacons can solve this memory scaling problem by reducing the landmarks in the network to a finite number. Also, robots share the explored map to another robot during rendezvous, limiting the landmark reduces the data to be sent from one robot to another. Therefore, the bandwidth required for data transfer is reduced leading to an increase in performance time of robots.

Another advantage of using radio beacons is avoiding the data association problem of landmarks. Since each radio beacon has a unique network address, identification of radio beacon becomes easy. Hence the usage of radio beacons avoids complex algorithms like Nearest neighbor (NN) or Join Compatibility Branch and Bound (JCBB) which are used for identification of common landmarks while map merging.

Radio beacons also do not require line of sight like many other sensors for detection of a landmark. This helps in detection of landmarks without coming into the line of sight and also makes detection of another robot easier.

1.3 Objective

The main objective of this thesis is to implement multi-robot SLAM using range sensors like radio beacons to explore the map of the environment in less time. This implementation does not make any assumptions about the initial locations of landmarks or initial positions of robots. This implementation reduces the time required by robots to explore the environment. This implementation does not require any common landmark between the robot's maps to merge the map explore by different robots. It is however assumed that all robots know the unique MAC address of each robot exploring the environment.

1.4 Contribution

This work provides a novel method of creating a map using multiple robots. The full particle filter is used for the detection of moving robots. Different robots create maps of their environments in their own reference frame and later share data during rendezvous. Maps in different reference frames are merged without having prior knowledge of robots location and without any help of bearing data. A simulation is built on MATLAB for the implementation of this algorithm. This implementation by simulation is tested with multiple robots without any common landmarks and also with common landmarks.

1.5 Organization

This thesis is organized into five chapters. The first chapter includes the introduction and motivation which is present here. The second chapter includes background on multi-robot SLAM. The third chapter includes a description of different methods of single-robot and the implementation of single-robot SLAM used in this paper. The fourth chapter describes various issues and methods of multi-robot SLAM. It also covers the novel approach performed in this work. The fifth chapter includes simulation of the work performed using MATLAB and the sixth chapter concludes the work performed in this thesis.

CHAPTER 2: RELATED WORK

2.1 Range-only SLAM

Extensive attention was given to wireless sensor systems (WSN) recently for robot navigation in an indoor environment because of easy deployment, low cost, and easy maintenance. Several authors focus on sensors which can measure both range and bearing of the landmark (e.g., laser [11] or camera [12]). Some implementation focuses on low-cost range sensors which can measure only the range of the landmark and not the bearing (e.g. beacons [13]). Ward [14] was first to propose the implementation of localization using only range sensors. The receiver used time of flight of the acoustic impulses to estimate the distance from the mobile unit. The system proved to have a localization error within 14 cm for a receiver at a distance within 1.2 m. Later, a more advanced system was proposed using a combination of ultrasound and RF technology [15]

In many proposed methods, the Received Signal Strength Indicator (RSSI) was used to estimate the distance. Various methods have been proposed to obtain range measurement from RSSI signals. One of the initial systems using RSSI model was implemented by Bahl [16]. In this paper pre-built maps using RSSI signals were used to find a better estimate of robot position. It becomes more difficult for a robot to localize using only range sensors if the map of the environment is unknown, which is the SLAM problem. This problem is more difficult when only range measuring sensors are used as compared to the range and bearing measuring sensors as it is difficult for robots to find the orientation of landmarks. However, recent advances in WSN have increased the usage of range-only SLAM. Kantor [13] presented a method to implement SLAM using beacons. Monte Carlo and Kalman filter techniques gave a better estimate of the map. Leonard [17] used sonar as range-only data to localize the robot in the environment. Later, Leonard extended the paper [18] and used the Doppler effect of SONAR for implementing range-only SLAM underwater [19].

Samyak [20] presented a range-only SLAM using beacons. He proposed a unique implementation using particle filter where the location of landmarks are assumed to be independent of each other and robot position. This implementation was proved to outperform EKF SLAM based methods due to the non-gaussian noise generated by multipath. It was also shown that it can overcome negative information, unlike EKF SLAM. The multipath issue occurs because of the reflection of radio signals by the environment which leads to the same signal being received from different areas of the environment. This paper is an extension of Samyak's work.

2.2 Multi-robot SLAM

Multi-robot SLAM requires robots to move in different parts of the environment independent of each other and map the environment. Using multiple robots help in exploring the map of the environment in less time. Since multi-robot SLAM requires the creation of a common map of the environment, it is important for robots to follow a similar standard of map creation.

Multi-robot SLAM was initiated with an Extended Kalman Filter (EKF) approach. Multi-robot SLAM is a simple extension of single-robot SLAM in case of EKF. Numerous papers have been published based on EKF such as cooperative EKF [11], distributed multi-robot SLAM [21], outdoor elevation mapping [12]. All of these implementations use sensors which can sense both the range and bearing of the landmark such as laser scanner or camera. However, the complexity of EKF grows quadratically in state space dimension. Hence, it is difficult to practically implement multi-robot SLAM using EKF.

Nettleton [22] proposed the Sparse Extended Information Filter (SEIF) approach to multi-robot SLAM. This approach seems to be an efficient solution in the space dimension since creating a common map is just an addition of landmarks to the existing state vector. However, the algorithm proposed in this paper had a high computational cost. The map update was time logarithmic in the number of robots on the team. Hence, [23] enhanced this work and proposed a better approach to multi-robot SLAM using SEIF. Two primary issues were observed in SEIF; finding the relative poses and updating map and poses. The estimate of the relative pose can be found by detecting common feature using kdtree. Once the relative transformation is found, map data can be transformed into a global coordinate. Additivity and easy nature of fusion of landmarks is the key feature of multi-robot SLAM using SEIF. However, this implementation assumes that all noise follows a Gaussian distribution which might not always be a good model for sensor noise.

Thrun [24] proposed multi-robot SLAM using particle filter in 2001. This work was capable of handling multi-modal and Gaussian distributions. It used a laser scanner for map creation and scan matching for detecting robots. However, it assumed the initial pose of robots to be approximately known. Later, Howard [25, 26] extended this work in 2006 and proposed multi-robot SLAM using a Rao-Blackwellised particle filter (RBPF). This paper didn't assume a known initial position of the robot. Howard was successfully able to merge the map of robots as well as solve the loop closure problem. Howard also used the laser for map creation and laser scan for map merging technique.

Kurazume [27] on the other hand proposed cooperative positioning system (CPS) for solving the SLAM problem. Tobata [28] extended CPS to multiple robots. Here, robots are divided into two categories known as parent and child robots. Parent robots have all sensors required to map the environment and detect another robot while child robots act as moving landmarks which helps parent robot to perform better localization. Since in CPS robots know the relative position of each other hence the cooperative mapping problem is reduced to mapping with known poses. In CPS, the child robots remain stationary and act as a landmark when the parent robot is moving and then the parent robot remains stationary when the child robots are moving. This procedure continues until the target position has arrived.

As described above, numerous papers have been published on multi-robot SLAM using sensors which can sense both range and bearing. Most of the papers use a laser scanner for the implementation. However, each scan stores a large amount of data which contributes to the complexity of the algorithm. This complexity increases with the scan making multi-robot SLAM complex. Multi-robot SLAM using only range sensors can reduce the complexity. However, it is difficult to find the orientation of another robot using only range sensors. Hence multi-robot range-only SLAM becomes a difficult task, therefore, making it less popular. Recently, Dali [10] proposed multi-robot SLAM using only range sensors which reduces the need for extensive data storage, hence reducing the complexity of the algorithm. However, all robots were built with IMU sensors and it was assumed that all robots have the same magnetic north. Dali used node memory to store a map of one robot in a node and later that map can be transferred to another robot, whenever another robot explores that node. This increased the memory requirement at every node. This implementation can create a problem when a robot's map data is huge as nodes require a large amount of memory to store the data.

CHAPTER 3: SINGLE-ROBOT SLAM

The SLAM problem arises when a robot navigates in an unknown environment without the knowledge of its pose. The only knowledge the robot posses are control inputs ($u_{0:t}$) and measurements ($z_{0:t}$). The term "Simultaneous Localization and Mapping" explains the problem which is mapping an unknown environment while simultaneously localizing the robot in this map. SLAM problem is harder than the problem of localization of the robot in a known environment. It is even more difficult than mapping the environment with known robot poses since the robot pose is to be estimated. There are two main forms of SLAM [29] :

1. Online SLAM: Online SLAM involves estimating the posterior of a map and the current position of the robot over given observations and controls. A probabilistic representation of Online SLAM is as follows :

   $$p(x_t, m | z_{1:t}, u_{0:t-1}) \tag{3.1}$$

   here x_t indicates current position of robot, m indicate the map, $z_{1:t}$ and $u_{0:t-1}$ indicates observation and control of a robot respectively from start time to current time.

2. Full SLAM: Full/Offline SLAM involves estimating the posterior of the entire path of the robot along with map over given observations and controls. A probabilistic representation of offline SLAM is as follows :

   $$p(x_{1:t}, m | z_{1:t}, u_{0:t-1}) \tag{3.2}$$

   here m indicate the map, x_t, $z_{1:t}$ and $u_{0:t-1}$ indicates robot poses, observation and control of robot respectively from start time to current time.

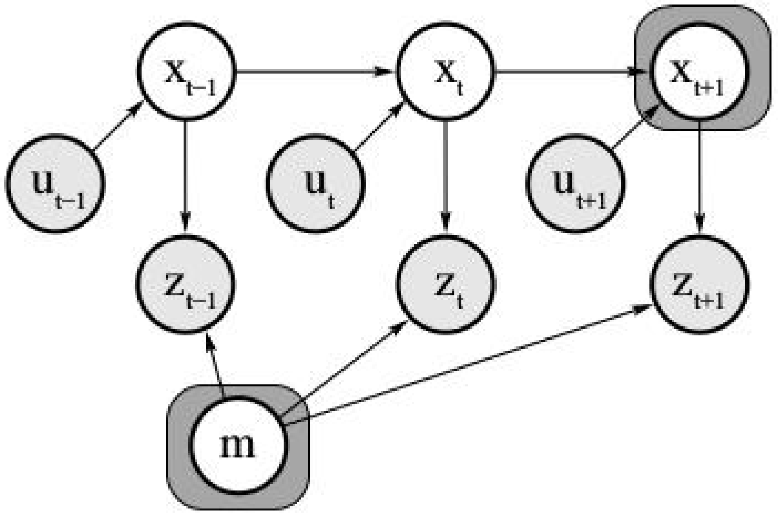


Figure 3.1: Graphical model of the online SLAM problem [4].

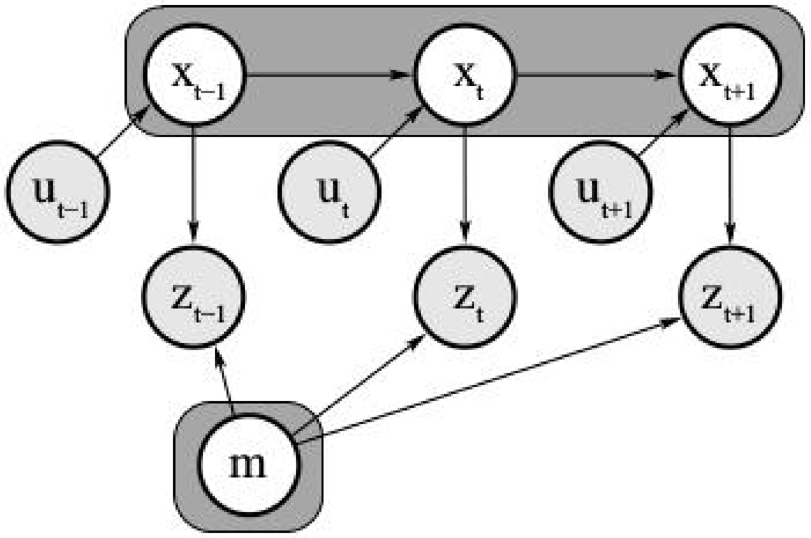


Figure 3.2: Graphical model of the Full SLAM [4].

3.1 SLAM Methods

SLAM can be differentiated based on sensor type or filtering methods used for implementation. Different sensors that can be used for the implementation of SLAM includes range sensors, or range and bearing sensors. Examples of range sensors include sonar and beacons which gives only range measurements of landmarks. SLAM implemented using range sensors are known as range-only SLAM. Examples of sensors with range and bearing include laser scanner or camera, which not only gives range data but also gives bearing towards landmarks. SLAM can also be distinguished based on filtering methods. This method differs by the way they handle the uncertainty when they identify a landmark. Kalman Filter, Extended Kalman Filter, Particle Filter are some of the examples of SLAM based on their filtering technique [20].

3.1.1 Kalman Filter

The Kalman filter creates a better state estimate of an unknown variable with a series of noisy measurements. A Kalman filter can be used in SLAM to estimate the map and pose of a robot with the help of controls and noisy sensor measurements. It has 2 steps: prediction and update. The prediction step is used to predict the state of an unknown variable with given control input and the update step is performed using a measurement. In SLAM, the prediction steps the kinematic model from a sensor and update step uses a measurement model from the sensor to detect landmarks. However, the Kalman filter assumes the noise to be zero-mean Gaussian and model to be linear.

3.1.2 Extended Kalman Filter

A Kalman filter assumes the model to be linear. Linear transformation of a Gaussian random variable is always a Gaussian. However, in the real world, it is not possible for all robots to have a linear model. Applying a nonlinear transformation to a Gaussian variable leads to a non-Gaussian distribution. This will cause the Kalman

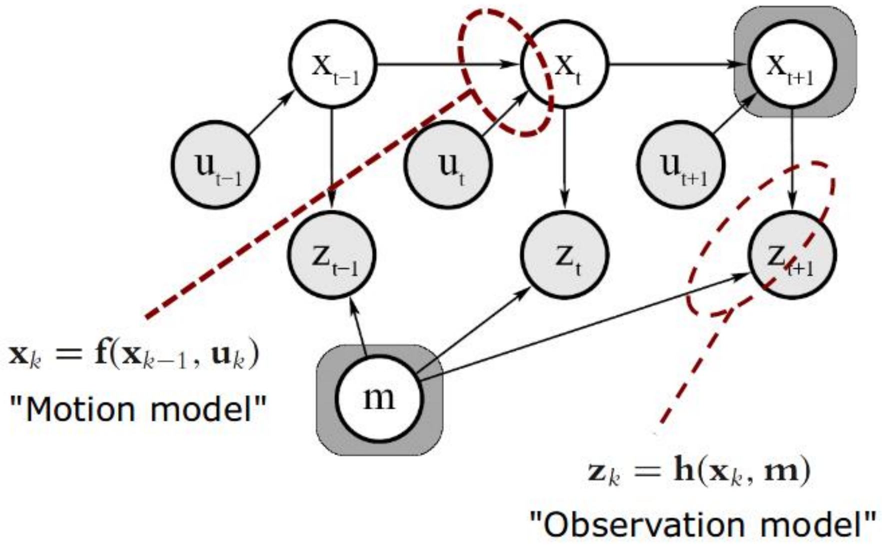

x_{t-1}
x_t
x_{t+1}
u_{t-1}
u_t
u_{t+1}
z_{t-1}
z_t
z_{t+1}
m
$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)$
"Motion model"
$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{m})$
"Observation model"

Figure 3.3: Graphical model of the Full SLAM with motion and observation model.

filter to fail. Hence, various forms of the Kalman filter have been proposed, such as an Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Hybrid Kalman Filter. The most widely used of all is the Extended Kalman Filter (EKF). EKF uses Taylor expansion for linearizing the non-linear model. Linearization of a function is obtained by finding the tangent to that function. This linearization allows the Gaussian distribution to be transformed into Gaussian.

3.1.3 Particle Filter

The particle filter is the nonparametric implementation of the Bayes Filter. The particle filter approximates the Bayes filter by sampling it into a number of discrete samples. One of the advantages of the particle filter is its ability to transform a Gaussian distribution from a non-linear model shown in Figure 3.4 . The particle filters are found to be successful in robot localization where a robot has to localize in given global map [30]. Particle filters have also been proven successful in the kidnapped robot problem [31] where a robot is carried to an arbitrary location. The particle filter is successful in robot application because of two reasons. First is that it can be applied to any model that is probabilistic. Second is that it does not require

[4]

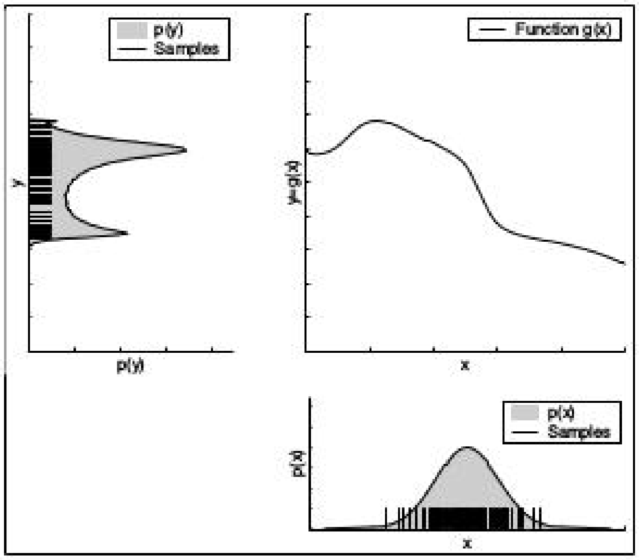


Figure 3.4: Non-linear Transformation of Gaussian model using particle filter. The lower right graph represents a Gaussian distribution sample into discrete particles. The top right function represents a non-linear model. The top left graph represents the transformation of Gaussian distribution over the non-linear model [4]

fixed computation time, in fact, its accuracy increases with time. The accuracy of the particle filter also depends on the number of particles. The Particle filter guarantees to provide a correct result for an infinite number of particles and also it is said that the accuracy of particle increase with the number of particles. The more particles the higher are the chances of particles getting converged at the right location.

particle filter consist of four tasks:

1. Propagate set of particles: Particles are propagated around the possible solutions of the state variable
2. Calculate state estimation: Estimation of the current state is calculated based on given values. In the case of localization, state estimation is performed with the help of control command. If a robot knows its previous state and the control command executed the robot can perform state estimation by applying control command to previous state also known as motion model.

3. Compute particle weight: Particle weight is calculated based on state estimation and observation. In the case of robot localization observation is used to determine the weight of the state estimation. Particles with state estimation close to observation get high weight.

4. Resampling: This is the most important step of the particle filter. Resampling transforms the particles while keeping the size unchanged. Particles are resampled based on their weights. Particles with higher weight get sampled more, while particles with lower weights are not sampled. Resampling can be related to the Darwinian idea of survival of the fittest. It forces the particles to move towards the observation posterior. There are different methods of resampling. Importance sampling and low variance sampling just to name few.

Algorithm 1 The Particle Filter (X_{t-1}, z_t, u_t) [29]

```
$\bar{X}_t = X_t = \emptyset$
 $x_t \approx p(x_t|u_t, x_{t-1})$
**for** $m = 1$ to $M$ **do**
    $w_t^{[m]} = p(z_t|x_t^{[m]})$
    $\bar{X}_t = \bar{X}_t + < X_t^{[m]}, w_t^{[m]} >$
**end for**
**for** $m = 1$ to $M$ **do**
    draw $i$ with probability $\approx w_t^{[i]}$
    add $x_t^{[i]}$ to $X_t$

**end for**
**return** $X_t$
```

Figure 3.5 demonstrates the localization of the robot on the global map. In this example, particles are used to represent the state variable which consists of robot pose. Figure 3.5(a) depicts the map of the environment with particles spread all over the robots as a robot can be anywhere on the map. Dots in the figure represent the state/particles of the robot. As a robot moves along this map, the measurement model is used to resample the state of the robot. As shown in Figure 3.5(b) bad particles die out after resampling as the weight assigned to those particles is low enough. As shown

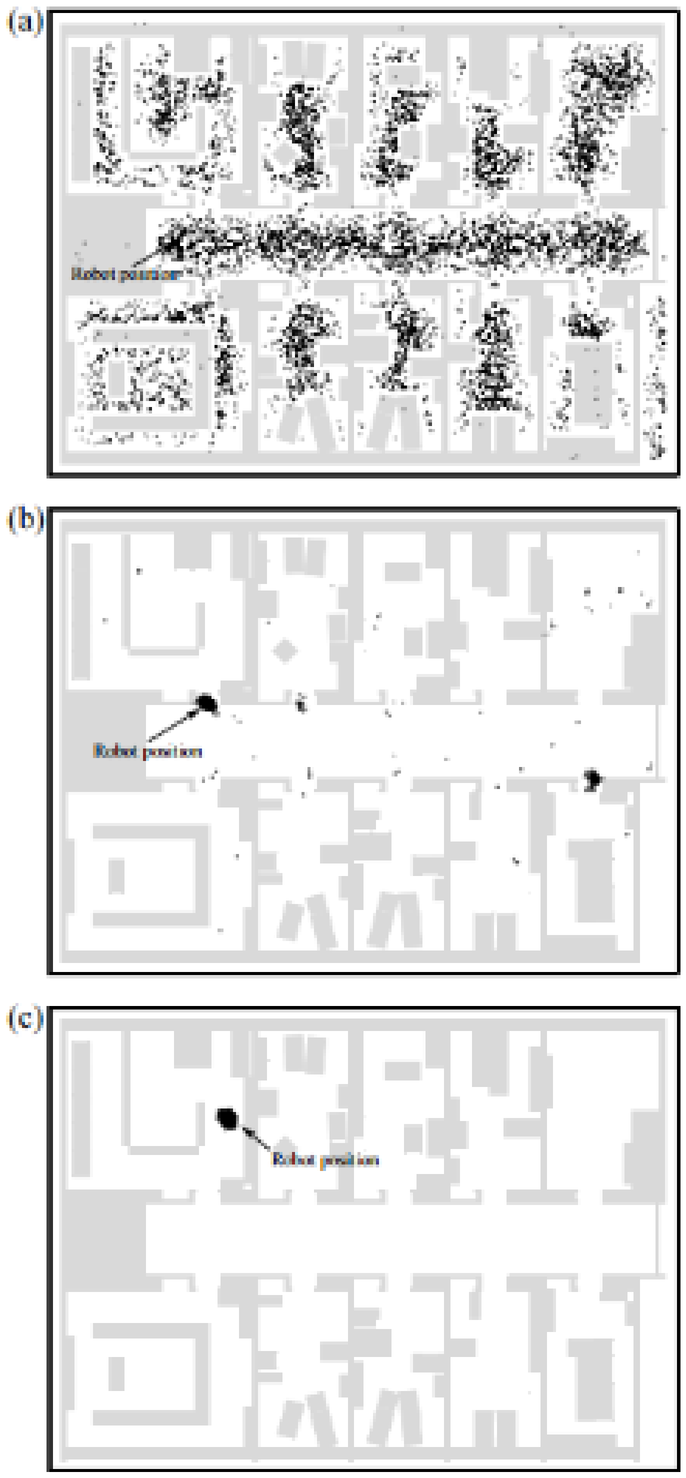


Figure 3.5: Localization using particle filter. Figure (a) demonstrates particles being propagated everywhere uniformly. Figure (b) shows 3 clusters converged due to the symmetrical nature of the passage. Figure (c) shows particles being converged at one point after the robot encountered structurally unique features [20].

in Figure 3.5(b) particles are converged into 3 different areas due to the symmetry of the hallway. once the robot moves inside a room into an environment which has structurally unique features, it is able to converge robot to a proper location in the global map.

3.2 Particle Filter SLAM

SLAM implementation of the particle filter uses the same core algorithm as described in Algorithm 1. Algorithm 2 represent SLAM using particle filter. Here every particle contains a map and poses of the robot counter to just pose of a robot as described in Algorithm 1. state vector for ith particle is given as

$$< x, y, \theta, x_1, y_1, x_2, y_2, x_3, y_3 ....x_n, y_n >$$

where x, y, θ represent pose of a robot and x_i, y_i represent coordinate of ith landmark. Generally, the start of the robot is considered as (0,0). Initially, particles are generated uniformly as shown in lines 1-4. The motion model is used to create a state estimation of state vector as shown in 6-10. Based on the observation, weights are assigned to particles as shown in line 11-14. Resampling is performed based on weights assigned as shown in lines 16-19. Particles with lower weight die off and hence a correct map of the environment is created

3.3 Single-robot Range-only SLAM Implementation

Single-robot range-only SLAM is implemented using particle filter. This implementation is small extension of [20]. In this implementation, it is assumed that the location of a landmark is independent of each other and also independent of robot pose. It is also assumed that robot pose is independent of all the landmark location. Each landmark location is represented with a unique set of particles and also robot position is represented with a different set of particles. Initially, particles for a robot are generated around (0,0), and the start position of a robot is considered as global (0,0) for the map. Since beacons only give range measurement from a landmark,

Algorithm 2 Particle filter SLAM [20]

```
1: Initialization:
2: for $t = 1$ to number of particles do
3:     $X_{(t)}$ = particles uniformly distributed
4: end for
5:
6: State estimation :
7: for $t = 1$ to number of particles do
8:     $x_t \approx p(x_t, m|u_t, x_{t-1})$
9: end for
10:
11: Weight assignment:
12: for $t = 1$ to number of particles do
13:     $w_t = p(m|X_t, Z_t)$
14: end for
15:
16: Resampling
17: for $t = 1$ to $T$ do
18:     draw $i$ with probability $\approx w_t^{[i]}$
19:     add $x_t^{[i]}$ to $X_t$
20:
21: end for
22: return $X_t$
```

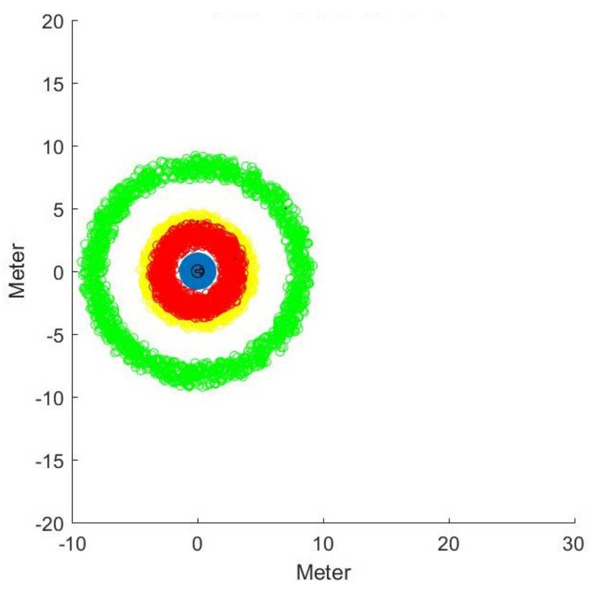


Figure 3.6: Initial phase of single-robot range-only SLAM. Blue circles represent particles for a robot. Red, yellow and green circles represent particles for landmark 1, 2 and 3 respectively. Black circle with arrow mark inside it represent the real position of the robot and black dot represent the real location of the landmark.

particles for a landmark is initialized with a circular disk as shown in Figure 3.6. In Figure 3.6 particles for robot position is represented by blue circles. As one can see it is initialized at the global origin. As one can see the circular disk of particles is created for each range measurement received from beacons. Red, yellow and green circles represent particles for landmark 1, 2 and 3, respectively. The size of the disk can vary depending upon the implementation. Here size of the disk is considered to be 0.5 m.

For every iteration, robot pose is predicted with a motion model. Odometry is used here as a motion mode for state prediction. After the prediction step, observation is used to assign a weight to each particle. Depending upon observation for each landmark, weight is assigned to each particle. Weight is calculated based on the difference between the measured and predicted range. High weights are assigned to particles with measure range near to predicted range. Resampling was done using importance sampling. Resampling process was used to avoid particles with low weights. Figure

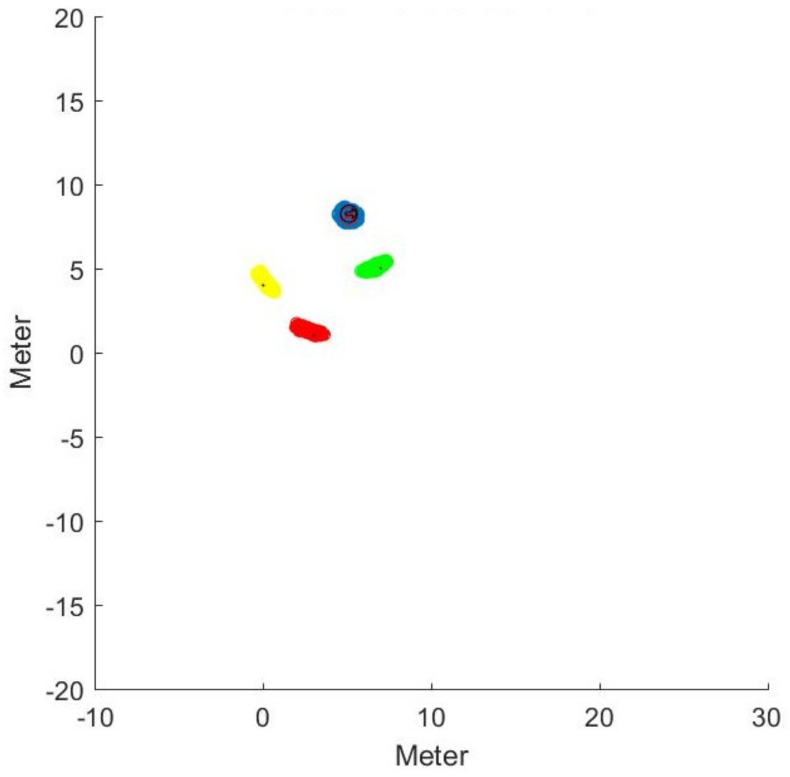


Figure 3.7: Final phase of single-robot range-only SLAM. Blue circles represent particles for the robot. Red, yellow and green circles represent particles for landmark 1, 2 and 3 respectively. Black circle with arrow mark inside it represent a real position of the robot and the black dot represents the real location of the landmark.

3.7 shows the final result of this method.

Trilateration/MultiLateration is used in order to localize a robot. Trilateration is defined as the method of finding the absolute or relative location of a point using range measurements from 3 different points with the help of circular geometry. The multilateration is defined as the use of more than 3 measurements for the same approach.

Trilateration/Multilateration uses the distance between the robot and the beacons. Hence it requires a minimum of 3 beacons to get 3 measurement reading. Trilateration uses the distance to each converged landmark particles to get the estimation of robot location and assign weight accordingly. Figure 3.8 depicts a case of trilateration with an estimation of distance error where the estimation of distance error is because of

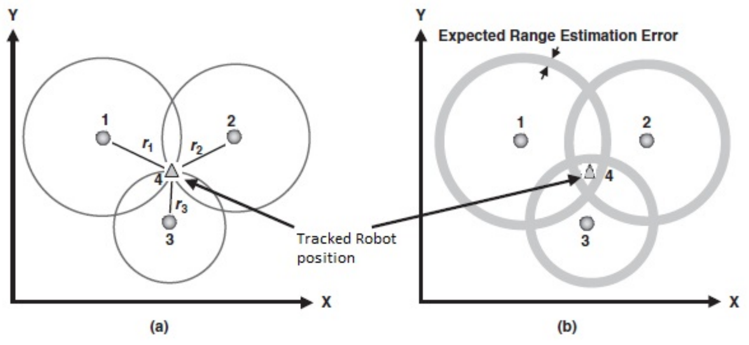


Figure 3.8: Trilateration used to localize robot with the help of converged landmarks.

noisy measurement.

Combination of the particle filter with trilateration enhances the prediction of the pose of the robot. It was observed that the convergence error range from .1 m to .5 m for landmark and robot particles.

Algorithm 3 shows the algorithm used for the implementation. Particle sets X_1 to X_N represent the N landmarks and particle set X_N+1 represent the robot. Landmarks are uniformly initialized with observed range data in lines 3-8. Lines 10-12 represent the initialization of robot around the origin as the robot start point is considered as an origin for a global coordinate. State estimation is updated with motion model as shown in lines 15-17. Lines 21-43 represent the weight assignment and resampling for each particle set. Weights are assigned to particles of landmark based on the received range measurement. The difference between each particle of one set and mean of the particle set $X[N + 1]$ give the expected range measurement for each particle. This expected range measurement is subtracted from measured range to get weight $w_t^{[n]}$ for each particle of the landmark. Particles are resampled based on weights assigned to each particle. This step is performed for each set of particles representing landmarks. Lines 21-25 represents the weight assignment process and lines 26-28 represent the resampling process. Robot particles are updated only when three or more landmarks are converged. Lines 33-37 depicts the weight assignment

Algorithm 3 Single-robot range-only SLAM implementation

```
1: Initialization:
2: Landmarks:
3: for $n = 1$ to number of landmarks do
4:     for $t = 1$ to number of particles do
5:         $X_{(t)(n)}$ = particles uniformly distributed
6:         $\bar{X}_{(t)(n)} = \emptyset$
7:     end for
8: end for
9: Robot:
10: for $t = 1$ to number of particles do
11:     $X_{(t)(N+1)}$ = particles at the origin
12: end for
13:
14: State estimation Step :
15: for $t = 1$ to number of particles do
16:     $x_t^{[N+1]} \approx p(x_t|u_t, x_{t-1}^{[N+1]})$
17: end for
18:
19: Weight assignment and resampling Step:
20: Landmarks:
21: for $n = 1$ to number of landmarks do
22:     for $t = 1$ to number of particles do
23:         $w_t^{[n]} = p(z_t|X_t^{[n]}, mean(X_t^{[N+1]}))$
24:         $\bar{X}_t^{[n]} = \bar{X}_t^{[n]} + < X_t^{[n]}, w_t^{[n]} >$
25:     end for
26:     for $t = 1$ to $T$ do
27:         draw $i$ with probability $\approx w_t^{[i]}$
28:         add $x_t^{[i]}$ to $X_t^{[n]}$
29:
30:     end for
31: end forreturn $X_t^{[1:N]}$
32: Robot:
33: if (3 landmarks converged) then
34:     for $t = 1$ to number of particles do
35:         $w_t^{[n]} = p(z_t|X_t^{[N+1]}, mean(X_t^{[n]}))$
36:         $\bar{X}_t^{[N+1]} = \bar{X}_t^{[N+1]} + < X_t^{[N+1]}, w_t^{[n]} >$
37:     end for
38: end if
39: for $t = 1$ to $T$ do
40:     draw $i$ with probability $\approx w_t^{[i]}$
41:     add $x_t^{[i]}$ to $X_t^{[N+1]}$
42:
43: end for
44: return $X_t^{[N+1]}$
```

process for particles of robot pose. In the case of robot pose weight assignment, mean of converged landmark particles are subtracted from each particle of robots position to get the error of robot position with respect to each landmark. This error is added to get the total error. Maximum error corresponds to lower weight and vice versa. Once the weights are assigned to each particle of robot pose, robot particles are resampled as shown in lines 39-43.

This implementation has higher computation cost and it scales with the number of landmarks. However, since the number of landmarks is limited in this implementation, its computation cost was found to be better than other approaches which use both range and bearing for example feature based SLAM.

CHAPTER 4: MULTI-ROBOT SLAM

Multi-robot systems are found to be useful in many complex exploration task and other applications. Multi-robot systems have proven to be quicker and more accurate with regards to the mapping of an environment [32]. In some cases, there is a need for a multi-robot system to accomplish a task. For instance, without the assistance from different robots, a solitary robot might be helpless against an unfriendly domain or adversaries, such as in some military activities or investigating an insecure structure. In a few different applications, a robot may get help from another close-by robot during an emergency, for example, during failures or malfunctions [33]

Multi-robot SLAM is found to be more efficient in the exploration of an unknown environment [7]. In some cases like huge environments, usage of multiple robots can reduce the time of exploration. It was also found to be useful in decreasing the convergence error of landmark location. Multi-robot SLAM using two robots can be formulated as:

$$p(x^a_{1:t}, x^b_{1:t}, m | z^a_{1:t}, u^a_{0:t-1}, x^a_0, z^b_{1:t}, u^b_{0:t-1}, x^b_0)$$

where $x^a_{1:t}$, $z^a_{1:t}$ and $u^a_{0:t-1}$ denotes position observation, and control command of *robot a* respectively. Here $x^b_{1:t}$, $z^b_{1:t}$ and $u^b_{0:t-1}$ denotes position, observation, and control command of *robot b* respectively. Here m denotes the map of environment, and x^a_0 and x^b_0 denote the initial position of *robot a* and *robot b*, respectively. Applications of multi-robot SLAM include rescue, museum guidance, restaurant servant and surveillance [34]. Multi-robot SLAM is advantageous as tasks can be divided among many robots but at the same time, it has various issues.

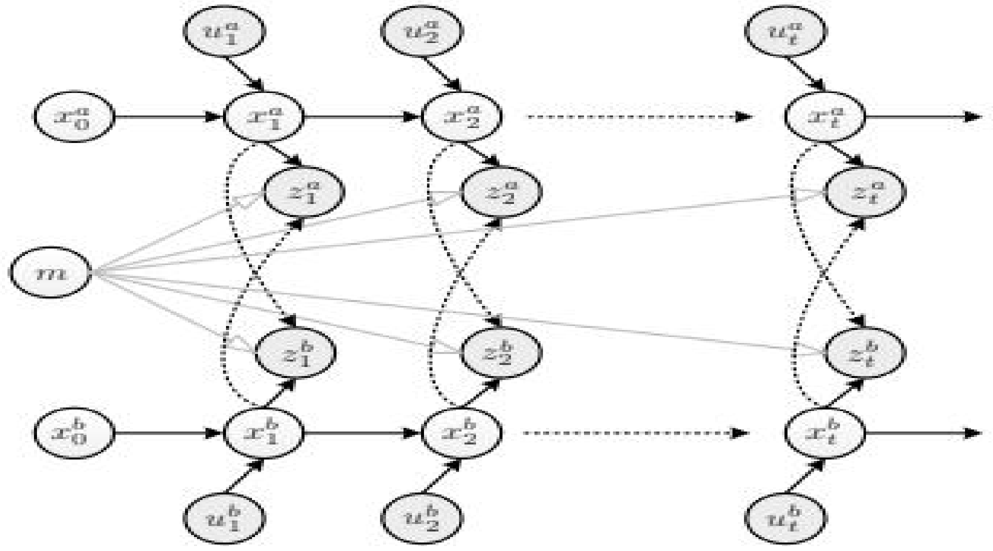


Figure 4.1: Bayes net for particle filter multi-robot SLAM with the unknown initial location. At time s, robot particle is converged and $\triangle_s$ is known.

4.1 Multi-robot SLAM issues

While multi-robot SLAM has many advantages, it also has various issues and complexities. Following are many known issues [7, 35] in multi-robot SLAM:

4.1.1 Robot Pose Estimation

It is necessary for robots to know the coordinate transformation [36] between the detected robot's reference frame and it's own reference frame for it to merge the map transferred by the detected robot. If the initial location of robots with respect to some global reference frame is known then this problem is solved [37, 24]. However, in practical applications robots do not always know their relative position at the initial stage. The relative position of robots can be found for robots with unknown initial location [38, 39], but it depends on how accurate the sensors are. Measured relative distance and bearing can have a noise which will lead to inaccurate map merging.

4.1.2 Robot Rendezvous

If the initial position of robots is unknown then robots are obliged to meet each other in order to find the transformation matrix and create a common map. In a real-time system, there can be a possibility that robots never meet each other. Hence, there should be a mechanism by which robots can meet each other while exploring an unknown environment [40, 41].

4.1.3 Coordination Exploration

One of the main reason for moving from single-robot SLAM to multi-robot SLAM is efficiency in the exploration of an environment. To increase the efficiency of map exploration, it is important for robots to navigate to an unexplored area rather than all robots exploring the same area. This includes robots to have knowledge of their relative position and coordinate to move to feasible frontiers for map exploration [40, 42, 43] where frontiers are the extreme of a map which is unexplored at the opposite end [44, 45].

4.1.4 Map Merging

As robots always create a map in their own reference frame, merging two maps in the different reference frame is not an easy task. Various techniques have been proposed for merging maps with different reference frames [46, 41]. However, it is important for the map to be merged with proper alignment because if the map is not merged in proper alignment, then it will create the wrong estimation of the environment. In order to avoid such issues, consistent representation of the map is required. Also, it is important to find duplicate landmarks across the map while merging [38].

4.1.5 Limited Communication

Multi-robot SLAM can be implemented using a centralized system or a distributed system. If the system is centralized, then there is a problem that robots are restricted to relocate only within the area of communication with the centralized system. Also, if the centralized system communication crashes then the whole system crashes. In case of a distributed system, robots can share information whenever rendezvous occurs but if the rendezvous was after a long time then there is a large information which has to be transferred between robots. This volume of information needs high bandwidth, and this can create a problem during communication [36].

4.1.6 Complexity and Memory Requirement

It is necessary to design an algorithm which is scalable with respect to the number of robots. If the complexity of the algorithm increases with an increasing number of robots then there arise issues while including new robots for exploring a large environment. It is also required that the algorithm should also be capable of solving SLAM in real time.

4.1.7 Intrinsically Dynamic Environment

Constantly changing environments can also create problems as two robots passing through the same area will see different features. This will create a problem later on while map merging or integrating data from one robot to another [47].

4.1.8 Heterogeneous Robots and Sensors

Different kind of robots has different capabilities. The quadcopter can see the environment in a way that a land robot cannot and vice verse. Hence, multi-robot SLAM will be more effective if different robots having different capabilities and sensors are able to cooperatively explore and map the environment. Wurm [48] and Michael [49] has demonstrated a good example of map exploration with heterogeneous robots.

4.1.9 Synchronization

For any machine to communicate with each other it is important that the clock is synchronized between the robots.Synchronization can be done online or offline. Time synchronization by Chroy is a suitable choice. One of the examples of online

Synchronization by Chrony is Network Time Protocol (NTP) which is also used in ROS. In case of offline time synchronization, time can be inserted on a regular basis.

4.1.10 Performance Measurement

It is difficult to measure the performance of the map created by robots as the exact map of the environment is not known and also it is hard to detect correctness of the trajectory of robots.

4.2 Multi-robot SLAM methods

Multi-robot SLAM can be implemented in different ways. All the robots need to coordinate with each other in order to create a common map. This coordination can be done in a centralized server or each robot can coordinate independently. Depending upon how robots coordinate multi-robot SLAM can be divided into 2 types; Centralized and Distributed

4.2.1 Centralized

In this implementation, there is a centralized server where each robot sends data and hence create a common map [11, 50]. A centralized system is responsible for receiving data from all robots and creating a common map out of it. There are chances of high space and time computation in centralize system as a huge amount of data will be sent to a centralized system. It requires high bandwidth in order to receive all the information from robots. The highly complex task might be impossible becomes it requires a continuous transfer of huge data and keeping track of all robots position in the global map. Centralize system can also fail if the central system responsible for all operation crashes.

Centralize robots also have some advantages. The map can be explored effectively with the help of centralized systems. The centralized system can command robots to navigate to the unexplored area, thus exploring the environment in an inefficient manner. It reduces the chances of robots navigating to already known location.

4.2.2 Decentralized

A decentralized system can be viewed as similar to insects, which don't have a specialist that controls the entire framework and the behavior of each robot is independent of other [51]. Decentralized design can be completely dispersed in which all robots are equivalent as for control, or various leveled, in which they are privately concentrated. Some of the decentralized systems are explained in [52, 53, 54]. There is a constraint of the nearest neighbor in the communication of distributed systems. Decentralize system is more reliable and robust since each robot just provides local sub-map. One of the advantages of a distributed system is that the failure of one robot will not affect the whole system.

The distributed robot can be implemented in different ways. EKF, SEIF and particle filter are some of the famous approaches to multi-robot SLAM. Multi-robot SLAM using EKF requires to find the transformation matrix between two different reference frame and merging the map accordingly. SEIF is simple in terms of implementation of multi-robot SLAM as the addition of landmark from another robot is an additive process. However, both approaches assume Gaussian noise distribution.

4.3 Multi-robot SLAM using particle filter

Recently, the various approach have been proposed on multi-robot SLAM using particle filter. Multi-robot SLAM using particle have been found to be an efficient approach as it does not assume Gaussian distribution either does it assume linear model. Implementation of multi-robot SLAM using particle filter can be distinguished into 4 sections based on the assumptions:

4.3.1 Known Initial Position

The simplest approach assumes that the initial pose of robots is known to each other [37, 11]. This solves the problems of robot estimation. Hence the multi-robot SLAM becomes easier as the traces from different robots can be treated as the sensor data of a single robot. Let $x^a_{1:t}$ indicates trajectory of *robot a*, $z^a_{1:t}$ indicates the corresponding observation and $u^a_{0:t-1}$ indicates the corresponding control commands. Let $x^b_{1:t}$ indicates trajectory of *robot b*, $z^b_{1:t}$ indicates the corresponding observation and $u^b_{0:t-1}$ indicates the corresponding control commands. Multi-Robot SLAM can be formulated as

$$\begin{aligned} p(x^a_{1:t}, x^b_{1:t}, m | z^a_{1:t}; u^a_{0:t-1}, x^a_0, z^b_{1:t}; u^b_{0:t-1}, x^b_0) = \\ p(x^a_{1:t}, x^b_{1:t} | z^a_{1:t}, u^a_{0:t-1}, x^a_0, z^b_{1:t}, u^b_{0:t-1}, x^b_0) \times \\ p(m | z^a_{1:t}, u^a_{0:t-1}, x^a_0, z^b_{1:t}, u^b_{0:t-1}, x^b_0, x^a_{1:t}, x^b_{1:t}) \end{aligned} \tag{4.1}$$

$$\begin{aligned} p(x^a_{1:t}, x^b_{1:t}, m | z^a_{1:t}; u^a_{0:t-1}, x^a_0, z^b_{1:t}; u^b_{0:t-1}, x^b_0) = \\ p(m | z^a_{1:t}, z^b_{1:t}, x^a_{1:t}; x^b_{1:t}) \times \\ p(x^a_{1:t} | z^a_{1:t}, u^a_{0:t-1}, x^a_0) \times p(x^b_{1:t} | z^b_{1:t}; u^b_{0:t-1}, x^b_0) \end{aligned} \tag{4.2}$$

Here $p(m|z^a_{1:t}, z^b_{1:t}, x^a_{1:t}; x^b_{1:t})$ indicate map distribution while $p(x^a_{1:t}|z^a_{1:t}; u^a_{0:t-1}, x^a_0) \times p(x^b_{1:t}|z^b_{1:t}; u^b_{0:t-1}, x^b_0)$ indicates distribution over the trajectory of robot. This is assuming that the trajectory of robots and initial position of the robots are independent of each other. State of the particles are represented as $< x^a(i)_t, x^b(i)_t, m(i), w(i) >$.

Since the initial pose is assumed to be known the only problem remains here is finding the transformation matrix and creating a common map of the environment. In [37, 55] feature based multi-robot SLAM is demonstrated with known initial positions. The particle filter is used to estimate the pose of robots and EKF is used to estimate landmarks location. Hence, all the particles contain different EKF estimation of landmarks. Each particle is used for updating a single global map. Prediction and update step is performed for each robot while weight is calculated by multiplying each term in a single robot SLAM. Kai in [37] came up with real data experiment and concluded that convergence error from multi-robot slam is 0.5 m while that from single robot SLAM is 0.7 m

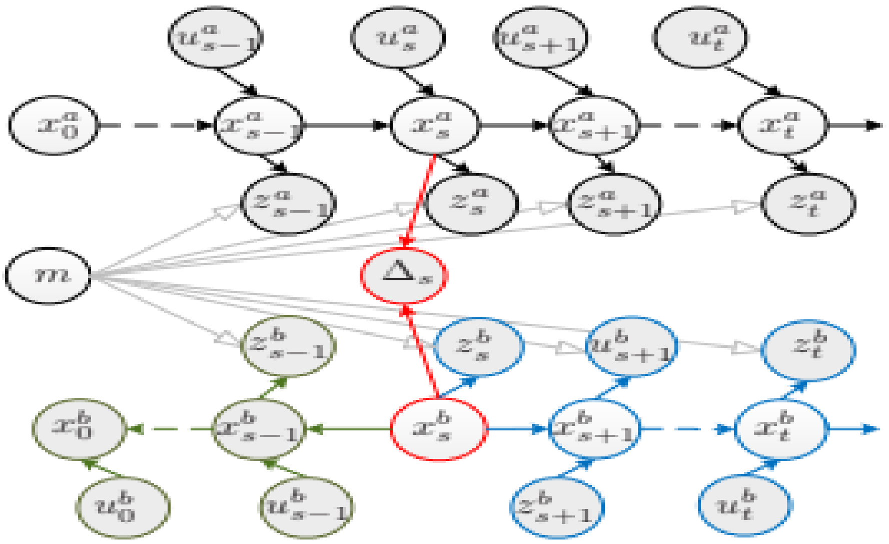


Figure 4.2: Bayes net for particle filter multi-robot SLAM with the unknown initial location. At time s, robot particle is converged and $\triangle_s$ is calculated.

4.3.2 Rendezvous

In this approach, robots do not know the initial position of each other but it is assumed that robots will meet each other at some point [38, 39]. In this approach, each robot creates map independently and then they share information and create a common map at the event of rendezvous. There are two disadvantages of this method, one is that the map will never merge if robots never meet each, while other is that if robots are moving along the same trajectory then it will not be detected and there will be no use of multi-robot SLAM. Since robot's relative position is not known hence there are three problems in this approach:

1. Finding relative position: The relative position of robots can be found by different sensors. Howard [26] has used a laser scanner as a sensor for robots. In [36] robots are equipped with laser tilt camera which helps them to measure the relative distance between robots along with their relative uncertainty. In [56] 3D camera is used as a sensor for detection of landmarks as well as for detection of another robot.

2. Finding the transformation matrix: Zhou [38] has come up with an approach of finding the transformation matrix to represent all landmark and robot position from the robot's local coordinate frame to the global coordinate frame.

3. Merging map: Once robots have received all the data and transformation matrix has been calculated, data can be incorporated as if it was sensor and odometry data of its own. Carlone [36] has demonstrated this approach with two robots.

Howard [26] has come up with online multi-robot SLAM using particle filter. Howard firstly explained itâs approach with known initial poses and later on about unknown relative pose estimate of robots. According to Howard, multi-robot SLAM with unknown initial position can be stated as

$$\begin{aligned} p(x^a_{1:t}, x^b_{1:t}, m | z^a_{1:t}, u^a_{0:t1}, x^a_0, z^b_{1:t}, u^b_{0:t1}, \triangle^b_s) = \\ p(m | x^a_{1:t}, z^a_{1:t}, x^b_{1:s1}, z^b_{1:s1}, x^b_{s+1:t}, z^b_{s+1:t}) \\ \times p(x^a_{1:t} | z^a_{1:t}, u^a_{0:t1}, x^a_0) \times p(x^b_{1:s1} | z^b_{1:s1}, u^b_{0:s1}, x^a_s, \triangle^b_s) \\ \times p(x^b_{s+1:t} | z^b_{s+1:t}, u^b_{s:t1}, x^a_s, \triangle^b_s) \end{aligned} \tag{4.3}$$

where $\triangle^b_s$ denotes transformation from *robot b* co-ordinate frame to *robot a* co-ordinate frame and s is the time of rendezvous. Here *robot b*'s data is divided into two sections known as a casual and an acasual instance. Casual instance defines the particle after rendezvous time while an acasual instance indicates particles before rendezvous time. Here, old data from *robot b* is processed at the same time as the recent data from *robot a*. All three particles are used at the same time to update the single map. Since data from another robot is incorporated along with the map exploration and hence, this method is known as online SLAM algorithm. One of the drawbacks of this algorithm is that it takes O(nm) time each cycle where n is the number of robots and m is the number of particles because of the fact that the observations (old or new) are processed at the same time. Another drawback of this algorithm is latency. Suppose *robot c* meet *robot b* at time t=p. It will take $2\times s - p$ time for processing data from *robot c* which is a huge time. In the worst case, latency can be $(n-1)\times t$ where n is the number of robots.

4.3.3 Location Hypothesis

In this method, it is not assumed that robots will meet each other at some point. In this approach, robots start with unknown initial location but as soon as robots come under a communication range, they start sharing sensor data to create a hypothesis of their relative location [40, 41]. This approach solved the coordination exploration problem of SLAM. Additionally, to this, there is one more practical approach but this approach will fail if robots never come in contact with each other. In this approach, once a hypothesis is made, it is necessary to verify the hypothesis. This verification is done with the help of a rendezvous event. If this rendezvous event fails then robots continue to explore the map individually and create a new hypothesis of their relative location. Once a hypothesis is verified then the transformation matrix is found and the map is merged by exchanging data as explained before.

In [40] Fox has demonstrated this coordination exploration of robots. Fox has not only used this hypothesis approach to approximate the relative location but also came up with an efficient environment exploration strategy. Fox formed a cluster of robots whenever robots meet each other. It is assumed that robots can communicate within a cluster and can share data with each other. Cluster size increases with robots. Robots are not added in the cluster until and unless the position of the robot is predicted exactly. Robots in a cluster share a map, and position of all robots in a shared map is known. To verify the position of the recently explored robot, one of the robots in the cluster is sent to the detected robot position and the position is verified. A robot is added in a cluster only after verifying the position. Laser scanner and marker on the robot is used to identify the robot. In order to explore the environment in a more efficient manner, Fox came up with a cost and utility function.The cost is found by the minimum cost path between frontier and robot. In case of a robot moving towards a hypothesis, the cost is found by finding the minimum cost path between robot and meeting point plus the cost of whether two robots will meet or not. The utilities is given by the expected area a robot will explore at that frontier. If the target is hypothesis then the utility is given by the expected utility of meeting robot. The decision of moving the robot towards hypothesis or frontier is done depending upon the expected utility and cost. Cost should be low and utility should be high for the task to be assigned to the robot Once pairwise utilities and cost are found, the linear solver is used for finding the optimal target for all robots. It is usually observed that the robot moves towards hypothesis only if the cost of moving to that position is less and utility is high. This makes sure that the environment is explored in a more efficient manner.

4.4 Implementation

In the implementation of multi-robot SLAM, a single robot explores the map by Algorithm 3. Different particle sets are used to represent landmarks and robots. All robots explore the map of the environment independent of each other. This paper does not assume any prior knowledge of robots positions with respect to each other. It is assumed that the robot rendezvous event will occur in this implementation. In this implementation, each robot has its own reference frame. Here, each robot is equipped with beacon sensors. All robots know the unique IDs of all the robots exploring the environment. Since robot's start point is considered as (0,0) for both hence both robots have their own global reference frame.

This implementation is tested in MATLAB simulation. $40\times40m^2$ area of the environment was selected to explore the map of the environment. The maximum distance by which a robot can detect a beacon is 10 m. Each robot has a range of 10 m to detect the landmark. Hence in order for a robot to detect each other without any common landmark there should be a distance of more than 10 m between detected robot and the landmark explored by another robot. This addition of 10 m for the distance between landmark and robot and detected robot shows that having a $40 \times 40 m^2$ of environment can be used to explore the environment in an efficient manner.

The robot rendezvous event is used to merge the map of the environment. During robot rendezvous, each robot propagates particle for another robot in the ring of radius equal to the range measurement received by the robot. Once the particle is propagated, a detected robot is localized by updating particle using control command and then resampling by the help range measurement between self-robot and detected robot. Common landmark if there is any is also used to localize the detected robot. Once localized, the map is received from the detected robot and the transformation matrix is applied to merge the map.

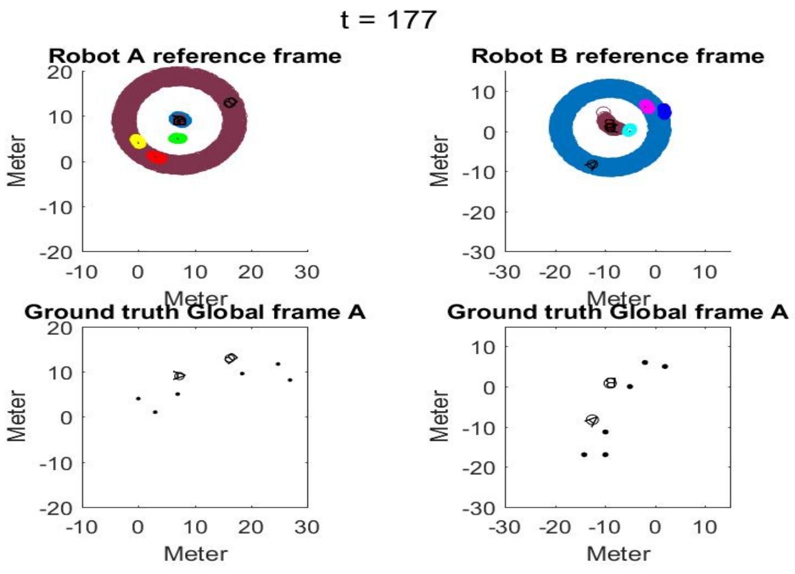


Figure 4.3: Rendezvous event between *robot a* and *robot b*. *robot a* detect *robot b* and propogate ring of particles. Brown circle indicates particles for *robot b*. Blue circle indicates particles for *robot a*. Yellow, green and red circles indicate particles of landmarks explored by *robot a*

Let us consider two robots *robot a* and *robot b* for detail explanation of the algorithm. Whenever a robot rendezvous event occurs, a circular set of particles are created for the detected robot as shown in Figure 4.3. Here, blue circles denote particles for *robot a* and brown circles denote particles for *robot b*. Let us consider one side for better explanation. As shown in Figure 4.3 particles are propagated in ring shape around *robot a* with the help of range measurement received from sensor. Once *robot b* is detected, landmark IDs are shared between *robot a* and *robot b*. If there is any common landmark, then range measurement between common landmarks and *robot b* is used to converge particles of *robot b*. Apart from common landmark, *robot a* also receives control command from *robot b* and apply motion model to all particles of *robot b*. This particles are later updated by assigning weight based on the range measurenment received and expected range measurement between *robot a* and *robot b*. This weights are used to resample the particles of *robot b*. Particles are resmaple till the x, y position of the robot is below 0.3 m variance and the orientation of the robot is below $0.1°$. The reason for this selection was to reduce the error while transferring data from one robot to another. In this way *robot b* is localized in *robot a* global map. *Robot b* is considered to be localized once particle for *robot b* are below a thresahold point. Once *robot b* is localized, *robot a* receives data from *robot b* which is the pose of *robot b* in global reference frame of *robot b* and particles of landmark in *robot b* global reference frame.

Once data is received from *robot b*, *robot a* transform data from global *robot b* reference frame to global *robot a* reference frame. This is performed by transforming map from global *robot b* reference frame to local *robot b* reference frame and then from local *robot b* reference frame to global *robot a* reference frame.

Pose of *robot b* in global reference frame of *robot b* is used to find the transformation matrix from Global B reference frame to local B reference frame. The position of *robot b* in global *robot a* reference frame is used to find transformation matrix from local *robot b* reference frame to global *robot a* reference frame. Once the transformation matrix is found then this matrix is applied to each particles of landmarks in *robot b* global reference frame to transform it to *robot a* global reference frame.

4.4.1 Transformation

Transformation of a point from one coordinate frame to another requires it to not only translate but also rotate depending on the destination reference frame. Let's take two reference frames; A and B as shown in Figure 4.4.

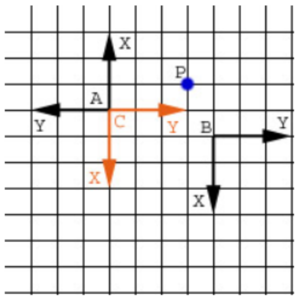


Figure 4.4: Transforming data from reference frame A to reference frame B

Here a point in reference frame A will have a different coordinate in reference frame B. The reference frame A and B differ by both rotation and translation. The transformation matrix can be calculated to transform data from A to B(M_{AB}) in two steps; finding transformation matrix to transform from reference frame A to reference frame C (M_{AC}) and then transformation matrix to transform data from reference frame C to reference frame B (M_{CB}). Transforming data from reference frame A to C involves rotating reference frame by 180° anticlockwise. The rotation matrix is given as:

$$rotationMatrix = \begin{vmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix} \tag{4.4}$$

Hence M_{AC} is given as:

$$M_{AC} = \begin{vmatrix} cos(180) & -sin(180) & 0 \\ sin(180) & cos(180) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Now transformation from reference frame C to A is a translation by 4 unit along x and 1 unit along y. Translation matrix is given as :

$$translationMatrix = \begin{vmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{vmatrix} \tag{4.5}$$

where t_x and t_y are unit shift along x and y respectively. Hence M_{CB} is given as:

$$M_{CB} = \begin{vmatrix} 1 & 0 & -1 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{vmatrix}$$

Combining these two gives M_{AB} as:

$$M_{AB} = M_{AC} \times M_{CB}$$

4.4.2 Flowchart

Figure 4.5 shows the flowchart of the implementation.

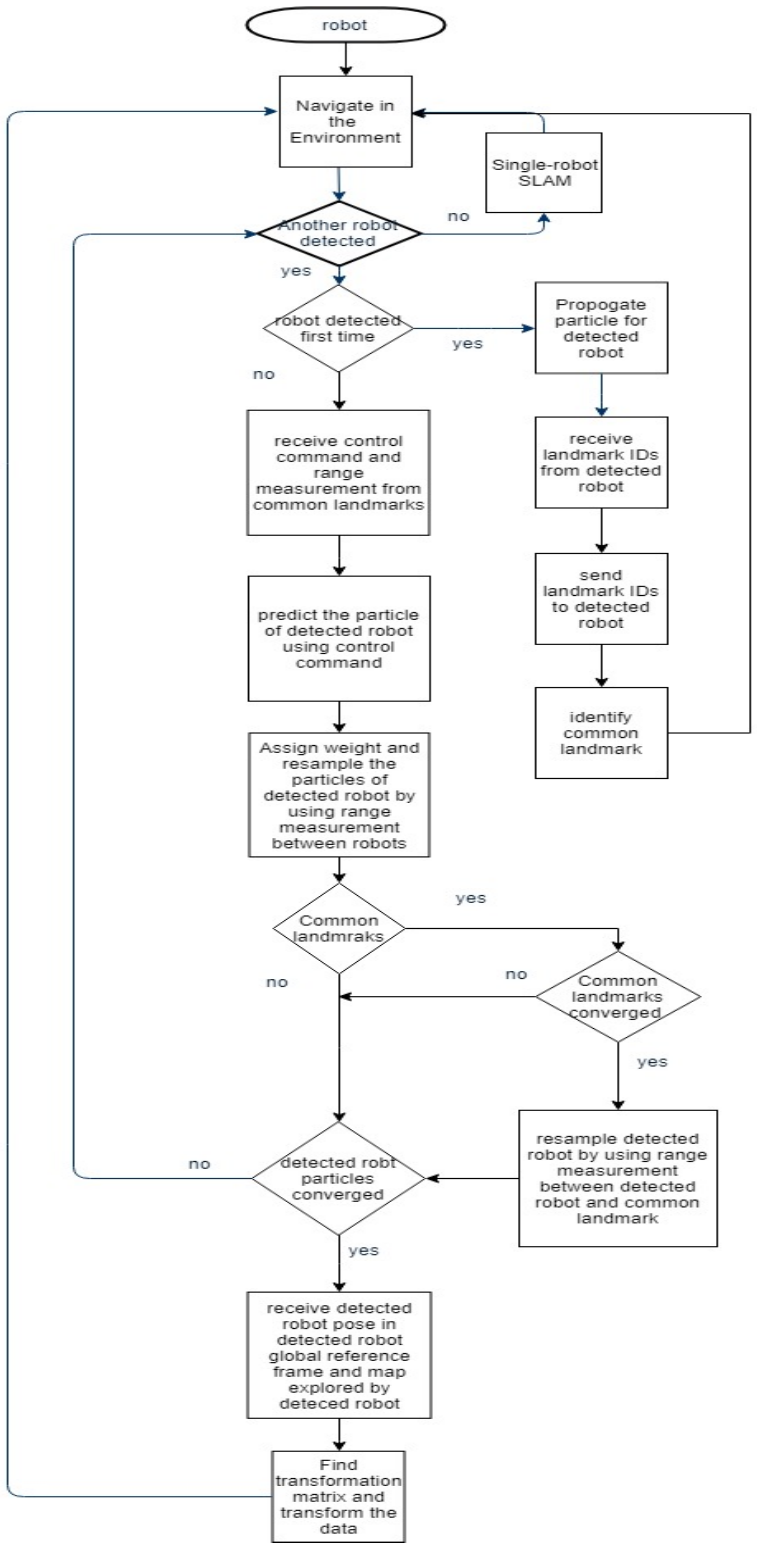

robot
Navigate in the Environment
Single-robot SLAM
Another robot detected
no
yes
robot detected first time
yes
no
Propogate particle for detected robot
receive control command and range measurement from common landmarks
receive landmark IDs from detected robot
send landmark IDs to detected robot
predict the particle of detected robot using control command
identify common landmark
Assign weight and resample the particles of detected robot by using range measurement between robots
yes
Common landmraks
no
no
Common landmarks converged
yes
no
detected robt particles converged
resample detected robot by using range measurement between detected robot and common landmark
yes
receive detected robot pose in detected robot global reference frame and map explored by deteced robot
Find transformation matrix and transform the data

Figure 4.5: Flowchart describing the implementation of multi-robot SLAM

4.4.3 Algorithm

Independently each robot executes Algorithm 3 to explore the map of the environment. Algorithm 4 shows the algorithm used to implement the multi-robot SLAM. Once rendezvous occurs, robot ID is added to detectedrobotID (line 3) and particles are initialized for the detected robot as shown in lines 4-7. The robot sends and receives all the landmark IDs known to the robot. The robot receives a control command from the detected robot (line 13) and predicts the position of particles of robots as shown in Lines 14-16. The robot finds the predicted range measurement between the mean of the particles of its position and detected robot particles. This prediction is subtracted by the measured data to find the weight. This weight is assigned to each particle of the detected robot and later used for resampling. Lines 18-25 depict the weight assignment and resampling of the detected robot with the help of robot range measurement. The robot also uses range measurement from a common landmark to update the particles of the robot. The robot receives a range measurement between detected robot and common landmark from detected robot. Lines 27-34 shows how range measurement received from a detected robot is used to update the particles of a detected robot. If a detected robot is converged then robot receives the number of landmarks detected robot have and transform data for all the non-common landmarks.

Algorithm 5 shows the algorithm used to transform the data. Let there be two robots *robot a* and *robot b*. Lets consider side of *robot a*. In this algorithm, robotPose is the pose of *robot b* in *robot b* global reference frame and teamRobotPose is pose of *robot b* in *robot a* global reference frame. First *robot a* transform the landmark coordinate from *robot b* global reference frame to *robot b* local reference frame. Line 3 shows the transnational matrix used and Line 5 shows the rotational matrix for coordinate frame transformation from *robot b* global reference frame to *robot b* local reference frame. Final transformation matrix is found by multiplying rotational matrix to

Algorithm 4 Multi-robot range-only SLAM implementation

```
1: if (!detecedRobotID.empty()) then
2:     if (new robot is detected) then
3:         detecedRobotID = [detecedRobotID, newRobotID]
4:         Initialization
5:         for $t = 1$ to number of particles do
6:             $X_{(t)}(N + 1 + length(detectedRobotID)$ = particles distributed
7:         end for
8:         receive landmark IDs from detected robot
9:         send landmark IDs to detected robot
10:    end if
11:    for $i = 1$ to length(detectedRobotID) do
12:        Prediction
13:        receive control command $(u_t)$ for detectedRobotId[i]
14:        for $t = 1$ to number of particles do
15:            $x_t^{[N+1+i]} \approx p(x_t^i | u_t, x_{t-1}^{[N+1+i]})$
16:        end for
17:        Weight Assignment and Resampling
18:        for $t = 1$ to number of particles do
19:            $w_t^{[n+1+i]} = p(z_t | X_t^{[N+1+i]}, mean(X_t^{[n+1+i]}))$
20:            $\bar{X}_t^{[N+1+i]} = \bar{X}_t^{[N+1+i]} + < X_t^{[N+1+i]}, w_t^{[n+1+i]} >$
21:        end for
22:        for $t = 1$ to $T$ do
23:            draw $i$ with probability $\approx w_t^{[i]}$
24:            add $x_t^{[i]}$ to $X_t^{[n]}$
25:        end for
26:        if (common Landmark) then
27:            receive range measurement from detectedrobotId[i] to landmark
28:            for $t = 1$ to number of particles do
29:                $\bar{X}_t^{[N+1+i]} = \bar{X}_t^{[N+1+i]} + < X_t^{[N+1+i]}, w_t^{[n]} >$
30:            end for
31:            for $t = 1$ to $T$ do
32:                draw $i$ with probability $\approx w_t^{[i]}$
33:                add $x_t^{[i]}$ to $X_t^{[n]}$
34:            end for
35:        end if
36:        if detectedRobotId[i] is converged for first time then
37:            Receive number of landmarks nl
38:            Receive position(robotPose) of detectedRobotId[i]
39:            for li = 1 to nl do
40:                if li is not a common landmark then
41:                    Transform data
42:                end if
43:            end for
44:            Add landmark particles to existing state
45:        end if
46:    end for
47: else
48:     Single robot SLAM
49: end if
```

transnational matrix. This matrix is used to transform data to *robot b* local reference frame as shown in Line 6. Once we have data in *robot b* local reference frame, the transformation matrix is found to transform data from *robot b* local reference frame to *robot a* global reference frame. This is done by finding the transnational and rotational matrix as shown in lines 7-9. Here transnational matrix is multiplied by rotation matrix to find the final transformation matrix. This transformation matrix is applied to landmark coordinate of *robot b* local reference frame to get landmark in *robot a* global reference frame. This is performed for each particles of landmark in *robot b* global frame to each particles of *robot b* in *robot a* global reference frame.

Algorithm 5 Transformation Matrix (robotPose, teamRobotPose)

```
1: $X_t = \emptyset$
2: for li = 1 to number of particles do
3:     translationMatrix =
           $\begin{vmatrix} 1 & 0 & -robotPose[li,1] \\ 0 & 1 & -robotPose[li,2] \\ 0 & 0 & 1 \end{vmatrix}$
4:     $\theta$ = -robotPose[li,3]
5:     rotationMatrix =
           $\begin{vmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix}$
6:     landmark in local frame = (rotationMatrix*translationMatrix)*landmark[i]
7:     translationMatrix =
           $\begin{vmatrix} 1 & 0 & -teamrobotPose[li,1] \\ 0 & 1 & -teamrobotPose[li,2] \\ 0 & 0 & 1 \end{vmatrix}$
8:     $\theta$ = teamrobotPose[li,3]
9:     rotationMatrix =
           $\begin{vmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix}$
10:    landmark in global frame = (translationMatrix*rotationMatrix)*(landmark in local frame)
11:    $X_t = X_t$+(landmark in global frame)
12: end for
13: return $X_t$
```

CHAPTER 5: RESULTS

This chapter present different results on simulation with the help of two robots named *robot A* and *robot B*. The results presented here are with different noise and different scenarios. Different scenarios include robots having no common landmark, one common landmark and robots exploring full map after the data transfer. Noise includes both Gaussian and non-Gaussian noise with and without zero means.

The result images are divided into two columns. The left column indicates *robot A*'s global reference frame and right section indicates *robot B*'s global reference frame. The blue circles indicate *robot A*; brown circles indicate *robot B*; yellow, red, and green circles indicate particles for landmarks explored by *robot A*; and blue, magenta, cyan indicates particles for landmarks explored by *robot B*. The silver circles indicate particles of a common landmark. The black dot indicates ground truth.

Three types of noise are used in these results: two are Gaussian noise with 0 mean and 0.05 m and 0.5 m co-variance respectively, the third one is a non-Gaussian noise. It was found that the error in the convergence of landmark, as well as robot particles, were within the range of 0.1 m to 0.8 m. The noise used for the implementation is shown below:

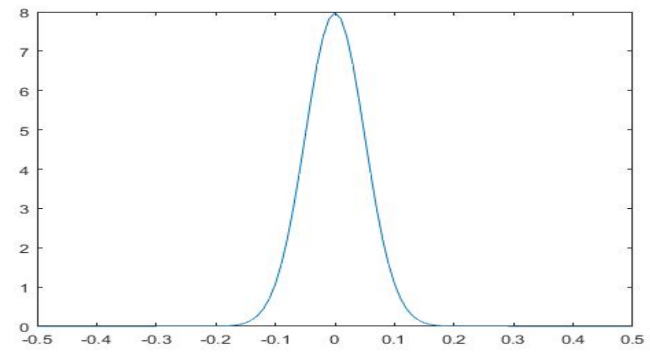


Figure 5.1: Gaussian noise used with zero mean and 0.05 variance

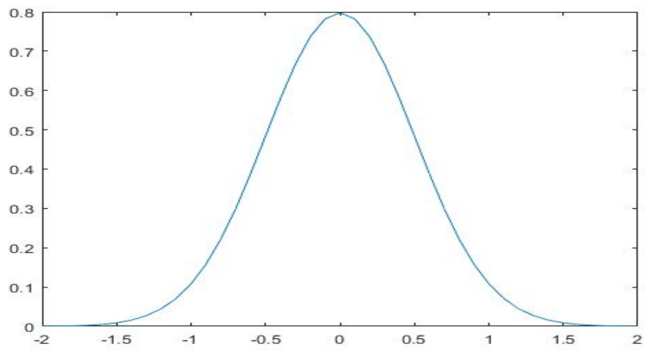


Figure 5.2: Gaussian noise used with zero mean and 0.5 variance

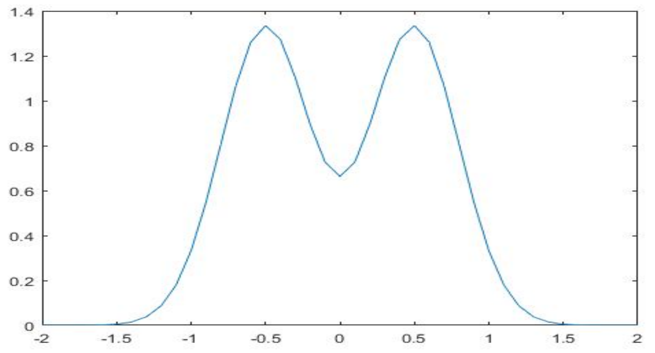
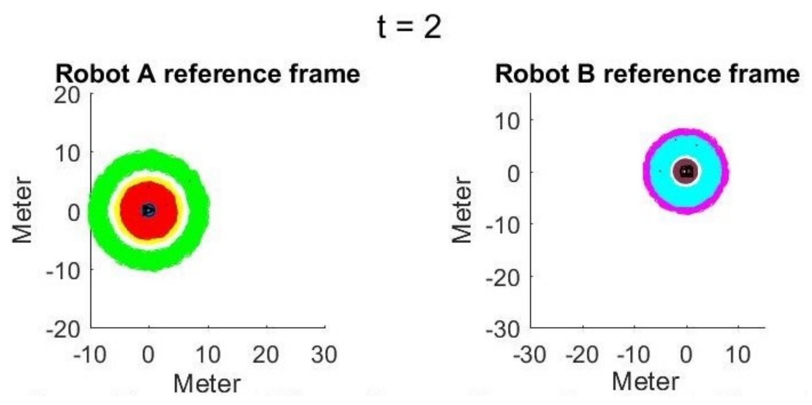


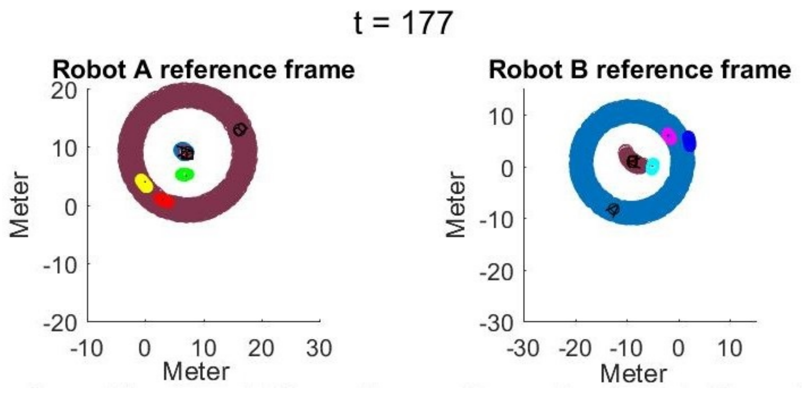
Figure 5.3: Non-Gaussian noise with non-zero mean

The results are divided into three sections. The first section covers the case where there are no common landmarks. The second section covers the case with one common landmark. The third case covers the result when each robot explores a full map after receiving a map from a detected robot. In the first two cases, the variance of the transferred map is huge because of the reference frame transformation involved. However, the mean was still found to be close to the ground truth. The third case shows that when a robot tries to move to the map of the detected robot, they successfully converge the particles of landmark and the variance is reduced. All of the section consists of the same environment and two robots exploring the map with the noise described above.

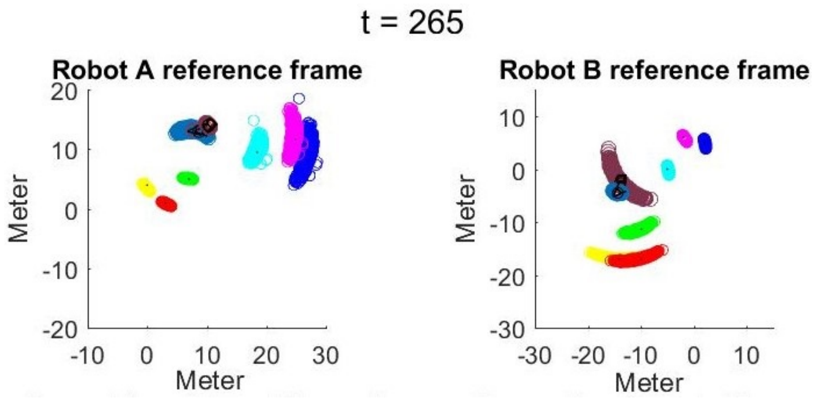
Figure 5.4 shows different stages of robot navigation. Figure 5.4(a) indicates an initial phase of navigation at time t = 2. In this phase, both *Robot A* and *Robot B* encountered three different landmarks. The blue circles in the left column show *Robot A* particles propagated around (0,0) in *Robot A*'s global reference frame. The *Robot A* propagate particles for the detected landmarks in a ring shape as explained in the implementation section of chapter 4. The yellow, red, and green circles indicate particles for landmark 1, landmark 2, and landmark 3 detected by *Robot A* respectively. The brown circles in the right column indicate *Robot B* particles propagated around (0,0) in *Robot B*'s global reference frame. The blue, magenta, and cyan circles indicate particles of detected landmarks propagated in a ring shape. Since both robots do not know the initial position of each other and they don't share any common global map, the reference frame for both robots is different. The *Robot A* and *Robot B* implements single-robot SLAM as explained in implementation section of chapter 3 till time t = 177 until rendezvous occurs. Figure 5.4(b) shows a rendezvous event. The brown circles in the left column show particles for *Robot B* propagated in a ring shape as *Robot A* sense only range measurement from *Robot B*. After this time, multi-robot SLAM is implemented by both robots to localize detected robot in its own map. Once localized, each robot receives map explored by another robot and apply a transformation matrix to merge map. Figure 5.4(c) shows the final stage of the map after map merge. Figure 5.5 shows the ground truth of robots. Figure 5.5(a) shows ground truth for both *Robot A* and *Robot B* global reference frame. Figure 5.5(b) shows the orientation of *Robot B*'s global reference frame with respect to *Robot A* global reference frame. The table 5.1 shows the actual landmarks locations and observed landmarks locations for both robots in their reference frames. The table also shows the error in landmark convergence in the meter.



(a)

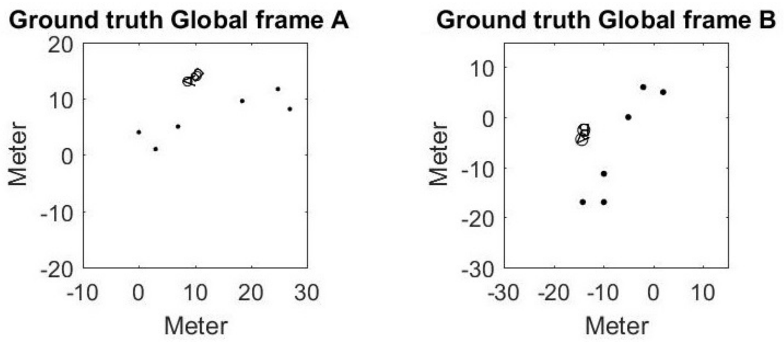


(b)

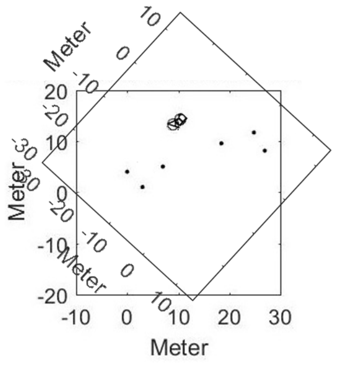


(c)

Figure 5.4: Robot navigation stages: (a) Initial; (b)Rendezvous; (c) Map merge



(a)



(b)

Figure 5.5: Ground truth of robot navigation: (a) Ground truth; (b) Map correspondence

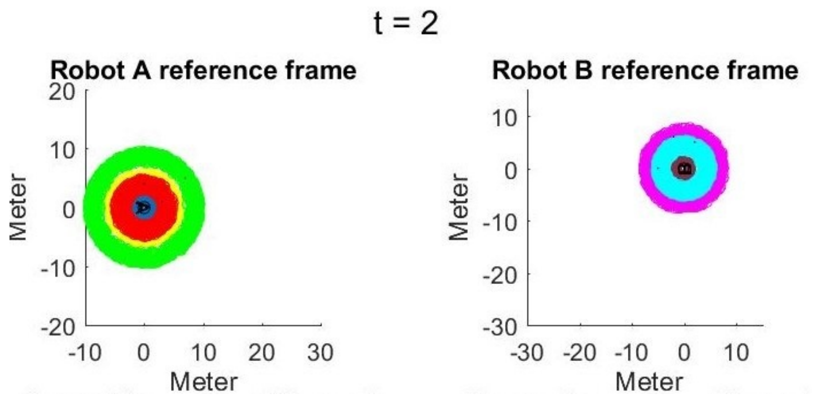
Table 5.1: Ground truth and observation for multi-robot SLAM

Reference fame B landmark coordinate					Reference fame A landmark coordinate				
Ground truth		Observed by robot		Error	Ground truth		Observed by robot		Error
x	y	x	y	meter	x	y	x	y	meter
2	5	2.230	4.588	0.2090	0	4	-0.202	4.120	0.0900
-2	6	-1.929	5.891	0.1670	3	1	3.140	1.390	0.2960
-5	0	-4.638	-0.086	0.3930	7	5	7.120	4.751	0.1180
-14.442	-16.970	-14.445	-16.459	0.2190	26.949	8.121	26.134	8.96	0.2380
-9.899	-16.970	-10.261	-16.529	0.4900	24.828	11.656	24.278	11.350	0.4620
-9.899	-11.313	-9.825	-11.374	0.3150	18.464	9.535	18.650	9.92	0.3530

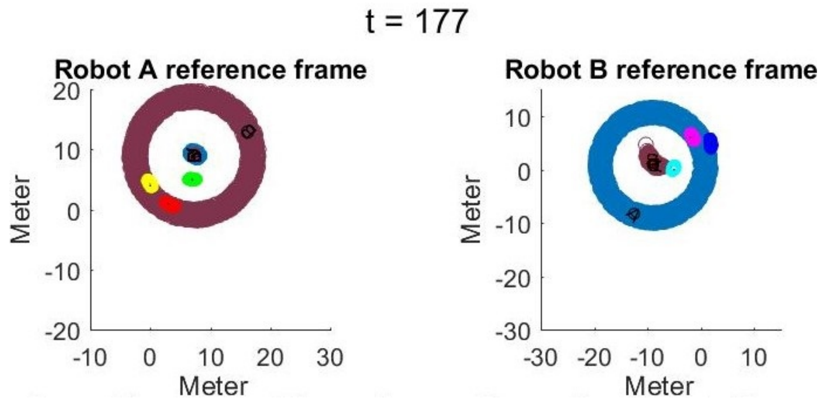
The results are divided into three sections as explained above. All the section contains the same map like the one shown in Figure 5.4. The result in all three scenarios as displayed in the same manner as explained above. All the results contain three stages of navigation: Initial, Rendezvous and map merge. The observed and ground truth of all landmarks is also displayed in the table for each test cases.

5.1 Case 1: No Common landmark

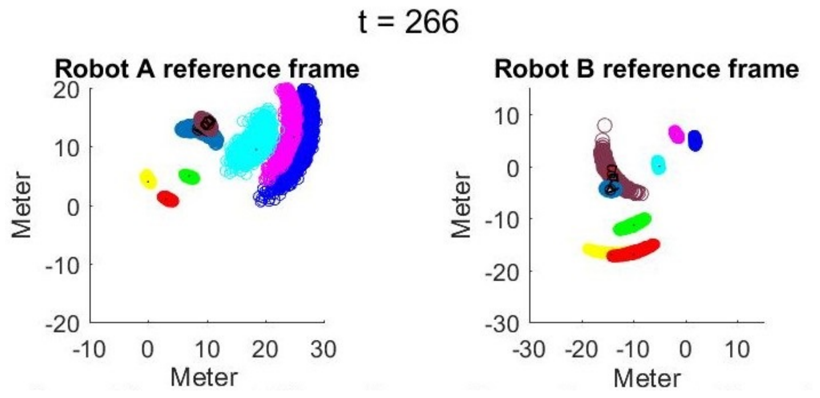
In this case, there is no common landmark between two robots. This increases the time of localizing detected robot. Figures 5.6 and 5.7 shows results and ground truth with Gaussian noise of zero means and 0.05 m variance respectively. Figures 5.8 and 5.9 shows the result and ground truth with Gaussian noise of zero mean and 0.5 m variance respectively. Figures 5.10 and 5.11 shows result and ground truths of map merging with non-Gaussian noise respectively. Tables 5.2, 5.3 and 5.4 shows the comparison of ground truth and observation of *Robot A* and *Robot B*.



(a)

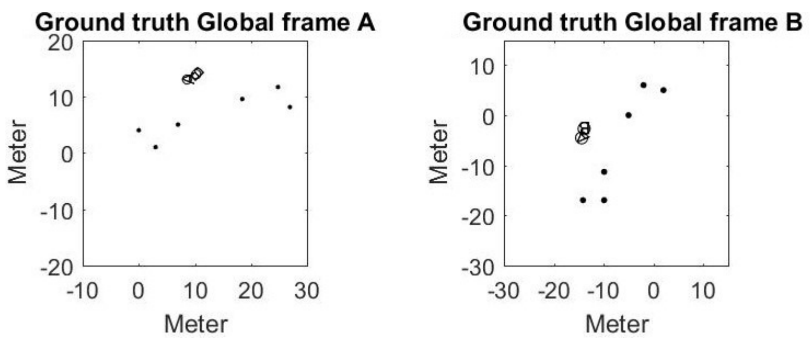


(b)

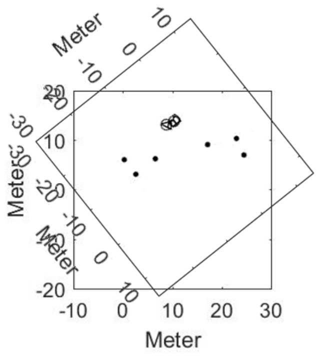


(c)

Figure 5.6: Robot Navigation stages for no common landmark case with Gaussian noise of zero mean and 0.05 m variance: (a) Initial; (b)Rendezvous; (c) Map merge



(a)

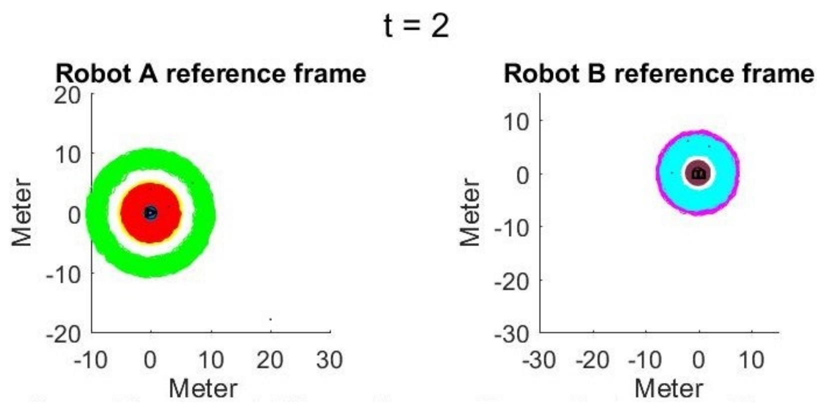


(b)

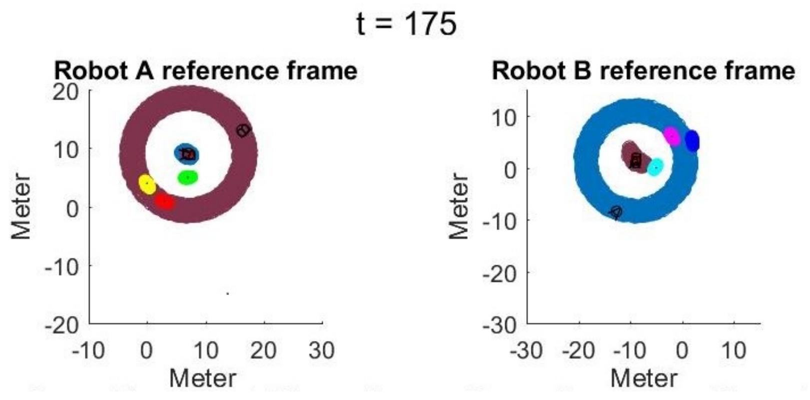
Figure 5.7: Ground truth for Gaussian noise of zero mean and 0.05 variance: (a) Ground truth; (b) Map correspondence

Table 5.2: Ground truth and observation for no common landmark and 0.05 m Gaussian noise

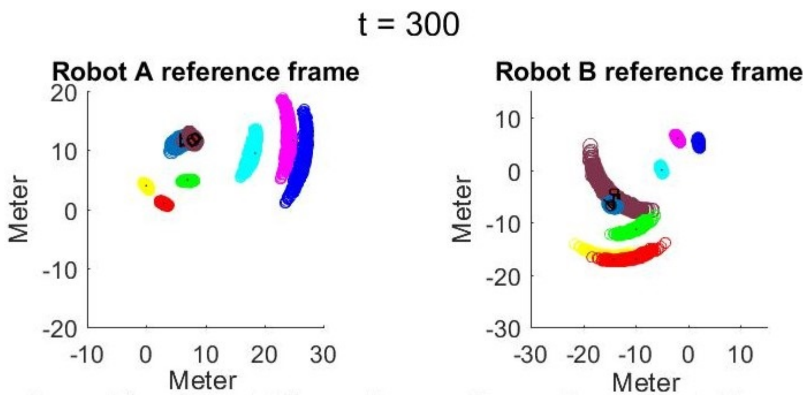
Reference fame B landmark coordinate					Reference fame A landmark coordinate				
Ground truth		Observed by robot		Error	Ground truth		Observed by robot		Error
x	y	x	y	meter	x	y	x	y	meter
2	5	1.8740	4.9840	0.1270	0	4	0.0150	4.4290	0.4290
-2	6	-1.7070	6.0840	0.3050	3	1	3.4420	1.0340	0.4440
-5	0	-5.1520	0.1880	0.2420	7	5	6.9920	4.8680	0.1320
-14.442	-16.970	-14.8410	-16.6050	0.4890	26.949	8.121	26.2580	8.1460	0.6915
-9.899	-16.970	-10.0440	-16.5690	0.4270	24.828	11.656	24.6660	11.9470	0.3331
-9.899	-11.313	-10.2580	-11.3780	0.3640	18.464	9.535	18.1870	9.8250	0.401



(a)

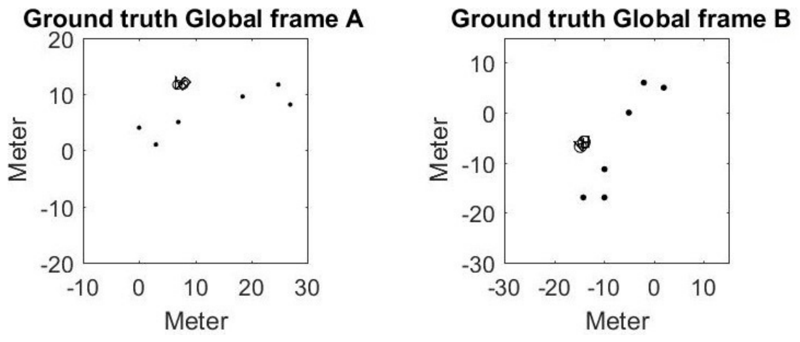


(b)

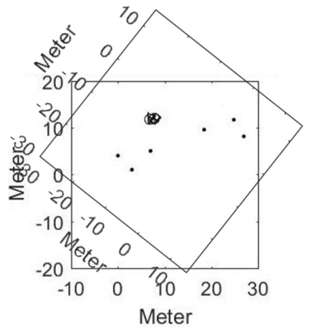


(c)

Figure 5.8: Robot Navigation stages for no common landmark case with Gaussian noise of zero mean and 0.5 m variance: (a) Initial; (b)Rendezvous; (c) Map merge



(a)

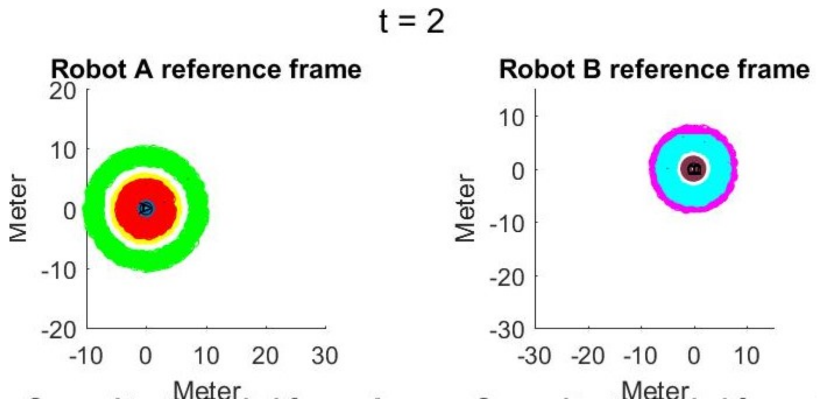


(b)

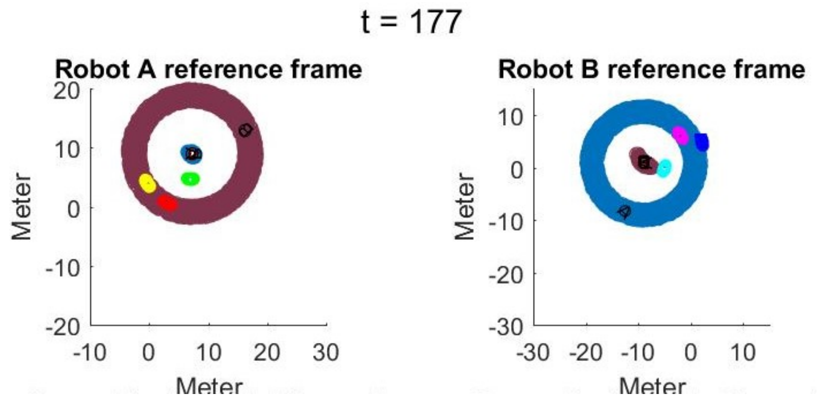
Figure 5.9: Ground truth for Gaussian noise of zero mean and 0.5 variance: (a) Ground truth; (b) Map correspondence

Table 5.3: Ground truth and observation for no common landmark and 0.5 m Gaussian noise

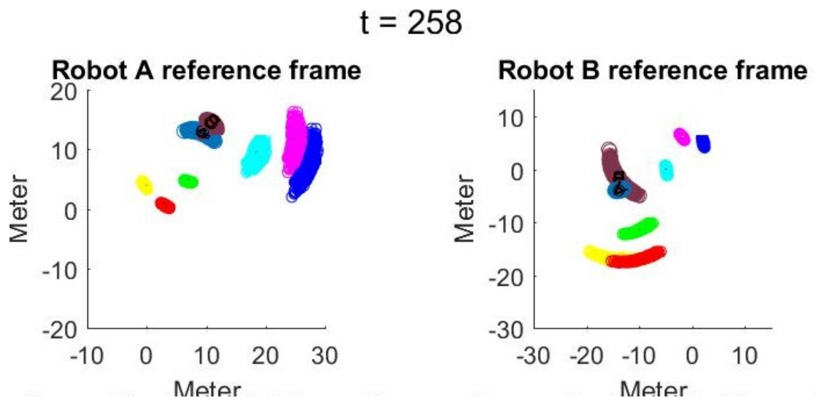
Reference fame B landmark coordinate					Reference fame A landmark coordinate				
Ground truth		Observed by robot		Error	Ground truth		Observed by robot		Error
x	y	x	y	meter	x	y	x	y	meter
2	5	2.2310	4.5890	0.4720	0	4	0.0680	3.9290	0.0980
-2	6	-1.9290	5.8910	0.1300	3	1	3.0540	0.9760	0.0590
-5	0	-4.3380	-0.0860	0.6670	7	5	7.0630	4.8200	0.1910
-14.442	-16.970	-14.4450	-16.4590	0.5940	26.949	8.121	26.4470	8.1100	0.5030
-9.899	-16.970	-10.3620	-16.5300	0.6390	24.828	11.656	24.2270	11.4760	0.6240
-9.899	-11.313	-9.8250	-11.3750	0.0960	18.464	9.535	17.9470	9.0360	0.6940



(a)

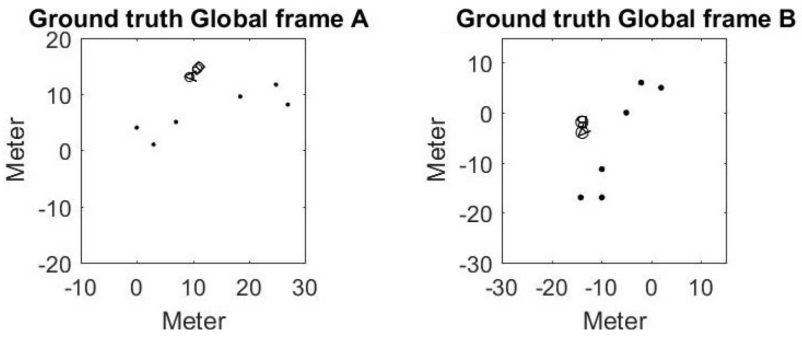


(b)

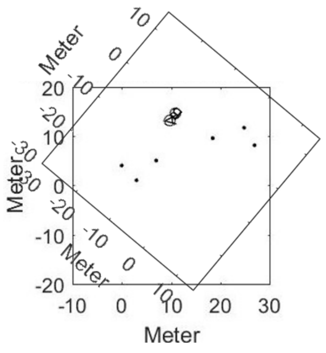


(c)

Figure 5.10: Robot Navigation stages with non-gaussian noise: (a) Initial; (b)Rendezvous; (c) Map merge



(a)



(b)

Figure 5.11: Ground truth for non-gaussian noise: (a) Ground truth; (b) Map correspondence

Table 5.4: Ground truth and observation for no common landmark and non-Gaussian noise

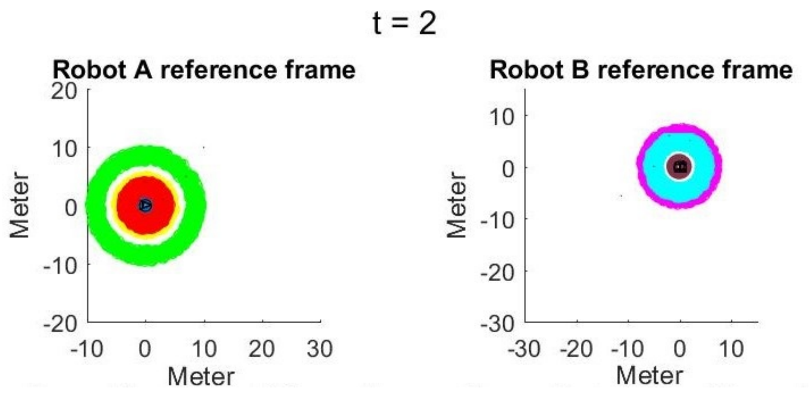
Reference fame B landmark coordinate				
Ground truth		Observed by robot		Error
x	y	x	y	meter
2	5	2.1960	5.0700	0.2090
-2	6	-1.7630	6.1230	0.2670
-5	0	-4.6160	-0.0870	0.3930
-14.442	-16.970	-15.008	-16.7520	0.8190
-9.899	-16.970	-10.5860	-17.0490	0.6900
-9.899	-11.313	-10.1550	-11.4980	0.3150

Reference fame A landmark coordinate				
Ground truth		Observed by robot		Error
x	y	x	y	meter
0	4	-0.3800	3.9120	0.3900
3	1	3.1240	0.8490	0.1960
7	5	6.8920	5.0480	0.1180
26.949	8.121	26.3410	8.3120	0.6380
24.828	11.656	24.3680	11.7000	0.4620
18.464	9.535	18.2400	9.4180	0.2530

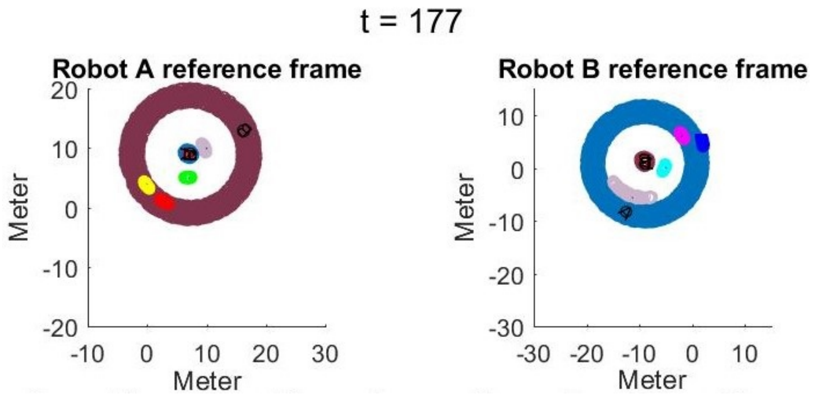
The tables 5.2, 5.3 and 5.4 shows the error in convergence along with the observation of robots. As shown in tables, the error of convergence for landmarks is less when the noise is Gaussian with 0 mean and 0.05 m variance compared to Gaussian noise of 0 mean and 0.5 m variance. The noise of landmark convergence is increased in third noise which is non-Gaussian non zero mean noise. However, the maximum error in converge of landmarks with non-Gaussian noise was 0.8196 m. The average of errors of landmark convergence in case of noise with 0.05 m variance is 0.365, while in case of Gaussian noise with 0.5 m variance it is 0.415 and in case of non-Gaussian noise with non zero mean it is 0.485 m. This shows that the average error of the algorithm even in case of the non-Gaussian nose was within 0.5 m.

5.2 Case 2: Common landmark

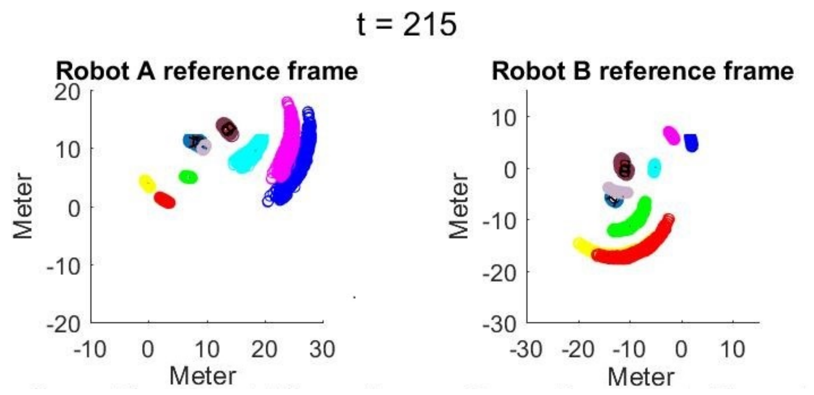
In this case, there is one common landmark between two robots. It was observed that the localization of the detected robot took less time because of a common landmark as a common landmark also contributes to localizing the detected robot. Figures 5.12 and 5.13 shows result and ground truth with Gaussian noise of zero mean and 0.5 m variance respectively. Figures 5.14 and 5.15 shows result and ground truth of map merging with non-Gaussian noise respectively. Table 5.6, and 5.6 shows the comparison of ground truth and observation of *Robot A* and *Robot B*.



(a)

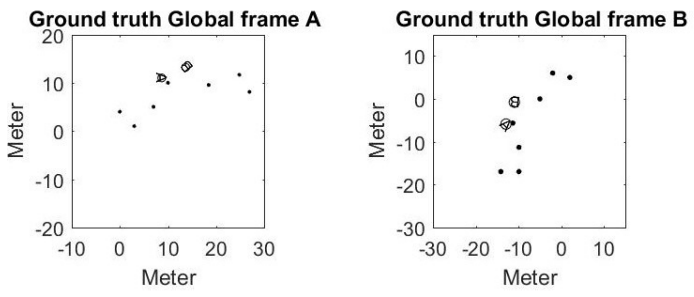


(b)

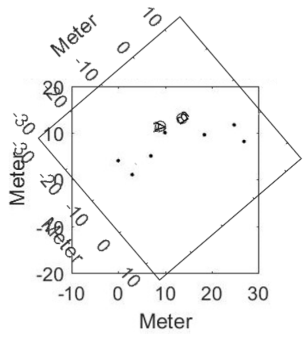


(c)

Figure 5.12: Robot Navigation stages for common landmark with noise of zero mean and 0.5 variance: (a) Initial; (b)Rendezvous; (c) Map merge



(a)

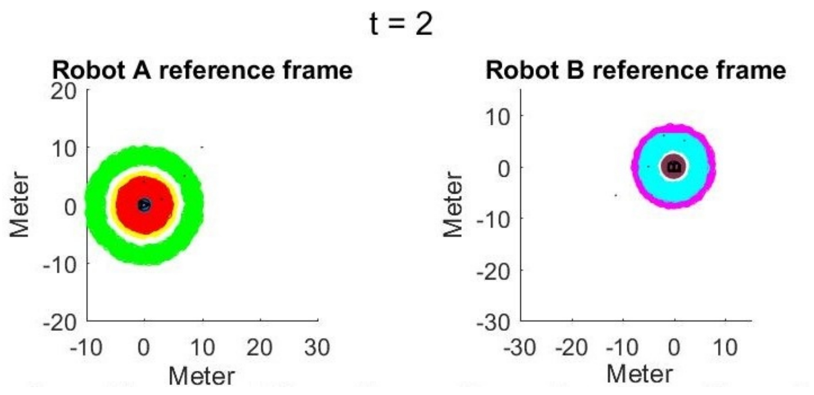


(b)

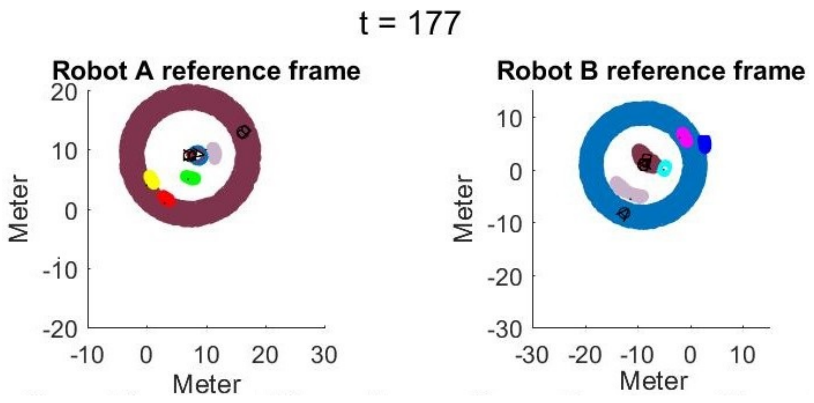
Figure 5.13: Ground truth for common landmark with Gaussian noise of zero mean and 0.05 variance: (b) Ground truth; (b) Map correspondence

Table 5.5: Ground truth and observation for common landmark and 0.5 m Gaussian noise

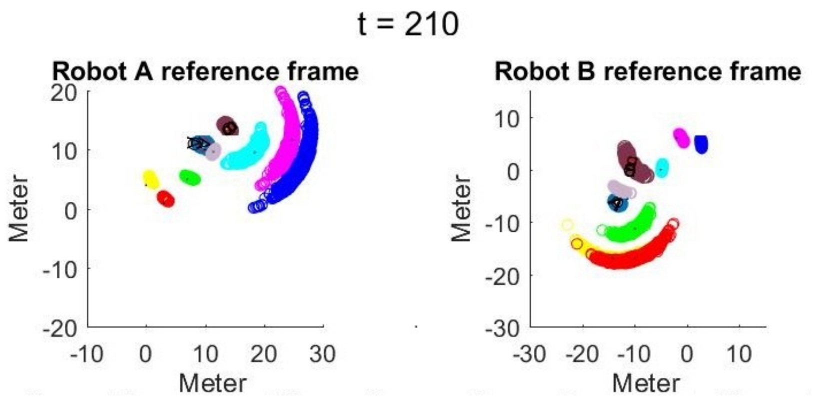
Reference fame B landmark coordinate					Reference fame A landmark coordinate				
Ground truth		Observed by robot		Error	Ground truth		Observed by robot		Error
x	y	x	y	meter	x	y	x	y	meter
2.0000	5.0000	1.9680	4.9900	0.0340	0	4.0000	-0.1730	3.8410	0.2350
-2.0000	6.0000	-1.8230	6.1900	0.2590	3.0000	1.0000	2.8690	0.9760	0.1340
-5.0000	0	-5.1640	0.0160	0.1640	7.0000	5.0000	6.7780	5.0460	0.2270
-11.3140	-5.6570	-11.910	-4.9300	0.69	10.0000	10.0000	9.6200	10.1160	0.3970
-14.1420	-16.9710	-13.9480	-16.6820	0.310	26.9500	8.1210	26.0860	7.8920	0.8930
-9.8990	-16.9710	-9.1210	-16.2540	0.712	24.8280	11.6570	24.2630	11.3160	0.6600
-9.8990	-11.3140	-9.4470	-10.9100	0.6780	18.4640	9.5360	17.7310	9.3050	0.7690



(a)

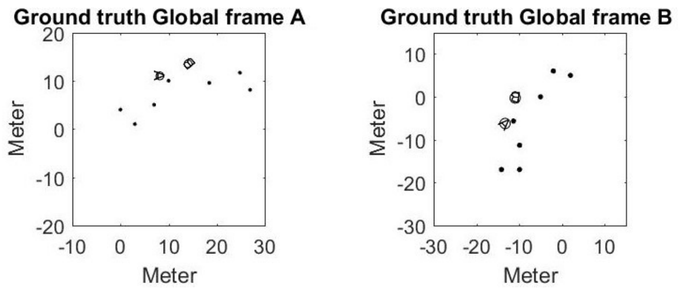


(b)

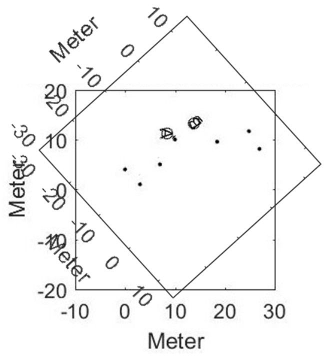


(c)

Figure 5.14: Robot Navigation stages for common landmark with non-Gaussian noise: (a) Initial; (b)Rendezvous; (c) Map merge



(a)



(b)

Figure 5.15: Ground truth for common landmark with non-Gaussian noise: (a) Ground truth; (b) Map correspondence

Table 5.6: Ground truth and observation for common landmark and non-Gaussian noise

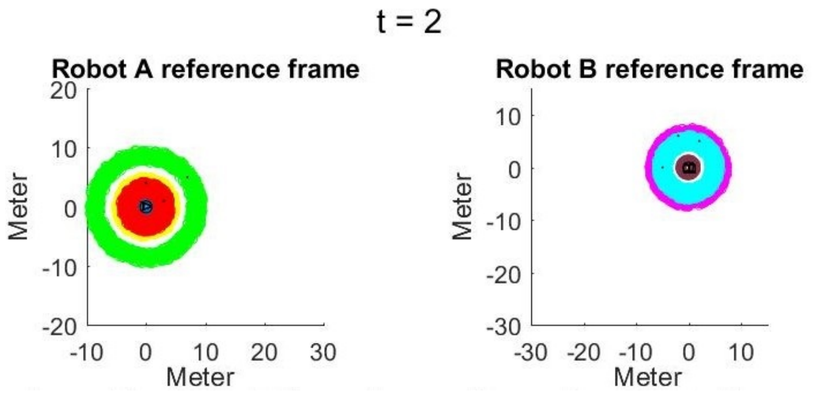
Reference fame B landmark coordinate				
Ground truth		Observed by robot		Error
x	y	x	y	meter
2.00	5.0000	1.9680	4.9900	0.7600
-2.00	6.000	-1.8230	6.1900	0.9900
-5.00	0	-5.1640	0.0160	0.3800
-11.3140	-5.6570	-11.910	-4.1300	0.713
-14.1420	-16.9710	-13.9480	-16.6820	0.4500
-9.8990	-16.9710	-9.1210	-16.2540	0.3400
-9.89900	-11.3140	-9.4470	-10.9100	0.2500

Reference fame A landmark coordinate				
Ground truth		Observed by robot		Error
x	y	x	y	meter
0	4.0000	0.4200	4.9400	0.321
3.0000	1.0000	3.2900	1.7100	0.7670
7.0000	5.0000	7.5200	5.1900	0.5570
10.0000	10.0000	10.4300	9.4900	0.6170
26.9500	8.1210	26.4000	8.0900	0.5510
24.8280	11.6570	24.3400	11.3300	0.5920
18.4640	9.5360	17.8800	9.3900	0.6060

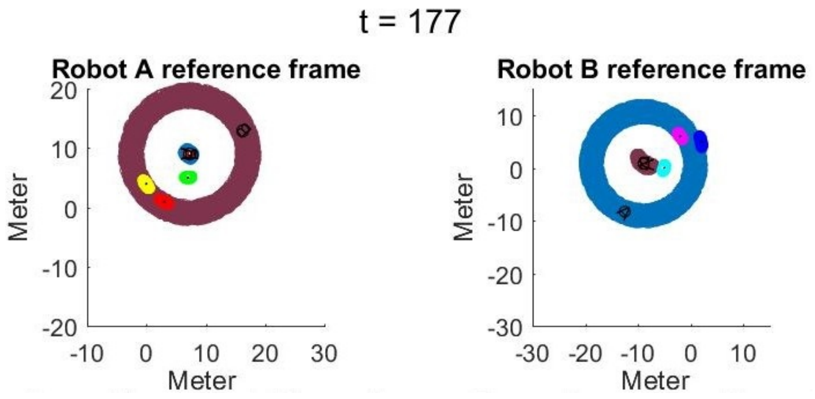
The tables 5.5 and 5.6 shows the error in convergence along with the observation of robots. As shown in tables, the error of convergence for landmarks is less for Gaussian noise of 0 mean and 0.5 m variance compared to non-Gaussian noise. However, the maximum error in converge of landmarks with non-Gaussian noise was 0.99 m. The average of errors of landmark convergence in case of noise with 0.5 m variance is 0.415 and in case of non-Gaussian noise with non zero mean it is 0.563 m. This shows that the average error of the algorithm even in case of the non-Gaussian nose was within 0.6 m.

5.3 Case 3: Full exploration by both robots

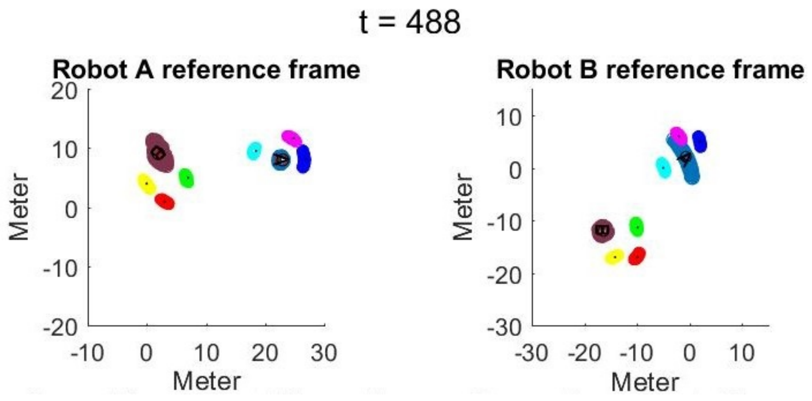
In this case, both robots explore the environment even after receiving a map from a detected robot. It was observed that the variance of landmark particle was reduced and the robot was able to converge the particle very well. Figures 5.16 and 5.17 shows result and ground truth with non-Gaussian noise. Table 5.7 shows the comparison of ground truth and observation of *Robot A* and *Robot B*.



(a)

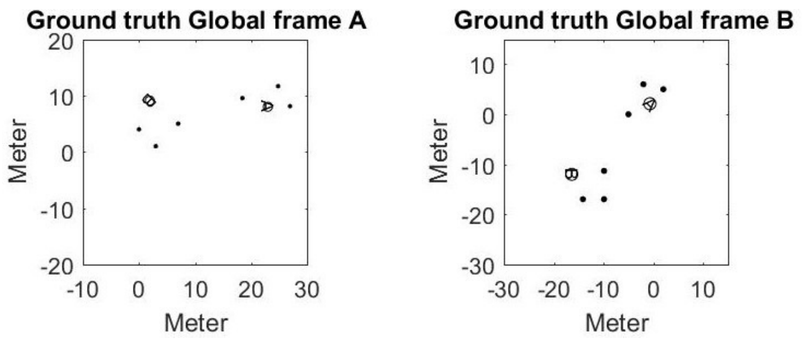


(b)

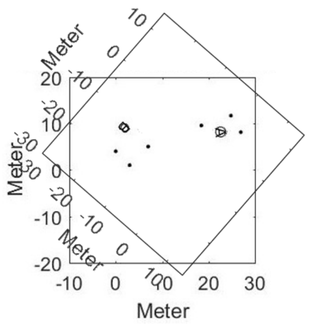


(c)

Figure 5.16: Robot Navigation stages for full exploration with Gaussian noise of zero mean and 0.5 variance: (a) Initial; (b)Rendezvous; (c) Map merge



(a)



(b)

Figure 5.17: Ground truth for full exploration with non-Gaussian noise: (a) Ground truth; (b) Map correspondence

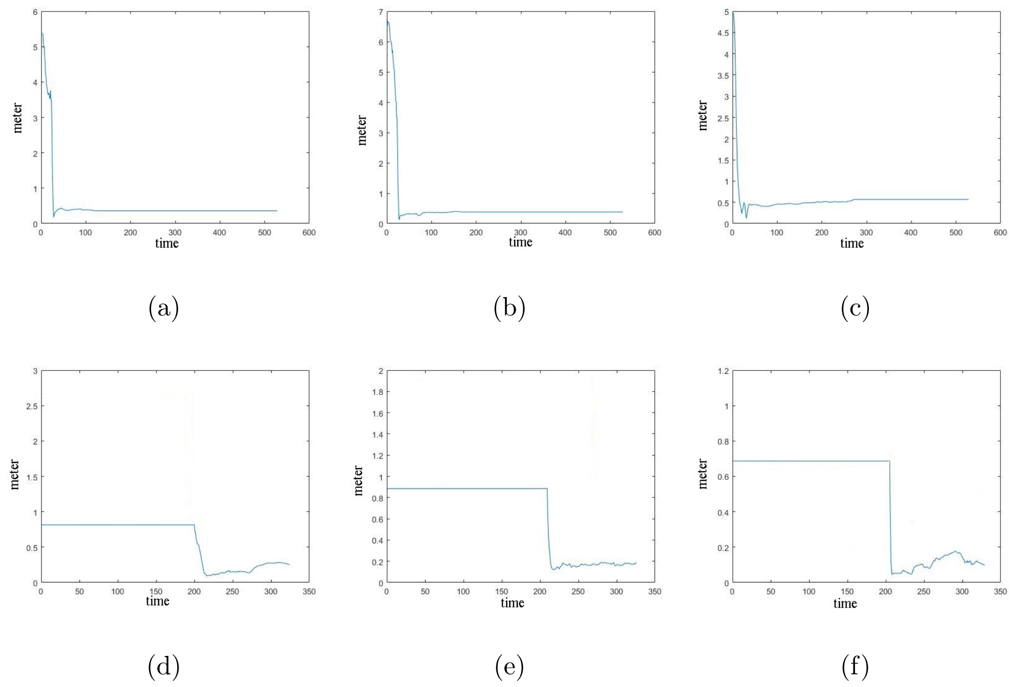
Table 5.7: Ground truth and observation for full exploration and non-Gaussian noise

Reference fame B landmark coordinate					Reference fame A landmark coordinate				
Ground truth		Observed by robot		Error	Ground truth		Observed by robot		Error
x	y	x	y	meter	x	y	x	y	meter
2.0000	5.0000	1.9680	4.9900	0.0100	0	4.0000	-0.0300	3.8900	0.1100
-2.0000	6.0000	-1.8230	6.1900	0.0600	3.0000	1.0000	3.0700	0.9400	0.0900
-5.0000	0	-5.1640	0.0160	0.1400	7.0000	5.0000	6.7100	4.8600	0.3200
-11.3140	-5.6570	-11.910	-4.1300	0.2200	10.0000	10.0000	26.6000	7.9500	0.3900
-9.9000	-16.9700	-13.9480	-16.6820	0.1400	26.9500	8.1210	24.4100	11.7100	0.4200
-9.9000	-11.3100	-9.1210	-11.2540	0.1800	24.8280	11.6570	18.1400	9.5200	0.3200

the table 5.7 shows the error for each landmark convergence in both *Robot A* and *Robot B* reference frame. This result shows that the maximum convergence error was

0.390 and the average error of all particle convergence was 0.2 m.

Figure 5.18 shows the graph of landmark estimation error over time for full exploration. This error is the difference between the mean of landmark particles and the actual location of the landmark. Figures 5.18(a), 5.18(b) and 5.18(c) shows the graph of error estimation for landmark 1, landmark 2 and landmark 3 explored by *robot A* respectively. This error starts with huge number but eventually reduced to a value below 1 m. Figures 5.18(d), 5.18(e) and 5.18(f) shows the graph of error estimation for landmark 4, landmark 5 and landmark 6 respectively. This error starts with a small number because these are the landmarks transferred by *robot B* to *robot A*. This error reduces as *robot A* try to converge the particles of landmarks.



(a) (b) (c)

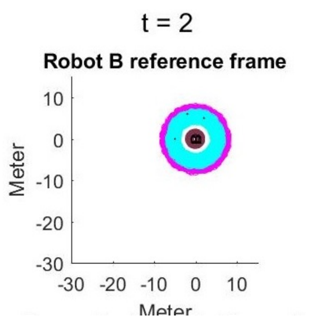
(d) (e) (f)

Figure 5.18: Landmark error over time for Full exploration: (a)Landmark 1 error; (b)Landmark 2 error; (c)Landmark 3 error; (d)Landmark 4 error; (e)Landmark 5 error; (f)Landmark 6 error;

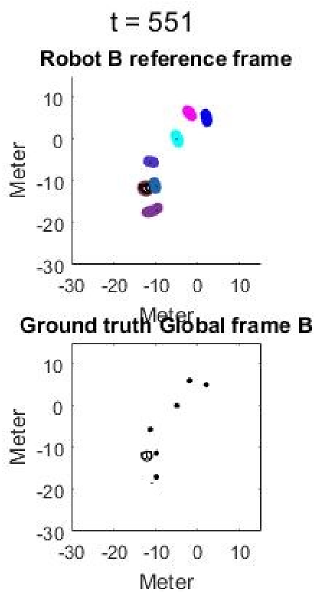
This result shows that the convergence of error was within 1 m for all the scenarios shown in this paper. It was also observed that the increase in variance and error due to reference frame transformation can be overcome if a robot tries to converge the particles of landmarks after map merge.

5.4 Comparison with single-robot SLAM

This section shows the result of single-robot SLAM with multi-robot SLAM for the same environment. Figure 5.19 shows the result and ground truth with single-robot SLAM. figure 5.19(a) shows the initial stage of navigation with a single robot SLAM. Figure 5.19(b) shows the final stage and ground truth for single-robot SLAM implementation. It is observed that the time taken for single-robot to map the environment with the same path has been used in multi-robot SLAM by *Robot B* is huge. The *Robot B* alone takes 551 seconds to map the environment while Figure 5.14 shows it took 210 seconds for two robots to map the environment.



(a)



(b)

Figure 5.19: Robot Navigation stages for full exploration with Gaussian noise of zero mean and 0.5 variance: (a) Initial; (c) Final

The table 5.8 shows the result with single-robot SLAM. It is observed that the error of single-robot SLAM is almost equivalent to multi-robot SLAM. The average of all error with single-robot SLAM is 0.3 m. This result shows that multiple robots are capable of estimating a map of the environment in less than half of the time then single-robot with a comparable error of convergence

Table 5.8: Ground truth and observation for single-robot SLAM

Reference fame B landmark coordinate				
Ground truth		Observed by robot		Error
x	y	x	y	meter
2.0000	5.0000	2.2631	4.9862	0.1646
-2.0000	6.0000	-1.8619	6.0552	0.1488
-5.0000	0	-4.8777	0.0466	0.1309
-11.3140	-5.6570	-11.1609	-5.4900	0.2263
-9.9000	-16.9700	-10.7223	-17.0890	0.8309
-9.9000	-11.3100	-10.1042	-11.1663	0.2497

CHAPTER 6: CONCLUSIONS

In this paper, many of the known issues with multi-robot SLAM and existing solutions to multi-robot SLAM are explored. Most of the methods use both range and bearing sensors in order to implement multi-robot SLAM. However, it is seen that range and bearing sensors like laser scanner consume a huge amount of memory. This can be a problem when it comes to exploring the large environment. Multi-robot SLAM is mostly used for exploring large environments as it is very hard for a single robot to explore large environments. Hence, there was a need to focus on reducing the memory consumption while exploring huge environments.

A novel approach has been proposed to create a map of the environment using multiple robots. The range-only SLAM was implemented with multiple robots. This approach was tested via simulation with different patterns of noise. This algorithm result was satisfactory even with non-zero non-Gaussian noise. The covariance of converged landmarks and detected robots was very small which lead to converging the map with less error. The final error of this apprach using simulation was below 0.6 m which is comparable to Kai [37] who received an error of 0.5 m. However, the environment used for implementation of this paper is different then that by other papers hence it is hard to have a direct comparison as the noise changes with the environment.

In conclusion, the performance of the new approach to multi-robot range-only SLAM was satisfactory. The average range of error for landmark convergence was 0.1 m to 1 m for extreme scenarios.

6.1 Future Work

Future work for this topic would be to implement the algorithm on actual robots and test the performance with ground truth.

Another line of research can be to move robots close to each other after detection. It is necessary for robots to localize other detected robots, which cannot be performed if robots move away from each other. This can be achieved by moving towards the increased signal strength.

Another enhancement can be coordinated exploration of the environment. It is important for robots to explore different environments rather then all robots moving towards the same environment for efficient exploration of the map.

REFERENCES

[1] L. M. Paz, J. D. Tardós, and J. Neira, "Divide and conquer: EKF SLAM in $o(n)$," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1107–1120, 2008.

[2] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based SLAM," *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007.

[3] M. Montemerlo and S. Thrun, "Fastslam 2.0," *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, pp. 63–90, 2007.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.* MIT press, 2005.

[5] J. Huh, W. S. Chung, S. Y. Nam, and W. K. Chung, "Mobile robot exploration in indoor environment using topological structure with invisible barcodes," *ETRI journal*, vol. 29, no. 2, pp. 189–200, 2007.

[6] J. Aulinas, Y. R. Petillot, J. Salvi, and X. Lladó, "The SLAM problem: a survey,," *CCIA*, vol. 184, no. 1, pp. 363–371, 2008.

[7] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[8] J.-L. Blanco, J. González, and J.-A. Fernández-Madrigal, "A pure probabilistic approach to range-only SLAM.," in *ICRA*, pp. 1436–1441, Citeseer, 2008.

[9] S. Shue and J. M. Conrad, "Procedurally generated environments for simulating RSSI-localization applications," in *Proceedings of the 20th Communications & Networking Symposium*, p. 7, Society for Computer Simulation International, 2017.

[10] D. Sun, A. Kleiner, and T. M. Wendt, "Multi-robot range-only SLAM by active sensor nodes for urban search and rescue," in *Robot Soccer World Cup*, pp. 318–330, Springer, 2008.

[11] J. W. Fenwick, P. M. Newman, and J. J. Leonard, "Cooperative concurrent mapping and localization," in *ICRA*, pp. 1810–1817, Citeseer, 2002.

[12] R. Madhavan, K. Fregene, and L. E. Parker, "Distributed cooperative outdoor multirobot localization and mapping," *Autonomous Robots*, vol. 17, no. 1, pp. 23–39, 2004.

[13] G. Kantor and S. Singh, "Preliminary results in range-only localization and mapping," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2, pp. 1818–1823, Ieee, 2002.

[14] A. Ward, A. Jones, and A. Hopper, "A new location technique for the active office," *IEEE Personal communications*, vol. 4, no. 5, pp. 42–47, 1997.

[15] Y. Wang, S. Goddard, and L. C. Perez, "A study on the cricket location-support system communication protocols," in *2007 IEEE international conference on electro/information technology*, pp. 257–262, IEEE, 2007.

[16] P. Bahl, V. N. Padmanabhan, *et al.*, "Radar: An in-building rf-based user location and tracking system," in *IEEE infocom*, vol. 2, pp. 775–784, INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 2000.

[17] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[18] P. Newman and J. Leonard, "Pure range-only sub-sea SLAM," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, pp. 1921–1926 vol.2, Sep. 2003.

[19] J. Djugash, S. Singh, and P. Corke, "Further results with localization and mapping using range from radio," in *Field and Service Robotics*, pp. 231–242, Springer, 2006.

[20] S. L. Shue, N. S. Shetty, A. F. Browne, and J. M. Conrad, "Particle filter approach to utilization of wireless signal strength for mobile robot localization in indoor environments,"

[21] R. Chellali *et al.*, "A distributed multi robot SLAM system for environment learning," in *2013 IEEE Workshop on Robotic Intelligence in Informationally Structured Space (RiiSS)*, pp. 82–88, IEEE, 2013.

[22] E. W. Nettleton, P. W. Gibbens, and H. F. Durrant-Whyte, "Closed form solutions to the multiple-platform simultaneous localization and map building (slam) problem," in *Sensor fusion: Architectures, algorithms, and applications IV*, vol. 4051, pp. 428–438, International Society for Optics and Photonics, 2000.

[23] S. Thrun and Y. Liu, "Multi-robot SLAM with sparse extended information filers," in *Robotics Research. The Eleventh International Symposium*, pp. 254–266, Springer, 2005.

[24] S. Thrun, "A probabilistic on-line mapping algorithm for teams of mobile robots," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.

[25] A. Howard, G. S. Sukhatme, and M. J. Mataric, "Multirobot simultaneous localization and mapping using manifold representations," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1360–1369, 2006.

[26] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[27] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 1250–1257, IEEE, 1994.

[28] Y. Tobata, R. Kurazume, Y. Noda, K. Lingemann, Y. Iwashita, and T. Hasegawa, "Laser-based geometrical modeling of large-scale architectural structures using co-operative multiple robots," *Autonomous Robots*, vol. 32, no. 1, pp. 49–62, 2012.

[29] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, 2002.

[30] J. J. Borenstein, H. R. Everett, and L. L. Feng, *Navigating mobile robots : systems and techniques.* A K Peters, 1996.

[31] S. Engelson and D. McDermott, "Error correction in mobile robot map learning," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pp. 2555–2560, 1992.

[32] S. B. Williams, "Efficient solutions to autonomous mapping and navigation problems," 2001.

[33] D. B. S. Portugal, "Robocops: a study of coordination algorithms for autonomous mobile robots in patrolling missions," Master's thesis, 2009.

[34] B. Siciliano and O. Khatib, *Springer handbook of robotics.* Springer, 2016.

[35] R. Gupta and J. M. Conrad, "A Survey on Multi-robot Particle Filter SLAM," in *IEEE SoutheastCon 2019*, IEEE, 2019.

[36] L. Carlone, M. K. Ng, J. Du, B. Bona, and M. Indri, "Rao-Blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 243–249, IEEE, 2010.

[37] K.-C. Ma and Z. Ma, "Multi-robot simultaneous localization and mapping (Multi-SLAM),"

[38] X. S. Zhou and S. I. Roumeliotis, "Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 1785–1792, IEEE, 2006.

[39] M. Wu, F. Huang, L. Wang, and J. Sun, "Cooperative multi-robot monocular-slam using salient landmarks," in *2009 International Asia Conference on Informatics in Control, Automation and Robotics*, pp. 151–155, IEEE, 2009.

[40] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006.

[41] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart, "Map merging for distributed robot navigation.," in *IROS*, pp. 212–217, 2003.

[42] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1, pp. 476–481, IEEE, 2000.

[43] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[44] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Aaai/Iaai*, pp. 852–858, 2000.

[45] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3, pp. 3016–3023, IEEE, 2002.

[46] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1384–1397, 2006.

[47] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," in *ICRA*, vol. 1, pp. 842–849, 2003.

[48] K. M. Wurm, C. Stachniss, and G. Grisetti, "Bridging the gap between feature- and grid-based SLAM," *Robotics and Autonomous Systems*, vol. 58, no. 2, pp. 140–148, 2010.

[49] N. Michael, S. Shen, K. Mohta, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, *et al.*, "Collaborative mapping of an earthquake damaged building via ground and aerial robots," in *Field and Service Robotics*, pp. 33–47, Springer, 2014.

[50] T. Tong, H. Yalou, Y. Jing, and S. Fengchi, "Multi-robot cooperative map building in unknown environment considering estimation uncertainty," in *2008 Chinese Control and Decision Conference*, pp. 2896–2901, IEEE, 2008.

[51] G. A. Bekey, *Autonomous robots: from biological inspiration to implementation and control.* MIT press, 2005.

[52] T. Fukuda and Y. Kawauchi, "Cellular robotic system (cebot) as one of the realization of self-organizing intelligent universal manipulator," in *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 662–667, IEEE, 1990.

[53] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE transactions on robotics and automation*, vol. 14, no. 2, pp. 220–240, 1998.

[54] M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, *et al.*, "Evolving self-organizing behaviors for a swarm-bot," *Autonomous Robots*, vol. 17, no. 2-3, pp. 223–245, 2004.

[55] A. Gil, Ó. Reinoso, M. Ballesta, and M. Juliá, "Multi-robot visual SLAM using a rao-blackwellized particle filter," *Robotics and Autonomous Systems*, vol. 58, no. 1, pp. 68–80, 2010.

[56] N. E. Özkucur and H. L. Akın, "Cooperative multi-robot map merging using fast-SLAM," in *Robot Soccer World Cup*, pp. 449–460, Springer, 2009.