Muti-robot SLAM and communication

Yu Wang
Department of Mechanical Engineering
Stanford University
Stanford, US
yuwangme@stanford.edu

Yixian Li
Department of Mechanical Engineering
Stanford University
Stanford, US
s031820@stanford.edu

Jeff Zhang
Department of Mechanical Engineering
Stanford University
Stanford, US
jefzhang@stanford.edu

Kexin Weng
Department of Mechanical Engineering
Stanford University
Stanford, US
kexinw@stanford.edu

***Abstract*—**
In this paper, we summarize the state-of-the-art Robot SLAM, and formalize the final project to develop a system of multi-robot SLAM which enables communication. We build the system from scratch. Following a carefully defined model, we wrap the system into a comprehensive MATLAB API. A map generation tool for both feature-based map and occupancy grid are developed. We highlight several designs in our SLAM solution. A heuristic corner detection method enables accurate locating of corners. We engineer a cost function in the control policy to accelerate exploration. Detailed implementation results on Extended Kalmon Filter and Particle Filter are discussed. Finally, we propose a communication mechanism between the multi-robot system for future development. ***Index Terms*—Multi-robot, SLAM, network, communication**

I. Introduction

An important research area in robotics and self-driving vehicles in recent years is simultaneous localization and mapping (commonly abbreviated as SLAM). SLAM problems arise when a robot does not have access to a map, nor does it know its state. The interest in robot's exploration on unseen maps and landscapes has spiked in recent years and extensively applied in industry. Multiple extensions from SLAM have been developed and studied; one prominent example is multi-robot SLAM, which involves multiple robots interacting and collaborating to explore an unknown map. In our project, we investigate different methods of map updating using multi-robot SLAM, and how they compare with single robot SLAM, in a simplified 2D environment as describe in section II.

A key characteristic of SLAM problems is the joint handling of continuous and discrete variables and spaces. Naturally, this characteristic comes from the requirement to concurrently estimate the state of a robot, which is often represented as continuous (random) variables, and localize its position in an estimated map by reasoning about the relation between the robot and other objects in the map, which is typically a discrete problem.

Representation of map in SLAM has been long categorized into two types – topological and metric maps. The former describes the connectivity among significant locations, and the latter focus on the geometric description of the environment. While the boundary of two representation can be fuzzy, geometric description generally shares greater resolution at a higher computational cost. Occupancy grid map has achieved popularity under this category, represented as fine grids that hold the probability of obstacles. Another family of metric maps are object maps which use basic shapes such as polygons, lines etc., to depict the objects over the space. It bears limitations in complex object description, and this problem can be solved by integrating some grids representation to form a hybrid map.

Extended Kalman Filter (EKF) is one of the most influential algorithms for SLAM. EKF assumes Gaussian random noise in the robot dynamics and sensor model and hence is generally computationally efficient. But EKF also suffers from a few drawbacks, due to its Gaussian belief representation and Gaussian noise assumptions, including the restriction on feature-based map representation and limitation to only positive sample. These drawbacks can be partially overcome by using other computationally tractable methods as described in the following texts in this section and subsection III-A.

In the rest of this section, we review the literature on the regime of SLAM and multi-robot interaction. We will conclude with a survey of potential data sets, demonstrations, and implementations for our project.

A. Single robot SLAM

In a paper from H. Durrant-Whyte and T. Bailey [3], a comprehensive summary of SLAM is provided. As mentioned in the above texts, the idea behind SLAM is to jointly time-update and measurement-update the estimates for both robot's state x_k and map m. The update formulas take the following mathematical form:

$$P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) \tag{1}$$

$$= \int P(x_k|x_{k-1}, u_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1}, \tag{2}$$

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \frac{P(z_k|x_k, m)P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})}, \tag{3}$$

where x denotes the state of robot; m denotes the constructed map; U is the control input; Z denotes the measurement data. In SLAM, because the observation of relative difference between landmarks is independent of the coordinate frame of the vehicle and successive observations from this fixed location, this gives further independent measurements of the relative relationship between landmarks. This property guarantees that our knowledge of the landmark converges as we take more measurements between these landmarks.

Two practical and classic algorithms that are frequently utilized by researchers for solving SLAM problems are presented and discussed below.

1) EKF-SLAM: EKF-SLAM is built upon general EKF [3]. It performs the same prediction step and similar update step as EKF with different definitions for mean and covariance:

$$\mu_{k|k} = \begin{bmatrix} \hat{x}_{k|k} \\ \hat{m}_k \end{bmatrix}, \tag{4}$$

$$P_{k|k} = \begin{bmatrix} P_{xx} & P_{xm} \\ P_{mx} & P_{mm} \end{bmatrix}_{k|k}. \tag{5}$$

The predict step is the same as before:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k), \tag{6}$$

$$P_{xx,k|k-1} = \nabla f P_{xx,k-1|k-1} \nabla f^T + Q_k, \tag{7}$$

where ∇f is the Jacobian of f evaluated at the estimated state $\hat{x}_{k-1|k-1}$.

The new update on observation incorporates our map knowledge.

$$\mu_{k|k} = \mu_{k|k-1} + W_k \left[z(k) - h(\hat{x}_{k|k-1}, \hat{m}_{k-1}) \right], \tag{8}$$

$$P_{k|k} = P_{k|k-1} - W_k S_k {W_k}^T, \tag{9}$$

where

$$W_k = P_{k|k-1} \nabla h^T S_k^{-1}, \tag{10}$$

$$S_k = \nabla h P_{k|k-1} \nabla h^T S_k^{-1}. \tag{11}$$

It is of note that ∇h is the Jacobian of h (measurement model) evaluated at $\hat{x}_{k|k-1}$ and $\hat{m}_{k-1}$.

However, as mentioned above, EKF-SLAM has a few limitations due to requiring that both process noise and measurement noise to be Gaussian, and the non-linearity of the system may cause divergence of the state estimation. The alternative method introduced below can alleviate some of these issues of EKF in solving the SLAM problem.

2) Rao-Blackwellised Filter: The Rao-Blackwellised Filter is an algorithm that is based on recursive Monte Carlo sampling (aka. particle filter), with an emphasis on reducing the sample space. The algorithm is stated in the paper [3] as followed:

1. For each particle, compute a proposal distribution, conditioning on the specific particle history, and draw a sample from it $x_k^{(i)} \sim \pi(x_k | X_{0:k-1}^{(i)}, Z_{0:k}, u_k)$. This new sample is (implicitly) joined to the particle history $X_{0:k}^{(i)}$, defined as $\{X_{0:k-1}^{(i)}, x_k^{(i)}\}$

2. Weight samples according to the importance function

$$w_{k-1}^{(i)} \frac{P(z_k | X_{0:k}^{(i)}, Z_{0:k-1}) P(x_k^{(i)} | x_{k-1}^{(i)}, u_k)}{\pi(x_k^{(i)} | X_{0:k-1}^{(i)}, Z_{0:k}, u_k)}, \tag{12}$$

where

$$P(z_k | X_{0:k}^{(i)}, Z_{0:k-1}) = \int P(z_k | x_k, m) P(m | X_{0:k-1}, Z_{0:k-1}) \mathrm{d}m. \tag{13}$$

3. If necessary, perform resampling. Resampling is accomplished by selecting particles, with replacement, from the set $\{X_{0:k}^i\}_i^N$, including their associated maps, with probability of selection proportional to w_k^i. Selected particles are given uniform weight, $w_k^i = \frac{1}{N}$.

4. For each particle, perform an EKF update on the observed landmarks as a simple mapping operation with known vehicle pose.

Our project's implementation will heavily focus on EKF-SLAM and Rao-Blackwellised Filter as a baseline for both single robot SLAM and multi-robot SLAM.

B. Multi-robot SLAM[7]

Multi-robot SLAM is an extension from single robot SLAM. In the paper by Kuzmin [10], a thorough review, classification, and comparison of existing SLAM methods for group of robots is provided. The following papers also report work on solving SLAM problems for multi-robot systems.

1) Extending SLAM to multiple robots[7]: This paper dives into the implementation of EKF-SLAM of multi robots. Though the paper is a project work and the simulation had not yet been tested successful, some insight is worthy to be taken away.

Section 2.2.1 addressed a solution to the case where we do not have a global coordinate. They proposed that they could define an origin in robot A's coordinate, and whenever the robots traded information, they could directly utilized a measurement of the relative pose to find that origin in robot B's coordinate.

Section 2.2.3 highlighted the concept of treating the other robot information as measurement. With the change of coordinate, one robot could convert the measurement of the other robot into its own.

The third section also provided detailed methodology of implementing EKF-SLAM on multi-robots, as well as the network they are building for communications. This could potentially be of reference to our paper.

2) Rao-blackwellized particle filters multi-robot SLAM with unknown initial correspondences and limited communication [1]: This paper dives into addressing the problem of multi-robot SLAM with limited communication and unknown relative initial poses. Based on single robot Rao-Blackwellized particle filters, they proposed a technique that jointly estimates SLAM posterior of two robots by fusing only the laser stabilized odometry and corresponding laser scanner measurements acquired by each teammate when they actually met each other.

The caliber of this method is described in section 3D in [1]. In short, they utilized a frame-transformation methodology that allowed two robots to process the received information given the relative pose measurements just when the data is just received (by inspecting on their posterior).

This paper provides insight into tackling the case when two robots do not share a wide network and only able to transmit limited information within a certain distance. The work also adopts RBPF method that we may potentially work upon.

C. Sensor Error and Mapping Techniques

One of the core challenges with SLAM is map reconstruction using noisy sensor measurements and odometry. A laser rangefinder, an example of a typical sensor that might be used in a SLAM robot, is subject to many sources of error both intrinsic and extrinsic to the sensor. Intrinsic errors include statistical errors, drift errors, cyclic errors, and misalignment errors [5]. Extrinsic errors, which heavily depend on the light environment, include scattering, and other light sources [8]. Odometry, typically measured with rotary encoders or IMUs, can suffer greatly from drifts as the robot moves in the environment. Therefore, as the robot moves and receives sensor data, the estimated map can distort and drift significantly from one time step to another.

Therefore, point matching techniques such as RANSAC and iterative closest point (ICP) are typically used to correct for the mapping errors.

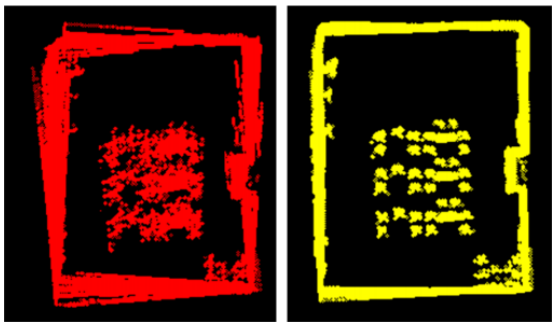

Fig. 1. Mapping with (right) and without (left) ICP [9]

Landmarks, rather than the much bigger set of all points in a 2D point cloud, are typically used to localize the robot in SLAM, and they are typically feature points such as corners [overview]. Although there are many sophisticated algorithms for extracting corners in images, a simple algorithm using neighboring points in a range finder measurement is sufficient to extract corners in 2D mapping [11].

D. Implementation

The implementation will be in `MATLAB`. We will make use of the `Navigation` toolbox [13], which includes APIs for map representation, sensor modeling, path planning, SLAM, etc.

We will generate maps of different complexity for the problem setup as described in section II.

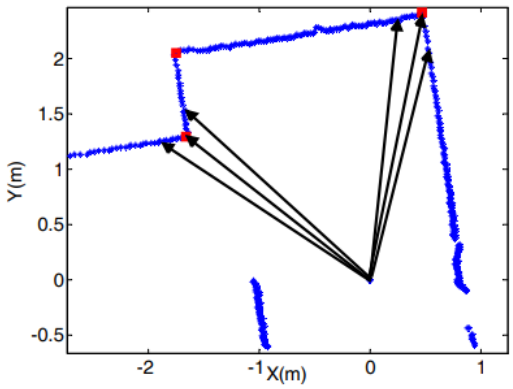

Fig. 2. Corner Detection Visualization [11]

II. Problem Setup

We propose to study multi-robot SLAM in the following setting. A team of $n \geq 2$ identical differential drive robots [2, 15] are collaborating on mapping an 2D interior with landmarks. Each robot carries a range finder [14, 17] as its sensor and can communicate with other robots within a certain distance d from it. The communication conveys information about the estimated state of the ego robot and the estimated area to be mapped. A similar problem setup can be found in [6].

In this study, we adopt a simple exploration policy for all robots, i.e., each robot follows a path that is at equal distance from all nearest obstacles. More details on exploration policy are provided in subsection V-B. We assume static environment in this study, i.e., the environment is changing over time and the robots do not interact with it. It is of note that the perceived environment for each robot may be dynamic due to the movement of other robots.

We propose to study multi-robot SLAM from the following three aspects. Firstly, we are interested in comparing various algorithms in this problem setup, in terms of the accuracy of estimated robot states and maps, robustness against process and measurement noises, time and space complexities, scalability, etc. Secondly, we will examine the effects of the total number of robots on the performance of algorithms. Thirdly, it is of interest to understand the effects of others factors such as the complexity of map, the choice of dynamic model and sensor model, the representation of landmarks, etc. Most of the above studies will be based on both theoretical and empirical evidence.

III. Methodology

A. Overview

We propose to use Extended Kalman Filter as the baseline method. GraphSLAM, FastSLAM, sparse extended information filers [16] and variants of particle filter [1, 4, 6, 12], including the Rao-Blackwellized particle filter, will be implemented and compared as part of the study.

B. Map Representation & Generation

In this project, we use both geometric representation and grid representation in the map, each of which has its advantages towards different algorithms. Geometric representation allows us to use landmarks to represent obstacles, form corners and detect edges in a continuous space. Grid representation would have better scalability and more intuitive manipulations in depicting complex shapes of obstacles and objects at a cost of more extensive calculation. Specifically, in Particle Filter Implementation, we adopt the occupancy grid map. For EKF implementation, we use feature-based map.

The complexity of the 2D map is a dimension we will explore. We start with a baseline map – for geometric, some simple polygon objects with landmarks (represented using lines); for a grid map, we define obstacles with regular shapes (circle/rectangle in 2D space). We propose a method to generate more maps with greater complexity. We take in an image and transform it into grayscale. For grid representation, we define pixels having value over some threshold as obstacles and else as free space. Computer vision algorithms will be used to detect edges and corners to form a geometric map with lines for the geometric model.

C. Dynamics and Robot State Model

We define each robot as a holonomic drive robot. The control input is $u = [v, \omega]^T$, where v and ω are our linear and angular velocities. Our robot state, x_r, is based on the global frame, and includes our information about location, heading, and linear and angular velocities.

Wheel encoder readings will serve as our basis for odometry, by feeding them directly into our EKF formulation, as we will discuss later in this paper.

D. Sensor Model and Landmark Extraction

Our sensor of choice is a range finder that returns a vector of distances, at a resolution of 1.40°(default in `Navigation` toolbox [13]). We will use a simplified error model which assumes a constant angular variance σ_θ^2 and a distance variance σ_r^2 proportional to the true distance. Therefore, the i^{th} index of our raw measurement vector for each time step t, represented as s_t^i, has distribution of the form of:

$$s_t^i = \begin{bmatrix} \hat{\theta}_t^i \\ \hat{r}_t^i \end{bmatrix} = \begin{bmatrix} \mathcal{N}(\mu_{\theta_t^i}, \sigma_{\theta_t^i}^2) \\ \mathcal{N}(\mu_{r_t^i}, \sigma_{r_t^i}^2 \sim r_t^i) \end{bmatrix} \tag{14}$$

Figure 3 and Figure 4 visualizes the 90% error range for a selection of simulated measured points first in polar space, then projected to Cartesian space. As one can see, the noise is Gaussian in polar space, but not in Cartesian space. This will be important in our EKF formulation later. Our rationale for this error model is two-fold: there is inherent constant uncertainty in angular position due to sensor resolution, and that the farther away a ray is reflected, the more likely it is affected by noisy effects such as scattering. As seen in Figure 4, the error in Cartesian space of individual points, using this sensor error model cannot be expressed with a covariance matrix. This problem is bypassed, since the uncertainty can be better approximated when we use a corner detection algorithm in the next section.

Note that in the EKF implemented, we don't use raw sensor data. Instead, we extract corners and use corner locations as our EKF measurement. This error model is only used to simulate real-world sensor noise.

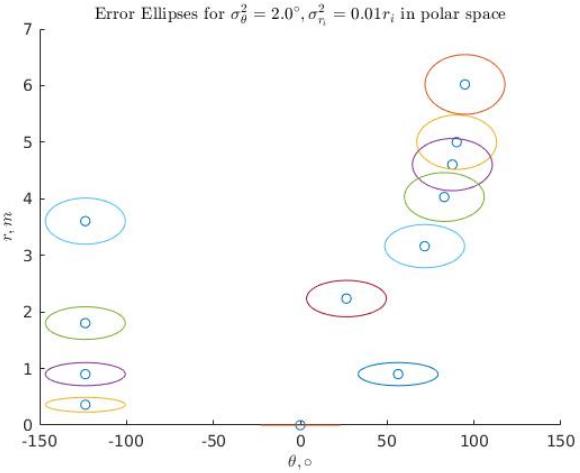

Fig. 3. Corner Detection Visualization [11]

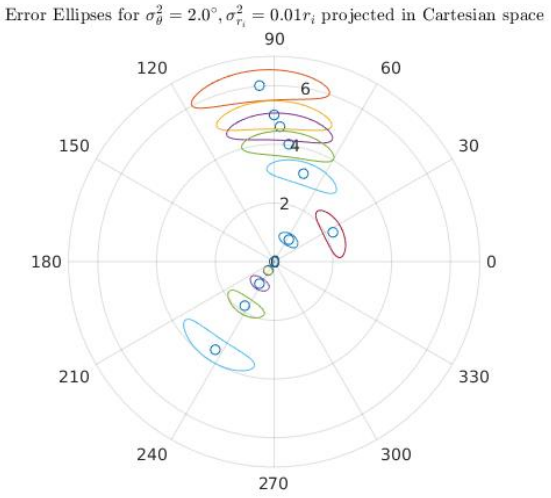

Fig. 4. Corner Detection Visualization [11]

Since our filter will use landmark locations rather than the entire collection of sensor measurements, we need to extract landmarks and assign permanent labels to each landmark that we see. To extract landmarks, we will use the methodology proposed by [11], summarized by the equation in Equation 15.

$$\sum_{j=m-M/2}^{m+M/2} |r_m - r_k| \geq r_{\text{corner\_threshold}}. \tag{15}$$

Our final landmark estimations, along with their error estimates, will be fed to our EKF pipeline as our measurement.

E. EKF Formulation

Our state variable, which we try to estimate, includes our robot state x_r as well as the global position of the individual landmarks, x_l. Our measurement variable includes the polar space position of landmarks in the robot frame, as well as our wheel encoder tick changes.

The detailed process follows the algorithm provided in single robot's EKF-SLAM section in Introduction. Where the dynamics model f derives from the differential drive robot model from (C), and the observation model undergoes a transformation from Polar space to Cartesian space, and from local frame to global frame.

IV. Design Highlights

A. Corner Detector

Computer vision techniques are typical practice for corner detection. Yet in our SLAM problem, they perform poorly due to discontinuities and high computational cost. We adopt a 3-step algorithm to estimate the location of corners, and derive errors for the estimation.

Corner detector takes in a vector of range measurements as input, in forms of [ranges, angles], angles ordered from $[-\pi, \pi]$. We plot ranges with respect to angles, and observed points form parabolas as in Figure 5. The corner candidates are the intersections of parabolas.

Therefore, based on this observation, the 3-step algorithm can be described in high level as:

1) Use a sliding window on sensor data to compute a corner "heuristic" to find corner candidates, then use non-maximum suppression to remove duplicates.
2) Use total least squares (TLS) to line-fit points on either side of candidate points, then compute angle θ between two lines to remove false positives if θ is smaller than a threshold.
3) Recompute intersection of two lines for center estimation

The estimation error of corner is defined as an expected TLS residual, i.e. variance along the direction perpendicular to the two lines that intersects at this corner, with calculation described in Equation 16

$$\sigma^2 = \sum_{i}^{n} (d_{\perp}^{(i)2})/n \tag{16}$$

n is total number of points along the two lines that intersects at the corner. $d_{\perp}^{(i)}$ is the distance between the i^{th} measurement point to its corresponding fitted line.

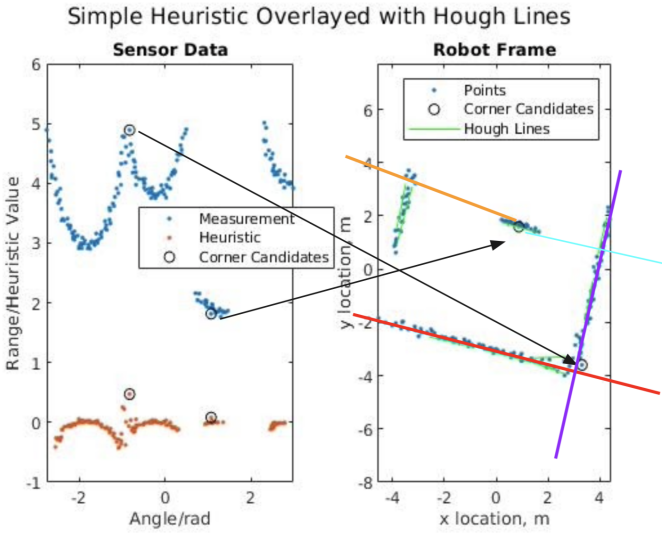

Fig. 5. Corner Detection Visualization

Detailed implementation are described in pseudo code algorithm 1.

Data: Paired vectors [ranges], [angles]
Result: a vector of corner estimation
initialization:
m ← 7, for window size
thres ← 0.075, for corner candidate
corner_candidate ← [false,false,...,false]
$i \leftarrow 0$
while *i < length(ranges)* **do**
  ranges ← circshift by i indices
  window←ranges(1:2m+1)
  wl←window(1:m)
  wr←window(m+1:2m+1)
  h_value ← $\frac{1}{||window||_{l2}}$(∑(wl-ranges(i))∑(wr-ranges(i)))
  if *h_value > threshold* **then**
    corner_candidate[i] ← true;
    current section becomes this one;
  else
    go back to the beginning of current section;
  end
end
nonMaximumSuppression(ranges,corner_candidate)
for *i, corner_candidate(i)==True* **do**
  line_l ← $TLS(left\_points(i))$;
  line_r ← $TLS(right\_points(i))$;
  if *angle_between(line_l,line_r) > threshold* **then**
    is_corner(i) = True;
  end
end

Algorithm 1: Corner Detector Pseudo Code

We produce sound results for corner detection with $TP =$

220, $FP = 5$, $FN = 25$. Evaluation metrics F1-score, Precision, and recall are summarized in Table I. A typical visualization is displayed as Figure 6

Metrics	Value
F1-score	0.936
Precision	97.7%
Recall	89.8%

TABLE I
CORNER ESTIMATION EVALUATION

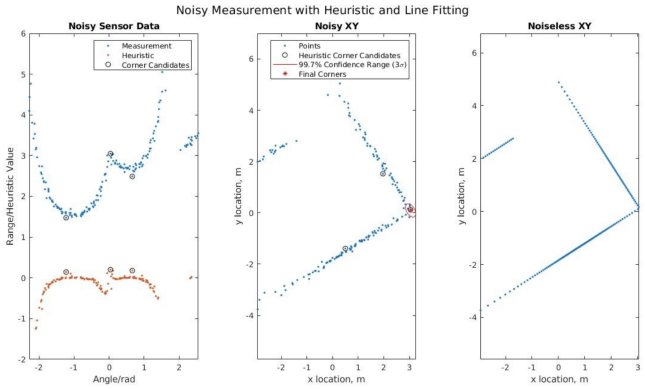

Fig. 6. Noise Visualization of Corner Estimation

B. Exploration Policy

To achieve fast exploration over the space, we deploy a cost function field over the robot location, which determines the direction the robot navigate towards. Magnitude of position update in each step is constant.

The cost field value at map location $[x, y]$ given robot current position $P_t = [p_1, p_2]$ comprises of two parts: (1) stay cost (2) obstacle cost, as in Equation 17.

Robot compares the cost values in the grids located on a range circle, and move towards the grid with the lowest cost.

$$C[x, y|P_t] = C_S[x, y|P_t] + C_O[x, y|P_t] \tag{17}$$

- C_S represents the stay cost, which is a exponentially decay 2D Gaussian cost centered about robot's current position as in Equation 18, Equation 19.

$$C_S[x, y|P_t] = \alpha \cdot C_S[x, y|P_{t-1}] + c_s[x, y|P_t] \tag{18}$$

$$c_s[x, y|P_t] = 3 \cdot e^{-(x-p_1)^2/r} \cdot e^{-(y-p_2)^2/r} \tag{19}$$

  α is the decay factor, weighting historical exploration, and we use $\alpha = 0.99$ for implementation. c_s is the new stay cost to be added in each update, which is Gaussian around the robot position and is independent of time. $r = 1$ is a tuned parameter characterizing the Gaussian. Gaussian distribution of c_s indicates that grid's stay cost becomes larger when robots explore neighboring area, thus encouraging robot to explore new area.

- C_O represents the obstacle cost, which is calculated by looping over all N obstacles, and find the maximum of individual obstacle Gaussian cost (c_o), as expressed in Equation 20,Equation 21. We use $O^{(i)} = [o_1^{(i)}, o_2^{(i)}]$ to represent i^{th} obstacle's location in the map.

$$C_O[x, y] = \max_i^N c_o^{(i)}[x, y] \tag{20}$$

$$c_o^{(i)}[x, y] = \text{convolve2D}(F^{(i)}, G_{kernel})[x, y] \tag{21}$$

  where $F^{(i)}$ is the map-size matrix, characterizing obstacle i^{th} location by having value 1 at location $[o_1^{(i)}, o_2^{(i)}]$ and all other elements being zeros. G_{kernel} is the Gaussian kernel characterizing the obstacle costs distribution around its location.
  Obstacle cost is independent of time. Use of maximum over N obstacles's cost distributions is more meaningful than summation in cases when obstacles clusters in small area (e.g. a tunnel), we want to avoid explosion of costs, and still enable the robot to explore through.

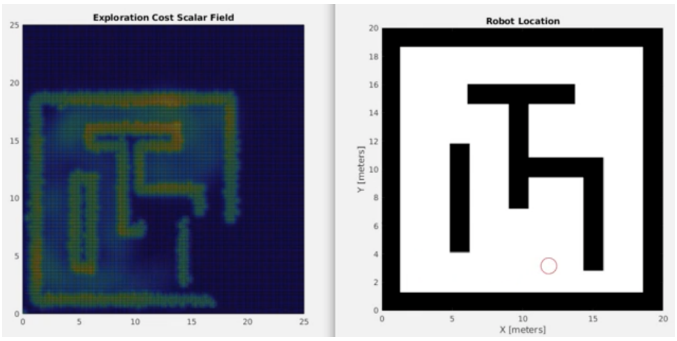

Fig. 7. Cost Function Visualization

V. SLAM IMPLEMENTATION

A. Extended Kalman Filter

In this section, we will walk through the one of the main algorithms we are implementing for the project. In order to apply EKF to this problem, we rely on using a feature-based map, where corner detection would be extremely useful. In addition, since we are using the range finder, we do not observe every landmark at the same time, and we can not establish a correspondence between each measurement and landmark. Given all these information, a basic algorithm has been proposed and setup as followed.

1) Overall algorithm: The rationale of our algorithm is adapted from Stanford AA274A lecture on SLAM, and is shown in Fig. 7. It demonstrates a method of hypothesizing landmarks and finding correspondences between measurements and existed landmarks at each EKF update step. For each measurement, the algorithm loops through all existed landmark, and establishes the correspondence by finding the landmark that has the smallest Mahalanobis distance with the current measurement. If this distance is greater than a constant,

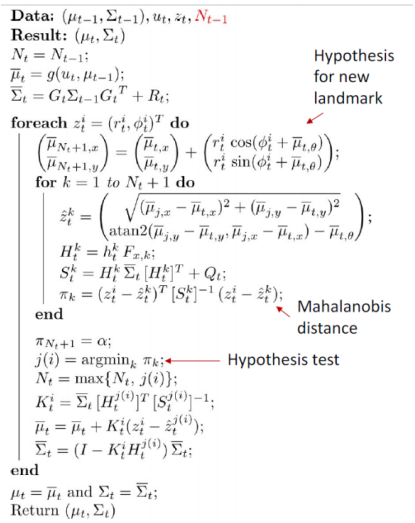

Fig. 8. EKF SLAM with unknown correspondence algorithm

we hypothesize a new landmark. However, our measurement is local x and y coordinate, so the measurement model and Jacobian is different. Typically, they are subjected to a local transformation. For more detail, please refer to the Github repository at the end of the report.

2) Initial condition: We define initial μ_0 and initial Σ_0 to be:

$$\mu_0 = [0, 0, 0, ..., 0] \in \mathbb{R}^{3+2N} \tag{22}$$

and,

$$\Sigma_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & \infty & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & \infty & ... & 0 \\ & & & & ... & & \\ 0 & 0 & 0 & 0 & 0 & ... & \infty \end{bmatrix} \in \mathbb{R}^{3+2N \times 3+2N} \tag{23}$$

where N is designed as a quantity greater than the maximum feature points in the map.

3) Testing and performances: From the algorithm, we tested out some simple demos to validate the correctness of the algorithm. We specify three landmark positions at [6, -2], [6, 2], and [9, 0].

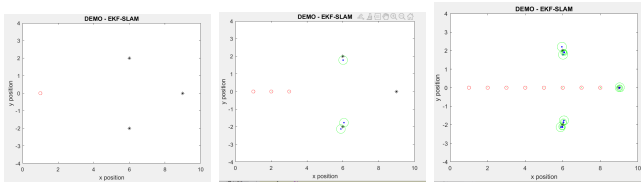

Fig. 9. System evolve and landmark estimation.

Fig.9 shows a simple demo of a driven robot running on a straight line. At first, no measurement was taken, as the robot approached a landmark, it hypothesized the landmark and then further became more and more certain about the landmark's position.

4) Problems and Solution: The major problem of this algorithm is that it would falsely hypothesizes a new landmark and incorporate it into mean and co-variance matrix easily even if the threshold is properly tuned. We experimented with several other control input such as high angular rate, the algorithm attempts to hypothesize a new landmark frequently. A way to potentially resolve this issue would be merging nearby landmarks which are too close to one another and reinitialize the eliminated mean and co-variance. After fixing this issue, we would implement our EKF-SLAM algorithm onto our factorized map. This will be left as future work.

B. Particle Filter

Particle filter is implemented. The algorithm is shown below

1) Sample particles from prior distribution of the poses of robots.
2) Occupancy grid mapping (since map is represented as occupancy grids)
3) Compute measurement likelihood. Re-sample particles.

In the following texts of this section, we demonstrate particle filter using two robots. We run the particle filter for 40 time steps; each time step is 1 unit of time. 10 particles are used for each robot. We initialized the robot positions at locations where the two robots can not see each other. This is to avoid the two robots appear in the other robot's sensor measurement.

Figure 11 shows the estimated map after 40 time steps. Figure 10 shows the true trajectories and estimated trajectories of the two robots.

A unique issue that we faced was related to the perfect controller that was used. The perfect controller is assumed to be able to achieve any desired state in a single time step. As a direct consequence, the robots can maneuver in a significantly non-smooth way, and hence making their poses difficult to estimate.

VI. Design of Network Communication

Our original problem setup proposal includes a network communication component. We proposed that robots can share estimated map or estimated state when certain conditions are met. Network communication can be expected to be critically when robots appear in the sensor range of each other. In practice, implementation of such network communication proved to be difficult. Specifically, alignment of estimated map from different robots can be difficult. This issue is even more prominent if at the time of communication, the robots have been mostly exploring different areas on the map. Potentially, we can make simplifying assumptions, such as the initial poses of robots are known or all robots start from the same position.

VII. Future Work

Future work of this project involves fixing the issues from EKF and particle filter algorithms. Once resolved and the robot could simulate for a consistent amount of time, we would

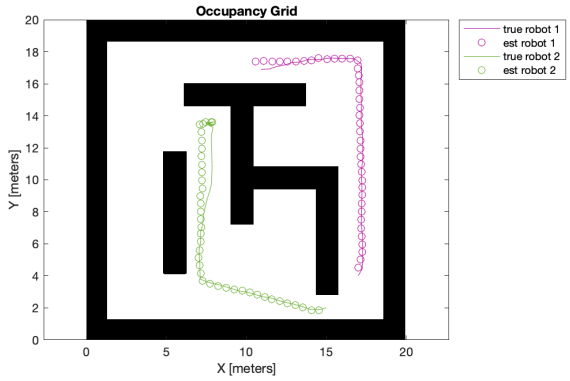

Fig. 10. Trajectory of two robot overlaid on true map. 40 time steps.

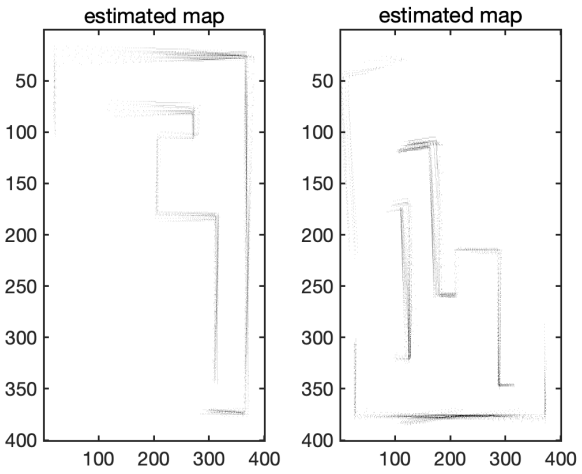

Fig. 11. Estimated map from two robots. 40 time steps.

plan on building a communication system as described above. Further investigation over multi-robot network construction would be reviewed and attempted.

VIII. GitHub repo link

We collaborated on this project on GitHub. The codes can be found at https://github.com/yuwangme/multirobot-slam.

Acknowledgment

We would like to thank Mac and all teaching staff for a great quarter!

References

[1] Luca Carlone, Miguel Kaouk Ng, Jingjing Du, Basilio Bona, and Marina Indri. Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication. In *2010 IEEE international conference on robotics and automation*, pages 243–249. IEEE, 2010.

[2] Yongoug Chung, Chongkug Park, and Fumio Harashima. A position control differential drive wheeled mobile robot. *IEEE Transactions on Industrial Electronics*, 48(4):853–863, 2001.

[3] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[4] Arturo Gil, Óscar Reinoso, Mónica Ballesta, and Miguel Juliá. Multi-robot visual slam using a rao-blackwellized particle filter. *Robotics and Autonomous Systems*, 58(1):68–80, 2010.

[5] K.S. Hashemi, P.T. Hurst, and John Oliver. Source of error in a laser rangefinder. *Review of Scientific Instruments*, 65:3165 – 3171, 11 1994.

[6] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256, 2006.

[7] Ethan Howe and Jennifer Novosad. Extending slam to multiple robots, 2005.

[8] Jacobs. Jacobs, v. et. all: Analyses of errors associated with photometric distance in goniophotometry 1 analyses of errors associated with photometric distance in goniophotometry. 2015.

[9] Achmad Aditya Kusumo, Bayu Sandi Marta, Bima Sena Bayu Dewantara, and Dadet Pramadihanto. 2d mapping and localization using laser range finder for omnidirectional mobile robot. In *2019 International Electronics Symposium (IES)*, pages 126–131, 2019.

[10] Maxim Kuzmin. Review. classification and comparison of the existing slam methods for groups of robots. In *2018 22nd Conference of Open Innovations Association (FRUCT)*, pages 115–120, 2018.

[11] Shifei Liu, Mohamed Atia, Yanbin Gao, and Aboelmagd Noureldin. Adaptive covariance estimation method for lidar-aided multi-sensor integrated navigation systems. *Micromachines*, 6, 02 2015.

[12] Weilian Liu. Slam algorithm for multi-robot communication in unknown environment based on particle filter. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–9, 2021.

[13] Matlab navigation toolbox, 2021a. The MathWorks Inc., Natick, Massachusetts, United States.

[14] Marc Rioux. Laser range finder based on synchronized scanners. *Applied optics*, 23(21):3837–3844, 1984.

[15] Se-gon Roh and Hyouk Ryeol Choi. Differential-drive in-pipe robot for moving inside urban gas pipelines. *IEEE transactions on robotics*, 21(1):1–17, 2005.

[16] Sebastian Thrun and Yufeng Liu. Multi-robot slam with sparse extended information filers. In *Robotics Research. The Eleventh International Symposium*, pages 254–266. Springer, 2005.

[17] Qilong Zhang and Robert Pless. Extrinsic calibration

of a camera and laser range finder (improves camera calibration). In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2301–2306. IEEE, 2004.