INVESTIGATION OF MULTI-ROBOT FASTSLAM

Ping-Jung Liu
University of Chicago
Chicago, Illinois
howard5758@gmail.com

Yuqian Gong
University of Chicago
Chicago, Illinois
yuqian919@uchicago.edu

June 13, 2019

ABSTRACT

Robot localization and mapping are two of the main research field in robotic estimation and planning. Specifically, simultaneous localization and mapping, SLAM, of robot has drawn great attention for quite some years because of its application in surveying unknown locations with rough GPS signals or provide further assistance in modern autonomous vehicles. On the other hand, the concept of multi-robot systems has also gotten popular because of the power in cooperation among robots. Given the success of single robot SLAM, the idea of using multiple robots to perform SLAM as a team became an additional research field in recent years. Supposedly, multiple robots should be able to explore unknown environments more efficiently than using single robot, but challenges include, but not limit to, map fusion and communication among robots. This project presents comparison of the performance of using single and multiple robots in a SLAM problem.

1 Introduction

Robot localization involves estimating the poses of robots using control inputs, sensor observations, and map information. On the other hand, mapping is the problem of estimating the locations of certain landmarks in the environment while knowing the robots' current poses. SLAM, as its name suggests, performs the two tasks simultaneously. Robots use localization results to enhance mapping accuracy and the other way around.

The extended Kalman Filter, EKF, method is one of the most widely used approach when solving SLAM problems; EKF performs well when distributions are mostly Gaussian but tend to diverge when the data are not distributed that way. In this project, we first mitigated the downside of EKF by using particle filter to estimate robot's locations, but still applying EKF when updating the estimation of landmarks' estimations and co-variances; this method has been given the name FastSLAM. We then extended the implementation of single robot FastSLAM to support multiple robots SLAM in hopes of enhancing efficiency and accuracy.

Though the method we implemented to merge the information stored in multiple robots seems trivial, which will be further discussed in later sections, it is in fact a reasonable and straight forward approach that enhances the performance of SLAM. It can be shown by error plots and visualizations that SLAM problems can indeed benefit from multiple robots.

The SLAM problem in this project is a rectangular field with twenty-four landmarks and no obstacles. The landmarks are all labeled, which means known data correspondences and sensors are able to directly observe labels of landmarks. Please see figure 1 for the visualization of the map.

The rest of the paper is organized as follow. In section 2, we go through the theoretical background and implementation of single robot FastSLAM, along with numerical results and visualizations that demonstrate the difference between using various number of particles. In section 3, we discuss the challenge of expanding to multi-robot FastSLAM and its implementation, also with corresponding results and visualizations to show advantages of using multiple robots in SLAM. In section 4 and 5, we draw a conclusion and share thoughts on potential future projects.

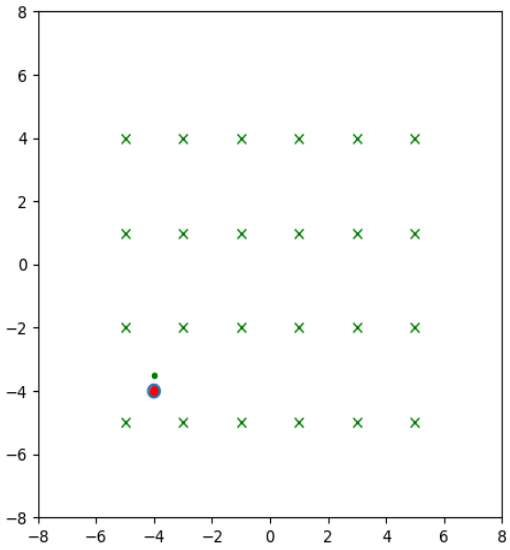

Figure 1: Visualization of map. The Xs represents landmarks.

2 SLAM, Motion Model, Observation Model

Unlike all localization methods, which require map information to perform accurate localization of targets, SLAM does not depend on a map of the working environment. SLAM focuses on performing localization of robots and mapping of environment simultaneously.

In our problem setting, each robot will be provided a starting pose, a list of control inputs, and a list of relative coordinates of the observed landmarks at each time step. The algorithm will then use these information to estimate the positions of robots and landmarks at every time steps.

In this section we will go through the motion and observation models of our simulated robots to explain how the robots make transitions and obtain measurements. The explanation are mostly extracted from the problem description of PS3, TTIC31170 Robotics Planning, Learning, and Estimation.

2.1 Motion Model

Our robots have the following dynamics:

$$x_t = x_{t-1} + (d_t + v_{1,t})cos(\theta_{t-1}) \tag{1}$$

$$y_t = y_{t-1} + (d_t + v_{1,t})sin(\theta_{t-1}) \tag{2}$$

$$\theta_t = \theta_{t-1} + \delta\theta_t + v_{2,t} \tag{3}$$

where the control data u_t = [d_t $\delta\theta_t$] consists of the body-relative forward distance that the robot moved d_t and its change in orientation $\delta\theta$t, and v_t = [$v_{1,t}$, $v_{2,t}$] in N (0, R) is zero-mean Gaussian noise that captures uncertainty in the forward velocity and angular rate.

2.2 Observation Model

Assume also that the robot observes the relative (x, y) coordinates of each feature within the field-of-view of it's sensor, denoted as Z_x and Z_y. THe sensor is a LIDAR with maximum range of 4 units and an angular field-of-view of 180 degrees) according to the following measurement model:

$$Z_{x,t} = cos\theta_t(x_m - x_t) + sin\theta_t(y_m - y_t) + w_{x,t} \tag{4}$$

$$Z_{y,t} = -sin\theta_t(x_m - x_t) + cos\theta_t(y_m - y_t) + w_{y,t} \tag{5}$$

where (x_m, y_m) denotes the coordinates of the observed landmark and w in N (0, Q_t) denotes zero-mean Gaussian measurement noise. We can also find the inverse of these equations to perform updates on landmarks.

3 FastSLAM

Before diving into the technical details of FastSLAM, we will go through its general idea and the pros against using EKF. Just like regular particle filters, robot position posterior is represented by a set of weighted particles. In plain words, the weight of a particle indicates how close the pose of the particle is to that of the ground truth robot. On the other hand, each landmark of each particle is represented by a 2X2 EKF system and for each observation we perform standard EKF update on these EKFs.

Kalman filters are based on the assumption that both state transition and observation models are linearly Gaussian with noise. While this assumption limits the usage of EKF, it also has high convergence and handle uncertainty with great potency. The problem setting in this project is in fact most suitable for EKF, which is why we will not present or compare its results with the methods of focus in this project.

Particle filters are extended from the idea of Monte Carlo simulations. Since the Baysean posterior is estimated by set of particles, particle filters can handle non-linearity and non-Gaussian noise with ease. The challenge of particle filters thereby lies in the growth of complexity as particle numbers get larger. Especially in the case of particle filter SLAM, each particle will store its own imagination of all the observed landmarks.

3.1 Single robot FastSLAM Algorithm

Factorization of the FastSLAM posterior:

$$\begin{aligned} & p(x_{0:t}, l_{1:M} \mid z_{1:t}, u_{1:t}) \\ & = p(x_{0:t} \mid z_{1:t}, u_{1:t}) p(l_{1:M} \mid x_{0:t}, z_{1:t}) \\ & = p(x_{0:t} \mid z_{1:t}, u_{1:t}) \prod_{i=1}^{M} p(l_i \mid x_{0:t}, z_{1:t}) \end{aligned} \tag{6}$$

Figure 2 shows the graphical structure of a single robot FastSlam model where m is the landmark variable. The graphical structure provides important information of landmark variables: the landmarks are conditionally independent given the sequence of robot states and observations. Therefore given the sequence of robots states and observations, there is no need to compute the joint probability of all landmarks. As the last line of equation (6) shows, we can factor the joint probability of landmarks as product of probabilities of individual landmark given x and z, where each individual landmark pose can be estimated with 2 dimensional EKFs.

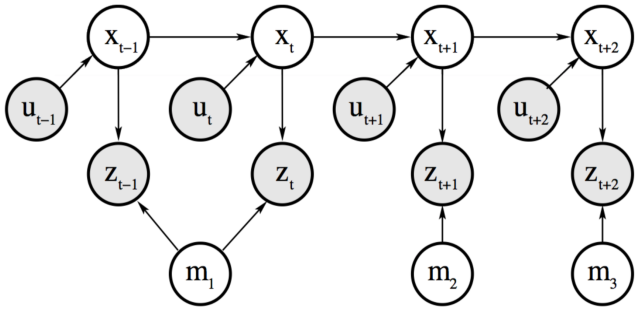

Figure 2: *Graphical structure of FastSLAM*

In the following algorithm for single robot FastSLAM, we see that at each time step t, each particle maintains an estimated pose of itself and a set of estimated pose and covariance of observed landmarks. Same as particle filter algorithm, the weight of each particle will reflect its importance in determining the mean estimated pose of the true robot. For each particle, if the input observation of landmark is never seen before, we will initialize the estimated pose and covariance for this landmark; it this the input landmark is seen before, we will update the estimated pose and covariance for this landmark as in EKF. In addition, the weight of each particle will be also updated based on the similarity of its measurement prediction to the true robot's measurement.The more similar they are, the higher weight this particle will have. All variables associated with unobserved landmarks will remain unchanged. After updating the

weight of each particle, we will always resample particles. Notice that different from the EKF, there is no prediction step here when estimating the landmarks.

Algorithm 1 Single robot FastSLAM

$FastSLAM(z_t, c_t, u_t, X_{t-1})$:
for $k = 1$ *to* N **do** ▷ loop over all particles
  Let $(x^k_{t-1}, (\mu^k_{t-1}, \Sigma^k_{1,t-1}), ...)$ *be particle* k *in* X_{t-1}
  x^k_t ~$p(x_t \mid x^k_{t-1}, u_t)$ ▷ sample pose
  $j = c_t$ observed feature
  if *feature* j *never seen before* **then**
    $\mu^k_{j,t} = h^{-1}(z_t, x^k_t)$ ▷ initialize mean
    $H = h'(\mu^k_{j,t}, x^k_t)$ ▷ calculate Jacobian
    $\Sigma^k_{j,t} = H^{-1} Q_t (H^{-1})^T$ ▷ initialize covariance
    $\omega^k = p_0$ ▷ default importance weight
  else
    $\hat{z}^k = h(\mu^k_{j,t-1}, x^k_t)$ ▷ measurement prediction
    $H = h'(\mu^k_{j,t-1}, x^k_t)$ ▷ calculate Jacobian
    $Q = H \Sigma^k_{j,t-1} H^T + Q_t$ ▷ measurement covariance
    $K = \Sigma^k_{j,t-1} H^T Q^{-1}$ ▷ calculate Kalman gain
    $\mu^k_{j,t} = \mu^k_{j,t-1} + K(z_t - \hat{z}^k)$ ▷ update landmark mean
    $\Sigma^k_{j,t} = (I - KH)\Sigma^k_{j,t-1}$ ▷ update landmark covariance
    $\omega_k = \mid 2\pi Q \mid^{-\frac{1}{2}} exp\{-\frac{1}{2}(z_t - \hat{z}^k)^T Q^{-1} (z_t - \hat{z}^k)\}$ ▷ update importance factor for each particle
  for *all unobserved features* j' **do**
    $(\mu^k_{j',t}, \Sigma^k_{j',t}) = (\mu^k_{j',t-1}, \Sigma^k_{j',t-1})$ leave unchanged
$X_t = resample((x^k_t, (\mu^k_{1,t}, \Sigma^k_{1,t}), ..., \omega^k)_{k=1,...,N})$
return X_t

3.2 Testing Results

Looking at the mean estimated poses of the robot and landmarks at the 500th time step, the estimated locations of landmarks along the beginning of the path are more accurate than those close to the end of the path. The figure 4 also shows that more particles will yield smaller errors of robot poses over all time steps.In next section, we are interested in comparing the errors of robot and landmark poses in a single robot case with those in a two-robot case.

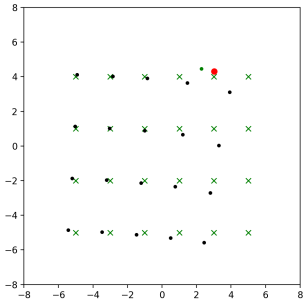

Figure 3: Single robot pose estimate at time step 500

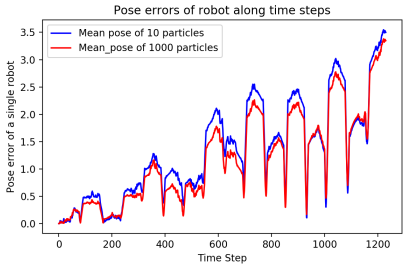

Figure 4: Pose errors of robot along time steps

4 Multiple Robots FastSLAM

Intuitively, using multiple robots to perform SLAM simultaneously will unquestionably boost the efficiency of FastSLAM. The main challenge of multiple robots FastSLAM lies in exchanging information among robots, and what information should each robot share with each other. In this section we present a straight forward but reasonable

approach to implementing multiple robots FastSLAM. The potency of this approach will be demonstrated with visualization and error plots.

Figure 5: *Robot pose errors: single robot vs. two robots*

4.1 Algorithm

There are two main approaches to this problem; first, we could treat the robots as individuals and share their landmarks information when necessary; second, we could combine the robots into one integrated system and each resample/update will occur on the robots as a whole.

While some existing multiple robots project choose to treat the robots as a single system, after we tested some implementations and considered the reasoning behind this approach, we believe it will in fact lower the accuracy of individual robots compared to them working individually. Assume two robots that observed entirely different sets of landmarks throughout their time steps. If the algorithm somehow influence one robot's weight with the other robot's observations, which are entirely irrelevant, it will create additional unnecessary for that robot.

Based on the above reasoning, we choose to consider each robot as individually working SLAM robot. When the robots are all observing different sets of landmarks, they should have the exact same performance as when they are working alone. Each robot should store a list of its own landmark estimations. When multiple robots observe the same landmark, they share information by calculating the mean of all the estimations and co-variance matrices of that landmark and update the values to these robots. For example, say we have robots R1, R2, R3. Assume R1 and R2 have never seen landmark L1 and R3 has seen L1 before. When R1 first observe L1, it will initialize an estimation, e1, for this landmark using the steps from the previous section, then extract the estimation, e3, of L1 that was stored in R3. We calculate the mean of e1 and e3 and store the resulting estimation in both the landmark lists of R1 and R3. The same also procedure occurs every time when the a robot performs EKF update on an observed landmark.

No pseudo-code will be provided in this section because the only difference of our multiple robots SLAM is when combining landmark information of observed landmarks, otherwise same as single robot version.

4.2 Testing Results

First, it is important to note a trivial but valid advantage of using multiple robots. The speed of environment coverage will almost always be faster than using a single robot, unless we provide each robot with the exact same set of initial positions, controls, and observations. Below is a visualization of two robots starting from opposite positions, with respect to (0, 0), and the exact same control inputs:

This figure at the very least shows two robots simultaneously performing SLAM on separate portions of the map. We could potentially combine the two landmark lists into a full coverage mapping.

What's more exciting is to use the information from both robots to refine their individual estimations of robot positions and landmarks. The following error plot compares errors of using a single robot to the errors of both robots in the multiple robots scenario.

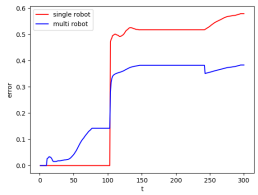

Figure 6: landmark 12 errors

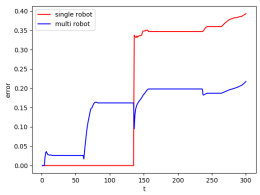

Figure 7: landmark 13 errors

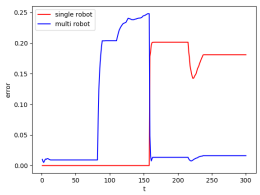

Figure 8: landmark 14 errors

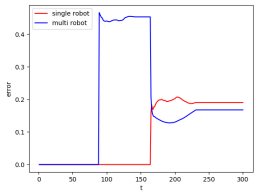

Figure 9: landmark 15 errors

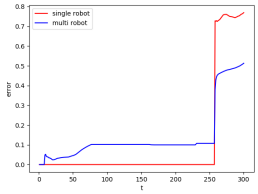

Figure 10: landmark 16 errors

We can observe, from Figure 5, that for the first 170 time steps, the errors of using multiple robots and single robot remain mostly the same; this is because the two robots have not observed any common landmark yet, which implies they are working purely as individuals.

After 170 time steps, the two robots started to observe common landmarks from the middle two columns and performed landmark merging. Unfortunately, we can see that the errors of robot positions fluctuated and the decrease of errors from multiple robots was not consistent. On the other hand, if we look at Figure 6-10, which are the errors of five of the commonly observed landmark estimations, we can observe that the errors of using single robot were evidently higher than that of using two robots after the landmarks became commonly observed. (Zero error means the landmark has not been observed yet.)

It is important to note that, because the two robots have directly opposite paths, the means of their estimations for a single common landmark will be close to the actual location of that landmark; the two robots can thereby have better estimates on the commonly observed landmarks.

5 Discussion

An obvious drawback of FastSLAM is that, unlike EKF-SLAM, which provides elegant close forms for prediction, update, and augmentation steps, it depends on the weighted mean of particles; FastSLAM's performance is thereby largely influenced by the number of particles. Because each particle needs to store its own estimations for all observed landmarks, as the number of particles gets large, the computation power required grows significantly.

Another remark worthy of pointing out is the way we perform landmark merging in this project. It is highly based on the fact that the two robots have directly opposite paths. While this method could still be generalize to other multiple robots scenarios, the enhancements might not be as obvious. This leads to the potential future works section.

6 Future Works

As mentioned in the previous section, there are quite a few other information merging techniques yet to explore. We suggest starting with weighted mean of estimations from different robots; the weights of each could be calculated by the co-variance matrices; the lower the uncertainty of an estimation, the higher its weight. Adding more robots will also

improve the robustness of SLAM by having more refined estimations of landmarks. We suggest designing paths that are mostly symmetric with slight overlapping to allow for faster mapping coverage.

Extending the implementation to actual autonomous robots, or drones, will create loads of challenging projects with great potential. Provided the working algorithm, the keys will most likely be deriving valid control and measurement models and actual testing.

7 Conclusion

In the first half of the research, the team performed deep studies on the theoretical background of single robot FastSLAM. We implemented the algorithm from Probabilistic Robotics by Sebastian Thrun as our base SLAM program with single robot. We generated error plots using different numbers of particles to verify the rationale of our program and show it can perform SLAM with decent potency, though not as robust as EKF-SLAM.

In the second half of the project, we tried out numerous implementations of multiple robots SLAM, but the majority of them did not provide enhancement to single robot scenario, which is why we choose to leave them out of this report. We decided to let the robots work as individuals and only merge landmarks when they are observed by multiple landmarks. Resulting error plots demonstrate that the performance of multiple robots SLAM are the same as single robot when they observed entirely different sets of landmarks and the performance of the former are better than the latter when there are overlapping observations. We pointed out the disadvantages of FastSLAM, that it requires a lot more computation power to work with large number of particles and suggested using weighted mean to merge overlapping landmarks.

Though the method we decided on at last was not based on solid theoretical background, the project is still able to demonstrate the effectiveness of multiple robot SLAM. We believe with further fine tuning and implementation on actual robots, people could conduct many more exciting relevant projects.

8 Acknowledgements

We sincerely appreciate Professor Matthew R. Walter and Charles Schaff for designing and teaching such an enlightening course. We also hope to thank them for all the help and guidance throughout the term.

References

[1] Chen, Shi-Ming, et al. "Multirobot FastSLAM Algorithm Based on Landmark Consistency Correction." *Mathematical Problems in Engineering*, vol.2014, 2014.
[2] Montemerlo,Micheal, et al. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *Proc. AAAI Nat'l Conf. Artificial Intelligence*, 2002.
[3] Stachniss,Cyrill. 'FastSLAM - Feature-based SLAM with Particle Filters.' Robot Mapping. The Albert Ludwig University of Freiburg, Baden-Württemberg. Jan, 2013.
[4] Thrun, Sebastian, et al. *Probabilistic Robotics*. The MIT Press, 2005.