FastSLAM with Unknown Data Association and Association Error Analysis

Xiao Li, Wei-Chun Lu, Bhushan Ghadge and Ryan Wunderly

***Abstract*—This paper reports on the implementation and discussion of the FastSLAM algorithms. Three FastSLAM algorithms, (i) FastSLAM with known data association, and (ii) FastSLAM1.0 and (iii) FastSLAM2.0 with unknown data association, are implemented and tested in the landmark-based scenario. In addition, a mapping matrix norm induced error is proposed to estimate and analyze the SLAM performance with unknown data association. Finally, we tried to implement on the real dataset and provides some discussion and comments.**

***Index Terms*—mobile robots, extended Kalman filter, particle filter, FastSLAM, data association**

I. Introduction

Simultaneous localization and mapping (SLAM) problem attracts large attention in recent two decades. As unmanned vehicles and exploring robots that operate in danger uninhabited region are rapidly developed, SLAM problem becomes more important for the navigation in unknown environments. SLAM problem includes two parts, mapping and recovering robot pose with observation data. The common approach to SLAM problem is extended Kalman filter (EKF) SLAM [1], [2], which use EKF in both mapping and locating problem. However, EKF-SLAM has two main limitations: (i) its algorithm is computational and (ii) Gaussian cannot represent multi-modal distribution of the different data association hypothesis. FastSLAM [3] algorithm, which integrates EKF and particle filter, offers advantages over EKF-SLAM. The particle filter, which is utilized to recover robot pose, can handle the multi-hypothesis of different data association decisions. On the other hand, FastSLAM estimates the map feature by EKFs. [3] also proposed a method of an efficient tree to improve the computational problem. In general, FastSLAM is conceptually simple, easy to implement, and thus widely used.

In NAVARCH 568, we have implemented the localization using EKF, unscented Kalman filter, particle filter, and in-EKF. Based on this experiment, we would like to investigate further in SLAM problem. In this paper, we implement the algorithms of basic FastSLAM with known data association, and FastSLAM1.0 and FastSLAM2.0 with unknown data association in MATLAB. FastSLAM2.0 is the improved version of FastSLAM1.0, which overcomes the problem of mismatch between proposal and posterior distribution. The three algorithm are tested in the landmark-based scenario. The robot moves with linear and angular control inputs, and the sensor gives noisy observations with range and bearing measurements of the sensed landmarks. Also, we proposed a mapping matrix norm induced error to estimate and discuss the SLAM performance with unknown data association. Finally, we tried to implement on the real dataset and provides some discussion about our experience.

The remainder of the report is organized as follows. Section II describes the model and algorithm of FastSLAM with known data association. Section III states the algorithms, FastSLAM1.0 and FastSLAM2.0, with unknown data association. Section IV presents the results of the three algorithm in the landmark-based scenario. Section V discusses some lessons in our implementation, and section VI concludes the work.

The authors are with the University of Michigan, Ann Arbor, MI 48109, USA. hsiaoli@umich.edu, wclu@umich.edu, bghadge@umich.edu, rywunder@umich.edu

II. Preliminaries And FastSLAM With Known Data Association

A. Model Setup

Derived from FastSLAM algorithm in [4], the proposed FastSLAM model is operated under a feature-based map. For the robot itself, at each time step t, the robot position and pose can be uniquely determined by its state vector x_t (e.g. $x_t = [x\ y\ \theta]^T$). With control input u_t, the robot at x_{t-1} will move to a new pose x_t. This motion can be formulated as a linear or nonlinear map as:

$$x_t = g(x_{t-1}, u_t), \tag{1}$$

where the control input is subjected to an additive noise $w_t \sim \mathcal{N}(0, M(u_t))$.

The map is consisting of multiple features (e.g. landmarks) $\mathcal{L}$. At each time step, the robot moves from one pose to another and carry out measurement task to the surroundings. For those features $l = \{l^j\} \in \mathcal{L}$ within robot perceptual range, corresponding measurements $z_t = \{z^j\}$ will have high possibility to be obtained by the robot. The measurement model can be illustrated as:

$$z_t = h(x_t, l) \text{ and } z_t^j = h(x_t, l^j), \tag{2}$$

where a single measurement is independent of others and subjected to an additive Gaussian noise $v_t \sim \mathcal{N}(0, Q_t)$.

For the FastSLAM task, we do not have prior-knowledge about either robot pose or features' positions. By means of Monte-Carlo process, we use Rao-Blackwellized particle filter to track poses of both robot and features. For the FastSLAM with known data association, we know how many features we have in the map while the correspondence between measurements and features is deterministic. Therefore, contents of each particle are certain at the first stage. For the map consisting of

N features, at time step t, the k'th ($k = 1, ..., M$) particle is composed of robot pose $x_t^{[k]}$ and N extended Kalman filters (EKF, i.e. $\left\langle \mu_{j,t}^{[k]}, \Sigma_{j,t}^{[k]} \right\rangle$ where $i = 1, 2, ..., N$) which keep tracking of the estimation position $\mu_{j,t}^{[k]}$ and covariance $\Sigma_{j,t}^{[k]}$ associated with feature $l^j \in \mathcal{L}$. In a word, in each time step, we have a particle set $Y_t = \{Y_t^{[k]}\}$ which can be formulated as:

$$Y_t^{[k]} = \left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, ..., \left\langle \mu_{N,t}^{[k]}, \Sigma_{N,t}^{[k]} \right\rangle \right\rangle$$

B. Main Recursive Algorithm

The complete FastSLAM algorithm with known data association is shown in Algorithm 1. The main algorithm of FastSLAM can be divided to motion model update, measurement correction and resampling. At beginning of each recursive iteration which correspond to robot moving to new position, we sample new robot $x_t^{[k]}$ pose for each particle out of distribution (line 2 in Algorithm 1) derived from motion model in Eq. (1):

Algorithm 1 FastSLAM_known

Require: $Y_{t-1} = \{Y_{t-1}^{[k]}\}$, $w_t = \{w_t^{[k]}\}$, $z_t = \{z_t^j\}$, u_t
1: **for** $k = 1 \rightarrow M$ **do**
2: $x_t^{[k]} \sim p(x_t | x_{t-1}^{[k]}, u_t)$ ▷ motion model update
3: **for** $z_t^j \in z_t$ **do** ▷ multiple measurement update
4: **if** feature j is new **then** ▷ initialize new features
5: $\mu_{j,t}^{[k]} = h^{-1}(z_t^j, x_t^{[k]})$ ▷ initialize mean
6: $H = h'(\mu_{j,t}^{[k]}, x_t^{[k]})$ ▷ measurement Jacobian
7: $\Sigma_{j,t}^{[k]} = H^{-1} Q_t H^{-T}$ ▷ initialize covariance
8: **else** ▷ EKF update observed features
9: $\left\langle \mu_{j,t}^{[k]}, \Sigma_{j,t}^{[k]} \right\rangle$, $w_t^{[k]}$ = EKF_update($x_t^{[k]}$, z_t^j,... $\left\langle \mu_{j,t-1}^{[k]}, \Sigma_{j,t-1}^{[k]} \right\rangle$, $w_{t-1}^{[k]}$)
10: **end if**
11: **for** all other feature $j' \neq j$ **do** ▷ unobserved
12: $\left\langle \mu_{j',t}^{[k]}, \Sigma_{j',t}^{[k]} \right\rangle = \left\langle \mu_{j',t-1}^{[k]}, \Sigma_{j',t-1}^{[k]} \right\rangle$
13: **end for**
14: **end for**
15: **end for**
16: $n_{eff} = 1/sum(\{(w_t^{[k]})^2\})$ ▷ effective particles' number
17: **if** $n_{eff} < n_t$ **then**
18: Y_t, w_t = resampling(Y_t, w_t) ▷ Resampling
19: **end if**
20: **return** $Y_t = \{Y_t^{[k]}\}$, $w_t = \{w_t^{[k]}\}$

As we know the measurement z_t^j correspondence to feature $\left\langle \mu_{j,t}^{[k]}, \Sigma_{j,t}^{[k]} \right\rangle$, we can readily determine if a new feature has been observed. For each new observed features, we need to initialize their tracking EKF by estimated mean out of the inverse measurement model:

$$\mu_{j,t}^{[k]} = h^{-1}(z_t^j, x_t^{[k]}) \tag{3}$$

The initial covariance is estimated using uncertainty propagation through 1st order Taylor Expansion of nonlinear functions. Shown in Algorithm 4 in Appendix, for previously observed features, we update the mean and covariance by a classical EKF manner and the weight $w^{[k]}$ for corresponding particles.

As for other features which do not have associated measurements at this time step, their EKF mean and covariance remain unchanged. After update all the M particles, if the numbers of effective particle n_{eff} is lower than the threshold n_t , we need to carry out a low variance re-sampling for all the M particles ($Y_t^{[k]} \in Y_t$ where $k = 1, ..., M$) to discard dubious outliers by Algorithm 5 attached in Appendix.

III. FastSLAM With Unknown Data Association

A. FastSLAM1.0

As proposed by [4], the key idea of FastSLAM with unknown data association is to determine the association in a statistical manner. With multiple measurements at each time step, by Rao-Blackwell theorem, the prediction by measurement correction as well as the assignment of data association can be refined significantly. Different from the particle in II, the particle in FastSLAM1.0 introduces a new attribute $N_t^{[k]}$ which is the number of maintained features for k'th particle at time step t. Therefore, the particle can be formulated as:

$$Y_t^{[k]} = \left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, ..., \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]} \right\rangle \right\rangle$$

Algorithm 2 FastSLAM1.0

Require: $Y_{t-1} = \{Y_{t-1}^{[k]}\}$, $w_t = \{w_t^{[k]}\}$, $z_t = \{z_t^j\}$, u_t
1: **for** $k = 1 \rightarrow M$ **do**
2: $x_t^{[k]} \sim p(x_t | x_{t-1}^{[k]}, u_t)$ ▷ motion model update
3: **for** $z_t^i \in z_t$ **do**
4: **for** $j = 1 \rightarrow N_{t-1}^{[k]}$ **do** ▷ measurement likelihood
5: ${\hat{z}_t}^j = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ ▷ prediction
6: $H_j = h'(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ ▷ Jacobian
7: $Q_j = H \Sigma_{j,t-1}^{[k]} H^T + Q_t$ ▷ Covariance
8: $w_j = p(z_t^i \mid z_t^i \sim \mathcal{N}({\hat{z}_t}^j, Q))$ ▷ likelihood
9: **end for**
10: $w_{1+N_{t-1}^{[k]}} = p_0$ ▷ new feature: default importance
11: $w_t^{[k]} = max(w_j)$ ▷ ML correspondence
12: $\hat{c} = argmax_{idx}(w_j)$ ▷ ML index: $i = \hat{c}$
13: $N_t^{[k]} = max(N_{t-1}^{[k]}, \hat{c})$ ▷ update features' number
14: **for** $j = 1 \rightarrow N_t^{[k]}$ **do** ▷ measurement update EKF
15: **if** $j = \hat{c} = 1 + N_{t-1}^{[k]}$ **then**
16: *initialize new feature's EKF
17: $i_{j,t}^{[k]} = 1$ ▷ initialize counter variable
18: **else if** $j = \hat{c} < N_{t-1}^{[k]}$ **then**
19: *EKF update
20: $i_{j,t}^{[k]} = i_{j,t-1}^{[k]} + 1$ ▷ update counter
21: **else** Remain unchanged
22: **end if**
23: **end for**
24: **end for**

for $j = 1 \rightarrow N_t^{[k]}$ **do**
  if $\mu_{j,t-1}^{[k]}$ is in perceptual range of $x_t^{[k]}$ and $\forall z_t^i \in z_t,\ i \neq j$ **then**
    $i_{j,t}^{[k]} = i_{j,t-1}^{[k]} - 1$ ▷ false feature measurement
    if $i_{j,t}^{[k]} < 0$ **then**
      delete feature j ▷ remove dubious features
    end if
  else Remain unchange
  end if
end for
end for
*resampling measure
return $Y_t = \{Y_t^{[k]}\},\ w_t = \{w_t^{[k]}\}$

*denotes for identical measures as in Algorithm 1.

The correspondence for each measurements is determined in a maximum likelihood (ML) fashion. The algorithm assigns weight of correspondence to each exist features by its measurement likelihood. As the measurement is not perfect and could be false sometime, the dubious measurement might introduce significantly disturbance to features' position estimation which will eventually lead to failure. The FastSLAM1.0 introduces a counter variable $i_{j,t}$ to record accumulative numbers of measurements associated with feature j by the time step t. If a certain feature falls into robots perceptual range and there is no correspondent measurements, the algorithm will degrade its associated counter i_j to a point that the counter is negative. In that case, we can rule out the dubious feature j with high confidence.

B. FastSLAM2.0

FastSLAM2.0 was proposed to solve the important problem of FastSLAM1.0. The critical idea of FastSLAM2.0 is sampling poses from the proposal distribution by taking the measurement z_t in to consideration. For FastSLAM1.0, the proposal distribution is based on robot's motion model, which may be largely different from the posterior distribution. Hence, many particles distribute in low posterior probability locations. In FastSLAM2.0, to improve this situation, the pose $x_t^{[k]}$ is drawn from the posterior

$$x_t^{[k]} \sim p(x_t | x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t})$$

where c_t is the correspondence. Note that the samples based not only on the control input u_t but also on the measurement z_t, and thus its mathematical derivation is more involved. More detail can be found in [4]. The main algorithm is shown in Algorithm 3. Here introduce two Jacobians $H_{x,j}$ and $H_{m,j}$ (line 6-7 in Algorithm 3), which are the Jacobians with respect to pose x_t and map feature m_{c_t} respectively, to linearize the sensor model h using first order Taylor approximation.

$$h(m_{c_t}, x_t) \approx \hat{z}_t^{[k]} + H_m(m_{c_t} - \mu_{c_t,t-1}^{[k]}) + H_x(x_t - \hat{x}_t^{[k]})$$

Algorithm 3 FastSLAM2.0

Require: $Y_{t-1} = \{Y_{t-1}^{[k]}\},\ w_t = \{w_t^{[k]}\},\ z_t = \{z_t^j\},\ u_t$
1: **for** $k = 1 \rightarrow M$ **do**
2: **for** $z_t^i \in z_t$ **do**
3: **for** $j = 1 \rightarrow N_{t-1}^{[k]}$ **do** ▷ measurement likelihood
4: $\hat{x}_{t,j} = g(x_{t-1}^{[k]},\ u_t)$ ▷ motion prediction
5: $\bar{z}_t{}^j = h(\mu_{j,t-1}^{[k]},\ \hat{x}_{t,j})$ ▷ sensor prediction
6: $H_{x,j} = \nabla_{x_t} h(\mu_{j,t-1}^{[k]},\ \hat{x}_{t,j})$ ▷ pose Jacobian
7: $H_{m,j} = \nabla_{m_j} h(\mu_{j,t-1}^{[k]},\ \hat{x}_{t,j})$ ▷ map Jacobian
8: $Q_j = H_{m,j} \Sigma_{j,t-1}^{[k]} H_{m,j}^T + Q_t$
9: $\Sigma_{x,j} = [H_{x,j}^T Q_j^{-1} H_{x,j} + M(u_t)]^{-1}$
10: $\mu_{x_t,j} = \Sigma_{x,j} H_{x,j}^T Q_j^{-1} (z_t^i - \bar{z}_t{}^j) + \hat{x}_{t,j}$
11: $x_{t,j}^{[k]} \sim \mathcal{N}(\mu_{x_t,j},\ \Sigma_{x,j})$ ▷ sample pose
12: $\hat{z}_{t,j} = h(\mu_{j,t-1}^{[k]},\ x_{t,j}^{[k]})$
13: $w_j = p(z_t^i \mid z_t^i \sim \mathcal{N}(\hat{z}_{t,j}, Q_j))$ ▷ likelihood
14: **end for**
15: $w_{1+N_{t-1}^{[k]}} = p_0$ ▷ new feature: default importance
16: $\hat{c} = argmax_{idx}(w_j)$ ▷ ML index: $i = \hat{c}$
17: $N_t^{[k]} = max(N_{t-1}^{[k]}, \hat{c})$ ▷ update features' number
18: **for** $j = 1 \rightarrow N_t^{[k]}$ **do**
19: **if** $j = \hat{c} = 1 + N_{t-1}^{[k]}$ **then**
20: $x_{t,i}^{[k]} \sim p(x_t | x_{t-1}^{[k]}, u_t)$ ▷ sample pose
21: $w_i^{[k]} = p_0$
22: *initialize new feature's EKF
23: $i_{j,t}^{[k]} = 1$ ▷ initialize counter variable
24: **else if** $j = \hat{c} \leq N_{t-1}^{[k]}$ **then**
25: $x_{t,i}^{[k]} = x_{t,j}^{[k]}$
26: $L = H_{x,j} M(u_t) H_{x,j}^T + Q_j$
27: $w_i^{[k]} = p(z_t^i \mid z_t^i \sim \mathcal{N}(\hat{z}_{t,j}, L))$
28: *EKF update
29: $i_{j,t}^{[k]} = i_{j,t-1}^{[k]} + 1$ ▷ update counter
30: **else** Remain unchanged
31: **end if**
32: **end for**
33: **end for**
34: $x_t^{[k]} = x_{t,i}^{[k]} / \sum_i (x_{t,i}^{[k]})$ ▷ update particle pose
35: $w^{[k]} = w^{[k]} \prod_i w_i^{[k]}$ ▷ update particle weight
36: **for** $j = 1 \rightarrow N_t^{[k]}$ **do**
37: **if** $\mu_{j,t-1}^{[k]}$ is in perceptual range of $x_t^{[k]}$ and $\forall z_t^i \in z_t,\ i \neq j$ **then**
38: $i_{j,t}^{[k]} = i_{j,t-1}^{[k]} - 1$ ▷ false feature measurement
39: **if** $i_{j,t}^{[k]} < 0$ **then**
40: delete feature j ▷ remove dubious features
41: **end if**
42: **else** Remain unchange
43: **end if**
44: **end for**
45: **end for**
46: *resampling measure if needed
47: **return** $Y_t = \{Y_t^{[k]}\},\ w_t = \{w_t^{[k]}\}$

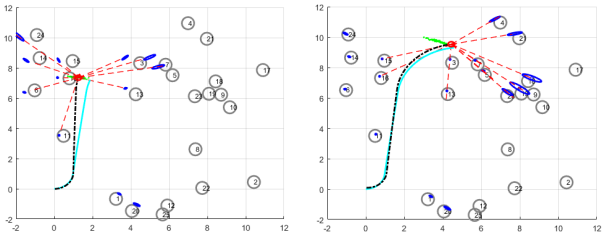
*denotes for identical measures as in Algorithm 1.

Under this linear approximation, the desired sampling pose distribution is a Gaussian $\mathcal{N}(\mu_{x_t,j},\ \Sigma_{x,j})$ (line 9-11 in Algorithm 3), and the importance factor $w_i^{[k]}$ is given by another Gaussian distribution $\mathcal{N}(\hat{z}_{t,j}, L)$ (line 26-27 in Algorithm 3). Therefore, the robot pose can be sampled from the posterior.

IV. Experimental Results

A. FastSLAM With Known Data Association

The key feature of FastSLAM is highly elaborated by the Monte-Carlo sampling which is embedded in particle filter. We used a small self-generate odometry data as well as the plotting functions from [5]. The trajectory of highest weight particle is plotted in in Fig.1 marked with dash line in black. The particle weight is constantly corrected and re-weighted by new incoming measurements. With new noisy measurements, apart from making feature mapping bias, the particles' weight will be wrongly manipulated by outlier measurements. As we can see from Fig.1.(a), the best-weight trajectory is immediately drifted away from the ground true by new landmark measurements. After few more steps of measurements on new occurred landmarks, the particle weights become stable and converge to ground true trajectory in Fig.1.(b).



(a) Trajectory drift (b) Correction by observation

Fig. 1: Particle filter correction demonstration on a 14×14 scaled map with 25 landmarks (ground true trajectory (Cyan), trajectory of best particle (black dash), 100 particles (green), robot (red ellipse), sensor measurement line (red dash),landmarks (grey circle) and Gaussian ellipse estimation of landmarks (blue)).

Shown in Fig.2, at terminal stage, the robot estimated trajectory in black dash line is pretty close to the ground true trajectory.

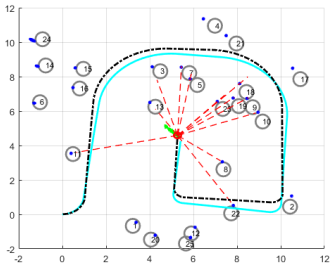


Fig. 2: Final localization and mapping result.

Looking into the mapping error, shown in Fig.3, we plot observed landmarks along with their mean squared position estimation error. As we can see from the figure, at the very beginning, the error in terms of magnitude is considerably large. Illustrated by error bar in red, the error varies up and down with large uncertainties. As the robot mapping around the map and having the complete set of features observed, the error converges to a steady value as well as its error bar narrowing down to zero. As for the robot localization error (trajectory error), it's relatively stable and well-contained below $15m$.

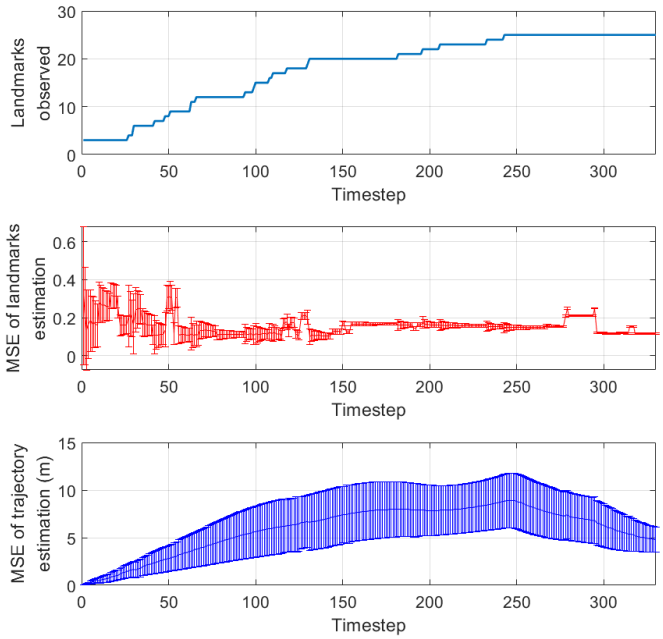


Fig. 3: Mapping and localization mean squared error.

B. Matrix-Norm-Induced Error in Mapping With Unknown Data Association

In IV-A, the measurement data association is given. Therefore, the error calculation can be readily done landmarks-wise. However, with unknown data association, the estimation error can not be quantified by position errors only where the data association error should be taken into account. For an instance, in FastSLAM1.0 and FastSLAM2.0, dubious features might occur and be counted as new features for a short period (will be further discussed with examples in IV-C). In that case, this type of data association errors are also essential for estimation error calculation.

By means of problem formulation, from the beginning to time step t, we have in total K numbers of features appeared in robot's perceptual field which should be the theoretical numbers of features maintained by the algorithm. The theoretical position $[m_x, m_y]^T$ of K landmarks can be formulated into a matrix as

$$\mathcal{M} = \begin{bmatrix} m_x^1 & m_x^2 & \cdots & m_x^K \\ m_y^1 & m_y^2 & \cdots & m_y^K \end{bmatrix}.$$

For the estimated $N_t^{[k]}$ numbers of landmarks' position stored in particle k, we can formulate their estimated positions into matrix form

$$\mathcal{U} = \begin{bmatrix} \mu_x^1 & \mu_x^2 & \cdots & \mu_x^{N_t^{[k]}} \\ \mu_y^1 & \mu_y^2 & \cdots & \mu_y^{N_t^{[k]}} \end{bmatrix}.$$

At time step t for particle k, the association A between the theoretical features position $\mathcal{M}$ and the estimated position $\mathcal{U}$ can be expressed as $\mathcal{U}A = \mathcal{M}$. If the mapping is perfect and $\mathcal{U} = \mathcal{M}$, the association matrix A is a identical matrix. Therefore, the least square association can be calculated as

$$A = \mathcal{U}^{+}\mathcal{M}$$

where $\mathcal{U}^{+}$ is the pseud-inverse of $\mathcal{U}$. Moreover, A might not be square as $K \neq N_t^{[k]}$ is possible for reason of dubious measurements. Thus, in order to measure the error of A in matrix logarithm (matrix need to be square) norm form without changing its norm, we apply reduced singular value decomposition (SVD) to A as:

$$A_{N_t^{[k]} \times K} = U_{N_t^{[k]} \times K} \Sigma_{K \times K} V^*_{K \times K}$$

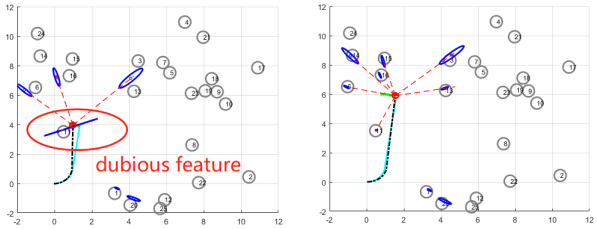
where by properties of SVD $\left\| A_{N_t^{[k]} \times K} \right\|_2 = \|\Sigma_{K \times K}\|_2$. Then, the error Err of estimation of position as well as data association can be expressed as

$$Err = \|logm(\Sigma_{K \times K})\|_2 \tag{4}$$

where $logm$ is matrix logarithm map. Again, if the mapping is perfect and $\mathcal{U} = \mathcal{M}$, the association matrix A is a identical matrix and the error calculated using Eq.(4) should be zero.

C. FastSLAM1.0 With Unknown Data Association

As the data association is not given in this scenario, the algorithm might assign measurements of pre-observed features as new features. Shown in Fig.4, the dubious measurements with polarized covariance ellipse is circle in red, This phenomenon is mainly induced by the corresponding feature counter is varying near zero. Therefore, the feature is eliminated and reinitialized by the algorithm for multiple times.



(a) Dubious features circled (b) Correction by observation

Fig. 4: Particle filter correction with dubious measurements demonstration on a 14×14 scaled map with 25 landmarks and 100 particles (green).

If more observation corresponding to this dubious feature is obtained with robot proceeding, the dubious feature will either converge to stable position or be discarded. Shown in Fig.5, at terminal stage, the robot estimated trajectory in black dash line is pretty close to the ground true trajectory.

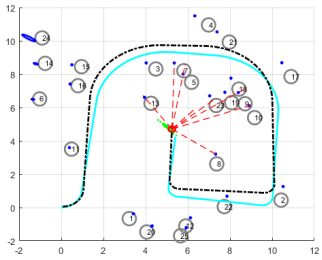


Fig. 5: Final localization and mapping result.

As we can see from Fig.6, the estimated feature numbers is aligning close to theoretical feature numbers with a few jumps over theoretical line where dubious features occurs. At the very beginning, the matrix norm induced mapping error in terms of magnitude is considerably large. As the robot mapping around the map and having the complete set of features observed, the error converges to a steady value. As for the robot localization error (trajectory error), it's relatively stable and well-contained below $15m$.

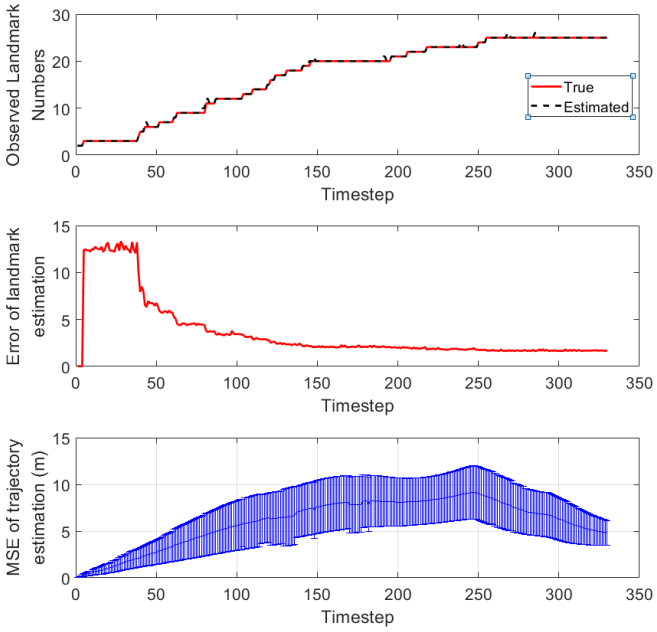


Fig. 6: Mapping matrix norm induced error and localization mean squared error.

D. FastSLAM2.0 With Unknown Data Association

Similar to FastSLAM1.0 with unknown data association, the final result of FastSLAM2.0 is shown in Fig. 7, and the error of landmark estimation and trajectory is shown in Fig. 8.

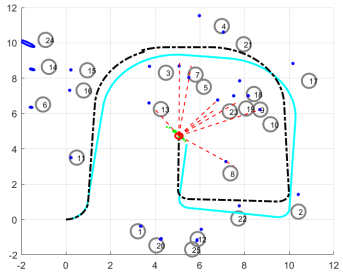


Fig. 7: Final localization and mapping result of FastSLAM2.0.

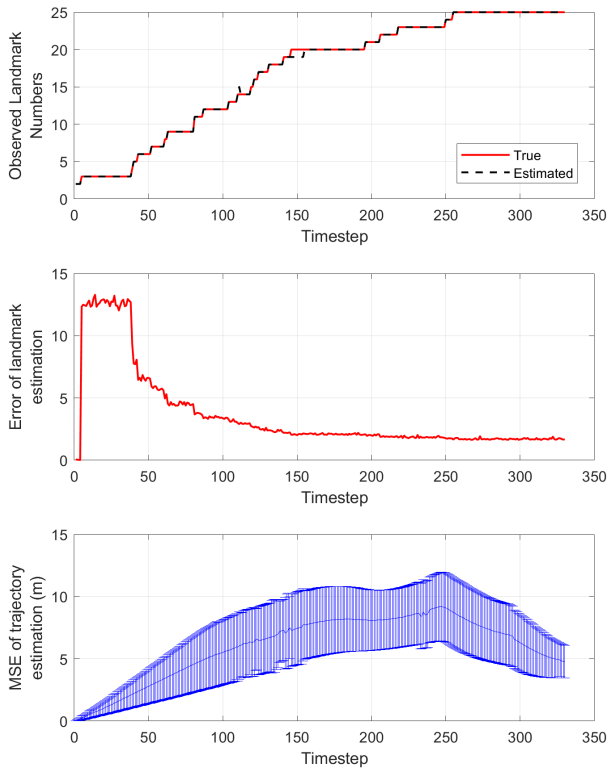
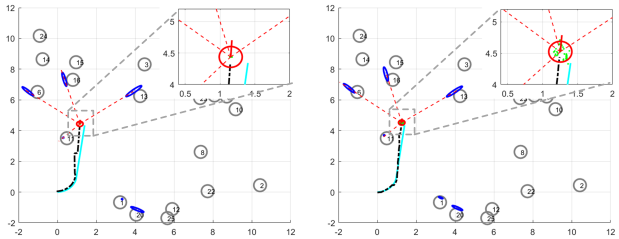


Fig. 8: Mapping matrix norm induced error and localization mean squared error of FastSLAM2.0.

The estimated trajectory in close to the ground truth in Fig. 7, and, from Fig. 8, the terminal trajectory error is about 5 meters. Although the error grows in the middle of the operation, it decreases after the robot senses the observed landmark again and corrects its estimations. The observed feature numbers is almost the same with the theoretical ones. In this case, the small peaks due to the dubious features are less than the results of FastSLAM1.0. The scenario that the theoretical landmark number is larger than estimated one occurs because the robot thinks the two observation corresponding to the same landmark.

However, the difference between FastSLAM1.0 and FastSLAM2.0 is not obviously in this case. This is because the particles are scarcely resampled in this example. The key concept of FastSLAM2.0 to improve the problem of mismatch between proposal and posterior distribution. The mismatch leads to low diversity of particles after resampling when the accuracy of control input is low relative to that of robot's sensor. Thus, to distinguish the difference between FastSLAM1.0 and FastSLAM2.0, we rerun the example by giving larger motion noise and increasing the resample threshold. As shown in Fig. 9, the particles loss the diversity after resampling with FastSLAM1.0, while the particles still maintain its diversity with FastSLAM2.0. From this test, the improvement of FastSLAM2.0 is notable.



(a) low-diversity particles (b) diversity of particles still remain

Fig. 9: The diversity of particles (green) after several resampling with unknown data association (a) FastSLAM1.0 and (b) FastSLAM2.0. algorithm

E. FastSLAM on Victoria Park Dataset

The Victoria Park dataset consists of laser data from a truck. In order to perform SLAM applications the trees are used as landmarks. Trees are difficult in the data association problem since there are not many unique features between trees. For our implementation we adapted the openSLAM fastSLAM code. [6].

In order to solve the data association problem we used the nearest neighbor algorithm. Since FastSLAM uses a particle filter each particle can have a corresponding guess of what landmark it is associated with. Incorrect guesses disappear in the resampling space. The plotted trajectory is fairly consistent with the ground truth. It shows that FastSLAM with data association through gated nearest neighbor algorithm works well.

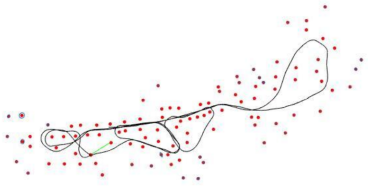


Fig. 10: Victoria Park trajectory [7].

V. Discussion And Limitations

For the unknown data association, [3] also proposed the method of an efficient tree to enhance the efficiency of the algorithm. However, we did not include this approach in our algorithms. Thus, it ran slowly when dealing with large amount of data. It would be better to rewrite and implement the efficient tree algorithm.

For the implementation on VictoriaPark dataset, we first tried to use our own FastSLAM implementation and wanted to compare it with our result of iSAM method using gtsam, as shown in Fig. 12 attached in Appendix. However, our code was inefficient and the data association was incorrect. We attempted to match landmarks according to their tree diameters from the laser data. However, the tree diameters were noisy. To prove that fast slam would work we adapted the code from OpenSLAM.org [6]. This code used the gated nearest neighbor approach for data association. This implementation worked well enough to generate a trajectory for the dataset. This showed us how important data association is when solving the SLAM problem with FastSLAM. It is one of the reasons we did not implement on a newer dataset like KITTI since the data association problem was even harder in a stereo LiDAR dataset without uniform landmarks like trees. Moreover, later, we find that the poor performance mainly attributes to our fastSLAM's ability to reject trajectory noise statistically while loop closure is still need to be implement in the future. To testify our conjecture, we used the ground true pose of the car in [7] to test our mapping performance only.

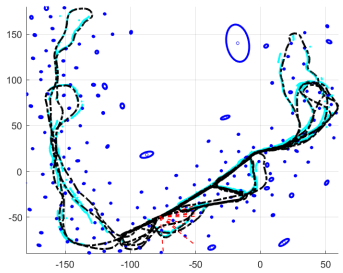


Fig. 11: Testing mapping performance of FastSLAM2.0.

Shown from Fig. 11, the mapping performance is quite satisfactory by achieving convergence for a majority of features' locations. Meanwhile, mapping results if some trees are still noisy by having large covariance ellipse which is caused by not having enough measurement updates.

VI. Conclusion

This paper reports on our study and implementation about FastSLAM algorithms. We implemented three algorithms in FastSLAM family, basic FastSLAM with known data association, and FastSLAM1.0 and FastSLAM2.0 with unknown data association. We also investigated the difference and improvement between them by simulated example. The difference of FastSLAM and FastSLAM1.0 is the unknown of the landmark number, and thus FastSLAM1.0 developed the approach to determine the corresponding landmarks by maximum likelihood (ML) and introduce a mechanism to add new features and rule out dubious ones. This mechanism can be obviously seen in the simulated example. FastSLAM2.0 improved the mismatch between proposal and posterior distribution in FastSLAM1.0. In the simulated results, FastSLAM2.0 evidently still maintain particle diversity after resampling comparing to FastSLAM1.0. In addition, we also try to implement our algorithms on real dataset. However, for the Victoria Park dataset, we still need to improve our efficiency of out algorithm and tuning the parameters. Our algorithm did not work on the Victoria Park dataset since we solved the data association problem incorrectly. We relied on the diameter of the tree to give us information on the "ID" of the landmark. This was incorrect and we needed to get a nearest neighbor approach working. Because of this we relied on results from a separate modified github repo for the victoria park dataset. That repo used FastSLAM with a gated nearest neighbor technique. This worked well even with the uniformity of the trees in victoria park. We learned that solving the data association problem is a key component of getting FastSLAM to work well. Solving data association is sometimes non-trivial like it is with newer datasets like KITTI, which make FastSLAM harder to use than visual SLAM algorithms like ORB-SLAM on those datasets.

Appendix

Github: https://github.com/XiaoLiSean/fastSLAM

Video: https://www.youtube.com/watch?v=l1vEpVRktko

A. Pseudo-code for EKF Update

Algorithm 4 EKF_update

Require: $x_t^{[k]}$, z_t^j, $\left\langle \mu_{j,t-1}^{[k]}, \Sigma_{j,t-1}^{[k]} \right\rangle$, $w_{t-1}^{[k]}$

$\hat{z_t}^j = h(\mu_{j,t-1}^{[k]},\ x_t^{[k]})$ ▷ estimated measurement

$H = h'(\mu_{j,t-1}^{[k]},\ x_t^{[k]})$ ▷ measurement Jacobian

$Q = H\Sigma_{j,t-1}^{[k]}H^T + Q_t$ ▷ measurement covariance

$K = \Sigma_{j,t-1}^{[k]}H^T Q^{-1}$ ▷ Kalman gain

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t^j - \hat{z_t}^j)$ ▷ update mean

$\Sigma_{j,t}^{[k]} = (I - KH)\Sigma_{j,t-1}^{[k]}$ ▷ update covariance

$w_t^{[k]} = w_{t-1}^{[k]} \cdot p(z_t^j \mid z_t^j \sim \mathcal{N}(\hat{z_t}^j, Q))$ ▷ update weight

return $\left\langle \mu_{j,t}^{[k]}, \Sigma_{j,t}^{[k]} \right\rangle$, $w_t^{[k]}$

B. Pseudo-code for Particle Resampling

Algorithm 5 resampling

Require: $Y_t = \{Y_t^{[k]}\}$, $w_t = \{w_t^{[k]}\}$

$W_c \leftarrow$ cumulative sum of $w_t = \{w_t^{[k]}\}$ ▷ generate CDF
$r = rand()/M$ ▷ generate random number in $(0, 1/M)$
$j = 1$ ▷ Dummy index to reassign particles
for $i = 1 \rightarrow M$ **do**
    $u = r + (i-1)/M$ ▷ climb along the CDF
    while $u > W_c(j)$ **do**
        $j = j + 1$
    end while
    $Y_t^{[i]} = Y_t^{[j]}$, $w_t^{[i]} = 1/M$ ▷ Reassign particles
end for
return $Y_t = \{Y_t^{[k]}\}$, $w_t = \{w_t^{[k]}\}$

C. Victoria Park dataset with iSAM

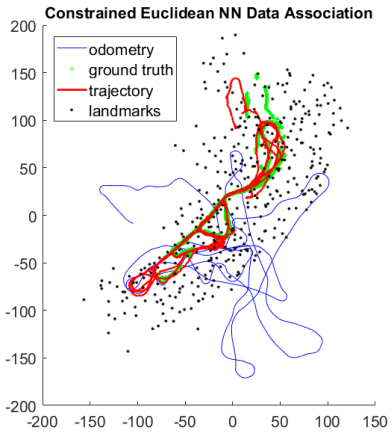


Fig. 12: The result of Victoria Park dataset with iSAM.

ACKNOWLEDGMENT

REFERENCES

[1] P. Moutarlier and R. Chatila, "An experimental system for incremental environment modeling by an autonomous mobile robot," *1st International Symposium on Experimental Robotics*, Jun. 1989.

[2] ——, "Stochastic multisensory data fusion for mobile robot location and environment modeling," *5th Int. Symposium on Robotics Research*, 1989.

[3] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data," *Journal of Machine Learning Research*, vol. 4, May 2004.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, 3rd ed. MIT Press, 2005.

[5] "Slam-algorithms-octave," https://github.com/kiran-mohan/SLAM-Algorithms-Octave.

[6] "Openslam github repository," https://github.com/OpenSLAM-org/openslam_ufastslam/tree/master/code.

[7] J. Guivant and E. Nebot, "Simultaneous localization and map building: Test case for outdoor applications," in *IEEE Int. Conference on Robotics and Automation*, 2002.