Multi-Robot Simultaneous Localization and Mapping (Multi-SLAM)

Kai-Chieh Ma, Zhibei Ma

***Abstract*— In this project, we are interested in the extension of Simultaneous Localization and Mapping (SLAM) to multiple robots. By definition, SLAM is the problem where the robot needs to incrementally build a map of this environment while using this map to estimate its absolute position simultaneously. We successfully implemented both single-robot SLAM and multi-robot SLAM using particle filters. By comparison between these two approaches, we showed that the results from multi-robot SLAM outperform the single-robot SLAM in terms of localization and mapping quality.**

I. Introduction

Localization and mapping are two important topics in mobile robotic applications. Robot localization is the problem of estimating a robot's pose while mapping is the problem of constructing or learning a map which can be used later for the robot to localize itself. Interestingly, these two problems can actually be done simultaneously. In literature, it's known as the simultaneous localization and mapping problem (SLAM). It has been a hot topic in robotics for many years [9]. Motivated by the fact that SLAM problems are typically formulated as a single robot problem, we want to investigate the possibilities of multi-robot SLAM (Multi-SLAM). In the project, we implemented the particle-based filtering algorithms for both single-SLAM and Multi-SLAM and showed that, in terms of robot pose error and map quality, the algorithm performs better than the counterpart algorithm of single SLAM. The details of the approaches, implementation and experiments are documented comprehensively in this report accordingly.

II. Related Work

SLAM also known as Concurrent Mapping and Localization (CML) is one of the fundamental challenges of robotics, dealing with the building the map and determining the location of the robot simultaneously. In the beginning, both the map and the vehicle position are unknown, and the robot has a known kinematic model through the unknown environment, in which there are several landmarks. Several research groups and researchers have worked and are currently working in SLAM, and the most commonly used sensors can be categorized into laser-based, sonar-based, and vision-based systems [1].

Robotic map-building can be traced back to 25 years ago, and since the 1990s probabilistic approaches (i.e. Kalman Filters (KF), Particle Filters (PF) and Expectation Maximization (EM)) have become dominant in SLAM. The three techniques are mathematical derivations of the recursive Bayes rule. The main reason for this probabilistic techniques popularity is the fact that robot mapping is characterized by uncertainty and sensor noise, and probabilistic algorithms tackle the problem by explicitly modeling different sources of noise and their effects on the measurements [8].

Many approaches have been proposed to tackle the single-SLAM problem. Kalman filters are Bayes filters that represent posteriors using Gaussians. Kalman filters are based on the assumption that the state transition and the measurement functions are linear with added Gaussian noise. There are two types variations of KF in SLAM: the Extended Kalman Filter (EKF) and its related Information Filtering (IF). Lots of SLAM approaches are based on EKF [2]. EKF could handle nonlinearities from the real world, by approximating the robot motion model using linear functions while the IF could propagate the inverse of the state error covariance matrix. There are several advantages that the IF filter has over the KF: 1. The data is filtered by simply summing the information matrix and vector, providing more accurate estimates, rather than the approximation [10]; 2. IF are more stable than KF. 3. EKF is quite slow when estimating high dimensional maps.

Beyond these two approaches where posteriors are represented using Gaussians, an particle-based filtering approach was proposed. In literature, it's called Particle Filter (PF), which is a recursive Bayesian filter that is implemented in Monte Carlo simulations. It does the estimation by a set of random point particles representing the Bayesian posterior. The particles make it possible to handle highly nonlinear sensors and non-Gaussian noise. However in dealing with SLAM problems, one fundamental drawback of the classic PF is that each time it detects a landmark, the computational complexity will grow, which is not suitable for map-building. However, there are still several works proposed to deal with the SLAM problem by using a variant of PF, such as FastSLAM [5] and FastSLAM2.0 [6]. The comparison of different filtering algorithms are shown below.

Advantages and Disadvantages of Filtering Approaches		
Filter	Pros	Cons
KF	high convergence handle uncertainty	Gaussian assumption slow in high dimension
IF	stable and simple accurate	data association problem need state estimate
PF	handle nonlinearities handle non-Gaussian noise	growth in complexity

There are few works which have been done to investigate

The authors are with the Department of Computer Science at the University of Southern California, Los Angeles, CA 90089, USA. {kaichiem, zhibeima}@usc.edu

on Multi-SLAM. The most famous one is from Andrew Howard [3], it describes an on-line algorithm for Multi-SLAM with particle filters.

III. Problem Definitions

In is section, we concisely give the formal definitions of the localization, mapping, SLAM and Multi-SLAM problems, respectively.

A. Localization

Localization is the problem of estimating a robot's coordinates relative to an external reference frame. Formally, let $\boldsymbol{x} = [x, y, \theta]^T$ be the pose of a 2D robot in the external reference frame. Given the control input $\boldsymbol{u}_{1:t}$ and the sets of measurements, $\boldsymbol{z}_{1:t}$, where $\boldsymbol{z}_t = \{\boldsymbol{z}_t^1, \boldsymbol{z}_t^2, \ldots, \boldsymbol{z}_t^N\}$ is the set of N measurements at time t and the known map m, the localization problem is to find the posterior distribution of robot poses, $p(\boldsymbol{x}_{0:t}|\boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}, m)$, where $\boldsymbol{x}_{0:t}$ is essentially the trajectory of the robot.

B. Mapping

Mapping is the problem of constructing the map m given the sets of measurements $\boldsymbol{z}_{1:t}$ and the trajectory of the robot defined through the sequence of all poses, $\boldsymbol{x}_{1:t}$. Note that in mapping problems, the trajectory is assumed to be known and fully observable so that controls $\boldsymbol{u}_{1:t}$ play no role in mapping. Mathematically, it's equivalent to finding the posterior over the map, $p(m|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{0:t})$

C. SLAM and Multi-SLAM

From the definitions of the individual localization and mapping problem. We can see that an accurate map is needed for localization while a precise pose estimate is needed for mapping. Therefore, SLAM is inherently a harder problem than either localization or mapping since it manages to solve the both problems simultaneously. For single-SLAM problems, it's finding the joint posterior distribution of the map and robot poses, that is, $p(\boldsymbol{x}_{1:t}, m|\boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t})$, where $\boldsymbol{x}_0$ is the initial pose of the robot. The extension to multiple robots is as follows: $p(\boldsymbol{x}_{1:t}^1, \boldsymbol{x}_{1:t}^2, m|\boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2)$, where the superscripts denotes the robot number. (For notational simplicity, from now on, we will only use two robots in formulations)

IV. Approaches and Methods

In our problem settings, we are dealing with 2D ground robots which are controlled by forward velocity and angular velocity commands. The pose state of the robot is represented by its 2D position and orientation. The map we are constructing is landmark-based. Here we represent the whole map as m which actually contains the 2D position of each landmark in our environment. In the section, we first mathematically describe the motion and measurement model for the robots used in our project, and then we present the algorithms for both Single and Multi SLAM.

A. Motion Model

In our project, the velocity motion model is used. It assumes that we can control a robot through two velocities, a rotational and a translational velocity. Let the translational velocity at time t be v_t, and the rotational velocity ω_t. Hence, we have $\boldsymbol{u}_t = [v_t, \omega_t]^T$. Assume that the robot pose at time t is given by $\boldsymbol{x}_t = [x, y, \theta]^T$, and after Δt time of motion, the ideal robot will be at $\boldsymbol{x}_{t+1} = [x', y', \theta']^T$. Using simple kinematics and trigonometry, one can easily show that

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t\Delta t) \\ \omega_t\Delta t \end{bmatrix} \tag{1}$$

By "ideal" we mean the robot motion is noise-free, but in reality, robot motion is subject to noise. Therefore the actual velocities always differ from the commanded ones by a noise term. More precisely, we assume the actual velocities are given by

$$\begin{bmatrix} \hat{v}_t \\ \hat{\omega}_t \end{bmatrix} = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} + \begin{bmatrix} \epsilon_{\alpha_1 v_t^2 + \alpha_2 \omega_t^2} \\ \epsilon_{\alpha_3 v_t^2 + \alpha_4 \omega_t^2} \end{bmatrix} \tag{2}$$

Here ϵ_{b^2} is a zero-mean Gaussian error with variance b^2. Thus in this model, the true velocity equals the commanded velocity plus some small, additive error (noise). Here we assume the standard deviation of the error is proportional to the commanded velocity. The parameters α_1 to α_4 (with $\alpha_i \geq 0$ for $i = 1, \ldots, 4$) are robot-specific error parameters. Intuitively, the larger the noise, the larger these parameters should be. The motion equation given above implies the radius of the circular segment and the distance traveled is influenced by the control noise. However, the fact that the trajectory is circular is still not affected. The assumption of circular motion can lead to a degeneracy [9]. Therefore, to generalize our motion model accordingly, we assume the robot also performs a rotation $\hat{\gamma}_t$ at its final pose. Thus, the θ' in Eq. (1) is replaced by

$$\theta' = \theta + \hat{\omega}_t\Delta t + \hat{\gamma}_t\Delta t \tag{3}$$

with $\hat{\gamma}_t = \epsilon_{\alpha_5 v_t^2 + \alpha_6 \omega_t^2}$. Again, α_5 and α_6 are the robot-specific parameters determining the variance of the additional rotational noise. In sum, the final motion model is as follows:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{\hat{v}_t}{\hat{\omega}_t}\sin\theta + \frac{\hat{v}_t}{\hat{\omega}_t}\sin(\theta + \hat{\omega}_t\Delta t) \\ \frac{\hat{v}_t}{\hat{\omega}_t}\cos\theta - \frac{\hat{v}_t}{\hat{\omega}_t}\cos(\theta + \hat{\omega}_t\Delta t) \\ \hat{\omega}_t\Delta t + \hat{\gamma}_t\Delta t \end{bmatrix} \tag{4}$$

B. Measurement Model

In our project, the feature-based measurement model is used. It assumes the features have been extracted from the raw measurements and all later calculation are based on these features. The key advantage of this model is the reduction of complexity since the feature space is usually low-dimensional. In particular, the features used in our project are called landmarks. They are physical objects in the workspace for robot navigation. The details of feature extraction will be discussed in V-A. Here we simply assume the feature we get contains the range and the bearing of the landmark relative

to the robot's local coordinate frame. We also assume that the correspondence of landmarks is known and exact after feature extraction (That is, no data association issues in our implementation.). Recall that the map we are constructing is a set of N features, $m = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_N\}$, where $\boldsymbol{m}_i$ is the coordinate in the global frame. The relation between the local frame feature and the global coordinate can be derived using standard geometric laws. Specifically, the measurement model of i-th feature at time t with its corresponding i-th landmark in the map is given by

$$\begin{bmatrix} r_t^i \\ \phi_t^i \end{bmatrix} = \begin{bmatrix} \sqrt{(\boldsymbol{m}_{i,x} - x)^2 + (\boldsymbol{m}_{i,y} - y)^2} \\ atan2(\boldsymbol{m}_{i,y} - y, \boldsymbol{m}_{i,x} - x) - \theta \end{bmatrix} + \begin{bmatrix} \epsilon_{\sigma_r^2} \\ \epsilon_{\sigma_\phi^2} \end{bmatrix} \quad (5)$$

Similar to previous section where the model is subject to noise, the model is added with zero-mean Gaussian noise, $\epsilon_{\sigma_r^2}$, $\epsilon_{\sigma_\phi^2}$ with variances σ_r^2 and σ_ϕ^2, respectively. Note that σ_r^2 and σ_ϕ^2 are also part of robot-specific parameters.

C. Single FastSLAM Algorithms

In this subsection, we briefly describe the FastSLAM algorithm for single SLAM problems [7]. It's based on Rao-Blackwellized particle filter, which is a variant of PF. Conceptually, Rao-Blackwellized particle filtes uses particles to represent the posterior over some variables, along with Gaussians (or some other parametric distributions) to represent all other variables. Concretely, FastSLAM uses particle filters for estimating the robot path while using multiple EKF to estimate map features locations individually. The advantage of FastSLAM over other SLAM algorithms arises from the fact that they can cope with non-linear robot motion models. This advantage is more important when the kinematics are highly non-linear, or when the pose uncertainty is relatively high. One other key property of FastSLAM is that it solves both the full SLAM problem and the on-line SLAM problem. That is, the FastSLAM is formulated to calculate the full path posterior and it also estimates one pose at-a-time.

Now we describe how a particle is represented and updated in details. Each particle contains an estimated robot pose, denoted by $\boldsymbol{x}_t^{[k]}$, a set of Kalman filters with mean $\mu_{i,t}^{[k]}$ and covariance $\Sigma_{i,t}^{[k]}$, one for each feature m_i in the map, and a weight $w^{[k]}$. Here $[k]$ is the index of the particle. With this particle representation, the basic steps of the FastSLAM algorithm are as follows. (Assuming there are M particles in total.)

- Do the following M times:
  1) Retrieval: Retrieve a pose $\boldsymbol{x}_{t-1}^{[k]}$ from the previous particle set Y_{t-1}.
  2) Prediction: Sample a new pose $\boldsymbol{x}_t^{[k]} \sim p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^{[k]}, \boldsymbol{u}_t)$.
  3) Measurement update: For each observed feature, $\boldsymbol{z}_t^i$, incorporate it into the EKF by updating the mean $\mu_{i,t}^{[k]}$ and covariance $\Sigma_{i,t}^{[k]}$.
  4) Importance weight: Calculate the importance weight $w^{[k]}$ for the new particle.
- Resampling: Sample, with replacement, M particles, where each particle is sampled with a probability proportional to $w^{[k]}$

The key property to the successful approximation of posteriors lies in the step of calculating importance weight and resampling. Intuitively, the importance weight is calculated to represent the likelihood of the particle based on the current measurements. In the resampling step, particles with higher weights are more likely to be preserved while particles with lower weights are more likely to be eliminated. In this way, we are continuously approximating the posteriors with those most likely particles. This property also implies that with larger number of particles, we can approximate the posteriors more accurately.

D. Multi FastSLAM Algorithms

In this subsection, we show how the FastSLAM can be extended to handle multiple robots. In fact, provided that the initial robot poses are known, the single SLAM algorithm can be generalized to handle multiple robots. The goal is to simultaneously estimate the posteriors over two robot trajectories and one common map. Mathematically, it's

$$p(\boldsymbol{x}_{1:t}^1, \boldsymbol{x}_{1:t}^2, m | \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2) \quad (6)$$

$$= p(\boldsymbol{x}_{1:t}^1, \boldsymbol{x}_{1:t}^2 | \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2) \quad (7)$$

$$p(m | \boldsymbol{x}_{1:t}^1, \boldsymbol{x}_{1:t}^2, \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2)$$

$$= p(\boldsymbol{x}_{1:t}^1 | \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2) \quad (8)$$

$$p(\boldsymbol{x}_{1:t}^2 | \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2)$$

$$p(m | \boldsymbol{x}_{1:t}^1, \boldsymbol{x}_{1:t}^2, \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2)$$

$$= p(\boldsymbol{x}_{1:t}^1 | \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1) p(\boldsymbol{x}_{1:t}^2 | \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2) \quad (9)$$

$$p(m | \boldsymbol{x}_{1:t}^1, \boldsymbol{x}_{1:t}^2, \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2)$$

$$= p(\boldsymbol{x}_{1:t}^1 | \boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{x}_0^1) p(\boldsymbol{x}_{1:t}^2 | \boldsymbol{z}_{1:t}^2, \boldsymbol{u}_{1:t}^2, \boldsymbol{x}_0^2) p(m | \boldsymbol{x}_{0:t}^1, \boldsymbol{x}_{0:t}^2) \quad (10)$$

From Eq. (6) to Eq. (7), the basic property of conditional probability is applied to decouple m from the other terms. From Eq. (7) to Eq. (8), we assume that these two trajectories are independent. From Eq. (8) to Eq. (9), we assume that the motion controls, measurements and the initial pose of one robot do not depend on the pose of the other. From Eq. (9) to Eq. (10), the Markov assumption is applied. Markov assumption postulates that past and future data are independent if one knows the current state $\boldsymbol{x}_t$. Therefore, $\boldsymbol{z}_{1:t}^1, \boldsymbol{u}_{1:t}^1, \boldsymbol{z}_{1:t}^2, \boldsymbol{x}_0^2$ are discarded given that $\boldsymbol{x}_{0:t}^1, \boldsymbol{x}_{0:t}^2$ are known. Based on the derivation, the particle filter for Multi-SLAM can be constructed as follows. Now each particle is a tuple containing $(\boldsymbol{x}_t^{1[k]}, \boldsymbol{x}_t^{2[k]}, m^{[k]}, w_t^{[k]})$. Following the framework in IV-C. The prediction step is now generalized to predict both $\boldsymbol{x}_t^{1[k]}$ and $\boldsymbol{x}_t^{2[k]}$. The measurement update step remains the same except that we now have two sets of measurements coming from two robots. As for the importance weights, since we have factored the posterior of the joint probability, the importance weight is simply the multiplication of each term in single-SLAM cases. The detailed algorithm is pseudo coded in in Alg. IV-D. For the detailed derivation of the EKF updates on each landmark, please refer to the book [9].

Algorithm 1 Multi-SLAM Algortihm

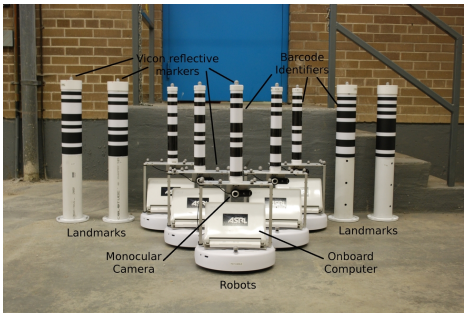
procedure MULTI-SLAM($\boldsymbol{Z}_t^1, \boldsymbol{Z}_t^2, \boldsymbol{u}_t^1, \boldsymbol{u}_t^2, Y_{t-1}$)
  for $k = 1$ to M **do**
    retrieve $\left\langle \boldsymbol{x}_{t-1}^{1[k]}, \boldsymbol{x}_{t-1}^{2[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \right\rangle \right\rangle$
    // N is number of landmarks
    $\boldsymbol{x}_t^{1[k]} \sim p(\boldsymbol{x}_t^1 | \boldsymbol{x}_{t-1}^{1[k]}, \boldsymbol{u}_t^{[1]})$
    $\boldsymbol{x}_t^{2[k]} \sim p(\boldsymbol{x}_t^2 | \boldsymbol{x}_{t-1}^{2[k]}, \boldsymbol{u}_t^{[2]})$
    for each feature $\boldsymbol{z}$ in $\boldsymbol{Z}_t^1$ **do**
      if feature $\boldsymbol{z}$ never seen before **then**
        $\mu_{i,t}^{[k]} = h^{-1}(z, \boldsymbol{x}_t^{1[k]})$
        $H = h'(\boldsymbol{x}_t^{1[k]}, \mu_{i,t}^{[k]})$
        $\Sigma_{i,t}^{[k]} = H^{-1} Q_t (H^{-1})^T$
        $w^{1[k]} = p_0$
      else
        $\hat{z} = h(\mu_{i,t-1}^{[k]}, \boldsymbol{x}_t^{1[k]})$
        $H = h'(\boldsymbol{x}_t^{1[k]}, \mu_{i,t-1}^{[k]})$
        $Q = H \Sigma_{i,t-1}^{[k]} H^T Q^{-1} + Q_t$
        $K = \Sigma_{i,t}^{[k]} H^T Q^{-1}$
        $\mu_{i,t}^{[k]} = \mu_{i,t-1}^{[k]} + K(\boldsymbol{z} - \hat{\boldsymbol{z}})$
        $\Sigma_{i,t}^{[k]} = (I - KH)\Sigma_{i,t-1}^{[k]}$
        $w^{1[k]} = |2\pi Q|^{-\frac{1}{2}} \exp(-\frac{1}{2}(\boldsymbol{z} - \hat{\boldsymbol{z}})^T Q^{-1} (\boldsymbol{z} - \hat{\boldsymbol{z}}))$
      end if
    end for
    for each feature $\boldsymbol{z}$ in $\boldsymbol{Z}_t^{[2]}$ **do**
      // similar to the above
    end for
    for all other unobserved landmarks i **do**
      $\mu_{i,t}^{[k]} = \mu_{i,t-1}^{[k]}$
      $\Sigma_{i,t}^{[k]} = \Sigma_{i,t-1}^{[k]}$
    end for
  end for
  $Y_t = \emptyset$
  for $i = 1$ to M **do**
    draw random k with probability $\propto w^{1[k]} * w^{2[k]}$
    add $\left\langle \boldsymbol{x}_t^{1[k]}, \boldsymbol{x}_t^{2[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, \left\langle \mu_{N,t}^{[k]}, \Sigma_{N,t}^{[k]} \right\rangle \right\rangle$
  end for
end procedure

V. IMPLEMENTATION

A. SLAM Dataset

In our project, we use the open dataset accquired from [4]. It includes dataset which are suitable to do experiments on single-SLAM and multi-SLAM. Each dataset contains odometry (instead of control inputs, "odometry" may be a misused of terms in their description of the dataset) and (range and bearing) measurement data from 5 robots, as well as accurate groundtruth data for all robot poses and 15 landmark positions. Each robot and landmark has a unique identification number encoded as a barcode (with known dimensions). Images captured by the camera on each robot (at a resolution of 960×720) are rectified and processed to detect the barcodes. The encoded identification number as well as the range and bearing to each barcode is then extracted. The camera on each robot is conveniently placed to align with the robot body frame.

(a)



(b)

Fig. 1. (a) The environment where the dataset was collected; (b) Robot platforms used to collect the dataset.

The dataset is collected in the indoor environment shown in Fig. 1(a). Robots move to randomly preset waypoints in a 15m × 8m indoor space while logging odometry data, and range-bearing observations to landmarks and other robots. Fig. 1(b) shows the experimental robot. This robot is the iRobot Create with 34cm in diameter.

B. ROS

Considering the possibility of extending our work to simulations or real robots, we decided to use ROS as the development tools and the infrastructure of our system. The Robot Operating System (ROS) is a set of software libraries and tools that help build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools. And it's all open source. The node diagram of our project is shown in Fig. 2. The starting node is $data\_reader$, which reads the data from the open dataset, and publishes the processed data to $slam\_runner$, in which multiple SLAM algorithms are implemented and run. Finally, $slam\_runner$ publishes the results of SLAM algorithms to $points\_and\_lines$ for post-processing and then $points\_and\_lines$ publishes at most 5 groundtruth paths and 5 SLAM paths to rviz for dynamic visualization.

C. Visualization

For convenience, we use the rviz tool for our final visualization. Rviz (ROS visualization) is a 3D visualizer for displaying sensor data and state information from ROS. We use $nav\_msgs/Path$ in rviz to show the paths, $visualization\_msgs/MarkerArray$ to show the

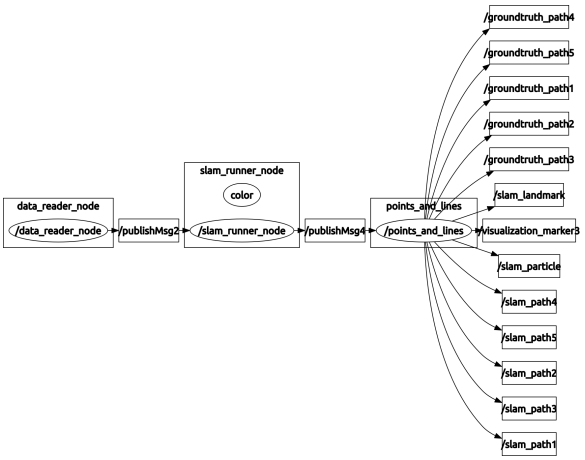


Fig. 2. The node structure of our project shown by using the rqt_graph command

landmarks, and $geometry\_msgs/PoseArray$ to show the particles.

VI. Results and Evaluation

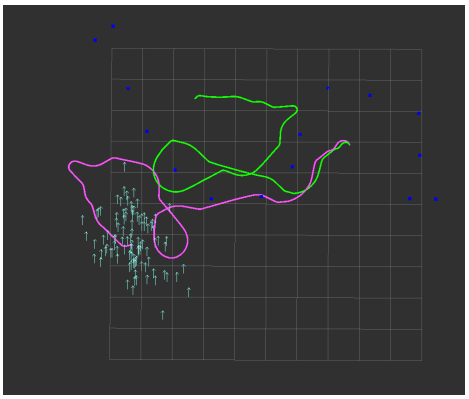
We conducted several experiments to validate both Single-SLAM and Multi-SLAM algorithms, all of which were implemented in C++. The experiments were performed on a system with an Intel i5 2.2GHz processor and 4GB RAM.

A. Dead-reckoning method vs. Single-Robot SLAM

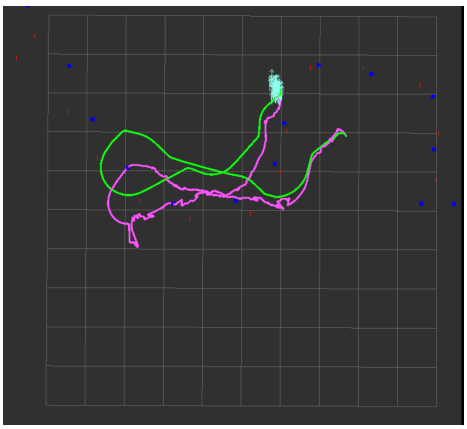
The trajectory results of Single-SLAM w/o incorporating measurements are shown in Fig. 3. The blue dots are groundtruth landmarks. The green path is the groundtruth path of the robot, while the purple one is the path that we calculated from the single-SLAM. The light-blue cluster is the approximated distribution of the robot pose estimated using particles. Fig. 3(a) shows the result of the trajectory without incorporating the measurement information. We can see that without doing it (also known as the dead-reckoning method), the error from the noisy robot motion eventually accumulates unboundedly compared to the Single-SLAM algorithm shown in Fig. 3(b). The numerical comparison is also shown in Fig. 4(a). To show the applicability of Single-SLAM, the numerical error of Single-SLAM is further shown in Fig. 4(b). The average error over 1400 frames (roughly 5 mins in real time) is 0.6m, which is acceptable considering robot's diameter is 0.34m. We also measured the mapping quality in terms of the landmark position error(the distance between our estimated landmark position and the groundtruth landmark position) in Fig. 4(c). The average error over 15 landmarks is about 0.6m and we can see that as time goes by, the error gradually converges.

B. Single-SLAM vs. Multi-SLAM

The experiments on multi-SLAM are also conducted. Fig. 5 is the result of 3 robots SLAM path. We can see that the resulting trajectories of each robot are still close to the groundtruth trajectories. Fig. 6(a) shows the numerical trajectory error between Single-SLAM and Multi-SLAM. The performance on multi-SLAM is better than single-SLAM in



(a)



(b)

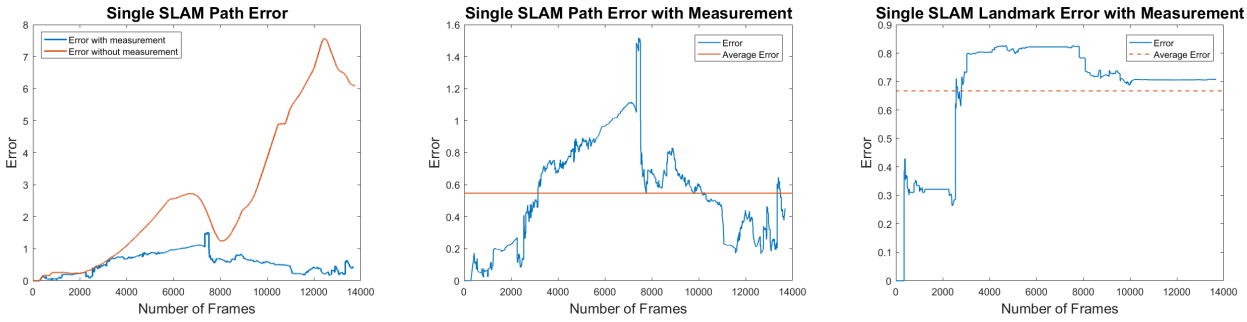
Fig. 3. (a) Single-robot trajectory without incorporating measurements (dead-reckoning method); (b) Single-robot SLAM with measurements incorporated

terms of the average error over the whole trajectory. For Multi-SLAM, it's about 0.42m, while it's 0.6m roughly for Single-SLAM. The Multi-SLAM also performs better on the map construction, which is shown in Fig. 6(b). The converged error in Multi-SLAM is only 0.5m compared to 0.7m in Single-SLAM. From the experiment, we've shown that multi-SLAM's performance is better than single-SLAM's performance.

VII. Strengths and Weaknesses

A. Strengths and weaknesses of the Multi-SLAM algorithm

From the derivation in section IV-D, we see that the algorithm is an exact extension from single-SLAM algorithm without approximations. The experiment results in section VI also show that the multi-SLAM can help improve the individual localization and mapping. However, there are assumptions which might not always be true in real world and have to be taken into account. For example, the assumption that the measurements of one robot do not depend on other poses of robots may not hold anymore if one robot can partially occlude the measurements and affect the accuracy of data association of the measurements. Fortunately in the dataset, the issue does not happen because the identity of each measurement is detected using barcodes, and therefore can be treated as exact values. Another issue that might arise from the algorithm is that if one robot's uncertainty is too



(a) (b) (c)

Fig. 4. (a) Comparison of path error between dead-reckoning method and single-robot SLAM; (b) Single-robot SLAM path error; (c) Map quality in terms of landmark position error.

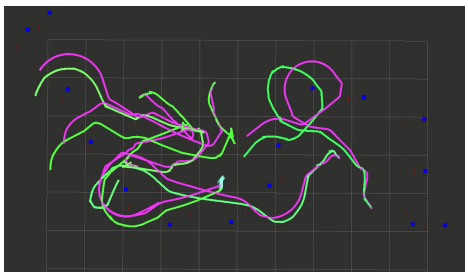
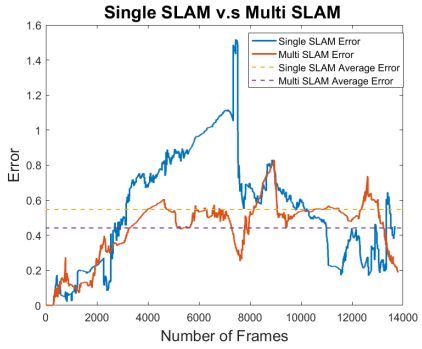
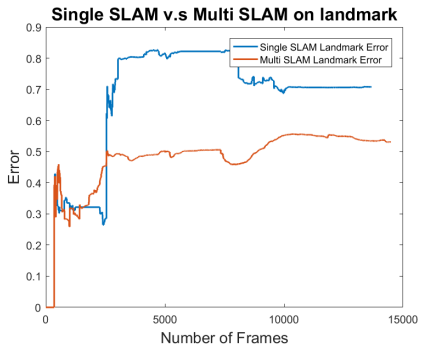


Fig. 5. Multi-robot SLAM path.



(a)



(b)

Fig. 6. (a) Comparison of path error between single-SLAM and Multi-SLAM; (b) Comparison of landmark error between single-SLAM and Multi-SLAM.

large, the performance could be worse because the ”polluted” information is incorporated. However this can actually be mitigated by tuning the robot-specific parameters described in section IV-A and section IV-B to be larger in order to reflect the level of uncertainty of this particular robot.

B. Strengths of our project

We successfully implemented both single-robot SLAM and multi-SLAM, and use rviz to do the visualization. Our result is acceptable with small deviated from the groundtruth.

Our project is based on ROS, which is readily extensible for our future work, such as the simulation(Gazebo) and real robots,

The visualization of our project is clear and easy to understand. The links of our demos are in the APPENDIX II.

C. Weaknesses of our project

In our experiment, we only use the the same parameter in motion model and measurement mode due to the time pressure. It would be better if we could do more experiments with different parameters.

VIII. Summary

In our project, we implemented particle-based filtering algorithms for Single and Multi SLAM and showed that, in terms of robot pose and map (landmark-based) quality, the Multi-SLAM algorithm performs better than the counterpart algorithm of Single-SLAM. For future work, we want to do data-driven parameter optimization for both motion model and measurement model in order to improve the performance of SLAM algorithms. Active exploration could be also another future work we can work on. Currently, the paths traversed by the robots in the dataset are randomly preset, meaning the mapping space is not actually explored in an efficient way. To achieve full autonomy, the robot should autonomously plan a suitable motion policy in order to visit unknown areas while minimizing the uncertainty on its pose. Therefore, active exploration planning algorithms are also promising as a direction of research.

APPENDIX I
CODE

Kai-Chieh Ma, Zhibei Ma, 2016, Github repository, `https://github.com/markcsie/slam_project`

APPENDIX II
VIDEOS

All the demos of our project are created by rviz in ROS.

The demo of single-robot SLAM:
`https://www.dropbox.com/s/322l1n7eunbttn6/single.mp4?dl=0`

The demo of multi-robot SLAM with 3 robots:
`https://www.dropbox.com/s/ts1870vq3u35576/multi3.mp4?dl=0`

The demo of multi-robot SLAM with 5 robots:
`https://www.dropbox.com/s/7n0uvcxir6ca9rc/multi5.mp4?dl=0`

APPENDIX III
RESPONSIBILITY

Kai-Chieh Ma: Single SLAM, Multi Robot SLAM, ROS.

Zhibei Ma: Data collection, Single SLAM, RVIZ Visualization.

REFERENCES

[1] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó. The slam problem: A survey. In *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, 2008.

[2] A. J. Davison and D. W. Murray. Simultaneous localization and map-building using active vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):865–880, July 2002.

[3] A. Howard. Multi-robot simultaneous localization and mapping using particle filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[4] K. Y. Leung, Y. Halpern, T. D. Barfoot, and H. H. Liu. The utias multi-robot cooperative localization and mapping dataset. *The International Journal of Robotics Research*, 30(8):969–974, 2011.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

[6] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[7] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai*, pages 593–598, 2002.

[8] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. to appear.

[9] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

[10] S. Thrun and Y. Liu. Multi-robot slam with sparse extended information filters. Springer, 2003.