A Hybrid Approach for Multiple-robot SLAM with Particle Filtering

Sajad Saeedi, Michael Trentini, and Howard Li

***Abstract*—In this paper, a hybrid algorithm for multiple-robot SLAM is proposed that combines the advantages of particle filtering and map merging. The proposed algorithm does not rely on rendezvous and calculates the unknown relative poses from the local maps of the robots. As another contribution, the uncertainty of the relative poses is taken into account by propagating the uncertainty to the past and future information using a novel algorithm. Moreover, once the relative poses are known, the integration of the information from all robots is performed using a novel batch-mode algorithm, which is a fast and efficient approach to deal with the time complexity problem. The experimental results show the effectiveness of the proposed algorithms.**

***Index Terms*—Simultaneous Localization and Mapping (SLAM), Multiple Robots, Particle Filter, Map Merging.**

I. Introduction

An autonomous robot needs to address two critical problems to survive and navigate within its surroundings: mapping the environment and finding its relative location within the map. Simultaneous localization and mapping (SLAM) is a process which aims to localize an autonomous mobile robot in a previously unexplored environment while constructing a consistent and incremental map of its environment. Correlation and dependency of localization and mapping on each other raise the complexity of the problem and necessitate accurately solving these two problems at the same time.

While single-robot SLAM is challenging enough, moving to a platform of multiple robots adds another layer of challenge. In a multiple-robot environment, robots must incorporate all available data to construct a consistent global map, meanwhile localizing themselves within the global map. In the previous works [1], [2], [3], and [4] map merging as a solution to this problem was introduced, which requires sharing information and processed data, such as the maps of the robots, among the robots. It is also possible to share unprocessed data such as raw laser observations and develop a global map by processing them. Compared with the map merging solutions, sharing raw data has the advantage of being more flexible when updating the incremental mapping and localization, since access to all unprocessed data is available. In both approaches, sharing processed data or raw data, there are various problems and limitations such as unknown initial relative poses of robots, uncertainty of the relative poses, updating maps and pose, complexity, and communication delays.

A. Literature review

There are few works formulating and performing multiple-robot SLAM with nonlinear filters. Some solutions, such as [5], [6], [7], [8], [9], [10], [11], [12], and [13] are based on GraphSLAM or different variations of the Kalman filter. In this section, key existing methods using particle filtering are briefly reviewed.

In S. Thrun's solution [14] for multiple-robot SLAM, which is based on particle filtering, it is assumed that the relative poses of the robots are known or they start from nearby locations.

A. Howard's solution [15] relies on direct encounters between robots to calculate the deterministic relative transformations. He proposes the idea of virtual robots to integrate past measurements after encounters between robots. Using the virtual robot concept, once robots meet each other, measurements are processed in reverse temporal order to integrate past observations with the global map.

Gil et al. [16] proposes a feature-based multiple-robot SLAM. The focus of the work is on managing the data association of visual features, assuming the visual descriptors have been affected with noise.

L. Carlone et al. [17] also apply a particle filter for multiple-robot SLAM as in [15]. Their contribution includes considering the uncertainty of the relative transformation in the mapping and using grid mapping [18] for particle filtering.

Generally, the spectrum of SLAM algorithms includes many different approaches each with remarkable capabilities. At the two extreme ends of the spectrum lie two methods: EKF-SLAM and GraphSLAM [19]. EKF-SLAM approximates the nonlinearity of the system by linearizing the system model. Also, EKF-SLAM is computationally expensive; each time acquired information is resolved into a probability distribution which makes it proactive but slow. GraphSLAM solutions such as iSAM [20] and HOGMAN [21] also depend on approximation by Taylor expansion. Its difference with EKF-SLAM is that it accumulates information and therefore is considered to be a lazy and an off-line algorithm [19]. The particle filtering approaches are nonlinear filtering solutions; therefore, the system models are not linearized. Although particle filtering is computationally expensive, it is shown that a good implementation such as FastSLAM [19] and grid mapping [18] can reduce the complexity of the algorithm. Considering these advantages, authors of the paper decided to improve upon the current particle filter-based multiple-robot SLAM algorithms such as [15]. In this paper, laser odometry is used to associate measurements, the maps are occupancy grid maps, and for simplicity, a centralized approach was chosen for implementation.

B. Contribution

An important problem in multiple-robot SLAM is finding the relative poses of the robots. Once the relative poses are known, propagating the uncertainty of the relative poses to the maps and trajectories, and also integrating the past information in an efficient manner are other major problems. The contributions of the current work include a few novel improvements in different aspects of multiple-robot SLAM. These improvements are considering uncertainty of the relative transformation between the robots, updating maps and poses, and integrating map matching with particle filtering.

- **Particle regeneration**: The uncertainty of the relative transformation of the poses of the robots is usually ignored in the cooperative mapping and localization algorithms. This can affect the confidence in maps and poses. The proposed particle regeneration method in Algorithm 1 takes care of the uncertainty of the transformation during the mapping and localization process.

- **Batch update**: This method, proposed in Algorithm 2, allows us to update past maps and poses, prior to the time that the relative poses are known, by processing them in batch, rather than reprocessing individual data items one by one. The obvious benefit of this algorithm is saving time by avoiding reprocessing past data and also saving memory by not storing all past raw data.
- **Hybrid method**: The proposed method in Algorithm 3 deals with the problem that if the transformation between robots is not known then it has to be extracted from other sources of information such as maps or line-of-sight observations. In general, it is not guaranteed that robots will meet each other at a point to calculate the relative transformation from line-of-sight. However, by processing maps, this problem can be solved. Previous map merging solutions are integrated with the current work to make it a practical and operational solution. By adding map merging, the proposed solution becomes a hybrid solution where maps and raw data are used to build and maintain a global map.

The rest of the paper is organized as follows: Section II presents the proposed method, Section III presents some experimental results, and Section IV makes some general conclusions and discusses future work.

II. Proposed Method

The notation in this work follows the standard presented in *Probabilistic Robotics* [19]. Suppose the pose of a robot from time 1 to time t is shown by the sequence $\{x_1, x_2, ..., x_t\}$. This sequence is shown in the compact form of $x_{1:t}$. Also, since multiple agents are involved, the identification number of each robot appears as a superscript in the state variable. Therefore, $x^i_{1:t} \equiv \{x^i_1, x^i_2, ..., x^i_t\}$ shows the state of the i^{th} robot from time 1 to time t, where $i = 1, ..., n$ and n is the number of the robots. The same notation applies to the observation and control signals, $z^i_{1:t}$ and $u^i_{1:t}$. Notice that sometimes the robots' identification numbers are represented by alphabetical characters.

In this work, the problem of the multiple-robot SLAM with unknown relative poses and indirect observations (robots observing mutual scenes) is studied. Results from the previous works of the authors [3] are used to calculate the transformation; however, the problem of integrating the information from multiple sources is yet to be solved. Throughout the paper, for simplicity, the problem is presented for two robots and when possible, it is explained for more than two robots.

Chronologically, when the relative poses are calculated on-the-fly, the SLAM problem for multiple robots can be divided into two parts: before the relative poses are known and after that. Before the relative poses are known, it is not possible to make a global map and have a joint posterior for poses because there is no common ground to connect the robots to each other. However, after the relative poses are known, the robots can share data and generate a global map. At the time that the relative poses are known, a particle filter is constructed which proceeds to integrate all information until the end of the mapping. Before that time, robots were mapping and localizing individually and independently. Once the relative pose are known, the proposed *batch integration* method is used to integrate the past information.

In particle filter-based multiple-robot SLAM, each particle holds the trajectory of each robot, a map, and the weight associated with the particle. One of the major issues in multiple-robot SLAM is the propagation of the uncertainty of the transformation on the joint posterior of the poses and the map. This concern can be taken care of by proper sampling of the particles which is explained in the next two subsections. Once the relative poses are known and the effect of uncertainty of the transformation is applied to them, multiple robot SLAM with known relative poses is used to update the maps and poses as observations are received.

A. Batch Integration

The batch integration is concerned with the integration of the information, maps and poses, from multiple robots prior to the time that the relative poses were known. In batch update, maps and poses are updated in batch mode. The batch mode means that the past data is not processed item by item; rather, maps and poses which were already processed and contain a lot of information, are updated in bulk. Batch update involves two main steps: first the uncertainty of the given relative transformation should be handled. Then the past information should be updated and integrated. The first step is addressed using the *particle regeneration* method. The second step is handled by the *batch update* method.

1) Particle Regeneration: Assume that at time s the transformation between the two robots a and b is given with a Gaussian distribution as follows:

$$p(\delta^{ab}_s) = |2\pi\Sigma^{ab}_s|^{-\frac{1}{2}} \exp\left\{\frac{-(\delta^{ab}_s - \mu^{ab}_s)^T {\Sigma^{ab}_s}^{-1} (\delta^{ab}_s - \mu^{ab}_s)}{2}\right\} \quad (1)$$

$$\sim \mathcal{N}(\delta^{ab}_s; \mu^{ab}_s, \Sigma^{ab}_s), \quad (2)$$

where $\mathcal{N}(\delta^{ab}_s; \mu^{ab}_s, \Sigma^{ab}_s)$ denotes a Gaussian distribution with the mean of μ^{ab}_s and the covariance of Σ^{ab}_s defined as follows

$$\mu^{ab}_s = \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_\psi \end{bmatrix}, \qquad \Sigma^{ab}_s = \begin{bmatrix} \sigma^2_{xx} & \sigma^2_{xy} & \sigma^2_{x\psi} \\ \sigma^2_{xy} & \sigma^2_{yy} & \sigma^2_{y\psi} \\ \sigma^2_{x\psi} & \sigma^2_{y\psi} & \sigma^2_{\psi\psi} \end{bmatrix}. \quad (3)$$

μ_x and μ_y represent the translation and μ_ψ denotes the rotation. In practice, the values of the uncertainty is determined by the algorithm which calculates the relative poses. Generally, when a particle filter is constructed, the initial distribution, which particles are drawn from, is the belief that exists about the random variable. For example, the distribution can be Gaussian or even a delta function, where all particles have equal values and weights. The same principle applies for multiple robots performing SLAM. Without loss of generality, let us assume that the map and the pose of robot b is being transformed to the coordinates of robot a using the transformation in equation (2). At time s, a new particle filter is constructed which includes poses of both robots and a global map. (Note that the maps developed by each robot prior to time s will be used in a process called *batch update* to make a global map from the past information. The constructed particle filter will incrementally add the new measurements, received after time s, to this global map.) For the particle filter, the poses related to robot a can be selected from its last known pose. However, for robot b, the situation is different because its poses have to be transformed first. On the other hand, the transformation is uncertain and its effect on the pose and map should be taken into account. The proposed solution deals with this issue in three steps as follows.

- **Particle transformation**: All particles, representing the pose and map of robot b, are transformed according to the nonlinear transformation function. Once each particle is transformed according to the uncertain transformation, the result is an uncertain distribution which is approximated as explained in the next step.
- **Gaussian approximation**: Once each particle is transformed, the resulting pose of the transformed particle will have a nonlinear distribution. This nonlinear distribution is approximated with a Gaussian distribution. The motivation for the Gaussian approximation is to provide efficient and feasible computation for further processing. The main advantage of the Gaussian approximation is saving time.
- **Gaussian sampling**: From each Gaussian distribution assigned to the transformed pose, new particles are generated. These particles are used to update maps and poses in the multiple-robot SLAM framework. Notice that each particle before the transformation holds a map. Once a particle is transformed, not only its pose is transformed, but its map is also transformed. The transformation of the map with uncertainty was already studied by authors' previous work in [22]. The transformed maps will be used as will be explained in the *batch update*.

These three steps have been illustrated in Fig. 1 and are explained in detail.

Particle transformation:
At time s, let us assume that the distribution of the pose of robot b before the transformation is given by $p(\hat{x}_s^b)$. The distribution is modelled by $\hat{n}^b$ particles. Fig. 1-a shows this distribution. The small bars show the particles representing the distribution. One of the particles is highlighted to show the effect of the uncertain transformation on it. Fig. 1-b depicts the uncertain and nonlinear transformation. The uncertainty bounds are also depicted in this figure. Once the i^{th} particle, $\hat{x}_s^{(i)b}$, is transformed by this uncertain transformation, the particle will also have an uncertain distribution (Fig. 1-c), which can be modelled by a Gaussian distribution.

Gaussian approximation:
Mathematically, the pose of each particle before the transformation, $\hat{x}_s^{(i)b}$, and after the transformation, $\check{x}_s^{(i)b}$, are related by the following function:

$$\check{x}_s^{(i)b} = f(\mu_s^{ab}, \hat{x}_s^{(i)b}) \tag{4}$$

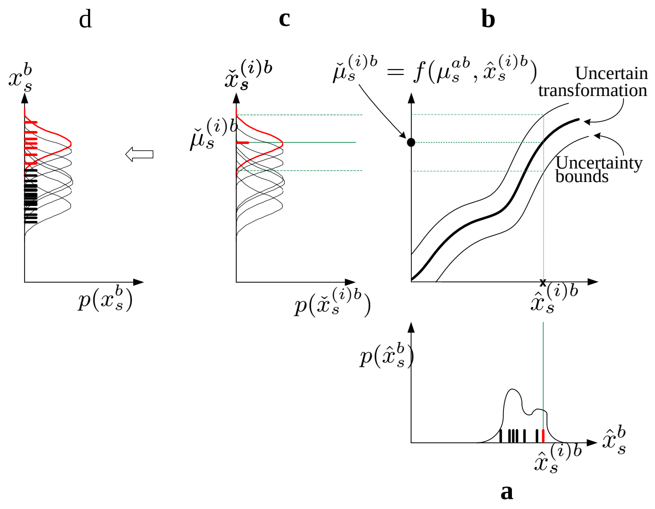

Fig. 1. The proposed particle regeneration includes three steps: particle transformation, Gaussian approximation, and Gaussian sampling. **a)** The distribution of the pose of robots b before the transformation, at time s. Small bars are particles, modelling the distribution. A particle is highlighted to demonstrate the particle regeneration algorithm. **b)** Nonlinear and uncertain transformation function. **c)** All particles in **a** are mapped through the transformation function. The result of the transformation of each particle is approximated with a Gaussian distribution. **d)** From each Gaussian distribution, samples are generated to model it with particles.

where $f(.)$ is the transformation function, defined as

$$\check{x}_s^{(i)b} = f(\mu_s^{ab}, \hat{x}_s^{(i)b}) \tag{5}$$

$$= \begin{bmatrix} \cos\mu_\psi & -\sin\mu_\psi & 0 \\ \sin\mu_\psi & \cos\mu_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \hat{x}_s^{(i)b} + \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_\psi \end{bmatrix}. \tag{6}$$

Once the i^{th} particle is transformed, its pose is modelled with a Gaussian distribution given as

$$p(\check{x}_s^{(i)b}) = |2\pi\check{\Sigma}_s^{(i)b}|^{-\frac{1}{2}} \exp\left\{\frac{-(\check{x}_s^{(i)b}-\check{\mu}_s^{(i)b})^T\check{\Sigma}_s^{(i)b^{-1}}(\check{x}_s^{(i)b}-\check{\mu}_s^{(i)b})}{2}\right\} \tag{7}$$

$$\sim \mathcal{N}(\check{x}_s^{(i)b}; \check{\mu}_s^{(i)b}, \check{\Sigma}_s^{(i)b}), \tag{8}$$

where the mean and covariance of the Gaussian is defined according to equation (6) as

$$\check{\mu}_s^{(i)b} = f(\mu_s^{ab}, \hat{x}_s^{(i)b}), \tag{9}$$

$$\check{\Sigma}_s^{(i)b} = F_\delta\, \Sigma_s^{ab}\, F_\delta'. \tag{10}$$

F_δ is the Jacobian of the transformation defined as

$$F_\delta = \frac{\partial f(\mu_s^{ab}, \hat{x}_s^{b(i)})}{\partial(\mu_s^{ab})}. \tag{11}$$

The Jacobian is calculated as

$$F_\delta = \begin{bmatrix} 1 & 0 & r_1 \\ 0 & 1 & r_2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{12}$$

where, to simplify the representation, scalars r_1 and r_2 are defined as follows:

$$r_1 = [-sin\mu_\psi \quad -cos\mu_\psi \quad 0]\hat{x}_s^{b(i)}, \tag{13}$$

$$r_2 = [cos\mu_\psi \quad -sin\mu_\psi \quad 0]\hat{x}_s^{b(i)}. \tag{14}$$

Therefore, once every particle is transformed, its pose will be represented by a Gaussian distribution, where the mean of the Gaussian distribution is defined according to equation (9) and its covariance is defined according to equation (10). Notice that with each particle, there is a map which is also affected by the uncertain transformation. The effect of the uncertain transformation on the maps was presented in authors' previous work [22].

Gaussian sampling:
When a particle filter is extended to include more state variables, the number of the particles must be increased to sufficiently represent the distribution of the new state. For example, if the pose and map of robot a and robot b are represented by 7 particles for each, then to represent the pose and map of both robots by a particle filter, a good approximation might be 50 particles. The reason for the increased number is that more particles are needed to approximate a bigger state vector.

Each of the transformed particles, $\hat{x}_s^{(i)b}$, should be regenerated to cover the bigger state space which is used in the particle filter initialized right after the moment that the relative poses are known. This is shown in Fig. 1-d. The transformed distribution shown in the figure is sampled to generate more particles. This sampling has two advantages. First, it represents the Gaussian distribution with particles which is required in particle filtering. Second, the particle set is increased as required for a bigger state space. Briefly, the red particle in 1-a, after the uncertain transformation, is represented by many red particles in 1-d. Each particle of each Gaussian distribution, shown in 1-c, holds a map, which is also affected by the transformation.

Algorithm 1 summarizes particle regeneration with its three steps: *particle transformation*, *Gaussian approximation*, and *Gaussian sampling*. The algorithm assumes that the information from robots b should be transformed into coordinates of robot a. The algorithm takes the particle set representing the pose and map of robots b before the transformation is applied, $\hat{S}_s^b$, and outputs a particle set which represents the transformed state of the robot, S_s^b. In line 1, the set of output particles is initialized. In lines 2-3, the mean and covariance of the transformation matrix are assigned to the relevant parameters. In line 5, a loop starts to process all given particles. ($\hat{n}_s^b$ is cardinality of $\hat{S}_s^b$.) First particle transformation and Gaussian approximation are performed. In line 7, for each particle, the mean of the transformed particle is calculated. In line 8, the covariance of each transformed particle is calculated. For this line, the result in equation (10) is used. In line 9, based on the calculated mean and covariance, a Gaussian distribution is assigned to model the pose of each transformed particle. In line 10, the $\mathrm{lup}(\cdot)$ function returns the transformed map of the particle using the linearized uncertainty propagation, proposed in the authors' previous work [22]. For the pose of each transformed particle, there exists a Gaussian distribution, allowing Gaussian sampling to be performed. From each distribution, $[\frac{n}{\hat{n}^b}]$ poses are chosen and added to the output particle set with their corresponding maps (lines 11-14), where n is the desired population of the particles for multiple-robot

Algorithm 1 Particle regeneration:
$\mathrm{S}_s^b = \mathrm{particleRegeneration}(\hat{\mathrm{S}}_s^b, \delta_s^{ab})$

```
Require: set of particles of posterior before the transformation at time $s$: $\hat{\mathrm{S}}_s^b$,
    uncertain transformation between robots $a$ and $b$ at time $s$: $\delta_s^{ab}$.
Ensure: set of particles of posterior after the transformation at time $s$: $\mathrm{S}_s^b$.
1:  $\mathrm{S}_s^b \leftarrow \emptyset$
2:  $\mu_s^{ab} \leftarrow \mathrm{mean}(\delta_s^{ab})$
3:  $\Sigma_s^{ab} \leftarrow \mathrm{cov}(\delta_s^{ab})$
4:  $\hat{n}^b \leftarrow \|\hat{\mathrm{S}}_s^b\|$
5:  for $i = 1 \rightarrow \hat{n}^b$ do
6:      $< \hat{x}_s^{b(i)}, \hat{m}_s^{b(i)}, \hat{w}_s^{b(i)} > \leftarrow \hat{\mathrm{S}}_s^{b(i)}$
7:      $\breve{\mu}_s^{(i)b} \leftarrow f(\mu_s^{ab}, \hat{x}_s^{(i)b})$
8:      $\breve{\Sigma}_s^{(i)b} \leftarrow F_\delta \, \Sigma_s^{ab} \, F_\delta'$
9:      $p(\breve{x}_s^{(i)b}) \sim \mathcal{N}(\breve{x}_s^{(i)b}; \breve{\mu}_s^{(i)b}, \breve{\Sigma}_s^{(i)b})$
10:     $m_s^{b(i)} \leftarrow \mathrm{lup}(\hat{m}_s^{b(i)}, \mu_s^{ab}, \Sigma_s^{ab})$
11:     for $j = 1 \rightarrow [\frac{n}{\hat{n}^b}]$ do
12:         sample $x_s^{(j)b} \sim p(\breve{x}_s^{(i)b})$
13:         $\mathrm{S}_s^b \leftarrow \mathrm{S}_s^b \cup < x_s^{(j)b}, m_s^{b(i)}, \hat{w}_s^{b(i)} >$
14:     end for
15: end for
16: return $\mathrm{S}_s^b$
```

SLAM. Notice that the maps for each Gaussian distribution are similar. Also, particle weights are not affected by the transformation. However, once the particles are regenerated, their weights have to be normalized.

2) Batch Update: In the previous section, it is explained that at the moment the uncertain relative poses of the robots are known, particles holding maps and pose are transformed. So far, the poses of both robots are at global coordinates. Also, maps of both robots have been transformed. For both maps and poses, the uncertainty of the transformation has been propagated. Now the question is how all poses and maps in the global coordinates can be integrated to create a joint posterior and a global map for the past information. As mentioned, to answer this question, batch update is used. Batch update allows us to integrate all past information, prior to the time that the relative poses are known, into the global coordinates without processing past raw sensor data items individually. The obvious advantage of the batch update is that it saves time by avoiding reprocessing past information.

Similar to the previous section, assume that particle set representing the state of robots a and b in the global frame at time s is given by S_s^a and S_s^b. First, these uncorrelated particles should be rearranged into a new particle set to represent the state for both robots. All particle should be used to generate a higher dimension particle set. This is performed by uniformly drawing particles from both sets. Then maps of particles are fused to make a joint map from the past information. Algorithm 2 summarizes the batch update for two robots. The algorithm takes sets S_s^a and S_s^b as inputs. The cardinality of set S_s^a is n^a. But in Algorithm 1 the cardinality of set S_s^b was updated to be n particles, which is the desired number of

Algorithm 2 Batch update: $\mathrm{S}_s^{ab} = \mathrm{batchUpdate}(\mathrm{S}_s^a, \mathrm{S}_s^b)$

```
Require: set of particles representing posterior at time $s$: $\mathrm{S}_s^a$,
    set of particles representing posterior at time $s$: $\mathrm{S}_s^b$.
Ensure: set of particles representing joint posterior at time $s$:
    $\mathrm{S}_s^{ab}$.
 1: $\mathrm{S}_s^{ab} \leftarrow \emptyset$
 2: $\mathrm{S}_s^a \leftarrow \mathrm{populate}(\mathrm{S}_s^a, n)$
 3: for $i = 1 \to n$ do
 4:     $< x_s^{(i)a}, m_s^{(i)a}, w_s^{(i)a} > \leftarrow \mathrm{rand}(\mathrm{S}_s^a)$
 5:     $< x_s^{(i)b}, m_s^{(i)b}, w_s^{(i)b} > \leftarrow \mathrm{rand}(\mathrm{S}_s^b)$
 6:     $m_s^{(i)ab} \leftarrow \mathrm{fuse}(m_s^{(i)a}, m_s^{(i)b})$
 7:     $w_s^{(i)ab} \leftarrow w_s^{(i)a} w_s^{(i)b}$
 8:     $\mathrm{S}_s^{ab} \leftarrow \mathrm{S}_s^{ab} \cup < x_s^{(i)a}, x_s^{(i)b}, m_s^{(i)ab}, w_s^{(i)ab} >$
 9: end for
10: $\mathrm{S}_s^{ab} \leftarrow \mathrm{normalize}(\mathrm{S}_s^{ab})$
11: return $\mathrm{S}_s^{ab}$
```

particles for the joint particle filter. Determining the number of the particles is another important problem in particle filtering which is out of the scope of the paper. The output of the algorithm is S_s^{ab} with cardinality of n. This set represents the joint posterior of the poses and map for both robots. In line 1, the output of the algorithm is initialized with an empty set. In line 2, the number of particles in the input set of S_s^a increased to the same number as in S_s^b. The desired number of particles should be n which is greater than n^a. In line 3, a loop starts to update all particles. In lines 4 and 5, particles without replacement are uniformly selected from the input sets. The function $\mathrm{rand}(\cdot)$ does this operation. In these two lines, particles are chosen randomly and combined later to make a new set of particles. The motivation to perform random selection is to have diversity of the particles at the output of the algorithm, using the input particle sets. In line 6, For each pair of the selected particles, their maps are fused. For map fusion, the method based on the entropy filter, proposed in [4] by the authors, is used. In map fusion using the entropy filter, the additive property of the log odds representation of occupancy of the cells is utilized to reflect the uncertainty of the map cells. Note that if maps are feature-based maps, map fusion solutions such as [12] can be used. In line 7, weights of the particles are multiplied to generate the weight of the resulting particle. Finally, in line 8, the new particle is added to the output particle set. At the end, particle weights of the output set are normalized to sum to one.

Algorithm 3 summarizes the proposed solution, integrating all proposed algorithms. This algorithm can be thought of as a hybrid method, where map merging is used to identify the relative poses and also a multiple-robot particle filter is used to integrate the information from multiple sources.

The algorithm runs single-robot SLAM if the relative poses are unknown. Otherwise it implements multiple-robot SLAM. The distinction between these are made by the variable $mode$. If the relative pose is known, $mode$ is $multipleRobotSLAM$; otherwise it is $singleRobotSLAM$. Inputs of the algorithm are control actions u_t^a and u_t^b, observations z_t^a and z_t^b, the previous state of the robot represented by particles P_{t-1}, and $mode$. The algorithm returns the new particle set P_t and the $mode$. In lines 2-4, the algorithm starts with single-robot SLAM. In line 5, maps are matched to check if it is possible to calculate the relative pose from maps. For metric maps, to calculate the relative poses, the Hough peak matching algorithm [23] and [3] is used, in which the Hough peaks of the maps are matched to calculate the relative transformation. If it fails to find the relative pose (success = false), the state of each robot is returned and single-robot SLAM is implemented in the next iteration (line 7). If the relative pose can be calculated (success = true), then the relative pose and particle sets are used to implement the batch integration algorithm (lines 9-10), the mode is changed to $multipleRobotSLAM$, and a new particle set representing a state for multiple robots is returned. Once the mode is changed to $multipleRobotSLAM$, in lines 16-19, multiple-robot SLAM with known relative poses is implemented.

Algorithm 3 Hybrid method:
$(\mathrm{P}_t, mode) = \mathrm{hybrid}(u_t^a, u_t^b, z_t^a, z_t^b, \mathrm{P}_{t-1}, mode)$

```
 1: if $mode = singleRobotSLAM$ then
 2:     $(\mathrm{S}_{t-1}^a, \mathrm{S}_{t-1}^b) \leftarrow \mathrm{P}_{t-1}$
 3:     $\mathrm{S}_t^a = \mathrm{singleRobotSLAM}(\mathrm{S}_{t-1}^a, u_t^a, z_t^a)$
 4:     $\mathrm{S}_t^b = \mathrm{singleRobotSLAM}(\mathrm{S}_{t-1}^b, u_t^b, z_t^b)$
 5:     $(\delta_t^{ab}, success) \leftarrow \mathrm{relativePose}(\mathrm{S}_t^a, \mathrm{S}_t^b)$
 6:     if $success = false$ then
 7:         $P_t \leftarrow (\mathrm{S}_t^a, \mathrm{S}_t^b)$
 8:     else
 9:         $\mathrm{S}_t^b = \mathrm{particleRegeneration}(\mathrm{S}_t^b, \delta_t^{ab})$
10:         $\mathrm{S}_t^{ab} = \mathrm{batchUpdate}(\mathrm{S}_t^a, \mathrm{S}_t^b)$
11:         $mode = multipleRobotSLAM$
12:         $P_t \leftarrow \mathrm{S}_t^{ab}$
13:     end if
14:     return $(\mathrm{P}_t, mode)$
15: else
16:     $\mathrm{S}_{t-1}^{ab} \leftarrow \mathrm{P}_{t-1}$
17:     $\mathrm{S}_t^{ab} = \mathrm{multipleRobotSLAM}(\mathrm{S}_{t-1}^{ab}, u_t^a, u_t^b, z_t^a, z_t^b)$
18:     $P_t \leftarrow \mathrm{S}_t^{ab}$
19:     return $(\mathrm{P}_t, mode)$
20: end if
```

B. Advantages and Disadvantages

The proposed batch integration method, consisting of Algorithm 1, particle regeneration and Algorithm 2, batch update, creates a filter once the relative poses are calculated and updates the maps and poses. It also takes into account the uncertainty of the relative poses and propagates it onto past and future information. An obvious advantage of batch integration is that it saves time by avoiding reprocessing past data items. Also it requires less memory, since storing maps takes less space than storing raw data. In general, storing maps, whether feature maps, topological maps, or metric maps, requires less space than storing the raw data that the maps are built from. In fact the reduction in space comes at the cost of processing the raw data. A disadvantage of the method is that it lacks the flexibility to reprocess past data items, individually, which would make it possible to apply line-of-sight observations or redefine the past trajectories of the robots.

III. Experimental Results

In this section simulated and real-world experiments are presented and the results are analysed. The simulated experiment was performed using feature-based SLAM and the real-world experiment was performed using view-based SLAM.

A. Simulated Experiment

For this experiment, the simulated robots, observations, and control actions were developed in MATLAB. Two robots are used for this simulation. Also, it is assumed that the relative transformation between the robots is not known, and it is calculated from the map.

Based on Algorithm 3, the robots start in the $singleRobotSLAM$ mode. Periodically, their maps are matched, following line 5 in Algorithm 3. For the implementation of the function $\mathrm{relativePose}(\cdot)$, the iterative closest point (ICP) algorithm was used, in which features are considered as points to be matched. Once the matching is acceptable based on the error of the matching, the relative transformation between the maps is used to determine the relative transformation between the robots, and the mode is switched to $multipleRobotSLAM$. New particles are generated for this mode, and the robots continue to perform multiple-robot SLAM with known relative poses. In the following paragraphs, detailed analysis of important issues such as the uncertain transformation and map merging in feature-based maps are introduced.

1) Transforming Features with an Uncertain Transformation: In author's previous work [22], transforming metric maps with an uncertain transformation was studied. For feature-based maps, the same formulation applies, except that for feature maps only certain parts of the proposed solution are required which will be explained by an example.

Fig. 2 shows the effect of an uncertain transformation on features. It is assumed that the features are translated by $[3\,,\,2]$ along X and Y axes and rotated $15°$. The transformation is uncertain and the covariance of the transformation, defined in equation (3), is given as follows:

$$\Sigma_{tr} = \begin{bmatrix} 0.1 & 0.02 & 0.01 \\ 0.02 & 0.15 & 0.01 \\ 0.01 & 0.01 & 0.099 \end{bmatrix}. \qquad (15)$$

Fig. 2-a and Fig. 2-b show the result of the transformation when the uncertainty of the feature, q, before the transformation is zero. Using linearization, the covariance of the transformed point, Σ_r, is calculated based on the following relation:

$$\mathbf{\Sigma}_r = F_{xy\psi}\, \mathbf{\Sigma}_{tr}\, F_{xy\psi}^T + F_q\, \mathbf{\Sigma}_q\, F_q^T. \qquad (16)$$

where $F_{xy\psi}$ and F_q are the Jacobians of the transformation function, $f(\cdot)$, and are given based on:

$$F_{xy\psi} = \frac{\partial f(\psi, x, y, q)}{\partial(\psi, x, y)},\ F_q = \frac{\partial f(\psi, x, y, q)}{\partial(q)}. \qquad (17)$$

$\mathbf{\Sigma}_q$ is the covariance of the point q which in Fig. 2-a and Fig. 2-b is zero, meaning that the feature before the transformation has no uncertainty. In Fig. 2-b, the feature is in a different location, compared with Fig. 2-a. According to equation (16), the uncertainty of the transformed point is a function of the position of the feature point, q. Therefore, points at different locations will experience different uncertainties after the transformation.

Fig. 2-c and Fig. 2-d show the result of the uncertain transformation, when the uncertainty of the feature, along X and Y axes, before the transformation is not zero. For this case, the uncertainty of the features before the transformation is

$$\mathbf{\Sigma}_q = \begin{bmatrix} 0.2 & 0.05 \\ 0.05 & 0.1 \end{bmatrix}. \qquad (18)$$

Similarly, in Fig. 2-d, the feature is in a different location, compared with Fig. 2-c. Since the uncertainty of the transformed point is a function of the position of the feature point, points at different locations with similar covariance matrices before the transformation will experience different uncertainties after the transformation. Note that the uncertainty of the transformed point in Fig. 2-d is larger than the one in Fig. 2-b, because in Fig. 2-d, the initial uncertainty of the point before the transformation is not zero.

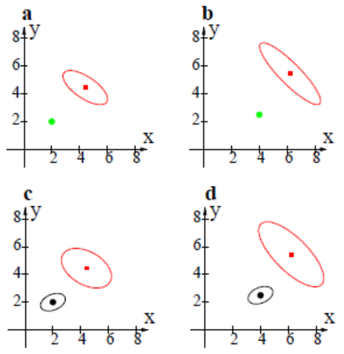

Fig. 2. Transformation of a feature. Green points show features with no position uncertainty. Black points show uncertain features with their position uncertainty ellipse. Red squares demonstrate the transformed features. For all examples, the translation is $[3\,,\,2]$, and the rotation is $15°$. The transformation for all examples is uncertain. **a)** A deterministic feature is transformed. **b)** Another deterministic feature, which is far from the center, is transformed. **c)** An uncertain feature is transformed. **d)** Another uncertain feature, which is far from the center, is transformed.

2) Transforming Poses with an Uncertain Transformation: Once the relative transformation between the robots is known, not only features but also poses of the robots should be transformed. Fig. 3 shows a robot in a black triangle, located at $[2\,,\,2]$ with an initial orientation of $30°$. The pose of the robot is translated by $[3\,,\,2]$ along X and Y directions and rotated $15°$. The transformation is uncertain and its covariance is given in equation (15). The covariance of the transformed pose is calculated using equation (10). Since it is not easy to visualize the covariance ellipse for position and orientation of the transformed robot, 100 pose samples are generated from the calculated covariance and illustrated by red triangles. The red dotted ellipse shows only the covariance of the position of the robot. Note that initial uncertainty of the pose of the robot before the transformation is taken care of by the diversity of the particles and this figure represents the pose of only one particle.

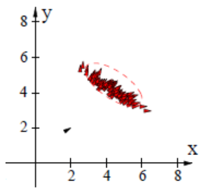

Fig. 3. Uncertain transformation of the pose of a robot. To visualize the uncertainty of the transformed pose, 100 pose samples resulting from the uncertain transformation are shown.

3) Merging Feature Maps: Once two feature maps are aligned by a transformation, they must be merged to generate a global map. Merging maps can be done by copying all features from one map to another; however, due to noise and uncertainty of the position of the features, duplicate features might exist. These duplicates should be merged to avoid incorrect mapping. To do this, if the distance between two features from aligned maps is less than a predefined threshold, they should be merged and considered as one feature. This has been shown for two features in Fig. 4. Note that since averaging is a linear operation, the covariance of the merged point is the result of the linear summation of the covariances of the points before merging.

4) Results: Fig. 5-a shows the environment in which the proposed hybrid algorithm has been tested. Two robots start from different positions. Without loss of generality, the origin of the first robot is assumed to be the origin of the global coordinates. Each robot is represented by two triangles: a green triangle which shows the actual pose of the robot and a red triangle which shows the estimated pose of the robot. For the first robot, both triangles are located at the origin. For the second robot, the actual position is at $[15\,,\,-10]$ with the orientation of $0°$. The estimate of this robot is located at the origin. Before the relative transformation is known, each robot develops its own map. The maps are overlaid in the global frame, without being aligned. Fig. 5-b shows the simulation at time $t = 27$ s. All features identified by the second robot are misplaced (shown by black arrows) because the relative transformation is not known. Note that for the same reason, the estimated pose of the second robot in the global coordinates, shown by a red triangle, is different from its actual pose in the global frame (shown by a red arrow). As mentioned, after each measurement update, map matching is performed. At time $t = 28$ s, map matching is able to calculate the required transformation for the maps. The calculated translation is $[15.26\,,\,-10.18]$ and the orientation is $0.81°$. At this point the map and the pose of the second robot are transformed to the global coordinates, maps are merged, and particles are regenerated to perform multiple-robot SLAM, as was demonstrated in the previous experiment.

The duration of the experiment, performing two loops in the environment, was 869 s. 30 particles were used for each robot when they were doing single-robot SLAM and 100 particles for both robots when doing multiple-robot SLAM. Fig. 5-c shows the accumulated error of the localization of the map's features for the duration of the experiment.

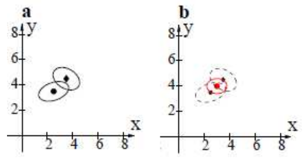

Fig. 4. Merging two features. **a)** Two features whose separation is smaller than a given threshold. **b)** The new merged feature is shown in red.

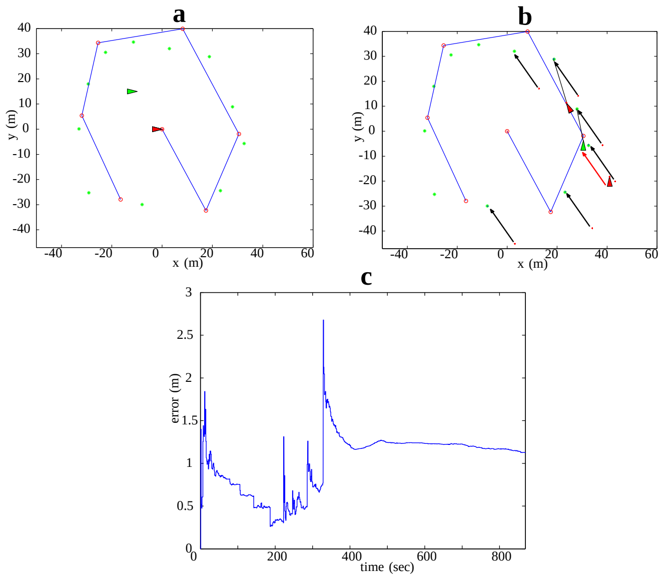

Fig. 5. The hybrid SLAM algorithm. **a)** The test environments with two robots. **b)** Maps and robots before the relative poses are known. **c)** The accumulated mean squared error of the position of the landmarks. Notice that the error is not normalized by the number of the features.

B. Real-world experiment

The real-world experiment was done with two CoroBot robots, and the results were compared with A. Howard's work [15]. The experiment was performed in a 10×7 meters VICON lab at the University of New Brunswick. Fig. 6 shows the test environments, robots, and obstacles. The VICON system records the true trajectories of the robots. The results of the proposed algorithm and A. Howard's algorithm [15] were compared with the ground truth information. The proposed algorithm was implemented with C++ in Robot Operating System (ROS). The single-robot SLAM is based on [18]. Fig. 7 demonstrates the map of the test environment at the beginning and end of the mapping process.

Initially, the robots do not know their relative pose. They perform the map merging algorithm explained in [23] until the relative poses are determined. Then $mode$ of the proposed algorithm is switched to $multipleRobotSLAM$, and the uncertainty of the pose are taken into account, using the proposed particle regeneration algorithm. Previous maps are also updated using the proposed batch integration algorithm.

The algorithm was studied for two cases. In the first case, the uncertainty of the relative poses was ignored (as presented in [15]), but in the second case, it was taken into account. After running each case for ten times, to stand for the probabilistic nature of the particle filter, it is noticed that in average the error of the trajectory (compared with the VICON trajectory) in the second case, the proposed hybrid algorithm, is 4.5% less than the first case, A. Howard's algorithm [15]. The time that was required to update the information in the batch mode was very small and was almost equal to processing ten laser scan measurements (≈ 0.25 seconds). But in A. Howard's algorithm, a significant amount of time is spent to process all the past data (≈ 12 seconds).

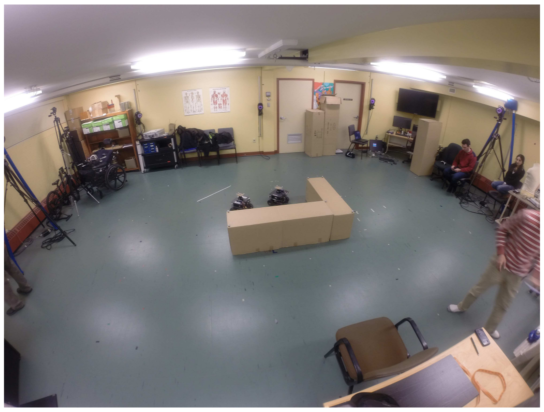

Fig. 6. For the real-world experiment, two CoroBot robots were used in a VICON lab.

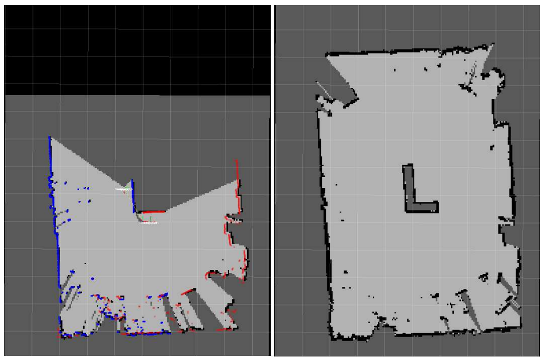

Fig. 7. Map of the VICON lab at the early stage. Blue and red points are laser beams of the Corobot robots (left). Final map of the VICON lab (right).

IV. Conclusion and Future Work

In this paper, a novel solution for mapping and localization by a team of robots was presented. The presented solution makes the assumption that the relative poses are unknown. The unknown relative poses is addressed by integrating a map merging algorithm with the particle filtering. Moreover, the uncertainty of the relative poses was taken into account using a novel algorithm. Additionally, the integration of the information was performed in a batch mode which reduces the time complexity. At the end, the developed algorithms were verified in simulated and real-world environments.

In the future, it would be desirable to perform more theoretical work to further improve the time and space complexity of the particle filter-based multiple-robot SLAM.

References

[1] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE: Special Issue on Multi-Robot Systems*, vol. 94, no. 7, pp. 1384–1387, 2006.

[2] S. Carpin, "Fast and accurate map merging for multi-robot systems," *Autonomous Robots*, vol. 25, pp. 305–316, 2008.

[3] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Map merging using Hough peak matching," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4683–4688.

[4] ——, "Efficient map merging using a probabilistic generalized Voronoi diagram," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4419–4424.

[5] V. Indelman, E. Nelson, N. Michael, and F. Dellaert, "Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization," in *International Conference on Robotics and Automation (ICRA)*, 2014, pp. 593–600.

[6] E. Olson, J. Strom, R. Goeddel, R. Morton, P. Ranganathan, and A. Richardson, "Exploration and mapping with autonomous robot teams," *Communications of the ACM*, vol. 56, no. 3, 2013.

[7] A. Cunningham, K. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1093–1100.

[8] A. Cunningham and F. Dellaert, "Large-scale experimental design for decentralized SLAM," in *Proceedings of SPIE, Unmanned Systems Technology XIV*, vol. 8387, 2012.

[9] K. Been, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," in *International Conference on Robotics and Automation (ICRA)*, 2010, pp. 3185–3192.

[10] L. Andersson and J. Nygards, "C-SAM: Multi-robot SLAM using square root information smoothing," in *International Conference on Robotics and Automation (ICRA)*, 2008, pp. 2798–2805.

[11] T. A. Vidal-Calleja, C. Berger, J. Sol, and S. Lacroix, "Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 654 – 674, 2011.

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

[13] S. Thrun and Y. Liu, "Multi-robot SLAM with sparse extended information filers," *Springer Tracts in Advanced Robotics*, vol. 15, pp. 254–266, 2005.

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

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

[16] A. Gil, O. Reinoso, M. Ballesta, and J. Miguel, "Multi-robot visual SLAM using a Rao-Blackwellized particle filter," *Robotics and Autonomous Systems*, vol. 58, pp. 68–80, 2010.

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

[18] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: The MIT press, 2005.

[20] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *International Conference on Robotics and Automation (ICRA)*, 2007, pp. 1670–1677.

[21] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *International Conference on Robotics and Automation (ICRA)*, 2010.

[22] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Group mapping: A topological approach to map merging for multiple robots," *IEEE Robotics and Automation Magazine*, vol. 21, no. 2, pp. 60–72, 2014.

[23] ——, "Map merging for multiple robots using hough peak matching," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1408–1424, 2014.