Neural Network-based Multiple Robot Simultaneous Localization and Mapping

Sajad Saeedi, Liam Paull, Michael Trentini and Howard Li

***Abstract*—In this research, a decentralized platform for Simultaneous Localization and Mapping (SLAM) with multiple robots is developed. Each robot performs single robot view-based SLAM using an Extended Kalman Filter (EKF) to fuse data from two encoders and a laser ranger. To extend this approach to multiple robot SLAM, a novel occupancy grid map fusion algorithm is proposed. Map fusion is achieved through a multi-step process that includes image pre-processing, map learning (clustering) using neural networks, relative orientation extraction using norm histogram cross correlation and a Radon transform, relative translation extraction using matching norm vectors and then verification of the results.**

The proposed map learning method is a process based on the Self Organizing Map (SOM). In the learning phase, the obstacles of the map are learned by clustering the occupied cells of the map into clusters. The learning is an unsupervised process which can be done on-the-fly without any need to have output training patterns. The clusters represent the spatial form of the map and make further analyses of the map easier and faster. Also, clusters can be interpreted as features extracted from the occupancy grid map so the map fusion problem becomes a task of matching features. Results of the experiments from tests performed on a real environment with multiple robots prove the effectiveness of the proposed solution.

***Index Terms*—Simultaneous Localization and Mapping (SLAM), Self Organizing Map (SOM), Map Fusion, Radon Transform.**

I. Introduction

Robotics systems and missions are becoming more complex. As a result, it is important for agents and sub-systems to work in parallel and share information to achieve difficult tasks more efficiently. One such task is SLAM or Simultaneous Localization and Mapping. In SLAM, a robot is placed in an *a priori* unknown environment and tries to build a map of the environment and also situate itself within the map simultaneously [1]. SLAM is useful in a variety of missions, such as: rescue operations, fire fighting, underwater and space explorations and surveillance among others. The task is very challenging because it involves processing large amounts of data from different heterogenous sensors such as cameras [2], inertial measurement units (IMUs), and laser rangers [3]. Moreover, the task must be completed in real-time with restricted onboard processing capabilities. Different variations of single robot SLAM have been proposed in the literature to account for variations in data fusion algorithms and sensing devices. Approaches include competition-based neural networks [4], Sparse Extended Information Filter (SEIF) [5], and Extended Kalman Filtering (EKF) [6] among others [7], [8].

More recently, researchers have focused on developing distributed systems to achieve SLAM. In multiple robot SLAM, many robots are used to map an unknown environment without any prior knowledge of their locations or the locations of other members of the team. Using multiple robots or agents can have many advantages, for example: the total time required to explore an environment can be greatly reduced, system performance can be improved because more information is being provided by different sources, and a distributed system is more robust to failures. However, multiple robot SLAM has the added difficulty that the maps generated from each robot must be merged to make a global map. Each agent must achieve its own localization and mapping goals while also contributing to global goals and cooperating with others. Map merging is made more difficult by the uncertainty in the individual maps and also the fact that the robots can not be assumed to know their relative poses.

Different approaches exist in the literature for overcoming these challenges. A feature-based multiple robot SLAM algorithm is proposed in [9], which uses an information filter (IF). In [10], a fuzzy logic based multiple robot SLAM is proposed where fuzzy sets are used to represent the uncertainty of the position and the fuzzy intersection operation is used to fuse data. The proposed method in [11] uses a particle filter (PF). This method uses the limiting assumption that the robots will meet each other at a point. At the meeting point, relative position and orientation of the robots are extracted and then relative initial position are determined by backwards calculation. The occupancy map merging solution proposed in [12] is an effective solution for multiple robot SLAM. In this method, a similarity index for the maps is developed based on the map-distance. Then, similar patterns in two maps are found based on a random walk algorithm. The random walk approach is not acceptable because it may be highly time consuming and inefficient, and certain patterns in maps can cause failures.

All of the previously published methods either have invalid assumptions or are so computationally intensive that the robots will have to move very slowly for real-time operation. A more efficient computational approach will result in the robots being able to move more quickly through the environment and, consequently, achieving the task in a shorter time.

Neural network algorithms are powerful artificial intelligence solutions which can be applied to any problem involving learning or training. In multiple robot SLAM there is the need to learn a map so neural networks is a good approach. After

Manuscript received January 15, 2011.

Sajad Saeedi G. is with the Department of Electrical and Computer Engineering, University of New Brunswick, Fredericton, Canada, sajad.saeedi.g@unb.ca

Liam Paull is with the Department of Electrical and Computer Engineering, University of New Brunswick, Fredericton, Canada, liam.paull@unb.ca

Michael Trentini is with Defence Research and Development Canada, Suffield, Alberta, Canada, Mike.Trentini@drdc-rddc.gc.ca

Howard Li is with the Department of Electrical and Computer Engineering, University of New Brunswick, Fredericton, Canada, howard@unb.ca

learning a map, any integration with other maps will be much faster and simpler due to the reduced dimensionality.

Many different neural network architectures exist. Feed-forward networks are popular due to their simple structure, however, the training methods require supervision, or input and output patterns, to train the network. Since the map is unknown, these input-output pairs are not available. Kohonen networks, or self organizing maps (SOM), which have a recurrent structure, use unsupervised training to reduce the dimensionality of input data. This means that only input patterns are required to train the network. This type of network is a good candidate to solve the map fusion problem because the weights and parameters of the network are calculated without any need to specify output patterns. The popularity of SOM in engineering applications is due to its ability to train without supervision, small developmental effort required, and adequate rejection of outliers [13].

There is no known literature that applies neural networks to the multiple robot SLAM problem. The proposed method is motivated by the fact that developing a fast and reliable data processing method for a team of multiple robots is very important for both accuracy and scalability of the result. A multi-step algorithm is proposed where the SOM is used to extract important information from the map for faster data processing in the subsequent steps.

In this paper a new method of map fusion for multiple robot SLAM is introduced. A new algorithm is proposed which is based on neural network theory. In this new approach, the map is learned by clustering, which reduces complexity and increases speed and accuracy. The maps are fused into a global map using relative transformation matrices determined using the clustered points.

To summarize, the contribution of this research is a layered processing algorithm including:

- A high level map segmentation algorithm for occupancy grid map preprocessing. This step prepares the map for SOM clustering.
- Application of SOM to cluster or learn the preprocessed map. By clustering, important features of the map are extracted into cluster points, which are used for map fusion. The motivation for clustering using SOM is its unsupervised training ability, and the accuracy and speed of the response.
- Inclusion of map uncertainty in the learning phase. The map is built using sensors that have inherent noise. This noise generates uncertainty in the resulting map and should be considered during the clustering process.
- Estimation of the relative transformation matrix of two maps using the cluster points. The cluster points are used to extract the relative orientation and translation of the maps.
- The use of surface norms to determine which cluster points from two different maps should be selected to determine the relative transformation.

The rest of the paper is organized as follows: Section II contains some background on SLAM and SOM, Section III presents the proposed methods, Section IV contains result of two experiments, and Section V makes some general conclusions and discusses future work.

II. Background of Research

The objective of this mission is for a team of robots to efficiently explore an unknown environment. This task will be achieved using many results from SLAM and SOM theory.

A. Localization

Localization involves using natural landmarks or environmental characteristics to help a robot situate itself within a known map. This includes reading data from various sensors and associating that data in order to find correspondences. GPS is the most commonly used outdoor localization method. Vicon, a system of multiple fixed cameras, is widely used as an indoor localization method. The most basic form of localization is dead reckoning, which is simply an estimation of the vehicle pose by integrating estimates of its motion. *A priori* map based localization is based on exploring the environment in advance, and to have surveyed the landmark locations before the robot can begin to navigate autonomously.

B. Mapping

Robotic mapping is the task of generating a map while navigating through the environment and observing it with sensors. Four types of maps are commonly used for map representation: occupancy grids, feature maps, topological maps, and hybrid topological maps.

1) Occupancy Grid Maps: Maps are represented as a matrix of cells. Each cell represents the probability that a small rectangular area in the environment is occupied as a binary random variable. These maps work well in two dimensions, but in general scale very poorly with map size.

2) Feature Maps: Also referred to as landmark maps, these maps represent the environment by the global locations of parametric features. This type of map representation has been shown to be effective for localization, however, it relies on the presence of simple and distinct features.

3) Topological Maps: Maps that break the world up into locally connected regions and avoid the problems of maintaining a global reference frame. These types of maps can achieve more abstract representation of complex space at the cost of requiring more advanced algorithms for map generation.

4) Hybrid Topological Maps: Any map that uses some of the aspects of topological and other types of maps. Hybrid topological maps Have been shown to be effective for feature-based and view-based SLAM. In this paper occupancy grid maps will be transformed into a reduced topological representation for map merging using the SOM.

C. SLAM

The motivation for SLAM is to overcome the need for *a priori* maps as a mechanism for bounded pose uncertainty, and to enable map construction that is extensible and adaptive to environmental change. Feature-based SLAM is performed by storing landmarks in a map as they are observed by the

robot sensors, using the robot pose estimate to determine the landmark locations, while at the same time, using these landmarks to improve the robot pose estimate [14]. SLAM is then achieved through a combination of sensor data fusion and filtering and integrating maps based on the observed features. In view-based SLAM, no landmarks are extracted but data association methods like scan matching are used to improve the mapping and localization [6].

D. Multiple Robots

A group of robots can achieve goals one single robot cannot achieve. In addition, multi-robot systems can be more efficient and robust. It is more challenging to control a group of robots. New issues arise when a group of robots work together, for example:

- Complex and dynamic environments: Having an environment with multiple robots creates a dynamic world that is difficult to model. The interaction between the robot and the environment becomes more difficult to predict.
- Scalability of control: Each robot will need to achieve both global and local goals. In order to achieve global goals, a centralized control paradigm can be used. The centralized control paradigm is characterized by a complex central processing unit that is designed to solve the whole problem. The central unit must gather the data from the whole system. The solution algorithms are necessarily complex and problem specific and do not tend to scale well. Decentralized control paradigms are based on distributed control in which individual components react to local conditions simultaneously. These individual components interact with neighboring components to exhibit desired adaptive behaviors. The complex behaviors are a resultant property of the system of connections and are therefore not prone to scalability issues [15].
- Coordination: Groups of robots need to coordinate their behaviors in order to complete the tasks as a team while avoiding conflicts such as through negotiation or bargaining.
- Cooperation: Cooperation is achieved when agents are able to perform actions that have mutual benefits. Cooperation implies coordination [16].
- Limited communication channels: When multiple robots work together, information needs to be shared among robots. Limited communication channels with random time delays can create new challenges for a multiple robot system.

E. Self-Organizing Map

The SOM is a powerful neural network paradigm which can detect internal correlations of its input vectors and categorize them according to their similarity. The SOM is an orderly mapping of a data set with high-dimensional distribution onto a low-dimensional structure. It is mainly used for the visualization and abstraction of large data sets [17].

The mapping of SOM preserves the topologic-metric relations between input data. The mapping projection is performed using a matching process where a generalized model is assumed to be associated with each grid location. For each input item, the closest model is identified. Finally the collection of models is optimized such that all inputs are approximated [18].

The SOM is constructed based on the competitor networks where inhibitory and exhibitory effects of neurons on each other provide the learning infrastructure. An SOM is trained using the Kohonen learning rule and the inputs are presented in random order [19]. The training process starts with identifying the winning neuron for each given input vector. Then weight vectors in the neighborhood of the winner neuron are set to the average position of all input vectors. The process is performed for a specified number of iterations.

Consider an input sample, $x(k) \in \mathbf{R}^2$, where k is the sample index, with the last weights computed for the i^{th} neuron as $w_i(k) \in \mathbf{R}^2$. When a new input sample is applied, the improved weights of the neuron are determined using a recursive rule

$$w_i(k+1) = w_i(k) + h_i(k)(x(k) - w_i(k)), \qquad (1)$$

where h_i is the neighborhood function. The neuron with the minimum distance is called the winner [17]. More detailed training methods are found in [19].

III. Proposed Methods

Two important issues with multiple robot SLAM are finding the relative initial poses of the robots and coping with the uncertainty of the maps of the robots. Since the maps are considered to be occupancy grid maps, the fusion algorithm should take into account the associated uncertainties. In other words, more emphasis should be given to the cells with lower uncertainties of occupancy. In this research, the map learning process is performed by clustering the map into specific clusters using unsupervised Kohonen networks. The clusters represent the main characteristics and features of the map and make further map processing easier and faster.

The main goal of the proposed method is to find the relative transformation between maps as quickly and reliably as possible, while considering the uncertainties. An overview of the method is shown in Fig. 1. The inputs of the algorithm are the two occupancy maps and the output is their relative transformation.

A. Map Segmentation

The first operational layer that is performed is map segmentation. This preprocessing step helps to produce better results in the remaining steps. The motivation for this preprocessing is the observation that the map is usually composed of a few major segments which are relatively far from each other. The segmentation identifies discontinuous segments of obstacles located far enough from each other that they should be considered as separate. This is necessary so that the SOM algorithm can be run on each contiguous segment of obstacle in the map. Otherwise, the SOM will produce clusters that are not close to obstacles but could be the midpoint between two separate obstacles.

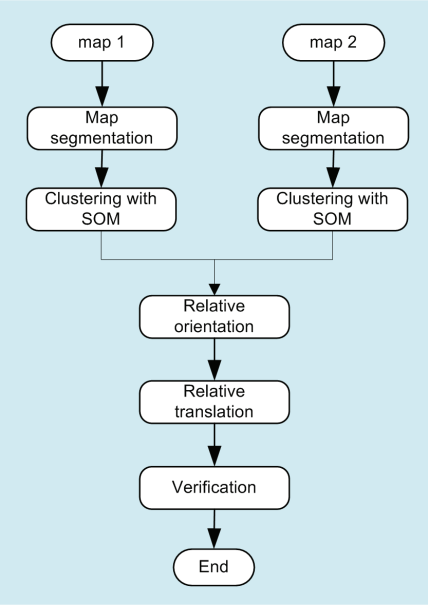

Fig. 1. Flow chart of the proposed algorithm

Algorithm 1 Map Segmentation

Input: Set of occupied cells from occupancy grid map (M_{occ})
Search radius (r)
Search radius increase step (Δr)
Output: n set of segments: C^i, $i = 1, \dots n$

1: $i \leftarrow 0$
2: $M \leftarrow M_{occ}$
while $M \neq \varnothing$ **do**
4: $i \leftarrow i + 1$
$p_0 \leftarrow$ random element of M.
6: $S \leftarrow \{p_0\}$.
while $S \neq \varnothing$ **do**
8: $p_s \leftarrow (x_s, y_s) \leftarrow$ random element of S
$C_s \leftarrow \{p = (x, y) | (x - x_s)^2 + (y - y_s)^2 \leq r^2\}$
10: $S_s \leftarrow \{p = (x, y) | r^2 < (x - x_s)^2 + (y - y_s)^2 < (r + 1)^2\}$
if $S_s = \varnothing$ **then**
12: $S_s \leftarrow \{p = (x, y) | r^2 < (x - x_s)^2 + (y - y_s)^2 < (r + \Delta r)^2\}$
end if
14: $S \leftarrow S \bigcup (S_s - C^i) - p_s$
$C^i \leftarrow C^i \bigcup C_s \bigcup S_s$
16: **end while**
$M \leftarrow M - C^i$
18: **end while**

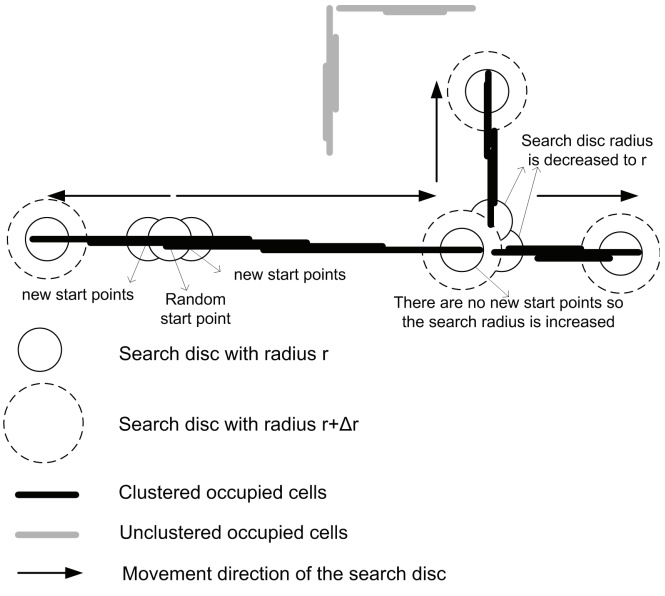

Fig. 2. Map Segmentation as described in Algorithm (1)

The pseudo-code for map segmentation is given in Algorithm 1. In line 2, M is defined as the set of all unclustered obstacle points in the occupancy grid map. A random point, $p_0 = (x_0, y_0)$, called a *start point* is selected from M in line 5. The set, S, is defined as the set of all start points. The set that contains all points in a segment i is $C^i, i = 1..n$. The algorithm continues for each segment as long as there are start points remaining in S to be processed.

For each start point, $p_s = (x_s, y_s)$, the set of neighboring points, C_s, is determined in line 9. The set of potential new start points, S_s is calculated in lines 10 and 12, where the ring radius begins as 1, and then increases to Δr if no results are found. The reason for doing this process in two steps is to try to limit the number of potential next start points as much as possible to increase the speed of the algorithm.

In line 14, the set of next start points is updated to include the members of S_s that are not already in the segment C^i, and the current start point, p_s is removed. The segment, C^i, is updated to include the values of C_s and S_s.

Once S is empty, the segmentation is finished. The points in the segment are removed from M and i is increased. The process continues until M is empty. Fig. 2 shows a graphical representation of the process.

The extracted segments of each map are passed to the next level where the SOM is applied on each segment to cluster it. Segments with small size do not provide enough information for map fusion, so they are not used for clustering.

B. Clustering with SOM

Once the map segmentation is complete, the clustering is performed on each segment. The clusters, which are generated by applying SOM, will be used to fuse the two maps (refer to Fig. 1). An occupancy grid map is composed of a large number of cells which are time consuming to process individually. The purpose of performing this clustering is to decrease the time required for map fusing. The clusters preserve the original information or shape of the obstacles or features from the map but represents them in a much more compact form.

Fig. 3 shows the simple structure of the network. In this case the data is a two dimensional position of an occupied cell in the occupancy grid map. This should not be confused with the locations of the neurons in the solution space, which are also two dimensional locations. In the training phase, the location of each occupied cell is presented as the input of the network. The network finds the best matching or winning

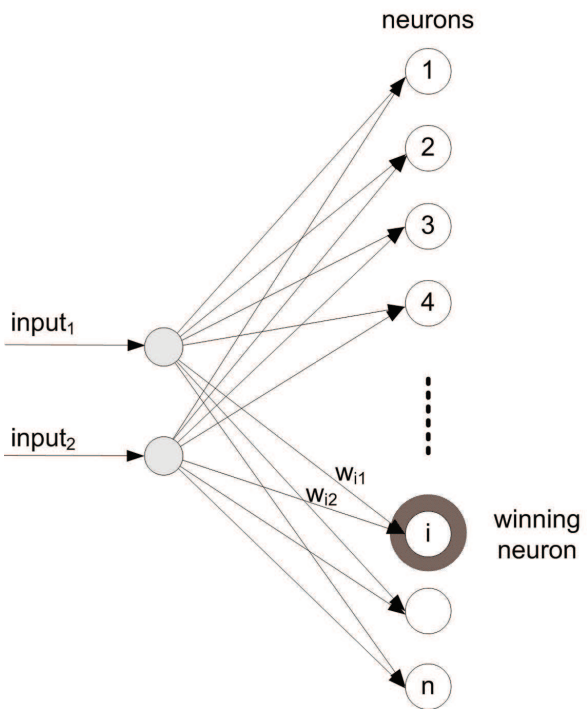

Fig. 3. Structure of the SOM.

neuron to be the one whose weights, or data, are closest to that input position. The weights corresponding to that neuron and the neurons that are in the neighborhood in the solution space are updated to better represent that input pattern. Once the algorithm is run for several iterations, the final weights are the clusters that represent the data.

The number of neurons or clusters, n is an important design parameter. Its selection is dependant on the size of the map, and it is noted that the choice of n has a significant effect on the quality of the results. In this paper the number of neurons is assigned proportionally to N, the number of the occupied cells in the map.

$$n = round(\frac{N}{\eta}), \tag{2}$$

where η is a scaling gain.

It is emphasized that the training process and then subsequent map fusion are fast enough that they can be implemented onboard a robotic platform.

1) Map Uncertainty: A major advantage of the occupancy grid map representation is the treatment of uncertainty. Values in an occupancy grid map are float values in the range $(-1, 1)$ where negative values indicate belief of an obstacle, and positive values indicate belief that the cell is free. At first, all cells are initialized to 0, indicating that the state of the cell is unknown. As sensor data is processed, the values in the map are adjusted. For example, if a cell is detected as being occupied by many scans, it will have a value closer to -1 than if it had only been detected as occupied a small number of times.

In order to account for this uncertainty, input patterns with higher certainty of being obstacles are presented more frequently as the input in the training of the SOM.

Here, each time a cell, C, is detected as occupied, the value is decreased by some tunable parameter Δ to a minimum of -1. Alternately, each time C is detected as free, its value is increased by Δ, to a maximum of 1. Only cells that have negative values (obstacles) are used in the training of the SOM. The relative frequency that each input sample is presented is determined by the relation:

$$f(C) = round(\frac{|C|}{\Delta}), \tag{3}$$

where the ratio of $|C|$ to Δ represents the number of times that the cell was detected as an obstacle. This value is rounded such that the cell C appears in the training phase with a frequency of $f(C)$. For example, if $\Delta = 0.1$, a cell with a value of -0.4 should appear as the input to the SOM four times more often than a cell with a value of -0.1.

Once the clustering has been performed, the relative transformation between the maps can be found efficiently. This involves extracting the relative orientation and translation separately as will be discussed in the next sections.

C. Relative Orientation

The arrangement of the cluster points by the SOM represents the approximate map of the world. The surface which is created by connecting the clustered points is a representation of the real surface of the world. Norm vectors are a way of mathematically representing a surface that is defined by points. By comparing the norm vectors of two maps, it is possible to determine if they are a match, and then subsequently extract their relative orientation. Fig. 4 shows an illustration of four cluster points. The surface generated by connecting adjacent clusters is termed the cluster surface (dotted line). The cluster surface norms are unit vectors that are perpendicular to the cluster surface and are equidistant from two clusters. The relative orientation between the two maps is determined by performing a 360° histogram on the directions of the cluster surface norms, and then matching the histograms of the two maps. Some sample histograms are shown in the experimental results section. This is similar to the circular cross correlation of histogram norms that is presented in [6] for aligning two consecutive local maps. However, because the number of data points has been effectively reduced through clustering, the size of the buckets in the histogram must be relatively large. As a result, it is not possible to get sufficiently accurate results. A tuning method based on the Radon transform is used to find a more accurate value for the relative orientation [20].

1) Tuning with the Radon Transform: The Radon transform is the projection of the image intensity along a radial line oriented at a specific angle [21]. The Radon image is generated by computing a Radon transform and varying the Radon angle, θ, of the radial line onto which the original image is being projected. For a given map, $m(x, y)$, the Radon transform along the radial line of angle θ is defined as:

$$r_\theta(x') = \int_{-\infty}^{\infty} m(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)dy', \tag{4}$$

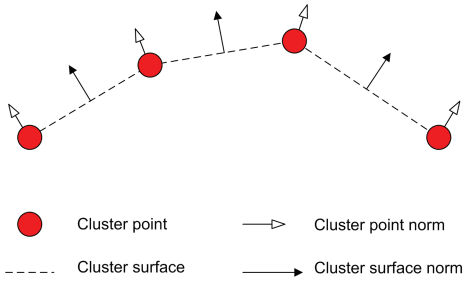

Fig. 4. surface norm is the perpendicular vector to the surface and the point norm is the average norm of the two line segments connecting the point to two adjacent points. If the point is connected to just one point, the point norm is the same as the surface norm.

where

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \quad (5)$$

The Radon image generated by computing a Radon transform is a collection of all possible projections for a larger set of angles. In a Radon image, the horizontal axis represents all possible angles and the vertical axis is the Radon transform for individual angles. One of the characteristics of the Radon image is that the peak points occur when the Radon angle, θ, aligns with straight line segments that occur in the image. As a result, it is possible to resolve the exact rotation by looking for peaks in the Radon images of both maps that are close to the approximate angle previously computed using the histogram method. The difference of the peak angles is the tuning angle that should be added to the original approximate angle as determined by:

$$\alpha = \alpha_{app} + \delta_{tuning}, \quad (6)$$

where α is the tuned rotation angle, α_{app} is the result of the cross correlation, δ_{tuning} is the tuning angle added to α_{app} to adjust it and is defined as:

$$\delta_{tuning} = \arg\min_{\theta}(\mathcal{R}_{\theta=0:180}(m_1)) - \arg\min_{\theta}(\mathcal{R}_{\theta=0:180}(T_\omega(m_2))), \quad (7)$$

where $\mathcal{R}_{\theta=0:180}(map)$ is the Radon image of the input map over the specified domain of angles (θ). The operation $T_\omega(map)$ is a rotation of the input, map, by ω and $m_1 = map_1$ and $m_2 = map_2$. The first $\arg\min(\cdot)$ yields an angle corresponding to a straight line of cluster-points in map_1. $T_\omega(m_2)$ rotates map_2 according to the extracted approximate rotation angle ($\omega = \alpha_{app}$). Now the second $\arg\min(\cdot)$ gives the angle corresponding to straight line of cluster-points of the rotated map_2. The difference between these two angles is the tuning angle to be used in (6).

Once the tuned relative orientation has been found, it is applied so that the relative translation can be found as will be discussed presently.

D. Relative Translation

Once the relative rotation has been determined, it is highly probable that points with a similar orientation as determined by the norm vectors actually correspond to the same point in the world. This fact is used to do an efficient search to associate points from different maps to each other by finding their relative translation. Referring again to Fig. 4, the cluster point norms are unit vectors that originate from the clusters at an angle that is the average of angles of the two adjacent cluster surface norms. For cluster points that are at the end of the cluster surface, the point norms are parallel to the surface norms. These point norms are used to find the relative translation between the two maps using an iterative approach similar to the Iterative Closest Point (ICP) algorithm [22]. Pseudo-code for the algorithm is presented in Alg. 2.

The inputs are $M_1 = \{m_1^1 ... m_1^{K_1}\}$, and $M_2 = \{m_2^1 ... m_2^{K_2}\}$, the sets of cluster points for map_1 and map_2 respectively after the relative rotation has been applied, and the associated sets of point norms $N_1 = \{n_1^1 ... n_1^{K_1}\}$, and $N_2 = \{n_2^1 ... n_2^{K_2}\}$ (the norm of m_1^k is n_1^k for all k. Similar for M_2). Without loss of generality, M_2 is being translated to M_1.

On line 5 a new set, P_k ($P_k \subseteq M_1$) is built that contains the points in M_1 that have norms within a matching angle threshold, ϵ of n_2^k (the point norm of m_2^k). If this set P_k is non-empty, then the point in P_k that has the smallest Euclidean distance from m_2^k is chosen as the winner and a correspondence is made. These correspondences are maintained in two new sets, PT_1 and PT_2, as shown on lines 7 and 8. If the set P_k is empty, it means that there were no points in M_2 with similar norms, or that this part of map_2 has no matching part in map_1, so no correspondence is made and the algorithm continues.

Once all of the points in M_2 have been considered, the best translation is found on lines 12-14. Conceptually, δ_x and δ_y are the means of errors between corresponding matching points from the two maps along the x and y axes respectively.

Once the translation is found, it is applied to the set M_1, the matching angle threshold is reduced and the algorithm is restarted. The stopping criterion is either a number of iterations or a minimum error threshold is reached between the two cluster sets M_1 and M_2.

It is noted that the selection of the matching angle threshold, ϵ, is quite important. Values that are too large give inaccurate results and values that are too small do not yield any correspondences. An effective tradeoff is to reduce the value of ϵ at every iteration. In this case it was initialized to $25°$ and reduced gradually until it reached a minimum of $4°$.

E. Verification

A final verification process is used to eliminate false matching or to select the best result of a few candidates. The verification method is based on the convergence of the performance index, J as given by:

$$J = \sum_{i=1}^{n} ||p_1^{\{i\}} - p_2^{\{i\}}||, \quad (8)$$

where n is the number of clusters, and p_1^i and p_2^i are the corresponding cluster points for map_1 and map_2 respectively. J is the sum of the squared Euclidian distances between

Algorithm 2 Translation realization based on the norms of the cluster points

Input: The matching angle threshold (ϵ)
The maximum number of iterations: $iterations_{max}$
The error threshold: J_{thresh}
Sets of cluster points M_1 and M_2 with associated sets of point norms N_1 and N_2
Output: Translation, T
$iterations \leftarrow 0$
repeat
  i = 1;
  for $k = 1 \rightarrow K_2$ **do**
    $P_k \leftarrow$ {All points $p = (x, y)|p = m_1 \in M_1$ whose associated norm n_1 satisfies $|n_1 - n_2^k| < \epsilon$}
    if $P_k \neq \varnothing$ **then**
      $PT_1[i] \leftarrow$ element of P_k that is closest to m_2^k
      $PT_2[i] \leftarrow m_2^k$
      $i \leftarrow i + 1$
    end if
  end for
  $\delta_x \leftarrow \frac{1}{i-1}(\sum_{l=1}^{i-1} PT_{1x}[l] - \sum_{l=1}^{i-1} PT_{2x}[l])$
  $\delta_y \leftarrow \frac{1}{i-1}(\sum_{l=1}^{i-1} PT_{1y}[l] - \sum_{l=1}^{i-1} PT_{2y}[l])$
  $T \leftarrow \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}$
  Each point in M_2 gets shifted by T
  $J = \sum_{l=1}^{i-1} ||PT_1^l - PT_2^l||$
  Reduce t_m.
until $J < J_{thresh}$ or $iterations > iterations_{max}$

corresponding matching points. To have a perfect matching, J should be a small number. J is calculated for each iteration of the relative translation calculation. If J converges then the fusion of the clusters can be considered as valid and the next verification method ensures the accuracy of the results.

In order to ensure that the entire maps have been accurately fused, the verification index given in (9) is used [12], where the relative orientation and translation have been applied to map_2 .

$$V(map_1, map_2) = \frac{agr(map_1, map_2) \times 100\%}{agr(map_1, map_2) + dis(map_1, map_2)}, \tag{9}$$

The two components $arg(m_1, m_2)$ and $dis(m_1, m_2)$ are defined as:

$$\begin{aligned} agr(map_1, map_2) &= \sharp\{p = (x, y)|map_1(p) = map_2(p)\}, \\ dis(map_1, map_2) &= \sharp\{p = (x, y)|map_1(p) \neq map_2(p)\}, \end{aligned} \tag{10}$$

where the operator $\sharp$ over a given set returns the cardinality of the set. The function $agr(.)$, the agreement index, is the number of known cells with equal status in both maps (either both occupied or both free). The function $dis(.)$, the disagreement index, is the number of cells which are known in at least one map and have unequal status.

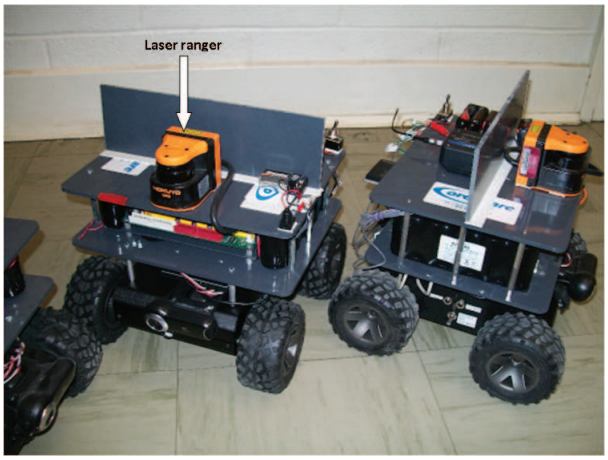

Fig. 5. Robots equipped with the laser rangers and encoders

The verification index $V(map_1, map_2)$ gives a measure of how similar the two maps are. A similarity index of 100% would mean that the two maps are identical.

F. Discussion: Scalability

The capability for supporting multiple agents is one concern of any robotics algorithm. As mentioned in section III-B, an important advantage of the SOM clustering method is a down-scaling of the grid map into cluster points. This down-scaling reduces the required time for map fusion and enhances scalability. The training phase of the SOM is computationally intensive, but time savings in the map fusion make it worthwhile. This allows more robots to be added to the team while still operating efficiently. The down-scaled map includes the most important information and any loss of information will be accounted for by fine tuning with the Radon transform.

Deploying a parallel hierarchial structure among the members of a team can save tremendous amounts of time. For example, in a group of six robots, robots 1 and 2 can fuse maps and submit the result to robot 3. Simultaneously, robots 4 and 5 can do the same and submit the result to robot 6. Then the global map is obtained from the map fusion from robots 5 and 6. The result should be submitted to all robots in the team.

IV. **Experiment**

Two experiments are performed on a real-world indoor environments. Two differential-wheeled robots built by CoroWare, Inc. are used and each is equipped with two High Speed Phidget Encoders and a UBG-05LN laser ranger from Hokuyo, as shown in Fig. 5. The laser ranger sensor has a range of 4500 millimeters with an angle resolution of 0.36 degrees resulting in 513 points for each scan. Data from these sensors are fused with an Extended Kalman Filter (EKF) [23]. The encoder is used to provide an estimation of the control signal. Consecutive scans of the laser ranger are preprocessed by an Iterative Closest Point (ICP) algorithm [24] to update the pose. The result is the position of the robot and the map of the environment which are shared with other robots. Maps are in the occupancy grid format with the size of 400×400

cells. Gray cells are unknown cells, white cells are free cells, and black cells are occupied cells. All figures and codes are developed using MATLAB functions and environment.

A. Experiment 1: Simple Test Environment

Fig. 6 shows the approximate floor plan and dimensions of an indoor environment used for the experiment. This room has been surveyed by two robots from different initial positions which are assumed to be unknown.

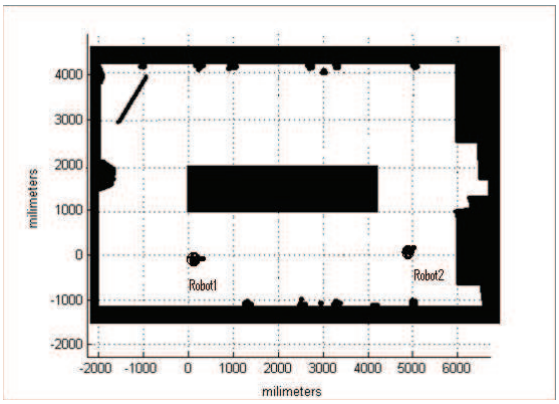

Fig. 6. A simple test environment

Fig. 7-a shows the local map provided by $robot_1$, and Fig. 8-a shows the local map provided by $robot_2$.

After applying map segmentation to both maps, Fig. 7-b and Fig. 8-b show the different map segments in different colors. Since small segments do not carry enough information for map fusion, only the larger segments from each of the maps are matched. Runs of the map segmentation on a 2.99 GHz dual core Intel based computer yielded an average run-time of about 1 second. The results of the clustering using SOM are shown in Fig. 7-c and Fig. 8-c. Clusters tend to keep the spatial characteristics of the map, so any transformation which can fuse the clusters is a good estimate for the fusion of the maps. The training of the SOM is the most time consuming step of the process. Referring to (2), map_1 has 685 occupied cells and map_2 has 715. Assuming η to be $18, 38$ neurons are assigned for map_1 and 40 for map_2. η is chosen by experiment. For 40 cluster points, the average time for training is about 8 seconds and the result converges after 5 iterations. Fig. 7-d and Fig. 8-d depict the normals histogram of the clustered points. Applying circular correlation to the orientation histograms, gives the relative orientation of 57.5 degrees. Fig. 9-a shows the clusters after the rotation by the cross correlation. Fig. 10-a and Fig. 11-a show the Radon images for map_1 and map_2 respectively after applying the approximate transformation. As Fig. 10-b and Fig. 11-b show, the peak point of the Radon image for map_1 happens to be at 91 degrees while for the rotated map_2 it is 93 degrees. The difference of 2 degrees is for tuning. Fig. 9-b shows the clusters after rotation tuning by the Radon transform. Using the proposed iterative approach, translation is found. In Fig. 9-c, the translated clusters are shown. The translation vector is calculated as $(49.3, 59.8)$ along x and y axis. Finally, Fig. 9-d shows both maps fused using the extracted rotation and translations. Fig 12 shows

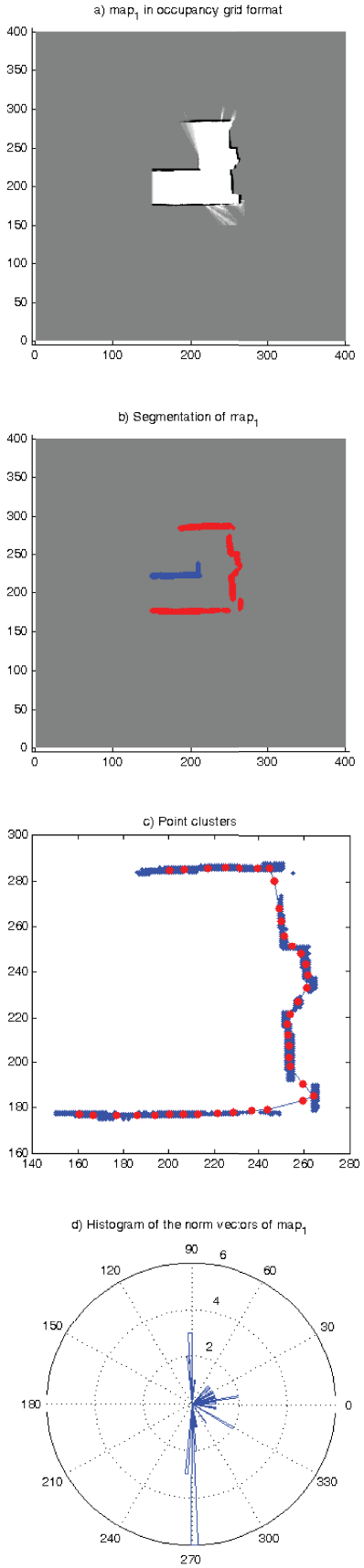

Fig. 7. **a)** map_1, occupancy grid map of the test environment provided by $robot_1$. **b)** Map segmentation in different colors developed using algorithm (1). **c)** Clusters developed using SOM. **d)** Histogram of the norm vectors.

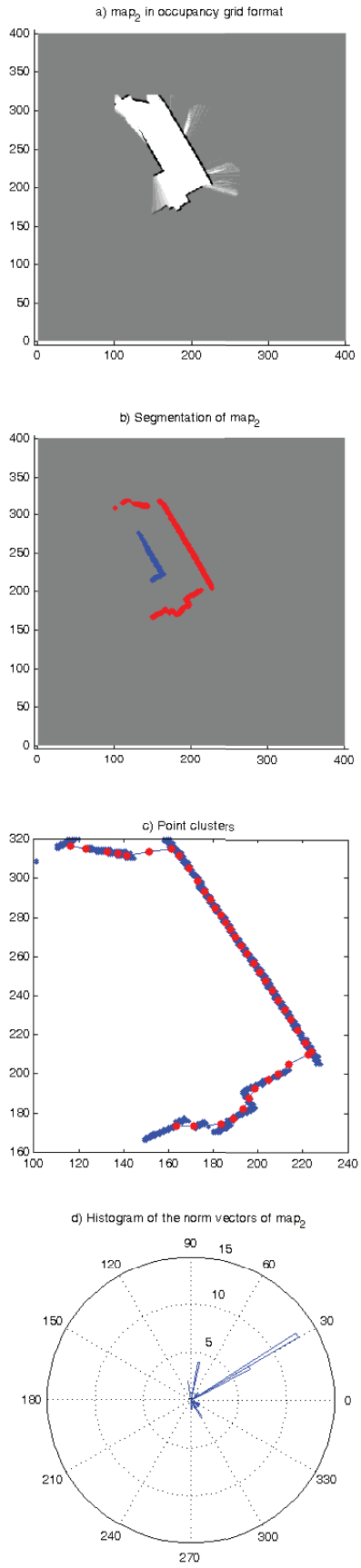

Fig. 8. **a)** map_2, occupancy grid map of the test environment provided by $robot_2$. **b)** Map segmentation in different colors developed using algorithm (1). **c)** Clusters developed using SOM. **d)** Histogram of the norm vectors.

the performance index defined in (8). The error converges after 15 iterations. The algorithm may be stopped at this point. The verification index, $V(map_1, map_2)$ defined in (9) is 95%, which shows that the two maps are indeed quite well matched.

B. Experiment 2: More Complex Environment

To demonstrate the effectiveness of the proposed methods, another experiment is performed in a larger environment with the approximate size of 17×10 meters. Fig. 13 shows the approximate floor plan of the test environment with the paths of the robots.

As in the previous experiment, each robot generates a local occupancy grid map of the environment. At certain specified time intervals, the robots share their local maps with each other. Fig. 14-a and Fig. 14-b show the two local maps generated by the robots. Note that the left part of the test environment was out of the range of the laser range scanners. The two maps, map_1 and map_2, are provided by $robot_1$ and $robot_2$ respectively. The proposed map fusion algorithm is assumed to be done on $robot_1$ with map_2 being integrated into map_1. Fig. 14-c shows the result of the map fusion of map_2 into map_1 using the proposed method. The transformation vector for this case is $\boldsymbol{T}_{x,y,\theta} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T = \begin{bmatrix} -1.2 & -32.9 & 26 \end{bmatrix}^T$. The verification index, $V(map_1, map_2)$ defined in (9) is 95%.

The proposed method has been compared with three other methods: adaptive random walk (ARW) map merging [12], map segmentation [20], and k-means clustering [25].

ARW is based on search and verification. In ARW, the similarity indices of two maps are calculated given a set of transformations. The set is composed of known rotations and translations. The transformation corresponding to the best similarity index is selected as the start point of an adaptive random walk search to merge the maps more accurately. Finding the initial transformation from the set takes the majority of the processing time.

Map segmentation is based on extracting thin lines from objects and walls of both maps and matching them. First, edges are found using Canny edge detection. Then, subsequent lines are extracted by smoothing the Canny edges. Edges are selected for cross correlation matching using a histogram filter.

K-means clustering is an iterative partitioning where each point belongs to the cluster with the nearest mean. This method clusters points relatively quickly, but the clusters lack any kind of structured order. The structured ordering of the SOM neurons is the essential advantage that allows the collection of neurons to maintain the topological information contained in the map. Tamayo et al. state this as, “k-means clustering is a completely unstructured approach, which proceeds in an extremely local fashion and produces an unorganized collection of clusters that is not conducive to interpretation”([26]). As a result, k-means is not a practical alternative to the SOM within the framework of the proposed algorithm in this paper.

Table I summarizes the comparison of the processing times for the two experiments. In the first experiment, the algorithm takes about 13 seconds to run. This is about one sixth of the time that the algorithm presented in [20] required to run on the

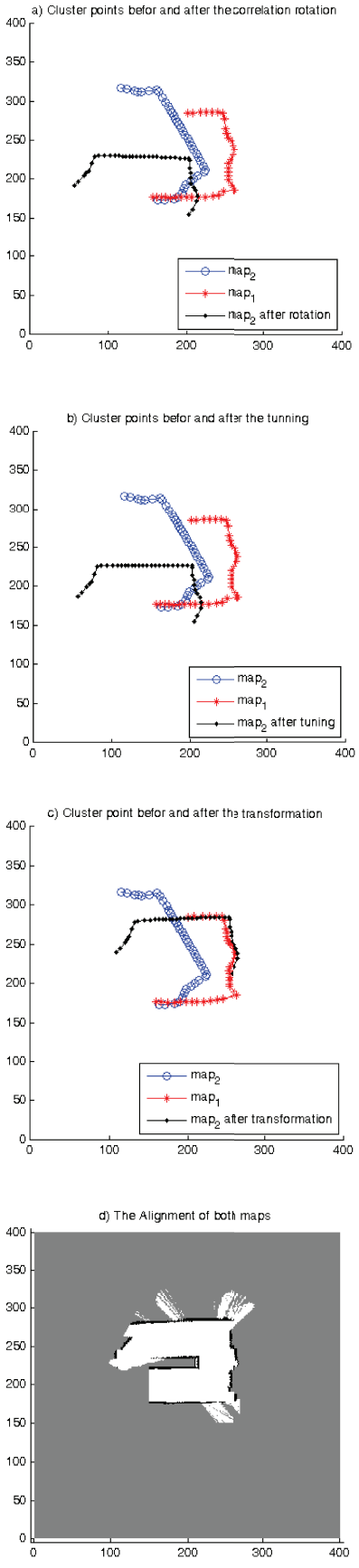

Fig. 9. **a)** Both maps and map_2 after the rotation. **b)** Both maps and map_2 after applying the rotation and the translation. **c)** map_1 and map_2 after applying the extracted relative transformation.

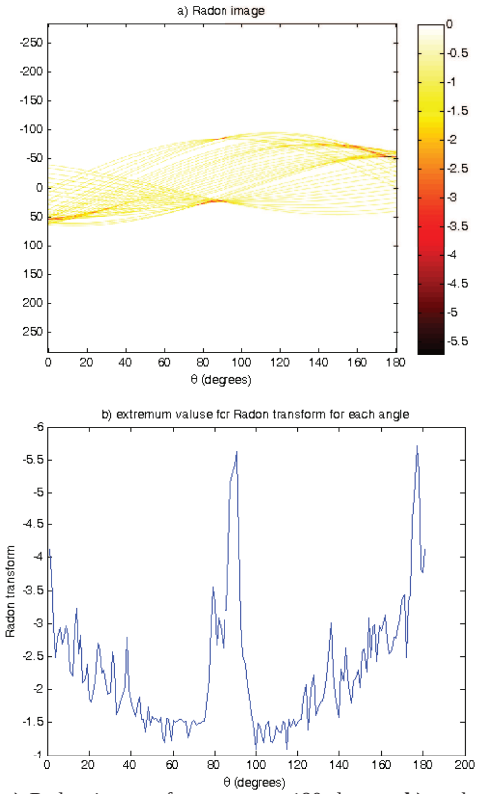

Fig. 10. **a)** Radon image of map_1 over 180 degrees **b)** peak point of the Radon image

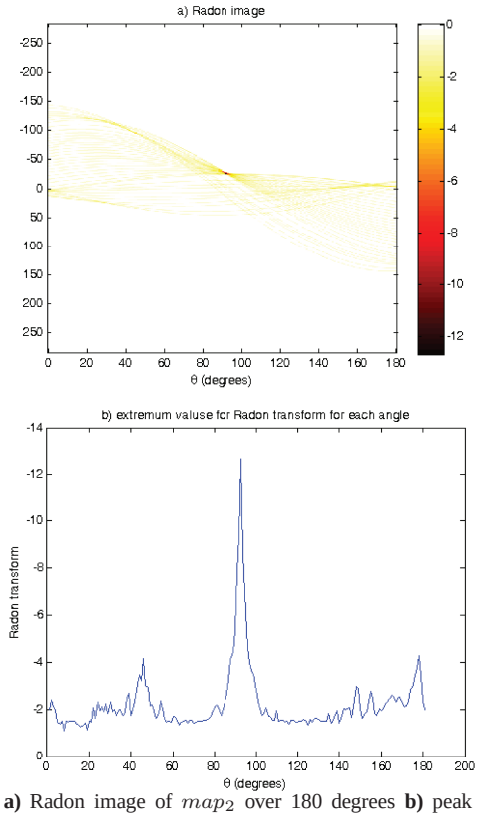

Fig. 11. **a)** Radon image of map_2 over 180 degrees **b)** peak point of the Radon image

identical environment producing similar results. For the second experiment, the entire algorithm required about 16 seconds to

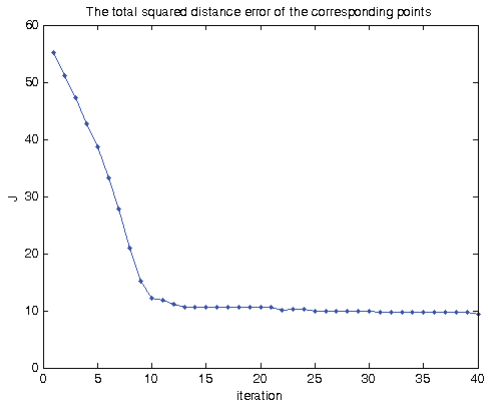

Fig. 12. Performance index of the map fusion based on the squared distance error of the corresponding points. The error converges after 15 iterations to a to a fixed error value.

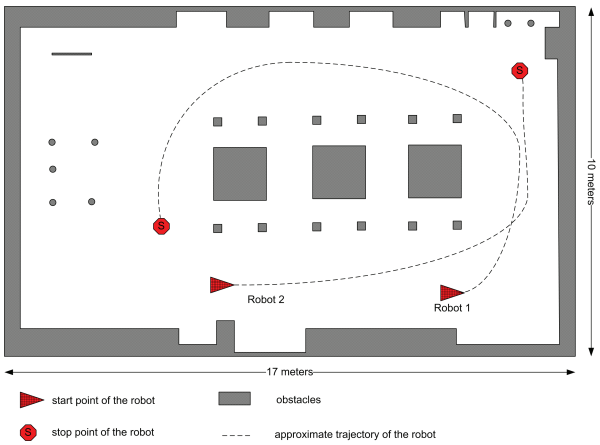

Fig. 13. A more complex environment. Starting and ending locations of the robots are marked with triangles and stop signs, respectively. The robot trajectories are shown as dashed lines.

run, which compares with over 1.5 minutes for the algorithm presented in [20]. To compare the results with ARW, a faster version of ARW is used where a set of 144 transformations have been investigated. In this case the proposed algorithm runs at least ten times faster. The reduced processing time is mainly due to the down-scaling of the map by SOM.

Method	Experiment 1	Experiment 2
SOM based	≈ 13 sec	≈ 16 sec
Map segmentation [20]	≈ 84 sec	≈ 95 sec
ARW map merging [12]	≈ 152 sec	≈ 160 sec

TABLE I
COMPARISON OF PROCESSING TIMES

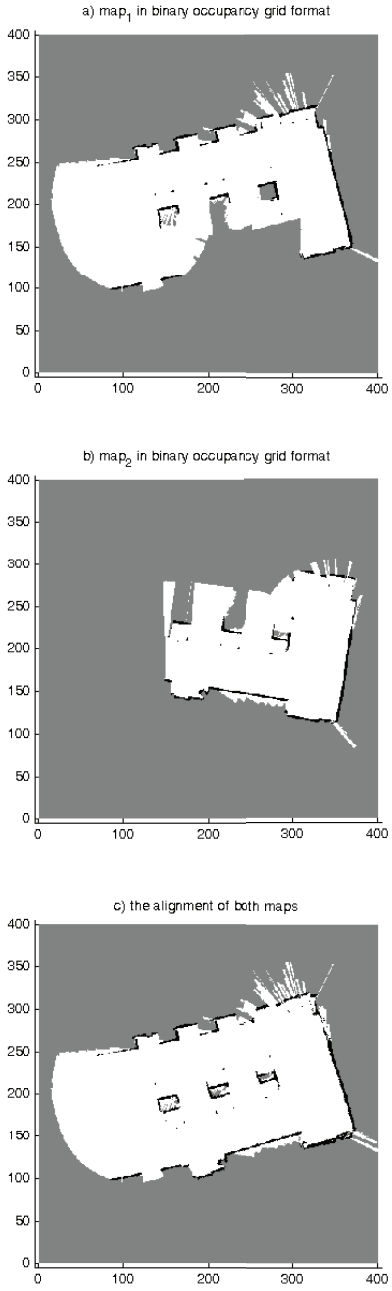

Fig. 14. Local maps provided by the robots based on the approximate dimensions and trajectories of Fig. 13. **a)** map_1 provided by $robot_1$, **b)** map_2 provided by $robot_2$, **c)** both maps after the alignment by the proposed algorithm.

V. CONCLUSION AND FUTURE WORK

In this paper, a new method for multiple robot Simultaneous Localization and Mapping is developed. The proposed method uses a self-organizing map to reduce the complexity of the acquired occupancy grid maps. The result is that map fusion can be achieved more efficiently and reliably. Results are verified with tests performed in real environments. This is the first known application of neural network theory to solve the multiple robot SLAM problem. In summary, novel aspects of this approach include: preprocessing of occupancy grid maps using map segmentation method, applying self organizing map to learn and cluster the map, determining the relative rotation and the translation of two maps using the cluster points. A major advantage of this approach is its fast result due to the unsupervised training method of the SOM. This will allow faster exploration and mapping of unknown environments, which is useful in a variety of different robotics applications.

In the future, incorporating information gained if the robots do happen to see each other in the environment could yield better results because it provides a way for the robots to

quickly and easily find their relative poses. Scalability of the proposed algorithm to large numbers of robots should also be investigated further.

ACKNOWLEDGEMENT

This research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Canada Foundation for Innovation.

REFERENCES

[1] H. D. Whyte and T. Bailey, "Simultaneous localization and mapping (slam): Part i the essential algorithms," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[2] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595–599, May 2009.

[3] E. Stump, V. Kumar, B. Grocholsky, and P. M. Shiroma, "Control for localization of targets using range-only sensors," *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 743–757, June 2009.

[4] G. Wyeth and M. Milford, "Spatial cognition for robots," *IEEE Robotics and Automation Magazine*, vol. 16, no. 3, pp. 24–32, September 2009.

[5] C. A. Borja, J. M. Mirats Tur, and J. L. Gordillo, "State your position," *IEEE Robotics and Automation Magazine*, vol. 16, no. 2, pp. 82–90, June 2009.

[6] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based slam," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, June 2008.

[7] B. D. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs, N.J., USA: Prentice-Hall, Inc., 1979.

[8] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based slam," *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, April 2007.

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

[10] K. LeBlanc and A. Saffiotti, "Multirobot object localization: A fuzzy fusion approach," *IEEE Transactions on Systems, Man and Cybernetics-Part B*, vol. 39, no. 5, pp. 1259–1276, October 2009.

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

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

[13] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, "Engineering applications of the self-organizing map," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1358–1384, 1996.

[14] T. Bailey, "Mobile robot localization and mapping in extensive outdoor environments," Ph.D. dissertation, School Aerospace Mech. Mechatronic Eng., University of Sydney, Sydney, NSW, Australia, 2000.

[15] H. Li, F. Karray, O. Basir, and I. Song, "A framework for coordinated control of multi-agent systems and its applications," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 38, no. 3, pp. 534–548, May 2008.

[16] B. Grocholsky, "Information-theoretic control of multiple sensor platforms," Ph.D. dissertation, Australian Centre for Field Robotics, University of Sydney, 2002.

[17] T. Kohonen, "Fast evolutionary learning with batch-type self-organizing maps," *Neural Processing Letters*, vol. 9, pp. 153–162, 1999.

[18] P. Somervuo and T. Kohonen, "Self-organizing maps and learning vector quantization for feature sequences," vol. 10, p. 151159, 1999.

[19] T. Kohonen, *Self-Organizing and Associative Memory*. Springer-Verlag, 1987.

[20] S. Saeedi, L. Paull, M. Trentini, and H. Li, "Multiple robot simultaneous localization and mapping," in *Intelligent Robots and Systems (IROS), Proceedings of the IEEE/RSJ International Conference on*, 2011.

[21] J. Radon, "On the determination of functions from their integral values along certain manifolds," *IEEE Transactions on Medical Imaging*, vol. 5, no. 4, pp. 170–176, August 1986.

[22] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," in *it IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1994, pp. 935–938.

[23] S. Thrun, W. Burgard, and D. FOx, *Probabilistic Robotics*. Cambridge, Massachusets, USA: The MIT press, 2005.

[24] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[25] G. A. F. Seber, *Multivariate Observations*. John Wiley and Sons, Inc., 2004.

[26] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander, and T. Golub, "Interpreting patterns of gene expression with self-organizing maps," in *Proceedings of the National Academy of Science*, vol. 96, 1999, pp. 2907–2912.