Multiple Robot Simultaneous Localization and Mapping

Sajad Saeedi†, Liam Paull†, Michael Trentini⋆ and Howard Li†

***Abstract*— In this research, a decentralized platform for SLAM with multiple robots has been developed. An EKF-based single-robot SLAM is extended to multiple-robot SLAM with a novel occupancy grid map fusion algorithm. Map fusion is achieved through a multi-step process that includes image pre-processing, segmentation, cross correlation, approximating the relative transformation matrix, tuning of the transformation through the Radon image transform and similarity index, and then verification of the result using either map entropy or a verification index. Results are shown from tests performed on a real environment with multiple robotic platforms.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is achieved through the process of exploring an unknown environment and generating a consistent map by fusing sensory information. Typical sensors used include encoders, Inertial Measurement Units (IMU), cameras [1], and laser rangers [2] among others. Using SLAM, the position of the robot can be updated and the environment map can be built [3].

The majority of past literature has focused on SLAM with one sensing platform [4], [5], [6], [7] and [8]. Using multiple robots for SLAM has the advantage that exploration and mapping tasks can be done faster and more accurately. In addition, a distributed system is more robust since the failure of one of the robots does not halt the entire mission [9]. Collaboration based operations such as fire fighting in forest and urban areas, rescue operation in natural disasters, cleaning marine oil spills, underwater and space exploration, security, surveillance and maintenance investigations need to be completed quickly and autonomously and require localization and mapping.

In [10], a feature-based multiple robot SLAM algorithm is proposed which uses an information filter (IF). Another feature-based multiple robot SLAM is proposed in [11] where fuzzy sets are used to represent uncertain position information and fuzzy intersection is used to fuse data. In [9], a solution is presented which is based on occupancy map merging. This method uses map-distance as the similarity index and tries to find similar patterns in two maps based on a simple random walk algorithm. This method has two main drawbacks: first, it is highly time consuming and, secondly, it fails when there are insufficient distinctive patterns in the maps. In [12], a method is proposed using a particle filter (PF) where it is assumed that the robots will meet each other at a point. At the meeting point, the robots know their relative positions and can work backwards to determine their relative initial positions.

In this paper, an improved cascaded multi-sensor data fusion algorithm is used for single robot SLAM. To achieve multiple robot SLAM, a new method of map fusion is introduced. An algorithm is proposed that finds common parts of each robot's map so that they can be fused into a global map using relative transformation matrices. The contributions of this research to multiple robot SLAM include:

- Applying Canny edge detection and introducing a smoothing method to extract unique characteristics of an occupancy grid map,
- Using a segmentation method and applying a correlation technique to find the common parts within two maps,
- Estimating the relative transformation matrix of two maps by approximate and exact pose extractions,
- Applying the Radon transform to tune the orientation angle,
- The use of image entropy to tune the translation vector, as well as verify the final result.

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

II. Multiple Robot SLAM

Multiple robot SLAM has the advantage that it is more efficient and robust to robot failure at the cost of increased development and computational complexity. In multiple robot SLAM, the map provided by each robot in its own reference coordinates is called the local map. Each local map is generated from coordinated laser scans. Each robot tries to integrate all of the local maps provided by the other robots to generate a global map of the environment. However, this is a difficult task because the required alignments or transformation matrices which relate these maps to each other are unknown. The transformation matrix between two robots is determined by their relative poses. As a result, once the relative positions at a certain time are known, the transformation matrix can be generated and used to merge the local maps from that point onwards. Therefore, map fusion can be achieved in two steps: first, the relative transformation matrices must be found, then, the local maps should be compared and fused if there are areas that are found to match.

A. Overview of proposed method

The proposed map fusion algorithm in this paper is based on a search and verification algorithm. Fig. 1 shows the algorithm for two robots, $robot_1$ and $robot_2$, with unknown

†COBRA Group at the University of New Brunswick, Fredericton, Canada, http://www.ece.unb.ca/COBRA/, {sajad.saeedi.g, liam.paull@unb.ca, howard}@unb.ca

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

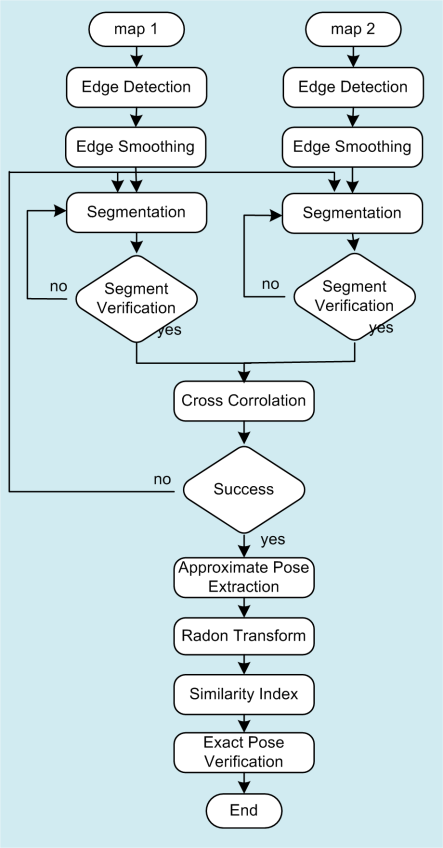

Fig. 1. The proposed map fusion algorithm. Two input maps, map_1 and map_2, are fused by finding their relative transformation matrix. No prior information is available regarding the relative position of the two respective robots, $robot_1$ and $robot_2$

relative positions and each with its own local map, map_1 and map_2 respectively. This algorithm is scalable and can be used for more than two robots. Maps in this algorithm are assumed to be in the form of occupancy grid maps [13].

According to Fig. 1, the flow diagram of the system, the inputs are map_1 and map_2. First, in the **Edge Detection** block, the Canny edges of the maps are extracted. Then, in the **Edge Smoothing** block, the extracted edges are smoothed to facilitate processing in the subsequent blocks. In the **Segmentation** block, a segment of obstacles in the smoothed map is selected. The segment is passed through the **Segment Verification** block to ensure that it contains enough unique geometric information to be used for comparison. The segmentation and verification process continues until an acceptable segment is found or a maximum number of search iterations is reached. The same process is done on map_2. If acceptable segments from both maps are found then the selected segments are tested for similarity in the **Cross Correlation** block. If the selected segments are deemed to be congruent, then they are passed to the **Approximate Pose Extraction** block where a rough estimate of the relative pose of the robots is determined. The orientation of the approximate relative pose is tuned in the **Radon Transform** block, and then the translation elements of the relative pose are tuned in the **Similarity Index** block. In the **Exact Pose Verification** block, the final results are verified. In the following sections, each block is explained in detail

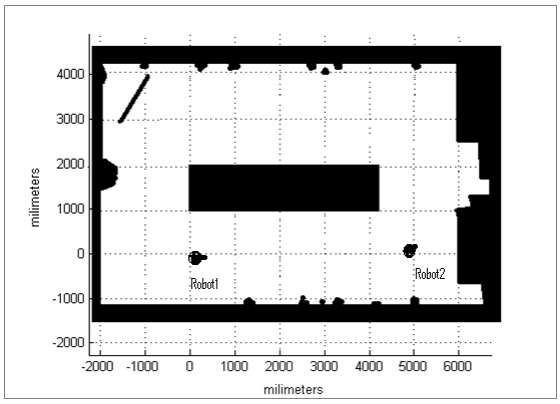

Fig. 2. A simple test environment

and experimental results from the output of each block are presented.

Fig. 2 shows the approximate floor plan and dimensions of an indoor environment used for experimentation. This room has been surveyed by two robots from different initial positions which are assumed to be unknown. Fig. 3-a shows the local map provided by $robot_1$, and Fig. 4-a shows the local map provided by $robot_2$. In both figures, gray cells are unknown, white cells are free, and black cells are obstacles. The size of all maps is 400×400 cells.

B. Edge Detection

The first processing block is **Edge Detection**. Here, noisy measurements are removed from both maps using a smoothing filter and then edges are detected using the Canny edge detection method [14]. This process is necessary because the occupancy grid mapping shows boundaries of obstacles as thicker than they are in reality due to the error in sensor measurements. Using edge detection, it is possible to extract the exact obstacle boundaries to be used in the other blocks so that the map can be processed more accurately and efficiently. Fig. 3-b and Fig. 4-b show the effects of edge detection on the two maps.

C. Edge Smoothing

The next block is **edge smoothing**. Here, the Canny edges are used to estimate the most probable locations of the obstacles. The objective is to remove redundant edges and select the most likely locations of the obstacles.

The error of laser measurements follows a Gaussian distribution. As a result of the stochastic nature of the laser sensor, the occupancy grid mapping will have more multiple cells representing one single point obstacle. The edge detection method can be thought of as a sampling of points from the Gaussian distribution that are equidistant from the mean. It is therefore reasonable to treat the middle point of nearby edges as the most likely location of the actual obstacle, where the edges are truncation lines of the Gaussian distribution. Suppose that we have two adjacent edges $\mathcal{C}_1$ and $\mathcal{C}_2$ that are output from the Canny edge detector. Assuming that the two edges are close enough that they actually only represent one true obstacle, the most probable position of the obstacle, $\mathcal{P}$, is given by

$$\mathcal{P} = \mathcal{C}_1 \oplus \mathcal{C}_2, \tag{1}$$

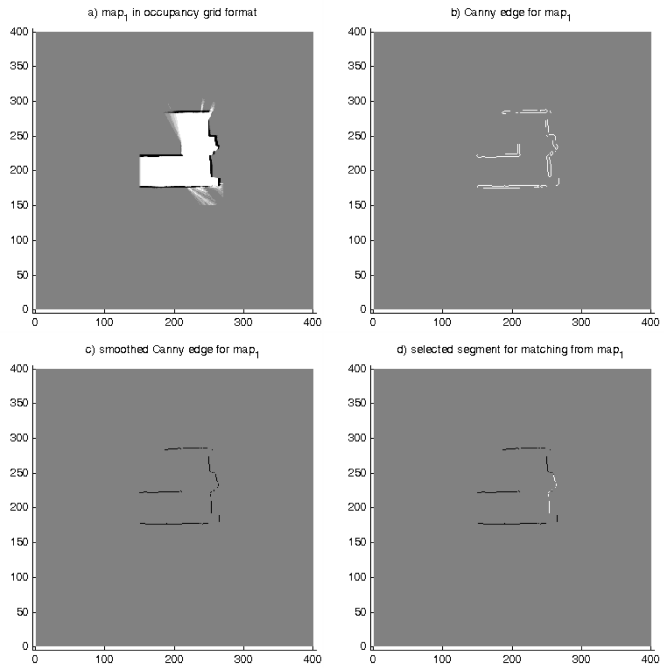

Fig. 3. **a)** map_1, occupancy grid map of the test environment by $robot_1$ **b)** Canny edges **c)** Smoothed edges **d)** Matching segment in white color

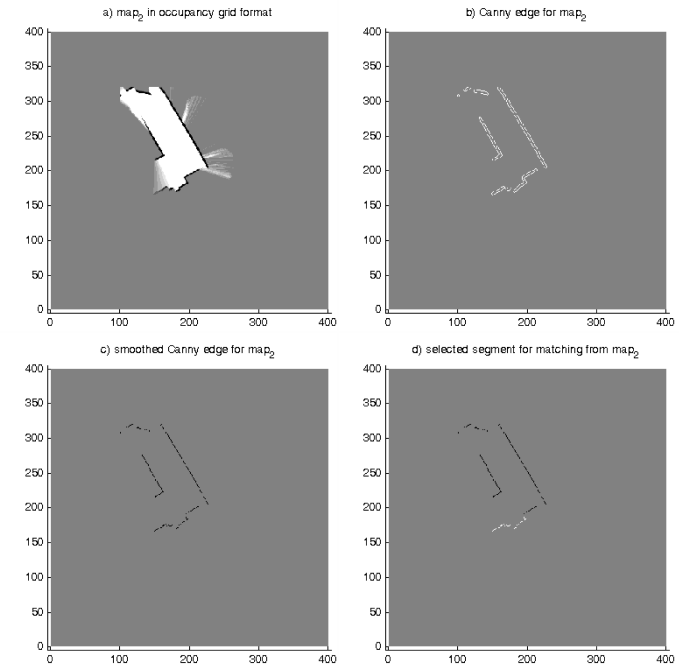

Fig. 4. **a)** map_2, occupancy grid map of the test environment by $robot_2$ **b)** Canny edges **c)** Smoothed edges **d)** Matching segment in white color

where the operator $\oplus$ performs middle curve extraction on two dimensional curves of $\mathcal{C}_1$ and $\mathcal{C}_2$, and outputs the set of points $\mathcal{P}$ that represent the best approximation of the actual location of the obstacle.

It is difficult to generate closed mathematical representations of $\mathcal{C}_1$ and $\mathcal{C}_2$, so the operator $\oplus$ can be defined as operating over the detected edges by scanning in the vertical and horizontal directions. In a vertical scanning, for example, all columns are processed for each row. If the distance between two occupied cells is bigger than one and less than some specified threshold d_t, then the middle point of the two occupied cells is considered to be the most probable occupied cell. If the two occupied cells are adjacent then they are both considered as occupied and as part of the same obstacle. If the occupied cells are separated by a distance greater than d_t, then no averaging is done and those two cells are considered to be distinct obstacles. Fig. 5 shows a simple example of the scanning process. Horizontal scanning is similar to vertical. The major benefit of smoothing edges is that the complexity and scale of the resulting maps are reduced without any significant loss of data. Fig. 3-c and Fig. 4-c show the results at the output of this block for map_1 and map_2.

D. Segmentation

The objective of the **segmentation** block is to try to select unique areas of each map so that they can be compared in the hope of finding parts of the maps that are shared. These shared features are used to find the relative transformation matrix between the two robots. The segmentation block looks for segments of the map which have unique geometric properties, similar to the way the human brain functions [15]. As a human being, our brain tries to find shared parts

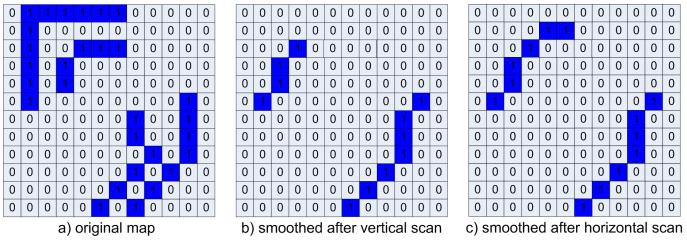

Fig. 5. Vertical and horizontal scans for a simple map: **a)** Original map **b)** Output after vertical scan and **c)** Output after horizontal scan

in different images and then identifies the transformation based on those shared parts. Specifically, parts of the map that contain curves or angles are selected as good candidate segments. It is also noted that segments should be made from the occupied cells. The reason for this that a normal occupancy grid map will have significantly less occupied cells than free or unknown cells.

The segment selection process begins with choosing a random start point in the map, which can be any point that represents an occupied cell. A fixed number of points, n, along the edge are then identified and represented as:

$$P(i) = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, i = 1..n, \tag{2}$$

where P is the segment and x_i and y_i denote the discrete points in the segment.

P is then processed to determine if it contains unique geometrical characteristics by generating and analyzing two other vectors: the Euclidian distance vector, D and the differential angle vector, Δ:

$$\boldsymbol{D} = \left[d_1, d_2, \ldots, d_n\right]^T, \boldsymbol{\Delta} = \left[\delta_1, \delta_2, \ldots, \delta_{n-1}\right]^T, \tag{3}$$

where

$$d_i = ||P(i) - P(0)||,$$

$$\delta_i = \angle(P(i) - P(0)) - \angle(P(i-1) - P(0)). \quad (4)$$

The vector $\boldsymbol{D}$ of the segment is invariant with respect to any rotation or translation of the segment. The vector Δ will differentiate between two segments which are mirror images.

The two characteristic vectors are a unique representation, $\boldsymbol{\Lambda}$, of the original segment P as given by:

$$\boldsymbol{\Lambda} = (\boldsymbol{D}, \boldsymbol{\Delta}). \quad (5)$$

Theorem 1: From $\boldsymbol{\Lambda}$ and the starting location of the segment, $P(0)$, it is possible to reconstruct the segment uniquely.

Proof:

By induction:

1) Base case: $P(0)$ must be known.

2) Induction step: Assume that k points in the segment are known. From d_{k+1}, it is known that the point $P(k+1)$ lies on a circular locus of radius d_{k+1} centered at $P(0)$. Consider that points $P(0)$ and $P(i-1)$ are already located. Let $\phi_k = \angle(P(k) - P(0)$, which is the known angle of point k given that its exact location is already known. Let $\phi_{k+1} = \angle(P(k+1) - P(0)$, which is unknown. From (4), $\delta_{k+1} = \phi_{k+1} - \phi_k$. Rearranging gives $\phi_{k+1} = \delta_{k+1} + \phi_k$. Since both terms on the right hand side are known, the angle ϕ_{k+1} can be calculated. Clearly, from the angle that $P(k+1)$ makes with $P(0)$, the point on the circular locus can be uniquely identified. ■

E. Segment Verification

To identify whether a segment is a good candidate for map fusion, a histogram method is used. Two histograms are constructed to verify a segment. The first one is the distance histogram, $hist_d$ which is generated from the distance vector, $\boldsymbol{D}$, and the second is the arc histogram, $hist_a$, which is generated from the arc vector which is defined as:

$$\boldsymbol{\Omega} = \left[\omega_1, \omega_2, \ldots, \omega_{n-1}\right]^T. \quad (6)$$

Each arc is defined as

$$\omega_i = d_{i+1} \sum_{j=1}^{i} \delta_j, \quad (7)$$

where $i = 1, 2, \ldots, n-1$. The histograms are generated using a prespecified number of bins. The number of chosen bins is important in order to get acceptable result. In this research, it is determined experimentally over a set of trials.

The distance histogram, $hist_d$, and arc histogram, $hist_a$ are represented as:

$$hist_d = \left[hd_1, hd_2, \ldots, hd_{nd}\right]^T, \quad (8)$$

$$hist_a = \left[ha_1, ha_2, \ldots, ha_{na}\right]^T, \quad (9)$$

where nd is the number of the bins and hd_i, $i = 1, 2, \ldots, nd$ is the value corresponding to each bin for $hist_d$, and na is the number of the bins and ha_i, $i = 1, 2, \ldots, na$ is the value corresponding to each bin for $hist_a$.

Fig. 6-a shows a flat segment with its distance and arc histograms in Fig. 6-b and Fig. 6-c, respectively. Fig. 7-a shows a non-flat segment. Its distance and arc histograms are depicted in Fig. 7-b and Fig. 7-c, respectively. If the segment is a wall or flat surface, then the distance histogram will have a flat shape, and the arc histogram will have almost all bins empty except for one. If there is an angle in the segment, then the distance histogram will not be flat and more than one bin of the arc histogram will be non-empty.

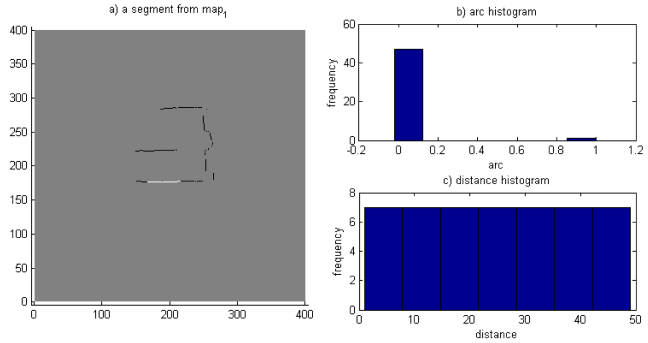

Fig. 6. **a)** A straight wall segment **b)** Arc histogram **c)** Distance histogram

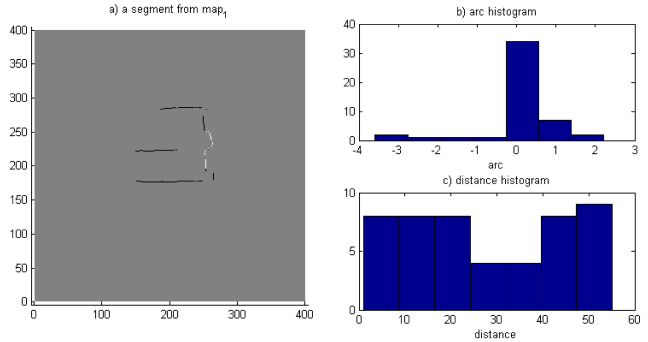

Fig. 7. **a)** A featured wall segment **b)** Arc histogram **c)** Distance histogram

To verify the acceptability of a segment, the following conditions should be met:

1) $hist_d(i) \neq 0, \forall i = 1 \ldots nd$

2) The distance histogram should not be smooth:

$$\max(hist_d) - \min(hist_d) \geq h_d, \quad (10)$$

where h_d is a predefined threshold value.

3) The segment should not be discontinuous. A discontinuity will cause a large value in the arc vector, so the condition:

$$\max(|\omega_i|) \leq \omega_c, \quad (11)$$

where ω_c is a predefined threshold value will reject segments with discontinuities. This novel histogram method is fast and is successful at identifying segments with unique geometric properties.

F. Cross correlation

In the **cross correlation** block, the distance and arc histograms of the two selected segments are compared using a cross correlation technique. The histograms are normalized and then a 2D cross correlation is applied. If the distance and arc histograms are similar, then the result of each cross correlation will be close to one and the segments are deemed to be a match. Otherwise, another segment from map_2 is selected and the process continues. If, for the selected segment from map_1, no correspondence from map_2 can be

found, then another segment from map_1 should be selected. If no shared segment can be found after a certain number of trials, then the two maps cannot be fused. To increase the confidence level of the result of the histogram matching, a correlation of the distance and arc vectors is also applied. Fig. 3-d and Fig. 4-d show two matching segments from map_1 and map_2 in white.

G. Approximate pose extraction

After finding matching segments from both maps, the **approximate pose extraction** block is executed. First, the approximate relative rotation is determined, and then the rotation is used to find the approximate translation. To determine the rotation, a line is fitted for each segment. The difference in the slopes of the two fitted lines gives the approximate rotation angle between the two maps. If the slope of the fitted line for the selected segment of map_1 is a_1 and of map_2 is a_2, then the approximate rotation angle, α_{app} is:

$$\alpha_{app} = \arctan a_1 - \arctan a_2 \tag{12}$$

Fig. 8-a shows both maps on the coordinates of map_1 with identified common segments. Fig. 8-b shows the fitted lines of the two similar segments. The angle between these two lines is an approximation of the relative orientation of the two maps. Once the approximate orientation has been found, the approximate translation can be determined. First, the approximate orientation is applied to align both segments. Next, the approximate translation is computed to be the difference in the centroids of the two segments as given by:

$$T_{app} = \begin{bmatrix} x_{app} \\ y_{app} \end{bmatrix} = \sum_{k=1}^{n_1} \frac{R_{\alpha_{app}} P_{seg1}(k)}{n_1} - \sum_{k=1}^{n_2} \frac{P_{seg2(k)}}{n_2}, \tag{13}$$

where $R_{\alpha_{app}}$ is the rotation matrix based on α_{app}, P_{seg1} and P_{seg2} are the sets of points in segments 1 and 2 respectively. n_1 and n_2 are the cardinalities of P_{seg1} and P_{seg2} respectively. Fig. 8-c shows both maps in the coordinates of map_1 after the approximate transformation. It is clear from the figure that this transformation is not accurate and needs further modification.

H. Pose tuning with the Radon Transform and Similarity Index

After finding the approximate rotation and translations, these elements should be tuned. The exact or tuned pose will be extracted in two steps in the **Radon Transform** and **Similarity Index** blocks as depicted in Fig. 1. First a Radon transform [16] is used to find the tuned rotation. The Radon transform is the projection of the image intensity along a radial line oriented at a specific angle. 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. In a Radon image, the horizontal axis represents all possible angles and the vertical axis is the Radon transform for individual angles. Fig. 9-a, b show the Radon images for map_1 and map_2 respectively after applying the approximate transformation. One of the

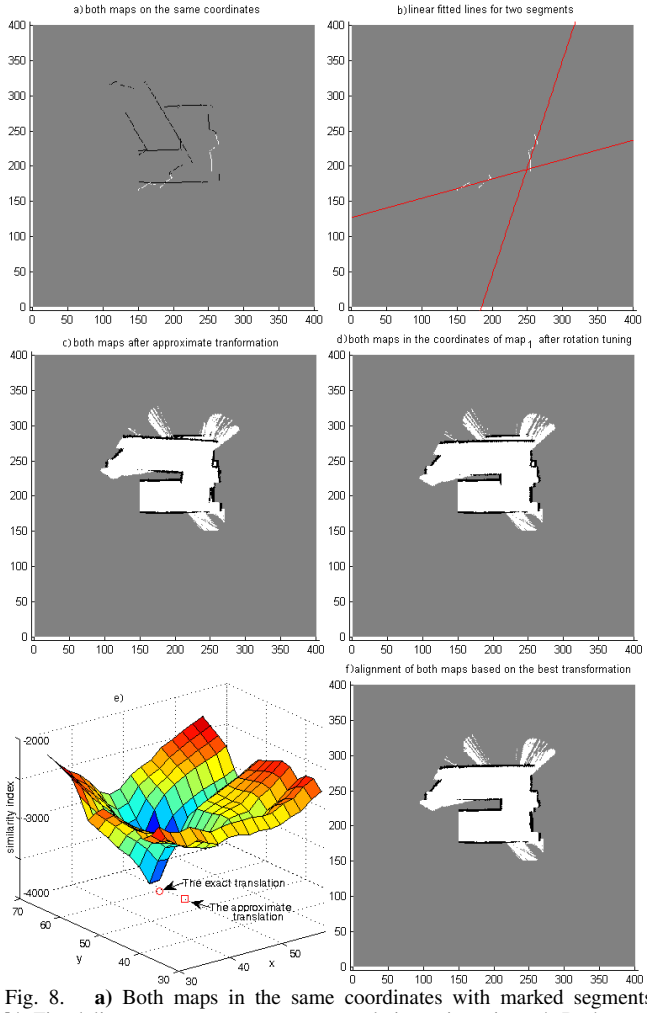

Fig. 8. **a)** Both maps in the same coordinates with marked segments. **b)** Fitted lines on segments to extract relative orientation. **c)** Both maps in the coordinates of map_1 after approximate transformation. **d)** Rotation improvement after Radon transformation. **e)** 3D representation of the similarity index over the search space. The best similarity index occurs at the best translation. **f)** Final alignment of both maps.

characteristics of the Radon image is that the peak points (shown in dark brown color) occur when the Radon angle, θ, aligns with straight line segments that occur in the image. The approximate rotation is used to ensure that the correct peak is selected from the Radon Transform. 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 already computed. As Fig. 9-c, d show, the peak point of the Radon image for map_1 happens to be at 90 degrees while for the approximately transformed map_2 it is 94 degrees. The difference of 4 degrees is the tuning angle that should be added to the original approximate angle as determined by:

$$\alpha_{ext} = \alpha_{app} + \delta_{tuning}, \tag{14}$$

$$\delta_{tuning} = \arg\min_{\theta}(\mathcal{R}_{\theta=0:180}(m_1)) - \arg\min_{\theta}(\mathcal{R}_{\theta=0:180}(T_{x,y,\theta}(m_2))), \tag{15}$$

where α_{ext} is the exact or tuned rotation angle and $\mathcal{R}_{\theta=0:180}(map)$ is the Radon image of the input map over the specified domain of angles (θ). $T_{x,y,\theta}(map)$ means that

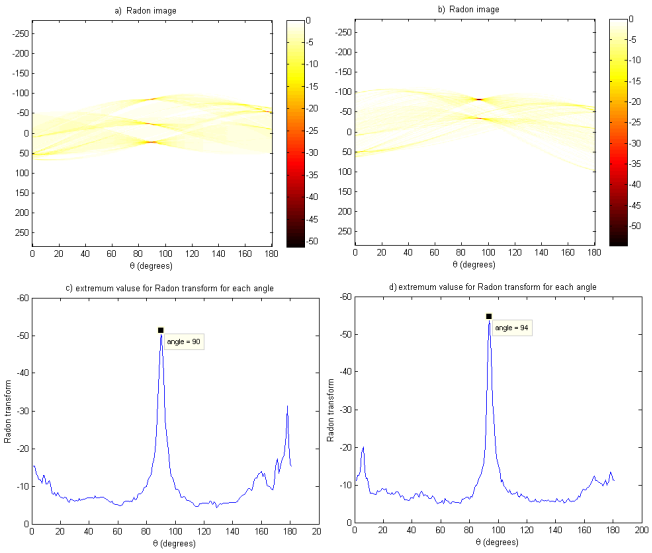

Fig. 9. **a, b)** Radon images of map_1 and map_2 over 180 degrees. **c, d)** Peak points of the Radon images at 90 and 94 degrees

the input map is translated according to the values of x, y and θ and $m_1 = map_1$ and $m_2 = map_2$.

Fig. 8-d shows both maps in the coordinates of map_1 after applying the proposed Radon tuning method. Clearly, this figure shows the effective improvement over the relative rotation angle shown in Fig 8-c.

In the **Similarity Index** block, the tuning of the translation matrix is performed. A performance index is used which has been introduced in [9]. This index is called the similarity index and measures the similarity of two maps over some desired region. When there is more overlap between two maps, there will be an extremum in the index. This index is composed of two components:

$$\begin{aligned} agr(map_1, map_2) &= \sharp\{p = (x, y) | map_1(p) = map_2(p)\}, \\ dis(map_1, map_2) &= \sharp\{p = (x, y) | map_1(p) \neq map_2(p)\}, \end{aligned} \tag{16}$$

where the operator $\sharp$ over a given set returns the cardinality of the set. The similarity index is defined as:

$$J_{similarity} = dis(map_1, map_2) - agr(map_1, map_2). \tag{17}$$

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. A smaller disagreement index and larger agreement index will result in a larger similarity index, $J_{similarity}$. The maximum absolute value represents the best translational match between the two maps. To find the maximal value, an exhaustive search in the neighborhood of the approximate translation is done. It is challenging to use guided search algorithms like Genetic Algorithms (GA) or Particle Swarm Optimization (PSO) because in many cases (not in this example) the best solution may occur very close to very poor solutions. The size of the search space is determined by experience based on the performance of the approximate translation method. In addition, if the total allowable time for searching can be determined, the

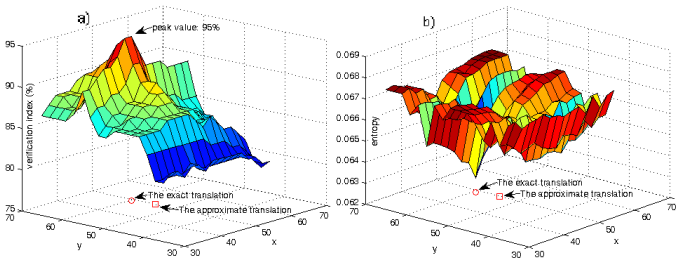

Fig. 10. **a)** 3D representation of the verification index over the search space for the fused maps. The maximum value occurs at the best translation. **b)** 3D representation of the entropy of the fused maps over the search space. The best translation occurs at the minimum entropy.

size of the search space can be specified such that the search is completed within the allotted time. Usually a search space with a radius of ten cells around the current approximate solution is acceptable considering the required time constraints and accuracy of the previous result. The best candidate translation vector is selected using the similarity index:

$$T_{ext} = \begin{bmatrix} x_{ext} \\ y_{ext} \end{bmatrix} = \arg \max_{\mathcal{S}(x,y)} (|J_{similarity}(x, y)|) \tag{18}$$

$$\begin{aligned} \{\mathcal{S}(x, y) \subset R^2 | x_{app} - \delta_x < x < x_{app} + \delta_x, \\ y_{app} - \delta_y < y < y_{app} + \delta_y\} \end{aligned} \tag{19}$$

where T_{ext} is the tuned translation vector and $\mathcal{S}$ is the search space, defined as a rectangle centered at $(x_{app}, y_{app}$ with dimensions $2\delta_x \times 2\delta_y$. Fig. 8-e shows a 3D representation of the similarity index over the entire search space. Fig. 8-f shows the final overlapped maps with exact rotation and translation elements.

I. Pose verification

After finding the best candidate for the transformation matrix, a verification is performed in the **Verification** block. One of two methods can be used. The first method, called similarity verification, uses the ratio of $agr(.)$ and $dis(.)$ as given by:

$$V(m_1, m_2) \mid_{R_{ext}, T_{ext}} = \frac{agr(m_1, m_2) \times 100\%}{agr(m_1, m_2) + dis(m_1, m_2)}, \tag{20}$$

where m_1 and m_2 are two input maps and $V(m_1, m_2)$ is the verification index. If $V(m_1, m_2)$ is close to 100%, then the proposed transformation can be accepted. Fig. 10-a shows the similarity verification. The maximum verification percentage occurs at the exact translation sequence so it is very likely that the translation matrix that has been found is correct.

Another method of verification which is proposed here is to use image entropy. Image entropy is defined as:

$$\mathcal{H} = -\sum p \log_2(p), \tag{21}$$

where p is the normalized histogram of an image. Entropy is a statistical measure of the randomness associated with an image and is usually applied to characterization of image texture. The following relation gives the best translation vector:

$$T_v = \begin{bmatrix} x_v \\ y_v \end{bmatrix} = \arg \min_{\mathcal{S}(x,y)} \{\mathcal{H}(J(map_1, T_{x,y}(map_2))\}$$

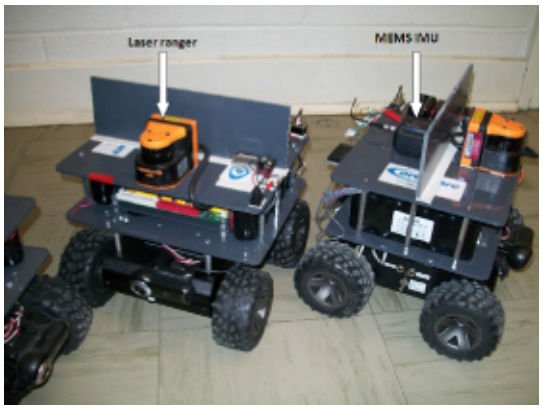

Fig. 11. Robots equipped with the laser ranger and MEMS IMU

where $T_{x,y}(map)$ means that a map is translated according to the values of x and y. $J(m_1, m_2)$ means both maps are in the same coordinate frame according to the coordinates of the first input, m_1. $\mathcal{H}(.)$ is the entropy of the image as defined in equation (21). $\mathcal{S}$ is the same as equation (19) and T_v is translation vector to be verified.

The image entropy can be used as an alternative metric to find the best translation candidate also. The search space for verification and translation extraction should be the same. If the entropy method is used to tune the translation matrix, then the result can verified with the similarity verification index. Fig. 10-b shows the entropy of both maps when they are in the coordinates of map_1. Once again it is noted that the minimum entropy corresponds to the transformation matrix that was found.

III. Experimental Results

In this research, multiple differential-wheeled robots are used. The robots are built by CoroWare, Inc. and each is equipped with three types of sensors: two high speed Phidget Encoders, a UBG-05LN laser ranger from Hokuyo, and a 3DM-GX1 Microstrain IMU, as shown in Fig.11. The laser ranger sensor can measure up to 4500 millimeters with an angle resolution of 0.36 degrees resulting in 513 points for each scan. Data from these sensors are fused with a cascaded EKF. 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 [17]. Specifically, data from the laser ranger and IMU are filtered sequentially to provide a more accurate estimate of the vehicle pose [13].

To demonstrate the effectiveness of the proposed methods, an experiment is done using three robots in an environment with the approximate size of 17×10 meters. Fig. 12 shows the approximate floor plan of the test environment. 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. 13-a,b,c shows the three local maps generated by the three robots. Note that the left part of the test environment was out of the range of the laser range scanners for all three robots. The three maps. map_1, map_2 and map_3, are provided by $robot_1$, $robot_2$ and $robot_3$ respectively. The proposed map fusion algorithm is assumed to be done on $robot_1$ with map_2 and map_3 being integrated into map_1.

Fig. 13-d shows the result of map fusion of map_2 into map_1 using the approximate translation and tuned rotation: $\boldsymbol{T}_{x,y,\theta} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T = \begin{bmatrix} 43 & -13 & 19.8 \end{bmatrix}^T$.

After applying the proposed tuning method for translation, the tuned transformation becomes $\boldsymbol{T}_{x,y,\theta} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T = \begin{bmatrix} 40 & -16 & 19.8 \end{bmatrix}^T$. Fig. 13-e shows the exact alignment for map_2 into map_1. The verification index after final alignment is 97% and the results of similarity index and image entropy are consistent.

The next step is to fuse map_1 with map_3. Similarly Fig. 13-f shows results of map fusion with the approximate translation and tuned rotation. Fig. 13-g shows the result of the tuned alignment. Approximate and exact translations for the fusion of map_1 and map_3 are $\begin{bmatrix} -74 & 23 & 192.8 \end{bmatrix}^T$ and $\begin{bmatrix} -67 & 32 & 192.8 \end{bmatrix}^T$, respectively. After the final alignment the verification index is 98%. The proposed entropy method is verified by the results.

Finally, Fig. 13-h shows all three local maps fused together in the local coordinates of map_1. The three robot team is able to map the environment more quickly than one robot alone. Each robot in the team generates a map of some part of the workspace and then, if there is some overlap between the maps, all local maps can be fused into a global map. It is important to note that the robots are not required to meet at any point to determine their relative positions, this information is generated from the maps.

IV. Conclusion

In this paper, a new method of multi robot SLAM is developed. The proposed method is based on improvements made to single robot SLAM algorithms. Results are verified with tests performed in real environments. Novel aspects of this approach include, preprocessing of occupancy grid maps using Canny Edge detection and smoothing, segmentation to find unique geometrical characteristics, cross correlation to find matching patterns in maps, methods for determining

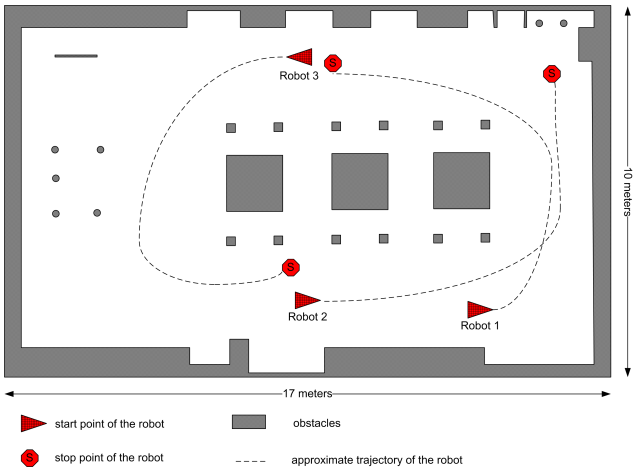

Fig. 12. Floor plan of the real world test environment. Starting and ending locations are marked with triangle and stop sign, respectively. The trajectory of each robot is depicted with dashed lines and obstacles are shown in dark color. All markings are approximate.

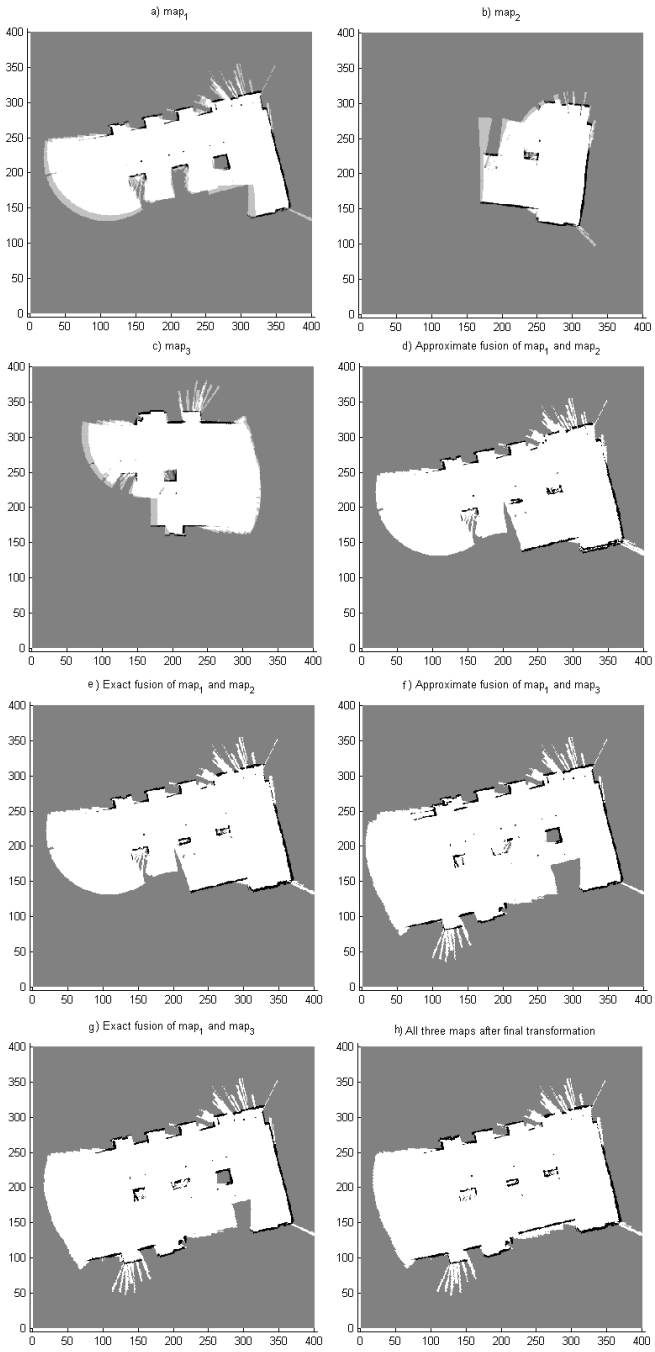

Fig. 13. Map fusion process for three local maps provided by three robots based on the approximate dimensions and trajectories of Fig. 12. **a)** map_1 provided by $robot_1$, **b)** map_2 provided by $robot_2$, **c)** map_3 provided by $robot_3$, **d)** alignment of map_1 and map_2 after approximated translation and tuned rotation, **e)** alignment of map_1 and map_2 after exact transformation, **f)** alignment of map_1 and map_3 after approximated translation and tuned rotation, **g)** alignment of map_1 and map_3 after exact transformation, **h)** final map fusion for all three maps

approximate and exact transformation matrices that relate the maps, and verification. A major advantage of this approach is that there is no need for the robots to meet each other to determine their relative positions. They can do localization and mapping independently without relying on tracking each other.

In the future, improving the approximate pose extraction by using better search algorithms will be investigated. Incorporating tracking information can also improve the results, meaning that if the robots do happen to see each other, they can find their relative poses quickly and easily and use them to give more accurate results.

ACKNOWLEDGEMENT

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

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, July-August 2004.

[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] 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.

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

[11] 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.

[12] 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.

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

[14] M. Wang and C. H. Lai, *A Concise Introduction to Image Processing*. CRC Press, 2008.

[15] J. Hawkins, *On Intelligence*. Times Books, 2004.

[16] 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.

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