IJE TRANSACTIONS A: Basics Vol. 32, No. 4, (April 2019) 608-616

International Journal of Engineering

Journal Homepage: www.ije.ir

Map-merging in Multi-robot Simultaneous Localization and Mapping Process Using Two Heterogeneous Ground Robots

S. Hadian Jazi*, S. Farahani, H. Karimpour

Department of Mechanical Engineering, University of Isfahan, Iran

P A P E R I N F O

Paper history:
Received 16 October 2018
Received in revised form 05 March 2019
Accepted 07 March 2019

Keywords:
Map-merging
Multi-agents Simultaneous Localization and Mapping
Ground Robot
Image Processing

A B S T R A C T

In this article, a fast and reliable map-merging algorithm is proposed to produce a global two dimensional map of an indoor environment in a multi-robot simultaneous localization and mapping (SLAM) process. In SLAM process, to find its way in this environment, a robot should be able to determine its position relative to a map formed from its observations. To solve this complex problem, simultaneous localization and mapping methods are required. In large and complex environments, using a single robot is not reasonable because of the error accumulation and the time required. This can explain the tendency to employ multiple robots in parallel for this task. One of the challenges in the multi-robot SLAM is the map-merging problem. A centralized algorithm for map-merging is introduced in this research based on the features of local maps and without any knowledge about robots initial or relative positions. In order to validate the proposed merging algorithm, a medium scale experiment has been set up consisting of two heterogeneous mobile robots in an indoor environment equipped with laser sensors. The results indicate that the introduced algorithm shows good performance both in accuracy and fast map-merging.

doi: *10.5829/ije.2019.32.04a.20*

1. INTRODUCTION

Rescue missions, security tasks, environmental exploration and many other similar tasks have motivated many researchers to study mobile robots autonomy. The most important issue in mobile robot studies is the navigation question. Localization which is about estimating the position of the robot in an unknown environment, mapping which means creating an accurate map of the environment and path planning, which corresponds to calculate a collision-free path between initial and goal point, are three basic subjects studied in the field of mobile robot navigation. Creating a map of the environment by a robot requires the position of the robot to be known and calculating the position of a robot in an environment requires the map of that environment. This is a complex problem named simultaneous localization and mapping (SLAM). Many researchers focused on SLAM problem and several solutions have been presented.

For decades, single-robot SLAM has been studied, but due to considerable advantages such as increasing chances of saving lives due to coordination in a rescue mission, reduced time of exploration in large unknown environments, efficiency and flexibility, multi-robots SLAM (MRSLAM) have received more attention in recent years. To reproduce a realistic model of the environment in a MRSLAM process, it is necessary for the information collected by different robots to be merged into a single map. This process is referred to as map-merging. Generally, map-merging process can be performed in two steps. The first step is finding a rotational and translational transformation between the maps and the second one is merging the aligned maps into a global or world map. Usually, the transformation between maps can be found based on the poses of the robots or the features of the maps generated by the robots [1]. If the relative positions of the robots are known, the map-merging process will be done directly and easily [2-5]. The relative positions can be calculated if the initial positions of the robots are known, or if the robots meet each other at a point, called rendezvous, or one robot is able to localize the others in its map.

On the other hand, when the robots do not know their

*Corresponding Author Email: *s.hadian@eng.ui.ac.ir* (S. Hadian Jazi)

Please cite this article as: S. Hadian Jazi, S. Farahani, H. Karimpour, Map-merging in Multi-robot Simultaneous Localization and Mapping Process Using Two Heterogeneous Ground Robots, International Journal of Engineering (IJE), IJE TRANSACTIONS A: Basics Vol. 32, No. 4, (April 2019) 608-616

relative positions, the map-merging process must be performed based on the overlaps between the maps. This case is more complex and challenging. There exist several solutions for this problem in the literature. Thrun and Liu [6] presented an algorithm to solve this problem in the case of lack of knowledge about the relative positions of the robots and the landmarks. They used a sparse extended information filter (SEIF) and a tree based algorithm to build a global map in a MRSLAM process. Carpin et al. [7] introduced a similarity measure and a motion planning algorithm to merge the local maps in a fast manner. They showed that their approach for map merging can even merge maps with low quality. Birk and Carpin [8] utilized a similarity measurement function to find the best transformation between local maps. They used the adaptive random walking algorithm to find a maximum overlap between the maps.

Saeedi et al. [9] presented a new method for map fusion using self-organizing map (SOM). In this method based on using a neural network, the complexity of the occupancy grid maps are reduced. The resulting reduction in maps complexities causes a fast and efficient map-merging. Saeedi et al. [10] also found the relative transformation between local grid maps using the Probabilistic Generalized Voronoi Diagram (PGVD).

Dinnissen et al. [11] studied the decision making process to find the right time of map merging to avoid uncorrect matches between local maps, using reinforcement learning. They assumed that the robots meet each other during the MRSLAM process. Li et al. [12] introduced an occupancy-likelihood based objective function and through using genetic algorithm, found the best transformation to merge the local grid maps in a MRSLAM process. The proposed method was implemented on two CyCab robots equipped with a two-dimensional laser sensor, GPS, and rotary shaft encoders mounted on the wheels in an outdoor environment. Park et al. [13] introduced a multi section algorithm to merge the occupancy grid maps generated by individual robots in a MRSLAM process. Their algorithm is based on maximal empty rectangles (MER). The maximal empty rectangles concept collects the free spaces of a map into larger rectangles rather than many pixels and produced a R-map. This reduces the complexity of the map [14].

In this study, first, the algorithm introduced by Park is presented and run for some examples. Then to overcome its inconveniences, a new map-merging algorithm is proposed. This algorithm is a centralized map-merging algorithm and is based on the map features. To show the performance of the proposed approach some experimental tests are performed using two heterogeneous robots in an indoor environment.

2. MAP-MERGING

A basic issue in multi-robot mapping is the merging of local maps prepared by robots. Obviously, to merge maps and produce a global map requires specific map-merging algorithms. Depending on whether the relative positions of the robots are known or not, the map-merging algorithm will be different. If the relation between the coordinate system of the robots is known from the beginning of the motion or in the process where the robots meet, the local maps are built in a common coordinate frame. In this case, map-merging can be easily performed. But if the relative position between the robots is not clear, the main emphasis will be on the features extracted from the raw data provided by the sensors or the maps. In this case, the maps will move and rotate relative to each other to achieve the best compatibility between them.

One of the latest map-merging algorithm has been introduced by Prak [14] named map-merging using R-maps. This algorithm is based on maximal empty rectangles. In this paper, as a first step, merging algorithm introduced by Park is briefly presented and its difficulties were revealed. Then a new map-merging algorithm is introduced and the results were compared.

2. 1. Map-Merging Using R-Maps

Park [14] applied the reduced element map concept, concisely named R-map, to merge local maps as a new method in map merging. Integrating the free space of a map into larger simple elements (with the largest area) to reduce the number of the map elements is the main concept of the R-map. Ahn and Jeon [15] introduced this concept for the first time. The map merging method introduced by Park is presented in algorithm 1.

Algorithm 1 *Map Merging using* $R-map$

procedure $MERGE(map^{(1)}, map^{(2)})$
  $[\boldsymbol{r}^{(1)}; \boldsymbol{c}^{(1)}] = RMAP(map^{(1)})$
  $[\boldsymbol{r}^{(2)}; \boldsymbol{c}^{(2)}] = RMAP(map^{(2)})$
  $\Lambda^{(1)} = ANGLE(\boldsymbol{r}^{(1)}, \boldsymbol{c}^{(1)})$
  $\Lambda^{(2)} = ANGLE(\boldsymbol{r}^{(2)}, \boldsymbol{c}^{(2)})$
  $map_r^{(1)} = rotate\ map^{(1)}\ by\ \Lambda^{(1)}$
  $map_r^{(2)} = rotate\ map^{(2)}\ by\ \Lambda^{(2)}$
  $define\ ratio = 0$
  while $ratio \neq 1$ **do**
    $\boldsymbol{r}_r^{(1)} = RMAP(map_r^{(1)})$
    $\boldsymbol{r}_r^{(2)} = RMAP(map_r^{(2)})$
    $(\Delta^{(1)}, \Delta^{(2)}) = TRIANGLE(\boldsymbol{r}_r^{(1)}, \boldsymbol{r}_r^{(2)})$
    $\Lambda_f = compare\ bisector - vector\ of\ \Delta^{(1)}, \Delta^{(2)}$
    $map_r^{(1)} = rotate\ map_r^{(1)}\ by\ \Lambda_f$
    $define\ ratio = w_{\Delta_1}^{(1)} / w_{\Delta_1}^{(2)}$
    $rescale\ map_r^{(2)}\ by\ ratio$
  end while
  $MAP = overlap\ centers\ of\ \Delta^{(1)}\ and\ \Delta^{(2)}$
end procedure

In this method, first the local grid maps (G-map) provided by different robots are converted to matrices with 0 and 1 entries for occupied and the free area, respectively. Then, using the mentioned matrix, maximal empty rectangles (MERs) of each local map are computed and G-map is converted to R-map. Information on the MERs is recorded in r(i) for the ith map (lines 2 and 3 of the algorithm 1). These information include the width and height of each MER and the position of its upper-left corner. Figure 1 shows the G-map and the R-map form of a sample map.

Also, the MER neighbors which are the MERs connected to it on each of the four directions (left, right, up and down) are found and recorded in c(i). These neighbors are called connections. For example the connection list of the first MER (Figure 1) is c(1) =[3 4].

As most human-made structures have corners with orthogonal angles, maps are represented and investigated in orthogonal frameworks. Maps including R-maps and G-maps are defined along orthogonal axes, too. It is reasonable to align the orthogonal axes occurring within the map parallel to the main orthogonal axes of the map. This leads to a selective orientation transformation between two maps consisting of only four 90 degrees rotations i.e. 0, 90, 180 and 270 degrees. The lines 4 and 5 compute the necessary angle of rotation for each map to align with the orthogonal axes. To perform this alignment process, first, some points on the edges of free spaces are found using the connections of MERs and then by using RANSAC algorithm, the rotation angles are found [14].

The next step in the map merging algorithm is finding common triangles. Common triangles consist of two sets of three rectangles located in two separate R-maps which have the best matches. To find common triangles, two sets of three MERs are selected from each map, namely i, j and k from the first map called $\Delta^{(1)}$ and e, h and g from the second called $\Delta^{(2)}$ and the following cost function is defined

$$e = \left[\frac{A_i}{d_{jk}} \frac{A_j}{d_{jk}} \frac{A_k}{d_{jk}} \alpha_j \beta_k\right]^T - \left[\frac{A_e}{d_{fg}} \frac{A_f}{d_{fg}} \frac{A_g}{d_{fg}} \alpha_f \beta_g\right]^T \quad (1)$$

where A is the area of selected MER and the angles α's and β's and distances d's are defined in Figure 2 which depicts two sets of MERs.

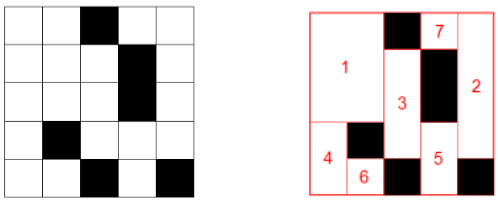

Figure 1. Grid map (left) and corresponding R-map (right) [14]

The most common triangles among others will have a minimum cost function between all sets [14]. This step is done via line 12 of the algorithm. Comparing the bisector-vectors between common triangles, the translational and rotational transformation between two maps can be obtained. Scaling factor, also, can be computed using the common MERs [14].

The algorithm 1 is performed to merge two sample local maps. The maps and the results of different steps of the algorithm are shown in Figure 3.

The time consumed in preparing the final map was about 155 seconds using a conventional PC which seems too long for a map merging process.

In order to evaluate the performance of algorithm 1, a real experiment was performd using two ground robots, Experia and Prawn. The robots are shown in Figures 4 and 5.

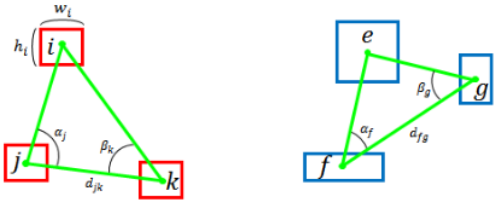

Figure 2. Two sets of three MERs selected from local maps [14]

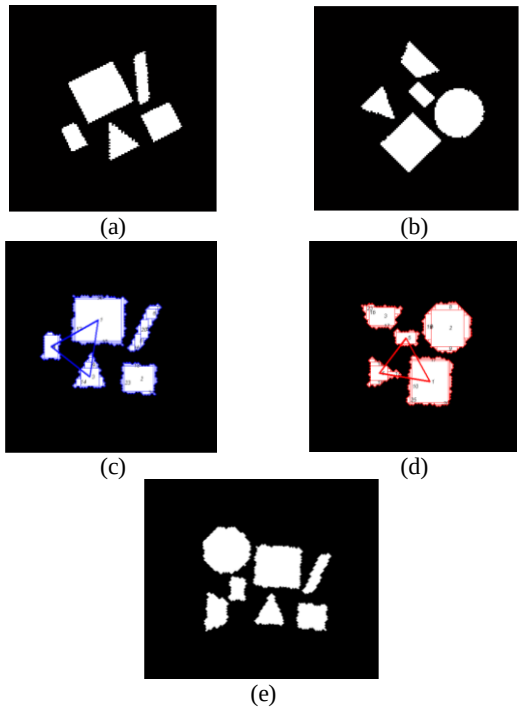

Figure 3. The results of performing the algorithm 1 for two sample maps: (a) and (b) first and second maps, (c) and (d) the maps in orthogonal orientation and common triangles of them, (e) the final merged map

Each robot has a laser sensor scanner, Hokuyo laser sensor for Experia and RpLidar laser sensor for Prawn. Laser scanners specifications are presented in Table 1. In addition, some other sensors like rotary encoders are available for odometry purposes. The whole process is implemented using robot operating system (ROS).

Each robot moves in the environment and provides its own local map. Then the two local maps are merged together and a global map of the environment is reconstructed. The Fast SLAM algorithm was employed for fusing the sensors data on each robot. The environment, which is a corridor with approximate dimensions of $5m \times 50m$, is shown in Figure 6.

Figure 7 shows a 2D map of the environment considered in this test. This map has been sketched using AutoCAD to compare with the final map built during the MRSLAM process.

Figure 4. Experia

Figure 5. Prawn

TABLE 1. Laser scanners specifications

Laser scanner	Angular resolution (deg)	field of view (deg)	measurement range (m)	Scanning time (ms/scan)
Hokuyo	0.36	240	0.06 to 4	100
RpLidar	1	360	0.2 to 6	200

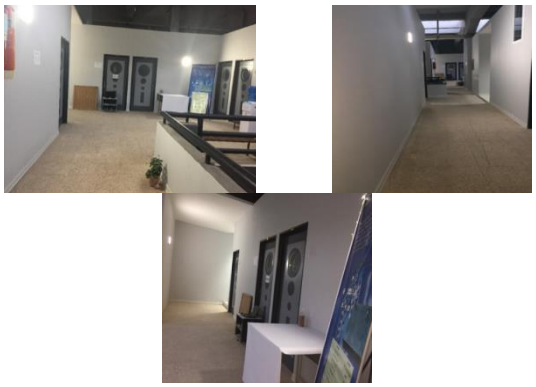

Figure 6. Images from real test environment

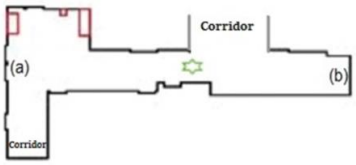

Figure 7. 2D Map of the environment considered for the first practical test

A specific local region is defined for each robot, (a) for Experia and (b) for Prawn. Each robot starts from a point which is not revealed to MRSLAM algorithm and ends its mission at the point marked with (*). When both robots reach this point, the maps produced separately by each robot are transferred to Experia and then merged using the mentioned map-merging algorithm.

Figures 8 and 9 show the maps provided by Experia and Prawn, respectively. The black lines in these figures indicate the walls and some environmental landmarks (here, a table and a water cooler). In the adjacent areas, protrusions were observed because of the fence and staircase. In white areas of the map, the robot has successfully measured and records the odometric and laser sensor data necessary to the mapping algorithm. But concerning gray areas, part of the data are missing due to such factors as topography, slipping wheel occurrence, computational error of the computer processors, etc. Red lines indicate the path of each robot. The robots were conducted manually until they reached a common situ.

As one can see, the two local maps have apparently not many common areas. The merging algorithm is executed for the above maps. The time consumed up to finding common triangles is about 100 seconds. The common triangles found by the algorithm are shown in Figure 10. The final merged map is also shown in Figure 11. As it is seen, the algorithm failed to find the correct common triangles. Consequently, the two maps could not merge correctly. It is obvious from a comparison of Figure 11 and Figure 7.

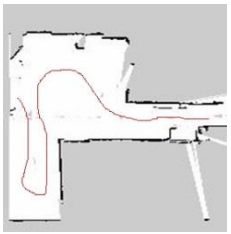

Figure 8. Local map provided by Experia (first practical test)

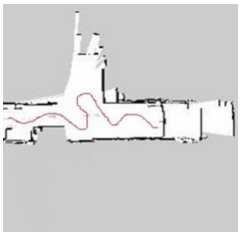

Figure 9. Local map provided by Prawn (first practical test)

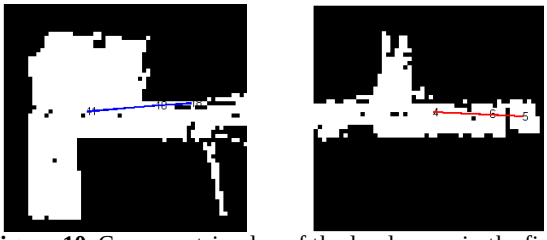

Figure 10. Common triangles of the local maps in the first practical test

Figure 11. Final merged map using algorithm 1 for local maps obtained from the first practical test

2. 1. Proposed Map-Merging Algorithm In this paper, a famous computer vision algorithm called scale-invariant feature transform (SIFT) is used for merging the local maps. the sift algorithm is first introduced by Lowe [16] and is a popular algorithm in the detection and description of image features. SIFT is presented in algorithm 2.

Algorithm 2 *SIFT for detecting image features*

procedure $SIFT(image)$
  $\boldsymbol{blurred\ out\ images} = constract\_scal\_space\ (image)$
  $\boldsymbol{DoGs} = DoG\ (\boldsymbol{blurred\ out\ images})$
  $\boldsymbol{approximate\_extrema} = DoG\_Extrema(\boldsymbol{DoGs})$
  $\boldsymbol{keypoints} = math\_correction(\boldsymbol{approximate\_extrema})$
  $\boldsymbol{final\_keypoints} = get\_rid\_of\_bad\_features(\boldsymbol{keypoints})$
  $\boldsymbol{keypoint\_descriptor} = descriptor(\boldsymbol{final\_keypoints})$
end procedure

The main idea in SIFT consists of finding key points or interesting points which are invariant to scaling, rotation and illumination. For this aim, in the first step, a scale space for the processed image is constructed; It means that the original image is taken then progressively blurred out images are generated from it. Then, the original image is resized to half size, blurred out images are generated again and this process kept repeating itself. Mathematically, a blurred out image is generated by convolving the Gaussian operator to the image pixels:

$$L(x,y;\sigma) = G(x,y,\sigma) * I(x,y) \quad (2)$$

where $I(x,y)$ and L are the image and blurred image, respectively. $G(x,y,\sigma)$ is the Gaussian operator. x, y are local coordinates and σ is the scale parameter. The * denotes the convolution operation. The Gaussian operator is defined as follows:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (3)$$

Figure 12 shows the scale space of a ball. The next step in SIFT algorithm is to calculate the Difference of Gaussians (DoGs). This is necessary to find out key points. The DoG is calculated from the difference of two consecutive scales as follows [16]:

$$D(x,y;\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma) \quad (4)$$

Figure 13 shows a graphical representation for calculating DoGs.

The third step is to find the key points of the image. To find key points, in first instance, the approximate local maxima/minima must be located in DoG images. To find these approximate local extrema, each pixel must be compared with its 26 neighbors (Figure 14).

A pixel is marked as a "key point" if it is the greatest or least of all its 26 neighbors. Usually, the extrema is laid between pixels and don't match exactly on a pixel. In this case, the extremas are only "approximate" and a subpixel extrema search has to be performed mathematically.

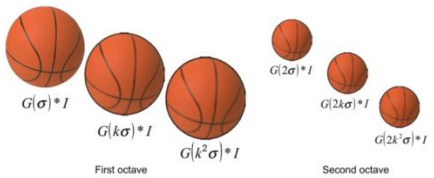

Figure 12. Scale space of a ball with first and second octaves

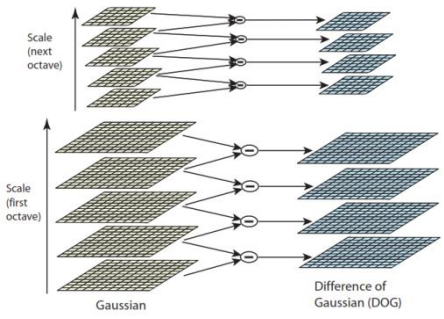

Figure 13. Calculating the difference of Gaussians using scale space; Right-scale space, Left-Difference of Gaussians [16]

Using Taylor expansion of $D(x, y; \sigma)$, the subpixel extrema is calculated as follows [16]:

$$\hat{\boldsymbol{x}} = -\left(\frac{\partial^2 D}{\partial \boldsymbol{x}^2}\right)^{-1} \frac{\partial D}{\partial \boldsymbol{x}} \quad (5)$$

where $\boldsymbol{x} = [x, y; \sigma]^T$ and the derivatives of D are calculated at the last approximate extrema. Using the subpixel extrema as key points instead of approximate extrema will increase the odds of matching and the stability of the SIFT algorithm.

A numerous quantity of key points may be obtained, some without enough contrast and some others lying along an edge. These key points are not useful in feature detections and must be omitted. To remove edge key points, Harris corner detector [17] can be used and for low contrast features, checking the key point intensity is a simple and useful way.

At this step, a set of key points is exploited with their scales. The scale of each key point is identical to the scale of its corresponding blurred image, making them scale invariant. The fourth step of SIFT algorithm is to assign an orientation measure to each key point to provide orientation invariance.

To assign an orientation to a key point, the gradient magnitude and orientation of all pixels around the key point are calculated as follows:

$$m(x, y) = \sqrt{\left(L(x+1, y) - L(x-1, y)\right)^2 + \left(L(x, y+1) - L(x, y-1)\right)^2}$$
$$\theta(x, y) = \tan^{-1}\left(\left(L(x, y+1) - L(x, y-1)\right) / \left(L(x+1, y) - L(x-1, y)\right)\right) \quad (6)$$

Then a histogram with 36 bins (each 10 degrees) is created using these magnitudes and orientations. The amount of histogram is proportional to the magnitude. The created histogram will have a peak at some points. The corresponding bin indicates the orientation assignable to the key point. Now, the objective of orientation invariance is also reached.

The final step in SIFT algorithm is to create a descriptor for each detected feature or key point. For this purpose, a 16*16 grid, partitioned into 4*4 windows, is considered around each key point (Figure 15). The gradient magnitudes and orientations are calculated within each 4*4 window.

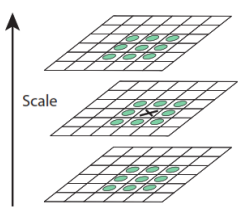

Figure 14. The neighbors of a pixel in DoGs [16]

Based on the orientation-measure results, an 8-bin histogram is created. For the sake of accuracy, the magnitude in each bin can also be considered depending on the distance from the key point. So a more remote pixel from the key point will have a lesser contribution to the histogram. Applying this for the sixteen 4*4 windows, an array of 4*4*8 number is created for each key point. Normalizing this array, the feature vector or the feature descriptor is created. For more details such as practical details, corrections and theoretical details, please refer to literature [16].

Now, everything is ready to merge the two maps. Algorithm 3 shows the map merging steps using SIFT.

Algorithm 3 *Map Merging using SIFT*

1: **procedure** $Merge(map^{(1)}, map^{(2)})$
2: $\boldsymbol{Image}^{(1)} = convert\_to\_image\ (map^{(1)})$
3: $\boldsymbol{Image}^{(2)} = convert\_to\_image\ (map^{(2)})$
4: $[\boldsymbol{keypoints}^{(1)}, \boldsymbol{descriptors}^{(1)}] = SIFT(\boldsymbol{Image}^{(1)})$
5: $[\boldsymbol{keypoints}^{(2)}, \boldsymbol{descriptors}^{(2)}] = SIFT(\boldsymbol{Image}^{(2)})$
6: $\boldsymbol{matched\_features} = match(\boldsymbol{descriptors}^{(1)}, \boldsymbol{descriptors}^{(2)})$
7: $\boldsymbol{Homography\ matrix} = RANSAC(\boldsymbol{matched\_features})$
8: $\boldsymbol{Final\ Map} = tranform(map^{(1)}, map^{(2)}, \boldsymbol{Homography\ matrix})$
9: **end procedure**

Merging algorithm, described here, consists of five steps. First, the robot-produced local maps are stored as images. In the next step, the features of the images are detected and their descriptors are computed. In the third step, the descriptors are matched with each other between two images. Feature matching is actually a straightforward process. The first feature from the first image is selected and the distances of its descriptor to all feature descriptors of the second image are computed. A matched feature in the second image has the smallest "distance" to the selected feature. This process is repeated for all features of the first image. David Lowe's ratio test can optionally be used for features matching [16]. Generally, only three pairs of matched features are needed to compute a transformation or homography matrix between two images.

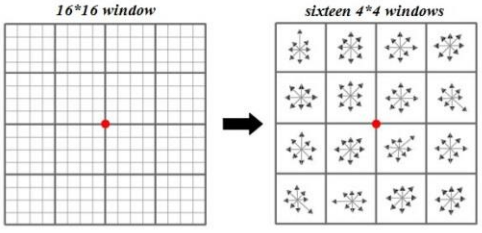

Figure 15. Selection of pixels around a key point for creating its descriptor

With more than three pairs, RANSAC algorithm [18] can be used for computing the homography matrix. Homography matrix computation is the fourth step of the merging algorithm. In the fifth step, the transformation calculated from the homography matrix is applied to images or maps and the two maps are then merged accordingly.

The proposed algorithm is executed for merging the maps obtained in the experiment phase (Figure 8 and Figure 9). The common key points of these two maps are shown in Figure 16. As one can see, the proposed algorithm could find the correct common features between two maps. After forming the homography matrix, the two maps are merged together. The result is shown in Figure 17. Comparing Figure 17 and Figure 7 shows that the local maps have been merged correctly. The time expended to merge the maps is about 20 seconds which is about 20% of the time consumed using R-map merging algorithm.

A second experiment is performed to map a more complex environment. This experiment is performed within a 50 m^2 location consisting of two compartments. The complete environment is shown in Figure 18. The local maps provided by Experia and Prawn are shown in Figures 19 and 20, respectively. Similar to the first practical test, the robots were conducted to explore their respective neighborhood manually and terminated their mission by checking a common landmark which was introduced earlier to them.

The matched points and the final merged map are also shown in Figures 21 and 22. As can be seen, the matched points have been found correctly and the merging process is well done.

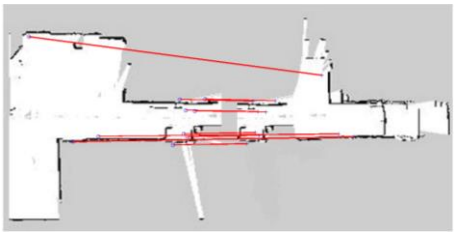

Figure 16. Key points matching using the proposed merging algorithm (first test)

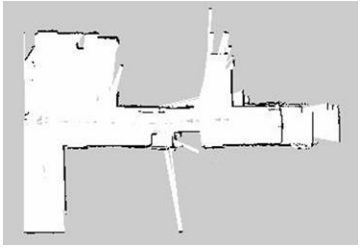

Figure 17. The final merged map obtained by using the proposed merging algorithm (first test)

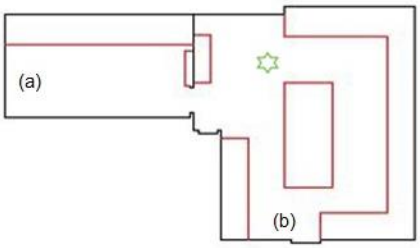

Figure 18. 2D Map of the environment considered for the second practical test

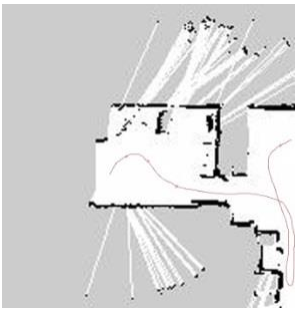

Figure 19. Local map provided by Experia (second practical test)

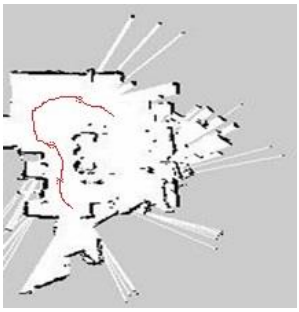

Figure 20. Local map provided by Prawn (second practical test)

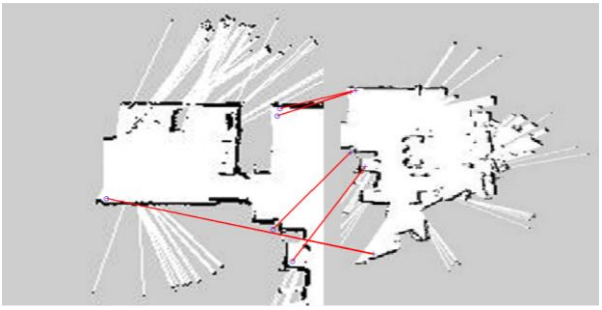

Figure 21. Key points matching using the proposed merging algorithm (second test)

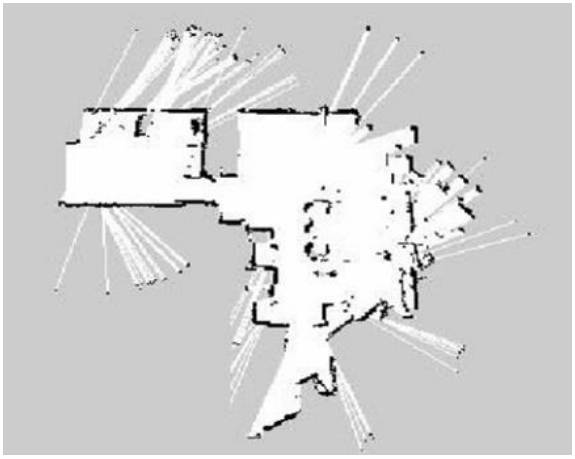

Figure 22. The final merged map using the proposed merging algorithm (second test)

3. CONCLUSION

In this paper, the map-merging algorithm using the maximal empty rectangles (MER) concept is presented and implemented. It is shown here that this strategy is not only time-consuming, but also failed in the case of low overlap area.

To overcome these two inconveniences, an image feature extraction algorithm named SIFT is employed and a fast and reliable map-merging algorithm is proposed. Using SIFT, the local maps provided by ground robots are converted into images, and the features of each image together with their descriptors get extracted. These features are made invariant to orientation and scaling. Then, the matching features between the two images are identified and a homography matrix is finally computed using RANSAC.

Experimental results show the proposed map-merging algorithm can merge maps with the least common areas. Also, the proposed algorithm is meaningfully faster than the map-merging algorithm applying R-maps.

4. ACKNOWLEDGEMENT

We would also like to express our gratitude to Advanced Mechatronics and Robotics Laboratory (ARMLAB), Mechanical Engineering Department, Isfahan University of Technology for their generous contribution in preparing the facilities and the environment required.

5. REFERENCES

1. Rone, W. and Ben-Tzvi, P., "Mapping, localization and motion planning in mobile multi-robotic systems," ***Robotica***, Vol. 31, No. 1, (2013), 1–23.
2. Fenwick, J. W., Newman, P. M., and Leonard, J. J., "Cooperative concurrent mapping and localization", in *2002 IEEE International Conference on Robotics and Automation*, (2002), 1810–1817.
3. Konolige, K., Fox, D., Ortiz, C., Agno, A., Eriksen, M., Limketkai, B., Ko, J., Morisset, B., Schulz, D., Stewart, B., and Vincent, R., "Centibots: Very large scale distributed robotic teams," ***Springer Tracts in Advanced Robotics***, Vol. 21, No. 1, (2006), 131–140.
4. Thrun, S., "A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots," ***The International Journal of Robotics Research***, Vol. 20, No. 5, (2001), 335–363.
5. Williams, S. B., Dissanayake, G., and Durrant-Whyte, H., "Towards multi-vehicle simultaneous localisation and mapping", in *Proceedings 2002 IEEE International Conference on Robotics and Automation*, Vol. 3, 2743–2748, (2002).
6. Thrun, S. and Liu, Y., "Multi-robot SLAM with sparse extended information filers," ***Robotics Research***, Vol. 15, No. 1, (2005), 254–266.
7. Carpin, S., Birk, A., and Jucikas, V., "On Map Merging," ***International Journal of Robotics and Autonomous Systems***, Vol. 53, No. 1, (2005), 1–14.
8. Birk, A. and Carpin, S., "Merging Occupancy Grid Maps From Multiple Robots", in *Proceedings of the IEEE*, Vol. 94, No. 7, 1384–1397, (2006).
9. Saeedi, S., Paull, L., Trentini, M., and Li, H., "Neural Network-Based Multiple Robot Simultaneous Localization and Mapping," ***IEEE Transactions on Neural Networks***, Vol. 22, No. 12, (2011), 2376–2387.
10. Saeedi, S., Paull, L., Trentini, M., Seto, M., and Li, H., "Efficient map merging using a probabilistic generalized Voronoi diagram", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4419–4424, (2012).
11. Dinnissen, P., Givigi, S. N., and Schwartz, H. M., "Map merging of Multi-Robot SLAM using Reinforcement Learning", in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 53–60, (2012).
12. Li, H., Tsukada, M., Nashashibi, F., and Parent, M., "Multivehicle cooperative local mapping: A methodology based on occupancy grid map merging," ***IEEE Transactions on Intelligent Transportation Systems***, Vol. 15, No. 5, (2014), 2089–2100.
13. Park, J., Sinclair, A. J., Sherrill, R. E., Doucette, E. A., and Curtis, J. W., "Map merging of rotated, corrupted, and different scale maps using rectangular features", in *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 535–543, (2016).
14. Park, J., "A Reduced Element Map Representation and Applications: Map Merging, Path Planning, and Target Interception". PhD Thesis: Aerospace Engineering, Auburn University, (2017).
15. Ahn, J. G. and Jeon, H. S., "R-Map : A Hybrid Map Created by Maximal Rectangles", in *ICCAS 2010*, 1336–1339, (2010).
16. Lowe, D. G., "Distinctive image features from scale-invariant keypoints," ***International journal of computer vision***, Vol. 60, No. 2, (2004), 91–110.
17. Harris, C. and Stephens, M., "A combined corner and edge detector.", in *Proceedings of Fourth Alvey vision conference*, Vol. 15, No. 50, 147–151, (1988).
18. Fischler, M. A. and Bolles, R. C., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," ***Communications of the ACM***, Vol. 24, No. 6, (1981), 381–395.

Map-merging in Multi-robot Simultaneous Localization and Mapping Process Using Two Heterogeneous Ground Robots

S. Hadian Jazi, S. Farahani, H. Karimpour

Department of Mechanical Engineering, University of Isfahan, Iran

P A P E R I N F O

Paper history:
Received 16 October 2018
Received in revised form 05 March 2019
Accepted 07 March 2019

Keywords:
Map-merging
Multi-agents Simultaneous Localization and Mapping
Ground Robot
Image Processing

چکیده

در این مقاله یک آلگوریتم ترکیب نقشه سریع و قابل اعتماد برای تولید نقشه کلی از یک محیط داخلی و در یک پروسه نقشه‌برداری و تعیین موقعیت همزمان چند رباتی ارائه شده‌است. یک ربات برای حرکت در یک محیط و پیدا کردن مسیر حرکت خود به نقشه‌ای از محیط نیاز دارد و همچنین برای تهیه نقشه از محیط به موقعیت خود در آن محیط وابسته است. این مساله پیچیده را نقشه‌برداری و تعیین موقعیت همزمان می گویند. در محیط های بزرگ و پیچیده، به دلایل متفاوتی از جمله انباشتگی خطا و طولانی‌شدن پروسه، استفاده از یک ربات برای نقشه‌برداری توجیه‌پذیر نیست. در این موارد معمولا از چند ربات برای این کار استفاده می‌شود. یکی از مهمترین چالشها در پروسه‌های چندرباتی نقشه‌برداری، ترکیب‌کردن نقشه‌های محلی تهیه‌شده توسط هر ربات از محیط و یا بخشی از آن و سپس تولید نقشه کلی آن محیط است. در این مقاله یک آلگوریتم مرکزی برای ترکیب نقشه بر پایه ویژگی‌ها و مشخصه‌های نقشه‌های محلی ارائه می‌شود که در آن به اطلاعاتی از قبیل موقعیت اولیه یا موقعیت نسبی ربات‌ها نیازی نیست. به‌منظور ارزشیابی این آلگوریتم به‌صورت تجربی از دو ربات متفاوت که به سنسورهای لیزری متفاوتی برای پویش محیط مجهز هستند می‌شود. نتایج آزمایش نشان می‌دهد که آلگوریتم معرفی‌شده دقت و سرعت مناسبی برای ترکیب‌کردن نقشه‌های محلی و تولید نقشه کلی از محیط دارد.

***doi**: 10.5829/ije.2019.32.04a.20*