Authors' version - Published in: Springer Tracts in Advanced Robotics,
vol. 83, pages 147-160, 2013

Multi-Robot Map Updating in Dynamic Environments

Fabrizio Abrate, Basilio Bona, Marina Indri, Stefano Rosa, and Federico Tibaldi *

Abstract Multi-robot systems play an important role in many robotic applications. A prerequisite for a team of robots is the capability of building and maintaining updated maps of the environment. The simultaneous estimation of the trajectory and the map of the environment (known as SLAM) requires many computational resources. Moreover, SLAM is generally performed in environments that do not vary over time (called *static* environments), whereas real applications commonly require navigation services in *dynamic* environments. This paper focuses on long-term mapping operativity in presence of variations in the map, as in the case of robotic applications in logistic spaces, where rovers have to track the presence of goods in given areas. In this context classical SLAM approaches are generally not directly applicable, since they usually apply in static environments or in dynamic environments where it is possible to model the environment dynamics. This paper proposes a methodology that allows the robots to detect variations in the environment, generate maps containing only the persistent variations, propagate thiem to the team and finally merge the received information in a consistent way. The team of robots is also exploited to assure the coverage of areas not visited for long time, thus improving the knowledge on the present status of the map. The map updating process is demonstrated to be computationally light, in order to be performed in parallel with other tasks (*e.g.*, team coordination and planning, surveillance).

F. Abrate
Istituto Superiore Mario Boella, via P.C. Boggio 61 10129 Torino, Italy
e-mail: fabrizio.abrate@ismb.it

B. Bona, M. Indri, S. Rosa, and F. Tibaldi
Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
e-mail: {basilio.bona,marina.indri,stefano.rosa,federico.tibaldi}@polito.it

* This work was supported by Regione Piemonte under the "MACP4Log" grant (RU/02/26), the "Piattaforma Tecnologica for the Internet of Things Project" and by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) under MEMONET National Research Project.

1 Introduction

Mobile robot systems have been involved in many successful applications including museum guide robots, surveillance, planetary exploration, search and rescue [13]. To successfully accomplish these tasks, the robots shall be able to build maps of unknown environments and to localize therein. The joint estimation of both the position and the map model is referred to as Simultaneous Localization And Mapping (SLAM). While the maturity of SLAM in single robot scenarios is recognized in many recent works, many issues arise when trying to extend these approaches to multi-robot scenarios. One of the first multi-robot approaches is given in [8], where a cooperative SLAM algorithm is proposed to merge sensor and navigation information from multiple autonomous vehicles, on the basis of stochastic estimation and feature-based landmark extraction from the environment. In [16] the Constrained Local Submap Filter (CLSF) is exploited to create a local submap of the features in the immediate vicinity of the vehicle, periodically fused into the global map of the environment. This representation reduces the computational complexity of maintaining the global map estimates as well as it improves the data association process. Some approaches, as [5] and [9], are based on Rao-Blackwellized particle filters (RBPF), while others [17] are based on Kalman filtering. The approach proposed in [10] is based on manifold representation of maps. This approach has been mainly designed to overcome limitations of existing SLAM methods, especially the sensitivity to false data associations. Other approaches like [4] speed up mapping by using multiple robots exploring different parts of the environment. In general, the problems in multi-robot systems are still related to the need for team coordination strategies and to the high computational and memory requirements depending on the number of robots and the map size. Moreover service robotic applications have to cope with intrinsically dynamic environments. Realistic applications require updated maps of the environments that vary over time, starting from a given initial condition. This is for instance the case of robotic applications in logistic spaces, where robots have to track the presence of goods in certain areas. The goods are stored in appropriate places, but during the day they can be removed and substituted by other items many times. In these scenarios classical SLAM approaches are not suitable, as it could be at least difficult or even impossible to model the dynamics of the environment. Furthermore when dealing with very large environments the memory requirements for multi-robot SLAM could become too high. The problem of keeping an updated map of the environment in order to preserve the robots localization, without investigating any specific goods tracking procedure, has been faced in [3] for the single-robot case.

In this paper this solution is extended to a multi-robot scenario. The concept of *time-map* is introduced to assign to each cell in the map a value representing its reliability. This time-map is used to merge in an appropriate way the changes detected locally by a robot and the updated maps received from the other team members. The effectiveness of the approach is improved by a simple team coordination strategy, which we propose to actively search for modifications in the map. Finally experimental

results of simulated and real tests are carried out to evaluate the effectiveness of the algorithm and its computational load.

2 Problem Formulation

A team of mobile robots, each endowed with a laser rangefinder and wireless connectivity, is supposed to be correctly localized with respect to the available environment map. Each robot is assumed to be in the *position tracking* state, as defined in [1] and [2].
Each robot uses an occupancy grid map of the environment in the localization algorithm to track its position over time. Such a map could have been manually created or previously built by a SLAM algorithm.
At discrete time instants k the environment changes, and the robots have to modify their map to take into account the variation. This phase is called *Δ-mapping* step.
The set of new maps collected up to time k is defined as

$$\mathscr{M}(k) = \{M_k\},\ k = 0, \ldots, K.$$

M_0 is the initial map, obtained by the SLAM procedure. The goal of the developed algorithm is to provide for each robot an estimate $\hat{M}_k$ of the map at each time step k. In order to take advantage of the multi-robot scenario these updated maps must be shared with the other robots, and this information has to be merged in order to create a map that is a good estimate of the current state of the environment.

Correct map merging is not sufficient; a coordination strategy of the team of robots it also needed to maximize the number of detected variations, balancing at the same the number of *Δ-mapping* processes among the robots.

3 The Approach

The guidelines of the proposed approach are described hereafter, whereas details about the specific processes of variations awareness, local *Δ-mapping* and map merging are given in Subsection 3.1.

In the proposed *Δ-mapping* approach the concept of *time-map* is introduced to merge properly the changes detected locally by a robot and the updated maps received from the other team members.
In a grid map each cell represents the belief on the occupation value of the corresponding area. Since the environment changes over time, the reliability of the stored value for the cells decreases over time. Therefore to each cell in the map a value in the range $[0-1]$ is assigned, related to the time passed since the cell has been visited for the last time. The set of these values at each time step is called *time-map* and defined as T_t.

The outline of the *Δ-mapping* algorithm, which runs on board of each rover, is described in Algorithm 1.

The algorithm takes as inputs the previous map $\hat{M}_{k-1}$ and the time-map T_{t-1}. p and l are the current robot pose and the current laser range reading respectively, P and L are two matrices collecting the values of p and l

Input: $\hat{M}_{k-1}$, T_{t-1}, p, l, P, L
Output: $\hat{M}_k$, T_t
1 T_t = **updateTimeMap**(T_{t-1}, p, l);
2 **if** *received map* $\hat{M}'$ T' **then**
3 $[\hat{M}'_{k-1}, T_t]$ = **mergeMap**($\hat{M}_{k-1}$, T_t, $\hat{M}'$, T');
4 $\hat{M}_{k-1} = \hat{M}'_{k-1}$;
5 **end**
6 **if** $\Delta - awareness$ **then**
7 $P = P + p$;
8 $L = L + l$;
9 **else**
10 **if** $P! = \emptyset$ **then**
11 $[\hat{M}_k]$ = **updateMap**($\hat{M}_{k-1}$, P, L);
12 $P = \emptyset$;
13 $L = \emptyset$;
14 **dispatchUpdatedMap**($\hat{M}_k$, T_t);
15 **end**
16 **end**

Algorithm 1: the Δ-mapping algorithm

$$P = \begin{bmatrix} \hat{x}^1, \hat{y}^1, \hat{\theta}^1 \\ \vdots \\ \hat{x}^n, \hat{y}^n, \hat{\theta}^n \end{bmatrix} \tag{1}$$

$$L = \begin{bmatrix} l^1 \\ \vdots \\ l^n \end{bmatrix} \tag{2}$$

where the n-th entry is the last element stored. These matrices are used to create a local Δ-map containing the changes in the environment detected by the robot.

The time-map T_t is updated every time a laser scan is available to the robot; a ray tracing procedure is applied for each angle of the scan, assigning a maximum value equal to 1 to every cell crossed by a ray. At each time step all the values in T_t are updated according to

$$T_t(i,j) = T_{t-1}(i,j) \cdot \left(1 - \frac{\Delta t}{C_t}\right) \tag{3}$$

where Δt is the time elapsed from the last update of T_t, and C_t is a time constant which defines the forgetting speed.
The time-map update depends only on Δt, therefore a common timebase among the team members is not required, avoiding the need of synchronization techniques over the net.

The algorithm is divided in two parts. The first part (lines 2-4) is performed only when the robot receives a map from another robot member of the team, while the

second part (lines 6-15) is performed only if a variation in the environment has been detected.

If a robot receives a new map $\hat{M}'$ and the relative time-map T', it updates the state of its map and its time-map by merging them with $\hat{M}_{k-1}$ and T_t respectively (line 3). At this point the resulting map contains the modifications perceived by the other robots (line 4).

If a modification is detected by the *Δ-awareness* block, recalled in Section 3.1, the algorithm stores the current robot pose and the relative laser range reading (lines 7-8).

If the *Δ-awareness* block does not detect any modification and P and L are not empty, a local *Δ-mapping* is performed, following the approach recalled in subsection 3.1 (line 10). The content of these vectors is used to create a local Δ-map $\Delta\hat{M}$, then $\Delta\hat{M}$ is aligned and merged with the old map $\hat{M}_{k-1}$, to obtain an updated map $\hat{M}_k$.

Finally the resulting map $\hat{M}_k$ and the current time-map T_t are dispatched to the other team members.

3.1 Δ-awareness, local Δ-mapping and map merging

In [3] the authors presented a single-robot approach that maintains an updated grid map of a dynamic environment, assuming an initial occupancy grid map available. The algorithm detects persistent variations in the environment and merges them with the previous map by using limited computational resources. It is composed by four blocks as shown in Figure 1.

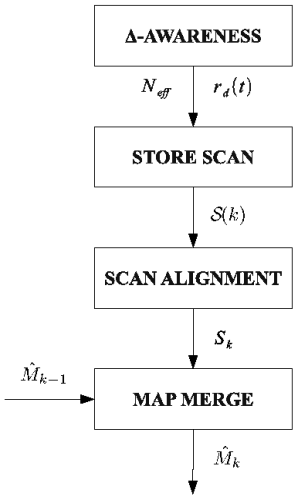


Fig. 1 The local Δ-mapping architecture.

The *Δ-awareness* block detects persistent variations in the environment, using a technique called *weighted recency averaging*, normally applied in tracking non-

stationary processes.
In this setting, the weighted recency averaging recognizes changes in the environment, under the hypothesis that the robot is correctly localized and never kidnapped.

The purpose of the *Store Scan* block is to select the laser scan readings suitable for building the local updated sub maps. These readings are stored in L with the corresponding robot poses stored in P.

The *Scan Alignment* block produces a Δ-map performing a consistent registration of the collection of scan readings contained in L. The approach maintains all the local frames of data as well as the relative spatial relationships between local frames, modeled as random variables and derived from matching pairwise scans or from rover poses stored in P.

The *Map Merge* block merges the output of the *Scan Alignment* block with the map $\hat{M}_{k-1}$. The goal of this block is to find a rigid transformation that overlaps Δ-map and $\hat{M}_{k-1}$, to create the current environment occupancy map $\hat{M}_k$. We adopted the algorithm proposed in [6], based on Discretized Hough transform and bidimensional correlation. The Discretized Hough transform finds the rotation that aligns Δ-map with $\hat{M}_{k-1}$, then the bidimensional correlation is applied to compute the translation that overlaps the two maps.

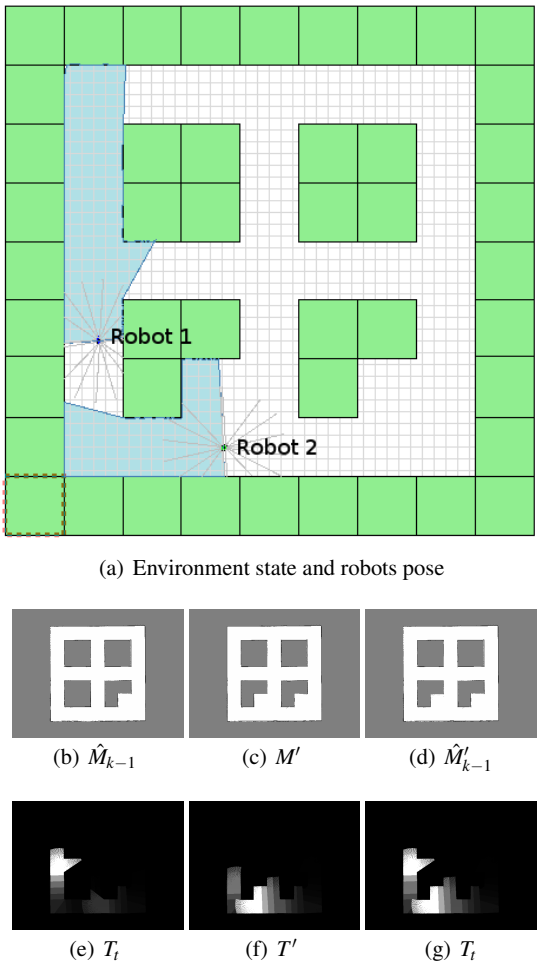
Local *Δ-mapping* in this work is the application of the *Scan Alignment* and *Map Merge* blocks.

In the *updateTimeMap* function in line 3 of Algorithm 1 the current maps $\hat{M}_{k-1}$ and T_t are updated according to M' and T' received from the other robots. For all the couples i, j every cell $\hat{M}_{k-1}(i, j)$ is updated if its value is older than the corresponding cell $\hat{M}'(i, j)$, so that the most recent (hence reliable) value is used. The information about the reliability is given by the time-maps T_t and T'.

When a robot receives a new map $\hat{M}'$ and a time-map T' from another robot, it merges the received time map with the previous map $\hat{M}_{k-1}$ and the local time-map T_t in order to produce $\hat{M}'_{k-1}$. T_t is also updated. For all the couples i, j the value of the cell $\hat{M}'_{k-1}(i, j)$ is set equal to the cell $\hat{M}'(i, j)$ if $T'(i, j) > T_t(i, j)$, otherwise it is set equal to $\hat{M}_{k-1}(i, j)$. The value of the cell $T_t(i, j)$ is set equal to $T'(i, j)$ if $T'(i, j) > T_t(i, j)$, otherwise it is not modified. Figure 2 shows the map merging process in a typical case. It can be noticed that changes received from another robot and local changes detected by the local *Δ-mapping* are merged in a consistent way. Cells belonging to areas that have been recently mapped have high corresponding time-map values (close to 1), so recent changes in the map resulting from a local *Δ-mapping* process are not discarded.

4 Coverage strategy

A team coordination strategy that actively searches modifications in the map has been developed. Without any coordination strategy all the robots could follow the same path or leave some areas not visited for a long time. This problem can be treated in partial similarity with the problem of multi-robot exploration. In the exploration approaches the aim is to discover a map starting from a completely unknown environment. In the case considered, the initial map is known, as well as



(a) Environment state and robots pose

(b) $\hat{M}_{k-1}$ (c) M' (d) $\hat{M}'_{k-1}$

(e) T_t (f) T' (g) T_t

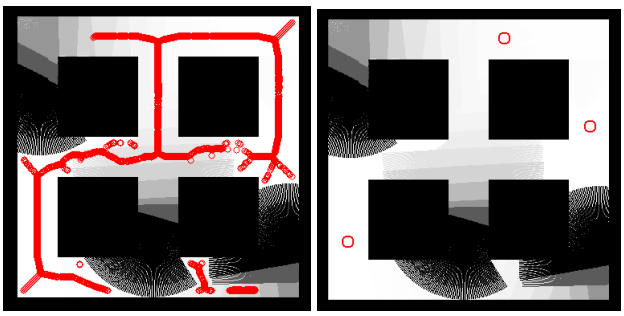
Fig. 2 Figures show the map merging in a typical case: 2(a) shows the pose of the robots, Robot 1 receives a map from Robot 2 and it uses it to update its map; 2(b) is the current map, 2(e) is the current time-map, 2(c) is the received map, 2(f) is the received time-map, 2(d) and 2(g) are the resulting map and time-map after the merging process.

the robot pose, but since the environment is persistently changing (pallets are added and removed), the reliability of the initial map decreases over time on the basis of the number of changes in the environment. For this reason, areas that have not been recently visited may become completely unknown, as the reliability of the map in those areas is very low.

Areas that need to be covered are the ones for which the corresponding value of the time-map is below a given threshold. For each robot, a set of points is extracted

to feed the path planning algorithms from a topological map, which is constructed from the grid-map representing the areas to be visited.

Many approaches obtain a topological representation from a grid-map, such as Voronoi diagrams [15] or topological operations [7]. The skeleton of an image is a good representation of the geometrical and topological properties of its shape, hence a morphological skeleton representation of the map is extracted using the algorithm described in [12], which is proven to be fast. Aset of points belonging to the skeleton is identified, with the constraint that each point has to be at a minimum distance from every other point. Each point becomes one goal point for the wavefront algorithm [11]. These goal points are then allocated to the team members by a distributed market-based task allocation algorithm described in the following subsection. Figure 3 shows how the goal points are obtained. In the time-map the black cells have the highest reliability and the white ones have the lowest reliability. In Figure 3(a) red points belong to the skeleton of the areas with reliability below a given threshold. In this case the team is composed by three robots, so three goal points are obtained as indicated in Figure 3(b).



(a) (b)

Fig. 3 Time-map and skeleton of areas to cover (a) and final goal points for three robots (b)

4.1 Distributed auction-based task allocation

Each goal point generated by the coverage strategy must be efficiently assigned to one of the robots in order to minimize travel time.

The *Hungarian method* performs a combinatorial optimization to solve the assignment problem in polynomial time. It guarantees the optimal solution, but it is a centralized algorithm, that requires a supervisor node and a matrix containing a row for each robot and a column for each task. Each cell contains the cost for the relative task. Moreover this approach requires the ability for all the robots to communicate, but this condition is not assured due to unreliable WiFi communication.

The used approach is then based on auctions, and it has been developed starting from the one proposed in [14]. Every goal point is assigned to an auction over a multicast network channel; the robots that receive the auction compute and send

back a bid. The auctioneer assigns the task to the robot with the best bid. The bid is computed according both to the robot's current position and to its queue of pending tasks. This approach does not guarantee the optimal solution, but it is robust to communication failures. The auctioneer is always a different robot, thus avoiding the problem of single point of failure.

5 Simulation Tests

The simulated environment of a logistic area already used in [3] and shown in Figure 4 is considered. The occupied green areas can be thought as containers or similar items stored before distribution. The environment is 35×35 m, the green areas in the center are 10×10 m and the corridors are 5 m wide.

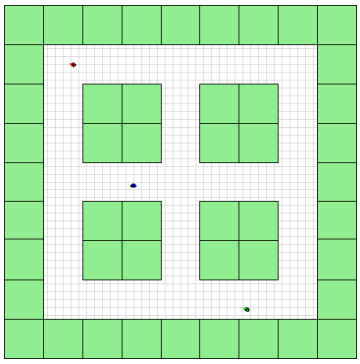


Fig. 4 The simulation environment.

$n = 3$ rovers are endowed with wheel encoders, a laser range finder and a WiFi board, and are able to localize themselves in the given environment. It is assumed that, once the rovers are correctly localized, a virtual fork-lift adds or removes one container every minute.
The rovers start moving with a simple obstacle avoidance policy. When the Δ-*mapping* process starts, the rovers move according to the coverage strategy described in Section 4. The quality of the map over time and the localization error are measured. The error on the estimate of the robot pose is strictly related to the quality of the map. Every Δ-*mapping* process induces some degradation of the map, due to the localization error which cannot be fully compensated by the *Map Merge* block.

Even after a consistent number of changes in the environment the rovers keep a map that is consistent with the environment and therefore the localization error remains low.

5.1 Simulation test 1

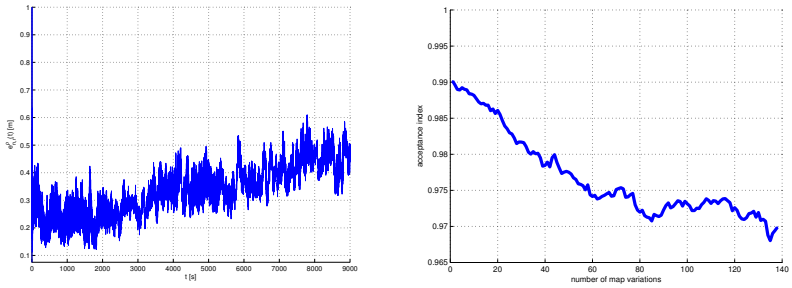
To demonstrate the effectiveness of the proposed approach first results related to $r = 10$ averaged runs are provided, where the *Δ-mapping* updating process lasts for approximately two hours each run.
The localization error of the i-th robot is defined as the distance between the ground-truth Cartesian position $(x_i^{gt}(t), y_i^{gt}(t))$ and its Cartesian position estimation as

$$e_i^{\rho}(t) = \sqrt{(x_i^{gt}(t) - \hat{x}_i(t))^2 + (y_i^{gt}(t) - \hat{y}_i(t))^2}. \tag{4}$$

We then define the average localization error for n robots over r runs as

$$e_{n,r}^{\rho}(t) = \frac{1}{r} \sum_{j=1}^{r} \sum_{i=1}^{n} \frac{e_i^{\rho}(t)}{n} \tag{5}$$

The localization error is reported in Figure 5(a). It can be noticed that the mean localization error remains lower than 0.6 m after approximately 2.5 hours. The qual-



(a) (b)

Fig. 5 Localization error and acceptance index for test 1.

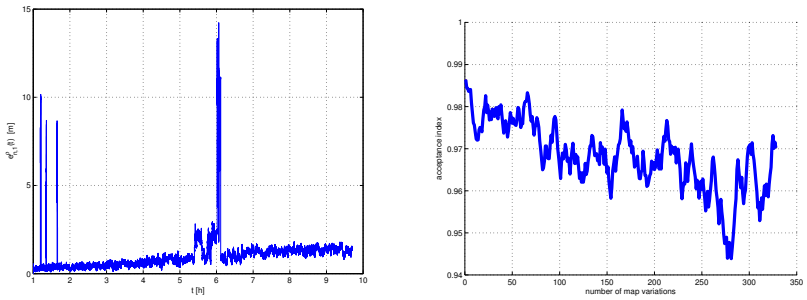
ity of the map for the duration of the test is also inspected. Visual inspection is often used, and numerical results by using the acceptance index described in [6] are also provided. They can be used as a measure of similarity between the real map and the estimated map.

Figure 5(b) shows the acceptance index mediate over the $n = 3$ robots and over $r = 10$ runs. After 140 variations the value obtained is 0.97, which is comparable with the one obtained with a typical grid-based SLAM algorithm (0.98).

5.2 Simulation test 2

Here the performances of the *Δ-mapping* process in long term operativity are tested. The simulation scenario is the same as for the previous test, but the virtual fork-lift adds and removes containers every two minutes. In this test the map updating pro-

cess lasts for approximately 9.5 hours, for a total number of 328 variations. Figure 6(a) shows the localization error for a single run, while Figure 6(b) shows the acceptance index over 328 variations. The sudden increase of the localization error after approximately 6 hours is due to one of the robots losing its localization for a short period of time. However as the robot receives an updated map it is able to recover itself. After 328 variations the acceptance index is still comparable with the one obtained in the previous test (see figure 5.1). Moreover, this acceptance index decreases to 0.97 after 9.5 hours, while in [3] the same error occurs after only 6 hours.



(a) (b)

Fig. 6 Localization error and acceptance index for test 2.

5.3 Simulation test 3

In this test the $n = 3$ robots perform different actions. The first robot performs Δ-*mapping* and sends map variations to the others team members; the second one only receives map variations but does not perform Δ-*mapping*; the last one neither perform Δ-*mapping* nor receives changes from the other team members. This test demonstrates the advantage in receiving map updates from other robots.

Figure 7 shows the localization error $e^{\rho}(t)$ for the three robots during a single run. Robot 1 remains well localized, while for robot 3 the error increases after approximately 3800 seconds; localization error for robot 2 starts to increase after 4720 seconds. This is due to the fact that robot 2 is able to merge the map updates received from robot 1, but this is still not sufficient in order to maintain a consistent map of the environment.

5.4 Computational load

In Figure 8 the CPU usage and memory occupation for each robot are reported. The algorithm runs on an Intel Core 2 Duo 2.4 Ghz with 2 GB of RAM. After approximately one minute the simulated fork lift starts to remove and add pallets,

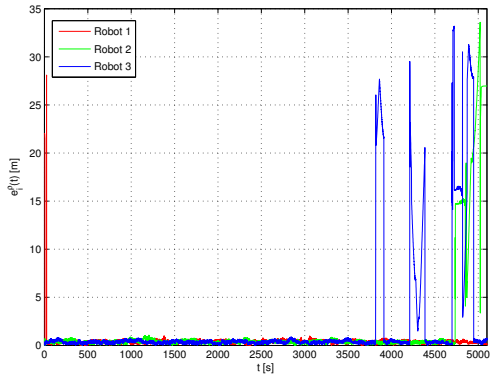


Fig. 7 Localization error for test 3.

and the *Δ-mapping* process starts. The peaks in CPU usage and memory occupation refer to the end of each local *Δ-mapping*, while the peaks in CPU usage only refer to the computation and assignment of the points to be visited. It can be noticed that after the beginning of the *Δ-mapping* process the memory usage steadily increases by only 12 MB, with peaks corresponding to the last phase of each local *Δ-mapping* process.

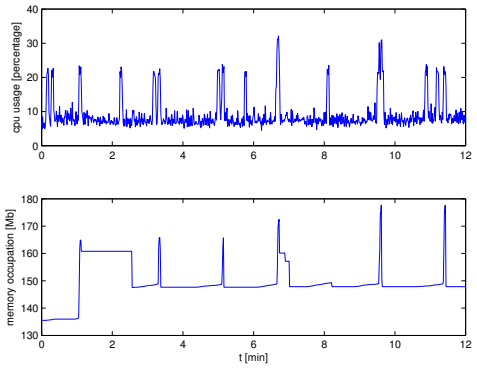
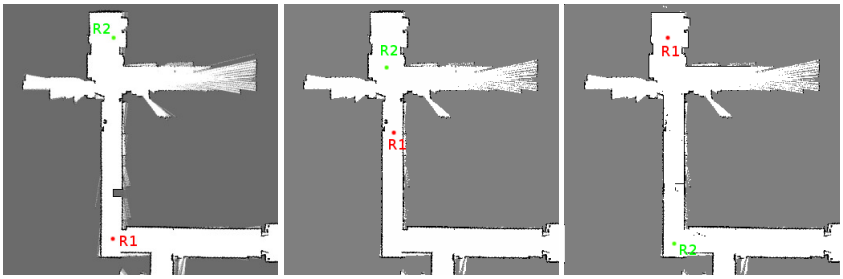


Fig. 8 CPU usage (upper plot) and memory usage (lower plot) in a simulated experiment.

6 Experimental Tests

An experiment in a real environment using two Pioneer P3DX robots has been carried out. Each robot is endowed with a SICK LMS200 laser rangefinder and a WiFi

board. A 1×1 m box has been placed in a 30×3 m corridor, and a classical Rao-Blackwellized SLAM process is first performed to obtain the map of the environment, as shown in Figure 9 (a). Then, the box is removed, R1 detects the absence of the box while travelling in the corridor, performs a *Δ-mapping* process and dispatches the map to R2, which updates its map (see Figure 9 (b)). Finally, the box is placed again in the previous place and R2 detects the presence of the box while travelling in the corridor, performs a *Δ-mapping* process and dispatches the map to R1, which updates its map (see Figure 9 (c)). It is worth noting that maps in Figure 9 (a),(b),(c) are the same for R1 and R2, even if they have not all perceived the same variations at the same time.
This preliminary test demonstrates the effectiveness of the proposed methodology in a simple but real scenario, since robots are able to merge the received maps from team members in a consistent way.



(a) Initial map (b) Dispatched map after the first change (c) Dispatched map after the second change

Fig. 9 The maps obtained during the experimental test.

7 Conclusions

In this work a methodology which is able to perform map updating in multi-robot applications dealing with dynamic environments is proposed. This methodology enables robots to detect variations in an environment, to generate an updated map containing only the persistent variations, to send this map to the other team members and to merge received maps in a consistent way. The approach is suitable for applications such as logistic applications, where a long-term operativity is required and the algorithm has to be computationally light and to use limited memory, in order to allow concurrent execution of other higher level services. Future works will be devoted to extensive experimental tests in real environments and to improvements of the coordination strategy.

References

1. Abrate, F., Bona, B., Indri, M., Rosa, S., Tibaldi, F.: Switching multirobot collaborative localization in symmetrical environments. In: IEEE International Conference on Intelligent RObots Systems (IROS 2008), 2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV) (2008)
2. Abrate, F., Bona, B., Indri, M., Rosa, S., Tibaldi, F.: Three state multirobot collaborative localization in symmetrical environments. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, pp. 1–6 (2009)
3. Abrate, F., Bona, B., Indri, M., Rosa, S., Tibaldi, F.: Map updating in dynamic environments. In: Proceedings of the 41st International Symposium on Robotics, pp. 296–303 (2010)
4. Birk, A., Carpin, S.: Merging occupancy grid maps from multiple robots. Proceedings of the IEEE **94**(7), 1384 –1397 (2006). DOI 10.1109/JPROC.2006.876965
5. Carlone, L., Ng, M.K., Du, J., Bona, B., Indri, M.: Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication. In: in Proceedings of IEEE International Conference on Robotics and Automation (2010)
6. Carpin, S.: Fast and accurate map merging for multi-robot systems. Auton. Robots **25**(3), 305–316 (2008). DOI http://dx.doi.org/10.1007/s10514-008-9097-4
7. Fabrizi, E., Saffiotti, A.: Extracting topology-based maps from gridmaps. In: IEEE Intl. Conf. on Robotics and Automation (ICRA, pp. 2972–2978 (2000)
8. Fenwick, J., Newman, P., Leonard, J.: Cooperative concurrent mapping and localization. pp. 1810 – 1817 vol.2 (2002). DOI 10.1109/ROBOT.2002.1014804
9. Howard, A.: Multi-robot simultaneous localization and mapping using particle filters. In: In Robotics: Science and Systems, pp. 201–208 (2005)
10. Howard, A., Sukhatme, G.S., Matarić, M.J.: Multi-robot mapping using manifold representations. Proceedings of the IEEE - Special Issue on Multi-robot Systems **94**(9), 1360 – 1369 (2006)
11. LaValle, S.: Planning Algorithms. Cambridge University Press (2004)
12. Maragos, P., Saffiotti, A.: Morphological skeleton representation and coding of binary images. In: IEEE Trans. on Acoustics, Speech, and Signal Processing (1986)
13. Siciliano, B., Khatib, O. (eds.): Springer Handbook of Robotics. Springer, Berlin, Heidelberg (2008). URL `http://dx.doi.org/10.1007/978-3-540-30301-5`
14. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers **C-29**(12), 1104–1113 (1981)
15. Thrun, S., Bücken, A.: Integrating grid-based and topological maps for mobile robot navigation. In: Proc. of the National Conference on Artificial Intelligence (1996)
16. Williams, S., Dissanayake, G., Durrant-Whyte, H.: Towards multi-vehicle simultaneous localisation and mapping. pp. 2743 –2748 (2002). DOI 10.1109/ROBOT.2002.1013647
17. Zhou, X.S., Roumeliotis, S.I.: Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems, pp. 1785–1792