Distributed Map Merging in a Robotic Network

Rosario Aragues and Jorge Cortes and Carlos Sagues

***Abstract*— The map merging problem in a robotic network consists of the construction of a global map of the environment using the partial (local) maps individually acquired by the robots. We build on ideas from distributed sensor fusion to provide an algorithmic solution to the map merging problem that uses the local maps as sensor measurements. We use a distributed average consensus algorithm that reaches asymptotic consensus on the value of the global map. The proposed solution can be used in a network with switching topology. We also present some experimental results of the algorithm.**

I. INTRODUCTION

Multi-robot applications are receiving a lot of attention in the last years. In these scenarios, a team of robots cooperatively perform some task in a more efficient way than a single robot would do. In addition to the classical issues associated to the operation of individual robots, these scenarios introduce novel challenges specific to the coordination of multiple robots.

Many multi-robot applications require that the agents know their own location and the location of other agents, as well as some information about the surrounding environment. Agents may have different knowledge of the environment due to the fact that they are located in different places and may observe only a portion of the environment. The map merging problem is motivated by the need to fuse the local robot information into a global knowledge of the environment with the objective of enabling the network to perform the task in an efficient way. The map merging problem consists of combining the environmental information acquired by all the robots in the team in order to build a global map that represents the global knowledge. This particular problem has been long studied in the robotic literature, see for instance [1]–[5].

In [1] a single global map is updated by all the robots. Robots search for features in the global map that have been observed by themselves along the exploration. Then, they use these coincident features to compute *implicit measurements* (the difference between the Cartesian coordinates of equal features must be zero) and use these constrains to update the map. In [2] maps are represented as constraint graphs, where nodes are scans measured from a robot pose and edges represent the difference between pairs of robot poses. Robot to robot measurements are used to merge two local maps into a single map. An optimization phase must be carried out in order to transform the constraint graph into a Cartesian map. [5] also represents the global map using a graph. Nodes are local metric maps and edges describe relative positions between adjacent local. The map merging process consists of adding an edge between the maps. Global optimization techniques are applied to obtain the global metric map. [3] merges two maps into a single one using robot to robot measurements to align the two maps and then detecting duplicated landmarks and imposing the implicit measurement constraints. [4] addresses the map merging problem using an *Information Filter* to acquire the local maps. This local map representation simplifies the merging process. [5] maintains a global map represented by a graph where nodes are local metric maps and edges describe relative positions between adjacent local. Map merging process consists of adding an edge between the maps. Global optimization techniques are applied to obtain the global metric map.

Other problem associated to multi-robot applications is that many existing algorithms require that each robot has the capability to communicate with all other robots at every time instant. A more realistic situation is when, at any time instant, robots can communicate only with a limited number of other robots, e.g., agents within a specific distance. These situations can be best modeled using communication graphs, where nodes correspond to the agents and edges represent communication capabilities between the robots. Additionally, since agents are moving, the topology of the graph may vary along the time, given rise to switching topologies, see for instance [6].

Among the general approaches to the synthesis of coordination algorithms for multi-robot systems we mention centralized and distributed strategies. In a centralized strategy, there exists a central node that compiles all the information from other robots, performs the computations, and propagates the processed information or decisions to the other nodes. Centralized approaches have many drawbacks: the whole system can fail if the central node fails, leader selection algorithms may be needed, and a (direct or indirect) communication of all agents with the central system may be required. On the other hand, in distributed systems, all robots play the same role, and therefore the computations can be distributed among all the agents. In addition, distributed systems are naturally more robust to individual failures.

In this paper, we propose a solution to the map merging problem for a robotic network modeled by a communication graph, where all computations are distributed among the agents and where, at every time step, robots only use its own

This work was supported by projects MEC DPI2006-07928 and IST-1-045062-URUS-STP

R. Aragues is with DIIS - I3A, University of Zaragoza, María de Luna, 50018 Zaragoza, Spain raragues@unizar.es

J. Cortes is with the Department of Mechanical and Aerospace Engineering, University of California San Diego, 9500 Gilman Dr, La Jolla, California, 92093-0411 , USA cortes@ucsd.edu

C. Sagues is with DIIS - I3A, University of Zaragoza, María de Luna, 50018 Zaragoza, Spain csagues@unizar.es

(local) data and the information received from its neighbors in the graph. The solution is based on distributed average consensus algorithms for data fusion problems [7], [8].

II. PROBLEM DESCRIPTION

We consider a situation where n robots have explored an unknown environment building a stochastic map based on their own observations. Robots stop exploring and merge their information in order to obtain a global estimate of the map. The n robots have communication capabilities that enables information exchange with other robots within a specific distance.

Each robot has observed and estimated the position of some static features in the environment. The *data association problem* consists of establishing relationships between the features observed by different robots. In this paper this problem is not discussed and a perfect data association is simulated. Every feature has associated an identifier so that the same features in different maps have the same id number and different features have different id. The *initial correspondence problem* consists of establishing a global reference frame for all robots so that all local maps are expressed in this reference. In this paper it is assumed that all local maps are represented in the same reference frame. These two problems are highly correlated and many solutions have been presented to solve them. For the initial correspondence problem, a simple (but not very flexible) solution is to initialize all robots at known relative positions. Another solution is to make all robots start from nearby positions; then, techniques from vision [9] can be used to recover the relative pose between the cameras (agents) with the condition that a minimal number of landmarks must be visible from all the cameras. Alternatively, robots may start from completely unknown poses and use robot-to-robot measurements to estimate the relative positions and orientations [3]. The data association problem simplifies if the features are expressed in the same reference frame. If the sensor is a camera, the use of SIFT or SURF feature descriptors [10] also simplifies the data association because different features are more easily detected.

The problem of creating a global estimate of the map can be expressed as the estimation of a vector of unknown and constant parameters $\theta \in \mathbb{R}^m$ (the static position of the features in the map), combining the noisy observations of the n distributed sensors (the local maps and covariances estimated by the robots). If we express the problem in this way, we can apply the distributed sensor fusion solution in [7] that we briefly describe next for reference.

A. Distributed averaging for sensor fusion

Assume all robots know the total amount of parameters to be estimated m and the order of their local parameters relative to $\theta \in \mathbb{R}^m$. Then, the local observations of each robot i, for $i \in \{1, \dots, n\}$, can be expressed as

$$y_i = A_i \theta + v_i, \tag{1}$$

where $y_i \in \mathbb{R}^m_i$ are the m_i observations (estimation of the feature positions), $A_i \in \mathbb{R}^{m_i \times m}$ relates the parameters θ and the observations y_i, and v_i is a random Gaussian variable with zero mean and covariance matrix $\Sigma_i \in \mathbb{R}^{m_i \times m_i}$.

The matrices A_i establish the relation between the parameters to be estimated (Cartesian coordinates of the features) and the local estimates of each robot. Since the local estimates are also the Cartesian coordinates of features and all local maps are expressed in the same reference frame, the matrices A_i are just a permutation of an identity matrix with additional zero-columns.

Collecting the information in the n robots we have

$$y = A\theta + v \tag{2}$$

with

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

We assume that the noises v_i are independent since every robot has constructed the map based on its own observations. The covariance of the matrix v is then $\Sigma = \mathbf{diag}(\Sigma_1, \cdots, \Sigma_n)$.

The maximum-likelihood estimate $\hat{\theta}_{ML}$ for the parameters θ based on the observations y and its error covariance matrix $\Sigma_{\hat{\theta}_{ML}}$ are

$$\begin{array}{rcl} \hat{\theta}_{ML} & = & \left(A^T \Sigma^{-1} A\right)^{-1} A^T \Sigma^{-1} y \\ \Sigma_{\hat{\theta}_{ML}} & = & \left(A^T \Sigma^{-1} A\right)^{-1} \end{array} \tag{3}$$

where

$$A^T \Sigma^{-1} A = \sum_{i=1}^{n} A_i^T \Sigma_i^{-1} A_i \tag{4}$$

$$A^T \Sigma^{-1} y = \sum_{i=1}^{n} A_i^T \Sigma_i^{-1} y_i \tag{5}$$

Following [7], we can solve this problem as a distributed average consensus problem on the values of two variables P and q, termed respectively *composite information matrix* and *composite information state*. This algorithm achieves asymptotic consensus (nodes reach consensus when $t \to \infty$) and has the property that computations in every node are carried out using only local information in node i and information from its neighbors in the graph.

The variables P and q are initialized at $t = 0$ as follows

$$\begin{array}{rcl} P_i(0) & = & A_i^T \Sigma_i^{-1} A_i \\ q_i(0) & = & A_i^T \Sigma_i^{-1} y_i \end{array} \tag{6}$$

At every step, P and q are updated using the value in the node and values from the neighbor nodes

$$P_i(t+1) = w_{ii}(t) P_i(t) + \sum_{j \in N_i(t)} w_{ij}(t) P_j(t) \tag{7}$$

$$q_i(t+1) = w_{ii}(t) q_i(t) + \sum_{j \in N_i(t)} w_{ij}(t) q_j(t) \tag{8}$$

where $N_i(t)$ is the set of neighbors of the node i and $w_{ii}(t), w_{ij}(t) \in [0,1]$ are some weights. In our case, we use the Metropolis weights given by

$$w_{ij}(t) = \begin{cases} \frac{1}{1+\max\{d_i(t),d_j(t)\}} & if\ j \in N_i(t) \\ 1 - \sum_{k \in N_i(t)} w_{ik}(t) & if\ i = j \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Using this iterative scheme, one can show [7] that, as t tends to infinity, all nodes reach consensus on the values of P and q, and the agreement values are given by

$$\lim_{t\to\infty} P_i(t) = \frac{1}{n}\sum_{i=1}^{n} A_i^T \Sigma_i^{-1} A_i, \tag{10}$$

$$\lim_{t\to\infty} q_i(t) = \frac{1}{n}\sum_{i=1}^{n} A_i^T \Sigma_i^{-1} y_i. \tag{11}$$

where n is the total number of nodes in the network.

Therefore, $\hat{\theta}_{ML}$ can be asymptotically computed by each node, combining P and q

$$\hat{\theta}_{ML} = \lim_{t\to\infty} P_i(t)^{-1} q_i(t) \tag{12}$$

In order to compute $\Sigma_{\hat{\theta}_{ML}}$, it is necessary that nodes also know n

$$\Sigma_{\hat{\theta}_{ML}} = \lim_{t\to\infty} \frac{1}{n} P_i(t)^{-1} \tag{13}$$

B. Global map merging as a sensor fusion problem

The approach described in the section cannot be directly applied to the map merging problem because the total number of parameters m is unknown (each robot only has information of the m_i parameters observed by itself). Therefore, the matrices A_i are unknown. The information available at every node is instead

$$y_i = \tilde{A}_i \tilde{\theta}_i + v_i, \tag{14}$$

where $\tilde{A}_i \in \mathbb{R}^{m_i \times m_i}$, $\tilde{\theta}_i \in \mathbb{R}^{m_i}$. The relationship between the unknown matrix A_i and the known $\tilde{A}_i$ is that the first matrix is a permutation of the columns of the second matrix with additional zero columns for those parameters in θ not observed by robot i.

Our goal is to simultaneous estimate m, A_i, $\hat{\theta}_{ML}$ and $\Sigma_{\hat{\theta}_{ML}}$ in a distributed way based on the information exchanged between each robot and its neighbors in the communication graph.

III. APPROACHES

We propose two alternative approaches to solve the problem of the global map merging. The first approach begins by reaching consensus on the total amount and order of the parameters to be estimated, and then computes the matrices A_i and executes the average consensus algorithm described in Section II-A. The second approach does not require the initial consensus stage on the number and order of the parameters. Instead, each node updates the number and order of parameters at every time step using the information of its neighbors and, simultaneously, arranges P and q according to this information.

A. First Approach

Let us provide an overview of the steps of the first approach:

BFS tree construction: The agents begin by constructing a BFS spanning tree in the undirected graph using a variation of the flood with termination algorithm, see e.g., [11]. As a result, all nodes in the graph know the identity of its parent and its children in the graph, and also know its role (root, leaf of regular node).

Parameter computation: After this, the leaves initiate the incremental computation of the identity and order of the parameters to be estimated. When all information is available to the root, it computes the final order and amount of parameters and uses a flooding algorithm to propagate this information to all nodes in the graph.

Distributed averaging: Finally, the nodes compute the matrices A_i and initiate the consensus algorithm described in Section II-A to compute the maximum-likelihood estimate of the global map.

Next, we describe each of these steps in more detail.

1) BFS tree construction: In order to construct the BFS tree, all nodes in the undirected graph must know if they are the root and also must know which nodes are their neighbors in the graph.

All nodes initialize ’parent id’ to null and ’children set’ to be the set of neighbors. The root node initiates the process sending a ’parent request’ to all its neighbors. When a node receives a ’parent request’ message, it checks the value of its ’parent id’; if it is null, then it updates this value to be the sender id; if the node already has a parent, it replays with a ’parent reject’ message. If during a step a node receives multiple ’parent request’ messages, it selects as a parent the node with the smallest id and sends a ’parent reject’ message to the other nodes. Nodes remove the parent id from the list of children.

When a node receives a ’parent reject’ message, it updates its children set, deleting the sender from this set.

A node with an empty children set is a leaf.

Some steps of this algorithm are illustrated in Fig. 1.

2) Incremental computation of identity and order of parameters: When a node detects that it is a leaf, it starts the process of computing the total amount and order of parameters. Every node has a vector with the identity of the parameters observed by himself during the exploration. Leaves sort their vectors and send them inside an ’up’ message to their parent.

Nodes in the graph compile all the identity vectors sent by their children and fuse this information with its own identity vector. Once a node has received ’up’ messages from all its children, it sends an ’up’ message to its parent with the resulting identity vector.

When the root has received all the information from its children, it computes the final (global) identity vector. This global vector contains all the identities of the parameters observed by all the nodes in the graph, without repetition, and sorted in ascendant order.

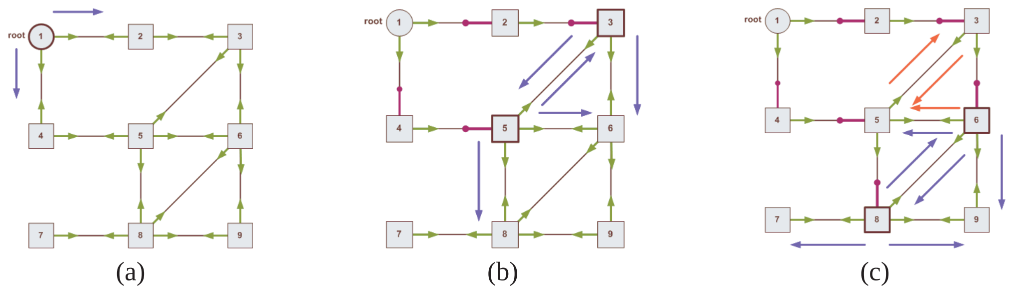


Fig. 1. **Example of BFS tree construction**. **(a)**: The root (circle) initiates the process sending a parent request to all its children (blue arrows). **(b)**: Nodes 3 and 5 update their parent and send parent requests (blue arrows) to all their children. **(c)**: Node 3 sends a parent reject (red arrow) to node 5 as a response to the parent request received. Node 5 behaves in the same way. Node 6 selects as parent the node with the minimal identifier (node 3) and sends a parent reject to node 5; then, it sends a parent request to all its children. Node 8 updates its parent and sends a parent request to its children.

Then the root sends this final vector to all its children in a 'down' message. Every node that receives a 'down' message, updates its identity vector to be the global vector and propagates this information sending a 'down' message to all its children. Using this global vector, nodes can compute the A_i matrices which are used in the algorithm that computes the global map.

Some steps of this algorithm are illustrated in Fig. 2.

3) Matrix computation and distributed averaging: Every node i has knowledge of the m_i parameters observed by himself: the y_i vector with the m_i observations, the $\Sigma_i \in \mathbb{R}^{m_i \times m_i}$ covariance matrix, the $\tilde{A}_i \in \mathbb{R}^{m_i \times m_i}$ observation matrix which relates parameters with observations, and its local identity vector $I_{\tilde{\Theta}_i} \in \mathbb{N}^{m_i}$. When the process of estimating the total amount and order of parameter finishes, the node also knows the global identity vector, which we name $I_\Theta \in \mathbb{N}^m$.

The first step is the expansion of the identity vector and the observation matrix. The expanded identity vector $I^+_{\tilde{\Theta}_i}$ is the result of appending to the end of $I_{\tilde{\Theta}_i}$ all the ids which are in I_Θ but are not in $I_{\tilde{\Theta}_i}$. Since the node has not observed these additional parameters, matrix $\tilde{A}_i$ can be expanded adding $m - m_i$ additional zero-columns at the end. This expanded matrix is called $\tilde{A}^+_i$.

Then, nodes compute the permutation that transforms $I^+_{\tilde{\Theta}_i}$ into I_Θ. Applying this permutation to the columns of $\tilde{A}^+_i$ nodes obtain the matrix A_i.

Once all matrices have been expanded and permuted, the distributed averaging algorithm to compute the global map can be executed.

B. *Second Approach*

In the second approach, nodes do not need to compute a tree in the graph and agree upon the order and amount of parameters before starting the average consensus algorithm. Instead, every node starts with the information related to the parameters observed by itself and, incrementally, discovers the new parameters in the information sent by its neighbors. Then, it adds this information to its own knowledge, propagating it to its neighbors in the next iteration.

At the beginning, all nodes have access to its local information. At $t = 0$, node i knows

- m_i the number of parameters observed by node i during the exploration;
- $m_i(0)$ the total number of parameters discovered by node i at $t = 0$ is equal to m_i;
- $y_i \in \mathbb{R}^{m_i}$ the observations of the parameters. This value does not change along the iterations;
- $\Sigma_i \in \mathbb{R}^{m_i \times m_i}$ the covariance matrix. This value also remains unchanged during the algorithm execution;
- $I_{\tilde{\Theta}_i}(0) \in \mathbb{N}^{m_i(0)}$ the vector with the identities of the parameters discovered by node i at time $t = 0$, which is initialized with the identifiers and the order observed by the node i during the exploration;
- $\tilde{A}_i(0) \in \mathbb{R}^{m_i \times m_i(0)}$ the observation matrix relating the parameters in $I_{\tilde{\Theta}_i}(0)$ an the observations in y_i.

Using this information, each node can compute the initial values for $\tilde{P}_i$ and $\tilde{q}_i$

$$\begin{aligned} \tilde{P}_i(0) &= \tilde{A}_i^T(0)\Sigma_i^{-1}\tilde{A}_i(0) \\ \tilde{q}_i(0) &= \tilde{A}_i^T(0)\Sigma_i^{-1}y_i \end{aligned} \tag{15}$$

At each iteration, nodes update the order and amount of parameters observed up to that time instant, arrange the matrices P and q accordingly, and compute the new values for P and q. At $t + 1$, node i can access the information received from its neighbors and the information stored at node i: $I_{\tilde{\Theta}_j}(t)$, which is the vector with the identities of the parameters discovered by node j at time t, and $\tilde{P}_j(t)$, $\tilde{q}_j(t)$ which are arranged according to $I_{\tilde{\Theta}_j}(t)$, for $j \in N_i(t) \cup \{i\}$.

First, node i uses $I_{\tilde{\Theta}_j}(t)$ for $j \in N_i(t) \cup \{i\}$ and computes the new order and amount of parameters $I_{\tilde{\Theta}_i}(t+1)$. Then, it expands and arranges the previous matrices $\tilde{P}_j(t)$, $\tilde{q}_j(t)$ according to this new order of parameters. For all $j \in N_i(t) \cup \{i\}$, $I^+_{\tilde{\Theta}_j}$ is the result of appending to the end of $I_{\tilde{\Theta}_j}(t)$ all the ids which are in $I_{\tilde{\Theta}_i}(t+1)$ but are not in $I_{\tilde{\Theta}_i}(t)$.

$$\tilde{P}^+_{j,I^+_{\tilde{\Theta}_j}} = \begin{bmatrix} \tilde{P}_j(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \tilde{q}^+_{j,I^+_{\tilde{\Theta}_j}} = \begin{bmatrix} \tilde{q}_j(t) \\ \mathbf{0} \end{bmatrix}. \tag{16}$$

Then, node i computes the permutation that transforms $I^+_{\tilde{\Theta}_j}$ into $I_{\tilde{\Theta}_i}(t+1)$ and applies this permutation to the rows and columns of $\tilde{P}^+_{j,I^+_{\tilde{\Theta}_j}}$ and to the rows of $\tilde{q}^+_{j,I^+_{\tilde{\Theta}_j}}$, obtaining respectively $\tilde{P}^+_{j,I_{\tilde{\Theta}_i}(t+1)}$ and $\tilde{q}^+_{j,I_{\tilde{\Theta}_i}(t+1)}$.

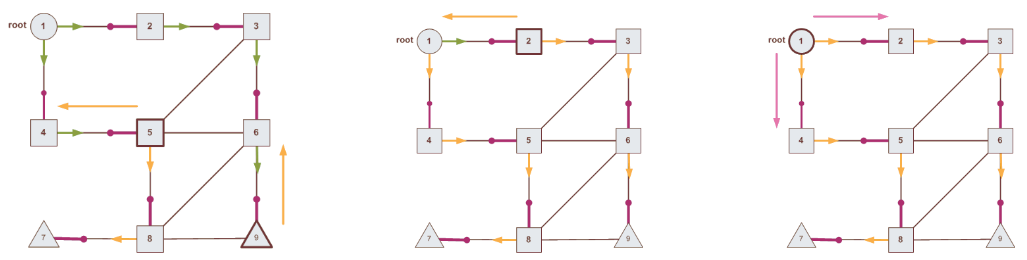


Fig. 2. **Incremental computation of identity and order of parameters**. **(a)**: Nodes 5 and 9 send an up message (yellow arrow) to their parent. **(b)**: The root receives an up message from node 2. It had previously received an up message from its other children (node 4). **(c)**: The root computes the final (global) parameter vector and starts a flooding process to communicate this vector to all the nodes. It sends a down message (pink arrow) to all its children.

Once all matrices are arranged according to the same parameter identity vector, an iteration can be carried out at node i to compute the new $\tilde{P}_i(t+1)$ and $\tilde{q}_i(t+1)$

$$\tilde{P}_i(t+1) = w_{ii}(t)\tilde{P}^+_{i,I_{\tilde{\Theta}_i}(t+1)} + \sum_{j\in N_i(t)} w_{ij}(t)\tilde{P}^+_{j,I_{\tilde{\Theta}_i}(t+1)}, \tag{17}$$

$$\tilde{q}_i(t+1) = w_{ii}(t)\tilde{q}^+_{i,I_{\tilde{\Theta}_i}(t+1)} + \sum_{j\in N_i(t)} w_{ij}(t)\tilde{q}^+_{j,I_{\tilde{\Theta}_i}(t+1)}. \tag{18}$$

Notice that the obtained matrices $\tilde{P}_i(t+1)$, $\tilde{q}_i(t+1)$ can be transformed into the matrices computed by the first approach $P_i(t+1)$, $q_i(t+1)$ by an expansion and arrangement process based on the parameter identity vectors $I_{\tilde{\Theta}_i}(t+1)$ and I_{Θ}. Therefore, the results obtained by both approaches are completely equivalent and the convergence speed is the same.

IV. DYNAMIC MAP MERGING

The problem solved by our algorithm is an static map merging where robots compute the global map for the local maps acquired until some time instant. If a more flexible solution is desired, where robots resume the exploration and in some posterior instant decide to merge their local maps again, a dynamic map merging strategy must be provided. In this section, we propose a solution to this problem.

It is worth mentioning that the distributed averaging algorithm in Section II-A also works when the topology of the network is switching. Therefore, it is not necessary for the robots to remain stationary while they compute the global map. The convergence of the algorithm is guaranteed so long as the collection of communication graphs that occur infinitely often is jointly connected. Robots can go on exploring and building local maps while they run the global map merging algorithm. Based on this property, we propose the following strategy for dynamic exploration:

- At time t_0, nodes begin their exploration, building a local map of the environment;
- At time instant t_1, robots create a new empty local map and start the map merging process for merging the local maps at time t_0 (the goal global map will be for time t_0);
- At time instant t_k, robots initiate the same process, creating a new empty local map for t_k and starting the map merging for maps at t_{k-1}. If desired, they can simultaneously continue with older map merging algorithms (for any previous t_l).

Since all local maps at time t_k are independent of all previous local maps, the computed global maps are also independent. This means that all nodes are computing and storing global submaps. All these global submaps can be merged into a whole global map using the submapping strategy in [12].

V. SIMULATIONS

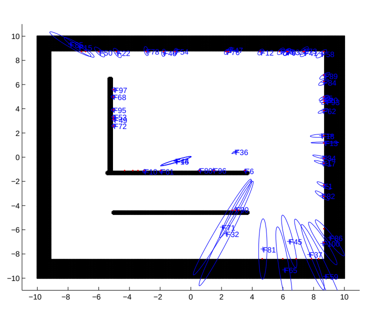
In order to show the performance of the algorithm, a simulation has been carried out where a team composed by 9 robots have explored an environment, obtaining a set of local maps, see Fig. 3(a-i).

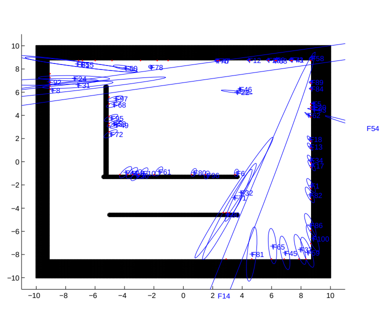
In the simulation, robots estimate their motion based on odometry information and sense the environment using a camera device that provides bearings to the landmarks. Every robot explores only a small portion of the environment so that none robot observes all the landmarks. Due to the short trajectories followed by the robots and to the nature of bearing-only data, landmark estimates present large uncertainty in the local maps. In Fig. 3(a-i) black dots represent obstacles, red dots are the ground-truth location of landmarks, blue crosses are the estimates of the landmark positions and blue ellipses are the estimated covariance.

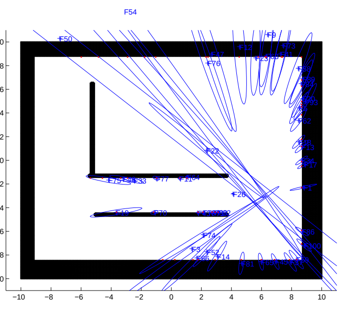
The goal of the map merging process is the combination of the local maps to obtain a the maximum-likelihood estimate for the global map (see Fig. 4(a)). Every node executes the algorithm described in this paper so that their estimates asymptotically approach this maximum-likelihood global map. Even thought the consensus is asymptotically reached, we can see that in practice, the convergence of the averaging algorithm is very fast, and in a few steps the estimates at every node approach the global map. In Fig. 4(b) we show the global map estimated by robot 1 at time step 3.

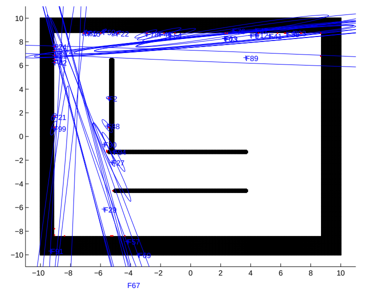
VI. CONCLUSIONS AND FUTURE WORK

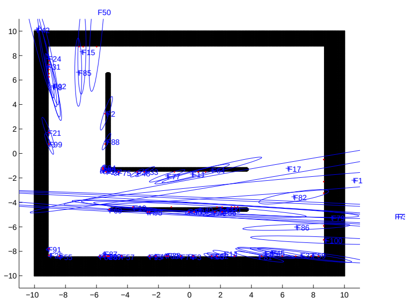
In this paper, we have formulated a global map merging problem as a sensor fusion problem. This has enabled us

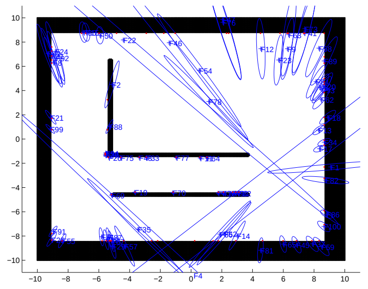

(a)

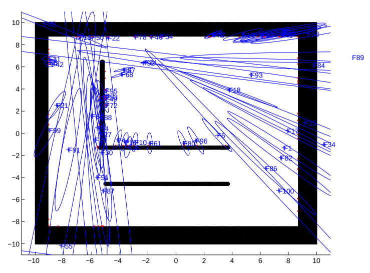

(b)

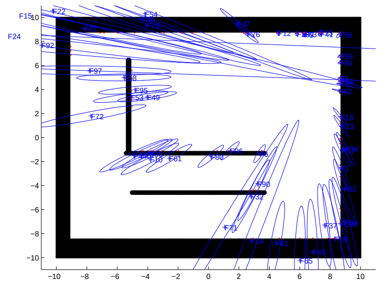

(c)


(d)


(e)


(f)


(g)


(h)

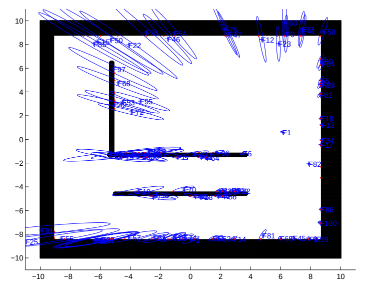
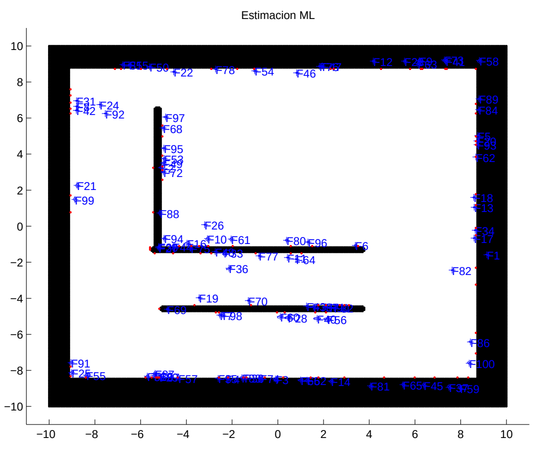
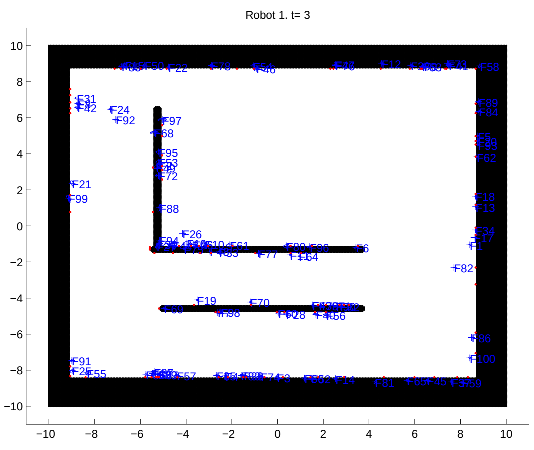

(i)

Fig. 3. **Local maps**. **(a-i)** Local maps obtained by robots 1 to 9. Red dots are the ground-truth location of landmarks. Blue crosses are the estimates of the landmark positions and blue ellipses are the estimated covariance.



(a)



(b)

Fig. 4. **Experimental results**. **(a)** Maximum-likelihood goal global map. **(b)** Global map estimated by robot 1 at $t = 3$

to propose a solution based on distributed averaging. The resulting algorithm is distributed and asymptotically correct.

This algorithm computes a static consensus, that is, it computes the average of some static inputs (the local maps). We have briefly discussed a dynamic fusion strategy in Section IV. As future work, we plan to investigate distributed dynamic consensus algorithms, which instead of computing an average of a static input, track the average of time-varying inputs. For map merging problems, this is equivalent to the situation where robots continue to explore and update their local maps while, simultaneously, running a distributed algorithm to track the global map.

Another area of future research is the design of optimal motion control strategies for improved coverage of the environment. The idea is that robots cooperatively decide their next movements in order to optimally explore more uncertain regions and improve the quality of the global map.

REFERENCES

[1] D. Rodríguez-Losada, F. Matía, and A. Jiménez, "Local maps fusion for real time multirobot indoor simultaneous localization and mapping," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, New Orleans, LA, USA, April 2004, pp. 1308–1313.

[2] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, July 2006.

[3] X. S. Zhou and S. I. Roumeliotis, "Multi-robot slam with unknown initial correspondence: The robot rendezvous case," in *in Proceedings of IEEE International Conference on Intelligent Robots and Systems*, Beijing, China, October 2006, p. 17851792.

[4] S. Thrun and Y. Liu, "Multi-robot SLAM with sparse extended information filers," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*. Sienna, Italy: Springer, 2003.

[5] H. J. Chang, C. S. G. Lee, Y. C. Hu, and Y.-H. Lu, "Multi-robot slam with topological/metric maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007*, October 2007, pp. 1467–1472.

[6] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, June 2008, manuscript preprint. Electronically available at http://www.coordinationbook.info.

[7] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Symposium on Information Processing of Sensor Networks (IPSN)*, Los Angeles, CA, Apr. 2005, pp. 63–70.

[8] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Approximate distributed Kalman filtering in sensor networks with quantifiable performance," in *Symposium on Information Processing of Sensor Networks (IPSN)*, Los Angeles, CA, Apr. 2005, pp. 133–139.

[9] C. Sagüés, A. C. Murillo, J. J. Guerrero, T. Goedemé, T. Tuytelaars, and L. V. Gool, "Localization with omnidirectional images using the 1d radial trifocal tensor," in *Proc of the IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 551–556.

[10] A. C. Murillo, J. J. Guerrero, and C. Sagüés, "Surf features for efficient robot localization with omnidirectional images," in *IEEE/RSJ Int. Conf. on Robotics and Automation*, 2007, pp. 3901–3907.

[11] D. Peleg, *Distributed Computing. A Locality-Sensitive Approach*, ser. Monographs on Discrete Mathematics and Applications. Philadelphia, PA: SIAM, 2000.

[12] S. Williams, G. Dissanayake, and H. Durrant-Whyte, "An efficient approach to the simultaneous localisation and mapping problem," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, 2002.