The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

A Distributed Maximum Likelihood Algorithm for Multi-Robot Mapping

Dario Lodi Rizzini, Stefano Caselli
RIMLab - Robotics and Intelligent Machines Laboratory
Dipartimento di Ingegneria dell'Informazione
University of Parma, Italy
E-mail {dlr,caselli}@ce.unipr.it

Abstract— **In the last decade, several algorithms, usually based on information filtering techniques, have been proposed to address multi-robot mapping problem. Less interest has been devoted to investigate a parallel or distributed organization of such algorithms in the perspective of multi-robot exploration.**

In this paper, we propose a distributed algorithm for map estimation based on Gauss-Seidel relaxation. The complete map is shared among independent tasks running on each robot, which integrate the independent robot measurements in local submaps, and a server, which stores contour nodes separating the submaps. Each task updates its local submap and periodically checks for inter-robot data associations. Gauss-Seidel relaxation is performed independently on each robot and afterwards on the contour nodes set on the server. Results illustrate the potential and flexibility of the new approach.

I. INTRODUCTION

Mapping has become an important task for several robotic applications. A global representation of the environment is often required by a mobile robot to perform other tasks like navigation. When such representation is not available, the robot has to build the map using its uncertain motion information and sensor observations. Map building and localization are coupled problems and in literature are often referred to as *Simultaneous Localization and Mapping (SLAM)*. In literature, several methods, either based on Bayesian filtering or on maximum-likelihood (ML) estimation, have been proposed to address SLAM and mapping problems.

Multi-robot SLAM is the construction of a joint map by a group of robots that concurrently explore a given environment. This problem raises additional issues with respect to the single-robot formulation. First, in a multi-robot context the map may be stored in a single repository or splitted into local submaps hosted by each robot. Second, the message exchange rate depends on the way the map is built. On one hand, robots continuously send the odometric information and the acquired observation to the map repository in a centralized solution. On the other hand, the required communication band is minimal, when the local maps are learnt independently by each robot and finally merged after the acquisition. Thus, multi-robot mapping algorithms may be classified with respect to the *degree of distribution* as well as to the required *communication rate*. In the initialization of a multi-robot system, it is usually assumed that the mutual initial poses of the robots are known either perfectly or with uncertainty.

ML methods are good candidates for multi-robot mapping, since they rely on the graphical formulation of the mapping problem. Indeed, the graphical model captures the connectivity between the variables of the problem allowing a decomposition of the map among different robotic platforms. Nonetheless, to the best of our knowledge multi-robot ML mapping has not been thoroughly investigated in the robotics community except for few recent works [1].

In a previous paper [2], we proposed a parallel Gauss-Seidel relaxation method to solve a constraint network. The algorithm is based on the constraint network decomposition and on the node reordering induced by this decomposition. In this paper, we develop a distributed mapping algorithm based on the previous contribution. In particular, the mapping system consists of *mapping units* and of a *server*. The mapping units are independent tasks running on each robot that integrate the measurements in a local map. The server handles the repository of the contour nodes, which separate robot submaps and have been introduced in [2]. Each mapping unit updates its local submap and periodically checks for inter-robot data associations. Gauss-Seidel relaxation is performed independently on each robot and afterwards on the contour nodes set on the server.

II. RELATED WORK

Several adaptations of single robot mapping techniques have been proposed for the multi-robot context. The standard *Extended Kalman Filter* is not considered suitable for multi-robot extension [3] due to the quadratic complexity of the algorithm that would be reflected into a quadratic complexity of the communication. Therefore, the research focused on *information filters*, which exploit the sparsity of information matrix to decompose the problem. Thrun *et al.* [4] proposes a multi-robot *Sparse Extended Information Filter* that exploits two properties of the filter: *additivity* of information, which allows the composition of multiple maps, and *locality*, which confines the submap updates to the pose and the landmarks detected by a single robot. Thus, the computation of the joint map is rather a batch operation consisting of a coordinate transformation of the submaps and a detection of corresponding landmarks. *Constrained Local Submap Filter* (CLSF) [5] also performs map merging in a "patch-work fashion", but it introduces a better definition of map-to-map and vehicle-to-vehicle data association through the computation of the *maximum common subgraph*. An offline map merging algorithm based on Hough transform has been proposed in [6], [7]. In [8] the *Split Covariance Intersection* method has been combined with information filter to extract submaps of size adaptable to the bandwidth of the channel.

Howard [9] proposes an adaptation of Rao-Blackwellized Particle Filter for multi-robot SLAM and addresses the problem of robot mutual observation. Several other works adopting particle filters and shared grid maps [10], [11] focus more on the issues of robot exploration than on mapping.

The extension of ML methods for multi-robot SLAM has been addressed sporadically and in quite recent works. While ML methods rely on graphical formulation, which is naturally adaptable to a decomposition of the mapping task among different robotic platforms, the interest of the robotic community for such approach increased only during the last few years. To our knowledge, the *Multi-Frontal QR factorization* (MFQR) algorithm [12] is the first attempt to address such issue. MFQR exploits the structural sparsity of the problem for a recursive tree-based factorization into small dense factors. The factorization can be performed in parallel with only the exchange of QR update messages. *Tectonic SAM* [13] is an algorithm not designed for multi-robot ML, but to speed-up the optimization by dividing the problem into submaps. Nonetheless, it is suitable for an adaptation. However, none of the mentioned works discusses the issue of vehicle to vehicle data association, which is the real challenge for robot communication, nor proposes an online incremental use of the algorithm. A recent work [1] discusses the application of several ML techniques, like gradient descent and relaxation, to multi-robot localization and mapping.

III. A DECOMPOSABLE GAUSS-SEIDEL SOLVER

This section discusses the general formulation of the maximum likelihood (ML) approach, a solution algorithm based on Gauss-Seidel relaxation to solve the resulting equations and a decomposition of the graph suitable for multi-robot mapping. Graph-based SLAM can be expressed according to *feature based* or *delayed-state* representations [14]. In the latter representation all the map variables correspond to robot poses, since the map features are implicitly referred to pose values either by anchoring sensor observations to a local frame [15] or by marginalizing feature-based maps [16]. Thus, the mapping problem is modelled as a graph whose nodes correspond to the robot poses and whose edges correspond to the constraints between pairs of poses. The Gauss-Seidel relaxation illustrated in the following may be applied to both representations, but in this paper we will refer to the more homogeneous delayed-state one.

Furthermore, we present a node reordering technique to achieve a decomposition of the constraint network into clusters that can be solved independently. Such decomposition corresponds to a block-border diagonal form of the linearized matrix associated to the network. The node reordering was proposed in [2] for a better exploitation of multi-core processors. Here, the decomposition is applied to distributed mapping.

A. Pose Graph and Gauss-Seidel Relaxation

Let $\mathbf{p} = [p_1 \ \cdots \ p_n]^T$ be the vector of robot poses $p_i = [p_{i_x}, p_{i_y}, p_{i_\theta}]$. The notation ${}^u p_i$ will be used to note that pose p_i belongs to partition with numeric id u, when distributed relaxation is discussed. Otherwise, the partition label will be omitted. Let δ_{ij} and Ω_{ij} be respectively the mean and the information matrix of an observation of node j seen from node i. Let $f_{ij}(\mathbf{p})$ be a function that computes a zero noise observation according to the current configuration of the nodes j and i

$$f_{ij}(\mathbf{p}) = \begin{bmatrix} (p_{j_x} - p_{i_x}) \ \cos p_{i_\theta} + (p_{j_y} - p_{i_y}) \ \sin p_{i_\theta} \\ -(p_{j_x} - p_{i_x}) \ \sin p_{i_\theta} + (p_{j_y} - p_{i_y}) \ \cos p_{i_\theta} \\ p_{j_\theta} - p_{i_\theta} \end{bmatrix} \quad (1)$$

Thus, the error on constraint $\langle i, j \rangle$ is given by

$$e_{ij}(\mathbf{p}) \quad = \quad f_{ij}(\mathbf{p}) - \delta_{ij} \quad (2)$$

Let $\mathcal{C} = \{\langle i_1, j_1 \rangle, \ldots, \langle i_M, j_M \rangle\}$ be the set of pairs of indices for which a constraint $\delta_{i_m j_m}$ exists. Then the aim of ML approach is to minimize the negative log-likelihood or error function on the observation

$$\chi^2(\mathbf{p}) \quad = \sum_{\langle i,j \rangle \in \mathcal{C}} e_{ij}^T(\mathbf{p}) \ \Omega_{ij} \ e_{ij}(\mathbf{p}) \quad (3)$$

Several numeric techniques have been proposed in order to find the minimum of $\chi^2(\mathbf{p})$. Here, we illustrate part of the relaxation algorithm proposed by Frese *et al.* [17]. The algorithm consists of two steps. First, the observation functions $f_{ij}(\mathbf{p})$ are linearized around the current value of the network configuration $\hat{\mathbf{p}}$

$$e_{ij}(\mathbf{p}) \approx f_{ij}(\hat{\mathbf{p}}) - \delta_{ij} + J_{ij}^i \Delta p_i + J_{ij}^j \Delta p_j \quad (4)$$

where J_{ij}^i and J_{ij}^j are the Jacobians of the observation function with respect to p_i and p_j evaluated in point $\hat{p}_i$ and $\hat{p}_j$, $\Delta p_i = p_i - \hat{p}_i$ and $\Delta p_j = p_j - \hat{p}_j$. In the following, the equations are written with respect to the increment variables $\Delta \mathbf{p}$. Since Eq. (1) only depends on poses i and j, there are no additional terms.

Then, the resulting negative log-likelihood $\chi^2(\mathbf{p})$ is approximated by a quadratic function [17]

$$\chi^2(\mathbf{p}) \quad \approx \quad \Delta \mathbf{p}^T \ A \ \Delta \mathbf{p} + 2 \Delta \mathbf{p}^T \ b + c \quad (5)$$

The minimum of the linearized function is easily found by solving the linear system $A \, \mathbf{p} = b$ and, thus, the value of c is not used in the computation. The method proposed in [17] to perform this final step is Gauss-Seidel relaxation. The value of each pose p_i is computed individually by solving the single block-row equation i with fixed value of p_j $(j \neq i)$. Let A_{ij} be the block of matrix A corresponding to block-row i and block-column j; let b_i be the values for block-row i.

Respectively, we have

$$A_{ij} = \sum_{\langle i,h\rangle \in \hat{\mathcal{C}}} {J_{ih}^{h}}^T \, \Omega_{ih} \, J_{ih}^{i} + \sum_{\langle h,j\rangle \in \hat{\mathcal{C}}} {J_{hj}^{h}}^T \, \Omega_{hj} \, J_{hj}^{j} \quad (6)$$

$$b_i = \sum_{\langle i,h\rangle \in \hat{\mathcal{C}}} {J_{ih}^{h}}^T \, \Omega_{ih} \, (f_{ih}(\hat{\mathbf{p}}) - \delta_{ih}) + \sum_{\langle h,i\rangle \in \hat{\mathcal{C}}} {J_{hi}^{h}}^T \, \Omega_{hi} \, (f_{hi}(\hat{\mathbf{p}}) - \delta_{hi}) \quad (7)$$

The relaxed solution of equation i at step k is

$$\Delta p_i^{(k+1)} = A_{ii}^{-1} \left(b_i - \sum_{j<i} A_{ij} \Delta p_j^{(k+1)} - \sum_{j>i} A_{ij} \Delta p_j^{(k)} \right) \quad (8)$$

The estimated value of Δp_i is determined by the neighbor poses, either already updated ($j < i$) or not ($j > i$). This procedure is performed iteratively until solution is reached with enough precision. Since A is a symmetric positive defined matrix, the convergence of the algorithm is guaranteed.

Gauss-Seidel relaxation is only the basic step of a multi-level relaxation (MLR) algorithm. MLR defines a hierarchy between nodes to solve the problem at different levels of resolution in order to speed up the convergence. However, the Gauss-Seidel algorithm can be conveniently decomposed in separate tasks that are performed independently. In the next section, we describe a parallel version of Gauss-Seidel relaxation.

B. Constraint Network Decomposition

Two properties can be noted about the structure of the ML mapping and about the Gauss-Seidel relaxation [2]:

1) *Sparsity*. The block-row update Eq. (8) of Gauss-Seidel depends on nodes p_j that are connected to the current node, i.e. the nodes with $A_{ij} \neq 0_{3\times3}$. The structure of the linearized information matrix A depends on the connectivity of the constraints network. Since the graph is built incrementally by one or more robots adding each pose in a trajectory or eventually closing loops, the resulting matrix is naturally sparse and locally connected.
2) *Variable Order*. While the order does not change the value to which the algorithm converges, it determines the dependencies among variables. In particular, at a given iteration k the updated value of pose $p_i^{(k+1)}$ depends both on already updated poses, whose index is $j < i$, and on the poses yet to be updated ($j > i$). The value of the last ones is known before starting a new iteration, so their values and the value of pose i can be computed independently. Thus, in order to compute the nodes of a cluster independently from the nodes connected to the cluster and not belonging to it, the *contour nodes* have to be computed at the end.

Therefore, the detection of clusters, which are groups of nodes internally connected and separated from each other by contour nodes, determines a convenient order. Since the nodes of the clusters do not depend on the nodes of other clusters, a single Gauss-Seidel iteration of the clusters can be executed independently with only a synchronization point given by the computation of contour nodes.

In distributed mapping, the independent clusters should correspond to the submaps built by the different robots. However, several differences arise with respect to the simple parallelization of Gauss-Seidel relaxation. First, if clusters are associated to a robot, they represent the trajectory of such robot and may not represent a local portion of the global map. Since no coordination of robots is assumed, the balance of cluster sizes and the minimization of the number of contour nodes are not guaranteed like in the case of the clustering algorithm used in [2]. Furthermore, the construction of clusters should be incremental and cluster balance should be achieved by a migration of nodes from a cluster to another one. Second, the data association of poses stored by different robots should be handled carefully in order to limit robot-to-robot communication. This data association also challenges clustering techniques because the addition of inter-robot constraints modifies the connectivity of the network (within the set of contour nodes). Third, Gauss-Seidel iterations should be performed asynchronously in each robot in order to avoid limitations to the efficiency of the solution.

IV. DISTRIBUTED MAPPING

In this section, the distributed system for multi-robot mapping is illustrated. The system consists of a *server* and of *mapping units*, one for each exploring robot. The server and the mapping units communicate using standard TCP sockets.

The server handles the contour nodes, which separate the clusters stored in different robots and represent the non parallelizable part of the joint map. Contour nodes are used as separators between pairs of nodes belonging to different submaps. The server adds a new contour node to its repository, when the initial poses of the robots are given and when new constraints between poses on different mapping units are established. Each mapping unit updates the submap using the odometry and the observations made by the local robot and performs Gauss-Seidel iteration on the local poses. Since this submap is built using only the sensor data collected by a single robot, the clusters of nodes are implicitly defined by the construction method.

In the following, the issues listed at the end of the previous section are discussed and addressed. In particular, a node cluster distribution technique to restore a minimum cut is described. Then, the problems related to inter-robot data association, i.e. the addition of a node or of a constraint between poses belonging to different mapping units, are presented and possible solutions are suggested. Finally, the Gauss-Seidel technique is applied to the distributed architecture in order to perform an asynchronous and efficient optimization of the constraint network.

A. Balanced Distributed Clustering

The poses of the constraints network are partitioned into clusters by construction, since each robot adds nodes and constraints to its submap. The robots concurrently update their local map and the submap sizes are approximatively the same, when only local updates occur. However, the inter-robot correspondences and the insertion of contour nodes break the submap size balance and scatter the spatial distribution of poses among clusters. In particular, when pose ${}^{u}p_j$ on mapping unit u is observed from pose ${}^{v}p_i$ on mapping unit $u \neq v$, constraint $\langle i, j \rangle$ is established between nodes belonging to different clusters. The distributed relaxation method described in the previous section requires that all clusters are separated by contour nodes. Thus, the following operation should be performed:

- pose ${}^{v}p_i$ is migrated to contour partition stored by server (label v is changed to contour label c) and an *alias* of ${}^{c}p_i$ is kept by cluster v;
- an alias of ${}^{c}p_i$ is also created on partition u.

Since addition of new nodes to the submap is usually sequential according to the trajectory of the robot, the migration of ${}^{v}p_i$ is likely to break the connectivity of cluster.

Therefore, a proper distribution of nodes among clusters, that balances the clusters and minimizes the number of contour nodes, is recommended for several reasons. First, such cluster balance improves the efficiency of Gauss-Seidel iteration, since contour nodes represent the part of the algorithm that cannot be executed in parallel. Second, the convergence to the correct solution is sped up if the connected parts are stored on the same map unit.

Balance could be achieved by recomputing the partions from scratch as described in [2], but the algorithm is difficult to apply to an architecture with distributed memory and requires extensive message exchange. We propose to meet these requirements by adopting a heuristic algorithm inspired by the one proposed in [18]. Each mapping unit periodically asks information about the mean cluster size and, if the size of its submap exceeds the given value, Algorithm 1 is applied to find the list of nodes suitable for migration. In the case of undersized partition, the mapping unit receives contour nodes from the server. The server performs a similar layering of the contour partition.

The proposed balancing cluster does not require a centralized and coordinated migration of nodes. For this reason, it does not achieve the optimum balance of the distributed graph like in [18], yet it avoids complete unbalance.

B. Inter-Robot Data Association

In order to obtain a joint global map from local submaps, the system must estimate the constraints between poses belonging to different submaps. Data association between two poses stored on the same robot can be performed using the standard association techniques, since the observations needed to detect the correspondences are available in the robot local memory. In the case of inter-robot association such assumption is not true and measurements should be transmitted to the mapping unit that requires such data. Furthermore, the candidates for correspondence are usually selected using e.g. Mahalanobis distance, whose value depends also from the robot making the observation. The initial poses of the robots should be known with enough precision. In the case of feature maps, a different approach based on structural subgraphs similarities [5] may be applied.

```
Data: $\mathcal{B}_u$: set of contour node aliases on this mapping unit.
Result: $\mathcal{L}$: queue of nodes ordered according to their transfer
        priority.
/* each node has fields $label$ and $deg$                    */
/* $\bar{deg}(n)$: number of adjacent nodes                  */
/*     $m$ with label $m.label < n.label$                    */
/* $\forall n, m \in \mathcal{L}, n < m$ iff                 */
/*     $n.label < m.label$ or                                */
/*     ($n.label == m.label$ and $n.deg < m.deg$)            */
$\forall n \in \mathcal{B}_u : n.label = -1$;
initialize queue $\mathcal{Q} = \mathcal{B}_u$;
while $\mathcal{Q}$ not empty do
    extract $n$ from $\mathcal{Q}$;
    foreach adjacent node $m$ of $n$ do
        if $m$ unlabeled then
            $m.label = n.label + 1$;
            $m.deg = \bar{deg}(n)$;
            insert $m$ in $\mathcal{Q}$;
            insert $m$ in $\mathcal{L}$;
        end
    end
end
```

Algorithm 1: Layering Algorithm.

In this paper, all correspondences are assumed to be known and data association problem is not addressed. Several other data association issues are not treated in this paper like the robot-to-robot correspondence and specific sensors association.

C. Asynchronous Gauss-Seidel Relaxation

When a Gauss-Seidel relaxation is applied for single robot mapping, an iteration is performed synchronously: the new value of a pose at iteration $k + 1$ is computed using the adjacent poses, whose values have been computed at current or at the previous step k as expressed by Eq. (8). The parallel version of Gauss-Seidel [2] also performs a synchronous update, although the execution is independent on each cluster. After each iteration, the problem is re-linearized according to the new values of the poses.

On a distributed architecture, the synchronous execution of a single iteration may turn quite inefficient since it would require extensive message exchange. An asynchronous optimization would limit the required bandwidth without sacrificing the effectiveness. In particular, each mapping unit solves its submap as an independent mapping problem by alternating the linearization of the constraints and a relaxation step. Each robot keeps locally the aliases of the contour nodes connected to at least one of the nodes of the submap. The values of contour poses remain unmodified in the process. When the local map has been updated, the pose values connected to contour nodes are sent to the server. The whole computation on each mapping unit is shown by Algorithm 2. An aspect not yet discussed is the use of accumulation variables a_n and b_n, which sum the terms for the solution of Eq. (8). In the algorithm, the accumulators are

```
Data: $\mathcal{G}_u = (\mathcal{N}_u, \mathcal{C}_u)$: the constraint network stored on
      mapping unit $u$, $\mathcal{B}$: set of contour node aliases on this
      mapping unit.
Result: $\mathcal{O}$: set of accumulator for contour nodes; the updated
        values of poses $\mathcal{N}_u$.
foreach iteration of relaxation do
    $\hat{\mathcal{C}} = \mathcal{C}_u \cup \{aliasesconstraint\}$;
    /* linearize constraints                                */
    foreach $< j, i > \in \hat{\mathcal{C}}$ do
        compute Jacobians $J_{ij}^i$ and $J_{ij}^j$;
        compute residual $r_{ij} = f_{ij}(\hat{\mathbf{p}}) - \delta_{ij}$;
    end
    /* solve cluster poses                                  */
    foreach $n \in \mathcal{N}$ do
        $a_n = \sum_{\langle i,n \rangle \in \hat{\mathcal{C}}} J_{in}^{n\,T}\ \Omega_{in}\ J_{in}^n$
            $+ \sum_{\langle n,j \rangle \in \hat{\mathcal{C}}} J_{nj}^{n\,T}\ \Omega_{nj}\ J_{nj}^n$;
        $b_n = \sum_{\langle i,n \rangle \in \hat{\mathcal{C}}} J_{in}^{n\,T}\ \Omega_{in}\ (r_{in} - J_{in}^i\ \Delta p_i)$
            $+ \sum_{\langle n,j \rangle \in \hat{\mathcal{C}}} J_{nj}^{n\,T}\ \Omega_{nj}\ (r_{nj} - J_{nj}^j\ \Delta p_j)$;
        $\Delta p_n = a_n^{-1}\ b_n$;
    end
end
/* accumulator for contour node aliases                     */
$\mathcal{O} = \{\}$;
foreach $n \in \mathcal{B}$ do
    $a_n = \sum_{\langle i,n \rangle \in \hat{\mathcal{C}}} J_{in}^{n\,T}\ \Omega_{in}\ J_{in}^n$
        $+ \sum_{\langle n,j \rangle \in \hat{\mathcal{C}}} J_{nj}^{n\,T}\ \Omega_{nj}\ J_{nj}^n$;
    $b_n = \sum_{\langle i,n \rangle \in \hat{\mathcal{C}}} J_{in}^{n\,T}\ \Omega_{in}\ (r_{in} - J_{in}^i\ \Delta p_i)$
        $+ \sum_{\langle n,j \rangle \in \hat{\mathcal{C}}} J_{nj}^{n\,T}\ \Omega_{nj}\ (r_{nj} - J_{nj}^j\ \Delta p_j)$;
    $\mathcal{O} = \mathcal{O} \cup \{a_n, b_n\}$
end
send accumulator set $\mathcal{O}$ to contour set;
```

Algorithm 2: Asynchronous Relaxation Algorithm.

also used to collect the contributions to be sent to the server. After receiving the updated values of map unit nodes, the server computes the value of contour nodes. The proposed solution reminds the tectonic SAM [13] that carries out local map adjustment using the so called intra-measurements. The main difference lies in the use of local reference frames for the poses on the submap.

Even though the described relaxation method is basically an offline algorithm, the map adjustment is performed after the incremental addition of new nodes and re-using the previously computed network. Expedients may be found in order to select the poses that need to be recomputed and to limit the optimization to such portion.

V. RESULTS

In this section, the proposed distributed constraints solver is validated and assessed. A preliminary version of the algorithm has been implemented and consists of two kinds of processes, a mapping unit and a server, which communicate using standard TCP sockets. The input data of each mapping unit consists of a sequence of poses and constraints representing the robot trajectory and the measurements. In order to obtain such sequences, constraint networks representing the activity of a robot on a specific path have been extracted from the commonly used dataset Intel Research Lab (INTEL) [19]. . Other sequences have been generated by simulating the exploration of an 80×20 m office-like environment (SIM) with Player-Stage. Data have been collected by 4 robots and the uncertainty on observation has been simulated by adding Gaussian noise with respectively $\sigma_{pos} = 0.02$ m and $\sigma_{pos} = 0.5°$ every 0.5 m.

The pose graphs obtained from INTEL and from SIM with and without Gauss-Seidel relaxation are shown in Figures 1 and 2. In both cases the joint map is shared among 4 mapping units whose submaps that can be distinguished by the use of different colors. The results show that the distributed algorithm is able to perform the required map adjustment. However, the size of contour nodes partition with respect to the other submaps after the exploration of the environment is significant. For example, the sizes of the four clusters are respectively 59, 58, 57 and 55, while contour partition consists of 169 nodes in the case of INTEL and similar results hold for SIM (215 contour nodes with clusters of about 80 nodes). A poor choice of the conditions activating cluster balance algorithm may be presumed. These preliminary results guarantee the correctness of the proposed approach. Further experiments, also in a real environment, are required to better assess the performance.

VI. CONCLUSION

In this paper, we proposed a distributed algorithm for map estimation based on Gauss-Seidel relaxation. The complete map is shared among independent tasks running on each robot, called *mapping units*, and a *server*. The mapping units integrate the robot measurements in local pose-graph submaps and the server stores contour nodes separating the submaps. The separation introduced by contour nodes allows the parallel execution of Gauss-Seidel relaxation on each mapping unit. The pose adjustment iterations are performed asynchronously by each task in order to make the operation more efficient. Furthermore, partition size balance possibly perturbed by node addition is restored by applying an incremental graph partitioning method.

We implemented a preliminar version of the algorithm consisting of processes that communicate with TCP sockets. Experiments have been carried out on multi-robot datasets adapted from single-robot datasets. In our future work, we expect to experimentally validate the proposed method on a real-robot setup and to address thoroughly the issue of distributed data association.

VII. ACKNOWLEDGMENTS

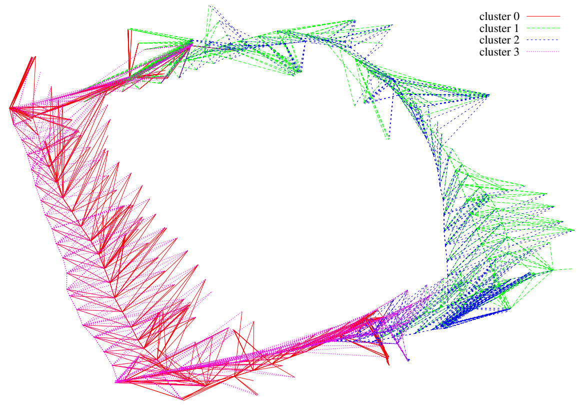
This research is partially supported by laboratory AER-TECH of Regione Emilia-Romagna, Italy.

REFERENCES

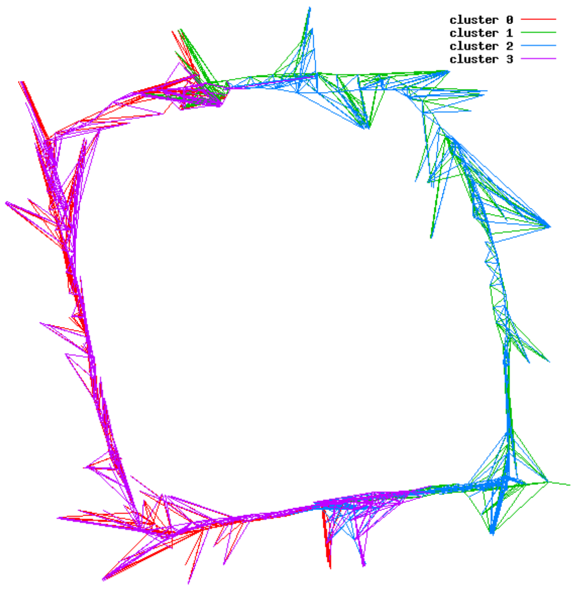
[1] E. Nerurkar, S. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[2] D. Lodi Rizzini and S. Caselli, "A parallel maximum likelihood algorithm for robot mapping," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[3] E. Nettleton, H. Durrant-Whyte, P. Gibbens, and A. Goektogan, *Sensor Fusion and Decentralized Control in Robotic Systems III*. G.T. McKee and P.S. Schenker, editors, 2000, vol. 4196, ch. Multiple platform localization and map building, pp. 337–347.

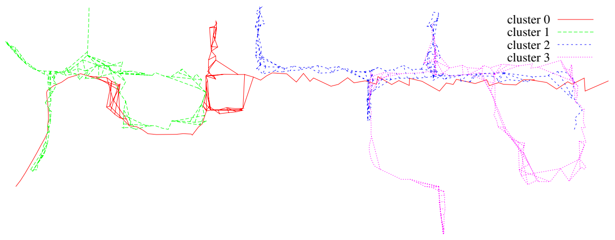


(a)

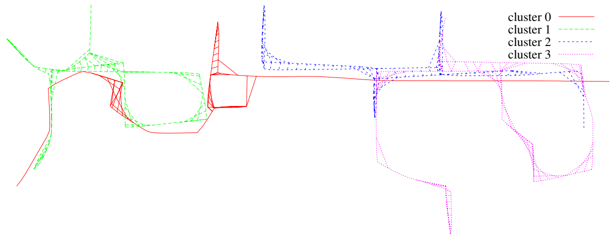


(b)

Fig. 1. Network from INTEL distributed in 4 submaps without the optimization (a) and with Gauss-Seidel relaxation (b).



(a)



(b)

Fig. 2. Simulated network SIM distributed in 4 submaps without the optimization (a) and with Gauss-Seidel relaxation (b).

[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] S. Williams, G. Dissanayake, and H. Durrant-Whyte, "Towards multi-vehicle simultaneous localization and mapping," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002, pp. 2743–2748.

[6] S. Carpin, "Merging maps via Hough transform," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[7] A. Censi and S. Carpin, "HSM3D: feature-less global 6dof scan-matching in the hough/radon domain," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[8] E. Nettleton, S. Thrun, H. Durrant-Whyte, and S. Sukkarieh, "Decentralised slam with low-bandwidth communication for teams of vehicles," in *Proc. of Int. Conf. on Field and Service Robots (FSR)*, 2004, pp. 337–347.

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

[10] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics and Automation*, vol. 21, no. 3, pp. 4376–86, June 2005.

[11] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, vol. 94, pp. 1325–1339, 2006.

[12] F. Dellaert and P. Krauthausen, "A multifrontal QR factorization approach to distributed inference applied to multi-robot localization and mapping," in *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2005, pp. 1261–1266.

[13] K. Ni, D. Steedly, and F. Dellaert, "Tectonic SAM: Exact, out-of-core, submap-based SLAM," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[14] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005, pp. 2428–2435.

[15] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Journal of Autonomous Robots*, vol. 4, pp. 333–349, 1997.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[17] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 1–12, 2005.

[18] C. Ou and S. Ranka, "Parallel incremental graph partitioning using linear programming," in *Proc. of conf. on Supercomputing*, 1994, pp. 458–467.

[19] A. Howard and N. Roy, "Radish: The robotics data set repository, standard data sets for the robotics community." [Online]. Available: http://radish.sourceforge.net/