A Balanced Distributed Graph-based Framework 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 different 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* and with respect 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 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 fully 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. RELAXATION ALGORITHM

Graph-based SLAM can be expressed according to *feature based* or *delayed-state* representations [14]. In the following, the Gauss-Seidel relaxation is applied to the solution of the map in the delayed-state form. According to this formulation the map consists only of robot poses obtained by matching observations anchored to a local frame [15] or by marginalizing feature-based maps [16].

Then map is described by the vector of robot poses $\mathbf{x} = (x_1 \ \cdots \ x_n)^T$ and by the constraints between pairs of poses. Each constraint $\langle j, i \rangle$ corresponds to the observation of robot pose x_j from another robot pose x_i. The likelihood function associated to the constraint network measures of a given network configuration $\mathbf{x}$ given the observations. Thus, the aim of ML approach is to compute the configuration that maximizes the likelihood, or equivalently minimizes the negative log-likelihood $\chi^2(\mathbf{x})$.

Several numeric techniques have been proposed in order to find the minimum of $\chi^2(\mathbf{x})$. Relaxation algorithms seem more appropriate for a distributed and asynchronous formulation. Such algorithms consist of two steps that are iteratively repeated until solution is reached with enough precision. First, the observation functions are linearized around the current estimated value of the network configuration $\mathbf{x}^{(k)}$ by computing their Jacobians. Second, the value of each pose x_i is computed individually by solving the single block-row equation i. The method proposed in [17] to perform this final step is Gauss-Seidel relaxation. In particular, the relaxed solution of equation i at step k is given by

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

where A_{ij} and b_i are the block-elements of the linearized system. The estimated value of x_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.

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 version of Gauss-Seidel relaxation suitable for a distributed implementation.

IV. BALANCED DISTRIBUTED MAPPING

The distributed system for multi-robot mapping 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 and the submap sizes are approximatively the same.

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 $v \neq v$, 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 the contour partition stored by the 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 in partition u.

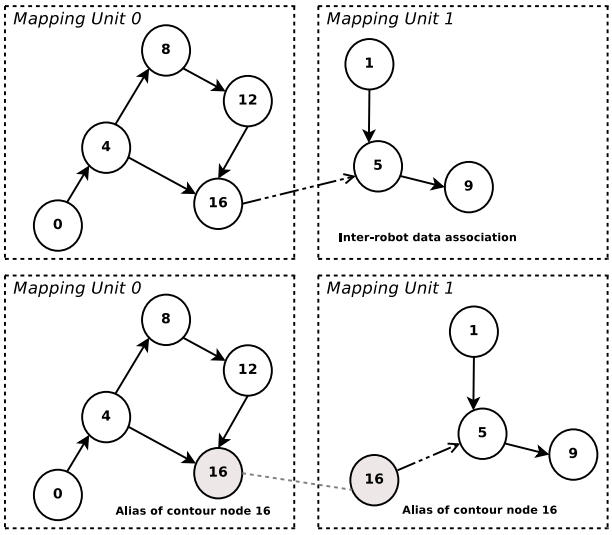

Fig. 1. Error per constraint for each submap during each experiment iteration.

The whole process is illustrated in Figure 1. 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 cluster connectivity.

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 in the same map unit.

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.

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 the heuristic algorithm illustrate by Algorithm 1 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, a layering algorithm is applied to find the list of nodes close to the border that are 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, but it starts from the alias nodes of the undersized partitions and labels the contour nodes with the numeric identifier of the partition. The migration may be performed periodically, when the mapping unit connects to the server.

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. (1). 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.

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.

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. Algorithm 2 illustrates how asynchronous relaxation works. In particular, each mapping unit u 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. In the algorithm, the accumulators are 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.

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

The proposed distributed constraints solver has been assessed using simulation data. A preliminary version of the algorithm has been implemented and consists of two kinds of processes, a mapping unit and a server. Figure 2 shows the simulated exploration of an 80×20 m office-like environment 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 $\sigma_{pos} = 0.5^\circ$ every 0.25 m. Figures 2(c)-(d) show the pose graph respectively estimated with odometry data and adjusted with Gauss-Seidel relaxation. The joint map is shared among 4 mapping units whose submaps can be distinguished by the use of different colors. The results show that the distributed algorithm is able to perform the required map adjustment. Differences with respect to groundtruth arise is the region with limited internal loop closures or inter-robot data associations.

Figure 3 shows the error per constraint associated to the single constraint network stored in each mapping unit for the experiment in Figure 2. The error has been computed also including the constraints connected to a contour node, hence not related only to the local submap. An iteration includes the independent update of the local submaps and the execution of Gauss-Seidel relaxation, i.e. first the relaxation is performed on intra-robot nodes and then the contour nodes are adjusted. Since the scale of y-axis has a limited range, we should be careful in commenting the result in Figure 3. However, the submap 3 seems affected by a larger error due to some internal inconsistencies and only after sharing nodes with other submaps the error is recovered. This interpretation is confirmed by trend of the number of nodes in each partition shown in Figure 4. Initially, the number of nodes grows linearly with the number of iterations since each mapping unit independently adds new nodes to the local map and

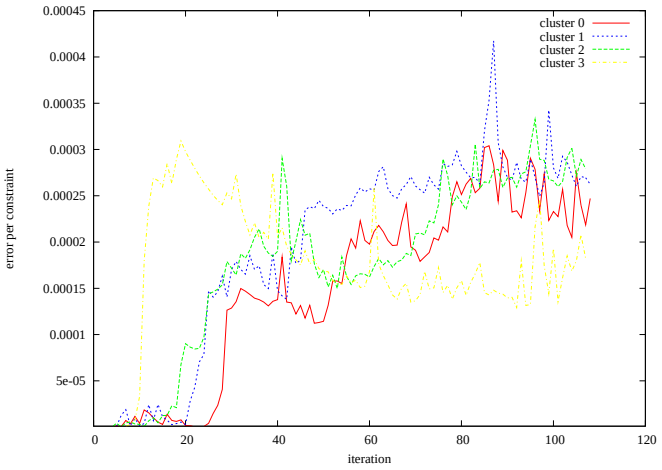

Fig. 3. Error per constraint for each submap during each experiment iteration.

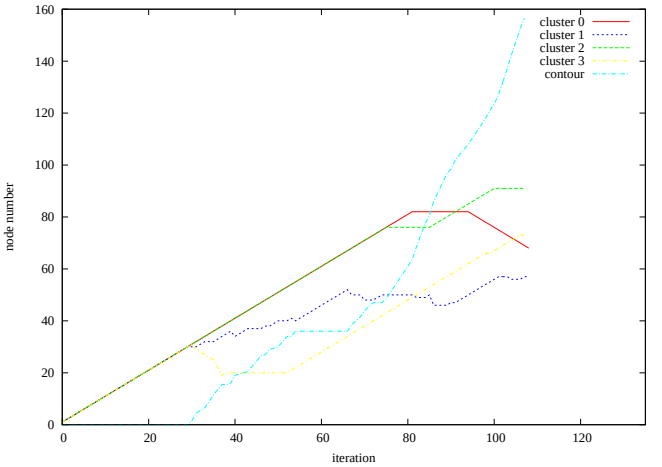

Fig. 4. Number of nodes in each submap and of contour nodes during each experiment iteration.

the number of contour nodes. When two robot trajectories intersect (around iteration 30), nodes are moved to contour partition and added to the network. With increase of contour nodes it seems that error in submap 3 tends to decrease, while the error on submap 2 seems to increase slightly.

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. Map adjustment is achieved by performing asynchronous Gauss-Seidel relaxation on each mapping unit. 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. Experiments have been carried out on multi-robot datasets

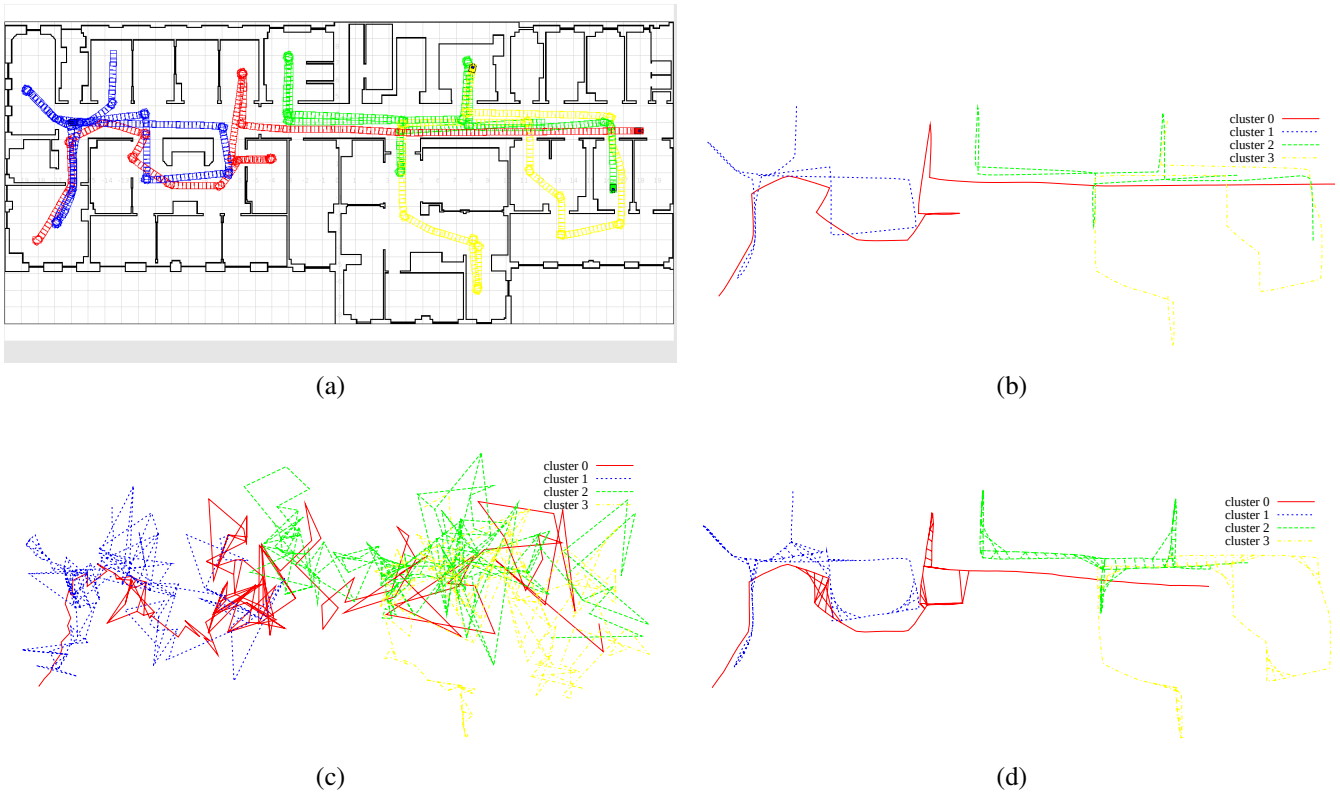

Fig. 2. A simulated environment partitioned in 4 submaps: a screenshot of the simulation (a); the true paths of the robots (b); the constraint network estimated with odometry (c) and solved with distributed relaxation (d).

adapted from single-robot datasets.

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.

[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.