Distributed Localization and Scene Reconstruction from RGB-D Data

Sergio Ayuso[1], Carlos Sagüés[1] and Rosario Aragüés[2,3]
[1]*Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, María de Luna 1 E-50018, Zaragoza, Spain*
[2]*Clermont Universite, Institut Pascal, BP 10448, F-63000, Clermont-Ferrand, France*
[3]*CNRS, UMR 6602, IP, F-63171, Aubiere, France*

Keywords: Network Robots, RGB-D Sensing, Distributed Systems, 3D Mapping and Localization.

Abstract: In this paper we present a method to make every robot of a team to compute a global 3D map of the scenarios explored by all the members, obtaining also the trajectories of the team. Every robot has a RGB-D device on board which gives RGB and depth data simultaneously and uses this information to build its own local map in real time. Once all robots have formed their local maps, they start a communication process to transform all maps to a common reference and merge them. The interest of this work is related to the establishment of the global reference and the management of the local point clouds to get correspondences between local maps which make possible to obtain the best possible transformation from the reference of every robot to the global reference.

1 INTRODUCTION

The increasing interest in multi-robot applications is motivated by the wealth of possibilities offered by teams of robots cooperatively performing collective tasks. The efficiency and robustness of these teams goes beyond what individual robots can do. In these scenarios, distributed strategies attract a high attention, especially in applications which are inherently distributed in space, time or functionality. These distributed schemas do not only reduce the completion time of the task due to the parallel operation, but also present a natural robustness to failures due to the redundancy. Our research is focused on distributed applications for perception tasks. Perception is of high importance in robotics, since almost all robotic applications require the robot team to interact with the environment. Then, if a robot is not able to obtain an environmental representation from others, or an a priori representation is not available, it must posses perception capabilities to sense its surroundings. Perception has been long studied for single robot systems and a lot of research has been carried out in the fields of localization, map building and exploration. Among the different sensors that can be used to perceive the environment, we are interested in visual perception, and the multiple benefits of using cameras have motivated the interest of many researchers. These benefits include the property that cameras are able to sense quite distant features so that the sensing is not restricted to a limited range. An additional kind of cameras of high interest are RGB-D devices (Fig. 1). They provide both regular RGB (Fig. 2, first row) and depth image information (Fig. 2, second row). Thus, it is possible to compute the landmark 3D position from a single image (Fig. 2, third row).

Figure 1: Example of a RGB-D camera.

A general overview of the achieved results, and the current and future research lines in distributed multi-robot systems can be found in (Parker, 2000). Many existent solutions for single robot perception have been extended to multi-robot scenarios under centralized schemes, full communication between the robots, or broadcasting methods. In (Fox et al., 2006) 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.

Distributed estimation methods (Olfati-Saber, 2007) maintain a joint estimate of a system that evolves with time by combining noisy observations taken by the sensor network. Early approaches sum

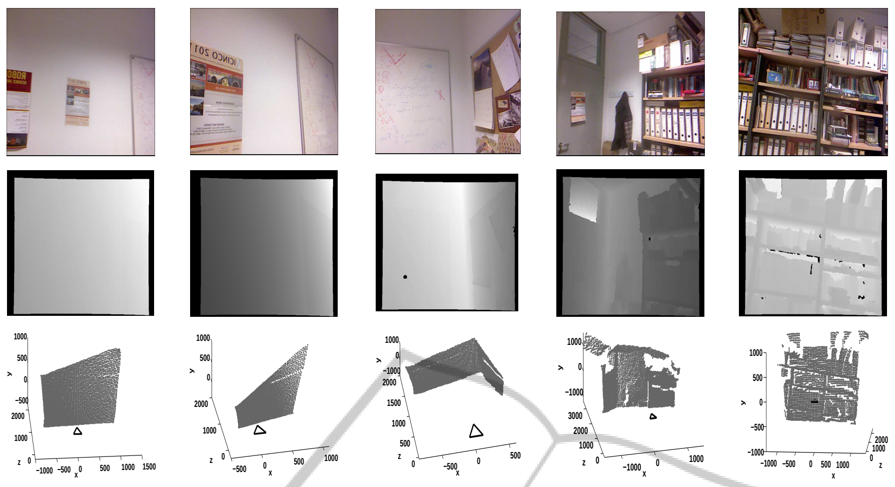

Figure 2: An example of the images obtained with the RGB-D sensor.

the measurements from the different agents in IF (Information Filter) form. If the network is complete, then the resulting estimator is equivalent to the centralized one. In general networks the problems of cyclic updates or double counting information appear when nodes sum the same piece of data more than once. Other approaches (Alriksson and Rantzer, 2006) use distributed consensus filters to average the measurements taken by the nodes. A related scenario with estimation is sensor fusion (Lynch et al., 2008; Calafiore and Abrate, 2009), where measurements acquired by several sensors are fused in a distributed fashion. Distributed perception methods must address specific challenges such as associating the elements observed by the robots in a globally consistent way, or computing the relative poses of the robots and establishing a common reference frame for the whole robot team.

The data association problem consists of establishing correspondences between different measurements or estimates of a common element. Traditional data association methods, like the Nearest Neighbor and Maximum Likelihood (Kaess and Dellaert, 2009), the Joint Compatibility Branch and Bound (JCBB) (Neira and Tardós, 2001), or the Combined Constraint Data Association (Bailey et al., 2000) are designed for single robot systems. Multi-robot approaches have not fully addressed the problem of data association. Many methods rely on broadcasting controls and observations or submaps, see e.g., (Gil et al., 2009), and solve the data association using a cycle-free order, thus essentially reducing the problem to that of the single robot scenario.

The problem of estimating the common reference frame for the team of robots is motivated by the fact that, in general, the robots start at unknown locations and do not know their relative poses. This information can be recovered by comparing their local maps and looking for overlapping regions. This approach, known as map alignment, has been deeply investigated and interesting solutions have been presented for feature-based (Thrun and Liu, 2003) and occupancy grid (Carpin, 2008) maps. However, it has the inconvenience that its results depend on the accumulated uncertainty in the local maps. Alternatively, the relative poses between the robots can be explicitly measured (Sagues et al., 2006).

In this work we propose a distributed solution to compute the 3D global map. Each robot obtains robust matches based on descriptors of the features in successive images, and it builds its 3D local map based on this information. As it may happen that the neighbor robots in communication have not common 3D maps, we search for robots in the network which have observed common regions of the 3D scene. Finally, all our robots obtain the same global map, which is composed of both individual and common regions. The robot with the most common parts with other robots is also selected as the reference robot.

2 ROBOT NETWORK

The multi-robot system is composed by N robots, each one with one RGB-D sensor to acquire the information from the scene. Additionally they can communicate with the other members of the team. The robots are labeled by $i \in \mathcal{V} = \{1, \ldots, N\}$ and each robot in the team $i \in \mathcal{V}$ can be univocally identified, for example, considering the IP addresses the robots use to communicate.

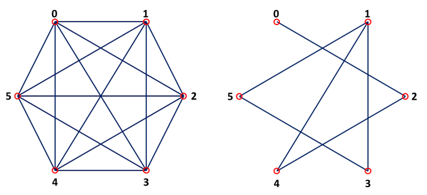

Figure 3: Two graphs with six robots. On the left is a complete communication graph and on the right an incomplete communication graph.

We assume that the limited communication capabilities imply that not all the robots will be able to directly exchange information with each other (Figure 3). These limitations can be modeled using a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ contains the pairs of robots that can directly communicate. We say that there is a communication link between i and j when they can directly exchange messages, which happens if and only if $(i, j) \in \mathcal{E}$. And we define the neighbors of a robot $i \in \mathcal{V}$ as the set of robots that can directly communicate with i,

$$\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}. \tag{1}$$

We will consider undirected communications. That is, $(i, j) \in \mathcal{E} \Leftrightarrow (j,i) \in \mathcal{E}$ and $j \in \mathcal{N}_i \Leftrightarrow i \in \mathcal{N}_j$. In this paper, we assume that the graph $\mathcal{G}$ is fixed over the time.

We formally define the consensus problem and present solutions to solve it using a distributed linear iteration. The consensus problem is formally defined as follows: Given initial conditions $x_i(0)$, $i = 1, \dots, N$, we define the consensus problem as the problem of making the state of all the robots regarding the quantity of interest reach the same value, computed as a function, f, of the initial observations: $x_i = x_j = f(x_k(0))$, $k = 1, \dots, N$, for all i and j in $\mathcal{V}$. This problem has a great importance in many robotic tasks such as sensor fusion and formation control. In the first case it is required for a proper representation of the environment whereas in the second is relevant to achieve a desired configuration, e.g., to make all the robots meet at a fixed point or explore a some region.

The solutions we are interested in correspond with distributed algorithms that follow a linear iteration scheme. Linear iterations are very easy to implement, as they only require to compute linear combinations of different quantities. This simplicity makes them very interesting to be used in a distributed setup. In addition, they represent an important class of iterative algorithms that find applications in optimization or in distributed decision making (Bullo et al., 2009). A linear iteration computes weighted sums of the different values to achieve this objective. Specifically, each robot updates its value of the quantity of interest computing a weighted sum of its previous value and that of its direct neighbors,

$$x_i(t+1) = w_{ii}(t)x_i(t) + \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t)x_j(t). \tag{2}$$

In the previous equation w_{ij} is the Metropolis weight associated to the information given by the neighbor j (Xiao et al., 2005). We refer to iteration (2) as the standard discrete time distributed consensus algorithm. The extension to quantities of the dimension of the RGB-D data acquired by the robot is straightforward, applying the same iteration rule for each one of the scalar components of the state vector independently.

If we consider the update rule of all the robots simultaneously, we can model the update as a N-dimensional discrete-time linear dynamical system with dynamics inherently related to the network structure,

$$\mathbf{x}(t+1) = \mathbf{W}(t)\mathbf{x}(t), \tag{3}$$

with $\mathbf{x}(t) = (x_1(t), \dots, x_N(t))^T$ the values of the data acquired by the robots in vectorial form and $\mathbf{W}(t) = [w_{ij}(t)] \in \mathbb{R}^{N \times N}$, the weight matrix generated using all the individual weights.

In case of using a distributed linear iteration like (2) then the algorithm is fully distributed because each robot is only using the information provided by its neighbors in the communication graph, and the robots do not need to know the topology of the whole network to execute these algorithms, they only require the information about their direct neighbors.

On the other hand, the use of this kind of iterations assume that the communications between the robots are synchronous and uncorrupted. Nevertheless, some of the most standard issues in communication can be handled by a proper modeling of the communication graph. For example, packet drops and communication failures can be seen as regular changes in the communication topology and asynchronous communications can be modeled by considering a directed communication graph. Therefore, if we design our algorithms in such a way that they can handle these topologies, we can expect them to be robust to these communication issues.

One interesting point is to assign the appropriate weights to the different elements so that different consensus objectives are achieved. In the case of averaging, we suppose that all the observations of all the robots have the same importance because the average weights all of them equally,

$$\bar{x} = \frac{1}{N} \sum_{i \in \mathcal{V}} x_i(0). \tag{4}$$

The average of the initial information of the robots can be achieved very easily using a linear iteration. We consider that the communication graph remains fixed during the execution of the whole iteration, i.e., $\mathbf{W}(t) = \mathbf{W}$ for all t in equation (3).

3 INDIVIDUAL MAPS

The camera on board of each robot gives RGB and depth information simultaneously. From them we get a point cloud having the 3D coordinates of each point and its color information. The XYZ information can be computed for each pixel $[i, j]$ of the image as:

$$\begin{aligned} X &= (i - c_i)\frac{depth(i,j)}{f} \\ Y &= (j - c_j)\frac{depth(i,j)}{f} \\ Z &= depth[i,j] \end{aligned} \tag{5}$$

being c_i y c_j the center of the image, $depth(i, j)$ the depth of the pixel and f the focal length of the camera.

The first step to obtain the local map of each robot is the search of common parts between the last point clouds captured. However the matching process between successive frames using directly the 3D information of the point cloud is not very robust. So, in a preliminary stage, SURF (Bay et al., 2006) descriptors are obtained from the 2D image of the scene in grayscale. This descriptor is based on the Haar wavelets responses and integral images in order to increase the computation speed. It turns out to be invariant to rotation, translation, scaling and changes in illumination.

Using these SURF descriptors, the process of matching is carried out in two steps. Firstly, the $k - NearestNeighbor$ (Muja and Lowe, 2009) is applied to find the most similar descriptors between the current image and the following one. The process is repeated, but this time starting with the second image and comparing it with the current one. Only the matches that appear in both directions are selected (Cross Check Filter). Although this procedure is quite robust, many outlier matches may remain, and we propose in a second stage to apply the Fundamental Matrix constraint. The 2D coordinates of two matched descriptors of two images, $\mathbf{x}_r$ and $\mathbf{x}_s$, are related with the fundamental matrix $\mathbf{F}$ as,

$$\mathbf{x}_r^T \mathbf{F} \mathbf{x}_s = 0 \tag{6}$$

To get better matches robust solutions have been proposed in the literature and we propose to use RANSAC (Fischler and Bolles, 1981) to compute the fundamental matrix. This robust method also allows to eliminate the correspondences which do not fit to the most voted solution in such a way that the resulting matches will (ideally) not have any outlier. In figure 4 we can observe the resulting good matches and the rejected matches using the RANSAC procedure.

Once good correspondences between images are given, the 3D information of these correspondences are recovered using the original point cloud. A test is carried out to check that all the depths are coherent being between 0.05 and 8 meters. Using that 3D information of the valid correspondences, RANSAC is applied again to obtain a preliminary 3D transformation between the last two frames. The figure 5 illustrates this method labeling the correspondences as *inliers* and *outliers*.

In order to obtain a more accurate transformation matrix $\mathbf{T}$ between the last two frames the ICP (*Iterative Closest Point*) (Besl and Mckay, 1992) algorithm can be computed. This matrix $\mathbf{T}$ is a $4x4$ matrix which represents the translation and rotation that has to be applied to one cloud to transform it to the reference of another cloud. Two matched points $\mathbf{x}_i \leftrightarrow \mathbf{x}_s$ are valid if they satisfy that the distance between $\mathbf{x}_r$ a $\mathbf{Tx_s}$ is smaller than a threshold. The ICP algorithm takes two point clouds $A = \{a_r\}$ y $B = \{b_s\}$ and one initial transformation $\mathbf{T}_0$ from B to A, and tries to reduce iteratively the distance between the points of A and B until the best transformation $\mathbf{T}$ that fits B with A is found. It must satisfy that the distance between corresponding points is less than some threshold. In figure 6 we show the procedure for one iteration of the algorithm.

As the point clouds obtained with the camera are very big, the point clouds used in the *ICP* algorithm are downsampled, using a 3D voxel grid with the same leaf size for all directions, in order to make the algorithm converge quickly. In figure 7 the process of downsampling a point cloud is shown.

In this step, the transformation between the last two frames (${}^i\mathbf{T}_{i-1}$) has been computed. As the reference between the previous image and the local reference is already computed (${}^{i-1}\mathbf{T}_0$), the transformation from the last image to the initial can be computed as (Figure 8),

$${}^i\mathbf{T}_0 = {}^i\mathbf{T}_{i-1} \cdot {}^{i-1}\mathbf{T}_0 \tag{7}$$

This transformation is applied to all the points in the original cloud to obtain all the points in the initial local reference of the robot. As some of these points may correspond to part of an already observed scene, they are eliminated in such a way that only new points are added to the local map. The criteria used to detect and eliminate points that have been already observed is based on the 3D Euclidean distance between points.

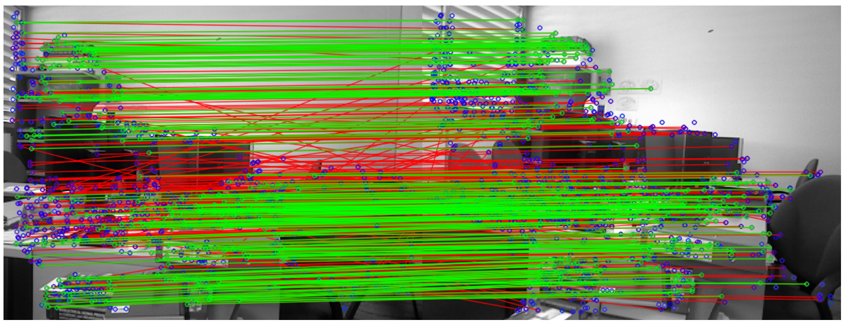

Figure 4: Matches of 2D descriptors. The green lines join good correspondences and the red lines the matches rejected with the Fundamental Matrix constraint and RANSAC.

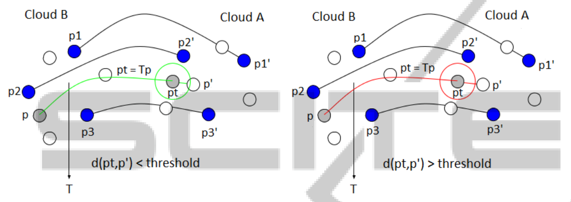

Figure 5: Checking of *outliers* using the 3D matrix transformation with the distance from $\mathbf{p}_j$ to the transformed corresponding point $\mathbf{pt}_i = \mathbf{Tp}_i$. Left: the distance from $\mathbf{p}_j$ to the transformed point $\mathbf{pt}_i$ is lower than the threshold (*inlier*). Right: the distance from $\mathbf{p}_j$ to $\mathbf{pt}_i$ is bigger than the threshold (*outlier*).

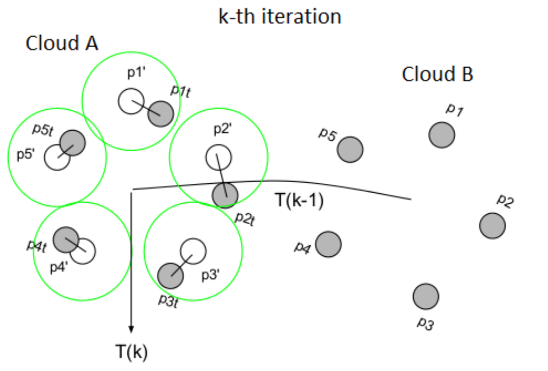

Figure 6: $K-th$ iteration of the *ICP*. The closest neighbor is found with the $\mathbf{T}_{k-1}$ of the previous cloud. From here the $\mathbf{T}_k$ is computed and the process is iteratively repeated to compute the optimal solution.

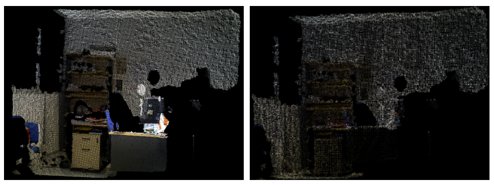

Figure 7: Original point cloud (left) and corresponding sampled point cloud (right)

The localization and orientation of the camera respect the local reference can be computed using the information provided by the transformation matrix ${}^{i}\mathbf{T}_0$. The 3D position is obtained directly from the translation vector $\mathbf{p}$ and the orientation is computed with RPY angles using the information of the rotation matrix $\mathbf{R}$ (Equation 8).

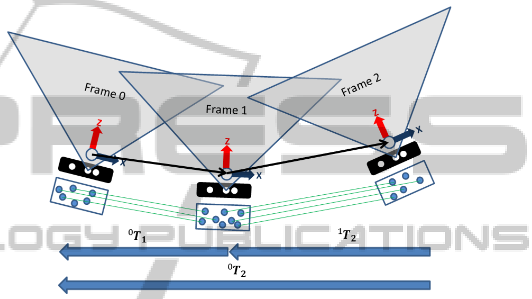

Figure 8: With transformation computed, the points of the last images are transformed and added to the local map. The points already existing in the local map are eliminated.

$$\mathbf{T} = \begin{pmatrix} \mathbf{R}_{3x3} & \mathbf{p}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{pmatrix} \quad (8)$$

If the motion of the camera between the last two poses is less than a threshold, the last frame captured is eliminated and not added to the local map.

4 DISTRIBUTED GLOBAL MAP

Once the robots have built their own local map, they start the communication process to fuse their information and create the global map. As commented in the introduction, one issue of relevance in multi-robot systems is the establishment of a common reference. Although there are several solutions to select the origin of the common reference here a particular solution is implemented.

In order to make the reading easier, we name i, j, k the robots in the network and r, s, u the point clouds of each robot. The r^{th} point cloud captured by the i^{th}

robot is denoted as c^i_r. The number of clouds captured by the i^{th} robot is m_i.

The robots are scattered forming a network with the communications described with an undirected graph $\mathcal{G}_{com} = \{\mathcal{V}_{com}, \mathcal{E}_{com}\}$. The robot i can directly exchange information with a set of neighbors $\mathcal{N}_i = \{j \in \mathcal{V}_{com} \mid (i,j) \in \mathcal{E}_{com}\}$. If robot i has to exchange information with robot j and they are not neighbors, then robot i finds the shortest path to communicate with robot j through their neighbors. The idea is that the robots try to find the robots in the network with whom they share common regions of their local maps, although they are not neighbors in the communication graph.

The communication between robots is established using the ROS library (Quigley et al., 2009), in particular the messages are send through ROS topics resulting in an unidirectional and streaming communication.

In the first stage, the SURF features extracted from the local maps are used to find 2D correspondences between the point clouds of all the robots in a distributed fashion. Given robots i and j, they look for matches between $c^i_r, r = 1..m_i$ and $c^j_s, s = 1..m_j$ establishing a ratio to represent the quality of the found matches for each pair of point clouds. Each robot stores the indexes of the point clouds r,s obtained.

One robot is selected to decide which is the robot with the most and best ratios of correspondences, which is chosen as the reference ref_g. At the same time it decides how the local maps will be combined to obtain the global map. When the local associations to be done are known, the corresponding transformation ${}^s\mathbf{T}_r$ of the local point clouds are computed with RANSAC and ICP in a similar way as the used to combine the point clouds in the local map. Additionally as the local point clouds are referred to the local reference ref_i, ref_j, this has to be considered in the computation.

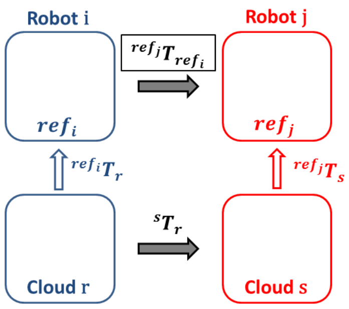

Figure 9: Transformations between the r,s point clouds of the robots i and j and the references ref_i, ref_j of those robots.

Robots have already computed the transformations of their local point clouds to their local references,

$$\begin{aligned} \mathbf{p}^{ref_i} &= {}^{ref_i}\mathbf{T}_r \cdot \mathbf{p}^r \\ \mathbf{p}^{ref_j} &= {}^{ref_j}\mathbf{T}_s \cdot \mathbf{p}^s \end{aligned} \tag{9}$$

As we want to compute ${}^{ref_j}\mathbf{T}_{ref_i}$, we use the above equations,

$$\begin{aligned} \mathbf{p}^{ref_j} &= {}^{ref_j}\mathbf{T}_s \cdot \mathbf{p}^s = {}^{ref_j}\mathbf{T}_s \cdot {}^s\mathbf{T}_r \cdot \mathbf{p}^r = \\ &= {}^{ref_j}\mathbf{T}_s \cdot {}^s\mathbf{T}_r \cdot ({}^{ref_i}\mathbf{T}_r)^{-1} \cdot \mathbf{p}^{ref_i} \end{aligned} \tag{10}$$

In such a way that

$${}^{ref_j}\mathbf{T}_{ref_i} = {}^{ref_j}\mathbf{T}_s \cdot {}^s\mathbf{T}_r \cdot ({}^{ref_i}\mathbf{T}_r)^{-1} \tag{11}$$

Finally as the robots propagate the transformations with their neighbors, all of them can transform their maps to the global reference and also they can compute the global map.

$${}^{ref_g}\mathbf{T}_{ref_i} = {}^{ref_g}\mathbf{T}_{ref_k} \cdot {}^{ref_k}\mathbf{T}_{ref_j} \cdot {}^{ref_j}\mathbf{T}_{ref_i} \tag{12}$$

which shows the situation where robot j has the transformation ${}^{ref_k}\mathbf{T}_{ref_j}$ and robot k has the transformation to the global reference ${}^{ref_g}\mathbf{T}_{ref_k}$.

In case one robot cannot find matches with any other robot, its local map will not be added to the global map, although the robot will be available for the communications of the team, if necessary.

5 REAL EXPERIMENTS

Several experiments have been carried out with real data. A RGB-D device in each robot is available to get the local information. The experiments have been accomplished in enclosed environments with different number of robots and communication graphs. Figure 10 shows the 3D maps built by a team of five robots. We also show the pose of the robots with triangles in the localization where the robot has captured a RGB-D image. Once the robots have built their own maps, they establish communication between them according to the communication graph and then they choose as the global reference, the local reference of the robot whose local map has the most and best matches with the rest of maps. In this experiment, the reference of the red robot has been selected as the global reference frame. The next step is to compute the transformation of each map to the global reference as indicated in section 4. In figure 11 we show the global map obtained by the team of five robots and their trajectories in the scene.

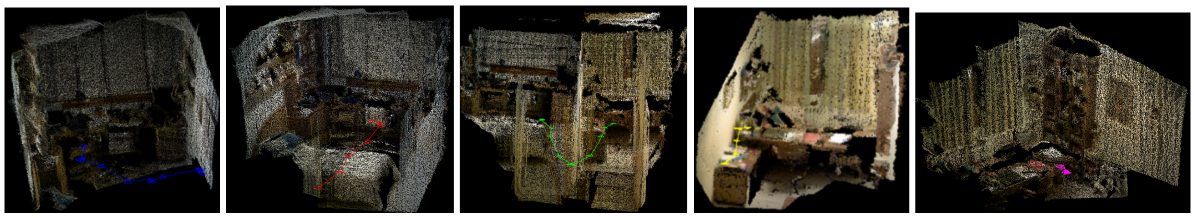
Figure 10: Local 3D map built by each robot of a team that contains the environment explored. The lines in different colors are the trajectories followed by robots, and the triangles represent the camera poses from which a RGB-D image has been captured.

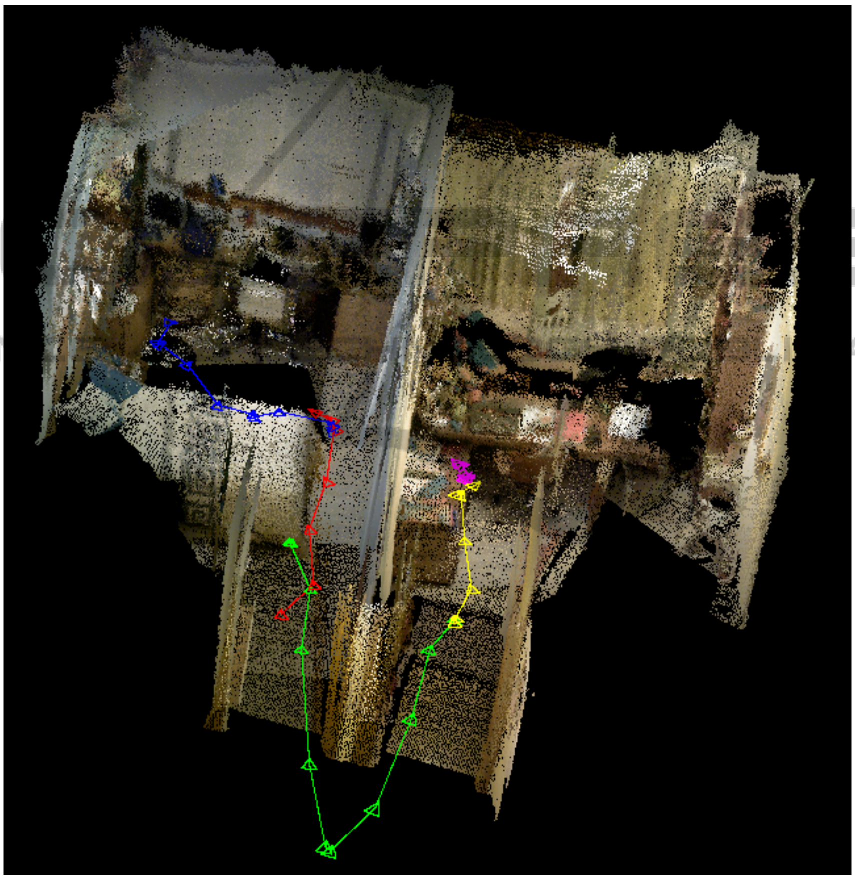
Figure 11: Global 3D map obtained by 5 robots visiting some rooms of a house. The trajectory of each robot is represented in different colors (red, blue, green, yellow and magenta).

In the implementation, each local maps has approximately 200.000 points and the global map 800.000 points. These maps are downsampled in order to decrease the memory use and the computation time of the communication process. The time needed to add a new point cloud to the local map is in average around 600 milliseconds, so the maps can be carried out in real time. The computation of the global map is in average 60 seconds and it's made once the robots have constructed their local maps.

All the experiments have been carried out using a computer with an *Intel Core i7-3630QM* processor and 6 Gb of RAM.

6 CONCLUSIONS

In this work we have presented a distributed method to make every robot in the network compute simultaneously the global map and the localization of every robot.

Our method allows each robot to build its local map and compute its trajectory in a robust and efficient way. After that, robots communicate with their neighbors and find the best global reference frame. All robots transform their maps to this global frame and finally they obtain a common global map.

The information used is RGB-D data. This makes possible to obtain accurate 3D information of the scene. The huge amount of data managed introduces the necessity of using 2D descriptors in some steps of the matching process. The real results presented show the goodness of the computed map.

ACKNOWLEDGEMENTS

This work was supported by projects Ministerio de Economía y Competitividad / Unión Europea DPI2009-08126, DPI2012-32100.

REFERENCES

Alriksson, P. and Rantzer, A. (2006). Distributed Kalman filtering using weighted averaging. In *Int. Symposium on Mathematical Theory of Networks and Systems*, Kyoto, Japan.

Bailey, T., Nebot, E. M., Rosenblatt, J. K., and Durrant-Whyte, H. (2000). Data association for mobile robot navigation: a graph theoretic approach. In *IEEE Int. Conf. on Robotics and Automation*, pages 2512–2517, San Francisco, USA.

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417.

Besl, P. J. and Mckay, H. D. (1992). A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256.

Bullo, F., Cortés, J., and Martínez, S. (2009). *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press. Electronically available at http://coordinationbook.info.

Calafiore, G. and Abrate, F. (2009). Distributed linear estimation over sensor networks. *International Journal of Control*, 82(5):868–882.

Carpin, S. (2008). Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, 25(3):305–316.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., and Stewart, B. (2006). Distributed multirobot exploration and mapping. *IEEE Proceedings*, 94(7):1325–1339.

Gil, A., Reinoso, O., Ballesta, M., and Julia, M. (2009). Multi-robot visual SLAM using a rao-blackwellized particle filter. *Robotics and Autonomous Systems*, 58(1):68–80.

Kaess, M. and Dellaert, F. (2009). Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, 57(12):1198–1210.

Lynch, K. M., Schwartz, I. B., Yang, P., and Freeman, R. A. (2008). Decentralized environmental modeling by mobile sensor networks. *IEEE Transactions on Robotics*, 24(3):710–724.

Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340.

Neira, J. and Tardós, J. D. (2001). Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897.

Olfati-Saber, R. (2007). Distributed Kalman filtering for sensor networks. In *IEEE Conf. on Decision and Control*, pages 5492–5498, New Orleans, LA.

Parker, L. E. (2000). Current state of the art in distributed robotic systems. In Parker, L. E., Bekey, G., and Barhen, J., editors, *Distributed Autonomous Robotic Systems 4*, pages 3–12. Springer Verlag.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

Sagues, C., Murillo, A. C., Guerrero, J. J., Goedemé, T., Tuytelaars, T., and Gool, L. V. (2006). Localization with omnidirectional images using the 1D radial trifocal tensor. In *IEEE Int. Conf. on Robotics and Automation*, pages 551–556, Orlando, USA.

Thrun, S. and Liu, Y. (2003). Multi-robot SLAM with sparse extended information filters. In *Int. Symposium of Robotics Research*, pages 254–266, Sienna, Italy.

Xiao, L., Boyd, S., and Lall, S. (2005). A scheme for robust distributed sensor fusion based on average consensus. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 63–70.