Distributed Data Association in Robotic Networks with Cameras and Limited Communications

Eduardo Montijano, Rosario Aragues, Carlos Sagues

***Abstract*— We address the data association problem of features observed by a robotic network. Every robot in the network has limited communication capabilities and can only exchange local matches with its neighbors. We propose a distributed algorithm that takes these local matches and, by their propagation in the network, computes global correspondences. When the algorithm finishes, each robot knows the correspondences between its features and the features of all the other robots, even if they cannot directly communicate. The presence of spurious local correspondences may produce inconsistent global correspondences, which are association paths between features observed by the same robot. The contributions of this work are the propagation of the local matches and the detection and resolution of these inconsistencies. We formally prove that after executing the algorithm, all the robots finish with a data association free of inconsistencies. We provide a fully decentralized solution to the problem, valid for any fixed communication topology and with bounded communications between the robots. Simulations and experimental results with real images show the performance of the method considering different features, matching functions and robotic applications.**

I. Introduction

The data association problem is of high interest in many robotic applications such as localization [1], mapping, exploration [2] and tracking [3]. In multi-robot systems, where the communications are limited, local matches between robots that directly communicate can be established by using existing matching methods. However, distant robots may have observed common features as well. In this paper, we address the problem of discovering these global correspondences in a distributed way when the robots perceive the world using cameras.

The problem of finding correspondences between two sets of observations has been deeply studied from different perspectives depending on the features used. Scan matching and Iterative Closest Point (ICP) [4] are popular methods for comparing two laser scans. In SLAM problems, three dimensional points with uncertainties are matched using the Nearest Neighbor [5], Maximum Likelihood [6], or B-splines [7]. Another popular method is the Joint Compatibility Branch and Bound [8], which considers the compatibility of many associations simultaneously. The Combined Constraint Data Association [9] builds a graph where the nodes are individually compatible associations and the edges relate binary compatible assignments. Switchable constraints [10] and consistent clusters [11] are considered to discard outliers in pose graph SLAM problems. Computer vision approaches match point or line features applying geometric constraints [12]–[14]. Image templates are used e.g., to recognize people [15], and bags of words [16] to recognize previously visited places. And there are many combinations of all the previous techniques with RANSAC [17] for higher robustness.

In robotic networks with multiple robots participating, these algorithms can be used in a first step to compute initial correspondences. For example, the localization of different robots [18] is done using homographies computed between pairs of robots. Multi-robot SLAM methods where the observations of all robots are broadcasted or sent to a central unit [19]–[21] associate the data following a cycle-free order, using the same methods as in single robot scenarios. None of these approaches has fully addressed the problem of multi robot data association, where additional mechanisms are required to obtain higher information than pair to pair matches.

In computer vision there are solutions to find correspondences of multiple views. Triplets of matches are considered in [22]. A multi-view matching where every pair of views is compared is presented in [23]. Then, their results are arranged in a graph where associations are propagated and inconsistencies are solved. The main limitation of these works is that the data from all the images need to be processed together, which implies a centralized scheme, communication between all the robots or broadcast.

Distributed approaches are more appealing. They present a natural robustness to individual failures of the robots because there are no central nodes of computation. Besides, they do not rely on any particular communication scheme or topology. Different distributed map merging methods are presented in [24], [25], but the data association between the robots is not treated in detail. The distributed data association in SLAM is tackled in [26], but the problem of inconsistencies is not treated. In [27] the data association of a considerably large set of images is considered. However, the method requires any image to be able to be associated with any other, which is not always possible in networks with limited communications.

In this paper, we propose a fully distributed solution for the data association problem in robotic networks with limited

E. Montijano is with Centro Universitario de la Defensa (CUD) and Instituto de Investigación en Ingeniería de Aragón (I3A), Zaragoza, Spain. emonti@unizar.es

R. Aragues is with Institut Pascal, UMR 6602 CNRS - UBP - IFMA, F-63000 Clermont-Ferrand, France, raragues@unizar.es

C. Sagues is with Departamento de Informática e Ingeniería de Sistemas - Instituto de Investigación en Ingeniería de Aragón (I3A), Zaragoza, Spain. csagues@unizar.es

This work was supported by Spanish projects Ministerio de Economía y Competitividad DPI2009-08126, DPI2012-32100, grants from the French program investissement d'avenir managed by the National Research Agency (ANR), the European Commission (Auvergne FEDER funds) and the Région Auvergne in the framework of the LabEx IMobS3 (ANR-10-LABX-16-01) and grants MEC BES-2007-14772 and AP2007-03282. The data set used in one of the experiments was provided by U. Frese and J. Kurlbaum.

communication valid for fixed topologies. Given the local matches established between neighboring robots, our algorithm allows each robot to find the correspondences with all the other robots in the network, even if they cannot communicate. Due to the presence of spurious matches, there may appear inconsistent correspondences, which are detected when chains of local matches create a path between two features observed by the same robot (Fig. 1). These situations must be correctly identified and solved, otherwise, any future merging of information will be wrong. In the paper we also provide two distributed algorithms that solve this problem taking into account the communication restrictions.

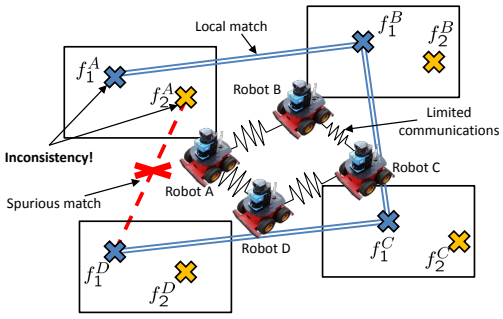


Fig. 1. A limited communication network with four robots A,B,C,D. Note that robot A communicates with robots B and D, but not with robot C. Chains of local matches correctly link features f_1^A and f_1^C of non-neighbor robots A and C. These chains also link two features of robot A, which is wrong (inconsistency). We propose methods to distributively discover associations between non neighbor robots, and to detect and solve inconsistencies.

The contributions of this paper are: (i) the propagation of the local matches, providing each robot the correspondences with all the other robots, even if they cannot directly communicate; (ii) a mechanism to detect inconsistent associations and to resolve these inconsistencies through the deletion of local matches, in function of their quality; (iii) a rigorous study of the properties of the whole procedure which proves that is fully distributed, requires a bounded amount of communication and finishes in finite time. In addition, the method makes mild assumptions on the local matching functions, and thus can be combined with a wide variety of features and local matchers and therefore, used in many different robotic scenarios. A preliminary version of this article appeared in [28]. Here, we present an improved detection algorithm which reduces the number of iterations, the complexity of the operations and the transmitted information. In this paper two different resolution methods are provided. The first one is based on distributed consensus and computes the edge with the largest error that breaks each inconsistency. The second one creates different independent spanning trees.

Along the paper we use the indices i, j and k to refer to robots and indices r, s, u to refer to features. The r^{th} feature observed by the i^{th} robot is denoted as f_r^i. Given a matrix $\mathbf{A}$, the notation $[\mathbf{A}]_{r,s}$ corresponds to the component (r, s) of the matrix, whereas $\mathbf{A}_{ij}$ denotes the block (i, j) when the matrix is defined by blocks. Similarly, $\mathbf{a}_r^i$ represents a row with the information corresponding to feature f_r^i, and $[\mathbf{a}_r^i]_s$ the s^{th} component of the row.

The remainder of the paper is arranged as follows. In Section II we give a formal definition to the problem. Section III presents our algorithm for the decentralized propagation of matches and detection of inconsistencies. In Section IV the methods to solve these inconsistencies are detailed. The results of our methods in different experiments are shown in Section V. Finally, in Section VI we present the conclusions of the work. The proofs of the theoretical results appear in the appendices.

II. Problem Description

A. Matching between two robots

As previously stated, our method consists of correctly propagating the local matches of neighboring robots through the network in a distributed fashion. In this section, we introduce the notation and describe the properties the local matcher must satisfy. Let us consider two robots i and j. Robot i (respectively j) observes a set $\mathcal{S}_i$ of m_i features, $\mathcal{S}_i = \{f_1^i, \dots, f_{m_i}^i\}$. We do not make any assumptions about the sets of features used by the robots, as they will depend on the application.

We let F be the local matching function, such that for any two sets of features, $\mathcal{S}_i$ and $\mathcal{S}_j$, $F(\mathcal{S}_i, \mathcal{S}_j)$ returns an association matrix $\mathbf{A}_{ij} \in \mathbb{N}^{m_i \times m_j}$ where

$$[\mathbf{A}_{ij}]_{r,s} = \begin{cases} 1 & \text{if } f_r^i \text{ and } f_s^j \text{ are associated,} \\ 0 & \text{otherwise,} \end{cases}$$

for $r = 1, \dots, m_i$ and $s = 1, \dots, m_j$. The function F must satisfy the following conditions.

Assumption 2.1 (Self Association): When F is applied to the same set $\mathcal{S}_i$, it returns the identity, $F(\mathcal{S}_i, \mathcal{S}_i) = \mathbf{I}$.

Assumption 2.2 (Unique Association): The association $\mathbf{A}_{ij}$ has the property that the features are matched in a one-to-one way,

$$\sum_{r=1}^{m_i} [\mathbf{A}_{ij}]_{r,s} \leq 1 \text{ and } \sum_{s=1}^{m_j} [\mathbf{A}_{ij}]_{r,s} \leq 1,$$

for all $r = 1, \dots, m_i$ and $s = 1, \dots, m_j$.

Assumption 2.3 (Symmetric Association): For any two sets $\mathcal{S}_i$ and $\mathcal{S}_j$ it holds that $F(\mathcal{S}_i, \mathcal{S}_j) = \mathbf{A}_{ij} = \mathbf{A}_{ji}^T = (F(\mathcal{S}_j, \mathcal{S}_i))^T$.

Additionally, the local matching function may give information of the quality of each association. The management of this information about quality is discussed in Section IV.

B. Centralized matching between n robots

Let us consider now the situation in which there are n robots, one of them being a leader with the n sets of features available. In this case F can be applied to all the pairs of sets of features, $\mathcal{S}_i$, $\mathcal{S}_j$, for $i, j \in \{1, \dots, n\}$. The results of all the associations can be represented by an undirected matching graph $\mathcal{G}_{cen} = (\mathcal{F}_{cen}, \mathcal{E}_{cen})$. Each node in $\mathcal{F}_{cen}$ is a feature and there is an edge between two features f_r^i, f_s^j only if $[\mathbf{A}_{ij}]_{r,s} = 1$.

For a consistent matching function, the matching graph, $\mathcal{G}_{cen}$, exclusively contains disjoint cliques, identifying features

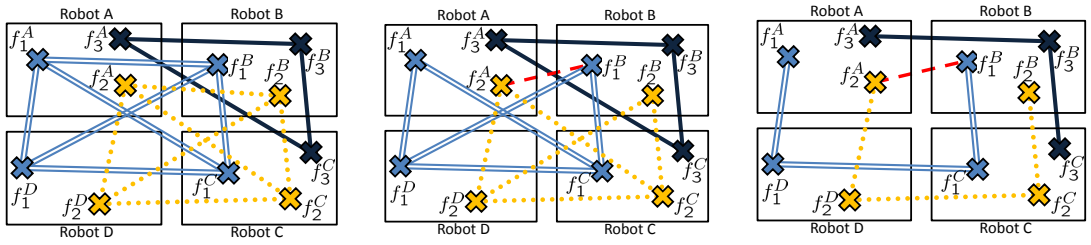


Fig. 2. Different association graphs. (a) Centralized matching with perfect association function. The graph is formed by disjoint cliques. (b) Centralized matching with imperfect association. Some edges are missed, (f_1^A, f_1^B) and (f_2^A, f_2^B), and spurious matches appear, (f_2^A, f_1^B). As a consequence, a subset of the features forms an *inconsistent set*. (c) Matching with limited communications. Now, the matches between robots A and C, and B and D, cannot be computed because they are not neighbors in $\mathcal{G}_{com}$. Moreover, the information available to each robot is just the one provided by its neighbors.

observed by multiple robots (Fig. 2 (a)). However, in real situations, the matching function will miss some matches and will consider as good correspondences some spurious matches (Fig. 2 (b)). As a consequence, inconsistent associations relating different features from the same set $\mathcal{S}_i$ may appear.

***Definition** 2.1 (**Association Sets and Inconsistencies**):* Given a matching graph, $\mathcal{G}$, an *association set* is a set of features such that they form a connected component in $\mathcal{G}$. Such set is an *inconsistent set* or an *inconsistent association* if there exists a path in $\mathcal{G}$ between two or more features observed by the same robot. A feature is *inconsistent* if it belongs to an inconsistent association.

In computer vision, centralized solutions to overcome this problem are found in [23], [27]. The latter one is also well suited for a distributed implementation but yet requires that any pair of images can be matched. In robotic networks this implies global communications, which are not always possible.

C. Matching between n robots with limited communications

Let us consider now that there is no leader with the information of the n robots. Instead of that, the robots are scattered forming a network with limited communications described with an undirected graph $\mathcal{G}_{com} = (\mathcal{V}_{com}, \mathcal{E}_{com})$. The nodes in the graph are the robots, $\mathcal{V}_{com} = \{1, \dots, n\}$. If two robots i, j can exchange information then there is an edge between them, $(i, j) \in \mathcal{E}_{com}$. Let $\mathcal{N}_i$ be the set of neighbors of robot i, $\mathcal{N}_i = \{j \mid (i, j) \in \mathcal{E}_{com}\}$.

In this case, due to communication restrictions, local matches can only be found within direct neighbors. As a consequence, the matching graph computed in this situation is a subgraph of the centralized one, $\mathcal{G}_{dis} = (\mathcal{F}_{dis}, \mathcal{E}_{dis}) \subseteq \mathcal{G}_{cen}$, (Fig. 2 (c)). It has the same set of nodes, $\mathcal{F}_{dis} = \mathcal{F}_{cen}$, but it has an edge between two features f_r^i, f_s^j only if the edge exists in $\mathcal{G}_{cen}$ and the robots i and j are neighbors in the communication graph,

$$\mathcal{E}_{dis} = \{(f_r^i, f_s^j) \mid (f_r^i, f_s^j) \in \mathcal{E}_{cen} \wedge (i, j) \in \mathcal{E}_{com}\}.$$

Along the paper, we name m_{sum} the number of features, $|\mathcal{F}_{dis}| = \sum_{i=1}^{n} m_i = m_{sum}$. We name d_f the diameter of $\mathcal{G}_{dis}$, the length of the longest path between any two nodes in $\mathcal{G}_{dis}$, and we name d_v the diameter of the communication graph, $\mathcal{G}_{com}$. The diameters satisfy $d_f \leq m_{sum}$ and $d_v \leq n$.

We name $\mathbf{A} \in \mathbb{N}^{m_{sum} \times m_{sum}}$ the adjacency matrix of $\mathcal{G}_{dis}$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \dots & \mathbf{A}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{n1} & \dots & \mathbf{A}_{nn} \end{bmatrix}, \text{ where} \tag{1}$$

$$\mathbf{A}_{ij} = \begin{cases} F(\mathcal{S}_i, \mathcal{S}_j) & \text{if } j \in \{\mathcal{N}_i \cup i\}, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{2}$$

Let us note that in this case none of the robots has the information of the whole matrix. Robot i has only available the sub-matrix corresponding to its own local matches $\mathbf{A}_{ij}, j = 1, \dots, n$. Under these circumstances the problem is formulated as follows: Given a robotic network with communications defined by a graph, $\mathcal{G}_{com}$, and an association matrix $\mathbf{A}$ scattered over the network, find the global matches considering the communication restrictions. In case there are inconsistencies, find them and solve them, also in a distributed way.

III. Propagation of Matches and Detection of Inconsistencies

Considering Definition 2.1 we observe that in order to match features of robots that are not neighbors and to detect any inconsistency between features, the paths that exist among the elements in $\mathcal{G}_{dis}$ should be computed. As the following lemma states [29], given a graph $\mathcal{G}$, the powers of its adjacency matrix contain the information about the number of paths existing between the different nodes of $\mathcal{G}$:

***Lemma** 3.1 (Lemma 1.32 [29]):* Let $\mathcal{G}$ be a weighted graph of order $|\mathcal{V}|$ with un-weighted adjacency matrix $\mathbf{A} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$, and possibly with self loops. For all $r, s \in \{1, \dots, |\mathcal{V}|\}$ and $t \in \mathbb{N}$ the (r, s) entry of the t^{th} power of $\mathbf{A}$, $[\mathbf{A}^t]_{r,s}$, equals the number of paths of length t (including paths with self loops) from node r to node s.

Then, if we compute the powers of $\mathbf{A}$ we can detect the different association sets by evaluating whether the number of paths is greater than zero or not.

These powers can be computed in a distributed way by creating local variables, $\mathbf{X}_{ij} \in \mathbb{N}^{m_i \times m_j}$, of the same dimension as the matrices $\mathbf{A}_{ij}$, initialized by

$$\mathbf{X}_{ij}(0) = \begin{cases} \mathbf{I}, & j = i, \\ \mathbf{0}, & j \neq i. \end{cases}$$

Denoting by $\mathbf{X}$ the whole matrix, as in eq. (1), the initial value is the identity matrix, $\mathbf{X}(0) = \mathbf{I} \in \mathbb{N}^{m_{sum} \times m_{sum}}$. After that, at each iteration the robots exchange the local matrices with their neighbors and update them using the local association matrices, $\mathbf{A}_{ij}$,

$$\mathbf{X}_{ik}(t+1) = \sum_{j \in \mathcal{N}_i \cup i} \mathbf{A}_{ij} \mathbf{X}_{jk}(t).$$

Noting that $\mathbf{A}_{ij}$ is zero for those robots that are not neighbors in the communication graph, it is observed that the method is equivalent to

$$\mathbf{X}(t+1) = \mathbf{A}\mathbf{X}(t),$$

and therefore, it achieves the desired result in a fully distributed manner, $\mathbf{X}(t) = \mathbf{A}^t$. Although this method is simple and effective, it requires big communications between the robots, because they need to send all the matrices $\mathbf{X}_{ik}(t)$ at each iteration to all their neighbors. In this paper we propose an improved algorithm that reduces the complexity of the operations, the amount of transmitted information and the number of execution steps.

Algorithm 1 shows the proposed method. Each robot initializes m_i vectors of dimension m_{sum}. We denote by $\mathbf{y}_r^i$ the r^{th} vector handled by robot i, which is related with the associations of feature f_r^i, and we initialize it using the local matches of this feature. The components are either 1, if f_r^i and f_s^j are locally associated, or 0 otherwise,

$$\mathbf{y}_r^i(0) = \{[\mathbf{A}_{i1}]_{r,1}, \ldots, [\mathbf{A}_{in}]_{r,m_n}\}.$$

Instead of computing the number of paths between two features, the idea is to compute if there exists a path or not. This is done by using logical values and logical "*or*" operations rather than integers, sums and products. At each iteration, t, the robots exchange the vectors $\mathbf{y}_r^i(t)$ with their neighbors (lines 5-6 of Algorithm 1). If two features, f_r^i and f_s^j, of neighbor robots are matched, then, by the transitivity of the associations, they are also matched with all the other features associated to them,

$$\mathbf{y}_r^i(t+1) = \mathbf{y}_r^i(t) \vee \mathbf{y}_s^j(t),$$

(lines 8-10 of Algorithm 1). The "*or*" operation implies that the components equal to one in $\mathbf{y}_s^j(t)$ are transmitted to $\mathbf{y}_r^i(t+1)$, without modifying those components that were already positive. Additionally, the robots speed up the process using their own vectors to find more associations. If f_r^i and $f_{r'}^i$ are both associated to any other feature f_s^j, where j does not need to be a direct neighbor of i, then they must be associated to all the other features associated to both f_r^i and $f_{r'}^i$, (lines 11-14 of Algorithm 1). Finally, the stop criterion is triggered when all the vectors of robot i have not changed any component at some iteration, as it means they will not change anymore because all the paths between associated features have been found (line 16 of Algorithm 1).

The advantages of using this Algorithm rather than computing the power matrices of $\mathbf{A}$ are the following. Firstly, the use of logical values allows us to avoid the large numbers that may appear when computing powers of the adjacency matrix. Secondly, the number of iterations is reduced by considering that, when two or more of the features observed by the same robot share a common third feature observed by a different robot, then eventually they will be associated with each other. Lastly, since we assume that all the robots know m_{sum}, instead of exchanging the whole vectors at each iteration, the robots only need to send to their neighbors, the indices of those components that have changed from zero to one in the previous iteration, also reducing the amount of communications. In the following we formalize the main properties of the algorithm. The proofs of the results are in Appendix I.

Algorithm 1 Propagation of matches - Robot i

Require: $\mathcal{S}_i$ and $\mathbf{A}_{ij}$
Ensure: All the global matches and inconsistencies are found
1: *– Initialize m_i vectors*
2: $\mathbf{y}_r^i(0) = \{[\mathbf{A}_{i1}]_{r,1}, \ldots, [\mathbf{A}_{in}]_{r,m_n}\}$
3: *– Propagation of the matches*
4: **repeat**
5: Send $\mathbf{y}_r^i(t)$ to the neighbors
6: Receive $\mathbf{y}_s^j(t)$ from the neighbors
7: **for all** $r, r' \in \mathcal{S}_i$ **do**
8: **if** f_r^i is associated to f_s^j ($[\mathbf{A}_{ij}]_{r,s} = 1$) **then**
9: $\mathbf{y}_r^i(t+1) = \mathbf{y}_r^i(t) \vee \mathbf{y}_s^j(t)$
10: **end if**
11: **if** f_r^i and $f_{r'}^i$ are both associated to any f_s^j,
12: ($[\mathbf{y}_r^i]_s(t) = [\mathbf{y}_{r'}^i]_s(t) = 1$, for some s) **then**
13: $\mathbf{y}_r^i(t+1) = \mathbf{y}_r^i(t) \vee \mathbf{y}_{r'}^i(t)$
14: **end if**
15: **end for**
16: **until** $\mathbf{y}_r^i(t+1) = \mathbf{y}_r^i(t)$, $\forall r \in \mathcal{S}_i$

Proposition 3.2 (Limited Communications): The amount of information exchanged by the network during the whole execution of Algorithm 1 can be upper bounded by $2m_{sum}^2$.

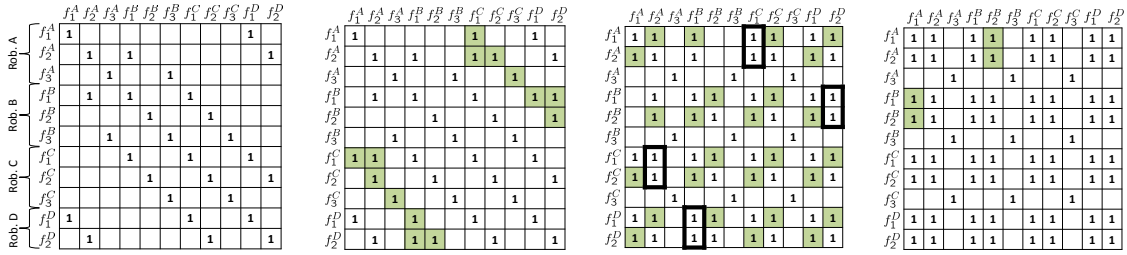
Proposition 3.3 (Correctness): After executing Algorithm 1 all the paths between features have been found and they are available to all the robots with features involved in them.

Theorem 3.4 (Limited Iterations): All the robots end the execution of the Algorithm 1 in at most $\min(d_f, 2n)$ iterations.

Remark 3.5 (Conservative Bounds): The bounds provided in Proposition 3.2 and Theorem 3.4 are conservative. In practice we should expect a better performance of the algorithm. In order to send $2m_{sum}^2$ data, it is required for the association graph to be strongly connected, i.e., for any pair of features there is a path of arbitrary length connecting them. This situation is unlikely to happen, since it would mean that all the features are associated with each other. In the experiments we empirically show that the actual iterations and communications are smaller than these bounds.

After the execution of the algorithm, every robot is able to detect all the features associated with its own ones. The robots are also able to extract from their own rows $\mathbf{y}_r^i(t)$ all the information of any inconsistency that involves any of its features. The feature f_r^i will be associated to all the features, s, such that $[\mathbf{y}_r^i(t)]_s = 1$.

The detection of the inconsistencies is done using two rules. A feature f_r^i is inconsistent if and only if one of the following conditions is satisfied:



(a) (b) (c) (d)

Fig. 3. Execution of the propagation algorithm to detect all the matches in Fig. 2 (c). A detailed explanation can be found in section III-A.

- There exists another feature observed by robot i, $f^i_{r'}$, with $r \neq r'$, such that $[\mathbf{y}^i_r(t)]_{r'} = 1$;
- There exist two features observed by other robot j, f^j_s and $f^j_{s'}, s \neq s'$, such that $[\mathbf{y}^i_r(t)]_s = 1$ and $[\mathbf{y}^i_r(t)]_{s'} = 1$.

In the first case, the robot knows that two or more of the features it has observed are associated, which cannot be possible because they are different features. In the second case the robot detects that there is another robot which has two or more features associated. If there is an inconsistent feature, the robots also know the rest of features that belong to the inconsistent set independently of who observed such features. Additionally, note that the propagation may find associations between neighboring robots that the local matcher did not find, provided that an alternative path of local matches associates them.

A. Example of execution

Figure 3 shows an example of how to use the algorithm. The example shows the propagation of the local associations shown in Fig. 2 (c). Each robot has only the information about the rows corresponding to the features it has observed. Fig. 3 (a) shows the matrix with the local matches of all the robots. The zeros have been omitted in the figure for a better representation. For simplicity here we will only explain the process for the robot A.

Initially, robot A instantiates three vectors of 11 components (the total number of features), corresponding to the three first rows of the matrix in Fig. 3. Looking at the first row, which is related to feature f^A_1, we observe that it has a value equal to 1 in the element corresponding to feature f^D_1, which is the only feature locally associated to it (besides the feature f^A_1 itself). In a similar way, the second row has 3 components equal to 1 and the third row 2 components. This implies that in the first communication round, robot A sends a total of $2 \cdot 7 = 14$ integers (56 bytes using standard codification) to its neighbors (robots B and D), specifying the components that have changed from zero to one, row and column.

After these communications and the execution of lines 8-10 of Algorithm 1 the rows have the form of Fig. 3 (b). The components with green background are the new associations found by the algorithm. For the case of the robot A, the first feature, f^A_1, is matched with the first feature of robot D, which is a direct neighbor of A, thus, $\mathbf{y}^A_1(2) = \mathbf{y}^A_1(1) \vee \mathbf{y}^D_1(1)$. The second feature, f^A_2, is matched with f^B_1 and f^D_2 so $\mathbf{y}^A_2(2) = \mathbf{y}^A_2(1) \vee \mathbf{y}^B_1(1) \vee \mathbf{y}^D_2(1)$. Finally $\mathbf{y}^A_3(2) = \mathbf{y}^A_3(1) \vee \mathbf{y}^B_3(1)$. After that, robot A detects that f^A_1 and f^A_2 share a common match with f^C_1. Therefore it executes lines 11-14 of the algorithm with these two features, as shown in Fig. 3 (c). The total number of associations found in these two steps (number of components with green background in the three rows handled by robot A in Fig. 3 (b) and (c)) is equal to 10, which means that the next message of robot A to robots B and D will be of size $2 \cdot 10 = 20$ integers (80 bytes).

In the next iteration (Fig. 3 (d)), robot A finds two more associations, therefore sending 4 integers (16 bytes), and after that the process is finished because there are no more matched features. The algorithm has found all the associations using 3 iterations and robot A has needed to communicate $56 + 80 + 16 = 152$ bytes, which certainly is much less than the information exchanged in the first phase to find the local matches. Finally, robot A knows that f^A_1 and f^A_2 belong to one inconsistency with features $f^B_1, f^B_2, f^C_1, f^C_2, f^D_1$ and f^D_2.

IV. Decentralized Resolution of Inconsistent Associations

Up to now, the robots know all the association matches affecting their features and which features are inconsistent. If there are not inconsistent features after the propagation, then the association is already globally consistent and the robots do not need to do anything else. Otherwise they need to solve the inconsistencies.

The existence of one inconsistency implies the existence of at least one spurious match in the local matching process. The problem of resolving one inconsistency consists in partitioning the association graph so that the features belonging to the same robot are disconnected after the partition, ideally deleting the spurious matches. Let us note that, even with the knowledge of the whole association graph, it is impossible to discern with 100% of confidence which local matches are spurious and which ones are good matches. Moreover, the problem of finding the best partition of the graph, according to some arbitrary metric, is NP-Hard because all the possible partitions should be analyzed.

The only thing that we know for sure is that in order to have a correct data association it is required for the association graph to be free of inconsistencies. Therefore, in this paper we carry out the resolution of inconsistent associations by deleting edges from $\mathcal{G}_{dis}$, so that the resulting graph is inconsistency-free.

Definition 4.1 (Inconsistency-Free Graph): Let N_C denote the number of inconsistent sets in $\mathcal{G}_{dis}$. The robots that detect an inconsistent set $\mathcal{C}$ are $\mathcal{V} \subseteq \mathcal{V}_{com}$. The number of

features from each robot $i \in \mathcal{V}$ involved in $\mathcal{C}$ is $\tilde{m}_i$ and the number of total features involved in $\mathcal{C}$ is denoted as c. We say $\mathcal{G}_{dis}$ is *inconsistency-free* if $N_C = 0$.

All the edges whose deletion transforms $\mathcal{G}_{dis}$ into an inconsistency-free graph, belong to any of the N_C inconsistent sets of $\mathcal{G}_{dis}$. Since the inconsistent sets are disjoint, they can be considered separately. From now on, we focus on the resolution of one of the inconsistent sets $\mathcal{C}$. The other inconsistent sets are managed in parallel in the same way.

In the rest of the section we provide two different distributed algorithms to transform $\mathcal{G}_{dis}$ into an inconsistency-free association graph. The first one, the *Maximum Error Cut*, considers the weights in the association graph in order to find the edge with the largest error that breaks a given inconsistency. The second method is based on a greedy deletion of edges to construct different *Spanning Trees* free of inconsistencies. The analysis about the goodness of the two methods is done in section V.

A. Resolution using the Maximum Error Cut

Most of the matching functions in the literature are based on errors between the matches, e.g., the Sampson distance or the Mahalanobis distance. Therefore, we can use these errors to find an inconsistency-free partition of $\mathcal{C}$, with good chances of discarding the outlier matches.

Let $\mathbf{E}$ be the weighted association matrix

$$[\mathbf{E}]_{r,s} = \begin{cases} e_{rs} & \text{if } [\mathbf{A}]_{r,s} = 1, \\ -1 & \text{otherwise,} \end{cases} \tag{3}$$

with e_{rs} the error of the match between r and s.

Assumption 4.1 (Properties of the Errors): The error between matches satisfies:

- $e_{rr} = 0, \forall r$;
- Errors are non negative, $e_{rs} \geq 0, \forall r, s$;
- Errors are symmetric, $e_{rs} = e_{sr}, \forall r, s$;
- Errors of different matches are different, $e_{rs} = e_{r's'} \Leftrightarrow [r = r' \wedge s = s'] \vee [r = s' \wedge s = r']$.

Note that this Assumption is, in general, fulfilled using any of the above mentioned matching functions. The distance of any feature with respect to itself is generally equal to zero and with respect to other feature it usually has a positive value. The last property is the only one that might sometimes be violated and for that reason we will discuss what happens when it is not fulfilled later in the section.

Since the inconsistency is already known there is no need to use the whole matrix, $\mathbf{E}$, but just the sub-matrix related with the inconsistency, $\mathbf{E}_{\mathcal{C}}$.

Definition 4.2 (Bridges and Cuts): In an inconsistent set, a *bridge* is an edge whose deletion divides the set in two connected components, i.e., it does not belong to a cycle. Given two inconsistent features, we define a *cut* as a bridge that, if it is deleted, it solves the inconsistency between the features.

Note that not all the bridges in one inconsistency are cuts. There are bridges that, if deleted, do not break the inconsistency because they do not belong to the path between the features to separate. Our goal is, for each pair of inconsistent features, find and delete the cut with the maximum error. We justify this deletion by assuming that the local matcher will work similarly for all the pairs of features in terms of quality, and that this quality will be in general high, as happens with methods such as JCBB or the epipolar constraint. Therefore, it seems like a good idea to respect as many local matches as possible instead of removing a large set of links, and specially to respect cycles in the association graph, as they represent strong association sets with high probability of being correct. Also, if we have several cuts to break one inconsistency, it is normal to assume that the spurious match is the one that in the local matching has the largest error. All these factors are considered by our algorithm with the intention of finding the correct inconsistency-free partition of the association graph.

Algorithm 2 shows the distributed solution that we propose to find the cuts using local interactions. In the following we give a detailed explanation of the whole process.

Algorithm 2 *Maximum Error Cut* - Robot i

Require: An inconsistent set $\mathcal{C}$ and the error matrix $\mathbf{E}_{\mathcal{C}}$

1: *– Initialize $\tilde{m}_i$ vectors*
2: $\mathbf{z}_r(0) = \{[\mathbf{E}_{\mathcal{C}}]_{r,1}, \ldots, [\mathbf{E}_{\mathcal{C}}]_{r,c}\}$
3: *– Propagation of the errors*
4: **repeat**
5: Send $\mathbf{z}_r(t)$ to the neighbors
6: Receive all $\mathbf{z}_s(t)$, from the neighbors
7: **for all** $r \in \mathcal{C}$ and $\mathcal{S}_i$ **do**
8: **if** f_r is associated to f_s ($[\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0$) **then**
9: $\mathbf{z}_r(t+1) = \max(\mathbf{z}_r(t), \mathbf{z}_s(t)\mathbf{P}_{rs})$
10: **end if**
11: **end for**
12: **until** $\mathbf{z}_r(t+1) = \mathbf{z}_r(t), \forall r$
13: *– Edge Deletion*
14: **while** robot i has inconsistent features r and r' **do**
15: Find the cuts $(s, s'), s \neq s'$, such that:
16: (a) $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'}$,
17: (b) For all $u \neq s$, $[\mathbf{z}_r]_s \neq [\mathbf{z}_r]_u$,
18: (c) For all $u \neq s'$, $[\mathbf{z}_{r'}]_{s'} \neq [\mathbf{z}_{r'}]_u$
19: Select the edge with largest error
20: Send message to cut it
21: **end while**

Similarly to the propagation step, in line 2 of the algorithm each robot initializes a set of $\tilde{m}_i$ vectors, as many as features in $\mathcal{S}_i$ belonging to the inconsistency, each one with dimension equal to the number of features in the inconsistency. Let $\mathbf{z}_r$ denote the vector associated to the r^{th} feature in the inconsistency, which without loss of generality we assume that belongs to robot i. We drop the super indices corresponding to robots because in this case we are interested in the global position of feature r in the inconsistency, without mattering which is the robot that is handling the vector related to this feature. Let us note that this value is available and known to the robots from the propagation stage. Each component of the vector is initialized by $[\mathbf{z}_r(0)]_s = [\mathbf{E}_{\mathcal{C}}]_{r,s}$ which recall that is equal to e_{rs} if features r and s are locally associated and -1 otherwise.

As in the propagation method, the robots exchange their

initial vectors with their neighbors. The robots update the value of the vectors $\mathbf{z}_r$ using the information of all the received vectors, $\mathbf{z}_s$, such that features r and s are associated, i.e., $[\mathbf{E}_\mathcal{C}]_{r,s} \geq 0$. The limited communications are implicit in the errors $[\mathbf{E}_\mathcal{C}]_{r,s}$, which are different than -1 only if the robots that have observed features r and s are direct neighbors in the communication graph. Then, the u^{th} component of $\mathbf{z}_r(t)$ is updated by

$$[\mathbf{z}_r(t+1)]_u = \begin{cases} \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_s) \text{ if } u = r \\ \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_r) \text{ if } u = s \\ \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_u) \text{ if } r \neq u \neq s. \end{cases} \quad (4)$$

The update rule is very similar to the "*or*" operation executed during the propagation of the matches. For all those elements that are not r or s, the update rule selects the maximum value between the two vectors, as the "*or*" operation does. The difference relies in the elements r and s, which consider the maximum of the other component of the received vectors, i.e., $[\mathbf{z}_r(t+1)]_r$ is updated taking the maximum of $[\mathbf{z}_r(t)]_r$ and $[\mathbf{z}_s(t)]_s$ and $[\mathbf{z}_r(t+1)]_s$ is updated taking the maximum of $[\mathbf{z}_r(t)]_s$ and $[\mathbf{z}_s(t)]_r$.

Update rule (4) can also be put in vectorial form

$$\mathbf{z}_r(t+1) = \max(\mathbf{z}_r(t), \mathbf{z}_s(t)\mathbf{P}_{rs}), \quad (5)$$

where the maximum is done element-wise and $\mathbf{P}_{rs}$ is the permutation matrix of the columns r and s. Basically, what the rule does is to transmit the errors over the network, in such a way that $[\mathbf{z}_r(t)]_u$ has the value of the error of the last link in the path that connects f_r with f_u. The rule is executed until $\mathbf{z}_r(t+1)$ remains equal to $\mathbf{z}_r(t)$ for all the vectors handled by each robot, as shown in lines 3-12 of Algorithm 2.

The following results state the main properties of this update rule. We demonstrate that the presented method converges in finite time. We also show the convergence values of the different elements. For clarity, we separate the analysis in two parts: the first result gives the values reached by the components that belong to bridges in the graph; the second result consider the features that form part of a cycle in the association graph. The proofs of these results appear in Appendix II.

Proposition 4.3 (Convergence): The dynamic system defined in (5) converges in a finite number of iterations and for any $r, s \in \mathcal{C}$ such that $[\mathbf{E}_\mathcal{C}]_{r,s} \geq 0$ the final value of $\mathbf{z}_r$ is the same as $\mathbf{z}_s\mathbf{P}_{rs}$. In addition, for any $r \in \mathcal{C}$, $[\mathbf{z}_r(t)]_r = 0, \forall t \geq 0$.

Theorem 4.4 (Values for Bridges): Let us consider one bridge, (s, u). Let $d(r, s)$ be the minimal distance in edges to reach node s starting from node r, then for all r such that $d(r, s) < d(r, u)$,

$$[\mathbf{z}_r(t)]_u \to [\mathbf{E}_\mathcal{C}]_{s,u} = e_{su}. \quad (6)$$

Equivalently, for all r such that $d(r, s) > d(r, u)$, $[\mathbf{z}_r(t)]_s \to [\mathbf{E}_\mathcal{C}]_{u,s} = e_{us} = e_{su}$.

Theorem 4.5 (Values for Cycles): Let us suppose the inconsistency has a cycle involving ℓ features. Let $\mathcal{C}_\ell$ be the subset of features that belong to the cycle. For any feature r

$$[\mathbf{z}_r(t)]_s \to \max_{u,u' \in \mathcal{C}_\ell} e_{uu'}, \quad \forall s \in \mathcal{C}_\ell \setminus \arg\min_{s' \in \mathcal{C}_\ell} d(r, s'). \quad (7)$$

Corollary 4.6: If there is a cycle $\mathcal{C}_\ell$ in the association graph, after the execution of Algorithm 2, for every feature r there exist at least two features s, s' in $\mathcal{C}_\ell$ as in (7) for which the elements $[\mathbf{z}_r]_s, [\mathbf{z}_r]_{s'}$ reach the same value.

On one hand, Theorem 4.4 formally proves the intuition behind equation (5), which was to transmit the errors so that $[\mathbf{z}_r(t)]_s$ reached the value of the error of the last link in the path that connects f_r with f_s. On the other hand, Theorem 4.5 states that the maximum error of all the links belonging to a cycle in the association graph will appear in as many elements of the vectors $\mathbf{z}_r(t)$ as features belonging to the cycle minus one. By Assumption 4.1 all the errors have different values, this means that if $[\mathbf{z}_r]_s$ is equal to $[\mathbf{z}_{r'}]_{s'}$, and this value appears only once in the whole vector, then the last link that connects f_r with f_s is the same one that connects $f_{r'}$ with $f_{s'}$, which can only be the link (s, s').

Taking this into account, the robots can analyze the final values of the vectors after the transmission of the errors to find the possible cuts to break the inconsistencies (lines 14-21 of Algorithm 2). Let us suppose that one robot wants to separate features r and r'. Any cut, (s, s') with $s \neq s'$, will satisfy three conditions in the variables $\mathbf{z}_r(t)$ and $\mathbf{z}_{r'}(t)$:

(a) $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'}$,
(b) for all $u \neq s$, $[\mathbf{z}_r]_s \neq [\mathbf{z}_r]_u$,
(c) for all $u \neq s'$, $[\mathbf{z}_{r'}]_{s'} \neq [\mathbf{z}_{r'}]_u$.

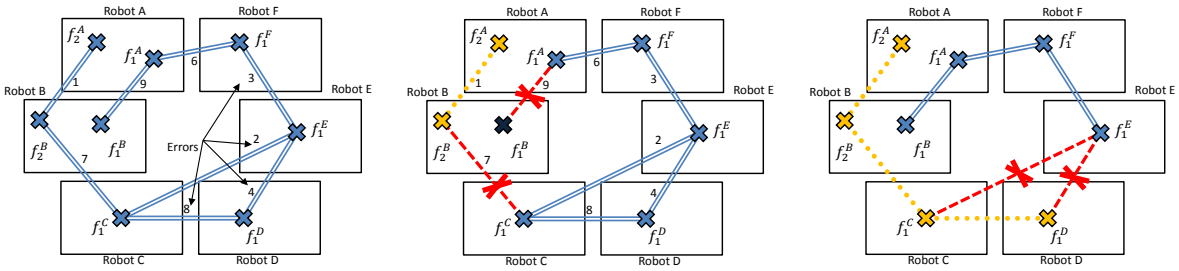
The first condition is extracted from the ideas in Theorem 4.4 whereas the other two are direct implications from Theorem 4.5. Finally, note that for any cut, $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'} = e_{ss'}$, i.e., the value of the cut equals the error of the local match. Therefore, the robots select from all the possible cuts to break the inconsistency, the one with the largest error. In case one robot has more than two features in the same inconsistency, the algorithm chooses two of the $\tilde{m}_i$ inconsistent features and selects the best cut for them. The cut separates all the $\tilde{m}_i$ features in two disconnected subsets. The process is repeated with each of the subsets until the inconsistencies are solved.

Remark 4.7 (Equal matching errors): If Assumption 4.1 is violated and two links belonging to an inconsistent set have the same error associated, then the propagation step will end up with two components of the vectors $\mathbf{z}_r$ having the same value. In this case the algorithm will not remove any of these two matches because it is interpreted as features that belong to a cycle. Nevertheless, as long as there are other cuts in the inconsistency the algorithm will still be able to solve it.

Finally, with our solution each robot can choose locally the best cut to break the inconsistency, keeping the communication constraints of the communication network. The algorithm has the drawback that it is not able to solve inconsistencies in which the inconsistent features belong to one cycle in the association graph. However, since the algorithm is able to detect this situation, a different approach can be used in this case to solve it.

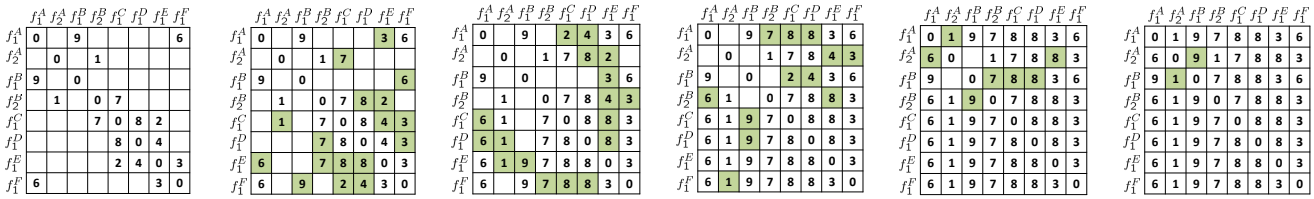
B. Resolution based on Spanning Trees

We propose an alternative algorithm to deal with the situations that the *Maximum Error Cut* does not solve. The method computes different spanning trees in each inconsistent set and,



(a) Inconsistency (b) *Maximum Error Cut* (c) *Spanning Trees*

Fig. 4. Example of execution of the resolution of one inconsistency using the two approaches. (a) Inconsistency. (b) Solution obtained using the *Maximum Error Cut* approach. (c) Solution obtained using the *Spanning Trees* algorithm. A detailed explanation is in section IV-C.



(a) (b) (c) (d) (e) (f)

Fig. 5. Example of execution of (5) for the inconsistency in Fig. 4 (a). Each subfigure (a)-(f) represents a new step of the algorithm, where the shaded elements are those which are updated at each iteration. In 6 steps the robots with inconsistent features are able to decide which links should be deleted. For more details see Section IV-C.

although the cuts done in the association graph are arbitrary, it has the property that it is able to solve all the inconsistencies. The algorithm constructs inconsistency-free components using a strategy close to a breadth-first search tree construction, where the robots include their inconsistent features in the different components in such a way that no inconsistencies appear.

Given an inconsistent set, we define the root robot, $i_\star$, as the robot with the most inconsistent features involved in the inconsistency. In case two robots have the same number of inconsistent features, the one with the lowest identifier is selected. The whole method is schematized in Algortihm 3.

In lines 1-6 of the algorithm, the root robot creates $\tilde{m}_{i_\star}$ components, as many as inconsistent features it has in the inconsistency, and initializes a different component $\mathcal{C}_q$ with each one of its features. Then, it tries to add to each component $\mathcal{C}_q$ the features directly associated to the feature assigned to that component by sending a *request* message to them. The rest of the robots do not do anything during the initialization and directly wait for *request* and *reject* messages.

Let us consider now that robot i receives a *request* message, coming from robot j, to join feature f_r^i to feature f_s^j in a specific component $\mathcal{C}_q$. Then four cases can be distinguished:

(a) f_r^i is already assigned to $\mathcal{C}_q$;
(b) f_r^i is assigned to a different component;
(c) other feature $f_{r'}^i$ is already assigned to $\mathcal{C}_q$;
(d) f_r^i is unassigned and no feature in i is assigned to $\mathcal{C}_q$.

In case (a), f_r^i already belongs to the component $\mathcal{C}_q$ and robot i does nothing, as shown in lines 9-10 of the algorithm. Lines 11 to 13 consider cases (b) and (c), where f_r^i cannot be added to $\mathcal{C}_q$ because it would create an inconsistency; in these two cases robot i deletes the link $[\mathbf{A}_{ij}]_{r,s}$ and replies with a *reject* message to robot j; when robot j receives the reject message (lines 19-21), it deletes the equivalent link $[\mathbf{A}_{ji}]_{s,r}$. Finally, in case (d) (lines 14-16), robot i assigns its feature f_r^i to the component $\mathcal{C}_q$ and the process is repeated.

Algorithm 3 *Spanning Trees* - Robot i

Require: Set of C different inconsistent sets
Ensure: $\mathcal{G}_{dis}$ is *inconsistency-free*

1: – *Initialization*
2: **for all** $\mathcal{C}$ such that i is root ($i = i_\star$) **do**
3: create $\tilde{m}_{i_\star}$ components, $\mathcal{C}_q$
4: assign each feature $f_r^{i_\star} \in \mathcal{C}$ to a different $\mathcal{C}_q$
5: send *request* to all features locally associated to $f_r^{i_\star}$
6: **end for**
7: – *Algorithm*
8: **for** each *request* from f_s^j to f_r^i **do**
9: **if** case (a) **then**
10: do nothing
11: **else if** cases (b) or (c) **then**
12: $[\mathbf{A}_{ij}]_{r,s} = 0$
13: send *reject* message to j
14: **else if** case (d) **then**
15: assign f_r^i to the component
16: send *request* to all its neighboring features
17: **end if**
18: **end for**
19: **for** each reject from f_s^j to f_r^i **do**
20: $[\mathbf{A}_{ij}]_{r,s} = 0$
21: **end for**

The algorithm ends its execution after no more than n communication rounds. When the algorithm finishes, each original inconsistent set $\mathcal{C}$ has been partitioned into $\tilde{m}_{i_\star}$ disjoint, inconsistency-free components. It may happen that

a subset of features remains unassigned. These features may still be inconsistent. The detection and resolution algorithms can be executed on the subgraph defined by this smaller subset of features obtaining in the end an association graph free of all the inconsistencies.

C. Example of execution

Let us consider one inconsistency as the one depicted in Fig. 4 (a) where the communication graph is a ring with an additional edge between robots C and E.

Figure 4 (b) shows the solution obtained using the *Maximum Error Cut* algorithm. The evolution of the $\mathbf{z}_r$ vectors is shown in Figure 5. Each figure, 5 (a) to 5 (f), represents a new iteration of the algorithm in (5). The -1 values are omitted for clarity. As an example of how it works, the fourth row in figure 5 (b), corresponding to f_2^B is obtained as follows. f_2^B executes (5) and updates its row in Fig. 5 (a) with the 2nd and 5th rows in Fig. 5 (a), sent by robots A and C because of features f_2^A and f_1^C. Robot B permutes the second and fourth element of the vector sent by robot A and the fourth and fifth element of vector sent by robot C and chooses the maximum (element to element) of the three vectors. As a result the sixth and seventh position in Fig. 5 (b) (features f_1^D and f_1^E) change their values. It is interesting to see how several elements in the different vectors receive the value "8", corresponding to the largest value within the cycle. Once rule (5) has finished, robots A and B look for the cuts to break their inconsistencies (Fig. 6). For robot B the best cut is the one matching features f_1^A and f_1^B. For the robot A the largest error is in the column associated to f_1^B. However, this is not a cut because both features have the same value in the same element. The next largest value is also discarded because it belongs to a cycle. Finally, the bridge with error 7 is selected because it is a cut and the match between f_2^B and f_1^C is deleted.

	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^A	0	1	9	7	8	8	3	6
f_2^A	6	0	9	1	7	8	8	3

	f_1^A	f_2^A	f_1^B	f_2^B	f_1^C	f_1^D	f_1^E	f_1^F
f_1^B	9	1	0	7	8	8	3	6
f_2^B	6	1	9	0	7	8	8	3

Fig. 6. Decisions to solve the inconsistency. Robot B chooses the edge (f_1^A, f_1^B). Robot A discards the elements with values 8 and 9 because they belong to a cycle and to an edge that does not solve its inconsistency respectively. The match between f_2^B and f_1^C solves the inconsistency and has the largest error.

The *Spanning Trees* solution is shown in Fig. 4 (c). The root robot to manage the inconsistency is the robot A. For each feature, robot A instantiates a different spanning tree. After 2 communications rounds, robots C and E send a request to D and also among them. f_1^D gets attached to f_1^C and the other edges are broken. After this point the algorithm has ended its execution and the new association graph is inconsistency-free.

V. Experiments

A. Simulations

We have designed a simulation environment using MATLAB to evaluate the performance of the proposed algorithms.

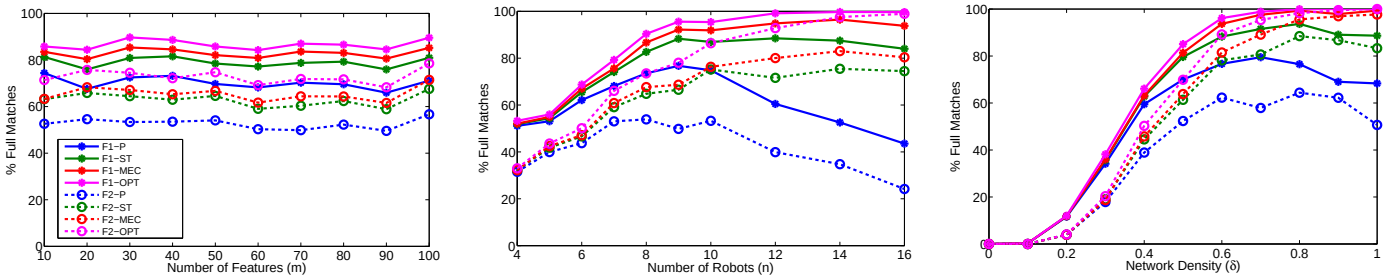
The environment considers a set of n robots observing the same m features. To find the local matches, we start from the perfect association graph of a fully connected graph. Then, we randomly remove a percentage of the perfect associations (p_m Missing Edges) and add a percentage of spurious matches (p_s Spurious Edges). The error of each match is randomly assigned between 0 and 10.

We assume the local matching to be deterministic, i.e., for the same matching function and pair of robots, the local matching is always the same. In this way we can repeat the experiment considering different network topologies. The networks are generated as random graphs, where each communication link has independent probability of existing, δ. We call this parameter the network density, because values close to 1 create networks with many links whereas small values of δ imply very sparse networks. Moreover, let us note that if $\delta = 1$, then the communication graph is complete and the robots have all the information about the matches available, which is equivalent to a centralized solution of the problem.

Since all the robots are observing all the features, we can define a metric to measure the correctness of the global matching. One match will be a *full match* when the n robots correctly associate the same feature. The ground truth solution is obtained when m full matches are obtained. Given a particular communication network and local matching, it is also interesting to analyze what is the best performance that our algorithms can obtain. In such case, the best possible solution is the one in which all the spurious links are removed and all the good links are kept for that particular association graph. We define this solution as the optimal distributed solution (OPT), which is computed propagating the local matches without the spurious links. In Fig. 7 (a)-(c) we show the percentage of full matches varying the number of features, of robots and the density of the network. The results are shown for two different matching functions, $F1 = [p_m, p_s] = [0.1, 0.1]$, and $F2 = [p_m, p_s] = [0.25, 0.05]$.

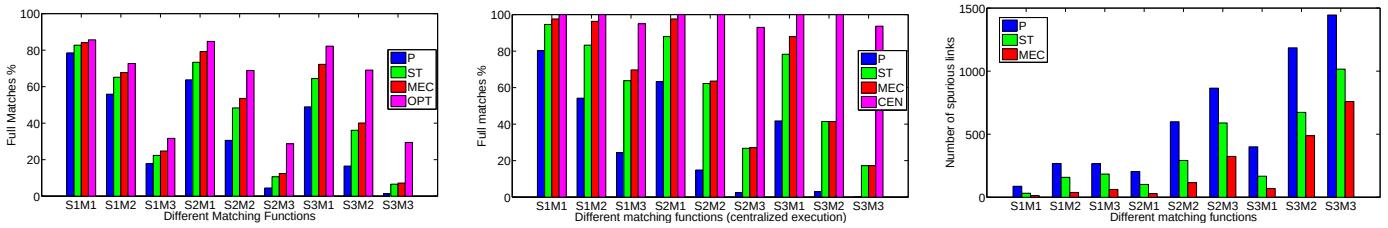
Influence of the number of features: Fig. 7 (a) shows the percentage of full matches of the optimal distributed solution (OPT), and the solutions obtained after the propagation (P), and after executing the *Maximum Error Cut* (MEC) and the *Spanning Trees* (ST) resolution methods for different values of m. Since the ground truth solution always corresponds to the 100% of full matches it is not depicted in the graphics. The number of robots is fixed and equal to $n = 8$ and the density of the network is $\delta = 0.5$. For each number of features we have repeated the experiment 100 times with different initial configurations. The number of features is a parameter without much influence on the obtained results. This makes sense because each association set is treated independently, and in general, with more features there are more sets, but not more complex inconsistencies. The performance of MEC is close to the OPT in any case. We can also see that there is a difference in the values depending on the local matcher, but we will analyze this later.

Influence of the number of robots: In Fig. 7 (b) we show the results for the same experiment fixing m to 15 features and varying the number of robots, n. As the number of robots is increased, the percentage of full matches after the propagation step is decreased because there are more outliers (and more inconsistencies). On the other hand, using any of the resolution



(a) Full matches versus m (b) Full matches versus n (c) Full matches versus δ

Fig. 7. Percentage of full matches for two matching functions and different number of features (m), robots (n) and network density (δ).



(a) Full matches (distributed) (b) Full matches (centralized) (c) Number of false positives

Fig. 8. Quality of different matchers. Figures (a) and (b) show the % of Full Matches for increasing percentages of missing and spurious links in the local associations in a distributed and a fully connected scenario. Figure (c) shows the number of spurious links (false positives) found before and after executing the resolution algorithms (in 100 iterations).

algorithms, the percentage is kept at good values (almost as good as the ground truth), which means that our resolution algorithms are robust to the number of robots participating.

Influence of the density of the network: In Fig. 7 (c) we show the results considering different densities of the network and fixed number of robots, $n = 8$, and features, $m = 15$. With more communication links between the robots our algorithms have a better performance. With few communication links it is more probable for a spurious match to pass undetected, whereas with more links, it will be easier to detect inconsistencies. This detection allows to improve the results by deleting more spurious edges. Let us also note that for $\delta = 1$ (all to all communications), the MEC algorithm obtains the 100% of full matches, which corroborates that the idea of deleting single links with large error is a good proxy for deleting spurious matches.

Influence of the local matching: In Figs. 7 (a)-(c) we can see that the matchers $F1$ and $F2$ return very different results. The quality of the function used for the local matching is a parameter that plays a fundamental role in our algorithm. In Fig. 8 (a)-(c) we have considered 9 different matchers, with $n = 8,\ m = 15$, and evaluated their performance in a distributed, Fig. 8 (a), and a fully connected scenario, Fig. 8 (b), comparing the results again OPT and a centralized implementation of OPT (CEN). For the distributed case we have also measured the amount of spurious links removed, Fig. 8 (c). The matchers have different values of $p_s,\ [S1, S2, S3] = [0.05, 0.15, 0.25]$, and different values of $p_m,\ [M1, M2, M3] = [0.1, 0.25, 0.5]$. Note that these values are in several cases, quite extreme, as it is very unlikely that a local matcher misses half of the associations and introduces a 25% of spurious links.

Our methods are more sensitive to missing links than to spurious links (columns with $M3$), where the gap w.r.t. OPT and CEN is notable. This happens because with fewer links it is harder to obtain a full match. Nevertheless, even using the worst matching functions, in Fig 8 (c) we can observe that a large number of the spurious links is removed. With more links, even if there are several spurious, by removing the appropriate matches we obtain a larger percentage of full matches, and almost the same results as OPT and CEN. Note that in all the cases, using the resolution methods we obtain better results than just considering the matching given by the propagation (P) of the local matches.

In conclusion, our algorithms are able to improve the initial data association, in the sense that they are able to identify and delete a great proportion of the spurious links introduced by the local matching. As a consequence, the global data association we obtain is better than the one obtained by just propagating the local associations. The performance is similar to OPT and for dense networks the results are close to the ground truth regardless of the local matcher.

B. Experiments with real data

We have also tested our proposal with real images considering different robotic scenarios such as robots moving in formation or multi-robot SLAM. In each example we have used different features and functions to find the local correspondences.

1) Data association in multi-robot formations using geometric constraints: In the first experiment we consider 6 robots moving in formation (around 5 m away from each other). Each robot acquires one image with its camera and extracts SURF features [30] (Fig. 9). The epipolar constraint plus RANSAC [12] is used for the local matching. The

detection and resolution of inconsistencies is analyzed for four different typical communication graphs (Fig. 10). The error function used for the *Maximum Error Cut* algorithm is the Sampson distance.

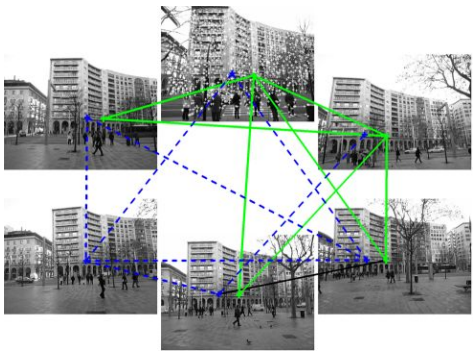
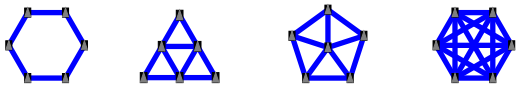


Fig. 9. Images acquired by 6 robots moving in formation. We show an example of one inconsistency solved with the *Spanning Trees* algorithm by deleting the black line. The blue lines show a full match and the green lines a partial match. For clarity, we do not show the rest of the matches or inconsistencies.



(a) Ring (b) Pyramidal (c) Star-Ring (d) Complete

Fig. 10. Formations used in the experiment.

We have chosen a man made scenario to be able to manually classify the matches. Although ground truth is not available in this example, by looking at the correspondences we have counted the amount of full matches. This number is very small or even zero due to missing matches and occlusions caused by the trees in the images. For that reason we also define a partial match when 3 or more robots correctly match the same feature, because in such case the propagation is required for the association. Even when a partial match does not represent a fully correct match, let us note that it is still free of any spurious information.

The results of the experiment can be seen in Table I. The number of features and links participating increases with the number of edges in $\mathcal{G}_{com}$ (first and second row in Table I). As a consequence, the propagation is able to find more inconsistencies (73 with a complete graph versus 1 with a ring topology), resulting in better results after the resolution. After executing the ST resolution algorithm, all the inconsistencies have been solved (second and third row in the ST block in Table I). The MEC is not able to solve one inconsistency out of 73 for the complete topology because the features that must be separated belong to a cycle. Fortunately, the algorithm is able to detect this situation, because the robots realize that there are no cuts, computing the *Spanning Trees* solution to resolve this inconsistency with results shown in parenthesis. Regarding the quality of the new association, using any of the two algorithms there are more full and partial matches than those found with the propagation, resulting in a better data association than the one obtained by plain propagation of the matches. In this aspect the MEC is able to obtain better

TABLE I

RESULTS FOR THE DATA ASSOCIATION OF IMAGES

Comm. graph	Fig. 10 (a)	(b)	(c)	(d)
Total Features	1727	1985	2126	2618
Total Links	1036	1345	1560	2326
AFTER PROPAGATION (P)				
Inconsistencies	1	14	29	73
Incons. feats.	7	89	183	510
Full Matches	7	10	13	15
Partial Matches	210	271	302	391
SPANNING TREES (ST)				
Deleted Links	1	17	40	126
Deleted False Positives	0	7	19	48
Inconsistencies	0	0	0	0
Incons. feats.	0	0	0	0
Full Matches	7	10	13	16
Partial Matches	211	284	329	460
MAXIMUM ERROR CUT (MEC)				
Deleted Links	1	20	38	108 (4)
Deleted False Positives	1	12	25	69 (2)
Inconsistencies	0	0	0	1 (0)
Incons. feats.	0	0	0	10 (0)
Full Matches	8	11	16	21
Partial Matches	210	282	324	462

results than the ST because it obtains a larger number of full and partial matches.

In Table II we show the number of iterations executed by our algorithms and the amount of information the robots exchange in KBytes (KB). First of all, let us note that the number of iterations of the propagation is always around $n = 6$, and therefore smaller than the conservative bounds given in Theorem 3.4. In terms of communications, the most expensive step is the initial one, in which the robots require to exchange the features with their neighbors to compute the local matches. Although only one iteration is required to send the features (same demands for all the networks), it is more demanding than the other steps because each SURF is composed by a 64-float descriptor, while in the propagation and the resolution steps the robots exchange pairs of integers. Note thus that methods based on multi-hop propagation of local features have larger communication consumptions than our approach. Let us also note that the amount of information exchanged in the propagation is in all the cases much smaller than $2m_{sum}^2$ and that the resolution algorithms have a very light communication load, since they are only executed in the presence of inconsistencies. MEC has slightly larger costs than ST since it requires to exchange the errors between the matches. Finally, in the last rows of Table II we have measured the percentage of time spent in the computations of the different steps of our algorithms in a single computer. It can be seen that the local matching of the features is the most demanding part of the algorithm, always requiring more than half of the total time.

2) Data association in multi-robot SLAM with stochastic maps: Our proposal is also of high interest in multi-robot exploration scenarios with limited communications. In this experiment each robot has explored a section of the environment and it has built a stochastic map using a SLAM algorithm. When the exploration finishes, the local maps are merged into a global map of the environment using the consensus-based distributed algorithm in [25].

TABLE II

COMMUNICATION FOR THE DATA ASSOCIATION OF IMAGES

Comm. graph	Fig. 10 (a)	(b)	(c)	(d)
LOCAL MATCHING				
Iterations	1	1	1	1
Mean(KB)	615.94	615.94	615.94	615.94
Max(KB)	698.88	698.88	698.88	698.88
PROPAGATION (P)				
Iterations	6	5	6	6
Mean(KB)	0.28	0.58	0.54	0.63
Max(KB)	0.92	1.55	1.88	2.44
SPANNING TREES (ST)				
Iterations	3	4	4	4
Mean(KB)	0.01	0.04	0.11	0.34
Max(KB)	0.02	0.28	0.70	1.26
MAXIMUM ERROR CUT (MEC)				
Iterations	9	9	10	14
Mean(KB)	0.01	0.13	0.28	0.72
Max(KB)	0.02	0.42	0.93	2.96
COMPUTATIONAL TIME (%)				
Extraction	8.8	6.6	6.3	3.7
Local Matching	76.2	77.0	78.3	68.3
Propagation	4.8	5.5	5.5	10.6
Resolution (ST)	5.0	5.2	4.2	6.4
Resolution (MEC)	5.2	5.7	5.7	11.0

Distributed map merging methods can cope with limited communication, switching topologies, or link failures in a much more robust way than solutions based on centralized schemes, all-to-all communication, or broadcasting methods, such as [21] for particle filters, [31] for multi-robot submaps, and [32] for graph maps of laser scan. Lately, some decentralized map merging algorithms have been proposed that, instead of using consensus-based solutions, rely on the storage and propagation of the measurements and odometry [33], or of the local maps [26] of all robots in the network. These methods however have the inconvenience that the memory cost does not scale well with the size of the network, i.e., if the number of robots is increased without changing the scene size, the memory cost increases as well. Consensus-based approaches as [25] do not suffer from this problem, since each robot keeps a single representation of the scene. The distributed data association method presented in this paper constitutes an interesting tool for these distributed map merging solutions.

We use a data set [34] with bearing information obtained with vision (Sony EVI-371DG). The landmarks are vertical lines extracted from the images. The measurements are labeled so that we can compare our results with the ground-truth data association. We select 8 sections of the whole path for the operation of 8 different robots and analyze the performance of the algorithm under 3 communication graphs (Fig 11). A separate SLAM is executed on each section, producing the 8 local maps (Fig. 12). In this case a full match is obtained when all the robots that observe one feature match it. As in many real scenarios, here the landmarks are close to each other, and the only information available for matching them are their cartesian coordinates. The local data associations are computed using the Joint Compatibility Branch and Bound (JCBB) [8] since it is very convenient for cluttered situations like the considered scenario. The JCBB is applied to the local maps of any pair of neighboring robots. The resulting local matches are depicted in Fig. 12.

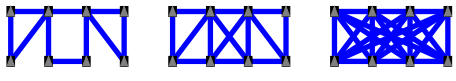


Fig. 11. Communication graphs between the 8 robots used for evaluating the data association of stochastic maps.

Robots establish the association between their features using the distributed data association described in this paper, and solve any inconsistent association detected. Recall that inconsistencies are always motivated by, at least, one spurious link; thus, if they are not removed, they may introduce serious deformations in the global map (Fig. 13). Observe that the global merged map using either the ground truth data association (Fig. 13 (b)), or the inconsistency-free association given by our methods (Fig. 13 (c)) appropriately resemble the scene (Fig. 13 (a)). Table III shows the results for the different network topologies in Fig. 11. We assign to each edge an error that depends on the number of matches between the local maps. Thus, we assume that an edge that belongs to a set with many jointly compatible matches has many chances of being a good edge. Between the edges belonging to the same set of jointly compatible associations, we use the individual Mahalanobis distance to slightly differentiate their errors. Then, we apply the two resolutions algorithms to solve the inconsistencies. The *Spanning Trees* approach, which does not take into account the errors associated to the edges, produces good results. For the three communication schemes, it improves the amount of full and partial matches. However, the *Maximum Error Cut* algorithm produces better results. The total number of edges deleted by this approach is lower, whereas the number of full matches is higher than for the *Spanning Trees* method.

TABLE III

RESULTS FOR THE DATA ASSOCIATION OF STOCHASTIC MAPS

Comm. graph	(a)	(b)	(c)
Total Features	194	194	194
Total Links	82	93	111
Total False Positives	4	13	21
AFTER PROPAGATION			
Inconsistencies	2	6	8
Incons. feats.	8	35	49
Full Matches	55	49	46
Partial Matches	4	3	2
SPANNING TREES			
Deleted Links	2	10	16
Deleted False Positives	2	6	10
Inconsistencies	0	0	0
Incons. feats.	0	0	0
Full Matches	55	53	50
Partial Matches	6	5	6
MAXIMUM ERROR CUT			
Deleted Links	2	7	14
Deleted False Positives	2	6	8
Inconsistencies	0	0	1 (0)
Incons. feats.	0	0	8 (0)
Full Matches	55	55	52
Partial Matches	4	3	3

Table IV shows the communication cost of the data association algorithm in KBytes (KB). The most expensive part consists of sending the local maps to neighbor robots. Therefore,

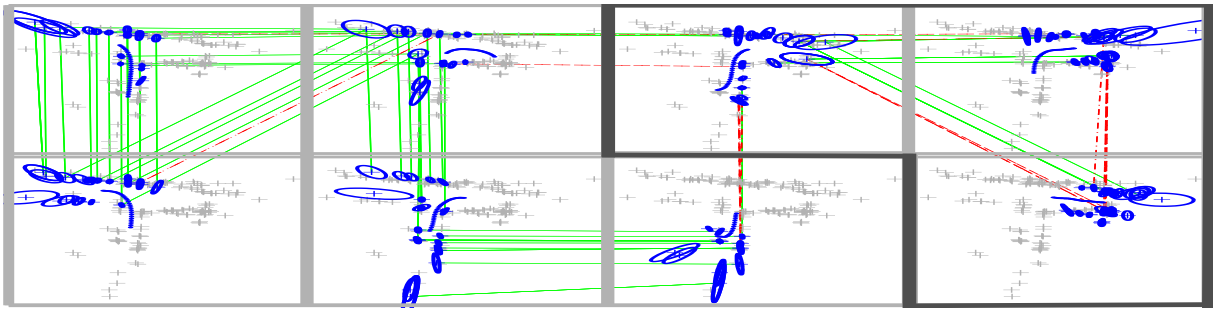
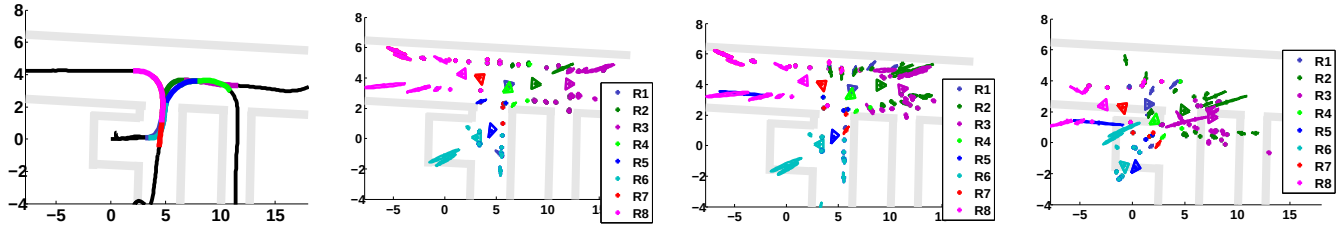


Fig. 12. Local maps acquired by 8 robots (blue) during their exploration. We also display the features observed by all the robots (gray crosses) to give an idea of the region explored by each robot. Each robot solves a local data association with its neighbors in the communication graph in Fig. 11 (a). Although many of the local edges are good (green solid lines), there are also some spurious matches (red dashed lines) that give rise to an inconsistency between 3 of the local maps (inside the dark gray area).



(a) Section explored (b) Ground truth data association (c) Inconsistency resolution (d) No inconsistency resolution

Fig. 13. Effects of using inconsistent data associations for merging the local maps. The section explored by the robots is composed of corridors and rooms (a). If robots knew the ground truth data association, the global merged map would resemble the original scene structure (b). The global merged map after resolving the inconsistencies with the *Maximum Error Cut* method appropriately resembles the scene as well (c). However, if inconsistencies are not removed, they may introduce serious deformations in the global merged map (d).

map merging methods based on multi-hop propagation of local maps will be more demanding in terms of communications than our approach. The propagation involves the exchange of a small number of integers (new associations found), and thus it has a much smaller cost. Finally, the inconsistency resolution, which is executed only on the inconsistent features, has a very low communication cost. The computational cost for the different parts of the algorithm can be found at the last rows of Table IV. As it can be observed, the computational effort for propagating matches and resolving inconsistencies is very low compared to the cost for establishing the local matches.

TABLE IV

COMMUNICATION FOR THE DATA ASSOCIATION OF STOCHASTIC MAPS

Comm. graph	(a)	(b)	(c)
LOCAL MATCHING			
Iterations	1	1	1
Mean(KB)	10.9	10.9	10.9
Max(KB)	27.9	27.9	27.9
PROPAGATION (P)			
Iterations	5	7	7
Mean(KB)	0.36	0.45	0.54
Max(KB)	0.50	0.71	0.84
SPANNING TREES (ST)			
Iterations	2	4	3
Mean(KB)	0.04	0.20	0.30
Max(KB)	0.12	0.33	0.56
MAXIMUM ERROR CUT (MEC)			
Iterations	4	8	8
Mean(KB)	0.05	0.46	0.85
Max(KB)	0.15	0.79	1.53
COMPUTATIONAL TIME (%)			
Local Matching	99.59	99.46	99.39
Propagation	0.395	0.501	0.473
Resolution (ST)	0.006	0.019	0.099
Resolution (MEC)	0.019	0.023	0.030

VI. CONCLUSIONS

We have presented distributed algorithms to compute consistent correspondences over a robotic network considering limited communications and vision sensors. The algorithms receive as input the local correspondences found between robots that can communicate. After that, a fully decentralized method to compute all the paths between local associations is carried out, allowing the robots to detect all the features that are associated with the ones they have observed and the inconsistencies that occur because of spurious local matches. In order to break these inconsistencies, two different algorithms have been presented that require only local communications. One of the algorithms considers the quality of each local match, when this information is provided by the local matcher. The other algorithm computes different spanning trees free of inconsistencies. We have evaluated our algorithms using simulated data and real images, considering different features and local matching functions where our algorithms can be used.

REFERENCES

[1] K. Y. K. Leung, T. D. Barfoot, and H. Liu, "Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 62 – 77, February 2010.

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

[3] N. F. Sandell and R. Olfati-Saber, "Distributed data association for multi-target tracking in sensor networks," in *IEEE International Conference on Decision and Control*, 2008, pp. 1085 – 1090.

[4] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3167 – 3172.

[5] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact pose slam," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 78 – 93, February 2010.

[6] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365 – 1378, December 2008.

[7] L. Pedraza, D. Rodriguez-Losada, F. Matia, G. Dissanayake, and J. V. Miro, "Extending the limits of feature-based SLAM with b-splines," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 353 – 366, April 2009.

[8] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890 – 897, December 2001.

[9] T. Bailey, E. M. Nebot, J. K. Rosenblatt, and H. F. Durrant-Whyte, "Data association for mobile robot navigation: a graph theoretic approach," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 2512 – 1517.

[10] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1879 – 1884.

[11] E. Olson, "Recognizing places using spectrally clustered local matches," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1157 – 1172, December 2009.

[12] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge: Cambridge University Press, 2000.

[13] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756 – 770, June 2004.

[14] J. J. Guerrero, A. C. Murillo, and C. Sagues, "Localization and matching using the planar trifocal tensor with bearing-only data," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 494 – 501, April 2008.

[15] R. Garg, D. Ramanan, S. Seitz, and N. Snavely, "Where's waldo: Matching people in images of crowds," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1793 – 1800.

[16] C. Cadena, D. Gálvez-López, J. Tardós, and J. Neira, "Robust place recognition with stereo sequences," *IEEE Transaction on Robotics*, vol. 28, no. 4, pp. 871 – 885, August 2012.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381 – 395, June 1981.

[18] L. Merino, J. Wiklund, F. Caballero, A. Moe, J. R. M. de Dios, P. Forssen, K. Nordberg, and A. Ollero, "Vision-based multi-uav position estimation," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 53 – 62, September 2006.

[19] A. Gil, O. Reinoso, M. Ballesta, and M. Juliá, "Multi-robot visual SLAM using a rao-blackwellized particle filter," *Robotics and Autonomous Systems*, vol. 58, no. 1, pp. 68 – 80, January 2010.

[20] X. S. Zhou and S. I. Roumeliotis, "Multi-robot slam with unknown initial correspondence: The robot rendezvous case," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 1785 – 1792.

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

[22] J. Yao and W. Cham, "Robust multi-view feature matching from multiple unordered views," *Pattern Recognition*, vol. 40, no. 11, pp. 3081 – 3099, November 2007.

[23] V. Ferrari, T. Tuytelaars, and L. V. Gool, "Wide-baseline multiple-view correspondences," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2003, pp. 718 – 725.

[24] K. Y. K. Leung, T. D. Barfoot, and H. Liu, "Distributed and decentralized cooperative simulatenous localization and mapping for dynamic and sparse robot networks," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 3841 – 3847.

[25] R. Aragues, J. Cortes, and C. Sagues, "Distributed consensus on robot networks for dynamically merging feature-based maps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 840 – 854, August 2012.

[26] A. Cunningham, K. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 1093 – 1100.

[27] S. Avidan, Y. Moses, and Y. Moses, "Centralized and distributed multi-view correspondence," *International Journal of Computer Vision*, vol. 71, no. 1, pp. 49 – 69, January 2007.

[28] R. Aragues, E. Montijano, and C. Sagues, "Consistent data association in multi-robot systems with limited communications," in *Robotics: Science and Systems*, 2010, pp. 97 – 104.

[29] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, electronically available at http://coordinationbook.info.

[30] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision*, 2006, pp. 404 – 417.

[31] S. B. Williams and H. Durrant-Whyte, "Towards multi-vehicle simultaneous localisation and mapping," in *IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 2743 – 2748.

[32] R. Vincent, D. Fox, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schulz, and B. Stewart, "Distributed multirobot exploration, mapping, and task allocation," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 1, pp. 229 – 255, 2008.

[33] K. Y. K. Leung, T. D. Barfoot, and H. H. T. Liu, "Decentralized cooperative simultaneous localization and mapping for dynamic and sparse robot networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3554 – 3561.

[34] U. Frese and J. Kurlbaum, "A data set for data association," June 2008. [Online]. Available: http://www.sfbtr8.spatial-cognition.de/insidedataassociation/

APPENDIX I
PROPERTIES OF ALGORITHM 1

This appendix presents the proofs related to the properties of Algorithm 1.

Proof of Proposition 3.2 (Limited Communications): By the definition of the operations in lines 9 and 13 of Algorithm 1 we can see that the components of $\mathbf{y}_r^i$ change their value at most once during the whole execution, for all $i \in \mathcal{V}_{com}, r \in \mathcal{S}_i$. This means that it is not necessary for the robots to send the whole blocks $\mathbf{y}_r^i$ to their neighbors but just the indices of the components that have changed their value from *false* to *true*. Each element can be identified by the row and the column and there are a total of m_{sum}^2 elements. Therefore, in the worst case, the amount of transmitted information through the network during the whole execution of the algorithm is $2m_{sum}^2$. ■

Proof of Proposition 3.3 (Correctness): If for some time t $[\mathbf{y}_r^i(t-1)]_u = 0$ and $[\mathbf{y}_r^i(t)]_u = 1, u = 1, \ldots, m_{sum}$ then a new path has been found in $\mathcal{G}_{dis}$ that includes the feature r. These new paths come either from the execution of line 9, or the execution of line 13.

Let t_i be the first time instant such that $\mathbf{y}_r^i(t_i) = \mathbf{y}_r^i(t_i - 1) \quad \forall r$ because no component has changed its value from zero to one for any of the features. This means that, for any feature in $\mathcal{S}_i$, there are no new paths with other features. By the properties of a path, it is obvious that if there are no new features at minimum distance t_i, it will be impossible that a new feature is at minimum distance $t_i + 1$. In addition, if no new paths at distance $t_i + 1$ can be found, the algorithm will not find new paths at distance $t_i + 2$ either. At this point the condition of line 16 is true and the algorithm ends with all the features connected to f_r^i contained in $\mathbf{y}_r^i(t_i)$. ■

Proof of Theorem 3.4 (Limited Iterations): We already know that the algorithm finishes in, at most, d_f iterations [28].

In the case that the matching does not contain any inconsistency $d_f \leq 2n$ and the result is valid.

Now let us suppose that there is one inconsistency. This implies that the communication graph, $\mathcal{G}_{com}$, contains one cycle of arbitrary length, ℓ. We divide the number of iterations in three parts. First $n-\ell$ iterations are required to ensure that the information of all the features belonging to the robots outside the cycle reaches at least one robot in the cycle.

Let us analyze now the iterations of the second part. In the worst case, the diameter of the subgraph defined by the cycle is $\ell/2$ and only $\ell+1$ features in the cycle form the inconsistency, which means that only one robot will execute, at some point, line 13 of Algorithm 1. It is clear that after $\ell/2+1$ iterations there will be at least two robots in the cycle, at maximum distance from each other $(\ell/2)$, with all the information. One of the robots, the one with the inconsistency, will obtain the information from the execution of line 13 in Algorithm 1. The other robot is the one with the common feature, detected by the first one. After this point $\ell/4$ iterations are required to share this information with the rest of the robots in the cycle and we can ensure that all the robots in the cycle have all the information about the inconsistency. Therefore, the second part requires $\ell/2+1+\ell/4 = \frac{3}{4}\ell+1$ iterations. If there are more than $\ell+1$ features inside the cycle forming the inconsistency the result is still valid.

With all the robots in the cycle knowing all the features that form the inconsistency, the number of additional iterations required to transmit the information to the rest of the network is upper bounded again by $n-\ell$. If we sum all the iterations we obtain $2n-\frac{5}{4}\ell+1$. Since the minimum length of a cycle is 3 the above quantity is always lower than $2n$. ■

APPENDIX II
PROPERTIES OF ALGORITHM 2

This appendix presents the proofs related to the properties of Algorithm 2.

Proof of Proposition 4.3 (Convergence): The features involved in the inconsistency form a strongly connected graph. For a given graph, the max consensus update is proved to converge in a finite number of iterations [29]. For any $r, s \in \mathcal{C}$ such that $[\mathbf{E}_\mathcal{C}]_{r,s} \geq 0$, by eq. (5) and the symmetry of $\mathbf{E}_\mathcal{C}$, the final consensus values of $\mathbf{z}_r$ and $\mathbf{z}_s$ satisfy, element to element, that

$$\mathbf{z}_r \geq \mathbf{z}_s \mathbf{P}_{rs} \text{ and } \mathbf{z}_s \geq \mathbf{z}_r \mathbf{P}_{sr}. \tag{8}$$

Using the properties of the permutation matrices, $\mathbf{P}_{rs} = \mathbf{P}_{sr}^T = \mathbf{P}_{sr}^{-1}$, we see that $\mathbf{z}_s \mathbf{P}_{rs} \geq \mathbf{z}_r$, which combined with eq. (8) yields to $\mathbf{z}_r = \mathbf{z}_s \mathbf{P}_{rs}$. For any feature, r, taking into account eq. (4), the update of the r^{th} element of $\mathbf{z}_r$ is

$$[\mathbf{z}_r(t+1)]_r = \max_{s \in \mathcal{C},\ [\mathbf{E}_\mathcal{C}]_{r,s} \geq 0} ([\mathbf{z}_r(t)]_r, [\mathbf{z}_s(t)]_s). \tag{9}$$

Recalling the first point in Assumption 4.1, the initial value of $[\mathbf{z}_r(0)]_r = e_{rr} = 0$, for all r, then $[\mathbf{z}_r(t)]_r = 0, \forall t \geq 0$. ■

Proof of Theorem 4.4 (Values for Bridges): First note that (s,u) is a bridge, therefore it creates a partition of $\mathcal{C}$ in two strongly connected, disjoint subsets

$$\mathcal{C}_s = \{r \in \mathcal{C} \mid d(r,s) < d(r,u)\},$$

$$\mathcal{C}_u = \{r \in \mathcal{C} \mid d(r,u) < d(r,s)\}. \tag{10}$$

In the above equations it is clear that $s \in \mathcal{C}_s$ and $u \in \mathcal{C}_u$.

We will focus now on the values of the s^{th} element of the state vector for the nodes in $\mathcal{C}_u$ and the u^{th} element for the nodes in $\mathcal{C}_s$,

$$[\mathbf{z}_r(t)]_u, r \in \mathcal{C}_s, \text{ and } [\mathbf{z}_r(t)]_s, r \in \mathcal{C}_u.$$

In the first case, for any $r \in \mathcal{C}_s \setminus s$, update rule (4) is equal to

$$[\mathbf{z}_r(t+1)]_u = \max_{r' \in \mathcal{C}_s,\ [\mathbf{E}_\mathcal{C}]_{r,r'} \geq 0} ([\mathbf{z}_r(t)]_u, [\mathbf{z}_{r'}(t)]_u), \tag{11}$$

because $r \neq u \neq r'$. The nodes in $\mathcal{C}_u$ are not taken into account because that would mean that (u,s) belongs to a cycle and it is not a bridge. The special case of feature s has an update rule equal to

$$[\mathbf{z}_s(t+1)]_u = \max_{r' \in \mathcal{C}_s, [\mathbf{E}_\mathcal{C}]_{s,r'} \geq 0} ([\mathbf{z}_s(t)]_u, [\mathbf{z}_{r'}(t)]_u, [\mathbf{z}_u(t)]_s). \tag{12}$$

In a similar way the updates for features in $\mathcal{C}_u$ are

$$[\mathbf{z}_r(t+1)]_s = \max_{r' \in \mathcal{C}_u,\ [\mathbf{E}_\mathcal{C}]_{r,r'} \geq 0} ([\mathbf{z}_r(t)]_s, [\mathbf{z}_{r'}(t)]_s),$$

$$[\mathbf{z}_u(t+1)]_s = \max_{r' \in \mathcal{C}_u, [\mathbf{E}_\mathcal{C}]_{u,r'} \geq 0} ([\mathbf{z}_u(t)]_s, [\mathbf{z}_{r'}(t)]_s, [\mathbf{z}_s(t)]_u).$$

Considering together all the equations and the connectedness of $\mathcal{C}_u$ and $\mathcal{C}_s$, all these elements form a connected component and they will converge to

$$[\mathbf{z}_r(t)]_u, [\mathbf{z}_{r'}(t)]_s \to \max_{r \in \mathcal{C}_s,\ r' \in \mathcal{C}_u} ([\mathbf{z}_r(0)]_u, [\mathbf{z}_{r'}(0)]_s). \tag{13}$$

Since all the features $r \in \mathcal{C}_s \setminus s$ are not associated with u, $[\mathbf{z}_r(0)]_u = -1$. Analogously, for all the features $r \in \mathcal{C}_u \setminus u$, $[\mathbf{z}_r(0)]_s = -1$. Finally, for the features u and s, by the second and third point of Assumption 4.1, $[\mathbf{z}_u(0)]_s = e_{us} = e_{su} = [\mathbf{z}_s(0)]_u \geq 0 > -1$. Therefore this subset of elements of the state vectors converge to the error of the edge (u,s), e_{us}. ■

Proof of Theorem 4.5 (Values for Cycles): We will use the following lemma to proof the result

Lemma 2.1: Let us consider a feature u, such that it is an articulation vertex, defined as a node in $\mathcal{C}$ whose deletion increases the number of connected components of $\mathcal{C}$. Denote $\mathcal{C}_u$ and $\mathcal{C}_{u'}$ the two partitions generated by its deletion. For any $r \in \mathcal{C}_u$, $s \in \mathcal{C}_{u'}$, $[\mathbf{z}_r(t)]_s$ will converge to the same value as $[\mathbf{z}_u(t)]_s$ and $[\mathbf{z}_s(t)]_r$ will converge to the same value as $[\mathbf{z}_u(t)]_r$. Moreover $[\mathbf{z}_r(t)]_u$ will converge to a different value than $[\mathbf{z}_s(t)]_u$.

Proof: Considering the fact that u is the only feature that connects $\mathcal{C}_u$ and $\mathcal{C}_{u'}$, the permutations in (4) for elements in $\mathcal{C}_u$ matched with u will not change the value of the components related to elements in $\mathcal{C}_{u'}$ and viceversa. On the other hand the permutations will affect the value of the $[\mathbf{z}_r(t)]_u$ and $[\mathbf{z}_s(t)]_u$ for features matched to u, shifting it to different positions in the two partitions. Then using Proposition 4.3 the result holds. ■

Proof of Theorem 4.5: Let $\bar{\mathcal{C}}_\ell = \mathcal{C} \setminus \mathcal{C}_\ell$ be the rest of the features in the inconsistency. Given a feature $r \in \bar{\mathcal{C}}_\ell$ there exists a unique $s \in \mathcal{C}_\ell$ such that there is at least one path of features in $\bar{\mathcal{C}}_\ell$ that ends in s. The uniqueness of s comes from the fact that if there were another feature $s' \in \mathcal{C}_\ell$, reachable

from r without passing through s, that would mean that r is also part of the cycle. Note that this does not discard the possibility that r and s belong to another cycle different than $\mathcal{C}_\ell$. Also note that $s = \arg\min_{s' \in \mathcal{C}_\ell} d(r, s')$.

Since s is the only connection with $\mathcal{C}_\ell$, then it is an articulation vertex and, by Lemma 2.1, for any $s' \in \mathcal{C}_\ell \backslash s$, $[\mathbf{z}_r]_{s'}$ will have final value equal to $[\mathbf{z}_s]_{s'}$. Therefore, if we show that (7) is true for the features belonging to the cycle then the theorem is proved.

Let us see what happens to features inside the cycle. First note that $r' = \arg\min_{s \in \mathcal{C}_\ell} d(r', s)$, $\forall r' \in \mathcal{C}_\ell$, and therefore, by Proposition 4.3, this element is always zero. Now, for any $r' \in \mathcal{C}_\ell$, if we consider another element $s \in \mathcal{C}_\ell$, such that r' is not directly matched to it, the update rule (4) is

$$[\mathbf{z}_{r'}(t+1)]_s = \max_{u \in \mathcal{C}_\ell, [\mathbf{E}_\mathcal{C}]_{r',u} \geq 0} ([\mathbf{z}_{r'}(t)]_s, [\mathbf{z}_u(t)]_s). \tag{14}$$

We have omitted other possible features that are directly matched to r' and that do not belong to $\mathcal{C}_\ell$ because they cannot be matched to s, otherwise they would belong to $\mathcal{C}_\ell$, and then, because of Lemma 2.1, they do not affect to the final value of $[\mathbf{z}_{r'}]_s$.

The special case of features in the cycle, s', directly matched to s has update rule equal to

$$[\mathbf{z}_{s'}(t+1)]_s = \max_{u \in \mathcal{C}_\ell \backslash s, [\mathbf{E}_\mathcal{C}]_{s',u} \geq 0} ([\mathbf{z}_{s'}(t)]_s, [\mathbf{z}_s(t)]_{s'}, [\mathbf{z}_u(t)]_s).$$

Due to the permutation, $[\mathbf{z}_{s'}]_s$ depends on the value of $[\mathbf{z}_s]_{s'}$. Then, by Proposition 4.3 and the connectedness of the cycle, in the end $[\mathbf{z}_r]_s$ will have the same value for all $r \in \mathcal{C}_\ell \setminus s$, and equal to the final value of $[\mathbf{z}_s]_{s'}$, for any s' in the cycle directly associated to s. By applying the same argument for any other element corresponding to a feature in $\mathcal{C}_\ell$ we conclude that after the execution of enough iterations of (4), for any $r \in \mathcal{C}_\ell$, $[\mathbf{z}_r]_s = [\mathbf{z}_r]_{s'}, \forall s, s' \in \mathcal{C}_\ell \setminus r$. Thus, each feature inside the cycle will end with $\ell - 1$ elements in its state vector with the same value, the maximum of all the considered edges, and (7) is true. ■

Eduardo Montijano (M'12) received the M.Sc. and Ph.D. degrees from the Universidad de Zaragoza, Spain, in 2008 and 2012 respectively, supervised by Prof. Carlos Sagüés. He is currently a Professor at Centro Universitario de la Defensa, in Zaragoza, Spain. His research interests are computer vision and consensus algorithms applied to multiple robots.

Rosario Aragues (M'12) is a Postdoc at the Clermont Universite, Institut Pascal, CNRS, UMR 6602, France, since April 2012. She received her M.S. and Ph.D. degrees in System Engineering and Computer Science from the University of Zaragoza, Spain in 2008 and 2012. Her research interests include multi-robot perception, map merging, and distributed consensus in robotic networks.

Carlos Sagüés (M'00, SM'11) received the M.Sc. and Ph.D. degrees from the Universidad de Zaragoza, Spain. During the course of his Ph.D. he worked on force and infrared sensors for robots. Since 1994 he has been Associate Professor and, since 2009 Full Professor with the Departamento de Informática e Ingenieía de Sistemas, Universidad de Zaragoza, where he has also been Head Teacher. His current research interest includes control systems, computer vision, visual robot navigation and multi-vehicle cooperative control.