

## Chapter 2

# Distributed Data Association

**Abstract** In this chapter, we address the association of features observed by the robots in a network with limited communications. At every time instant, each robot can only exchange data with a subset of the robot team that we call its neighbors. Initially, each robot solves a local data association with each of its neighbors. After that, the robots execute the proposed algorithm to agree on a data association between all their local observations. One inconsistency appears when chains of local associations give rise to two features from one robot being associated among them. In finite time, the algorithm finishes with a data association which is free of inconsistent matches. We show the performance of the proposed algorithms through simulations. Experiments with real data can be found in the last chapter.

**Keywords** Data association · Limited communication · Distributed systems · Parallel computation

## 2.1 Introduction

In multi-robot systems, a team of robots cooperatively perform some task in a more efficient way than a single robot would do. In this chapter, we address the data association problem. It consists of establishing correspondences between different measurements or estimates of a common element. It is of high interest in localization, mapping, exploration, and tracking applications [4]. There exists a wide variety of matching functions. The Nearest Neighbor (NN), and the Maximum Likelihood (ML), are widely used methods which associate each observation with its closest feature in terms of the Euclidean or the Mahalanobis distance [13, 15, 24]. Other popular method is the Joint Compatibility Branch and Bound (JCBB) [19], which considers the compatibility of many associations simultaneously. The combined constraint data association [5] builds a graph where the nodes are individually compatible associations and the edges relate binary compatible assignments. Over this graph, a maximal common subgraph problem is solved for finding the maximum clique in the graph. Scan matching and iterative closest point (ICP) [8] are popular methods for comparing two laser scans. Other methods, like the multiple hypothesis tracking, and the joint probabilistic data association, maintain many association hypothesis instead

of selecting one of them. And there exists many variations of these techniques that combine RANSAC [11] for higher robustness. All these matching functions operate on elements from two sets. One set usually contains the current observations, and the other one consists of the feature estimates. These sets may be two images, two laser scans, or two probabilistic maps.

Lately, many localization, mapping, and exploration algorithms for multi-robot systems have been presented. However, they have not fully addressed the problem of multi-robot data association. Some solutions have been presented for merging two maps [22, 24] that do not consider a higher number of robots. Many approaches rely on broadcasting all controls and observations measured by the robots. Then, the data association is solved like in a single robot scenario, using scan matching and ICP for laser scans [12, 14, 16, 21], or NN, ML, and visual methods for feature-based maps [13, 17]. Solutions based on submaps usually transform one of them into an observation of another. The local submaps are merged with the global map following a sequence [23], or in a hierarchical binary tree fashion [7].

In these methods, the problem of inconsistent data associations is avoided by forcing a cycle-free merging order. This limitation has also been detected in the computer vision literature. In [10] they approach an inconsistent association problem for identifying equal regions in different views. They consider a centralized scenario, where each two views are compared among them in a 2-by-2 way. Then, their results are arranged on a graph where associations are propagated and conflicts are solved. The work in [9], from the target tracking literature, simultaneously considers the association of all local maps. It uses an expectation-maximization method for both, computing the data association and the final global map. The main limitation of this work is that the data from all sensors needs to be processed together, what implies a centralized scheme, or a broadcast method.

All the previous methods rely on centralized schemes, full communication between the robots, or broadcasting methods. However, in multi-robot systems, distributed approaches are more interesting. They present a natural robustness to individual failures since there are no central nodes. Besides, they do not rely on any particular communication scheme, and they are robust to changes in the topology. On the other hand, distributed algorithms introduce an additional level of complexity in the algorithm design. Although, the robots make decisions based on their local data, the system must exhibit a global behavior.

In this chapter, we address the data association problem for distributed robot systems. Each of our robots posse a local observation of the environment. Instead of forcing a specific order for associating their observations, we allow the robots to compute its data association with each of its neighbors in the graph. Although this scenario is more flexible, it may lead to inconsistent global data associations in the presence of cycles in the communication graph. These inconsistencies are detected when chains of local associations give rise to two features from one robot being associated among them. These situations must be correctly identified and solved before merging the data. Otherwise, the merging process would be wrong and could not be undone. We approach this problem under limited communications. So, instead of comparing any two local observations among them, only the local

observations of neighboring robots can be compared. Besides, there is no central node that has knowledge of all the local associations and each robot exclusively knows the associations computed by itself. Then, each robot updates its local information by communicating with its neighbors. We present an algorithm where, finally, each robot is capable of detecting and solving any inconsistent association that involves any of its features.

## 2.2 Problem Description

We consider, a robotic team composed of  $n \in \mathbb{N}$  robots. The  $n$  robots have communication capabilities to exchange information with the other robots. However, these communications are limited. Let  $\mathcal{G}_{com} = (\mathcal{V}_{com}, \mathcal{E}_{com})$  be the undirected communication graph. The nodes 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  $i$ ,

$$\mathcal{N}_i = \{j \mid (i, j) \in \mathcal{E}_{com}\}.$$

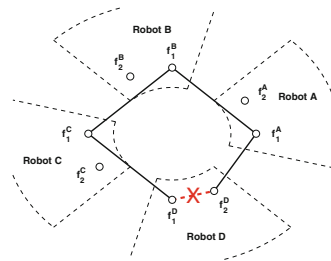
Each robot  $i$  has observed a set  $\mathcal{S}_i$  of  $m_i$  features,

$$\mathcal{S}_i = \{f_1^i, \dots, f_{m_i}^i\}.$$

It can compute the local data association between its own set  $\mathcal{S}_i$ , and the sets of its neighbors  $\mathcal{S}_j$ , with  $j \in \mathcal{N}_i$ . However, these data associations are not perfect. There may appear inconsistent data associations relating different features from the same set  $\mathcal{S}_i$  (Fig. 2.1). If the robots merge their data as soon as they solve the local data association, inconsistent associations cannot be managed since the merging cannot be undone. The goal of the algorithm is to detect and resolve these inconsistent associations before executing the merging.

In order to make the reading easy, along the chapter we use the indices  $i, j$ , and  $k$  to refer to robots and indices  $r, r', s, s'$ , to refer to features. The  $r$ th feature observed by the  $i$ th robot is denoted as  $f_r^i$ . Given, a matrix  $A$ , the notations  $A_{r,s}$  and  $[A]_{r,s}$

**Fig. 2.1** Robots A, B, C, and D associate their features comparing their maps in a two-by-two way. As a result, there is a path (dashed line) between  $f_1^D$  and  $f_2^D$ . This is an inconsistent association



correspond to the  $(r, s)$  entry of the matrix, whereas,  $A_{ij}$  denotes the  $(i, j)$  block when the matrix is defined by blocks. We let  $\mathbf{I}_k$  be the  $k \times k$  identity matrix, and  $\mathbf{0}_{k_1 \times k_2}$  a  $k_1 \times k_2$  matrix with all entries equal to zero.

### 2.2.1 Matching Between Two Cameras

Let  $F$  be a function that computes the local data association between any two sets of features,  $\mathcal{S}_i$  and  $\mathcal{S}_j$ , and returns an association matrix  $F(\mathcal{S}_i, \mathcal{S}_j) = \mathbf{A}_{ij}$  where  $\mathbf{A}_{ij} \in \mathbb{N}^{m_i \times m_j}$ ,

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

for  $r = 1, \dots, m_i$  and  $s = 1, \dots, m_j$ . We assume that  $F$  satisfies the following conditions.

**Assumption 1** (*Self Association*) When  $F$  is applied to the same set  $\mathcal{S}_i$ , it returns the identity,  $F(\mathcal{S}_i, \mathcal{S}_i) = \mathbf{A}_{ii} = \mathbf{I}$ .  $\square$

**Assumption 2** (*Unique Association*) The returned association  $\mathbf{A}_{ij}$  has the property that the features are associated 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  $s = 1, \dots, m_j$ .  $\square$

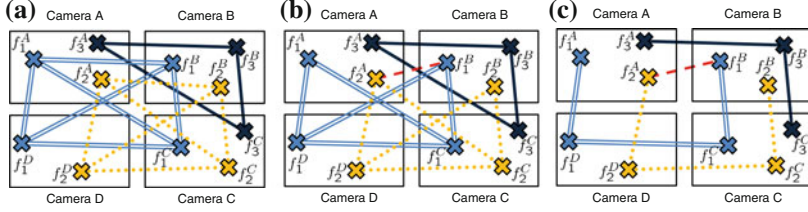
**Assumption 3** (*Symmetric Association*) Robots  $i$  and  $j$  associate their features in the same way. Given two sets  $\mathcal{S}_i$  and  $\mathcal{S}_j$  it holds that  $F(\mathcal{S}_i, \mathcal{S}_j) = \mathbf{A}_{ij} = \mathbf{A}_{ji}^T = (F(\mathcal{S}_j, \mathcal{S}_i))^T$ .  $\square$

Additionally, the local matching function may give information of the quality of each associations. The management of this information is discussed in Sect. 2.6.

We do not make any assumptions about the sets of features used by the cameras. However, we point out that the better the initial matching is, the better the global matching will be.

### 2.2.2 Centralized Matching Between $n$ Cameras

Let us consider now the situation in which there are  $n$  cameras and a central unit 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



**Fig. 2.2** Different association graphs. **a** Centralized matching with perfect association function. The graph is formed by disjoint cliques. **b** Centralized matching with imperfect association. Some links are missed,  $(f_1^A, f_1^B)$  and  $(f_2^A, f_2^B)$ , and spurious links appear,  $(f_2^A, f_1^B)$ . As a consequence, a subset of the features form a *conflictive set*. **c** Matching with limited communications. Now, the links between 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 camera is just the one provided by its neighbors

be represented by an undirected graph  $\mathcal{G}_{cen} = (\mathcal{F}_{cen}, \mathcal{E}_{cen})$ . Each node in  $\mathcal{F}_{cen}$  is a feature  $f_r^i$ , for  $i = 1, \dots, n$ ,  $r = 1, \dots, m_i$ . There is an edge between two features  $f_r^i, f_s^j$  iff  $[\mathbf{A}_{ij}]_{r,s} = 1$ .

For a perfect matching function, the graph  $\mathcal{G}_{cen}$  exclusively contains disjoint cliques, identifying features observed by multiple cameras (Fig. 2.2a). However, in real situations, the matching function will miss some matches and will consider as good correspondences some spurious matches (Fig. 2.2b). As a consequence, inconsistent associations relating different features from the same set  $\mathcal{S}_i$  may appear.

**Definition 1** An *association set* is a set of features such that they form a connected component in  $\mathcal{G}_{cen}$ . Such set is a *conflictive set* or an *inconsistent association* if there exists a path in  $\mathcal{G}_{cen}$  between two or more features observed by the same camera. A feature is *inconsistent* or *conflictive* if it belongs to an inconsistent association.  $\square$

Centralized solutions to overcome this problem are found in [3]. The latter one is also well suited for a distributed implementation but yet requires that any pair of images can be matched. In camera networks this implies global communications, which is not always possible.

### 2.2.3 Distributed Matching Between $n$ Cameras

Let us consider now that there is no central unit with all the information and there are  $n$  robots, each one with a camera and a process unit with limited communication capabilities. The robots are scattered forming a network with communications described with the undirected communication graph  $\mathcal{G}_{com} = (\mathcal{V}_{com}, \mathcal{E}_{com})$  introduced at the beginning of this section.

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 will be a subgraph of the centralized one,  $\mathcal{G}_{dis} = (\mathcal{F}_{dis}, \mathcal{E}_{dis}) \subseteq \mathcal{G}_{cen}$ ,

(Fig. 2.2c). 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 this chapter, 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  $\mathcal{G}_{dis}$ ,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \dots & \mathbf{A}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{n1} & \dots & \mathbf{A}_{nn} \end{bmatrix}, \quad (2.1)$$

where

$$\mathbf{A}_{ij} = \begin{cases} F(\mathcal{S}_i, \mathcal{S}_j) & \text{if } j \in \{\mathcal{N}_i \cup i\}, \\ \mathbf{0} & \text{otherwise.} \end{cases} \quad (2.2)$$

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

### 2.3 Propagation of Local Associations

Considering Definition 1 (Sect. 2.2.2), we observe that in order to find the data association sets with the relationship between the features observed by the different robots, it is required to compute the paths that exist among the elements in  $\mathcal{G}_{dis}$ . We show a process where robots start considering their local matches, and incrementally they propagate these local matches and discover all the paths between the features observed by the robot team. This information allows them as well to detect inconsistent associations (Definition 1). As the following lemma states [6], given a graph  $\mathcal{G}_{dis}$ , the powers of its adjacency matrix contains the information about the number of paths existing between the nodes of  $\mathcal{G}_{dis}$ :

**Lemma 1** (Lemma 1.32 [6]) *Let  $\mathcal{G}_{dis}$  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*

$i, j \in \{1, \dots, |\mathcal{V}|\}$  and  $t \in \mathbb{N}$  the  $(i, j)$  entry of the  $t$ th power of  $\mathbf{A}$ ,  $\mathbf{A}^t$ , equals the number of paths of length  $t$  (including paths with self-loops) from node  $i$  to node  $j$ .

**Algorithm 1** The computation of the powers of  $\mathbf{A}$  requires, a priori, the information about the whole matrix. We show now that this computation can also be done in a distributed manner [1]. Let each robot  $i \in \mathcal{V}_{com}$  maintain the blocks within  $\mathbf{A}^t$  associated to its own features,  $X_{ij}(t) \in \mathbb{N}^{m_i \times m_j}$ ,  $j = 1, \dots, n$ ,  $t \geq 0$ , which are initialized as

$$X_{ij}(0) = \begin{cases} \mathbf{I}, & j = i, \\ \mathbf{0}, & j \neq i, \end{cases} \quad (2.3)$$

and are updated, at each time step, with the following algorithm

$$X_{ij}(t+1) = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik} X_{kj}(t), \quad (2.4)$$

with  $\mathbf{A}_{ik}$  as defined in (2.2). It is observed that the algorithm is fully distributed because the robots only use information about its direct neighbors in the communication graph.

**Theorem 1** Let  $[\mathbf{A}^t]_{ij} \in \mathbb{N}^{m_i \times m_j}$  be the block within  $\mathbf{A}^t$  related to the associations between robot  $i$  and robot  $j$ . The matrices  $X_{ij}(t)$  computed by each robot  $i$  using the distributed algorithm (2.4) are exactly the submatrices  $[\mathbf{A}^t]_{ij}$ ,

$$X_{ij}(t) = [\mathbf{A}^t]_{ij}, \quad (2.5)$$

for all  $i, j \in \{1, \dots, n\}$  and all  $t \in \mathbb{N}$ .

*Proof* The proof is done using induction. First, we show that Eq. (2.5) is satisfied for  $t = 0$ . In this case, we have that  $\mathbf{A}^0 = \mathbf{I}$ , thus for all  $i, j \in \{1, \dots, n\}$ ,  $[\mathbf{A}^0]_{ii} = \mathbf{I}$  and  $[\mathbf{A}^0]_{ij} = \mathbf{0}$ , which is exactly the initial value of the variables  $X_{ij}$  (Eq. (2.3)).

Now we have that for any  $t > 0$ ,

$$[\mathbf{A}^t]_{ij} = \sum_{k=1}^n \mathbf{A}_{ik} [\mathbf{A}^{t-1}]_{kj} = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik} [\mathbf{A}^{t-1}]_{kj},$$

because  $\mathbf{A}_{ik} = \mathbf{0}$  for  $k \notin \{\mathcal{N}_i \cup i\}$ . Assuming that for all  $i, j \in \{1, \dots, n\}$  and a given  $t > 0$ ,  $X_{ij}(t-1) = [\mathbf{A}^{t-1}]_{ij}$  is true, then

$$X_{ij}(t) = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik} X_{kj}(t-1) = \sum_{k \in \{\mathcal{N}_i \cup i\}} \mathbf{A}_{ik} [\mathbf{A}^{t-1}]_{kj} = [\mathbf{A}^t]_{ij}.$$

Then, by induction,  $X_{ij}(t) = [\mathbf{A}^t]_{ij}$  is true for all  $t > 0$ . □

**Corollary 1** *The variables  $X_{ij}(t)$  contain the information about all the paths of length  $t$  between features observed by robots  $i$  and  $j$ .*

*Proof* By direct application of Lemma 1.  $\square$

Analyzing the previous algorithm the first issue to deal with is how to simplify the computation of the matrices in order to avoid high powers of  $\mathbf{A}$ . In the case, we are studying it is just required to know if there is a path between two elements in  $\mathcal{G}_{dis}$  and not how many paths are. This means that in this situation it is enough that  $[X_{ij}(t)]_{r,s} > 0$  in order to know that features  $f_r^i$  and  $f_s^j$  are connected by a path. Another issue is to decide when the algorithm in (2.4) must stop. Since the maximum length of a path between any two nodes in a graph is its diameter, then after  $d_f$  iterations the algorithm should stop. However, in general situations the robots will not know neither  $d_f$  nor  $m_{sum}$ , which makes this decision hard to be made a priori.

**Definition 2** We will say that two matrices  $\mathbf{A}$  and  $\bar{\mathbf{A}}$  of the same dimensions are equivalent,  $\mathbf{A} \sim \bar{\mathbf{A}}$ , if for all  $r$  and  $s$  it holds

$$[\mathbf{A}]_{r,s} > 0 \Leftrightarrow [\bar{\mathbf{A}}]_{r,s} > 0 \text{ and } [\mathbf{A}]_{r,s} = 0 \Leftrightarrow [\bar{\mathbf{A}}]_{r,s} = 0. \quad \square$$

In practice any equivalent matrix to the  $X_{ij}(t)$  will provide the required information, which allows to simplify the computations simply by changing any positive value in the matrices by 1. Moreover, the equivalency is also used to find a criterion to stop the algorithm:

**Proposition 1** *For a robot  $i$ , let  $t_i$  be the first time instant,  $t$ , such that  $X_{ij}(t) \sim X_{ij}(t-1)$  for all  $j = 1, \dots, n$ . Then robot  $i$  can stop to execute the algorithm at time  $t_i$ .*

*Proof* Let  $\bar{X}_{ij}(t)$  be the components in  $X_{ij}(t)$ , such that  $[X_{ij}(t-1)]_{r,s} = 0$  and  $[X_{ij}(t)]_{r,s} > 0$ . The cardinal,  $|\bar{X}_{ij}(t)|$ , represents the number of features  $f_s^j \in \mathcal{S}_j$  such that the minimum path length in  $\mathcal{G}_{dis}$  between them and one feature  $f_r^i \in \mathcal{S}_i$  is  $t$ . At time  $t_i$ ,  $X_{ij}(t_i) \sim X_{ij}(t_i - 1) \forall j$  for the first time, and then  $\sum_{j=1}^n |\bar{X}_{ij}(t_i)| = 0$  because no component has changed its value from zero to a positive. This means that there is no path of minimum distance  $t_i$  linking any feature  $f_r^i$  with any other feature in  $\mathcal{G}_{dis}$ . By the physical properties of a path, it is obvious that if there are no features at minimum distance  $t_i$ , it will be impossible that a feature is at minimum distance  $t_i + 1$  and all the paths that connect features of robot  $i$  with any other feature have been found.  $\square$

**Corollary 2** *All the robots end the execution of the iteration rule (2.4) in at most in  $d_f + 1$  iterations.*

*Proof* Recalling that the maximum distance between two nodes in  $\mathcal{G}_{dis}$  is the diameter of the graph, denoted by  $d_f$ , then  $\sum_{j=1}^n |\bar{X}_{ij}(d_f + 1)| = 0$  for all  $i = 1, \dots, n$ .  $\square$



If a robot  $j$  at time  $t$  does not receive the information  $X_{ij}(t)$  from robot  $i$  then it will use the last matrix received, because robot  $i$  has already finished computing its paths and  $X_{ij}(t) \sim X_{ij}(t-1)$ .

When the algorithm finishes, each robot  $i$  has the information about all the association paths of its features and the features of the rest of the robots in the network in the different variables  $X_{ij}(t_i)$ . It remains to analyze which features are conflictive and which are not.

**Algorithm 2** The robots detect all the conflictive features using two simple rules. A feature  $f_r^i$  is conflictive if and only if one of the following conditions are satisfied:

- (i) There exists other feature  $f_{r'}^i$ , with  $r \neq r'$ , such that

$$[X_{ii}(t_i)]_{r,r'} > 0; \quad (2.6)$$

- (ii) There exist features  $f_s^j$  and  $f_{s'}^j$ ,  $s \neq s'$ , such that

$$[X_{ij}(t_i)]_{r,s} > 0 \text{ and } [X_{ij}(t_i)]_{r,s'} > 0. \quad (2.7)$$

In conclusion, the proposed algorithm will be able to find all the inconsistencies in a finite number of iterations. The algorithm is distributed and it is based only on local interactions between the robots. Each robot only needs to know its local data associations. It updates its information based on the data exchanged with its neighbors. When the algorithm finishes, each robot  $i$  can extract from its own matrices  $X_{ij}(t_i)$  all the information of any conflict that involves any of its features. If the robot has any conflictive feature, it also knows the rest of features that belong to the conflictive set independently of the robot that observed such features. An algorithm to carry out the same process, but exploiting local information through the use of logical operations can also be used [18].

## 2.4 Algorithm Based on Trees

The resolution of inconsistent associations consists of deleting edges from  $\mathcal{G}_{dis}$  so that the resulting graph is conflict-free.

**Definition 3** Let  $C$  denote the number of conflictive sets in  $\mathcal{G}_{dis}$ . We say a conflictive set  $\mathcal{C}$  is *detectable* by a robot  $i$  if there exists a  $r \in \{1, \dots, m_i\}$  such that  $f_r^i \in \mathcal{C}$ . The set of robots that detect a conflictive set  $\mathcal{C}$  is  $R \subseteq \mathcal{V}_{com}$ . The number of features from each robot  $i \in R$  involved in  $\mathcal{C}$  is  $\tilde{m}_i$ . We say  $\mathcal{G}_{dis}$  is *conflict-free* if  $C = 0$ .  $\square$

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

The resolution problem consists of partitioning  $\mathcal{C}$  into a set of disjoint conflict-free components  $\mathcal{C}_q$  such that

$$\bigcup_q \mathcal{C}_q = \mathcal{C}, \text{ and } \mathcal{C}_q \cap \mathcal{C}_{q'} = \emptyset,$$

for all  $q, q' = 1, 2, \dots$ . The number of such conflict-free components is a priori unknown and it will be discussed later in this section.

Obtaining an optimal partition that minimizes the number of deleted edges is complicated. If there were only two inconsistent features  $f_r^i, f_{r'}^i$ , it could be approached as a max-flow min-cut problem [20]. However, in general there will be more inconsistent features,  $\tilde{m}_i \geq 2$ , within  $\mathcal{C}$  associated to a robot  $i \in R$ . Besides, there may also be  $\tilde{m}_j \geq 2$  inconsistent features belonging to a different robot  $j \in R$ . The application of [20] separately to any pair of inconsistent features does not necessarily produce an optimal partition. It may happen that a single edge deletion simultaneously resolves more than one inconsistent association. Therefore, an optimal solution should consider multiple combinations of edge deletions, what makes the problem computationally intractable, and imposes a centralized scheme. The algorithm presented is not optimal but is efficient and is proven to be correct and can be applied in distributed systems.

**Proposition 2** *Let  $R$  be the set of robots that detect  $\mathcal{C}$ . Let  $i_\star$  be the root robot with the most features involved in  $\mathcal{C}$ ,*

$$i_\star = \arg \max_{i \in R} \tilde{m}_i. \quad (2.8)$$

*The number of conflict-free components in which  $\mathcal{C}$  can be decomposed is lower bounded by  $\tilde{m}_{i_\star}$ .*

*Proof* Each conflict-free component can contain, at most, one feature from a robot  $i \in R$ . Then, there must be at least,  $\max_{i \in R} \tilde{m}_i = \tilde{m}_{i_\star}$  components.  $\square$

The resolution algorithm [1] constructs  $\tilde{m}_{i_\star}$  conflict-free components using a strategy close to a BFS tree construction. Initially, each robot  $i$  detects the conflictive sets for which it is the root using its local information  $X_{i1}(t_i), \dots, X_{in}(t_i)$ . The root robot for a conflictive set is the one with the most inconsistent features involved. In case two robots have the same number of inconsistent features, the one with the lowest robot id is selected. Then, each robot executes the resolution algorithm (Algorithm 2.4.1).

The root robot creates  $\tilde{m}_{i_\star}$  components and initializes each component  $\mathcal{C}_q$  with one of its features  $f^{i_\star} \in \mathcal{C}$ . Then, it tries to add to each component  $\mathcal{C}_q$  the features directly associated to  $f^{i_\star} \in \mathcal{C}_q$ . Let us consider that  $f_s^j$  has been assigned to  $\mathcal{C}_q$ . For all  $f_r^i$  such that  $[A_{ij}]_{r,s} = 1$ , robot  $j$  sends a component request message to robot  $i$ . When robot  $i$  receives it, it may happen that

- (a)  $f_r^i$  is already assigned to  $\mathcal{C}_q$ ;
- (b)  $f_r^i$  is assigned to a different component;

**Algorithm 2.4.1** Spanning Trees - Robot  $i$ 


---

```

1: – Initialization
2: for each conflictive set  $\mathcal{C}$  for which  $i$  is root ( $i = i_\star$ ) do
3:   create  $\tilde{m}_{i_\star}$  components
4:   assign each inconsistent feature  $f_r^{i_\star} \in \mathcal{C}$  to a different component  $\mathcal{C}_q$ 
5:   send component request to all its neighboring features
6: end for
7:
8: – Algorithm
9: for each component request from  $f_s^j$  to  $f_r^i$  do
10:  if (b) or (c) then
11:     $[\mathbf{A}_{ij}]_{r,s} = 0$ 
12:    send reject message to  $j$ 
13:  else if (d) then
14:    assign  $f_r^i$  to the component
15:    send component request to all its neighboring features
16:  end if
17: end for
18: for each component reject from  $f_s^j$  to  $f_r^i$  do
19:   $[\mathbf{A}_{ij}]_{r,s} = 0$ 
20: end for

```

---

- (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. In cases (b) and (c),  $f_r^i$  cannot be added to  $\mathcal{C}_q$ ; robot  $i$  deletes the edge  $[\mathbf{A}_{ij}]_{r,s}$  and replies with a reject message to robot  $j$ ; when  $j$  receives the reject message, it deletes the equivalent edge  $[\mathbf{A}_{ji}]_{s,r}$ . In case (d), robot  $i$  assigns its feature  $f_r^i$  to the component  $\mathcal{C}_q$  and the process is repeated.

**Theorem 2** *Let us consider that each robot  $i \in \mathcal{V}_{com}$  executes the distributed resolution algorithm (Algorithm 2.4.1) on  $\mathcal{G}_{dis}$ , obtaining  $\mathcal{G}'_{dis}$ ,*

- (i) *after  $t = n$  iterations no new features are added to any component  $\mathcal{C}_q$  and the algorithm finishes;*
- (ii) *each obtained  $\mathcal{C}_q$  is a connected component in  $\mathcal{G}'_{dis}$ ;*
- (iii)  *$\mathcal{C}_q$  is conflict free;*
- (iv)  *$\mathcal{C}_q$  contains at least two features;*

*for all  $q \in \{1, \dots, \tilde{m}_{i_\star}\}$  and all conflictive sets.*

*Proof* (i) The maximal depth of a conflict-free component is  $n$  since, if there were more features, at least two of them would belong to the same robot. Then, after at most  $n$  iterations of this algorithm, no more features are added to any component  $\mathcal{C}_q$  and the algorithm finishes.

(ii) There is a path in  $\mathcal{G}_{dis}$  between any two features belonging to a conflictive set  $\mathcal{C}$ . Therefore, there is also a path in  $\mathcal{G}_{dis}$  between any two features assigned to the same component  $\mathcal{C}_q$ . Since the algorithm does not delete edges from  $\mathcal{G}_{dis}$  within a component (case (a)), then  $\mathcal{C}_q$  is also connected in  $\mathcal{G}'_{dis}$ . Since none feature can be assigned to more than one component (case (b)), the components are disjoint. Therefore,  $\mathcal{C}_q$  is a connected component in  $\mathcal{G}'_{dis}$ .

(iii) By construction, two features from the same robot are never assigned to the same component  $\mathcal{C}_q$  (case (c)). Therefore, each component is conflict-free.

(iv) Each conflictive set has more than one feature. Because of Assumptions 1 and 2, each feature and its neighbors are conflict free. Therefore, each component  $\mathcal{C}_q$  contains, at least, its originating feature, and a neighboring feature. Thus, it has at least two features.  $\square$

**Corollary 3** *After executing Algorithm 2.4.1, the size of each conflict set  $\mathcal{C}$  is reduced by at least  $2 \tilde{m}_{i_*}$ , where  $\tilde{m}_{i_*} \geq 2$ .*  $\square$

When the algorithm finishes, each original conflictive set  $\mathcal{C}$  has been partitioned into  $\tilde{m}_{i_*}$  conflict-free components. It may happen that a subset of features remains unassigned. These features may still be conflictive in  $\mathcal{G}'_{dis}$ . The detection algorithm (Algorithm 2) can be executed on the subgraph defined by this smaller subset of features.

**Proposition 3** *Consider each robot  $i$  iteratively executes the detection (Sect. 2.3) and the resolution (Sect. 2.4) algorithms. Then, in a finite number of iterations, all conflictive sets disappear.*

*Proof* After each execution of the resolution algorithm, the size of each conflict set  $\mathcal{C}$  is reduced by, at least,  $2 \tilde{m}_{i_*} \geq 4$  (Corollary 3). Then, in a finite number of iterations, it happens that  $|\mathcal{C}| < 4$ . A set with 3 features  $f_r^i, f_{r'}^i, f_s^j$  cannot be conflictive; this would require the existence of edges  $(f_r^i, f_s^j)$  and  $(f_{r'}^i, f_s^j)$ , what is impossible (Assumption 2). A set with 2 features cannot be conflictive (Assumptions 1 and 2), and a set with a single feature cannot be inconsistent by definition. Therefore, there will be no remaining inconsistencies or conflictive sets.  $\square$

The main interest of the presented resolution algorithm is that it is fully distributed and it works on local information. Each robot uses its own  $X_{ij}(t_i)$  for detecting the root robot of each conflictive set. During the resolution algorithm, the decisions, and actions taken by each robot are based on its local associations  $\mathbf{A}_{ij}$ , and the components assigned to its local features. Moreover, each robot is responsible of deleting the edges from its local association matrices  $\mathbf{A}_{ij}$ , with  $j \in \{1, \dots, n\}$ . In addition, the presented algorithm works in finite time. Let us note that although we presented the algorithm for a single conflictive set, all conflictive sets are managed in parallel.

## 2.5 Feature Labeling

Simultaneously to the data association process, the robots assign labels to their features. After checking feature  $f_r^i$  is consistent, robot  $i$  assigns it a label  $L_r^i = (i_\star, r_\star) \in \mathbb{N}^2$  composed of a robot identifier  $i_\star$  and a feature index  $r_\star$  as follows [2]. Assume  $f_r^i$  and features  $f_s^j, f_{s'}^{j'}, \dots$  form a consistent association set in  $\mathcal{G}_{dis}$ , and thus, they are observations of a common landmark in the environment taken by robots  $i, j, j', \dots$ . Among all the candidates  $(i, r), (j, s), (j', s'), \dots$ , a unique label  $(i_\star, r_\star)$  is selected by the robots, e.g., the one with the lowest robot id. Then, robot  $i$  assigns this label to  $f_r^i, L_r^i = (i_\star, r_\star)$ ; the other robots  $j, j', \dots$ , proceed in a similar way so that finally,

$$L_r^i = L_s^j = L_{s'}^{j'} = \dots = (i_\star, r_\star).$$

We say a feature  $f_r^i$  is *exclusive* if it is isolated in  $\mathcal{G}_{dis}$ , corresponding to a landmark observed by a single robot  $i$ ; in this case, its label  $L_r^i$  is simply  $(i, r)$ . Otherwise, we say  $f_r^i$  is nonexclusive and it may either be *consistent* or *conflictive*. Consistent features are labeled as explained above, whereas robots wait until conflicts are *resolved* for labeling its conflictive features. The data association and labeling process finishes with an association graph  $\mathcal{G}_{dis}$  free of any inconsistent association and with all the features labeled. When the algorithm finishes, two features  $f_r^i, f_s^j$  have the same label,  $L_r^i = L_s^j$ , iff they are connected by a path in the resulting conflict-free  $\mathcal{G}_{dis}$ . The distributed data association and labeling algorithm is summarized in Algorithm 2.5.1. This strategy makes use of two subroutines to detect features and resolve inconsistencies that we explained in the previous sections.

Throughout this section, we use  $\tilde{\mathcal{S}}_i \subseteq \mathcal{S}_i$  for the set of unlabeled features at robot  $i \in \{1, \dots, n\}$  and let  $|\tilde{\mathcal{S}}_i|$  be its cardinality, i.e., the number of unlabeled features at robot  $i$ . The set of labels  $\mathcal{L}_i$  consists of the labels  $L_r^i$  already assigned to the features  $f_r^i \in \mathcal{S}_i \setminus \tilde{\mathcal{S}}_i$ . Given a matrix  $X_{ij}$  of size  $|\tilde{\mathcal{S}}_i| \times |\tilde{\mathcal{S}}_j|$ , we define the function  $\bar{r} = \text{row}(f_r^i)$  that takes an unlabeled feature  $f_r^i \in \tilde{\mathcal{S}}_i$  and returns its associated row in  $X_{ij}$ , with  $\bar{r} \in \{1, \dots, |\tilde{\mathcal{S}}_i|\}$ . Equivalently, we define the function  $\bar{s} = \text{col}(f_s^j)$  for features in  $\tilde{\mathcal{S}}_j$ . We let  $\tilde{\mathbf{A}}_{ij} \in \mathbb{N}^{|\tilde{\mathcal{S}}_i| \times |\tilde{\mathcal{S}}_j|}$  be like the local association matrix  $\mathbf{A}_{ij}$ , but containing exclusively the rows and columns of the unlabeled features of robots  $i$  and  $j$ .

Initially, all the features of each robot  $i$  are unlabeled,

$$\tilde{\mathcal{S}}_i = \{f_1^i, \dots, f_{m_i}^i\}, \quad \mathcal{L}_i = \emptyset.$$

Each robot  $i$  solves a local data association with each of its neighbors  $j \in \mathcal{N}_i$  and obtains the association matrix  $\mathbf{A}_{ij} \in \mathbb{N}^{m_i \times m_j}$ . Then, the robot locally detects its *exclusive* features  $f_r^i$  which have not been associated to any other feature,

$$[\mathbf{A}_{ij}]_{r,s} = 0 \quad \text{for all } j \in \mathcal{N}_i, j \neq i, \text{ and all } s \in \{1, \dots, m_j\}. \quad (2.9)$$

**Algorithm 2.5.1** Data association and labeling - Robot  $i$ 


---

```

1:  $\tilde{\mathcal{F}}_i \leftarrow \{f_1^i, \dots, f_{m_i}^i\}$ ,  $\mathcal{L}_i \leftarrow \emptyset$ 
2: Solve the local data association
3: ASSIGN_LABEL( $L_r^i = (i, r)$ ,  $f_r^i$ ) to each exclusive feature  $f_r^i$ 
4: while  $|\tilde{\mathcal{F}}_i| > 0$  do
5:   Run the detection algorithm 2
6:   Find each consistent feature  $f_r^i$  and its root  $f_{r_\star}^{i_\star}$ 
7:   ASSIGN_LABEL( $L_r^i = (i_\star, r_\star)$ ,  $f_r^i$ )
8:   Run the resolution algorithm
9:   Find each resolved feature  $f_r^i$  and its component id  $[i_\star, r_\star]$ 
10:  ASSIGN_LABEL( $L_r^i = (i_\star, r_\star)$ ,  $f_r^i$ )
11:  Find each exclusive feature  $f_r^i$ 
12:  ASSIGN_LABEL( $L_r^i = (i, r)$ ,  $f_r^i$ )
13: end while
14: function ASSIGN_LABEL( $L_r^i$ ,  $f_r^i$ )
15:    $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \{L_r^i\}$ ,  $\tilde{\mathcal{F}}_i \leftarrow \tilde{\mathcal{F}}_i \setminus \{f_r^i\}$ 
16: end function

```

---

Since an exclusive feature  $f_r^i$  is always consistent, robot  $i$  assigns a label  $L_r^i$  to it, composed of its own robot id and feature index and removes it from the set of unlabeled features,

$$L_r^i = (i, r), \quad \mathcal{L}_i = \mathcal{L}_i \cup L_r^i, \quad \tilde{\mathcal{F}}_i = \tilde{\mathcal{F}}_i \setminus \{f_r^i\}. \quad (2.10)$$

Since its unlabeled features in  $\tilde{\mathcal{F}}_i$  may be conflictive, it executes the detection algorithm (2.4) on this subset.

The detection algorithm (Algorithm 2) is executed on the subgraph of  $\mathcal{G}_{dis}$  involving the features in  $\tilde{\mathcal{F}}_i$ , for  $i \in \{1, \dots, n\}$ . When it finishes, robot  $i$  has the power matrices  $X_{ij} \in \mathbb{N}^{|\tilde{\mathcal{F}}_i| \times |\tilde{\mathcal{F}}_j|}$ , for  $j = 1, \dots, n$ , which contain the entries in  $\mathbf{A}^{\text{diam}(\mathcal{G}_{dis})}$  associated to the features in  $\tilde{\mathcal{F}}_i$  and  $\tilde{\mathcal{F}}_j$ . There is a path between  $f_r^i$  and  $f_s^j$  iff

$$[X_{ij}]_{\bar{r}, \bar{s}} > 0, \quad (2.11)$$

being  $\bar{r} = \text{row}(f_r^i)$  and  $\bar{s} = \text{col}(f_s^j)$ . These matrices give robot  $i$  the information about all the association paths of its features and the features of the rest of the robots in the network.

Then, each robot  $i$  detects its *consistent* features. After a feature  $f_r^i$  has been classified as consistent, its robot  $i$  proceeds to assign it a label. Here, we show how robot  $i$  decides the feature label  $(i_\star, r_\star)$ . Let us first give a general definition of the root robot of an either consistent or conflictive association set.

**Definition 4** The *root* robot  $i_\star$  for an association set is the one that has the most features in it. In case there are multiple candidates, it is the one with the lowest identifier. Equivalently, we define the *root* features  $f_{r_\star}^{i_\star}, f_{r_\star'}^{i_\star}, \dots$  as the features from the root robot that belong to the association set.  $\square$

Using the power matrices  $X_{i1}, \dots, X_{in}$ , robot  $i$  can find the number of features  $\tilde{m}_j$  from a second robot  $j$  that belong to the same association set than  $f_r^i$  with  $\bar{r} = \text{row}(f_r^i)$  as follows,

$$\tilde{m}_j = \left| \left\{ f_s^j \mid [X_{ij}]_{\bar{r}, \bar{s}} > 0, \text{ with } \bar{s} = \text{col}(f_s^j) \right\} \right|. \quad (2.12)$$

If we let  $\tilde{m}_\star$  be the maximum  $\tilde{m}_j$  for  $j \in \{1, \dots, n\}$ , then the root robot  $i_\star$  and root features  $f_{r_\star}^{i_\star}, f_{r_\star'}^{i_\star}, \dots$  for the association set of  $f_r^i$  with  $\bar{r} = \text{row}(f_r^i)$  are

$$i_\star = \min \{j \mid \tilde{m}_j = \tilde{m}_\star\}, \quad \{r_\star, r_\star', \dots\} = \left\{ s \mid [X_{ii_\star}]_{\bar{r}, \bar{s}} > 0 \text{ with } \bar{s} = \text{col}(f_s^{i_\star}) \right\}. \quad (2.13)$$

When  $f_r^i$  belongs to a consistent set, the root  $i_\star$  corresponds to the robot with a single feature  $f_{r_\star}^{i_\star}$  in the association set that has the lowest identifier,

$$\begin{aligned} i_\star &= \min \left\{ j \mid [X_{ij}]_{\bar{r}, \bar{s}} > 0 \text{ for some } \bar{s} \in \{1, \dots, |\tilde{\mathcal{S}}_j|\} \right\} \\ r_\star &= \left\{ s \mid [X_{ii_\star}]_{\bar{r}, \bar{s}} > 0 \text{ with } \bar{s} = \text{col}(f_s^{i_\star}) \right\}, \end{aligned} \quad (2.14)$$

where  $\bar{r} = \text{row}(f_r^i)$ . Robot  $i$  assigns to its feature  $f_r^i$  the label  $L_r^i = (i_\star, r_\star)$  and removes it from the set of unlabeled features,

$$L_r^i = (i_\star, r_\star), \quad \mathcal{L}_i = \mathcal{L}_i \cup L_r^i, \quad \tilde{\mathcal{S}}_i = \tilde{\mathcal{S}}_i \setminus \{f_r^i\}. \quad (2.15)$$

Thus, all features in the association set are assigned the same label. The robots proceed with all its consistent features in a similar fashion. For the features classified as conflictive, the resolution method (Algorithm 2.4.1) presented in the previous section is executed to solve the inconsistencies.

Let each component  $\mathcal{C}_q$  in Algorithm 2.4.1 have the identifier  $(i_\star, r_\star)$  composed of the root robot  $i_\star$  and root feature  $r_\star$  responsible of creating the component. When the resolution algorithm finishes, each feature  $f_r^i$  that has been assigned to a component  $(i_\star, r_\star)$  has become consistent due to the edge removals. We say that such features are *resolved*. Thus, all the resolved features with the same component id form a consistent association set. Each robot  $i$  uses the component id of  $f_r^i$  as its label,

$$L_r^i = (i_\star, r_\star), \quad \mathcal{L}_i = \mathcal{L}_i \cup L_r^i, \quad \tilde{\mathcal{S}}_i = \tilde{\mathcal{S}}_i \setminus \{f_r^i\}. \quad (2.16)$$

Additionally, due to edge removal, some unlabeled features  $f_r^i \in \tilde{\mathcal{S}}_i$  may have become exclusive. Robot  $i$  detects such features  $f_r^i$  by checking that

$$[\tilde{A}_{ij}]_{\bar{r}, \bar{s}} = 0, \quad \text{for all } j \in \mathcal{N}_i, \ j \neq i, \text{ all } \bar{s} \in \{1, \dots, |\tilde{\mathcal{S}}_j|\},$$

being  $\bar{r} = \text{row}(f_r^i)$ , and it manages them as in (2.10). The remaining features may still be conflictive. Each robot  $i$  executes a new detection-resolution iteration on these still unlabeled features  $\mathcal{S}_i$ .

In a finite number of iterations, all features of all robots have been labeled, and the algorithm finishes. The interest of the presented algorithm is that it is fully distributed and works on local information. Each robot  $i$  uses its own  $X_{ij}$  to classify its features.

## 2.6 Algorithm Based on the Maximum Error Cut

The previous resolution algorithm has the advantage of solving all the inconsistencies in an easy way. However, the algorithm does not use information about the quality of the matches. When this information is available, it can be used to select which links should be broken to get rid of the inconsistent associations.

Most of the matching functions in the literature are based on errors between the matched features. These errors can be used to find a better partition of  $\mathcal{C}$ . Let  $\mathbf{E}$  be the weighted symmetric association matrix

$$[\mathbf{E}]_{r,s} = \begin{cases} e_{rs} & \text{if } [\mathbf{A}]_{r,s} = 1, \\ -1 & \text{otherwise,} \end{cases} \quad (2.17)$$

with  $e_{rs}$  the error of the match between  $f_r$  and  $f_s$ .

**Assumption 4** The error between matches satisfies:

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

□

Since the inconsistency is already known there is no need to use the whole matrix but just the submatrix related with the inconsistency,  $\mathbf{E}_{\mathcal{C}}$ . Although all the errors in  $\mathbf{E}_{\mathcal{C}}$  are small enough to pass the matching between pairs of images, we can assume that the largest error in the path between two conflictive features is, with most probability, related to the spurious match.

**Definition 5** Given two conflictive features, we define a bridge as a *single link* whose deletion makes the conflict between those two features disappear. □

Note that not all the links in one inconsistency are bridges. There are links that, if deleted, would not break the inconsistency because:

- They do not belong to the path between the features to separate;
- They belong to the path, but they also belong to a cycle in the association graph, and therefore, they are not bridges.



Our goal is, for each pair of conflictive features, find and delete the bridge that links them with the maximum error.

Algorithm 2.6.1 shows a solution to find the bridges using local interactions. Along the section we explain in detail how it works. As we did in the detection algorithm (2.4), let each robot initialize its own rows of elements as  $\mathbf{z}_r(0) = \{[\mathbf{E}_{\mathcal{C}}]_{r,1}, \dots, [\mathbf{E}_{\mathcal{C}}]_{r,c}\}$ ,  $r \in \{1, \dots, \tilde{m}_i\}$ . Each robot manages the  $\tilde{m}_i$  rows corresponding to the conflictive features it has observed. The update rule executed by every robot and every feature is

$$\mathbf{z}_r(t+1) = \max_{s \in \mathcal{C}, [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0} (\mathbf{z}_r(t), \mathbf{z}_s(t) \mathbf{P}_{rs}), \quad (2.18)$$

where the maximum is done element to element and  $\mathbf{P}_{rs}$  is the permutation matrix of the columns  $r$  and  $s$ . We have dropped the super indices corresponding to robots because the limited communications are implicit in the error caused by direct associations, Eq. (2.17).

---

**Algorithm 2.6.1** *Maximum Error Cut - Robot  $i$* 


---

**Require:** Set of  $\mathcal{C}$  different conflictive sets

**Ensure:**  $\mathcal{G}_{dis}$  is conflict free

```

1: for all  $\mathcal{C}$  do
2:   Error transmission
3:    $\mathbf{z}_r(0) = \{[\mathbf{E}_{\mathcal{C}}]_{r,1}, \dots, [\mathbf{E}_{\mathcal{C}}]_{r,c}\}$ ,  $r = 1, \dots, \tilde{m}_i$ 
4:   repeat
5:      $\mathbf{z}_r(t+1) = \max_{s \in \mathcal{C}, [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0} (\mathbf{z}_r(t), \mathbf{z}_s(t) \mathbf{P}_{rs})$ 
6:   until  $\mathbf{z}_r(t+1) = \mathbf{z}_r(t)$ ,  $\forall r \in \tilde{m}_i$ 
7:   Link Deletion
8:   while robot  $i$  has conflictive features  $r$  and  $r'$  do
9:     Find the bridges  $(s, s')$  :
10:      (a)  $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'}$ ,  $s \neq s'$ ,
11:      (b) For all  $s'' \neq s$ ,  $[\mathbf{z}_r]_s \neq [\mathbf{z}_r]_{s''}$ ,
12:      (c) For all  $s'' \neq s'$ ,  $[\mathbf{z}_{r'}]_{s'} \neq [\mathbf{z}_{r'}]_{s''}$ 
13:     Select the bridge with largest error
14:     Send message to break it
15:   end while
16: end for

```

---

**Proposition 4** *The dynamic system defined in (2.18) 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 than  $\mathbf{z}_s \mathbf{P}_{rs}$ .*

*Proof* 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 [6]. For any  $r, s \in \mathcal{C}$  such that  $[\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0$ , by Eq. (2.18) 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

$$\mathbf{z}_r \geq \mathbf{z}_s \mathbf{P}_{rs} \text{ and } \mathbf{z}_s \geq \mathbf{z}_r \mathbf{P}_{sr} \quad (2.19)$$

Using the properties of the permutation matrices,  $\mathbf{P}_{rs} = \mathbf{P}_{sr} = \mathbf{P}_{sr}^{-1}$ , we see that  $\mathbf{z}_s \mathbf{P}_{rs} \geq \mathbf{z}_r$ , which combined with Eq. (2.19) yields to  $\mathbf{z}_r = \mathbf{z}_s \mathbf{P}_{rs}$ .  $\square$

Let us see the convergence values of the different elements. Considering again Eq. (2.18) for a given feature  $f_r$ , we can express it as a function of its elements and the  $u$ th component,  $[\mathbf{z}_r(t+1)]_u$ , is updated as follows:

$$\begin{aligned} & [\mathbf{z}_r(t+1)]_u \\ &= \begin{cases} \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_s) & \text{if } [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0 \wedge u = r \\ \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_r) & \text{if } [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0 \wedge u = s \\ \max([\mathbf{z}_r(t)]_u, [\mathbf{z}_s(t)]_u) & \text{if } [\mathbf{E}_{\mathcal{C}}]_{r,s} \geq 0 \wedge r \neq u \neq s \end{cases}, \end{aligned} \quad (2.20)$$

where the two first rows are due to the permutations. Let us first analyze the case in which the inconsistency does not contain any cycle.

**Theorem 3** *If  $\mathcal{C}$  is cycle free, then:*

- (i) *For any  $r \in \mathcal{C}$ ,  $[\mathbf{z}_r(t)]_r = 0, \forall t \geq 0$ .*
- (ii)  *$[\mathbf{z}_r(t)]_{s'} \rightarrow [\mathbf{E}_{\mathcal{C}}]_{r',s'} = e_{r's'}$ , where*

$$r' = \arg \min_{[A]_{r'',s'}=1} d(r, r''),$$

*and  $d(r, r'')$  is the distance in links to reach node  $r''$  starting from node  $r$ . In other words,  $f_{r'}$  is the closest feature to  $f_r$  directly associated to  $f_{s'}$ .*

*Proof* For any feature,  $f_r$ , taking into account Eq. (2.20), the update of the  $r$ th element of  $\mathbf{z}_r$ ,  $[\mathbf{z}_r(t+1)]_r$ , is

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

Recalling the first point in Assumption 4, the initial value of  $[\mathbf{z}_r(0)]_r = e_{rr} = 0$ , for all  $r$ , then  $[\mathbf{z}_r(t)]_r = 0, \forall t \geq 0$ .

The inconsistency does not have any cycles and there is a path between any two features, the conflict is a spanning tree. Let us consider one link,  $(f_{r'}, f_{s'})$ . The link creates a partition of  $\mathcal{C}$  in two strongly connected, disjoint subsets

$$\mathcal{C}_{r'} = \{r \mid d(r, r') < d(r, s')\},$$

$$\mathcal{C}_{s'} = \{s \mid d(s, s') < d(s, r')\}.$$

In the above equations it is clear that  $r' \in \mathcal{C}_{r'}$  and  $s' \in \mathcal{C}_{s'}$ .

We will focus now on the values of the  $s'$ th element of the state vector for the nodes in  $\mathcal{C}_{r'}$  and the  $r'$ th element for the nodes in  $\mathcal{C}_{s'}$ ,

$$[\mathbf{z}_r(t)]_{s'}, r \in \mathcal{C}_{r'}, \text{ and } [\mathbf{z}_s(t)]_{r'}, s \in \mathcal{C}_{s'}.$$

In the first case, for any  $r \in \mathcal{C}_r \setminus r'$ , update rule (2.20) is equal to

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

because  $r \neq s' \neq r''$ . The nodes in  $\mathcal{C}_{s'}$  are not taken into account because that would mean that  $\mathcal{C}$  has a cycle. The special case of feature  $f_{r'}$  has an update rule equal to

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

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

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

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

$$\max_{r \in \mathcal{C}_{r'}, s \in \mathcal{C}_{s'}} ([\mathbf{z}_r(0)]_{s'}, [\mathbf{z}_s(0)]_{r'}).$$

Since all the features  $r \in \mathcal{C}_{r'} \setminus r'$  are not associated with  $f_{s'}$ ,  $[\mathbf{z}_r(0)]_{s'} = -1$ . Analogously, for all the features  $s \in \mathcal{C}_{s'} \setminus s'$ ,  $[\mathbf{z}_s(0)]_{r'} = -1$ . Finally, for the features  $r'$  and  $s'$ , by the second and third point of Assumption 4,  $[\mathbf{z}_{r'}(0)]_{s'} = e_{r',s'} = e_{s',r'} = [\mathbf{z}_{s'}(0)]_{r'} \geq 0 > -1$ . Therefore, this subset of  $c$  elements of the state vectors converge to the error of the link  $(f_{r'}, f_{s'})$ ,  $e_{r',s'}$ . From Proposition 4 we can also see that for any  $r \in \mathcal{C}_{r'}$ ,  $[\mathbf{z}_r]_s$ ,  $s \in \mathcal{C}_{s'} \setminus s'$ , will converge to the final value of  $[\mathbf{z}_{s'}]_s$ .

The same argument applies for the rest of the links and the proof is complete.  $\square$

Let us see what happens now in the presence of cycles in the inconsistency.

**Theorem 4** *Let us suppose the inconsistency has a cycle involving  $\ell$  features. Let  $\mathcal{C}_\ell$  be the subset of features that belong to the cycle. After the execution of (2.18) it holds that:*

$$(i) \quad \forall r', s' \in \mathcal{C}_\ell, s' \neq r'$$

$$[\mathbf{z}_{r'}]_{s'} \rightarrow \max_{r,s \in \mathcal{C}_\ell} e_{rs}.$$

$$(ii) \quad \forall r' \notin \mathcal{C}_\ell, s' \in \mathcal{C}_\ell, s' \neq \arg \min_{s \in \mathcal{C}_\ell} d(r', s),$$

$$[\mathbf{z}_{r'}]_{s'} \rightarrow \max_{r,s \in \mathcal{C}_\ell} e_{rs}.$$

*Proof* In the proof, we will denote  $r_1, \dots, r_\ell$ , the set of features in  $\mathcal{C}_\ell$ . Without loss of generality we will assume that the links that form the cycle are  $(f_{r_1}, f_{r_2})$ ,  $(f_{r_2}, f_{r_3}), \dots, (f_{r_\ell}, f_{r_1})$ . For an easy reading of the proof of this result we will omit the time indices in the update equations. Let us consider the update rule (2.20) for element  $r_2$  of feature  $f_{r_1}$ ,

$$[\mathbf{z}_{r_1}]_{r_2} = \max([\mathbf{z}_{r_1}]_{r_2}, [\mathbf{z}_{r_2}]_{r_1}, [\mathbf{z}_{r_\ell}]_{r_2}),$$

where we have also omitted other possible features that are directly linked to  $f_{r_1}$  because if they are also linked to  $f_{r_2}$  they belong to  $\mathcal{C}_\ell$  and if not they do not affect to the final result.

From the above equation we observe that  $[\mathbf{z}_{r_1}]_{r_2}$  depends on the value of  $[\mathbf{z}_{r_2}]_{r_1}$ . At the same time this value is updated with

$$[\mathbf{z}_{r_2}]_{r_1} = \max([\mathbf{z}_{r_2}]_{r_1}, [\mathbf{z}_{r_1}]_{r_2}, [\mathbf{z}_{r_3}]_{r_1}),$$

which depends on the value of  $[\mathbf{z}_{r_3}]_{r_1}$ . If we keep with the chain of associations we reach the point in which  $[\mathbf{z}_{r_{\ell-1}}]_{r_1}$  depends on  $[\mathbf{z}_{r_\ell}]_{r_1}$ , which has update rule equal to

$$[\mathbf{z}_{r_\ell}]_{r_1} = \max([\mathbf{z}_{r_\ell}]_{r_1}, [\mathbf{z}_{r_{\ell-1}}]_{r_1}, [\mathbf{z}_{r_1}]_{r_\ell}).$$

As we have proved in Proposition 4, in the end  $[\mathbf{z}_{r_1}]_{r_2} = [\mathbf{z}_{r_2}]_{r_1}$ ,  $[\mathbf{z}_{r_2}]_{r_1} = [\mathbf{z}_{r_3}]_{r_1}, \dots, [\mathbf{z}_{r_{\ell-1}}]_{r_1} = [\mathbf{z}_{r_\ell}]_{r_1}$  and  $[\mathbf{z}_{r_\ell}]_{r_1} = [\mathbf{z}_{r_1}]_{r_\ell}$  because they are direct neighbors. This means that after the execution of enough iterations of (2.18),  $[\mathbf{z}_{r_1}]_{r_2} = [\mathbf{z}_{r_1}]_{r_\ell} = [\mathbf{z}_r]_{r_1}, \forall r \in \mathcal{C}_\ell \setminus r_1$ . By applying the same argument for any other feature in  $\mathcal{C}_\ell$  we conclude that after the execution of the update, for any  $r \in \mathcal{C}_\ell$ ,  $[\mathbf{z}_r]_{r'} = [\mathbf{z}_r]_{r''}, \forall r', r'' \in \mathcal{C}_\ell \setminus r$ . 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 links) and (i) is true. If there are any additional links inside the cycle the result is the same including in the max consensus the weights of these links.

Now let us consider the rest of the features in the inconsistency,  $\tilde{\mathcal{C}}_\ell = \mathcal{C} \setminus \mathcal{C}_\ell$ . Given a feature  $s \in \tilde{\mathcal{C}}_\ell$  two things can happen:

- $\exists$  unique  $r \in \mathcal{C}_\ell$  such that  $f_r$  and  $f_s$  are directly associated;
- $s$  is not directly associated with any feature in  $\mathcal{C}_\ell$  but there exists at least one path of features  $\in \tilde{\mathcal{C}}_\ell$  that ends in a unique feature  $r \in \mathcal{C}_\ell$ .

The uniqueness of  $r$  comes from the fact that if there were another feature  $r' \in \mathcal{C}_\ell$ , reachable from  $s$  without passing through  $r$ , that would mean that  $s$  is also part of the cycle. Note that this does not discard the possibility that  $r$  and  $s$  belong to another cycle different than  $\mathcal{C}_\ell$ .

As we have seen in the proof of Theorem 3, due to the fact that  $r$  is the only connection with  $\mathcal{C}_\ell$ , for any  $r' \in \mathcal{C}_\ell \setminus r$ ,  $[\mathbf{z}_s]_{r'}$  will have final value equal to  $[\mathbf{z}_r]_{r'}$  which proves (ii). On the other hand,  $[\mathbf{z}_s]_r$  will have the value of the link that connects it to feature  $r$  or, if  $f_r$  and  $f_s$  belong to another cycle different than  $\mathcal{C}_\ell$ , the maximum

error of all the links that form the second cycle. In both cases, doing a change in the names of the indices, we can see that (ii) is also true.  $\square$

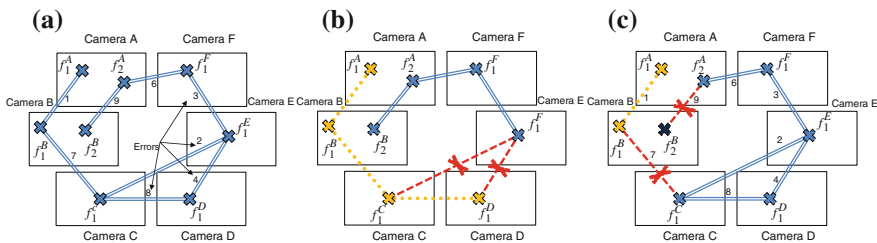
At this point we are ready to define the bridges in terms of the variables  $\mathbf{z}_r$  and to propose a criterion to select the bridge to break. The bridges,  $(f_s, f_{s'})$ , for any pair of conflictive features  $f_r$  and  $f_{r'}$  satisfy

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

The first condition comes from Theorem 3 and the other two come from Theorem 4. Note that for any bridge, the error of the bridge is the same as the value of  $[\mathbf{z}_r]_s$ ,  $[\mathbf{z}_r]_s = [\mathbf{z}_{r'}]_{s'} = e_{ss'}$ . Therefore, each node can look in a local way at its own rows and choose the best bridge that breaks the conflict, the one with the largest error. In case one robot has more than two features in the same conflict, finding the optimal cut becomes NP-hard. In this chapter, we use a greedy approach that returns good results. Our solution chooses two of the  $\tilde{m}_i$  inconsistent features and selects the best bridge for them. The bridge 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.

Note that we are considering only single-link deletions. Cycles in the association graph are sets of features strongly associated, and therefore, it is better not to break links there. If two conflictive features belong to the same cycle, then there are no bridges. However, the algorithm is also able to detect this situation and the *Spanning Trees* can be used to solve the conflict.

In conclusion, this algorithm is able to detect in a local way the best bridge to break each inconsistency. This provides a more solid criterion to solve the inconsistencies than just cutting arbitrary edges. Each robot is able to detect which set of links is best to cut in order to solve the conflicts regarding its own features. The algorithm also finishes in finite time and does not require much additional bandwidth because, as in the detection algorithm, the amount of transmitted information can be optimized. An example of execution of the algorithm is given in Fig. 2.3 and more details can be found in [18].

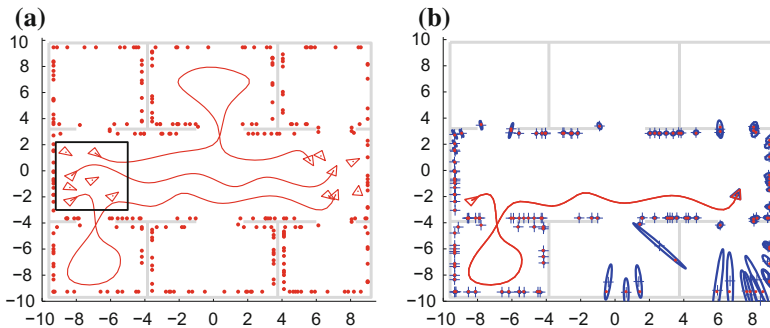


**Fig. 2.3** Example of execution of the resolution of one inconsistency using the two approaches. **a** Inconsistency. **b** Solution obtained using the *Spanning Trees* algorithm. **c** Solution obtained using the *Maximum Error Cut* approach

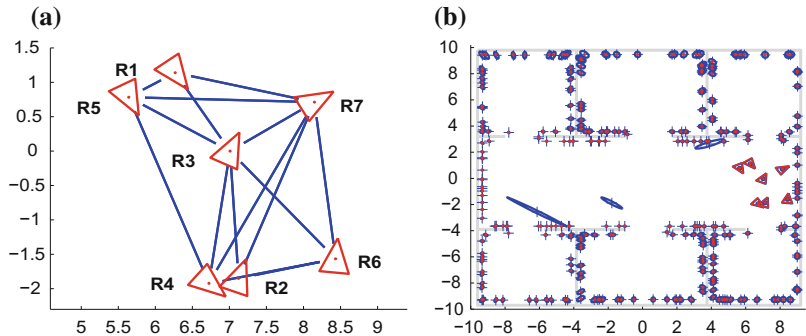
## 2.7 Simulations

We have carried out several simulations with a team composed by 7 robots exploring an environment of  $20 \times 20$  m with 300 features (Fig. 2.4). Each robot executes 70 motion steps along a path of approximately 30 m. The robots estimate their motion based on odometry information that is corrupted with a noise of standard deviation  $\sigma_x, \sigma_y = 0.4$  cm for the translations and  $\sigma_\theta = 1$  degree for the orientations. They sense the environment using an omnidirectional camera that gives bearing measurement to features within 360 degrees around the robot and within a distance of 6 m. The measurements are corrupted with a noise of 0.5 degrees standard deviation. Each robot explores the environment and builds its local map (Fig. 2.4b). Due to the presence of obstacles (gray areas), each robot may have not observed some landmarks.

When they finish the exploration, they execute the distributed data association algorithm explained in this chapter under the communication graph in Fig. 2.5a. The local data associations  $F(\mathcal{S}_i, \mathcal{S}_j)$  are obtained by applying the JCBB method [19] to the local maps of any pair of neighboring robots  $(i, j) \in \mathcal{E}_{com}$ . Since all the trajectories followed by the robots traverse the main corridor (Fig. 2.4) there is a high overlapping between their local maps (Table 2.1). Given any 2 local maps with approx. 122 features, there are approximately 89 true matches (ground truth). Although, the local data association method has found a high amount of the ground truth links (good links or true positives), it has also missed a few of them (missing links or false negatives). In addition, some additional links have been detected that link together different features (spurious links or false positives). From the 858 features within all the local maps, there are 300 different features in the ground truth sense (association sets). From them, 184 were observed by a single robot (ground truth exclusive features), and the remaining were observed by around 6 robots (ground truth size of the



**Fig. 2.4** A team of 7 robots explore an environment of  $20 \times 20$  m. **a** Gray areas are walls and red dots are the ground truth location of landmarks. The robots (triangles) start in the left (black box region) and finish in the right. **b** Local map estimated by robot 2. The landmarks close to its trajectory (red line) have been estimated (blue crosses and ellipses) with a high precision. Due to the presence of obstacles (gray areas) some of the landmarks have not been observed, or have been estimated with high uncertainty



**Fig. 2.5** **a** Communication graph associated to the final robot poses in Fig. 2.4. There is a link (blue solid line) between any pair of robot poses (red triangles) that are within a distance of 3 m. **b** Global map obtained after merging the local maps. Red dots and triangles are the ground truth position of the features and robot poses. The estimated feature positions are shown with blue crosses and ellipses. The map merging process is explained in detail in Chap. 4; here we display the global map estimated by robot 2 after  $t = 5$  iterations of the map merging algorithm

**Table 2.1** Local data associations

| Features             | Per local map          | Total        |
|----------------------|------------------------|--------------|
| Features observed    | 122                    | <b>858</b>   |
| Data associations    | Per pair of local maps | Total        |
| Links (ground truth) | 89                     | 2860         |
| Links                | 88                     | 2820         |
| Good links           | 85                     | 2750         |
| Missing links        | 3                      | 110          |
| Spurious links       | 2                      | 70           |
| Association sets     | Obtained               | Ground truth |
| Association sets     | <b>296</b>             | <b>300</b>   |
| Exclusive features   | <b>187</b>             | <b>184</b>   |
| Nonexclusive assoc.  | <b>109</b>             | 116          |
| Size of nonexclusive | 6.1                    | 5.8          |

nonexclusive). In the data association graph  $\mathcal{G}_{dis}$  however, only 296 association sets have been obtained, which means that different features have been mixed up together. There are 184 exclusive features (ground truth exclusive features), although the local data association algorithm has found 187 exclusive features. These additional three exclusive features appear due to the presence of the three outliers, the features with high covariance ellipses in Fig. 2.5b. Since their positions have been wrongly estimated, the local data association method has failed to correctly associate them.

The robots execute Algorithm 2.5.1 on the nonexclusive features to propagate the local matches and discover the associations between their features and the ones

**Table 2.2** Detection and resolution of inconsistent associations

| Detection        | Conflictive | Consistent nonexclusive | Consistent exclusive |
|------------------|-------------|-------------------------|----------------------|
| Association sets | 7           | 102                     | 187                  |
| Features         | 80          | 591                     | 187                  |
| Resolution       | Conflictive | Consistent nonexclusive | Consistent exclusive |
| Association sets | 0           | 116 (+14)               | 187                  |
| Features         | 0           | 671 (+80)               | 187                  |

observed by the other team members. In addition, they establish the labels for their features, and they detect and solve any inconsistent associations. From the 109 nonexclusive association sets, 102 of them are consistent, and its associated 591 features are classified as consistent (Table 2.2). The remaining seven sets are conflictive, and they have associated 80 conflictive features. After executing the resolution algorithm on the 80 conflictive features, all of them are resolved and the process finishes. The original seven conflictive sets are partitioned into 14 consistent nonexclusive sets. Due to these additional sets, the number of consistent nonexclusive association sets (Table 2.2, third row), which initially was 102 (Table 2.2, first row), is increased into 116 ( $102 + 14$ ) after executing the algorithm. Equivalently, the number of consistent nonexclusive features (Table 2.2, fourth row) which was 591 (Table 2.2, second row) becomes 671 ( $591 + 80$ ) since the 80 inconsistent features are resolved.

Table 2.3 compares the final data association graph and the ground truth information. Since the resolution algorithm is based on link deletion, the number of links here is lower than in Table 2.1. However, the number of association sets is closer to the ground truth results. From the 303 obtained association sets, three of them

**Table 2.3** Results after detecting and solving the inconsistencies

| Features             | Per local map          | Total        |
|----------------------|------------------------|--------------|
| Features observed    | 122                    | 858          |
| Data associations    | Per pair of local maps | Total        |
| Links (ground truth) | 89                     | 2860         |
| Links                | 87                     | 2794 (−26)   |
| Good links           | 85                     | 2746 (−4)    |
| Missing links        | 3                      | 114 (+4)     |
| Spurious links       | 2                      | 48 (−22)     |
| Association sets     | Obtained               | Ground truth |
| Association sets     | 303                    | 300          |
| Exclusive features   | 187                    | 184          |
| Nonexclusive assoc.  | 116                    | 116          |
| Size of nonexclusive | 5.7                    | 5.8          |



are due to the three outliers in Fig. 2.5b. Thus, there are 300 remaining association sets, which is exactly the same number of association sets in the ground truth data. The same behavior is observed regarding their sizes. This means that the resulting associations are similar to the ground truth ones in spite of the fact that they have less links. From the 26 links erased from  $\mathcal{G}_{dis}$ , 22 were spurious links, and only 4 were good links that now are missing. Robots use the obtained data association for computing the global map (Fig. 2.5b) as described in Chap. 4.

## 2.8 Closure

In this chapter, we have presented a distributed technique to match sets of features observed by a team of robots in a consistent way under limited communications. Local associations are found only within robots that are neighbors in the communication graph. After that, a fully distributed method to compute all the paths between local associations is carried out, allowing the robots to detect all the inconsistencies related with their observations. For every conflictive set detected, in a second step the method is able to delete local associations to break the conflict using only local communications. The whole method is proved to finish in a finite amount of time finding and solving all the inconsistent associations. We have studied the performance of the method for robots equipped with omnidirectional cameras in a simulated environment. Additional experiments with real data acquired with RGB-D and conventional cameras are presented in Chap. 5.

## References

1. R. Aragues, E. Montijano, C. Sagües, Consistent data association in multi-robot systems with limited communications, in *Robotics: Science and Systems*, Zaragoza, Spain, June 2010
2. R. Aragues, J. Cortes, C. Sagües, Distributed consensus algorithms for merging feature-based maps with limited communication. *Robot. Auton. Syst.* **59**(3–4), 163–180 (2011)
3. S. Avidan, Y. Moses, Y. Moses, Centralized and distributed multi-view correspondence. *Int. J. Comput. Vis.* **71**(1), 49–69 (2007)
4. T. Bailey, H. Durrant-Whyte, Simultaneous localization and mapping: part II. *IEEE Robot. Autom. Mag.* **13**(3), 108–117 (2006)
5. T. Bailey, E.M. Nebot, J.K. Rosenblatt, H. Durrant-Whyte, Data association for mobile robot navigation: a graph theoretic approach, in *IEEE International Conference on Robotics and Automation*, San Francisco, USA, April 2000, pp. 2512–2517
6. F. Bullo, J. Cortes, S. Martinez, *Distributed Control of Robotic Networks*. Applied Mathematics Series (Princeton University Press, Princeton, 2009), <http://coordinationbook.info>
7. C. Cadena, F. Ramos, J. Neira, Efficient large scale SLAM including data association using the Combined Filter, in *European Conference on Mobile Robotics*, Mlini/Dubrovnik, Croatia, September 2009, pp. 217–222
8. A. Censi, An accurate closed-form estimate of ICP's covariance, in *IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007, pp. 3167–3172
9. R.W. Deming, L.I. Perlovsky, Concurrent multi-target localization, data association, and navigation for a swarm of flying sensors. *Inf. Fusion* **8**(3), 316–330 (2007)

10. V. Ferrari, T. Tuytelaars, L. Van Gool, Wide-baseline multiple-view correspondences, in *IEEE International Conference on Computer Vision and Pattern Recognition*, Madison, USA, June 2003, pp. 718–725
11. M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**(6), 381–395 (1981)
12. D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, B. Stewart, Distributed multirobot exploration and mapping. *IEEE Proc.* **94**(7), 1325–1339 (2006)
13. A. Gil, O. Reinoso, M. Ballesta, M. Julia, Multi-robot visual SLAM using a rao-blackwellized particle filter. *Robot. Auton. Syst.* **58**(1), 68–80 (2009)
14. A. Howard, Multi-robot simultaneous localization and mapping using particle filters. *Int. J. Robot. Res.* **25**(12), 1243–1256 (2006)
15. M. Kaess, F. Dellaert, Covariance recovery from a square root information matrix for data association. *Robot. Auton. Syst.* **57**(12), 1198–1210 (2009)
16. K. Konolige, J. Gutmann, B. Limketkai, Distributed map-making, in *Workshop on Reasoning with Uncertainty in Robotics, International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, August 2003
17. H.S. Lee, K.M. Lee, Multi-robot SLAM using ceiling vision, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, USA, October 2009, pp. 912–917
18. E. Montijano, R. Aragues, C. Sagues, Distributed data association in robotic networks with cameras and limited communications. *IEEE Trans. Robot.* **29**(6), 1408–1423 (2013)
19. J. Neira, J.D. Tardós, Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robot. Autom.* **17**(6), 890–897 (2001)
20. C.H. Papadimitriou, K. Steiglitz, The max-flow, min-cut theorem (chapter 6.1), *Combinatorial Optimization: Algorithms and Complexity* (Dover Publications, New York, 1998), pp. 120–128
21. M. Pfingsthorn, B. Slamet, A. Visser, A scalable hybrid multi-robot SLAM method for highly detailed maps, in *RoboCup 2007: Robot Soccer World Cup XI*, Lecture Notes in Artificial Intelligence, vol. 5001, ed. by U. Visser, F. Ribeiro, T. Ohashi, F. Dellaert (Springer, Berlin, 2008), pp. 457–464
22. S. Thrun, Y. Liu, Multi-robot SLAM with sparse extended information filters, in *International Symposium of Robotics Research*, Italy, Sienna, October 2003, pp. 254–266
23. S.B. Williams, H. Durrant-Whyte, Towards multi-vehicle simultaneous localisation and mapping, in *IEEE International Conference on Robotics and Automation*, Washington, DC, USA, May 2002, pp. 2743–2748
24. X.S. Zhou, S.I. Roumeliotis, Multi-robot SLAM with unknown initial correspondence: the robot rendezvous case, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 2006, pp. 1785–1792

Parallel and Distributed Map Merging and Localization  
Algorithms, Tools and Strategies for Robotic Networks

Aragues, R.; Sagues, C.; Mezouar, Y.

2015, VIII, 116 p. 34 illus., Softcover

ISBN: 978-3-319-25884-3