Decentralized Methods for Cooperative Task Execution in Multi-robot Systems

Antonio Franchi

PhD Thesis

Advisor: Prof. Giuseppe Oriolo

Dottorato di Ricerca in Ingegneria dei Sistemi, XXII Ciclo
Coordinator: Prof. Carlo Bruni

Sapienza Università di Roma

Completed: December 8, 2009
Defended: February 18, 2010

Abstract

Research on multi-robot systems have received a lot of interest during recent years. In this thesis, a group of methods addressing some relevant multi-robot problems are presented. The first part deals with three environmental coverage problems. Chapter 1 describes the Sensor-based Random Graph (SRG) method for cooperative robot exploration. Chapter 2 describes the distributed visibility patrolling method, which can be viewed as a recurrent version of the exploration and is aimed to find optimal trajectories for a team of robots that must periodically visit a set of viewpoints in an environment with obstacles. In Chapter 3 the visibility-based pursuit-evasion method is presented, where a team of searchers must coordinate to guarantee detection of all evaders in an unknown environment while using only local information. The second part deals with localization and control problems in presence of anonymous measures. In particular Chapters 4 and 5 address the mutual localization, i.e., the estimation of the change of coordinates among the team members, using anonymous distance-bearing measures affected by false positives and false negatives. In Chapter 6 the anonymous-measurement setting is applied to the formation control field, addressing the encircling task.

to Izabela

Contacts

author:
antonio.franchi@mpg.tuebingen.de
http://www.antoniofranchi.com/robotics

advisor:
oriolo@dis.uniroma1.it
http://www.dis.uniroma1.it/~oriolo/

Contents

Introduction

The research on multi-robot systems have received a lot of interest during recent years, e.g., roughly 9-10% of the titles in IEEE International Conference on Robotics and Automation in the last decade refer to this topic. The field of application is large. The more common applications are *exploration and map building*, *sensor networks*, *surveillance*, *localization and tracking*, *transportation*, and *mobile infrastructures* (e.g., for communication).

Some of the main advantages in using a team of robots instead of a single robot are: improved time and space *performances*, due to the distributed task execution; a higher degree of *robustness*, due to a certain amount of independence between task completion and number of robots; the constructional *simplicity of the hardware*, based on the bio-inspired concept that many simple interacting units can act as a more complex organism. On the other hand, the development of algorithms for a practically effective multi-robot system is more complex and the related theoretical proofs are often harder to obtain. In fact, the concurrent evolution of many sub-systems and their mutual interaction, in the form of implicit physical relationships or explicit communication, must be taken into account. As a consequence, relevant design techniques for such robotics systems include decentralized control from control theory, distributed algorithms from computer science, wireless communication from telecommunications, multiagent cooperation from artificial intelligence, an so on.

In this thesis, a group of methods addressing five relevant multi-robot problems are presented. Chapter 1 describes the Sensor-based Random Graph (SRG) method for cooperative robot exploration, which performs a frontier-based exploration. It incrementally builds an exploration graph consisting in a fusion of many subgraphs created individually by each robot. The cooperation issues and the spatial conflict management are addressed in a distributed way. The sharing of portion of maps among different robots is essential in this algorithm. For this reason an estimate of the relative pose (mutual localization) is needed. This problem, which emerges also in many different contexts, is addressed in Chapters 4 and 5, in its relative and absolute version. In particular, inspired by the fact that our robots are equipped only with a range finder which is unable to distinguish the identity of the measured object, we have considered a more general class of mutual

localization problems. The mutual localization with anonymous position measures concerns the estimation of the whole team configuration using anonymous distance-bearing measures affected by false positives and false negatives. A multiple registration algorithm, a robust statistical paradigm called RANSAC, online data-association and EKF-filtering are used to disambiguate the team configurations and to reject erroneous measurements. On the other hand, Appendix A presents a classification of the ordinary mutual localization problems, in which identity is assumed known. In Chapter 6 we apply the anonymous-measurement setting to the formation control field, addressing the encircling task. In this case the mutual localization system result to be also useful to detect a non collaborative object which becomes a target to encircle. While the exploration is a batch task, since has a termination, the patrolling is a related task which requires to continuously monitor an area to detect intruders or other events. Chapter 2 describes the distributed visibility patrolling method. It finds optimal trajectories for a team of robots that must periodically visit a set of viewpoints in an environment with obstacles. This viewpoints may be though as a result of the SRG exploration algorithm. The robots are supposed to communicate only in visibility. The refresh time of the viewpoints (the maximum time between two consecutive visits) and the latency (the maximum time needed to transfer an information from each robot to any other robot) are considered as objective functions. Chapter 3 deals with another coverage task, the visibility-based pursuit-evasion method, where one or more searchers must coordinate to guarantee detection of all evaders in an unknown environment while using only local information. This task can be used to perform intruder detection, search and rescue, and other related security or monitoring tasks. The algorithm is based on the idea of maintaining complete coverage of the frontier between cleared and contaminated areas while expanding the cleared area. All these methods need a platform to be implemented on a real team of robots. Appendix B briefly describes the Multi-robot Integrated Platform we have implemented to make our experiments in a modular and efficient way.

Each chapter is organized in a similar self-contained way. First the scenario of the problem and its background in the literature are introduced. Second, the problem is stated in a formal way. Third the solution is presented, theoretically demonstrated and practically validated with simulations and/or experiments. Fourth the open points and new ideas are presented.

Part I

Cooperative Environmental Coverage

This part presents three multi-robot problems related by the common goal of covering an environment with the sensors.

In Chapter 1 the cooperative exploration problem is considered:

Problem (Cooperative Exploration) *Given a team of mobile robots with limited sensing and communication range*

- *cover once each point of an unknown area with at least one robot*
- *eventually build a model of the covered area (e.g., a map, a roadmap)*

Possible applications of the cooperative exploration are the discovering of static objects, map building, and planetary missions.

In Chapter 2 the cooperative optimal patrolling is considered:

Problem (Cooperative Optimal Patrolling) *Given a team of mobile robots with limited sensing and communication range*

- *continuously cover each point of a known area*
- *design the patrolling trajectories in order to minimize the maximum time*
  - *in which a point of the area remains unvisited (refresh time)*
  - *needed to exchange a message between any two robots using multi-hop (latency)*

The main differences with the exploration are that the environment is known (e.g., from a previous exploration) and hence the focus is on optimal time-planning, i.e., scheduling. Some possible application are the in the field of area keep-watching (e.g. oil-spill monitoring, forest-fire detection, track of border changes, surveillance of an explored area).

In Chapter 3 the cooperative pursuit evasion is considered:

Problem (Cooperative Pursuit-evasion) *Given a team of mobile robots with limited sensing and communication range*

- *cover once each point of an unknown area with at least one robot*
- *eventually detect any moving object in the covered area (e.g. lost person)*

The main differences with the exploration are that the presence of moving objects implies that the solutions must take care of recontamination and that the fact that no map is required implies that a finite-memory solution is expected. The main applications are the in the field of the discovering of moving objects (e.g., rescue (moving objects are victims to be saved), security (moving objects are invaders to be caught)).

Chapter 1

The Sensor-based Random Graph Method (SRG) for Cooperative Robot Exploration

This chapter presents the decentralized cooperative exploration strategy for a team of mobile robots equipped with range finders described in [1], endowed with recent improvements. A roadmap of the explored area, with the associate safe region, is built in the form of a Sensor-based Random Graph (SRG). This is expanded by the robots by using a randomized local planner which automatically realizes a trade-off between information gain and navigation cost. The nodes of the SRG represent view configurations that have been visited by at least one robot, and are connected by arcs that represent safe paths. These paths have been actually traveled by the robots or added to the SRG to improve its connectivity. Decentralized cooperation and coordination mechanisms are used so as to guarantee exploration efficiency and avoid conflicts. Simulations and experiments are presented to show the performance of the proposed technique.

1.1 Introduction

Exploration of unknown environments is one of the most challenging problems in robotics. This task typically requires a mobile robot to cover an unknown area while learning, at the same time, a model of the environment or locating a given object. A wide range of applications are conceivable, including automated surveillance, search-and-rescue operations, map building and planetary missions.

Using a multi-robot system has a number of potential advantages [2, 3]. A team of robots is expected to be able to complete an exploration task

faster than a single robot. Moreover, if a map of the environment is to be built, the redundant information provided by multiple robots can be used to increase the map accuracy and the quality of the localization [4]. To achieve these objectives, some sort of task decomposition and allocation is required. In practice, strategies to conveniently distribute robots over the environment should be devised so as to prevent the occurrence of spatial conflicts [5] and take advantage of the multi-robot architecture. Clearly, communication plays a crucial role in achieving a cooperative behavior with improved performance [6].

In most exploration strategies, the boundary between known and unknown territory (the *frontier*) is approached in order to maximize the information gain. A pioneering work for the multi-robot case is [7]: the robots merge the acquired information in a global gridmap of the environment, from which the frontier is extracted and used to plan individual robot motions. While this basic scheme lacks an arbitration mechanism preventing robots from approaching the same frontier region, in [8] it is proposed to negotiate robot targets by optimizing a utility function which takes into account the information gain of a particular region, the cost of reaching it and the number of robots currently heading there. The same decentralized frontier-based approach is used in [9], where a large-scale heterogenous team of mobile robots is used for exploration, mapping, deployment and detection tasks. In [10], the utility of a particular frontier region from the viewpoint of relative robot localization is also considered. In the incremental deployment algorithm of [11], robots approach the frontier while retaining visual contact with each other. An interesting multi-robot architecture in which robots are guided through the exploration by a market economy is presented in [12], whereas [13] proposes a centralized approach which uses a frontier-based search and a bidding protocol to assign frontier targets to the robots.

The present method builds on previous work [14, 15] on cooperative robot exploration based on local information only. In particular, the robots of the team cooperatively build a map of the environment in the form of a graph, called *Sensor-based Random Graph* (SRG), which is an evolution of the *Sensor-based Random Tree* (SRT) defined in [16, 17].

A *node* of the SRG contains a view configuration and the Local Safe Region (LSR) perceived from that location; an *arc* between two nodes represents a safe path between the corresponding configurations. The paths stored in the arcs may have been actually traveled by at least one of the robots, or added by joining directly connectable nodes to improve the connectivity of the roadmap. These special arcs are called *bridges* and essentially provide shortcuts.

With respect to [7, 8, 9, 10, 11, 12, 13], the distinctive aspects of the SRG method are the following.

- *Complete decentralization.* Each robot independently selects its next

destination towards the *local frontier* of its current LSR, thus approaching areas that appear to be unexplored on the basis of all the available information. This simple cooperation strategy is very efficient and automatically achieves a trade-off between information gain and navigation cost, without the need of defining and optimizing mixed utility functions.

- *Continuous replanning.* Since the LSR is bounded by the sensor perception range, the next destination of each robot is always nearby. These short-term plans are more secure, less binding and result in a more flexible decentralized task allocation, which can quickly adapt to new information that becomes available through communication. This also eliminates the necessity of enforcing artificial timeout conditions on the individual task execution.

- *Guaranteed coordination.* Another consequence of the short span of each robot plan is that coordination is needed to avoid conflicts only when the robots are close to each other. In particular, we are able to explicitly characterize this situation and provide guaranteed coordination strategies. In addition, a lower bound on the communication range that is needed to implement these strategies can be derived.

The chapter is organized as follows. Section 1.2 lists the assumptions under which the SRG method is presented. The SRG data structure is described in Sect. 1.3 . The architecture of the software implementing the exploration method on each robot is described in Sect. 1.4. Central in this architecture is the action planner, described in Sect. 1.5. The information encoded in the SRG stored in the memory of each robot is updated as explained in Sect. 1.6. The robots exchange information according to the communication protocol given in Sect. 1.7. In Sect. 1.8, some implementation issues are discussed. Finally, Sects. 1.9 and 1.10 present simulation and experimental results, respectively, of the proposed strategy.

1.2 Problem formulation

The cooperative SRG exploration method is presented under the following Assumptions.

1. The robots move in a planar workspace $\mathcal{W} \subseteq \mathbb{R}^2$.

2. Each robot is a disk of radius ρ, whose configuration q is described by the cartesian position of its center.

3. Each robot is *path controllable*, i.e., it may follow any path in its configuration space with arbitrary accuracy. This assumption is verified for free-flying as well as (most) nonholonomic mobile robots.

4. The robots are equipped with an omnidirectional sensory system which provides the *Local Safe Region* LSR(q), a description of the free space surrounding the robot at q. The LSR is a star-shaped subset of $\mathbb{R}^2$, whose maximum radius is bounded by the robot perception range R_p (Fig. 1.1).

5. Each robot can broadcast the information stored in its memory (or relevant portions of it) within a communication range R_c at any time. The robot ID number is included in the heading of any transmission. The robot is always open for receiving communication from other robots located inside R_c.

Many of these assumptions are only taken for simplicity and can be relaxed. The assumption of planar workspace is obviously not restrictive: 3D worlds are perfectly admissible as long as the sensory system allows the reconstruction of a *planar* LSR for planning the robot motion. Assumption 2 implies that the configuration space of each robot is a copy of the workspace with the obstacles grown so as to allow for the robot size [18]. This assumption is only taken for ease of presentation: the proposed method is readily applicable to robots with arbitrary shape. In Assumption 3, path controllability can be replaced with (simple) controllability provided that a *regional* path planner (i.e., an algorithm that generates feasible paths in a limited region) is available. Assumption 4, and in particular the star-shaped hypothesis, is consistent with the physics of the most common proximity sensors, i.e, range finders, but it also applies to more sophisticated perception techniques (e.g., panoramic vision).

At this stage, our exploration task can be informally defined as follows: the objective is *to cooperatively cover the largest possible portion of the workspace with sensor perceptions*. A more formal definition will be given in the following in connection with the termination condition for our method.

1.3 The sensor-based random graph (SRG)

The Sensor-based Random Graph (SRG) is a compact data structure used to represent the area explored by the team of robots as well as with the history of the exploration process in the form of a roadmap. Each node of the SRG represents a collision-free configuration q that has been visited by at least one robot. The result of the perception process at q is the associated Local Safe Region LSR(q). An arc between two nodes represents a collision-free path between the two configurations. This path may have been actually traveled by at least one robot, or added to the SRG to improve its connectivity (in this case it is called *bridge*, see Sect. 1.6).

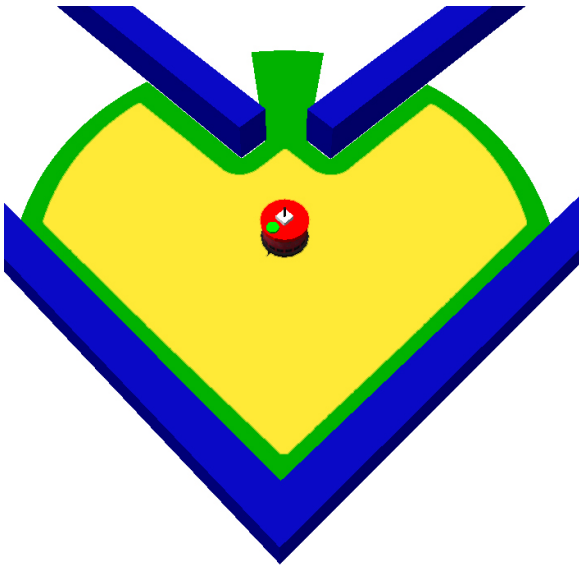

Figure 1.1: The lightly colored (green/yellow) area is the Local Safe Region (LSR) at the current configuration. Its lighter (yellow) subset is the Local Reachable Region (LRR). Obstacles are darkly colored (blue).

It should be emphasized that the SRG is a 'virtual' data structure, whose knowledge is *distributed* in the team [19]. The i-th robot knows its own version SRG_i of the graph. SRG_i is built by the robot on the basis of data acquired either by the robot itself or via communication with other robots. The SRG, which is the union of all the SRG_i's, is not explicitly represented at any level.

Each robot incrementally builds its own SRG_i by extending it in the most promising direction via a biased random mechanism. In doing this, it uses a local coordination strategy that takes into account the information coming from other robots in order to guarantee that the distributed knowledge is increased. As a result, the SRG as a whole is simultaneously extended in several directions towards currently unexplored areas.

For planning robot motions, the SRG method makes use of three structures that are directly derived from the LSR: the *Local Reachable Region* $\mathrm{LRR}(q)$, the *Local Frontier* $\mathrm{LF}(q)$ and the *Local Informative Region* $\mathrm{LIR}(q)$.

1.3.1 The local reachable region (LRR)

In the SRG method, the action planning domain is the robot configuration space and in particular the *Local Reachable Region* $\mathrm{LRR}(q)$, defined as the set of configurations that can be reached from q with the robot staying in $\mathrm{LSR}(q)$ (Fig. 1.1). Under Assumptions 2 and 3, the $\mathrm{LRR}(q)$ can be obtained by *eroding* the $\mathrm{LSR}(q)$ with the robot disk as structuring element [20], and

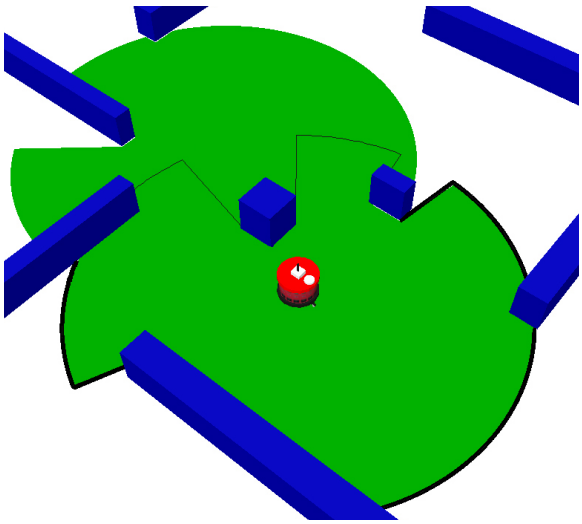

Figure 1.2: Frontier arcs (thick lines) and free arcs (thin lines) of the current LSR boundary.

then *extracting* the connected component containing q. The LRR is not star-shaped in general.

1.3.2 The Local Frontier (LF)

To identify promising exploration actions from the available information, the robot identifies the portion of the current LSR boundary leading to unexplored areas. To this end, the boundary ∂LSR is partitioned in *obstacle*, *free* and *frontier* arcs (see Fig. 1.2). An *obstacle arc* is a portion of the boundary of the obstacle region as detected from q. Under Assumption 4, these are reconstructed from the range scan by identifying contiguous readings that are smaller than the perception range R_p. Points of ∂LSR which fall in other LSRs stored in the SRG_i belong to *free arcs*. Any arc which is neither obstacle nor free is a *frontier arc*, and by construction identifies the transition from explored to unexplored regions. The union of the frontier arcs of $\mathrm{LSR}(q)$ is the *Local Frontier* $\mathrm{LF}(q)$.

A frontier arc of $\mathrm{LSR}(q)$ is computed from the whole SRG_i, which in turn is built using all the information available to the robot. This is the key to our decentralized cooperation mechanism aimed at optimizing the exploration performance of the team. Such mechanism is inherently local and contingent because it relies on communication between the robots.

1.3.3 The local informative region (LIR)

Given an $\mathrm{LSR}(q)$ and its $\mathrm{LRR}(q)$, a $q' \in \partial\mathrm{LRR}(q)$ is called a *local informative configuration* if there exists a point p on the Local Frontier $\mathrm{LF}(q)$ such that

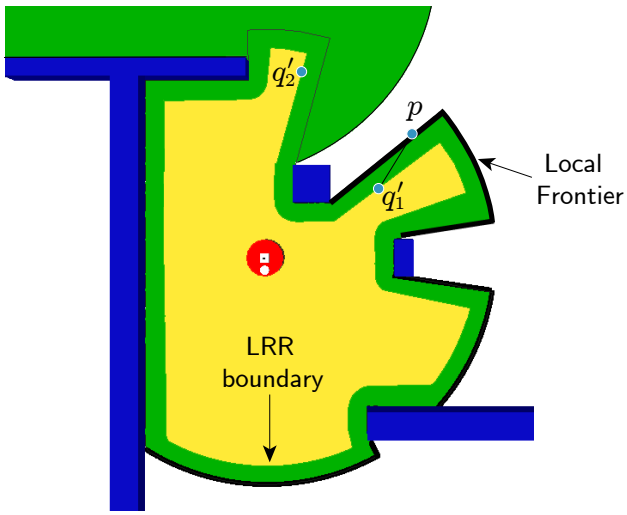

Figure 1.3: While q_1' is a local informative configuration, q_2' is not.

(see Fig. 1.3):

1. the open line segment (also called the *line of sight*) joining p and q' does not intersect the boundary $\partial\mathrm{LSR}(q)$;
2. $\|p - q'\| < R_p$.

The first condition guarantees that $p \in LF(q)$ is 'visible' from q' through a line of sight contained in $\mathrm{LSR}(q)$, while the second ensures that p is contained in the perception range at q'. Together they guarantee that a sensor scan taken from q' will 'push forward' the frontier arc containing p, thereby increasing the information about the explored workspace.

Conditions 1 and 2 may be satisfied also at configurations that belong to the interior of LRR. However, the above definition, according to which local informative configurations must be on the LRR boundary, aims at maximizing the information gain. A local informative configuration has positive *information gain* in the sense of [21, 22].

The *Local Informative Region* $\mathrm{LIR}(q)$ is the set of all local informative configurations; $\mathrm{LF}(q) = \emptyset$ implies $\mathrm{LIR}(q) = \emptyset$.

1.4 Functional architecture

For the sake of clarity, the SRG exploration will be described with reference to the functional architecture of the software running on each robot of the team, shown in Fig. 1.4. Blocks with thick edges represent processes, those with thin edges represent threads, and dashed rectangles represent data. Arrows indicate an information flow: thick for interprocess communication,

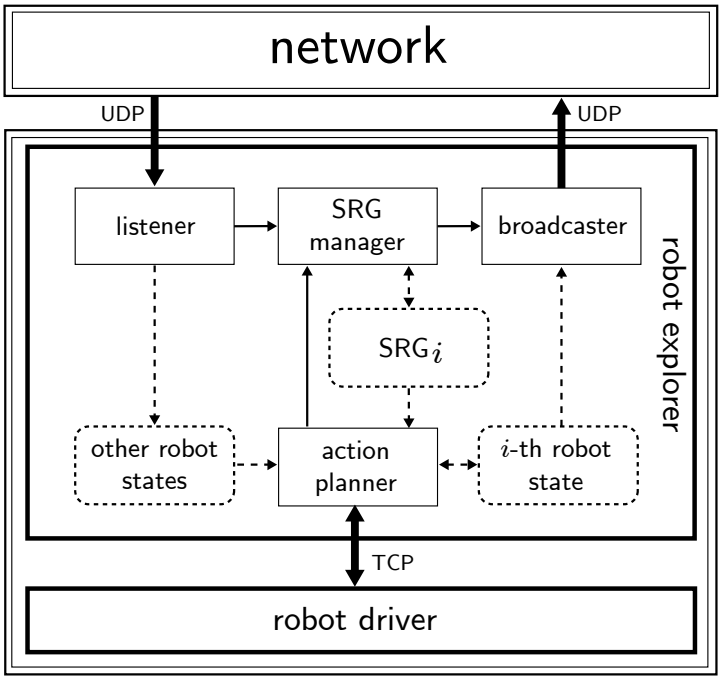

Figure 1.4: Functional architecture of the software implementing the SRG method on the i-th robot.

thin for communication between threads, dashed for read/write operation on data structures.

The *robot explorer* implements the SRG exploration algorithm, while the *robot driver* provides low-level primitives for motion, localization and perception (not discussed here). The two processes communicate through the TCP protocol, allowing a distributed instantiation of the architecture and providing a flexible integrated environment for simulation and experimental validation. With this architecture, in fact, the explorer and the driver do not need to run on the same machine, and the latter can be a real or a simulated robot.

The robot explorer is realized by four threads: the *action planner*, the *SRG manager*, the *broadcaster* and the *listener*. The action planner is the core of the robot explorer: it is in charge of choosing the next exploration action in a cooperative and coordinated way. The task of the SRG manager is to elaborate and continuously update the data stored in the SRG$_i$ on the basis of the information received from the action planner or, through the listener, from the rest of the team. In particular, the action planner makes available the LSRs acquired by the robot, while the listener provides the SRG$_j$'s built by other robots of the team with which communication has taken place. The SRG manager incorporates these data so as to maintain the consistency of local representations. To this end, self-localization and

mutual localization information coming from the robot driver (and, through the listener, from other robots) are used. Finally, the broadcaster transmits all the information currently available to the robot.

1.5 Action planner

The action planner for the i-th robot is described in pseudocode in Fig. 1.5. Its basic steps are first briefly discussed, and then detailed in the rest of the Section.

At the beginning, the robot is stationary in a position corresponding to a node of the SRG_i. The first operation that the robot performs is the identification of the *Group of Engaged Agents* (GEA), i.e, the other agents of the team with which cooperation and coordination are necessary. This is achieved by first building the *Group of Pre-engaged Agents* (GPA), i.e., the robots which are candidate to belong to the GEA, and synchronizing with them (line 1). These computations are performed by the robot on the basis of the information stored in its SRG_i and data coming from other robots (see Fig. 1.4). Once synchronization has been achieved, the action planner sends a 'perceive' command to the robot driver, receives from it the current LSR and makes it available to the SRG manager (line 2). When the perceptions of all the robots in the GPA have been received, the actual GEA can be built using simple geometry. Lines 1–3 are detailed in Sect. 1.5.1.

If the Local Informative Region LIR (computed by the SRG manager) of the current LSR is non-empty, the action planner selects a *target* configuration (also called *view* configuration in the following) on the LIR according to a randomized mechanism (line 5). A path reaching the target is then planned inside the current LRR (this guarantees that the path is safe, i.e., collision-free on the basis of the available knowledge). If the current LIR is empty, the action planner verifies whether there exists in SRG_i a node with non-empty LIR (line 6). In the positive case, it first finds the closest node with a non-empty LIR and plans a path leading to it on SRG_i (line 7). Then, it selects as target the first adjacent node on such path (line 8). See Sect. 1.5.2 for a commentary of lines 5–8.

After the target is selected, the robot checks if its GEA includes other robots. In the negative case, the robot directly moves to its target. Otherwise, the prospective paths of the robots in the GEA are checked for mutual collisions and accordingly classified in feasible and unfeasible paths (line 12, Sect. 1.5.3). If there are unfeasible paths, a GEA coordination phase takes place. A master robot is selected in the GEA, which may either confirm or modify (see Sect. 1.5.4) the current target of the robot. In particular, the robot move may be simply forbidden by resetting the target to its current configuration. Last, the target is received from the master (line 15) and the robot moves towards it (line 16).

Procedure `Action Planner`

```
1  build GPA and synchronize;
2  perceive LSR;
3  build GEA;
4  if LIR is non-empty then
5      select a target configuration on LIR and
       plan a path leading to it in the LRR;
6  else if there exists a node of SRG_i with non-empty LIR then
7      find the the closest node with non-empty LIR and
       plan a path on SRG_i leading to it;
8      set as target the first adjacent node on the path;
9  else
10     terminate exploration: homing;
11 if |GEA| > 1 then
12     check feasibility of the GEA robot paths;
13     if there are unfeasible paths then
14         choose a master in the GEA;
15         wait target from master;
16     issue 'move to target' command;
```

Figure 1.5: A pseudocode description of the action planner.

The action planner terminates the exploration process when the condition of line 9 is verified, i.e., when the LIRs of all nodes in the SRG_i are empty (hence, no local informative configurations remain in SRG_i). At this point, the robot enters a *homing* phase, in which it plans and follows a path on the SRG_i leading back to its starting configuration.

If all nodes of the SRG_i have empty Local Frontier, the above termination condition is obviously met. However, such condition may also be satisfied in the presence of non-empty LFs, as in the case of Fig. 1.6. Here, there is no further exploration action that will allow the robot to move/remove the LF (a typical situation when the workspace contains 'windows' across which the robot can 'see' but cannot 'pass').

It is also interesting to point out that, although our exploration task is cooperative in nature, the above termination condition is consistent with a decentralized approach, as it can be computed by each robot on its own. In fact, when no local informative configurations are left in its SRG_i, a robot can safely exit the exploration process, as it will never be able to contribute new perceived areas to the distributed SRG. Actually, our simulations and experiments have shown that, on the average, all robots tend to perform homing simultaneously, thereby supporting a claim of efficient exploration.

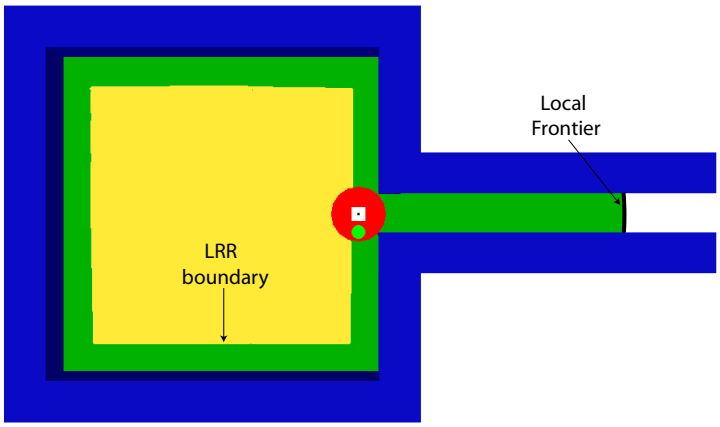

Figure 1.6: A typical situation in which the Local Informative Region is empty while the Local Frontier is not. Once the robot is in contact with the narrow passage boundary, no further exploration action will allow to move/remove the LF (the robot can 'see' but cannot 'pass' through the passage).

1.5.1 GPA/GEA construction

At the start of the action planner algorithm, the robot is stationary and needs to identify other robots whose LSRs may overlap with its own, in order to *cooperate* (avoid inefficient actions) and *coordinate* (avoid collisions) with them. The other robots may be stationary as well (in this case, their targets coincide with the current configuration) or moving towards a target; hence, a synchronization phase is needed.

Two robots are said to be *GPA-coupled* if the distance between their targets is at most $2R_p$, i.e., twice the perception range. The GPA of the robot is then built by grouping together all the robots to which it can be connected through a chain of GPA couplings (see Fig. 1.7, left). To achieve synchronization, the GPA is computed and updated until all its members are stationary; when this is achieved, the robot exits from this synchronization phase. If T is the maximum time required to reach the target, the upper bound of the waiting time is $(N-1)T$, where N is the number of robots of the team. T is bounded; in fact, the target configuration is at a distance at most $R_p - \rho$, since it is always in the current LIR.

The communication range R_c clearly plays a role in the GPA construction. Since the maximum distance between the robot and any other robot with which it is GPA-coupled is $3R_p - \rho$ (the other robot may still be moving to its target, which however cannot be farther than $R_p - \rho$ from the current configuration), it is sufficient to assume $R_c \geq 3R_p - \rho$ to guarantee that the GPA accounts for all the robots that are candidate to belong to the GEA.

Once the robot is synchronized with its GPA (and all the LSRs of the

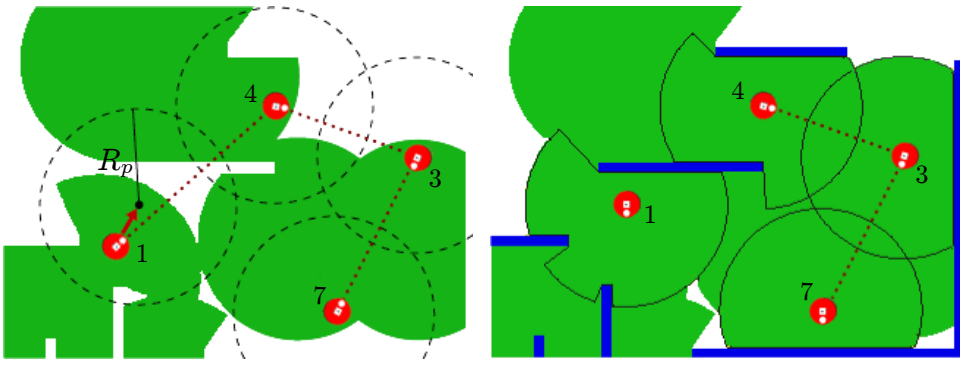

Figure 1.7: An example of GPA/GEA construction. *Left*: The GPA of robot 4 consists of robots 1,3,4,7: robot 1 is still moving towards its target point, while robots 3, 4 and 7 are stationary. The perception areas of the robots (prospective in the case of robot 1) overlap in pairs. *Right*: Once the LSR have been computed, only robots 3, 4, and 7 belong to the GEA of robot 4 since their LSRs overlap in pairs.

other robots in the GPA are available), it builds the GEA, i.e., the robots with which cooperation and coordination are actually necessary. If we define two robots to be *GEA-coupled* when their LSRs overlap, the GEA of the robot (see Fig. 1.7, right) is composed by all the GPA robots to which it can be connected through a chain of GEA couplings. Synchronization guarantees that all the GPA robots are stationary when the GEA is computed. The GEA is *symmetric*, i.e., it is the same for all robots in the group.

The GEA is a cornerstone of our method, as it identifies a group of robots that, in view of their vicinity, spontaneously agree to cooperate and coordinate with each other on a temporary basis. Such agreement can be reached with a limited communication range ($R_c \geq 3R_p - \rho$).

1.5.2 Target and path generation

If the current Local Informative Region is non-empty, the action planner chooses a target configuration in the LIR using a randomized mechanism.

In particular, the LIR is in general the union of disjoint arcs $a_1, a_2, ..., a_m$. First, one of these arcs is selected with a probability proportional to its length. Let a be the selected arc and s be the arc length parameter along a, with $s \in [0, L]$. The target configuration is selected on a by generating a random value s^* according to a normal distribution with mean value $L/2$ and standard deviation $\sigma = L/6$, and taking the configuration identified by $s = s^*$. At this point, a path to the target can be planned in the current LRR; note that, in view of Assumption 3, any such path can be executed by the robot (and is collision-free by definition).

A deterministic version of this target selection procedure can be envis-

aged, in which a specific configuration in the LIR is selected according to some fixed criterion, e.g., maximum information gain. However, our experience indicates that using this strategy the computational load is 4 or 5 times worse than our randomized version, which is not justified by the improvement in performance (traveled distance and number of views), which is lower than 5% on the average.

Thanks to the GPA synchronization phase, all the robots in a GEA plan at the same time, and therefore the cooperation mechanism encoded in the notion of Local Frontier is enforced on all the group. This 'agreement of intents' is realized without any centralized decision module.

1.5.3 GEA path feasibility check

Although the current Local Frontier of a robot cannot belong to the LSR of another robot of the GEA (see Fig. 1.2), the two prospective paths may still intersect (see Fig. 1.8).

Let $\mathcal{G}$ be the set of robots in the GEA. The prospective paths of the robots of $\mathcal{G}$ are checked to establish whether they are simultaneously feasible, i.e., they do not lead to collisions (for simplicity, the possibility of velocity scaling along the paths is not considered). All pairs of paths that intersect are identified, and the corresponding robots stored in the GEA *unfeasible subset* $\mathcal{G}_u$. The remaining robots are the GEA *feasible subset* $\mathcal{G}_f$. The complexity of this check is $O(|\mathcal{G}|^2)$.

1.5.4 Coordination

If the subset $\mathcal{G}_u$ of robots with unfeasible paths is non-empty, a coordination phase takes place locally. At first, a *master robot* within $\mathcal{G}$ is elected[1]. This can be accomplished in many ways through a deterministic procedure known by all the robots; for instance, the robot with the higher ID number can be chosen. Two cases are then possible:

1) If the robot is the master, it builds the vector of targets $\mathcal{Q}_\mathcal{G}$ which collects the target configurations received from the GEA robots. Then, it rearranges this vector so as to obtain a feasible collective motion. Here, rearrange may mean either simply accepting/resetting the target of a robot to the current configuration (i.e, authorizing/forbidding the move) or adding a third option, i.e., changing it to a new target. Correspondingly, we have devised two strategies, i.e., *coordination via arbitration* and *coordination via replanning* (see below).

2) If the robot is not the master, it waits for until the receipt of a specific signal from the master.

[1] A master robot is not actually needed when a deterministic coordination algorithm is chosen; in fact, in this case each robot can run the algorithm on its own, reaching the same solution as the others.

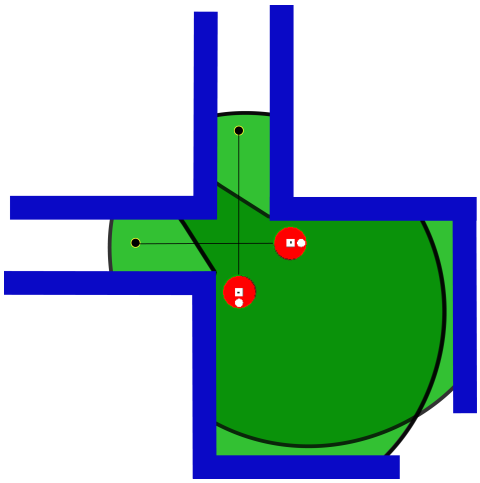

Figure 1.8: The prospective paths of robots belonging to the GEA may intersect as each of them tries to move towards its LF.

The final operation is to retrieve and return the robot's (possibly modified) own target from $\mathcal{Q}_{\mathcal{G}}$.

Coordination via arbitration

This strategy implements a simple arbitration mechanism on $\mathcal{G}$. All the robots contained in the feasible subset $\mathcal{G}_f$ are allowed to move (their target configuration is left unchanged). The robots in the unfeasible subset $\mathcal{G}_u$ are not allowed to move (their target is reset to the current configuration) with the exception of a single one whose motion is authorized (by construction, this strategy is guaranteed not to produce conflicts).

The selection of the authorized robot in $\mathcal{G}_u$ may be done on the basis of various criteria. The one we have used chooses randomly one of the robots (if any) whose LIR is empty. This strategy is motivated by the fact that, if their move is not authorized, such robots will have to wait for their path to become clear, as they cannot change their target (as opposed to robots whose LIR is non-empty, to which the random planner may propose a different target in the next cycle). An antithetical criterion would be to use a probability proportional to the LIR extension to choose randomly a robot in $\mathcal{G}_u$.

Coordination via replanning

This strategy tries to modify the targets of the robots in $\mathcal{G}$ so as to maximize the number of simultaneous feasible moves. This may be done by formalizing the problem as follows.

Consider the set of targets $\mathcal{Q}_{\mathcal{G}}$ associated to the GEA $\mathcal{G}$. Two targets in $\mathcal{Q}_{\mathcal{G}}$ are called *compatible* if they can be reached by the corresponding robots with paths that do not intersect. Let G be the *compatibility graph* associated to $(\mathcal{G}, \mathcal{Q}_{\mathcal{G}})$ and defined as the indirect graph whose nodes represent the robots in $\mathcal{G}$ and whose arcs join pairs of nodes with compatible targets. A *maximum clique* of G is a complete subgraph of G with maximum cardinality, corresponding to a maximum subset of robots with compatible targets. The identification of a *maximum clique* is a well-known NP-complete problem in the context of the graph theory [23, 24].

The ideal objective of the replanning strategy is to modify the set of targets $\mathcal{Q}_{\mathcal{G}}$ so as to maximize the cardinality of the associated maximum clique(s), with the constraint that the target of each robot is either accepted, changed to another configuration on the LIR (if this is non-empty) or to the current robot configuration (the move is not authorized). This is a very complex problem whose solution would require the computation of maximum cliques as a subproblem. To find a satisfactory solution in a given amount of time, we have adopted a randomized search technique, performed by the master as a sequential game with complete information.

1.6 SRG manager

The SRG manager updates SRG_i on the basis of the new information it receives, which consists of one or more nodes to be added to the graph. A node is a view configuration and comes with the associated LSR, a list of adjacent nodes and the corresponding arcs. Information may reach the SRG manager via two different routes (see Fig. 1.4). Views gathered by the i-th robot itself come from the robot driver through the action planner whereas those collected by other robots are received through the listener. Since each robot uses its own reference frame, views arriving via the second route must undergo a preliminary rototranslation, which is computed on the basis of mutual localization data.

For each new node, the SRG manager:

1. adds the node to SRG_i or, if it is already present, updates its LSR;
2. computes the LRR, the LF and the LIR;
3. updates the LF and the LIR of the nodes in SRG_i whose LSRs have a non-empty intersection with $\mathrm{LSR}(q)$.

After updating the SRG_i, an important operation is performed aimed at improving the connectivity of the graph. In particular, for each new node v that has been added, the SRG manager identifies the set W made of all nodes w of SRG_i that satisfy the following conditions:

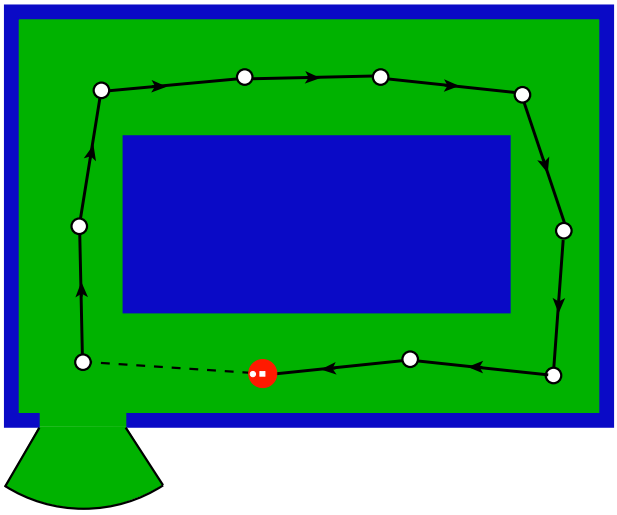

Figure 1.9: The importance of adding bridges should be clear from this example. Empty circles represent SRG nodes, while solid line segments are SRG arcs. Without the bridge (which is shown as a dashed line segment) the robot would have to trace back its path around the central obstacle in order to exit the room and continue the exploration.

1. the *graph distance* between v and w is greater than a certain threshold (typically, a small multiple of R_p);
2. $\mathrm{LRR}(v) \cap \mathrm{LRR}(w) \neq \emptyset$.

A *bridge* is then created for each pair (v, w), with $w \in W$, by planning a path joining the two nodes; note that a safe path between them certainly exists in view of the second condition. The first condition guarantees that only significant improvements to the graph connectivity are enforced; this avoids an excessive increase of the graph complexity.

Once a bridge has been created, it may be mapped to the SRG_i in two different ways, depending on the euclidean distance between v and w, which is bounded by $2(R_p - \rho)$. If $\|v - w\| < R_p - \rho$, the bridge directly becomes an arc. Otherwise, it is encoded as an arc-node-arc sequence, with the intermediate node placed inside $\mathrm{LRR}(v) \cap \mathrm{LRR}(w)$. In this way, we preserve the property that the distance between two adjacent nodes on the graph is bounded by $R_p - \rho$ (which we have used, for example, in Sect. 1.5.1).

At the time of its creation, a bridge has not been crossed by any robot of the team. Similarly, when a bridge is encoded as an arc-node-arc sequence, the intermediate node has not yet been visited by any robot.

Bridges essentially represent shortcuts that are very useful for performing an efficient exploration. In fact, as shown Fig. 1.5, every time the LIR of the current node is empty, the robot transfer itself to another node of the

SRG_i moving along the arcs of the SRG_i. If no bridge were added at all, as in Fig. 1.9, the connectivity of the free space would remain unexploited, resulting in longer paths during those transfers.

1.7 Broadcaster and listener

On each robot, the broadcaster and the listener respectively transmit and receive information. Conceptually, such information is organized in three possible messages:

- the *robot state*, i.e., its current configuration, target, GPA/GEA and step of the action planner being executed;
- the nodes between which an arc is to be created;
- the node, either new or already present in the SRG_i, with the associated LSR.

Each robot broadcasts its state on a regular basis, whereas its SRG data structure is only transmitted as new data is made available by the SRG manager. The listener receives asynchronous messages from the network and makes them available to the action planner (as other robot states) and the SRG manager. See again Fig. 1.4.

1.8 Implementation issues

Before presenting simulation and experiment results, we provide in this Section some details about our implementation of the SRG method.

Each Local Safe Region is stored in the form of an array of range readings, as returned by the robot range finder. Such an array is a discrete representation of the polar function which describes the LSR boundary. The corresponding Local Frontier LF can then be extracted as described in [17].

The Local Reachable Region can be efficiently built if a gridmap is used to represent the LSR (or the whole workspace). In this case, the LRR at q can be computed as follows:

1. represent the boundary and the interior of the LSR as occupied cells and empty cells, respectively;
2. apply an euclidean distance transform [25] to identify the set E of empty cells whose distance to the closest occupied cell is larger than ρ (the robot radius);
3. compute the LRR as the connected component of E which contains q.

Once the LRR is computed, the LIR can be obtained via a *ray casting* procedure. In particular, for each cell representing a configuration q' on the LRR boundary, the local frontier LF is inspected until a point p is found (if it exists) which satisfies the two conditions which identify local informative configurations (see Sect. 1.3.3). Heuristics can be used to speed up the search for such points at contiguous configurations.

In the implementation of the SRG method, it is also necessary for each robot to detect and remove *occlusions*, i.e., obstacle arcs that are caused by other robots rather than by environment obstacles. Candidate occlusions are identified directly from the range scan profile as protrusions of compatible size. They are then validated using the mutual localization method proposed in [26] and described in Chapter 5: occlusions that are attributed to actual robots are removed and re-classified as frontier arcs.

Other relevant implementation issues concern data transmission among the robots. First, we emphasize that the transmission of GPA/GEA information is necessary (and hence, performed) only if the communication range R_c is limited. Second, consider that the limitation of R_c does not mean that the cooperation and coordination area is accordingly limited — robots belonging to the same GPA/GEA may be farther than the communication range. In this case, a chain of communication is established to propagate the information between robots that are not in direct communication.

Moreover, in the broadcaster thread, it is not necessary to broadcast the whole SRG_i whenever it is modified. Simple strategies may be used to minimize the amount of transmitted information; for example, timestamps can be used by a robot to identity the portion of data which have been received so far. In this case, specific peer-to-peer communication strategies can be used to transmit and receive only new information.

1.9 Simulations

We present some simulations results to evaluate the performance of the SRG method. To assess the effectiveness of the notion of 'bridge', we have also implemented a version which does not add such structures; such version is referred to as SRT in the following, and essentially corresponds to the method described in [14]. The simulations are performed in Move3D [27], a software platform developed at LAAS-CNRS and dedicated to motion planning. The exploration team is composed of a varying number of MagellanPro robots. Each robot has a diameter of 0.40 m and carries a 360° laser range finder, with a perception range of 1.60 m. The communication range is set to its minimum admissible value, i.e., $R_c = 3R_p - \rho = 4.60$ m. Two nodes are candidate to be connected by a bridge if their graph distance is at least $3R_p$. Coordination is achieved via replanning.

The performance of the methods are evaluated in terms of *exploration*

time (the time required by the last robot of the team to return home) and *traveled distance* (the average distance traveled by each robot). These values are respectively expressed as a percentage of the values obtained with a team composed by a single robot using an SRT method. Environment coverage is not reported because it was complete in all cases. In view of the randomized nature of our method, numerical results for each scenario are averaged over 10 simulation runs.

The first two groups of simulations are performed in the same garden-like environment, which is a square area with a side of 17 m, and refer to different initial deployments of the team. In the first, the robots are initially scattered in the environment (as if they had been parachuted). In the second, more realistic for environments with a single main entrance, the exploration is started with the robots grouped in a cluster.

Figure 1.10 shows the progress[2] of a typical SRG exploration with a team of 8 robots and a scattered initial deployment. Green areas represent the region so far explored. The robots are represented by red circles with the ID number. The view configurations are marked by black points. Yellow segments represent paths traveled by the robots during the exploration. Bridges are depicted in blue and may or not have been traversed by the robots. Exploration time and traveled distance for teams of different cardinality are shown in Fig. 1.11. Average results are shown for both the SRG (squares) and the SRT (crosses) method; in all simulations, variance for these data was less than 5%. As the number of robots in the team increases, the exploration time quickly decreases and tends asymptotically to zero (consider that an increment in the number of evenly deployed robots corresponds to a decrement of the individual areas they must cover, until no motion at all is necessary). A similar behavior is observed for the traveled distance. The performances of the two methods for a scattered start tend to become similar as the number of robots increases. This is due to the scattered initial deployment, which leads each robot to perform most of the exploration 'on its own'. As a result, the bridges present in SRG are rarely traversed and the performance of the SRG method does not improve significantly over the SRT method.

Figure 1.12 shows the progress of another SRG exploration in the garden-like environment, now with a team of 4 robots and a clustered initial deployment. Figure 1.13 summarizes the performance of the SRG and SRT methods for teams of different cardinality. In this case, the exploration time asymptotically tends to a nonzero value, which approximately represents the time required by a single robot to perform a roundtrip between the cluster center and the farthest point in the environment. Instead the average distance traveled by each robot still tends to zero. The results show that, on

[2] Video clips of all simulations and experiments are available at the web page `http://www.dis.uniroma1.it/~labrob/research/multiSRG.html`.

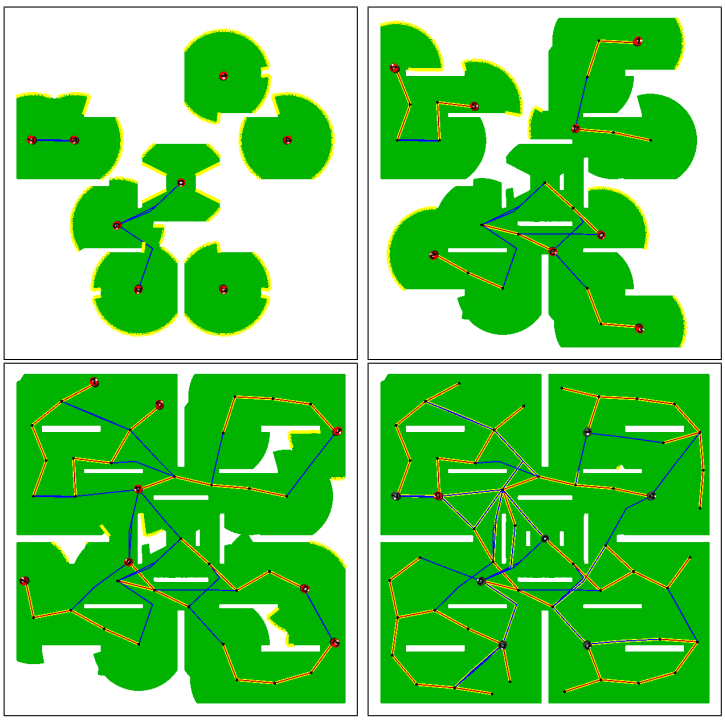

Figure 1.10: Simulation 1: SRG garden exploration with scattered start.

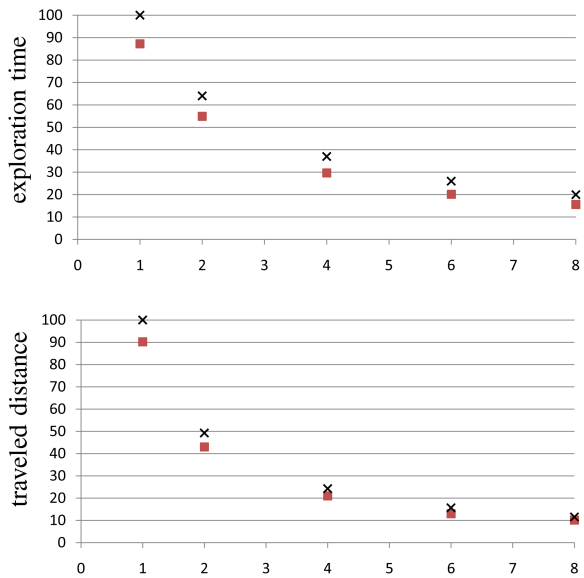

Figure 1.11: Garden exploration with scattered start: exploration time (above) and traveled distance (below) with teams of different cardinality. Results for SRG (squares) and SRT (crosses) explorations are shown.

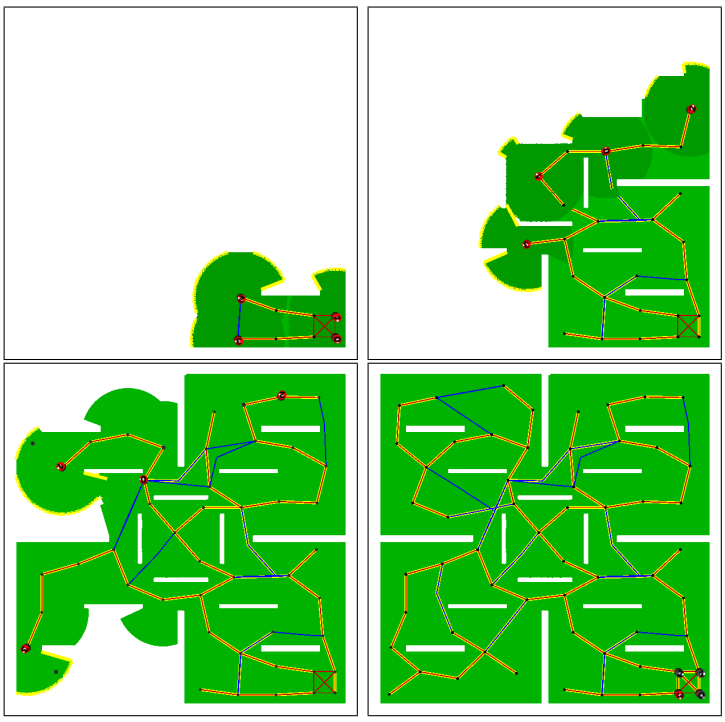

Figure 1.12: Simulation 2: SRG garden exploration with clustered start.

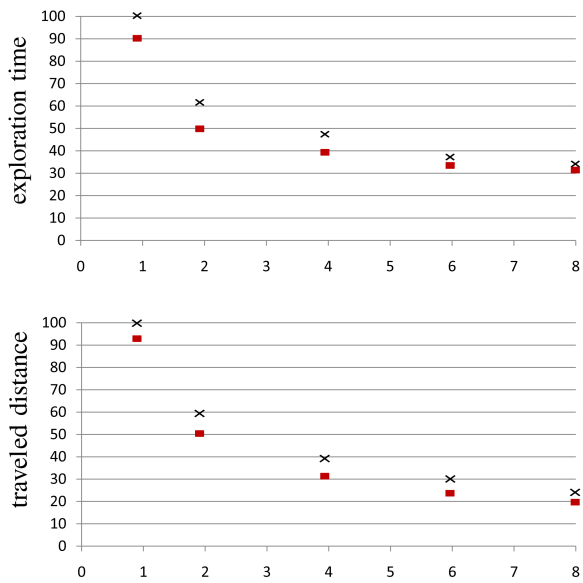

Figure 1.13: Garden exploration with clustered start: exploration time (above) and traveled distance (below) with teams of different cardinality. Results for SRG (squares) and SRT (crosses) explorations are shown.

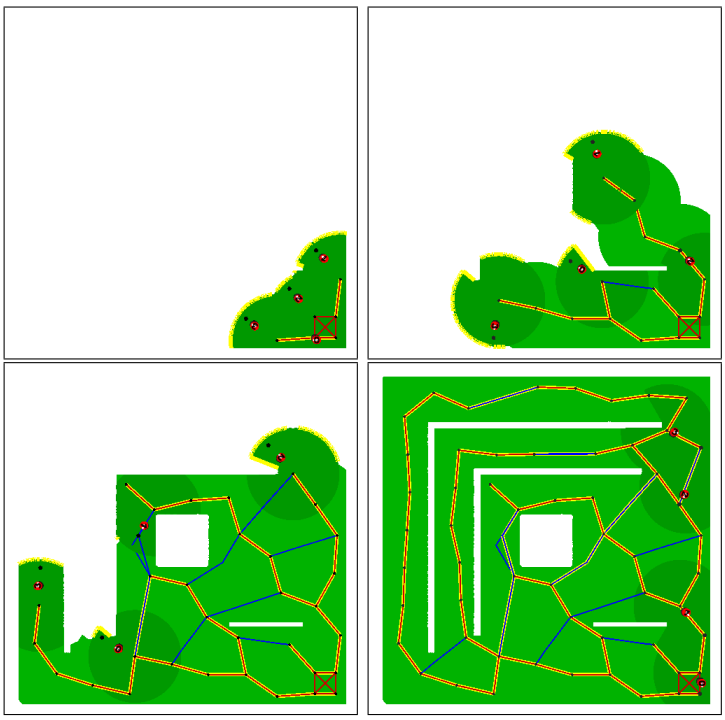

Figure 1.14: Simulation 3: SRG corridor exploration with clustered start.

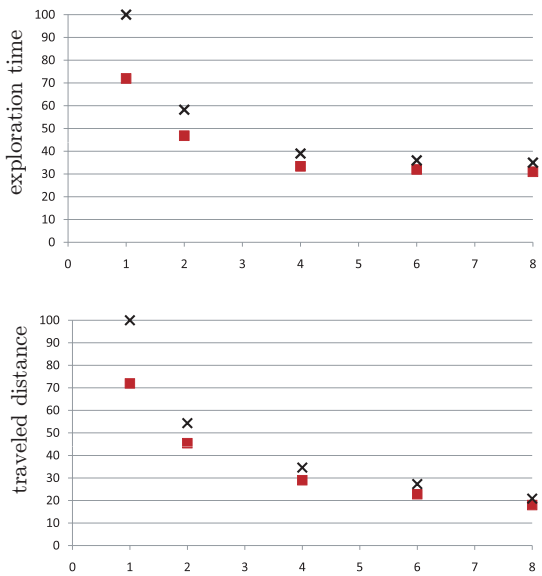

Figure 1.15: Corridor exploration with clustered start: exploration time (above) and traveled distance (below) with teams of different cardinality. Results for SRG (squares) and SRT (crosses) explorations are shown.

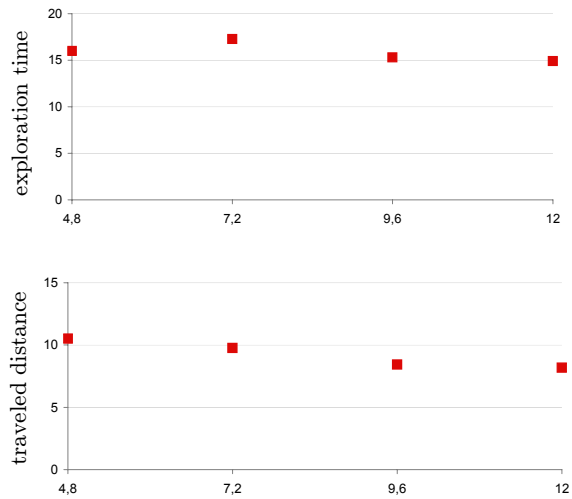

Figure 1.16: Garden exploration with a team of 8 robots and a clustered start: SRG exploration time (above) and traveled distance (below) for different values of the communication range R_c.

the average, the SRG method introduces a significant improvement in both the exploration time and the traveled distance.

This improvement becomes even more evident in the corridor environment (same size as the garden) used in the third group of simulations. Figure 1.14 shows the typical progress of an SRG exploration obtained with a team of 4 robots and a clustered initial deployment. From the numerical results in Fig. 1.15, obtained considering teams of different cardinality, it is clear that the marginal utility of an increase in the number of robots is higher for teams of small cardinality.

To investigate the influence of the communication range on the performance of the SRG method, we have repeated the first simulation with a team of 8 robots for increasing values of R_c (all satisfying the condition $R_c \geq 3R_p - \rho$). The results, shown in Fig. 1.16, indicate that a moderate improvement is obtained both in terms of exploration time and traveled distance. As R_c becomes comparable to the size of the square environment, however, an all-to-all communication condition is approached and the marginal utility of an increase in R_c tends to zero.

The cost associated to our coordination mechanism can be quantified by considering that in all our simulations the average cardinality of the GPA/GEA resulted to be lower than 2, and the percentage of exploration time spent by each robot in a waiting mode was around 15%.

1.10 Experiments

The SRG method has been experimentally validated using a team of Khepera III robots.

1.10.1 Description of the robots

Khepera III is the latest release of a family of two-wheel differentially driven mobile mini-robots developed by K-TEAM Corporation. The chassis of the robot is 7 cm high and 13 cm wide. It contains two motors, transmission elements, electronics, and a battery pack. A passive caster provides static stability to the vehicle. Each wheel is driven by a DC brushed servomotor coupled to the wheel via a 43.2:1 reduction. An embedded incremental encoder, placed on the motor axis, gives 16 pulses per revolution of the motor. Considering that the diameter of each wheel is equal to 4.1 cm, this results in a resolution of 691 pulses per revolution of the wheel, that correspond to 54 pulses per 0.1 cm of robot motion. The encoder resolution is by default set to the mode 4×, which corresponds to 2764 measures per wheel revolution.

In addition to the standard suite (infrared and ultrasonic sensors, serial, and USB communication), each robot has been equipped with a WiFi card for communication between robots of the team and/or with a remote computer, and a Hokuyo URG-04LX laser range finder, which is the sensor used for implementing the SRG method. Laser scans are acquired at a 10 Hz rate and are characterized by an angular resolution of 0.36°, radial resolution of 0.1 cm, maximum perception range R_p of 4 m. Since the scanning angle of the Hokuyo URG-04LX is 240°, when perceiving the robots perform a rotation on the spot to gather a 360° view.

1.10.2 Software and control architecture

Each Khepera III includes an Intel XSCALE PXA-255 400MHz processor, with embedded Linux operating system, a 64 MB RAM and a 32MB Flash memory, that allow to implement on-board the SRG method (according to the software architecture of Fig. 1.4). In the debugging phase, however, only the robot driver runs on the robot, while the explorer process runs on a remote computer. During the exploration, an additional process, called *visualizer*, is in charge of 'sniffing' and storing all the packets exchanged among the explorer processes in order to visualize and monitor the task progress.

A two-level architecture is adopted for controlling the motion of the robots. High-level velocity commands are issued at a 50 Hz rate by a trajectory tracking control scheme based on dynamic feedback linearization [28] and sent to the low-level PID controller of Khepera III, which is realized

with a PIC18F4431. The reference trajectories are the SRG arcs generated by the action planner. In particular, in the following experiments, each arc is mapped to a 2-phase maneuver, in which the robot first rotates so as to point to the next node, then travels to it along a line segment. A trapezoidal velocity profile is assigned over each phase.

Although a preliminary calibration of robot odometry and sensor parameters [29] was performed, dead reckoning proved to be inaccurate over relatively long paths, resulting in a degradation of the SRG method performance. Therefore, each robot has been provided with a basic self-localization module, in which incremental scan matching is used to correct odometric localization [30]. The information thus obtained is integrated in the high-level control law every 5 control cycles, due to the 10 Hz bound imposed by the scan acquisition frequency. In the presented experiments, the robots know their relative configurations at the start of exploration. Hence, mutual localization is maintained on the basis of this initial knowledge and self-localization data.

From a computational viewpoint, it is worth mentioning that the critical operations for the explorer process are graph managing and path search, while the most onerous operations for the robot driver are laser data acquisition and scan matching. Bandwidth is not an issue in our case, since the number of robots is limited. The used bandwidth is about 8 Kbyte/s times the number of robots in the same subnetwork.

1.10.3 Results

The exploration environments were built inside a rectangular arena measuring 1.90×3.70 (m). In view of the relatively small workspace, the perception range has been artificially limited to 1 m. The maximum cartesian velocity was set to 0.15 m/sec. The robot communication range was unlimited (this is not required by the SRG method). In all the experiments, the workspace was completely covered and the robots terminated the exploration task by completing the homing phase.

Figure 1.17 shows the environments used for the first two experiments, while Figs. 1.18 and 1.19 show the progress of the exploration and the SRG as reconstructed by the visualizer (a large red circle represents a robot, a small red point represents a node of the SRG) . The first environment is simply connected and its topology is correctly captured by the resulting SRG in the form of a tree (Fig. 1.18). Instead, the multiply connected environment of the second experiment results in an SRG that is a proper graph (Fig. 1.19). Note how the number of bridges (shown as dashed blue segments) in the second experiment is higher than in the first.

The overall performance of the SRG method in the two experiments is summarized in Table 1.20. The first three rows of data quantify the contribution of each robot of the team to the exploration task, in terms of

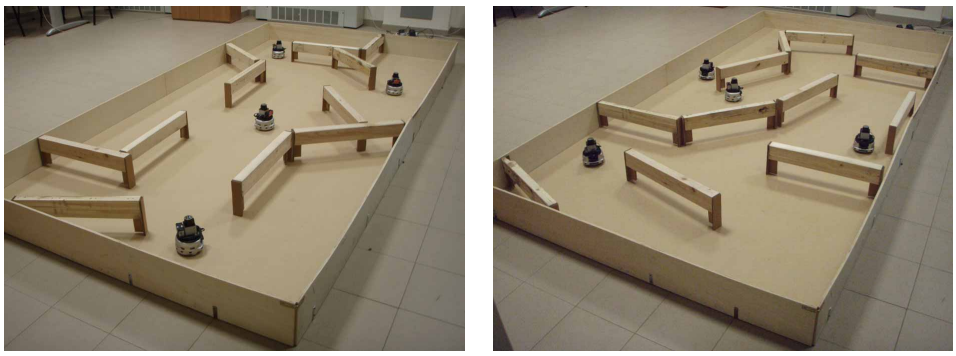

Figure 1.17: *Left*: Experiment 1. *Right*: Experiment 2.

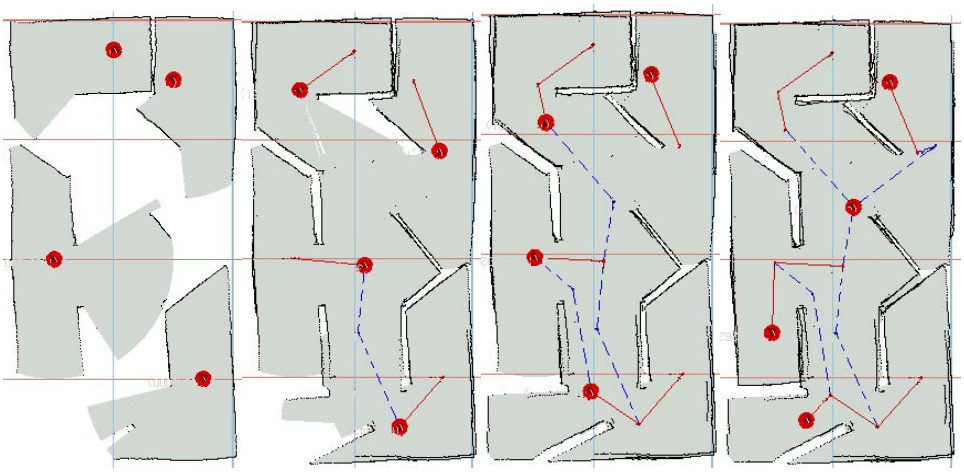

Figure 1.18: Experiment 1 as reconstructed by the visualizer.

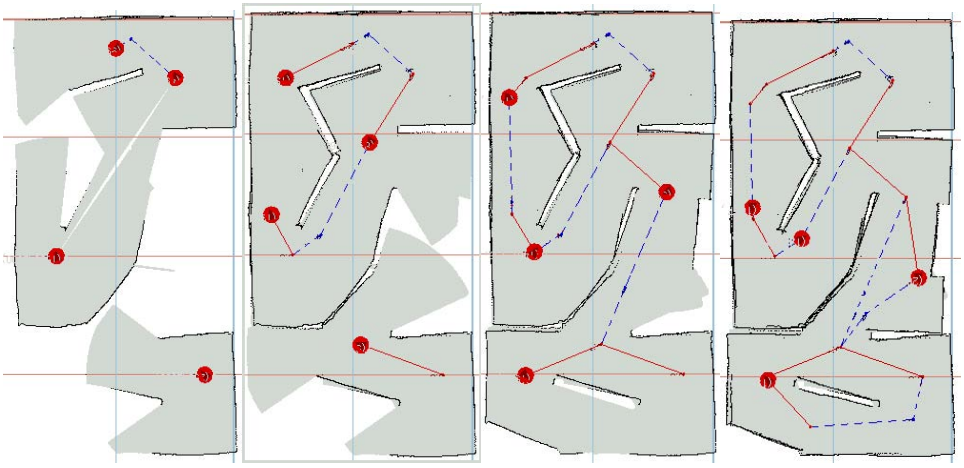

Figure 1.19: Experiment 2 as reconstructed by the visualizer.

	first experiment						
	robots				aggregate data		
	1	*2*	*3*	*4*	*mean*	*st dev*	*total*
# nodes	5	7	2	3	4	2	17
# total arcs	5	7	1	5	5	3	18
# bridge arcs	6	4	0	0	2	2	7
traveled distance (m)	3.34	2.65	1.28	2.35	2.41	0.86	9.62
exploration time (sec)	240	228	92	174	184	67	240
homing error (m)	0.053	0.055	0.036	0.022	0.042	0.016	0.166

	second experiment						
	robots				aggregate data		
	1	*2*	*3*	*4*	*mean*	*st dev*	*total*
# nodes	4	5	7	3	5	2	19
# total arcs	4	5	9	3	5	3	21
# bridge arcs	2	2	6	2	3	2	12
traveled distance (m)	3.42	4.05	3.98	1.27	3.18	1.30	12.72
exploration time (sec)	265	259	264	179	242	42	265
homing error (m)	0.020	0.074	0.018	0.156	0.067	0.065	0.268

Figure 1.20: Table of numerical results for the first and second experiment.

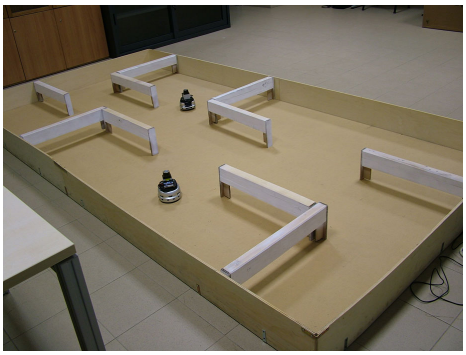

Figure 1.21: Experiment 3.

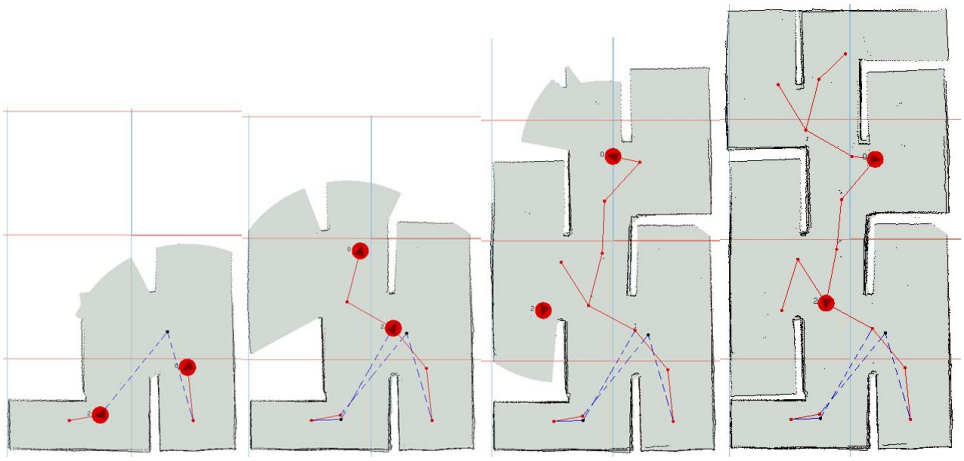

Figure 1.22: Experiment 3 as reconstructed by the visualizer.

number of nodes, arcs and bridges created by the robot. We also report the traveled distance, exploration time and homing error for each robot. These data collectively indicate that a good degree of collaboration has been achieved by the team, as robots that added less nodes to the SRG traveled a shorter distance and terminated the exploration in less time (this means that there was no time wasted in visiting already explored regions). Also, the aggregate data statistics show that the exploration task has been distributed on the individual robots with a satisfactory degree of uniformity. The homing error is reasonably small and (obviously) tends to increase with the traveled distance.

The third experiment was carried out in the exploration environment shown in Fig. 1.21, using a team of only two robots starting from a clustered formation. Again, the topology of the environment was correctly captured by the resulting SRG which has the structure of a tree (see Fig. 1.22). Note how, during the exploration, one of the two robots moves along the main axis of the rectangular area, pushing the frontier of the explored region towards the boundary of the arena; the second robot trails along and completes the exploration by moving Local Frontiers that were created but not approached by the first robot.

1.11 Open issues

In this chapter we have presented a decentralized strategy for cooperative robot exploration. A roadmap of the explored area, with the associated safe region, is built in the form of a compact data structure, called Sensor-based Random Graph (SRG). As it grows, the connectivity of the SRG is enhanced by adding bridges.

A simple and efficient decentralized cooperation mechanism is at the core of our method. This consists in an appropriate definition of the local frontier, by which each robot plans its motion towards areas that appear to be unexplored by the rest of the team on the basis of the available information. Local coordination guarantees that the collective motion of the team does not lead to collisions. Simulation and experimental results on a team of real robots have shown the satisfactory performance of the method both in ideal and practical conditions, even in the case of limited communication range.

We are currently working towards several objectives, among which we mention:

- To develop a version of the SRG method for the case of non-omnidirectional sensors along the lines of [31], by extending the configuration vector so as to include the orientation of the robot.

- To devise an SRG method for a team of heterogeneous robots, with different sensor capabilities and/or communication ranges.

- To perform a quantitative study of the robustness and scalability properties of the method.

1.11.1 Optimal bridge adding

An open issue is the optimal bridge adding, that arise when we have two graphs: a big graph and a smaller subgraph, and we want to expand the subgraph by mean taking edges and vertexes from the bigger one, in order to improve connectivity of the subgraph until a certain threshold, but maintaining as smaller as possible the size of subgraph.

Let $G = (V, E, W)$ be a graph with weighted edges. Let G' be a subgraph, $G' = (V', E', W') \subset G$, i.e. *i)* $V' \subset V$, *ii)* $E' \subset E$ and *iii)* the same edges have the same weights. Given two vertices $v_1, v_2 \in V'$, we denote with $d_{G'}(v_1, v_2)$ the distance between v_1 and v_2 on G' (intended as the weight sum of the minimal path). Moreover we denote the edges weights sum with $L(G') := \sum_{w \in W'} w$.

Consider a, not necessarily connected, undirected graph $G_t = (V_t, E_t, W_t)$, with weighted edges. Consider a subgraph $G_e = (V_e, E_e, W_e) \subset G_t$. Moreover, let be given a problem parameter $m \in \mathbb{R}^+$, called *desired distance*. Let be defined the following graph set, that we call $\mathcal{D}$:

$$\{G|\ G_e \subset G \subset G_t\}$$

$$\bigcap$$

$$\{G|\ \forall v_1, v_2 \in V_e,\ d_G(v_1, v_2) \leq \max(m, d_{G_t}(v_1, v_2))\} \tag{1.1}$$

Problem 1.1. *Find a graph $G^* \in \mathcal{D}$ such that:*

$$L(G^*) = \min_{G \in \mathcal{D}} L(G)$$

In other words we have to find an intermediate graph between G_e and G_t, say G^*, that *i)* satisfies the constraint that, if possible, the distance on G^* between two vertexes of G_e must be less than or equal to m, and *ii)* minimize L.

Some remarks:

1. We must add a technical requirement that prevents to have solutions with not useful isolated vertices:

$$|V^*| = \min_{\mathcal{D}} |V'|$$

2. Requirement 1.1 is equivalent to say that, $\forall v_1, v_2 \in V_e$, if $d_{G_t}(v_1, v_2) \geq m$ than it must be $d_{G^*}(v_1, v_2) = d_{G_t}(v_1, v_2)$, otherwise it must be $d_{G_t}(v_1, v_2) \leq d_{G^*}(v_1, v_2) \leq m$.

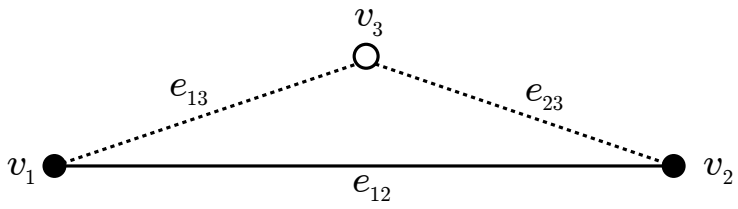

Figure 1.23: $G_t = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{13}, e_{23}\}\}$, the weights are euclidean distances between vertexes, and $G_e = \{\{v_1, v_2\}, \{e_{12}\}\}$.

3. If $m \to \infty$ than $G^* = G_e$. Instead it is not true that if $m = 0$ than $G^* = G_t$. Consider as counterexample the graph G_t in figure 1.23 where it is considered that weights are euclidean distances and that $V_e = \{v_1, v_2\}$, and $E_e = \{e_{12}\}$. In this case $G^* = G_e$ $\forall m$.

4. Connected components of G^*/G_e are called *bridges*.

Chapter 2

Optimal Patrolling with Communication Constraints

The problem of designing optimal trajectories to patrol an environment is the subject of this chapter. In both civil and military applications, it is of increasing importance to design strategies for a team of autonomous agents to detect the presence of intruders in a certain region of interest, or to monitor the topological changes of an environment. We start by presenting time optimization criteria, and by showing some fundamental properties of the patrolling problem. In the first part of the chapter, we present a distributed strategy the robots can implement to optimally patrol an environment described by a one dimensional graph. In the second part, we derive a heuristic for a more general situation, in which the environment is described by a general graph, and in which an optimal solution can not be computed in polynomial time. We conclude the chapter by showing some simulations on a realistic case.

2.1 Introduction

The recent development in the autonomy and the capabilities of mobile robots greatly increases the number of application suitable for a team of autonomous agents. Particular interest has been received by the tasks requiring continual execution, as the monitoring of oil spills [32], the detection of forest fires [33], the track of border changes [34], and the patrol or surveillance of an environment [35]. The surveillance of an area of interest requires the robots to continuously and repeatedly travel the environment, and the challenging problem consists of scheduling the robots trajectories. Indeed, depending on the environment and on the number of available robots, the design of the optimal trajectories may require the computation of the shortest tour visiting all the location of interest, in other words a TSP tour of the locations [36], or an optimal partitioning of the environment, a task which is also know as geometric covering [37]. Since both problems are well known to be NP-hard, also the design of a team trajectory falls into this class of

problems.

Several criteria have been proposed to evaluate the performance of a certain patrolling policy, and, often, the time gap between any two visits of a region, or the time needed to inform the whole team about a certain event is considered. Following [38], we will call refresh time and latency respectively the time between any two visits of the same location, and the time to inform the entire team of robots about an event. We will focus on the worst case analysis, even though the average refresh time and average latency cases remain of interest.

Because of the difficulty of finding optimal patrolling strategies, many existing solutions rely on heuristics, whose performance strongly depend upon the shape of the environment [39]. In [40] two type of strategies are presented, namely the cyclic and the partition-based strategy. In the cyclic strategy, the robots computes a closed route through the viewpoints, and travel repeatedly the route at maximum speed. Clearly, in the case of a single robot, the cyclic strategy has an optimal refresh time. In the partition-based strategy, the viewpoints are partitioned into M subsets, and each robot is responsible for a different partition. Once all the partitions have been assigned, each robot computes a closed tour visiting all the viewpoints it is responsible for, and repeatedly moves along that tour at maximum speed. As a result of [40], cyclic strategies are to be preferred whenever the patrolled graphs do not have long edges, while, otherwise, partition-based policies exhibit better performance.

Despite the general case, if the patrolling environment is one dimensional, optimal trajectories can be derived in polynomial time. In [35] it is shown that the optimal refresh time is obtained by equally dividing the region among the robots, and it is shown that an optimal latency is achieved by synchronizing the motion of the robots. In [38] a distributed algorithm, with optimal refresh time and latency, is proposed to spread and synchronize the robots. In particular, if L is the length of the perimeter to be patrolled, and M is the number of robots, the optimal refresh time is L/M while, assuming that two robots communicate only when they collide, the optimal latency is L.

The robots are usually considered to be identical and capable of sensing, communicating, and moving. We depart from the existing patrolling literature in the following ways. First, instead of considering that two robots can communicate only when they collide, as in [38], we assume that they are able to communicate whenever certain constraints are satisfied, e.g, when they see each other, or when they lie within a certain distance. It is worth noting that our communication assumption is general, since it takes advantage of a possible nonzero communication range, and it includes the previous case as a particular situation. Second, we represent the environment to be patrolled with an undirected graph G, and we assume that the robots are able to independently build such graph by exploring the environment. The

vertices of G correspond to a set of locations from which the sensors can entirely cover the environment, and each edge (i, j) reflects the possibility of both communicating and traveling between the locations i and j. Third and finally, we do not force the team of robots to patrol all the locations of the environment, and we leave the possibility to specify some particular points of interest.

The main contribution of this chapter are as follows. We characterize optimal refresh time and latency strategies for patrolling an environment described by a one dimensional graph. Our distributed algorithm allows to specify a discrete set of locations to be monitored, and it converges in finite time to an optimal solution also when the regions assigned to the robots have different length. For a perimeter of length L and a nonzero communication range, we show that, for a team of M robots moving at unitary speed, the refresh time of our policy is in general less that L/M, and the latency is less than L. As a preliminary step to characterize the optimal refresh time of a team trajectory, we present a distributed polynomial time algorithm to compute a clustering of a set of adjacent points, such that the maximum diameter of the clusters is minimized. Finally, we construct and characterize the performance of a heuristic to be applied when the graph describing the environment is not one dimensional.

The rest of the chapter is organized as follows. In Section 2.2 we define the notation and the problem under consideration. Sections 2.3 and 2.4 contain our main results. They present an optimal team trajectory for an environment described by a one dimensional graph, with respect to the refresh time and the latency criterion. In Section 2.5 a distributed version of the proposed procedure is implemented, and in Section 2.6 a heuristic to patrol an environment represented by a general graph is described. Finally, Sections 2.7, **??**, and 2.8 contains respectively some simulation results, some experiments, and our conclusions.

2.2 Problem formulation

We will be using the standard motion planning notation, and we refer the reader to [41] for a comprehensive treatment of the subject. We are given a team of $M > 2$ identical robots, capable of sensing, communicating, and moving in a connected environment.

Regarding sensing, we assume that the environment can be completely covered by simultaneously placing a robot at a set of $N > 3$ *viewpoints* in the configuration space. In other words, if $M = N$ robots were available and placed at the N viewpoints, then the union of the sensor footprint of each robot would provide complete sensor coverage in the environment. However, we assume $N > M$ so that at least one robot needs to visit more viewpoints for the entire environment to be monitored over time.

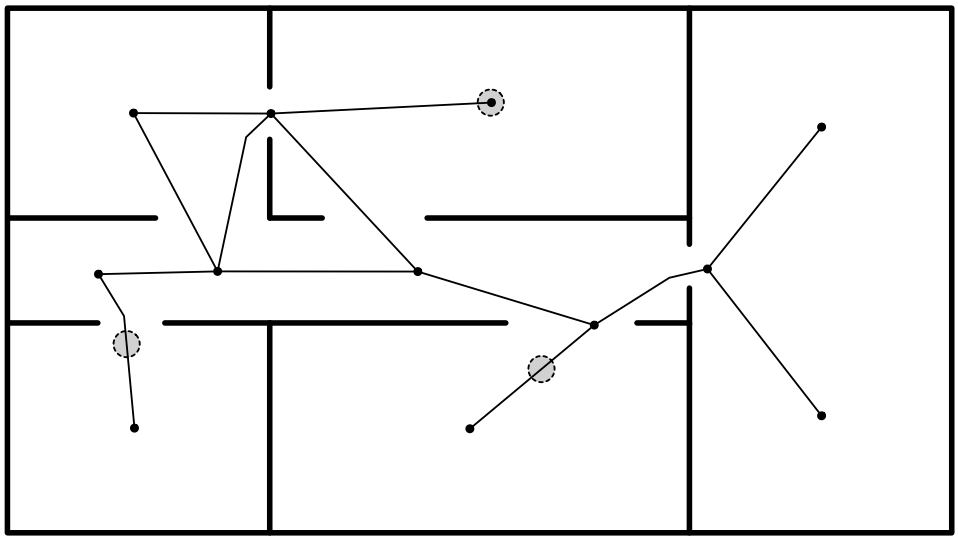

Figure 2.1: A polygonal environment and a graph with $N = 12$ viewpoints as nodes and shortest paths as edges. The $M = 3$ robots are holonomic and disk-shaped (the dotted circles grey-filled), and their centers must lie on the graph. The robot sensors are omnidirectional cameras, and the communication is specified by an r-limited visibility graph with respect to the center of the robots. The weight of the edge (i, j) coincides with the shortest path between i and j. Since the robots are not points, some paths are not straight lines.

.

Regarding communication, we associate with the environment an undirected graph G, whose vertices are the given N viewpoints, and in which there is an edge between two vertices if two robots placed at those viewpoints are able to communicate. We assume that the graph G is connected.

Regarding motion, we assume that the robots are holonomic, i.e., first order integrators, and move at speed upper bounded by 1. Additionally, we turn the graph G into a robotic roadmap [41] as follows: to each pair of viewpoints that are neighbors in G, we associate a unique shortest path connecting them. We assume that each robot remains always in the roadmap. (Notice that each edge of G corresponds to both a communication edge as well as a motion path.)

Remark (The graph G may arise from a distributed exploration algorithm) The set of viewpoints may be the result of an exploration algorithm on a generic environment, as described in [42] and [1]. As an example, Fig. 2.1 shows the graph G for a scenario in which the robots are disk-shaped, the configuration space coincides with $\mathbb{R}^2$, the environment is polygonal, and the sensors are omnidirectional cameras. The communication graph is the

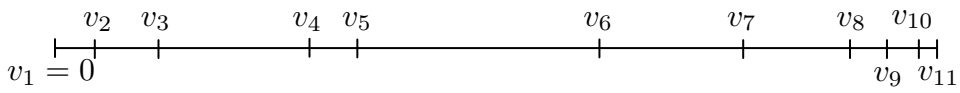

Figure 2.2: A set of $N = 11$ viewpoints $V = \{v_1, \ldots, v_{11}\}$

r-limited visibility graph with respect to the center of the robots [43], and the weights of the edges equal the shortest paths between the viewpoints. Note also that, in order to avoid collisions, the admissible paths joining two viewpoints are not always straight lines.

The problem of scheduling the trajectories of a team of autonomous robots to patrol an arbitrary environment is generically NP-hard. To see this, note that the patrolling problem requires sometimes to find an optimal clustering of a set of viewpoints disposed in the plane. Since the geometric clustering is NP-hard [37], also the problem of determining a set of optimal trajectories to navigate and patrol an environment is not solvable in polynomial time. In some special cases however, for instance when the graph G is a chain, the patrolling problem can be solved efficiently. We focus first on this simpler case, and we explain in Section 2.6 how to deal with more complex situations.

Assumption 2.1. *The undirected graph G is a chain.*

Under the Assumption 2.1, the patrolling problem is intrinsically one dimensional and can be easily mapped on the real line. Let $v_i \in \mathbb{R}_{\geq 0}$ denote the distance in the roadmap from the viewpoint of the first vertex to the viewpoint of the i-th vertex. We denote with $V = \{v_1, \ldots, v_N\}$ the set of those positive real numbers, and, for simplicity, we call them *viewpoints* (see Fig. 2.2). Similarly, the position of the robot i at time t, denoted with $x_i(t) \in \mathbb{R}_{\geq 0}$, is the distance in the roadmap from the viewpoint of the first vertex to the current robot configuration, that is, the length of the union of the paths in the roadmap joining the position of the robot and the first viewpoint in the environment. Because of the design of the roadmap, note that 1) the i-th and the j-th robot are allowed to communicate when $x_i(t) = v_k$ and $x_j(t) = v_{k+1}$ for some $k \in \{1, \ldots, N-1\}$, and 2) the distance $x_i(t)$ does not generally coincide neither with the Euclidean distance, nor with the length of the shortest path between the robot i and the first viewpoint.

A *team trajectory* x is a collection of M continuous and piecewise-differentiable functions $x_i : \mathbb{R} \to [0, v_N]$, for $i \in \{1, \ldots, M\}$, satisfying the constraint $-1 \leq \dot{x}_i \leq 1$ at almost all times. Denote with $X(t)$ the image of a team trajectory at time t, i.e., the union of the M positions of the team $\cup_{i=1}^{M}\{x_i(t)\}$. The *refresh time* of a team trajectory x, with respect to a set of viewpoints V, is the largest interval $I \subset \mathbb{R}_{\geq 0}$ such that at least

one viewpoint does not belong to $X(t)$, for every $t \in I$. Clearly, two team trajectories with the same image $X(t)$ have an equal refresh time.

Problem 2.1 (Minimize refresh time)**.** *Given a set of viewpoints V, find a team trajectory that minimizes the refresh time with respect to V.*

For a team trajectory, the *latency* is defined as the minimum time interval L necessary for a message generated at any time by any robot to reach all the other robots.

Problem 2.2 (Minimize latency with an optimal refresh time)**.** *Given a set of viewpoints V and a family of team trajectories that minimize the refresh time, find, within such family, a team trajectory that minimizes the latency.*

Note that, without any constraint on the refresh time, the problem of minimizing the latency is trivially solved by a team trajectory in which $x_i(t) = x_j(t)$ for all $i, j \in \{1, \ldots, M\}$, because, since the robots communicate at any time, the latency is zero. Notice also that the setup in [38] is obtained as a special case of our setup, by setting the communication radius to zero, and by assuming that the set of viewpoints is dense.

2.3 Optimal refresh time

We characterize in this Section a solution to Problem 2.1. A team trajectory x is *ordering invariant* if the order of the robots positions remains constant with respect to time, i.e., if there exists an indexing of the robots such that $x_i(t) \leq x_{i+1}(t)$, for each $i \in \{1, \ldots, M-1\}$, and for every instant t, as shown in Fig. 2.3. The following proposition restricts the search space for a solution to Problem 2.1 to the set of ordering-invariant team trajectories.

Proposition 2.1 (Ordering invariance)**.** *For every team trajectory, there exists an ordering-invariant team trajectory with the same refresh time.*

Proof. Let x be a team trajectory, and consider the permutation matrix function $P_x(t)$, that keeps track of the ordering of x at time t, i.e., such that the (i, j)-th entry of $P_x(t)$ is 1 if, at time t, the i-th robot occupies the j-th position in the chain of robots, and it is 0 otherwise. Since x is continuous, anytime the function $P_x(t)$ is discontinuous, the positions of the robots directly involved in the permutation overlap. Therefore, the ordering invariant team trajectory $P_x^{-1}(t)x(t)$ is feasible, and it has the same refresh rate as x, since the two team trajectories have the same image. An example is in Fig. 2.3. □

Given a team trajectory and an index $i \in \{1, \ldots, M\}$, the i-th *cluster* $C_i \subset V$ is the set of viewpoints covered at least once by the trajectory of the i-th robot. The diameter d_i of C_i is the maximum distance among the

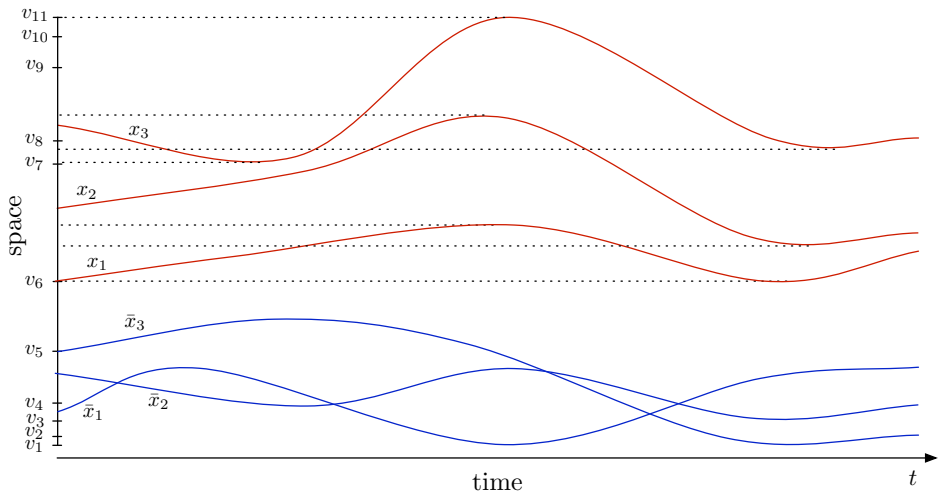

Figure 2.3: The team trajectory $x = \{x_1, x_2, x_3\}$ (red on top) has the clusters $C_1 = \{v_6\}$, $C_2 = \{v_7, v_8\}$, and $C_3 = \{v_7, v_8, v_9, v_{10}, v_{11}\}$. Their diameters are respectively $d_1 = 0$, $d_2 = v_8 - v_7$ and $d_3 = v_{11} - v_7$. It is ordering invariant but their clusters are not disjoint. Furthermore, since $C_1 \cup C_2 \cup C_3 \neq V$, the team trajectory x does not guarantee a finite refresh time. On the contrary, the team trajectory $\bar{x} = \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$ (blue at the bottom) is not ordering invariant.

viewpoints of C_i (see Fig. 2.3). Notice that, if a team trajectory has a finite refresh time, then the clusters cover the whole set V, i.e, $V = \cup_{i=1}^{M} C_i$. In the following Proposition, we show that an ordering invariant team trajectory, whose clusters form a partition of the viewpoints V, can be a solution to Problem 2.1.

Proposition 2.2 (Partitions)**.** *For every ordering invariant team trajectory x with bounded refresh time, there exists an ordering invariant team trajectory $\bar{x}$, which has a not greater refresh time, and in which the clusters are a partition of the viewpoints V.*

Proof. Let $C_1, \ldots, C_M$ be the clusters associated to the team trajectory x. Since x has a bounded refresh time then $V = \cup_{i=1}^{M} C_i$. Consider the partition of V defined as

$$\begin{aligned} \bar{C}_1 &= C_1 \\ \bar{C}_i &= C_i \setminus \cup_{i=1}^{i-1} C_i \quad i \in \{2, \ldots, M\}, \end{aligned}$$

and let $\bar{l}_i = \bar{b}_i - \bar{a}_i$, where $\bar{a}_i = \min \bar{C}_i$ and $\bar{b}_i = \max \bar{C}_i$, with $i \in \{2, \ldots, M\}$. Consider the viewpoint $\bar{a}_i$, and note that, by construction, it is visited by the robot i and possibly by the robots j, $j > i$. Because the robots trajectories are ordering invariant, then $x_i(t) \leq x_j(t)$, and hence the refresh time of

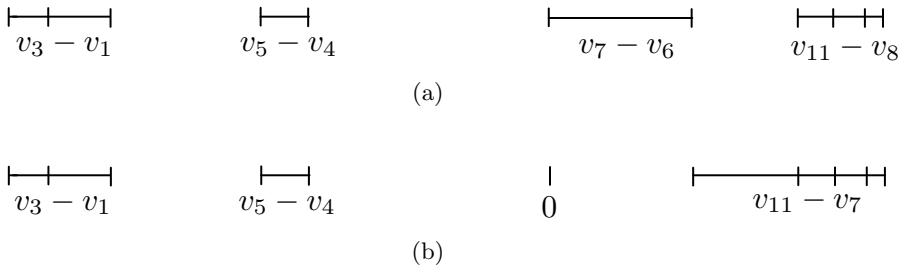

Figure 2.4: Two 4-partitions of the set of viewpoints V of Fig. 2.2. In 2.4(a) an optimal 4-partition with dimension $v_7 - v_6$. Sweeping trajectories with this partition as clusters have a minimum refresh time. In 2.4(b) a minimum average diameter 4-partition, with dimension $v_{11} - v_7 > v_7 - v_6$. A minimum average diameter 4-partition is obtained by removing the 3 among the longest edges.

the viewpoint a_i is no less than $2\bar{l}_i$, i.e., the time needed by the robot i to traverse its entire cluster C_i at maximum speed. It follows that the refresh time of the team trajectory x is lower bounded by $2\max_i \bar{l}_i$. To conclude the proof, note that a refresh time of $2\max_i \bar{l}_i$ is obtained by the team trajectory $\bar{x}$, in which each the i-th robot periodically sweeps the set $\bar{C}_i$ with a period non greater than $2\max_i \bar{l}_i$. □

Given a set of viewpoints V, an *M-partition* is a partition of V formed by M clusters. The *dimension* of an M-partition is largest diameter among its M clusters. We call an M-partition *optimal* if its dimension is the smallest among all the M-partitions of V, and we denote with $d_{V,M}$ the optimal dimension. For sake of simplicity, we will write d_M whenever the set V is clear from the context. As an example, for the viewpoints of Fig. 2.2, an optimal 4-partition is in Fig. 2.4(a).

Theorem 2.1 (Minimum refresh time)**.** *Given a set of viewpoints V, the minimum refresh time achievable with a team of M robots is $2d_M$, where d_M is the dimension of an optimal M-partition of V.*

Proof. The Theorem is a direct consequence of Propositions 2.1 and 2.2. □

To conclude this Section, a minimum refresh time team trajectory to patrol a set of viewpoints V is obtained by 1) computing an optimal M-partition of V, and 2) letting each robot sweep a cluster of the optimal M-partition with a period no less than $2d_M$. In the remaining part of this Section we show an algorithm to efficiently find an optimal partition.

Remark An M-partition that minimizes the average length of the diameters of the clusters, or equivalently the sum of the diameters of the clusters,

is obtained by removing the $M-1$ longest edges. Note that, in general, such partition does not minimize the dimension of the M-partition, and hence is not optimal in our sense. An example is in Fig. 2.4(b).

2.3.1 Optimal M-partition

Because the viewpoints lie on a line, the optimal M-partition is computed in polynomial time. We focus now on a centralized version of the clustering algorithm, and we present in Section 2.5 a distributed version of the same procedure.

Given a set of viewpoints V, we call *left-induced partition of length l* the partition defined as

$$C_i^l = \{v \in V \ : \ a_i \leq v \leq a_i + l\}, \quad i \in \{1, \ldots, M_l\}, \tag{2.1}$$

where

$$\begin{aligned} a_1 &= v_1 \\ a_i &= \min\{v \in V \ : \ v > a_{i-1} + l\}, \quad i \geq 2, \end{aligned}$$

and M_l is the smallest number such that the set $\{v \in V \ : \ v > a_{M_l} + l\}$ is empty, i.e., the cardinality of the left-induced partition of length l. Notice that, since the set V is clear from the context, we omit the dependence of M_l from V. As an example, for the set of viewpoints of Fig. 2.2, two left induced partitions are in Fig. 2.5(a). The cardinality of the left-induced partition M_l is monotonically non-increasing with l, and it is assumed to be right continuous, as it is shown in Fig. 2.5(b). Clearly $1 \leq M_l \leq N$, and, in particular, if $l > v_N$, then $M_l = 1$, and, if $l < \min_{i=2,\ldots,N} v_i - v_{i-1}$, then $M_l = N$. Also, if $l = v_N/M$, then $M_l \leq M$. Let $\{l_1, \ldots, l_{N-1}\}$ be the discontinuity points of the function M_l, we have

$$\begin{aligned} M_l &\leq k, \ \text{ if } \ l \geq l_k, \\ M_l &> k, \ \text{ if } \ l < l_k, \end{aligned} \tag{2.2}$$

where $k \in \{1, \ldots, N-1\}$. Note that two or more discontinuity points of M_l may coincide, so that the function M_l may not assume all the values of the set $\{1, \ldots, N\}$, as in Fig. 2.5(b) for $M_l = 9$.

Theorem 2.2 (Optimal partition)**.** *Given a set of viewpoints V, let d_M be the dimension of an optimal M-partition. Then d_M coincide with the smallest dimension of the left-induced partition of cardinality M, i.e., $l_M = d_M$.*

Proof. We want to show that d_M is one of the discontinuity points of the function M_l, i.e., that d_M verifies the conditions (2.2). By contradiction, if $M_l = M$ and $l < d_M$, then the left-induced partition of length l would be

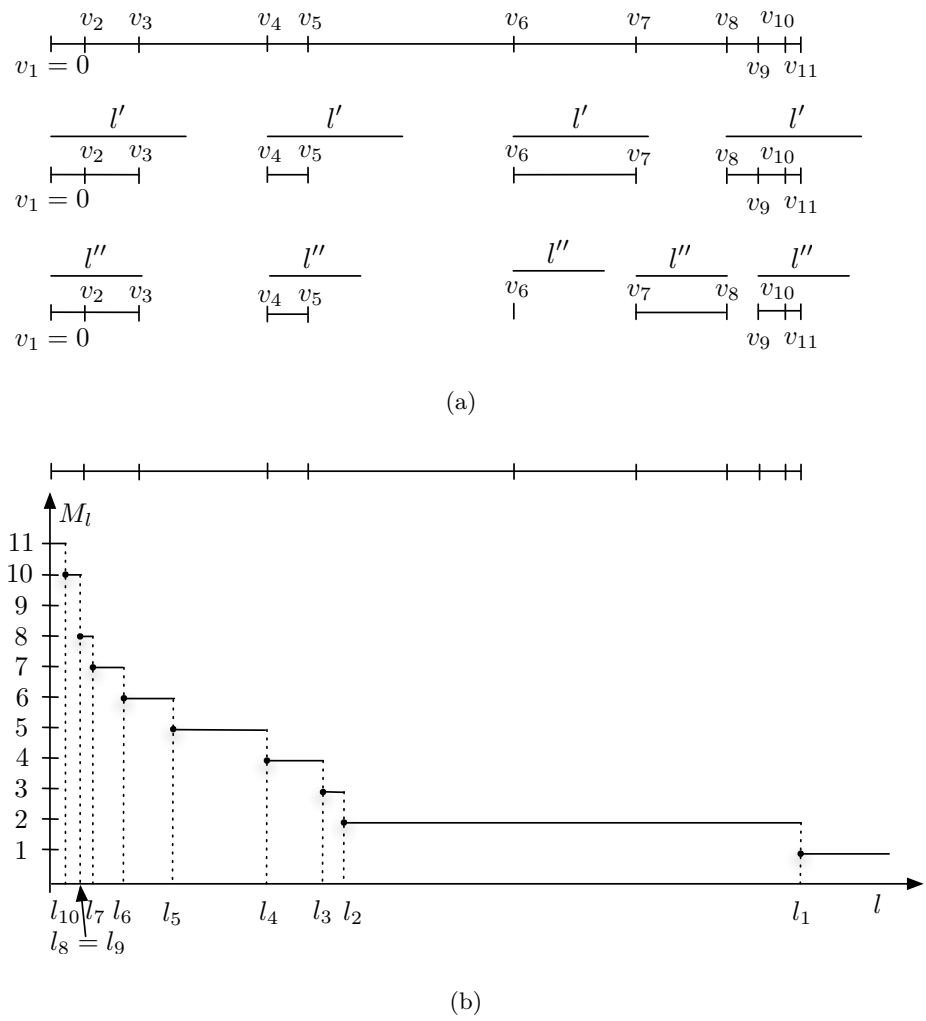

Figure 2.5: In 2.5(a) two left induced partition of the set of viewpoints of Fig. 2.2, corresponding to two different length l' and l'', with $l'' < l'$. The cardinalities are respectively $M_{l'}$=4, and $M_{l''} = 5$. In 2.5(b) the cardinality M_l of the left-induced partition of the set of viewpoints in Fig. 2.2, plotted as a function of the length l.

a partition of smaller dimension. If $M_l < M$ and $l < d_M$, then M clusters can be obtained from the left-induced partition of length l, and the resulting M-partition would still have shorter diameter than d_M. Therefore, if $l < d_M$, then $M_l > M$. Suppose now that $l \geq d_M$, and let $C = \{C_1, \ldots, C_{M_l}\}$ and $\tilde{C} = \{\tilde{C}_1, \ldots, \tilde{C}_M\}$ be respectively the left-induced partition of length l and the optimal partition. Note that $|C_1| \geq |\tilde{C}_1|$, since the cluster C_1 contains all the viewpoints within distance l from v_1, and hence also within distance d_M.

Algorithm 2: Optimal left-induced M-partition

input : $V \in \mathbb{R}^N_{\geq 0}$, $M \in \{1, \ldots, N-1\}$, $\varepsilon > 0$;

1 *Set* $a = 0$, $b = \frac{v_N}{M}$, $l = \frac{(a+b)}{2}$, *and let* C^{opt} *be the left-induced partition of length* b;
2 **while** $(b - a) > 2\varepsilon$ **do**
3 **if** $M_l > M$ **then**
4 $a = l$, $l = \frac{a+b}{2}$;
5 **else**
6 $C^{\text{opt}} = C$, $M^{\text{opt}} = M_l$, $b = l$, $l = \frac{a+b}{2}$;
7 **if** $M^{opt} < M$ **then**
8 Convert C^{opt} into an M-partition;

It follows $\max C_1 \geq \max \tilde{C}_1$, and also that $\min C_2 \geq \min \tilde{C}_2$. By repeating the same reasoning to the remaining clusters, we obtain that $\max C_M \geq \max \tilde{C}_M$, so that, if $|\tilde{C}| = M$ and $l \geq d_M$, then $|C| = M_l \leq M$. □

As a direct consequence of Theorem 2.2, an optimal M-partition of the set V, which coincides with the optimal left-induced M-partition, can be found by means of Algorithm 2.

Lemma 2.1 (Convergence of Algorithm 2)**.** *For a set of viewpoints* $V = \{1, \ldots, v_N\}$ *and a number* $1 \leq M \leq N-1$*, Algorithm 2 returns the optimal left-induced* M*-partition with precision* ε*.*

Proof. Algorithm 2 looks for the minimum length l^* that generates a left-induced partition of cardinality M. The length l^* coincides with one of the discontinuity points of M_l, and it holds $l^* \in (0, v_n/M)$. Indeed, l^* has to be positive because the desired cardinality M is smaller than the number of viewpoints. Also, $l^* < v_N/M$, because $(v_N/M)M = v_N$, i.e., M clusters of length v_N/M cover, on the real line, the whole distance expressed by the viewpoints V. Since the function M_l is monotone, and l^* is such that $M_l > M$ for every $l < l^*$, the interval $[a, b]$, as updated in Algorithm 2, contains the value l^* at every iteration. Finally, the length of the interval $[a, b]$ is divided by 2 at each iteration, so that, after $\log_2((v_N/M)/\varepsilon)$, the value l^* is computed with precision ε. □

Remark (Approximation factor ε) Given the set of viewpoints V, the value of ε that yields the exact length l_M of the optimal left-induced M-partition can be computed, but such computation requires a factorial number of operations.

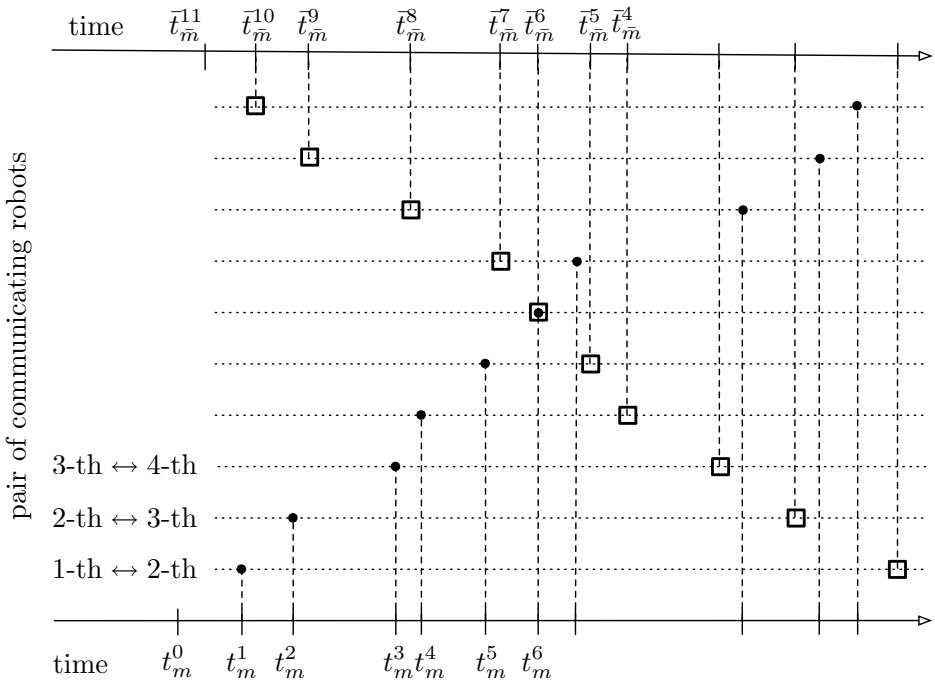

Figure 2.6: Communication instants for the routing of the m-th message from the first to the last robot (black points) and the $\bar{m}$-th message from the last to first robot (squares). The number of robots M is 10, the instant t_m^6 is in common.

2.4 Optimal latency

We describe a team trajectory that minimizes the latency of the communication among the robots, while maintaining an optimal refresh time, i.e., we propose a solution to Problem 2.2. Given an optimal M-partition of the viewpoints V, as described in Section 2.3, let $d_1, \ldots, d_M$ be the diameters of the clusters, and recall that the minimum refresh time is $2d$, where $d := \max_{i \in \{1,\ldots,M\}} d_i$. Each robot is confined within the two extremal viewpoints of its assigned set of the M-partition, and the $M-1$ edges dividing the assigned sets are not traveled by the robots.

The latency of a team trajectory is defined as the minimum time interval necessary for a message generated at any time by any robot to reach all the other robots, or, equivalently, because the communication graph among the robots has a chain structure, as the minimum real number L such that, at any time t, a message generated at one extreme of the chain is delivered to the opposite side within time $t + L$. Clearly, a message generated by the first robot needs to go through all the robots in the chain before reaching the M-th robot. Let $m \in \mathbb{N}$, we denote the routing instants of the m-th

message sent by the first robot as

$$t_m^1 < \ldots < t_m^{M-1}, , \tag{2.3}$$

where t_m^i is the instant in which the i-th robot passes the m-th message to the $(i+1)$-th robot. In a similar way, let $m \in \mathbb{N}$, and denote with

$$\bar{t}_m^{M-1} < \ldots < \bar{t}_m^1, \tag{2.4}$$

the routing instants of the m-th message generated by the M-th robot, where $\bar{t}_m^i$ is the instant in which the $(i+1)$-th robot passes the m-th message to the i-th robot, as in Fig. 2.6. Suppose that the first robot wants to send a message to robot M at time $t \in (t_{m-1}^1, t_m^1]$. The minimum time that guarantees the delivery of such message is

$$L_u(m) := t_m^{M-1} - t_{m-1}^1 = (t_m^1 - t_{m-1}^1) + (t_m^{M-1} - t_m^1). \tag{2.5}$$

Similarly, for a message generated by the M-th robot at time $t \in (\bar{t}_{m-1}^{M-1}, \bar{t}_m^{M-1}]$, the delivery time is at most

$$L_d(m) := \bar{t}_m^1 - \bar{t}_{m-1}^{M-1} = (\bar{t}_m^{M-1} - \bar{t}_{m-1}^{M-1}) + (\bar{t}_m^1 - \bar{t}_m^{M-1}). \tag{2.6}$$

In terms of the above expressions, the latency becomes

$$\max_{m, m \in \mathbb{N}} \left(\max \left(L_u(m), L_d(m) \right) \right). \tag{2.7}$$

It is worth noting that the expressions (2.5) and (2.6) underlines two different components that form the latency. In particular, the terms $(t_m^1 - t_{m-1}^1)$ and $(\bar{t}_m^{M-1} - \bar{t}_{m-1}^{M-1})$ correspond to the time between the generation, and the first passage of the message, i.e., they reflect the frequency of delivery of the messages. On the other hand, the terms $(t_m^M - t_m^1)$ and $(\bar{t}_m^1 - \bar{t}_m^{M-1})$ characterize the transport time along the chain of robots.

Let the diameter of the j-th set of the M-partition be maximum, i.e., $d_j = d$. Note that the exchange of messages between the pair of robots $(j-1, j)$ and $(j, j+1)$ has a frequency no greater than $1/(2d)$, because the speed of the robots is at most unitary, and the refresh time of the team trajectory needs to be $2d$. Hence, the j-th robot (as well as every robot assigned to a set of maximum diameter) represents a bottleneck for the frequency of delivery of the messages. It follows that a solution to Problem 2.2 can be searched among the team trajectories that are $2d$-periodic, since the frequency of delivery remains the maximum achievable, since the transport time is independent of the frequency of delivery. Finally, the minimization of the latency consists of finding the trajectories that optimize the maximum transport time for the two ways of the chain of robots.

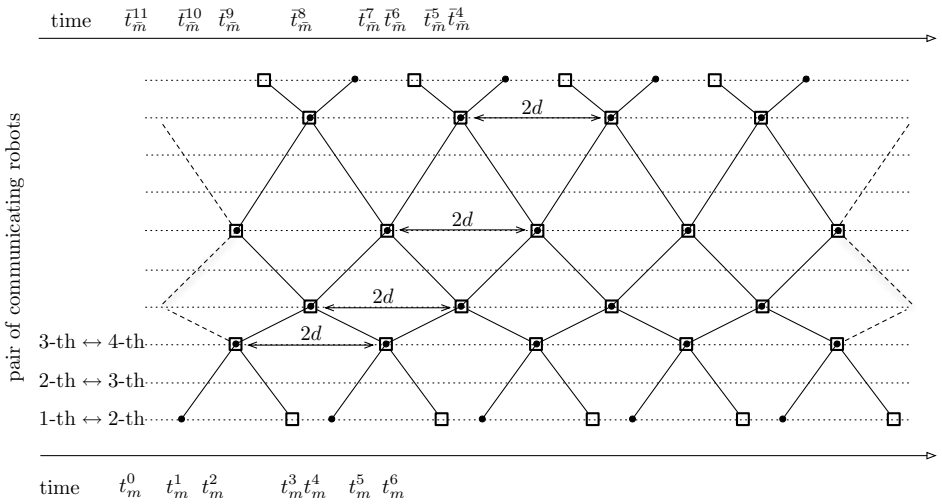

Figure 2.7: A $2d$-periodic framework of common instants. Since the 4-th robot has to patrol the cluster of maximum length, it represents a bottleneck.

Theorem 2.3 (Optimal latency)**.** *Let $d_1, \dots, d_M$ be the diameters of the partitions obtained with Algorithm 2 on the viewpoints $V = \{v_1, \dots, v_N\}$, and let $d = \max_{i \in \{1,\dots,M\}} d_i$. Consider the set S of the partial sums*[1] *of $\{d_i\}_{i=1}^M$, and the set $F^* = \{f_0^*, \dots, f_{C^*+1}^*\}$ defined as*

$$\begin{aligned} f_0^* &= 0 \\ f_i^* &= \arg \min_{f \in S, f - f_{i-1} \leq d} (d - (f - f_{i-1})), \quad i \in 1, \dots, C^* + 1, \end{aligned} \tag{2.8}$$

where $f_{C^*+1} = \sum_{i=1}^M d_i$. The latency of a $2d$-periodic team trajectory is lower bounded by

$$(C^* + 1)d + (f_1 - d_1) + (f_{C^*+1} - f_{C^*} - d_M) \tag{2.9}$$

Proof. If the trajectories are periodic, then any pair of routing sequences can be obtained by shifting the sequences (2.3) and (2.4) of a multiple of $2d$. In particular, $t_{m+1}^i = t_m^i + 2d$ and $\bar{t}_{m+1}^i = \bar{t}_m^i + 2d \; \forall m \in \mathbb{N}$, and $\forall i \in \{1, M-1\}$. Also, because the communication instants among the robots $\{2, \dots, M-1\}$ do not affect the latency, as underlined in the expressions (2.5) and (2.6), we consider team trajectories where the robots behave symmetrically while sweeping their cluster, so that the time needed to transfer a message from the robot $i-1$ to the robot $i+1$ equals the time needed to transfer a message from the robot $i+1$ to the robot $i-1$, for all $i \in \{2, \dots, M-1\}$. The symmetric framework describing the routing sequences has the form of Fig. 2.7, where the intersections between the up-going and down-going routing

[1] Sums of the form $\sum_{i=1}^m d_i$, with $m \leq M$.

sequences are marked. Note that the maximum time to transfer a message generated at any time t from the robot 1 to the robot M equals the time needed to go from the robot M to the robot 1. Since the team trajectory is $2d$-periodic and symmetric, the number of intersection instants C and the following time intervals are constant: 1) the time interval τ_1 between t_m^1 and the first intersection instants, 2) the time intervals between the intersection instants, that are all equal to d, and 3) the time interval τ_M between the last intersection point and t_m^{M-1}. Therefore the latency is at least

$$2d + (C-1)d + \tau_1 + \tau_M. \tag{2.10}$$

Indeed, the term $2d$ is the maximum time necessary for the robot 1 to pass the message to robot 2, and $(C-1)d + \tau_1 + \tau_M$ is the minimum time necessary for the robot 2 to communicate with the robot M. The minimum value for the expression (2.10) is achieved by a team trajectory that minimizes the number C of intersection points between the up-going and down-going routing sequences, i.e., by the expression (2.9). Indeed for each added intersection point the term $(C-1)d$ of (2.10) increases by d, and the term $\tau_1 + \tau_M$ can at most decrease by d.

□

Remark (Equally spaced viewpoints) The lower bound (2.9) implies that the latency is lower bounded also by

$$2d + \sum_{i=2}^{M-1} d_i. \tag{2.11}$$

The lower bounds (2.9) and (2.11) coincides if and only if the viewpoints V are equally spaced, i.e., $f_i = id$ for all $i \in \{0, C^* + 1\}$. As a particular case, if the communication range is zero, and $d_i = d$ for all $i \in \{1, \ldots, M\}$, then $L = \sum_i d_i$, i.e., the optimal latency equals the length of the environment to be patrolled.

In Section 2.5 we present a team trajectory that achieves the bound in (2.9), and that is therefore optimal.

2.5 Distributed algorithms

In order to distributely achieve the performce described in Sections 2.3 and 2.4, the robots need to solve two main tasks. First, the optimal left induced M-partition has to be found, and then a synchronization algorithm has to be implemented over the M sets of the M-partition.

The computation of the optimal left induced M-partition follows from Algorithm 2, by letting the robots compute the left-induced partition of

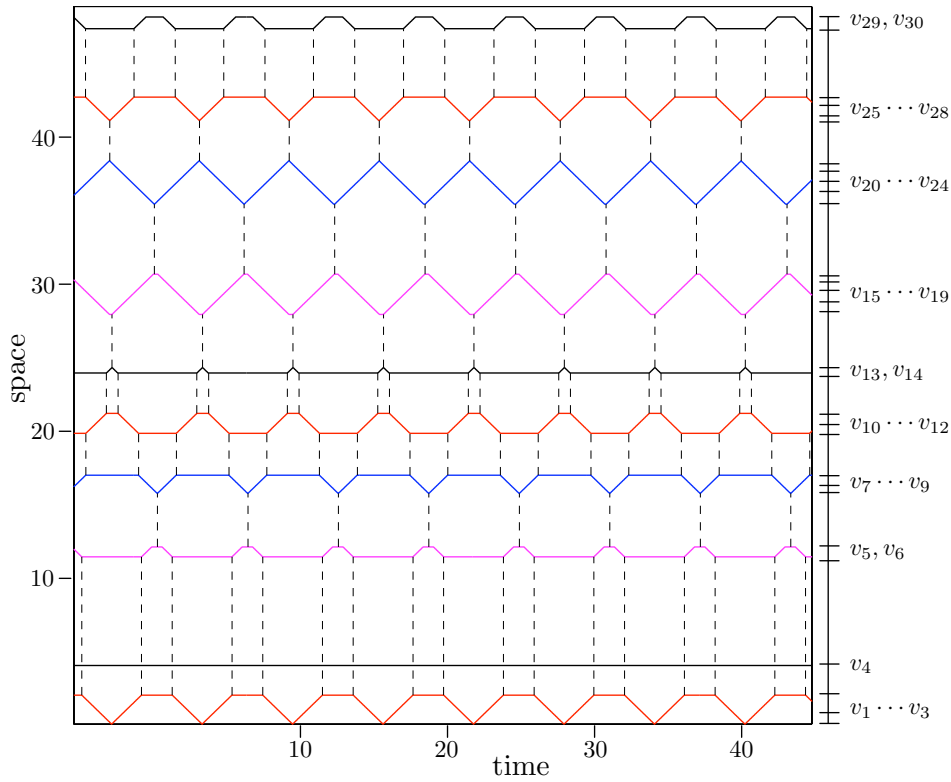

Figure 2.8: Optimal trajectories for the viewpoints in Fig. 2.12. The set of location F determines the behavior of the robots at the extreme viewpoints of their cluster. The communications are marked with dashed lines. Note that the communication radius is nonzero.

length l in a distributed way. Such computation can be performed with simple programming operations and it is not described here in details. Assume that the robots gather initially at the leftmost viewpoint of the environment, and suppose that the robots are able to explore the environment, measure the traveled distance, and recognize the viewpoints of the environment. The left-induced partition of length l can be computed by letting the robots move along the environment, and using the information coming from the odometer. In particular, if v_i is the leftmost viewpoint of i-th set of the partition, then the right extreme is the last viewpoint within distance l from v_i. If the last robot reaches the rightmost viewpoint, then $M_l \leq M$, and the successive action according to Algorithm 2 can be computed.

We focus now on the synchronization task. As usual, let the viewpoints be denoted with $V = \{v_1, \ldots, v_N\}$, let the robots be ordered from 1 to M, and assign to each robot a contiguous subset of the viewpoints. We say that a viewpoint v_i belongs to the set (2.8) if a communication occurs at v_i, and if such communication coincides with an intersection point of two

routing sequences. As an example, consider the viewpoints corresponding to the intersection points of the framework in Fig. 2.7. Let l_i and r_i be respectively the leftmost and rightmost viewpoints patrolled by the i-th robot, the following motion primitives are useful to define a team trajectory.

Definition 2.1 (Default-sweeping)**.** *The robot i implements the default-sweeping behavior if its actual position belongs to (l_i, r_i). The default-sweeping consists of sweeping at maximum speed the segment (l_i, r_i). When either l_i or r_i is reached, the i-th robot stops, and it waits until a communication with the neighbor occurs. As an exception, the robots 1 and M do not stop respectively at v_1 and v_N, because there is no neighbor to communicate with.*

The behavior of the robots after the communication with the neighbor depends upon the set (2.8), and it is described as follows.

Definition 2.2 (Naive-sweeping)**.** *The robot i implements the naive-sweeping behavior if its actual position coincides with l_i (r_i), and if l_i (r_i) does not belong to the set* (2.8)*. After communicating with the neighbor, if the i-th robot posses a communication token, then it passes the token to the neighbor, and waits until the token is returned to it. If the i-th robot does not have the communication token, then it waits until a communication token is passed to it. After receiving the token, the robot i moves toward the other side of its partition.*

Definition 2.3 (Left-sweeping)**.** *The robot i implements the left-sweeping behavior if its position coincides with l_i, and if l_i belongs to the set* (2.8)*. The i-th robot moves toward the rightmost viewpoint of its partition immediately after communicating with its left neighbor. The robots whose leftmost viewpoint belong to the set* (2.8) *posses a communication token at the beginning of the team trajectory.*

Definition 2.4 (Right-sweeping)**.** *The robot i implements the right-sweeping behavior if its position coincides with r_i, and if r_i belongs to the set F defined in* (2.8)*, i.e., there exists $f_j \in F$ such that $r_i = f_j$. Let d be the largest diameter of the partitions, then the i-th robot wait $\Delta = d - (f_j - f_{j-1})$ units of time after communicating with the right neighbor, and then moves toward the leftmost viewpoint of its partition.*

The lower bound in (2.9) is achieved by the following team trajectory, which is therefore optimal.

Theorem 2.4 (Optimal team trajectory)**.** *The team trajectory generated by the policies default-sweeping, naive-sweeping, left-sweeping, and right-sweeping solves the Problem 2.2.*

Proof. We start by showing that the team trajectory x described by the policies default-sweeping, naive-sweeping, left-sweeping, and right-sweeping

Algorithm 3: Minimum latency team trajectory

```
input : Patrolled sets, set of points F
The agents continuously sweep their cluster at maximum speed;
if a communication with the neighbor occurs then
    if position == left and left ∈ F then
        go right
    ;
    if position == right and right ∈ F then
        wait Δ units of time and then go left;
    if right, left ∉ F then
        wait for the neighbor to come back, then go towards the
        other end of the patrolled set;
```

has an optimal refresh time. Let d be the largest diameter of the partitions, and let F be the set defined in (2.8). Recall that F is such that the distance between any two consecutive elements is not greater than d, and note that, because of the waiting time Δ introduced in the right-sweeping, each viewpoint is visited at most every $2d$ instants of time. Because $2d$ is the time needed to sweep the partition of largest diameter, the team trajectory x has an optimal refresh time. For what concerns the latency, note that the time needed to transfer a message between any two consecutive points $f_i, f_{i+1} \in F$ equals d, and that the robots always move at maximum speed. It follows that the latency is given by the expression (2.10), and it is therefore optimal. □

For completeness, Fig. 2.8 shows the typical shape of the team trajectory described in Theorem 2.4. Notice that the trajectories of the robots are symmetric and periodic, and that $F = \{v_1, v_6, v_{14}, v_{19}, v_{24}, v_{30}\}$. Also, the pseudo code of a distributed algorithm that converges to the team trajectory generated by the motion primitives default-sweeping, naive-sweeping, left-sweeping, and right-sweeping is in Algorithm 3.

Lemma 2.2 (Convergence of Algorithm 3)**.** *Algorithm 3 converges to the team trajectory generated by the motion primitives default-sweeping, naive-sweeping, left-sweeping, and right-sweeping.*

Proof. A necessary and sufficient condition for the convergence of the proposed synchronization procedure is the connectivity of the communication graph, i.e., the possibility to send messages from the leftmost robot to the rightmost one. Indeed, if robot i does not communicate with robot $i+1$, then no synchronization can be achieved among the two robots, and hence, because of the linear communication topology, among the robots $\{1, \ldots, i\}$

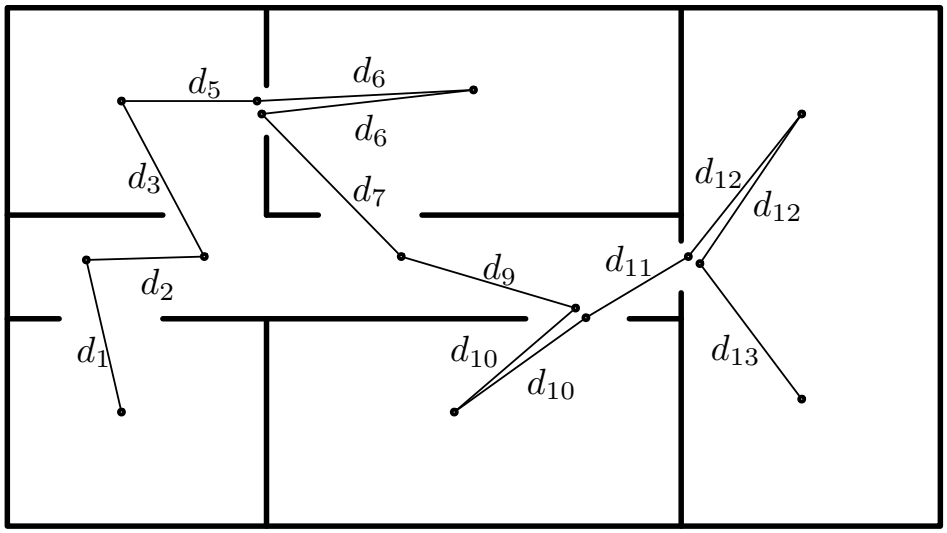

Figure 2.9: A chain graph obtained from the communication graph of Fig. 2.1. The tour that visits all the vertices has been computed starting from a spanning tree of the communication graph. Notice that 3 vertices (3 edges) are repeated twice in the chain graph, so that the refresh time and latency performance decreases.

and $\{i+1, \ldots, M\}$. Since from the partitioning procedure the robots know whether or not the viewpoints they are patrolling belong to the set defined in 2.8, as soon as two robots communicate, they are able to synchronize according to the motion primitives default-sweeping, naive-sweeping, left-sweeping, and right-sweeping. Trivially, the connectivity of the communication graph is guaranteed by Algorithm 3, because each robot waits until a communication with the corresponding neighbor occurs. □

2.6 Application to a generic graph

The optimal team trajectory described in Algorithm 3, as is, can not be implemented if the graph describing the environment has not a chain structure. For a general communication graph, optimal solutions can usually not be computed in polynomial time, and therefore effective heuristic behaviors are to be preferred. For instance, as it was pointed out in [38], one can always obtain a one dimensional communication graph by simply computing a tour visiting all the vertices. On the same note, let G be the undirected graph defined in Section 2.2, where the N vertices correspond to the viewpoints, and where two vertices are connected if it is possible for two robots to communicate when placed respectively at those vertices. Let τ be an open tour of G visiting all the N locations, and let $\bar{N}$ be te length of τ, i.e., the number

of edges in τ. Clearly there exists a tour τ such that $N-1 \leq \bar{N} \leq 2(N-1)$.[2] We associate a chain graph Γ with a minimum length tour τ, such that Γ has $\bar{N}+1$ vertices and $\bar{N}$ edge, and such that the length of the i-th edge of Γ equals the length of the i-th edge of τ, as in Fig. 2.9. In order to patrol the environment associated to G, we apply Algorithm 3 to Γ.

Theorem 2.5 (Performance ratio)**.** *Given a communication graph G, let δ be ratio of the longest to the shortest length of its edges. Denote with RT_G^{opt} the optimal refresh time over G, and with RT_G^{heur} the refresh time obtained by applying Algorithm 3 to the chain graph associated to any open tour visiting all the vertices of G. Then*

$$\frac{RT_G^{heur}}{RT_G^{opt}} \leq 4\delta, \tag{2.12}$$

Proof. Let M be the number of robots, N be the number of vertices of G, and $\underline{w}$ be the shortest length of the edges of G. Let Γ be the chain graph associated with any minimum length open tour visiting all the viewpoints. Since the number of vertexes of Γ is less than $2(N-1)$ (see footnote), the length of Γ is upper bounded by $2N\delta\underline{w}$. It follows that

$$\mathrm{RT}_{\mathrm{G}}^{\mathrm{heur}} \leq (4N\delta\underline{w})/M.$$

On the other hand, since $M < N$ by assumption, some of the robots need to move along G for all the viewpoints to be visited. Also, because each robot can visit only a location at a time, at least $\left\lceil \frac{N}{M} - 1 \right\rceil$ steps are needed to visit all the vertices of G, and therefore

$$\mathrm{RT}_{\mathrm{G}}^{\mathrm{opt}} \geq 2\left\lceil \frac{N}{M} - 1 \right\rceil \underline{w} \geq \frac{N}{M}\underline{w}.$$

By taking the ratio of the two quantities we get

$$\frac{\mathrm{RT}_{\mathrm{G}}^{\mathrm{heur}}}{\mathrm{RT}_{\mathrm{G}}^{\mathrm{opt}}} \leq 4\delta.$$

□

Note that, when δ grows, the performance ratio in (2.12) becomes arbitrarily large. For example, suppose that the locations graph is as in Fig.

[2] A simple way of computing τ is the following. Starting from any root of a spanning tree of G, let τ be a minimum length path that visits all the vertices of the spanning tree. Since any spanning tree has $N-1$ edges, and since each edge needs to be travelled twice to visit all the nodes, it follows that there exists a path τ of length $2(N-1)$. Notice that, since by assumption the communication graph G is connected, there always exists a path between any two consecutive locations.

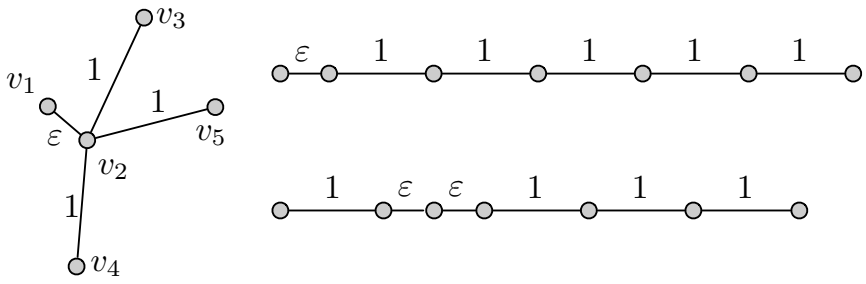

Figure 2.10: A communication graph (left), and all possible associated chain graphs (right). If the number of robots M is 4 then the performance ratio grows with $1/\varepsilon$.

2.10, and suppose that 4 robots are assigned to the patrolling task. An optimal strategy, as long as $\varepsilon < 1$, assigns the viewpoints $\{v_1, v_2\}$ to one robot, and v_3, v_4, and v_5 respectively to the second, third, and fourth robot, so that the refresh time equals 2ε. On the other hand, by patrolling any chain graph associated with a tour visiting the locations $\{v_1, v_2, v_3, v_4\}$, the refresh time is independent of ε, and it equals 2, because the optimal M-partition of any chain graph associated with a tour visiting all the viewpoints has dimension 1. As confirmed by this example, the bound (2.12) is tight.

2.7 Simulations

Suppose that we want to optimally patrol a two floors building, whose communication graph is in Fig. 2.11, with the aid of a team of 10 autonomous vehicles. As in the previous sections, the robots are assumed to be able to travel along the environment, and to communicate between any two adjacent vertices of the communication graph. After partitioning the environment using a distributed version of Algorithm 2, the robots synchronize their trajectories using Algorithm 3, and their trajectories are in Fig. 2.12. It is worth noting that only some edges are traveled but the robots, but yet all the viewpoints are visited.

2.8 Open issues

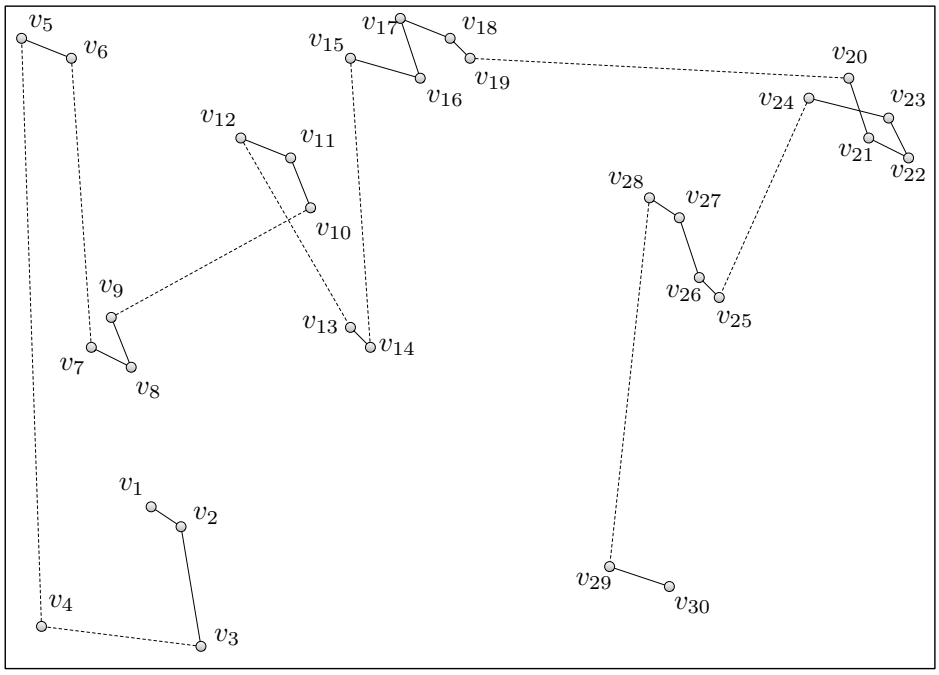

Figure 2.11: Communication graph for a two floors building. The crossing of edges corresponds to different level corridors. The environment has been divided into 10 partitions. The dashed edges are not traveled.

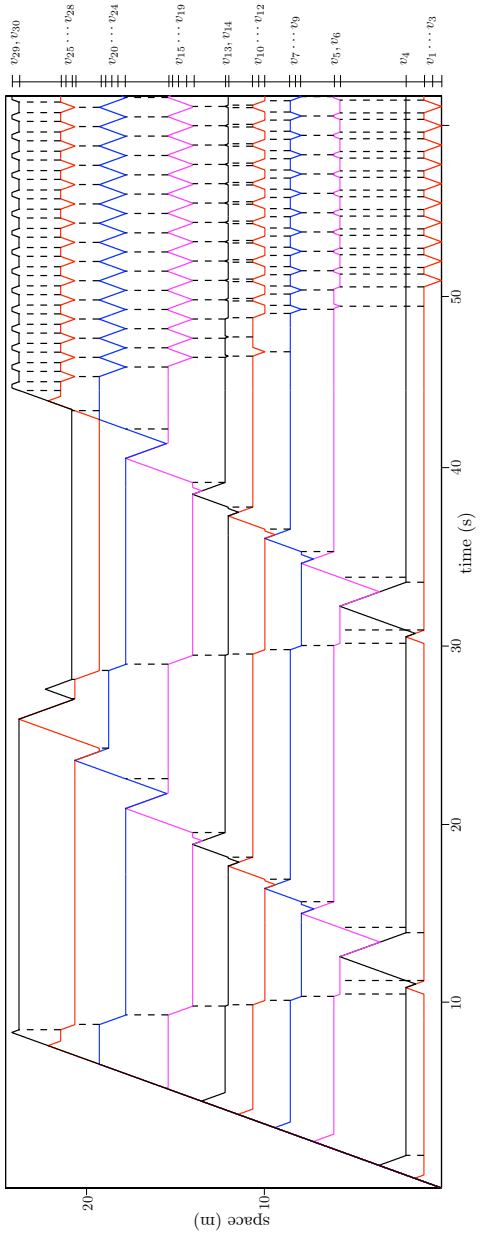

Figure 2.12: Optimal team trajectory. The speed of the robot is upper bounded by 3 m/s. The refresh time and latency of the proposed team trajectory are respectively 40% and 70% smaller than the performance obtained in the case of a 0 communication range.

Chapter 3

Distributed Pursuit-Evasion with Limited-Visibility Sensors

This chapter addresses a distributed, visibility-based pursuit-evasion problem in which one or more searchers must coordinate to guarantee detection of any and all evaders in an unknown planar environment while using only local information. Our motivation is to develop algorithms to enable teams of robots to perform bomb or intruder detection and other related security tasks. We present a distributed clearing algorithm for a team of $d-$searchers *with limited range sensors. Our algorithm is built around guaranteeing complete coverage of the* frontier *between cleared and contaminated areas while expanding the cleared area. A novel approach to storing and updating the global frontier enables our algorithm to be truly distributed. We demonstrate the functionality of the algorithm through simulations.*

3.1 Introduction

This chapter deals with a distributed pursuit-evasion problem for a team of robotic searchers in an unknown environment. The distributed *pursuit-evasion problem*, also known as the *clearing problem*, involves designing control and communication protocols such that the team of searchers will sweep an environment and detect any intruders which may be present. The pursuit-evasion problem has received a lot of attention in recent years because of its applications to safety and security. In this chapter, we describe a distributed environment clearing algorithm based on the concept of the frontier or boundary between cleared and uncleared or *contaminated* areas. Our algorithm can guarantee the detection of any intruders or, if there are insufficient searchers available, will clear as much area as it can while ensuring no cleared areas are recontaminated.

The literature on pursuit-evasion problems is vast, with many differ-

ent approaches studied. The most similar branch of prior work began with [44] and focuses on guaranteeing detection of evaders in planar environments. Gerkey et al [45] studied the case of a single searcher with limited field-of-view in a known, polygonal map. In [46], Sachs et al show that a single searcher can clear a broader class of unknown environments without localization or accurate distance measurements using an infinite range discontinuity sensor. There are also a number of works which study efficient evader detection where one or more searchers are tasked with probabilistically locating targets which move randomly, including [47]. Pursuit-evasion on graphs representing decompositions of environments is a closely related topic which goes back to [48] and includes recent works such as [49] and [50]. In addition, our work draws inspiration from methods for exploration and deployment of agents based on the frontier between explored and unexplored areas, including [7], [51], [42] and [1].

We present a distributed clearing algorithm for $d-$searchers, a searcher model with realistic limited range sensors. A well-known result from the literature is that computing the minimum number of searchers required to clear a general graph is NP-hard [48]. This result was extended in [52] to searchers with infinite range visibility sensors in a polygonal environment, and so solving for the minimum number of $d-$searchers to clear a non-polygonal environment is also NP-hard. Instead, we present an efficient, distributed algorithm which locally minimizes the number of searchers required, and demonstrate the algorithm's utility through simulations using the opensource Player/Stage robot software system [53].

There are three key contributions of this work. First, we adapt frontier-based multi-agent exploration methods for pursuit-evasion. These methods enable our algorithm to guarantee detection of evaders in unknown, multiply-connected planar environments which may be non-polygonal, a more general setting than has been covered in the pursuit-evasion literature. Second, we develop a method for picking the next positions for the searchers which locally optimizes the number of searchers required and the expected increase in the area cleared. Finally, and perhaps most importantly, we detail a novel method for storing and updating the global frontier between cleared and contaminated areas in a truly distributed manner.

This chapter is organized as follows. Section 3.2 provides definitions and states the problem which we are addressing. In Section 3.3 we examine a centralized version of our algorithm to clarify some of the details. The decentralized algorithm is presented in Section 3.4 and then demonstrated through simulations in 3.5. We conclude with a discussion of future work in Section 3.6.

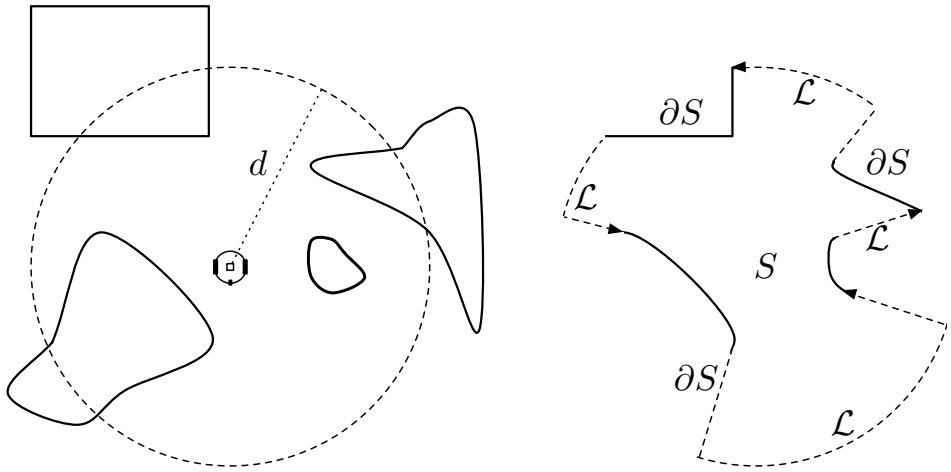

Figure 3.1: On the left, four obstacles surround a $d-$searcher and lie within the dashed circular region representing the area perceivable by the searcher's sensor without occlusions. The right image shows the boundary ∂S of the sensor footprint for this position, with dashed oriented arcs for the free boundary $\mathcal{L}$ and solid arcs for the local obstacle boundary.

3.2 Problem formulation

We are given a team of n robotic searchers with limited sensing and communication capabilities and finite memory, all initially placed at the same position in the free space of an unknown but limited planar environment. Let Q be the free space of the environment, which must be connected but can have holes and may be non-polygonal. The searchers are tasked with detecting evaders which can be arbitrarily small (even a single point) and can move arbitrarily fast, but continuously, through Q. The trajectories and initial positions of the evaders are unknown. We require that the control protocol uses a constant amount of memory per robot with respect to the size of Q.

The robot model we use, the $d-$*searcher*, is a holonomic (i.e., omnidirectional drive) mobile robot that can rotate and translate continuously at bounded speed through Q. Our model gets its name from the attached distance sensor which has a maximum range of $d > 0$ and an angular aperture of 2π. The sensor cannot penetrate obstacles but is capable of detecting any evaders visible to it. When the sensor is used only to detect evaders, i.e., the distance measurements are ignored, we refer to it as an *evader-detector*.

Let S denote the *footprint* of the sensor when a robot is in a generic configuration, as shown in Fig. 3.1. We say that a point is *guarded* by a robot if it belongs to the footprint of the sensor of that robot. The oriented *boundary* of the sensor footprint, ∂S of S, is a closed arc partitioned

into two sets: (1) the *local obstacle boundary* (all the points where the sensor has perceived an obstacle), and (2) the *free boundary*, denoted with $\mathcal{L}$, which consists of all the remaining points. Notice that while S is always a (star-shaped) simply connected region, $\mathcal{L}$ is not, in general, a connected set. We refer to the connected subsets of $\mathcal{L}$ as *free arcs*. The *orientation* of ∂S is defined in a counter-clockwise manner, such that a point moving along the boundary would have the internal part of S on the left. The free arcs constituting $\mathcal{L}$ inherit the orientation of ∂S and are an open subset of the topological manifold ∂S, with their endpoints on obstacles. The local obstacle boundary arcs, on the other hand, are closed in ∂S.

The *perception* of the sensor at a given point is the tuple $\{S, \partial S, \mathcal{L}\}$, i.e., a simply connected local obstacle-free region S called the footprint, surrounded by a closed oriented curve ∂S called the boundary, and the set of oriented free boundary arcs $\mathcal{L}$ of ∂S. We will also have use for the union of a number of perceptions from different points in Q. Let I be the union of a number of footprints from different points, which we refer to as the *inspected region*. Since our algorithm does not allow recontamination, I also represents the *cleared area*. Though I will be connected for our algorithm, it may not be simply connected, meaning that ∂I is a set of a closed oriented curves. As with ∂S, ∂I is partitioned into two sets: (1) the *obstacle boundary*, and (2) the set of remaining oriented arcs, called the *frontier boundary*, and denoted $\mathcal{F}$.

Finally, we require that a pair of robots are guaranteed to be able to communicate if their two sensor footprints intersect. In addition, we assume that two communicating robots can compute their relative poses, as a result of a mutual localization procedure [26]. The availability of any sort of global localization is not assumed.

With these definitions we can now state the goal of our algorithm: control the team of n searchers in order to maintain complete coverage of frontier $\mathcal{F}$ while expanding as much as possible the area of the cleared region, I, subject to limited communication and memory constraints.

3.3 The multi-robot clearing algorithm

For clarity, we have chosen to split the presentation of our clearing algorithm into two stages. In this Section we pretend that a central controller is commanding the searchers in order to describe the fundamental algorithm steps and the data structures involved. In Section 3.4 we detail the distributed implementation of the algorithm.

The team of n searchers is divided into two classes, the *frontier-guards* and the *followers*, and these roles are defined as follows.

- *Frontier-guard*: Each frontier-guard is assigned to a unique position $v \in Q$ called the guard's *viewpoint*, which can move during the evolu-

tion of the algorithm. The frontier-guard is required to quickly reach its assigned viewpoint and immediately report a perception from the viewpoint, i.e. the tuple $\{S, \partial S, \mathcal{L}\}$. In order to detect evaders each frontier-guard must also continuously monitor its sensor, using it as evader-detector.

- *Follower*: Each follower is assigned to follow a frontier-guard, and this assignment can change as the algorithm progresses. Each follower is only required to passively follow its frontier-guard, and can in principle use its sensor just for navigation.

As needed, the clearing algorithm will switch frontier-guards to followers, and vice-versa.

At the beginning of the algorithm all n searchers are clustered around a point in Q. One robot is selected as the initial frontier-guard and assigned its initial position as a starting viewpoint. All other robots are set as followers of this guard. The frontier-guard will then record the first perception, which initializes the main data stored during the evolution of the algorithm.

Whenever a frontier-guard records its perception from an assigned viewpoint, a new step k of the algorithm starts and the perception is classified as $\{S_k, \partial S_k, \mathcal{L}_k\}$ and called the k-th perception. We denote the total inspected region at step k as $I_k := \cup_{i=1}^{k} S_i$. The algorithm, however, does not use or store I_k or the obstacle portion of ∂I_k. One important innovation of this work is that it stores and updates only $\mathcal{F}_k$, the oriented frontier arcs of I_k. Since the obstacle boundary of the inspected region I_k is impossible for either searchers or evaders to cross, there are only two ways an evader can enter I_k: (1) by being inside of $S_k \backslash I_{k-1}$ at the instant in which the k-th perception is performed, or (2) by crossing $\mathcal{F}_k$. In this first case detection of the evader is immediate, the focus of our algorithm is thus on maintaining complete coverage of $\mathcal{F}_k$ and updating it when a new perception is added.

On the first step, $\mathcal{F}_1$ is directly initialized with the free boundary of the first perception $\mathcal{L}_1$; on each subsequent step k, $\mathcal{F}_k$ is computed from $\mathcal{F}_{k-1}$ and $\{S_k, \partial S_k, \mathcal{L}_k\}$ using the method detailed in Sec. 3.3.1. For each new $\mathcal{F}_k$ the following actions are performed:

1. Compute the next set of viewpoints V_{k+1}, which ensure that $\mathcal{F}_k$ remains guarded and that I_{k+1} will be a strict superset of I_k. As detailed in Sec. 3.3.2, this is achieved by updating V_k and adding any new viewpoints needed to cover the new portion of $\mathcal{F}_k$.

2. Assign each $v \in V_{k+1}$ to a nearby searcher and set the searcher to be a frontier-guard.

3. Assign all remaining searchers a nearby frontier-guard to follow.

4. Compute paths for all frontier-guards to reach their assigned viewpoint.

During the generation of paths the algorithm ensures that all the points of $\mathcal{F}_k$ will remain guarded by the frontier-guards during the path following; we will explain how this is achieved in Section 3.3.2. This fact guarantees that at every instant each point of $\mathcal{F}_k$ is guarded by at least one frontier-guard and thus I_k will remain clear. We refer to this feature as the *frontier guarding property.*

Assuming that $n \geq \max\{|V_k| \mid \text{for all } k\}$, the algorithm will terminate at the first step k_f where $\mathcal{F}_{k_f} = \emptyset$. At this point, ∂I_{k_f} will consist entirely of obstacle arcs and I_{k_f} will completely cover Q. Therefore, for every evader e in Q there exists at least one time step k_e during which it (1) crosses $\mathcal{F}_{k_e-1}$ for $k_e \in \{2, \ldots, k_f\}$, or (2) belongs to $S_{k_e} \backslash I_{k_e-1}$ for $k_e \in \{1, \ldots, k_f\}$. We can conclude, by means of the frontier guarding property, that every evader is eventually detected before the algorithm terminates.

3.3.1 Updating the frontier without a global map

At each step $k > 1$, the algorithm needs to compute the new frontier $\mathcal{F}_k$, i.e., the non-obstacle boundary of the inspected region $I_k = I_{k-1} \cup S_k$. The set $\mathcal{F}_k$ can be partitioned into two subsets, (1) the set $\mathcal{F}_{k-1}^{\text{Ext}}$ of arcs from the prior frontier $\mathcal{F}_{k-1}$ which do not belong to the closure of perception S_k, and (2) the set $\mathcal{L}_k^{\text{Ext}}$ of arcs from $\mathcal{L}_k$ which are not on the interior of the inspected region $I_k = \cup_{i=1}^{k} S_i$. While the computation of $\mathcal{F}_{k-1}^{\text{Ext}}$ from $\mathcal{F}_{k-1}$ and $\{S_k, \partial S_k, \mathcal{L}_k\}$ is immediate, in this section we describe a new method to compute $\mathcal{L}_k^{\text{Ext}}$ by using only the oriented arcs of $\mathcal{F}_{k-1}$ and $\{S_k, \partial S_k, \mathcal{L}_k\}$.

In all previous work including [1], $\mathcal{L}_k^{\text{Ext}}$ has been computed using S_k and I_{k-1}. The chief disadvantage of this prior procedure is that I_{k-1} has a non-constant size per robot with respect to the size of environment Q. Furthermore, at step k it is not possible in general to compute $\mathcal{L}_k^{\text{Ext}}$ using only the most recent perceptions from each frontier-guard.

Our new three-step method for computing $\mathcal{L}_k^{\text{Ext}}$ is based around the intersections of the oriented arcs of $\mathcal{F}_{k-1}$ and ∂S_k. Since it only requires the storage of $\mathcal{F}_{k-1}$, this method uses a constant amount of memory per robot regardless of the size of Q. Let $L_k^\star$ denote the set of points belonging to the intersection between the arcs of $\mathcal{L}_k$ and the arcs of $\mathcal{F}_{k-1}$ and $\bar{L}_k^\star$ the remaining points of the arcs of $\mathcal{L}_k$. The points of $\mathcal{L}_k^{\text{Ext}}$ can be either boundary or exterior points of I_{k-1}, the boundary points will belong to $L_k^\star$ while the exterior ones will belong to $\bar{L}_k^\star$. The following crucial result states that an arc in $\mathcal{L}_k$ can only switch from being on the interior or exterior of I_{k-1} at an intersection point in $L_k^\star$.

Lemma 3.1. *Let l be an arc in Q which does not intersect $\mathcal{F}_{k-1}$. If any point of l belongs to the exterior of I_{k-1}, then all of l belongs to the exterior of I_{k-1}. If any point of l belongs to the interior of I_{k-1}, then all of l belongs to the interior of I_{k-1}.*

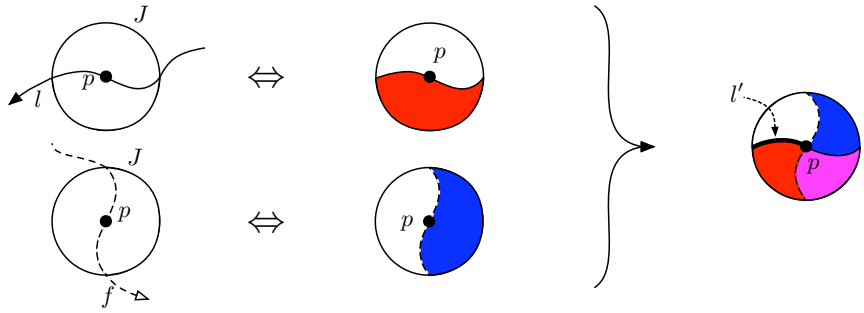

Figure 3.2: An example of the classification of the neighborhood J of a point $p \in L_k^\star$ where arcs $l \in \mathcal{L}_k$ and $f \in \mathcal{F}_{k-1}$ intersect. In the middle, the partitions of J induced by l and f are represented separately. The white regions on the right side of the oriented arcs indicate the exterior, and the colored regions on the left side of the oriented arcs indicate the interior. On the right, the single neighborhood shows the fusion of the two partitions of J. The bold part of l, denoted with l', indicates the points of l which are classified as belonging frontier $\mathcal{F}_k$ because they lie between a white and a colored region. Notice that in this case $p \in l'$.

Proof. Since l is in Q, it cannot cross the obstacle boundary of ∂I_{k-1}. Therefore, if l does not intersect $\mathcal{F}_{k-1}$, then it does not cross ∂I_{k-1}. □

The first step of the method is to classify the neighborhood on ∂S_k of each intersection point $p \in L_k^\star$ as either internal to I_{k-1} or not. An example of this neighborhood classification is shown in Fig. 3.2. The neighborhood classifications for all possible intersection cases are depicted in Fig. 3.3.

The second step of the method is to classify the ends of each arc $l \in \mathcal{L}_k$ in the neighborhood of the endpoints of the adjacent obstacle arcs. These neighborhoods can be classified using the following Lemma:

Lemma 3.2. *Let o denote a local obstacle arc of ∂S_k, let l_L and $l_R \in \bar{L}_k^\star$ denote the ends of the free arc segments on the left and right of o, respectively, in the neighborhood of the endpoints of o. Let $E_o \subset o$ be the set of endpoints of any frontier arcs of $\mathcal{F}_{k-1}$ which either begin or end on o, and which are, in the neighborhood of o, fully contained in the closure of S_k. Then:*

- *If $E_o = \emptyset$, either l_L and l_R are both internal to I_{k-1} or neither are.*
- *If $E_o \neq \emptyset$, whether l_L and l_R are internal to I_{k-1} or not depends whether the closest[1] $e \in E_o$ represents the beginning or end of a frontier arc. In particular, l_L is internal if the closest e is a beginning, and not otherwise. The opposite holds for l_R.*

[1] With respect to the distance on the arc o.

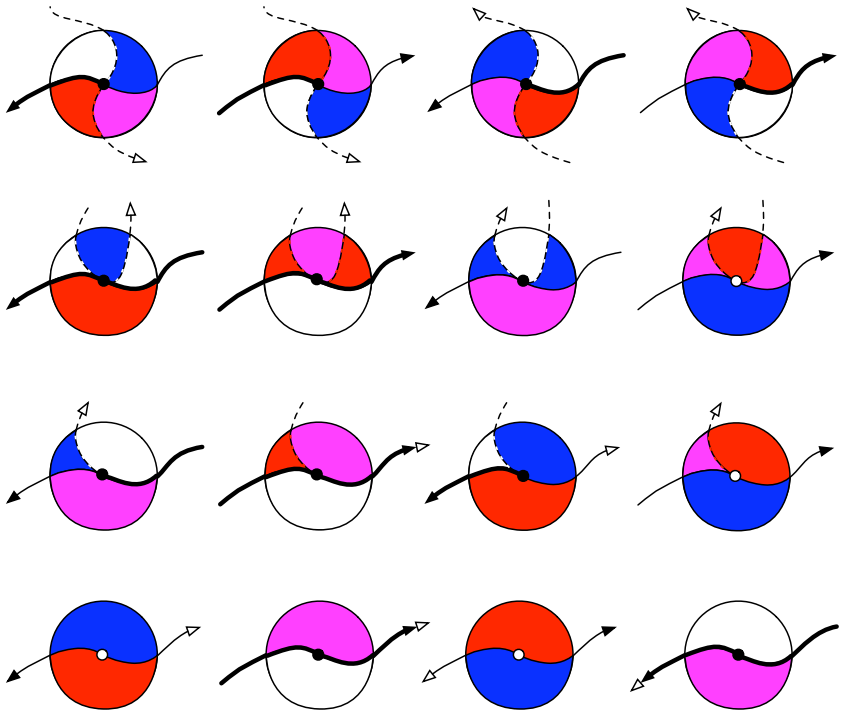

Figure 3.3: The classifications of the points of an arc $l \in \mathcal{L}_k$ in the neighborhood of all possible types of intersections with an arc $f \in \mathcal{F}_{k-1}$. Oriented arc l is drawn solid, while f is dashed and has a hollow arrow. Each row shows a different intersection type, with columns for the various reciprocal orientations of l and f. The first row shows isolated crossings, the second shows isolated tangents, the third shows joinings, and the fourth row shows segments where l and f overlap. The bold portions of l are the points classified as belonging to the new frontier $\mathcal{L}_k^{\mathrm{Ext}}$ because they lie between a white and a colored region.

Proof. If $E_o = \emptyset$, as shown in the first two cases of Fig. 3.4, then there exists a free arc connecting l_{L} with l_{R} which is contained in the interior of S_k and is close enough to o to not intersect $\mathcal{F}_{k-1}$. Therefore, we can apply Lemma 3.1.

If $E_o \neq \emptyset$, assume without loss of generality that $E_o = \{e\}$, i.e., it is a singleton, as shown in the third and fourth cases of Fig. 3.4. Then there exists a free arc connecting l_{L} with the 'nearest' half of the neighborhood of e which is in the interior of S_k and is close enough to o to not intersect $\mathcal{F}_{k-1}$, and similar claim holds for l_{R}. Therefore, we can again apply Lemma 3.1. □

The third and final step is to propagate the classification from the neighborhoods to all points of the arcs of $\mathcal{L}_k$. This propagation again exploits

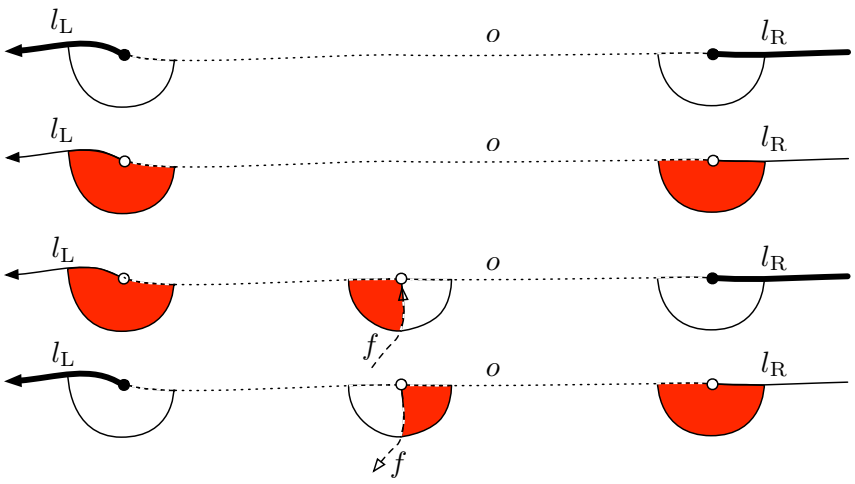

Figure 3.4: The obstacle arc o (dotted) has two adjacent free arcs (solid). The four cases are depicted. In the first two cases no internal frontier arc has an endpoint on o, so in the neighborhood of o the free arcs are classified as both frontier (bold) or both internal (thin). In the second two cases an internal frontier arc f (dashed) has an endpoint on o which induces a different classification on the two free arcs.

Lemma 3.1. Notice that, so long as the selection of viewpoints guarantees that either $L_k^\star \neq \emptyset$ or at least one local obstacle arc o has a non-empty E_o, this third step is well defined.

Combined, these three steps determine which segments of the k-th free boundary $\mathcal{L}_k$ are not in the interior of I_{k-1} and thus should be included in frontier $\mathcal{F}_k$.

3.3.2 Viewpoint planner

Our approach to viewpoint planning is similar to the exploration algorithm described in [1]. The properties of our viewpoint planning method are laid out in the following Proposition.

Proposition 3.1. *Given the frontier $\mathcal{F}_k$ and the set of prior viewpoints V_k, the viewpoint planner will select the smallest set of viewpoints $V_{k+1} \in I_k$ which satisfy the following constraints:*

1. *Area(I_{k+1}) will be strictly greater than Area(I_k), and*
2. *$\mathcal{F}_k$ is contained in the closure of $\cup_{v \in V_{k+1}} S(v)$.*

Within these constraints, the viewpoint planner will maximize the expected area exposed, Area(I_{k+1}) − Area(I_k).

Remark The viewpoint planner we present here is for circular sensor footprints of radius d. For more general footprints, such as a limited field-of-view, our clearing algorithm could also be applied provided a viewpoint selection method was developed which met the conditions of Proposition 3.1.

Let v_k be the viewpoint of the k-th perception. As detailed in Section 3.3.1, $\mathcal{F}_k$ can be partitioned into two sets: $\mathcal{F}_{k-1}^{\text{Ext}}$ (a subset of the prior frontier), and $\mathcal{L}_k^{\text{Ext}}$ (a subset of ∂S_k). Let $\mathcal{F}_{k-1}^{\text{Int}}$ be the portion of $\mathcal{F}_{k-1}$ which is inside the closure of S_k.

With the distributed application in mind, we simplify the planning of V_{k+1} by constructing it from V_k as follows:

1. Remove v_k.
2. Remove any $v \in V_k$ which was assigned to guard an obsolete portion of the frontier in $\mathcal{F}_{k-1}^{\text{Int}}$.
3. Add a set of new viewpoints V' to cover and expand the new frontier segments $\mathcal{L}_k^{\text{Ext}}$.

We will now describe how to choose V' around viewpoint v_k when $\mathcal{L}_k^{\text{Ext}} \neq \emptyset$.

A free arc $l \in \mathcal{L}_k$ is considered *relevant* for viewpoint planning if it contains one or more frontier arc fragments from $\mathcal{L}_k^{\text{Ext}}$. A relevant free arc may contain one or more frontier arc fragments, and each frontier arc fragment will be entirely contained in one relevant free arc. Let $\mathcal{L}_k^{\text{Rel}} \subseteq \mathcal{L}_k$ denote the set of relevant free arcs around v_k.

The goal of this local viewpoint planning can then be restated as partitioning the frontier points of each $l_{\text{Rel}} \in \mathcal{L}_k^{\text{Rel}}$ among the fewest possible new viewpoints V' while maximizing the expected exposed area beyond $\mathcal{L}_k^{\text{Ext}}$.

The first step in our method for local viewpoint planning is to determine how many new viewpoints will be needed to cover each $l_{\text{Rel}} \in \mathcal{L}_k^{\text{Rel}}$. As shown in Fig. 3.1, each l_{Rel} will be comprised of straight radial segments and circular segments with radius d. The possible configurations are: single radial; single curved; curved with radial on one side; or curved with radial segments on both sides. The following Lemma simplifies the determination of when a radial segment is covered by a viewpoint.

Lemma 3.3. *Let $v' \in S_k$ be a potential new viewpoint, and $r \in \mathcal{L}_k^{Rel}$ be a radial free arc segment. Let p be the far endpoint of r and $\overline{v'p}$ be the line segment between v' and p. If* $\operatorname{dist}(v', p) < d$ *and $\overline{v'p}$ only intersects ∂S at p, then open set r will be contained inside of $S(v')$.*

Proof. Our proof centers around the triangle T formed by v_k, v', and p. As r is a radial free arc segment, r is a connected subset of $\overline{v_k p}$. Then, since S_k has maximum radius d, $\operatorname{dist}(v', v_k) < d$. Combined with the fact that

$\operatorname{dist}(v', p) < d$, we can conclude that all of r is within d of v'. All that remains is to show that there are no obstructing obstacles inside of T.

We know that $\overline{v'p}$ is contained in the closure of S_k because it only intersects ∂S at p. Since S_k is star-shaped, both $\overline{v_k p}$ and $\overline{v_k v'}$ will also be contained in the closure of S_k. Then, as S_k is simply connected, we can conclude that the interior of T is in Q and, therefore, r will be contained inside of $S(v')$. □

There are two notable consequences of Lemma 3.3. First, for any l_{Rel} comprised solely of a radial segment, only one viewpoint will be needed. Second, for any l_{Rel} which contains both curved and radial segments, we only need to partition the curved segment: the viewpoint which covers an endpoint of the curved segment will also cover any attached radial segment.

To assist in selecting V' we introduce parameter $d_{\min} \in (0, d]$, the minimum distance between the v_k and any $v \in V'$. As will become clear, $d_{\min}$ encodes a trade-off in the algorithm: smaller values of $d_{\min}$ reduce $|V'|$ and thereby reduce the number of searchers required; larger values of $d_{\min}$ increase the expected area exposed and thereby reduce the number of iterations required to clear Q.

Let $\delta(l_{\text{Rel}})$ be the angular width of l_{Rel} measured counter-clockwise from the angle of the right-most frontier point on l_{Rel} to the angle of the left-most frontier point on l_{Rel}.

A single new viewpoint at least $d_{\min}$ from v_k can then cover an angular width of at most $\alpha(d_{\min})$ given by

$$\alpha(d_{\min}) = 2\arccos(d_{\min}/2d) \in [2\pi/3, \pi).$$

The number of viewpoints η necessary to cover l_{Rel} is then determined by the following Lemma:

Lemma 3.4. *For any $l_{Rel} \in \mathcal{L}_k^{Rel}$, η viewpoints will be required where $1 \leq \eta \leq 3$. In particular:*

- *if $\delta(l_{Rel}) \leq \frac{2\pi}{3}$, $\eta = 1$,*
- *if $\frac{2\pi}{3} < \delta(l_{Rel}) < \pi$, $1 \leq \eta \leq 2$,*
- *if $\pi \leq \delta(l_{Rel}) < 2\pi$, $2 \leq \eta \leq 3$, and*
- *if $\delta(l_{Rel}) = 2\pi$, $\eta = 3$.*

Proof. This result is a direct consequence of Lemma 3.3 and the fact that $\alpha(d_{\min}) \in [2\pi/3, \pi)$. □

For $\eta > 1$, the angular width of l_{Rel} is then partitioned such that the first viewpoint covers $[0, \delta(l_{\text{Rel}})/\eta]$, and each subsequent viewpoint covers the next equally sized slice.

After this first step of the local viewpoint planning method, we know how many new viewpoints v' are needed to cover each l_{Rel}. For every v' there are two points, p_1 and $p_2 \in l_{\text{Rel}}$ which must be covered: for a single radial frontier arc, p_1 and p_2 are the endpoints of the arc; for any other shape, p_1 and p_2 are the endpoints of the partition of the curved segment in l_{Rel} assigned to v'. Let $S_k(p_1)$ and $S_k(p_2)$ be the subsets of S_k which are known to be visible from p_1 and p_2, respectively.

The final step of our local viewpoint planning method is to optimize the placement of each v' to maximize the expected area it will expose. To achieve this, we choose v' as a point in $S_k(p_1) \cap S_k(p_2)$ which minimizes the sum of the distances to each frontier point in the partition of l_{Rel} assigned to v'.

By construction, this method of selecting V' guarantees that $\mathcal{L}_k^{\text{Ext}} \in \cup_{v' \in V'} S(v')$. The following Lemma states that it also ensures that $\text{Area}\,(I_{k+1}) - \text{Area}\,(I_k) > 0$:

Lemma 3.5. *For each $v' \in V'$, $S(v')$ will cover some new area $A \in Q$ where* $\text{Area}\,(A) > 0$ *and* $\text{Area}\,(A \cap I_k) = 0$.

Proof. Let $f \in l_{\text{Rel}}$ be a frontier segment assigned to v'. By definition, f is inside of Q and is a subset of $\mathcal{F}_k$. By Lemma 3.3, f will also be inside of the open set $S(v')$, so there must be an open set $A \subset Q$ which is inside of $S(v')$ but outside of I_k. □

3.4 The distributed clearing algorithm

For the distributed clearing algorithm the communication graph is in general disconnected, necessitating several changes from the centralized description. First, the global frontier must be stored and updated in a distributed manner. Second, viewpoint planning must be performed locally by the frontier-guards. Furthermore, the algorithm cannot rely on a global localization system. Finally, note that while the centralized version is synchronous, in the distributed setting it is possible for perceptions from disconnected searchers to be recorded at the same time.

The global frontier can be distributed by having each frontier-guard store its local frontier segments and update them through communication with its neighboring frontier-guards. This distributed storage and updating can always be achieved, since (1) by the frontier guarding property, each global frontier point is guarded by a frontier-guard; (2) the classification of $\mathcal{L}_k$ requires only those frontier segments which intersect it, and by assumption two robots whose footprints intersect are in communication and are mutually localized.

Once the local frontier for a frontier guard has been classified, viewpoint planning relies only on the local information. In addition, the execution of

Procedure `Expand`

```
Data: frontier,path
foreach follower in followers do
    Send(follower, "follow",path);
Move(path);
{S, ∂S, L} ← Perceive();
neighbFront ← UpdateNeighbFront();
frontier ← Frontier({S, ∂S, L},frontier,neighbFront);
DoBehavior("Frontier-Guard",S,frontier);
```

the path between viewpoints can be done without global localization; since both viewpoints lie inside the local perception, local odometry of reasonable accuracy suffices. In fact, frontier updating is also based only on current relative positions, not the absolute position. The distributed algorithm can, therefore, continue to clear an environment even if the searchers cannot determine where they started. The only requirement for success is that the bias of each step of incremental localization is small enough that it does not void the frontier guarding property or the increase in the cleared area, which is simple to obtain in practice.

The two classes of searchers from the centralized algorithm are each split in two, yielding four possible states: *expand*, *frontier-guard*, *follow*, and *wander*. These roles are described as follows:

- *Expand*: When a searcher is assigned a new viewpoint to move to, it enters the expand state until it reaches the viewpoint and records a perception.
- *Frontier-guard*: Each frontier-guard i will remain stationary at its viewpoint and has complete control over its local frontier segments, $\mathcal{F}_{k,i}$. It must communicate with its neighboring frontier-guards to keep $\mathcal{F}_{k,i}$ updated, plan viewpoints to cover and expand $\mathcal{F}_{k,i}$, and then decide whether to expand itself or dispatch a follower.
- *Follow*: As in the centralized setting, a follower's task is to passively follow their frontier-guard but they will now be given commands by the frontier-guard.
- *Wander*: Wanderers can be thought of as followers who have not yet found a frontier guard to follow. In the distributed setting, when a frontier-guard has no local frontier to guard, it and its former followers must wander until they come within communication range of a frontier-guard to follow.

Procedure `Frontier-Guard`

```
Data: S,frontier
if frontier is empty then
    Send(followers, "wander");
    DoBehavior("Wander");
(bestVP,NumVPs) ← ViewPointPlan(S,frontier);
path ← PathToViewPoint(S,bestVP);
if NumVPs == 1 then
    DoBehavior("Expand",frontier,path);
else
    if followers has at least one follower then
        follower ← PopFollower(followers);
        Send(follower, "expand",path);
        WaitForFollower(follower);
    else
        while no new neighbor and no followers do
            Sleep();
    DoBehavior("Frontier-Guard",S,frontier);
```

Procedure `Follow`

```
Receive(Leader,message,path);
switch message do
    case "follow"
        Move(path);
    case "expand"
        DoBehavior("Expand",∅,path);
    case "wander"
        DoBehavior("Wander");
```

Procedure `Wander`

```
SearchForLeader();
if leader found then
    DoBehavior("Follow");
if all searchers wandering then
    exit
```

The four-state state machine for the distributed algorithm is explained in in Fig. 3.5 and in the pseudocodes. The key subroutines consist of the

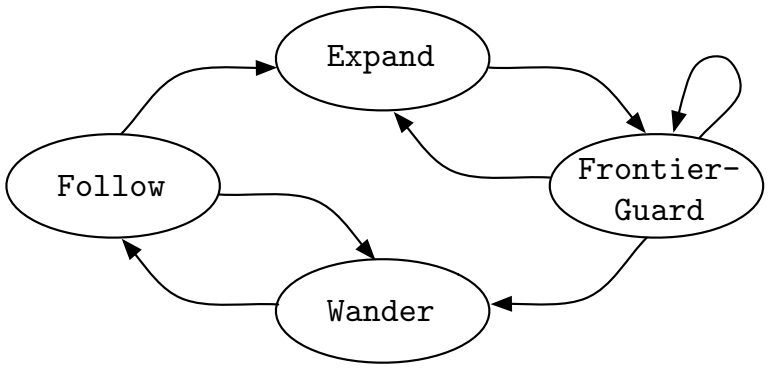

Figure 3.5: State machine diagram for the distributed clearing algorithm.

following:

- `UpdateNeighbFrontier/Frontier`: These two functions perform a localized version of the frontier update method described in Section 3.3.1. Searcher i first queries its neighbors for their current frontier segments, classifies $\mathcal{L}_{k,i}$ using its neighbor's segments, and then informs its neighbors if any of their frontier segments lie within $S_{k,i}$.

- `ViewPointPlan`: This function follows the viewpoint planning method laid out in Section 3.3.2. First, it determines how many viewpoints are needed based on the number and angular width of the relevant free arcs. Then, the single best new viewpoint is chosen from $S_{k,i}$.

- `PathToViewPoint`: This function determines the shortest path from the old viewpoint to the new viewpoint inside $S_{k,i}$. If the sensor footprint is star-shaped, this path is a straight line.

- `SearchForLeader`: This function does a random walk of the environment with two additional behaviors. First, if a wanderer encounters a frontier-guard or expander, then it switches to following this leader. Second, when two wanderers come in contact they may join together to form a wandering blob.

The behavior of the frontier-guards in this distributed clearing algorithm guarantees the frontier guarding property first discussed in Section 3.3. When expander i reaches its assigned viewpoint and makes a perception, it then enters the stationary frontier-guard state. So long as i remains a frontier-guard it will maintain complete coverage of the frontier segments in $\mathcal{F}_{k,i}$. Searcher i will only leave the frontier-guard state if either $\mathcal{F}_{k,i}$ is erased by a neighbors frontier update, or if i determines that one viewpoint is sufficient to expand while covering $\mathcal{F}_{k,i}$ and that the path to the viewpoint maintains coverage of $\mathcal{F}_{k,i}$.

The combination of the frontier guarding property and Lemma 3.5 guarantees that, assuming there are sufficient searchers available, the distributed algorithm will successfully clear all of Q. When the task is completed, all searchers will be in the Wander state. If all-to-one communication is available (for example, if all robots can communicate back to a central security center), then detecting task completion is trivial. In the most general case, the searchers will have to determine the task is complete by locating and querying the other searchers. In the absence of global localization or other means of assuring rendezvous, our proposal is that robots in the Wander state clump together when they find each other to form wandering blobs. Eventually, through the random walks of these growing blobs, all searchers will be joined into a single blob and task completion can be easily detected.

3.5 Simulations

To demonstrate the utility of the proposed pursuit-evasion algorithm, we implemented it in the open-source Player/Stage robot software system [53] using the Multirobot Integrated Platform [54]. Perceptions are implemented as local occupancy grids, with oriented frontier arcs handled as ordered sequences of cells. Each robot stores only its most recent perception and its local frontier.

The first simulation features three searchers clearing an environment with two large holes and three evaders. The progression in Fig. 3.6 begins at the top-left, where the robots start expanding from a corner of the map. The second screenshot shows the searcher clearing the lower hallway waiting for help to cover its two frontier segments. Once the central vertical hallway is cleared by another searcher, the trio continue on to complete their sweep. Fig. 3.8 shows the total percentage of the free space cleared by the robots for each iteration (where a new iteration begins with each recorded perception), as well as the number of frontier cells stored per frontier-guard.

The second simulation has six searchers tracking down five evaders in a larger multiply-connected environment. Two screenshots are shown in Fig. 3.7, where the robots begin in a corner of the map before spreading out to cover the free space. Notice that in the second image there are two groups of searchers which are separated and cannot communicate with each other. Fig. **??** shows the total area cleared and the frontier cells stored over the iterations of the algorithm. As in the previous example, this plot also shows that the number of frontier cells stored per frontier-guard is independent of the area cleared.

The third and final simulation consists of twelve searchers expanding in a vast empty environment. The final configuration where the agents cannot expand any further is shown in Fig. 3.9. Through only the local viewpoint optimizations discussed in Section 3.3.2, the final configuration

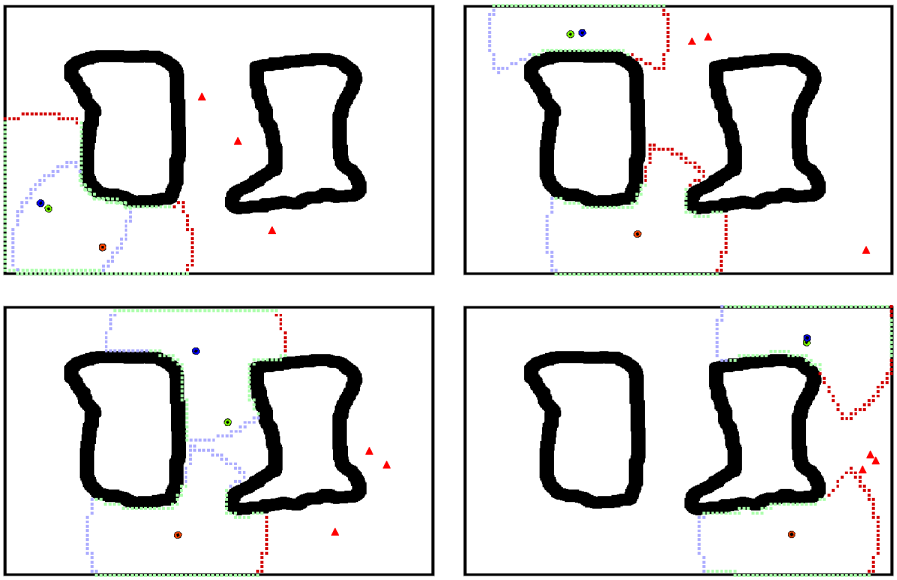

Figure 3.6: Screenshots from a simulation of three circular robots sweeping an environment with two holes to locate the triangular evaders. The images are ordered left-to-right and then top-to-bottom. The discretized boundary of each frontier-guard's current footprint, ∂S, is shown using colored squares where dark-red is used for local frontier segments, light-green for obstacle segments, and light-blue for the remaining free arcs.

closely resembles the maximum possible cleared area where the searchers are equally spaced along the circumference of a large circle with their sensor footprints just touching.

3.6 Open issues

There are a number of interesting future directions for this work. First, the specification of a general viewpoint planner which works for sensors with a limited field-of-view would extend the algorithm to a much broader class of searchers and sensor hardware. The development of an upper bound on the number of $d-$searchers required to clear an environment based on d and properties of the environment would also be a significant contribution. One potential extension of the algorithm would be to guarantee a connected communication graph for the team of searchers at all times, perhaps including a connection back to the initial entry point of the team. Finally, the algorithm could also be extended to three-dimensional environments.

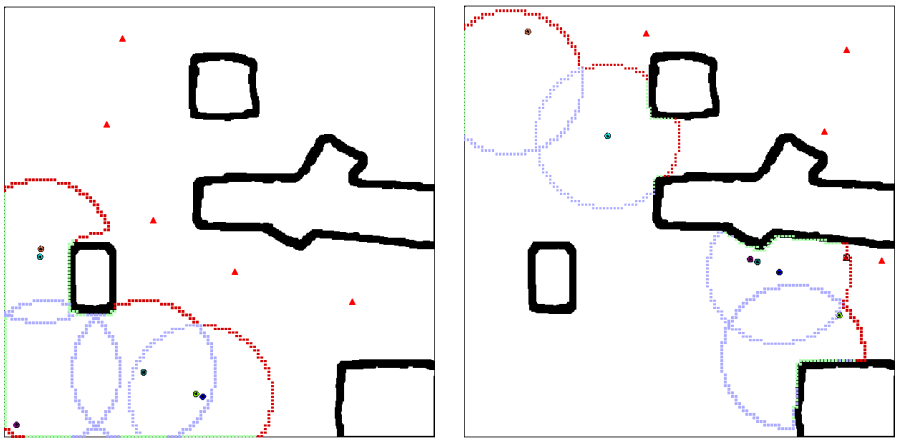

Figure 3.7: Two screenshots from a simulation of six circular robots clearing an environment containing five triangular evaders.

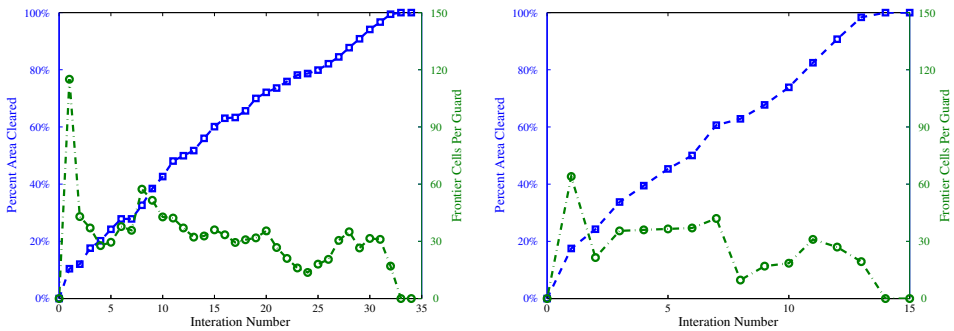

Figure 3.8: Plot showing statistics over the iterations of the simulation in Fig. 3.7 (left) and Fig. 3.6 (right). The percentage of the total free space exposed is plotted with blue squares against the left axis, while the frontier cells stored per frontier-guard is plotted with green circles against the right axis.

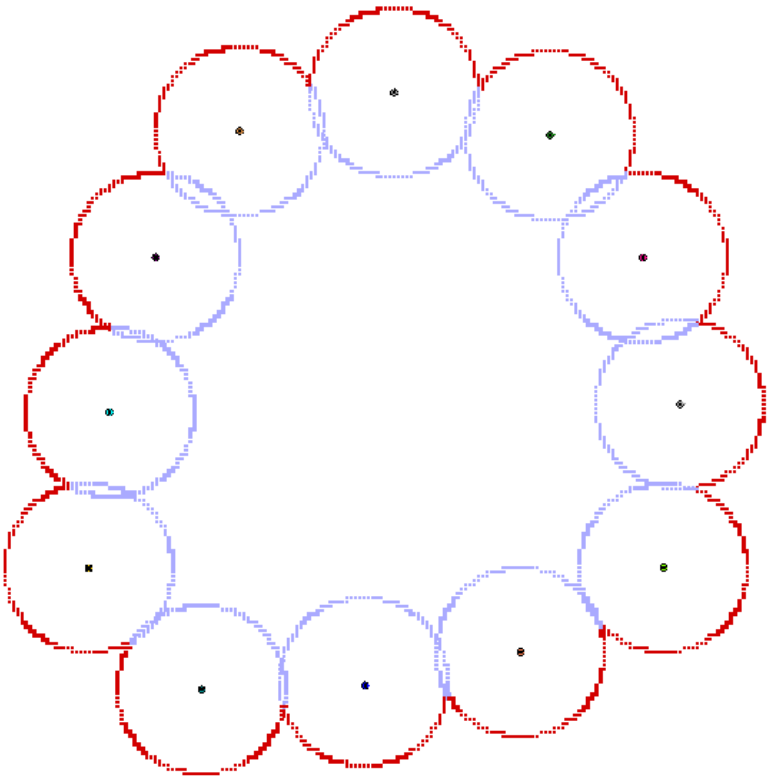

Figure 3.9: Final configuration of 12 robots clearing as much area as the can in an empty environment.

Figure 3.10:

Part II

Cooperative Tasks with Anonymous Measures

This part presents two problems in the field of the anonymous-measure scenario.

Problem (Mutual Localization w/o Perception Tagging) *Given a team of mobile robots with limited sensing and communication range, and untagged perceptions*

- *estimate the change of coordinates between the local frames of the robots*

Typical applications are all the tasks needing data exchange (e.g., sensor fusion, formation control).

Problem (Cooperative Target Encircling) *Given a team of mobile robots with limited sensing and communication range, and untagged perceptions*

- **detect** *a target and revolve around it, while being uniformly spaced and proceeding at a given speed*

Typical Applications are *observing*, i.e, retrieving data about an object from different viewpoints *escorting*, i.e., protecting a member of the team from exterior agents, *entrapment*, i.e., prevent the motion of an alien object.

Chapter 4

The Relative Mutual Localization with Anonymous Position Measures

This chapter theoretically formulates and investigates a novel problem called Mutual Localization with Anonymous Position Measures. This is an extension of Mutual Localization with Position Measures, with the additional assumption that the identities of the measured robots are not known. We have presented preliminary results regarding this topic in [55]. A necessary and sufficient condition for the uniqueness of the solution is presented, which requires $O(n^2/\log n)$ to be verified and is based on the notion of rotational symmetry in $\mathbb{R}^2$. We also derive the relationship between the number of robots and the number of possible solutions, and we classify the solutions in a number of equivalence classes which is linear with n. A control law that effectively breaks symmetric formations so as to guarantee the unique solvability of the problem is also proposed. Finally, we demonstrate the performance of this control law through simulations.

4.1 Introduction

The *Mutual Localization* problem has received a lot of attention in recent years [56, 57, 58], due to the fact that its solution is essential to perform many multi-robot tasks, e.g., coverage and deployment [59], exploration and map building [1], formation control [60], surveillance and monitoring [38], escorting and entrapment [61]. For a thorough classification see Appendix A.

Great relevance has been given in the literature to the determination of conditions for the uniqueness of the solution when the structure of available measures, represented with a measuring graph, is subject to changes. In [56] the range measurement case is addressed and the uniqueness of the solution is connected to various concepts of rigidity, while in [62] observability properties are studied in a dynamic scenario. In [57] this analysis is extended to the position measurement case, and in [58] the orientation of the sensor is also estimated. It is known [58] that the uniqueness of the solution is guaranteed if the measuring graph is complete.

We define an extension of the above problem, called *Mutual Localization with Anonymous Position Measures*, by adding the assumption that the set of relative position measurements of each robot is not ordered, in the sense that each measurement comes without the ID of the measured robot. This situation typically arises when the robot detection system is based on a feature extraction module that looks for physical characteristics that are common to all robots, e.g., size, color, or shape in a team of identical robots. For example, in [26] we have made use of a robot detector based on a laser range finder, that is unable to provide the identity of the measured robot.

In practical applications, adverse environmental conditions (unstable light or darkness, rain, fog, etc.) may hide the distinguishing features which are commonly used for identification. Hence, the possibility of relying on a localization system which does not require the identities of the robot makes the system more robust. On the other hand, the resemblance of the members of the team can be essential in missions where the leader identity must remain secret for security or disguising purposes, e.g., in escorting or intrusion. In the same spirit, other important fields of applicability are team missions requiring mimicry or stealth capabilities of the members. In fact, distinguishing features are invalidated or forbidden by such objectives.

In opposition to the standard Mutual Localization with Position Measures, the problem of Mutual Localization with Anonymous Position Measures may have more than one solution even if the measuring graph is complete. For example, consider the case of 4 robots that are arranged over the vertexes of a square, aiming their 'noses' cyclically at each other. Since the 4 set of measures of each robot are identical, all the 24 vertex associations corresponding to the permutations of the sequence $\{1, 2, 3, 4\}$ are feasible. Hence, in this situation, in spite of the complete availability of the the measures, the problem is not uniquely solvable. Note that, up to roto-translations, the number of possible vertex associations is $(n-1)!$ for a regular n-gon.

If the problem is formulated in a stochastic setting to take into account

measurement noise, the cases with multiple solutions correspond to large uncertainties in the probability density of the solution. This happens already when the configuration of the team is in the neighborhood of configurations that give rise to multiple solutions in the deterministic case. This fact emphasizes the general importance of this problem, which goes beyond the deterministic case.

It is also worth noting that multiple solutions appear, in particular, when the team is arranged along a regular pattern; this situation is not rare in multi-robot systems, and in fact many collective motion controllers attempt to achieve exactly this kind of formation, see for example [60].

A loosely related problem is the registration of multiple representations of the same scene, which has been widely investigated in the literature, from early works [63] to more recent ones [64].

The chapter is organized as follows. The problem under consideration is formalized in Section 4.2. The main findings of the chapter are presented in Sections 4.3 and 4.4. In particular, in Section 4.3 we characterize the situations in which the problem has multiple solutions, we establish the relationship between the number of possible solutions and the number of robots, and we propose an efficient representation of the solutions. A control law that breaks symmetric formations so as to guarantee the unique solvability of the problem is proposed in Section 4.4, and is validated through simulations in Section 4.5. Finally, some future work is discussed in Section 4.7.

4.2 Problem formulation

Assume that we have a group of n single-body mobile robots in a certain spatial arrangement on the plane. Each robot is equipped with a sensory system that provides the positions (not the orientations) of the other robots with respect to itself; these positions are anonymous, in the sense that they are not labeled with the identity of the robots. We want to investigate the conditions under which the spatial arrangement of the group can be uniquely reconstructed up to roto-translations from the simultaneous knowledge of all sensory data. Below, we give a formal statement of this problem.

The i-th robot $\mathcal{R}_i$, $i = 1, \ldots, n$, is a planar rigid body with an attached frame $\mathcal{F}_i$ (see Fig. 4.1a). The *pose* $x_i = (p_i, \theta_i)$ of $\mathcal{R}_i$ is an element of $\mathbb{R}^2 \times S^1$, with p_i representing the origin[1] of $\mathcal{F}_i$ expressed in a reference frame $\mathcal{F}$ and θ_i the orientation of $\mathcal{F}_i$ w.r.t. $\mathcal{F}$. Since $\mathbb{R}^2 \times S^1$ is homeomorphic to $SE(2)$, any pose may also be interpreted as a roto-translation.

[1]For simplicity, we shall use the same symbol (e.g., p) to indicate a point and its Cartesian coordinates; the actual meaning will be clear from the context.

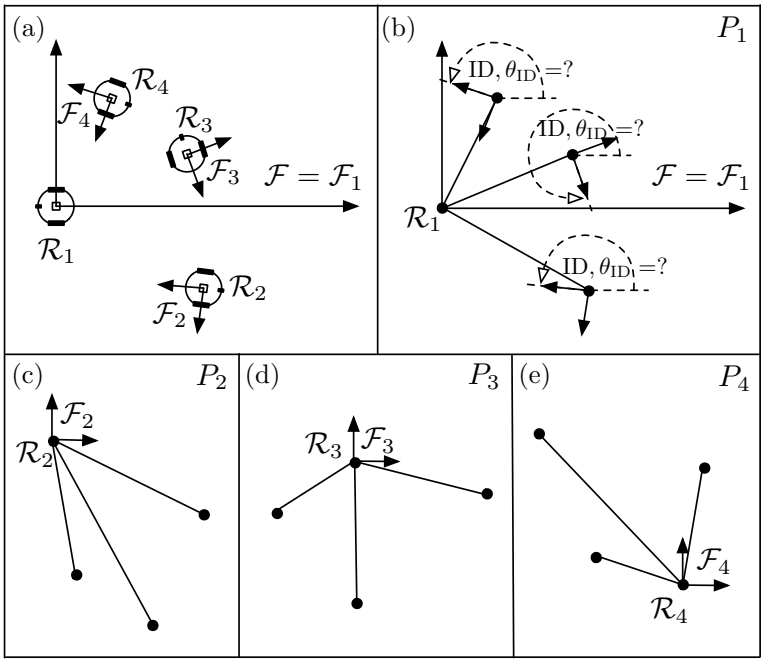

Figure 4.1: Mutual Localization with Anonymous Position Measures – Complete measuring graph. (a) Each robot expresses its measures in an attached frame $\mathcal{F}_i$; the reference frame $\mathcal{F}$ is chosen w.l.o.g. to be $\mathcal{F}_1$ (b) P_1 is the observation of $\mathcal{R}_1$ (c-e) P_2, P_3 and P_4 are the observations of the remaining robots $\mathcal{R}_2$, $\mathcal{R}_3$ and $\mathcal{R}_4$, respectively. Note that all observations are anonymous sets of points.

A *formation* is a set of n poses $\{x_1, \ldots, x_n\}$ in $\mathcal{F}$, with x_i assigned to $\mathcal{R}_i$. Since we are interested in computing the group formation up to roto-translations, we can set w.l.o.g. $\mathcal{F} = \mathcal{F}_1$, so that $x_1 = ((0\ 0)^T, 0)$. This means that all formations will be expressed in the frame attached to $\mathcal{R}_1$. Clearly, all results can be expressed in another frame $\mathcal{F}'$ provided that the pose of $\mathcal{R}_1$ w.r.t. $\mathcal{F}'$ is known.

Let $R(\phi) \in SO(2)$ denote the rotation matrix associated to an angle ϕ. As in [63] and [65], we denote by $x_a \oplus x_b$ and $x_a \ominus x_b$, respectively, the composition and the inverse composition of two poses, as defined by the following formulas:

$$x_a \oplus x_b = (p_a + R(\theta_a)p_b, \theta_a + \theta_b)$$
$$x_a \ominus x_b = (R(-\theta_b)(p_a - p_b), \theta_a - \theta_b),$$

and depicted in Fig. 4.2.

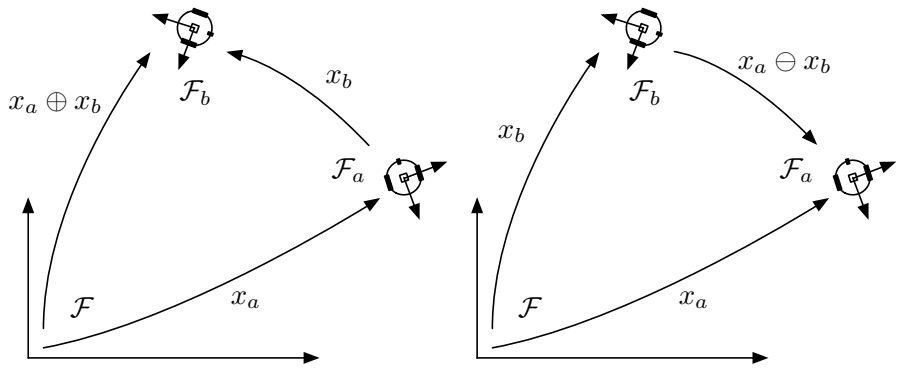

Figure 4.2: Pose compositions.

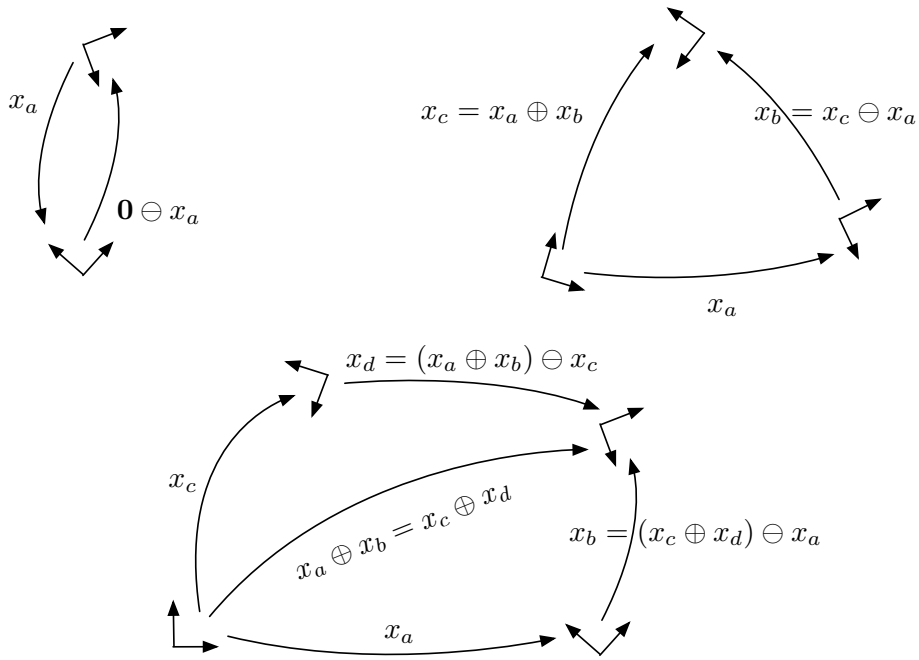

Figure 4.3: Pose composition properties.

It is worth noticing that $x_a \oplus x_b$ represents the rototranslation of x_b by

means of x_a. The operator $\oplus$ has the following properties

$$
\begin{aligned}
(x_a \oplus x_b) \oplus x_c &= x_a \oplus (x_b \oplus x_c), && (\oplus \textit{ is associative})\\
x_b \oplus x_a &\neq x_a \oplus x_b, && (\oplus \textit{ is not commutative})\\
x_a \oplus \mathbf{0} &= \mathbf{0} \oplus x_a = x_a. && (\textit{null element})
\end{aligned}
$$

The operator $\ominus$ has the following properties

$$
\begin{aligned}
(x_a \ominus x_b) \ominus x_c &\neq x_a \ominus (x_b \ominus x_c), && (\ominus \textit{ is not associative})\\
x_b \ominus x_a &\neq x_a \ominus x_b, && (\ominus \textit{ is not commutative})\\
x_a \ominus \mathbf{0} &= x_a \neq \mathbf{0} \ominus x_a. && (\textit{right null element})\\
x_a \ominus x_a &= \mathbf{0}, && (\textit{self inversion})
\end{aligned}
$$

The two operators have the following joint properties

$$
\begin{aligned}
x_b \oplus (x_a \ominus x_b) = (x_b \oplus x_a) \ominus x_b &= x_a, && (\ominus \textit{ and } \oplus \textit{ cancellation})\\
x_a \oplus x_b &= x_a \ominus (\mathbf{0} \ominus x_b), && (\oplus \textit{ exchange})\\
x_a \ominus x_b &= (\mathbf{0} \ominus x_b) \oplus x_a, && (\ominus \textit{ exchange})\\
(x_a \oplus x_b) \ominus x_c &\neq x_a \oplus (x_b \ominus x_c), && (\oplus \textit{ is not associative with } \ominus)
\end{aligned}
$$

where $\mathbf{0} = \big((0\ 0)^T, 0\big)$. Furthermore, the pose $\mathbf{0} \ominus x_a$ represents the pose of $\mathcal{F}$ with respect to $\mathcal{F}_a$, see Fig. 4.3.

The operators $\oplus$ and $\ominus$ are also used to compose two-dimensional position vectors with three-dimensional poses. In particular, given the coordinates p of a point expressed in $\mathcal{F}_i$, whose pose w.r.t. $\mathcal{F}$ is x_i, the operation $x_i \oplus p$ gives the coordinates of the same point expressed in $\mathcal{F}$. Conversely, given x_i and the coordinates p of a point expressed in $\mathcal{F}$, the operation $p \ominus x_i$ gives the coordinates of the same point expressed in $\mathcal{F}_i$, whose pose w.r.t. $\mathcal{F}$ is x_i. These operators may also be used with a set P of points, by letting $x_i \oplus P := \{x_i \oplus p \mid p \in P\}$, and $P \ominus x_i := \{p \ominus x_i \mid p \in P\}$.

In case of complete measuring graph, an *observation* P_i is a set of n distinct points in $\mathbb{R}^2$, one of which is always the origin. It represents the positions of the robots as measured by the i-th robot, i.e., relative to $\mathcal{F}_i$. Apart from the origin, which stands for $\mathcal{R}_i$ itself, P_i does not convey any information about the identity of the robot located at a certain point (*anonymity*), nor about its orientation.

In a more general case, the observation model changes depending on the characteristics of the considered sensory system. Altogether, we consider the following four observation models (in order of increasing generality):

1. *Complete measuring graph.* Any observation is made by the exact n measures of all the n robots of the team. Note that, in this case, all the observations of a given group are the same up to roto-translations. See Fig. 4.1b–e for examples of observations.

2. *Generic measuring graph.* Any observation is made by a subset of the exact measures of all the robots of the team. The cardinality of these subset is $m_i \leq n$. As a special case we consider the situation in which if a robot measures another robot then the robot is also measured by the other.

3. *False positives.* Any observation is made by a subset of the exact measures of all the robots of the team plus a set of exogenous positions coming from stochastic detecting errors (false positives). The observation does not convey any information to distinguish a correct measurement from a false positive.

4. *Noisy measurement.* Any observation is a probability density function on $\mathbb{R}^2$, which depends on (1) the pose of the measuring robot $x_i = (p_i^T\ \theta_i)^T$, (2) the positions of the other robots $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$, and (3) a stochastic set of exogenous positions Q_i coming from detecting errors (false positives):

$$f_i({}^i p | x_i, p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n, Q_i)$$

   This p.d.f. indicates the probability that there is a robot in a certain position ${}^i p$, expressed in the frame $\mathcal{F}_i$.

Correspondently, there are four possible Mutual Localization with Anonymous Position Measures problems. With respect to the observation model 1 we have the following problem.

Problem 4.1 (Mutual Localization with Anonymous Position Measures – Complete measuring graph)**.** *Given n observations $P_1, \ldots, P_n$, find all the possible pairs of functions*

$$\hat{p} : \{2, \ldots, n\} \to P_1 \backslash (0\ 0)^T$$
$$\hat{\theta} : \{2, \ldots, n\} \to [0, 2\pi)$$

with $\hat{p}$ bijective, such that

$$P_1 \ominus \hat{x}_i = P_i \qquad i = 2, \ldots, n \tag{4.1}$$

where $\hat{x}_i := (\hat{p}(i), \hat{\theta}(i))$.

Function $\hat{p}$ assigns each point of P_1 (with the exception of the origin) to one and only one robot in $\{\mathcal{R}_2, \ldots, \mathcal{R}_n\}$, whose orientation is then defined by $\hat{\theta}$ (see Fig. 4.1b). Note that $\mathcal{R}_1$ is directly associated to the origin, with orientation equal to zero, in all solutions to the problem. Stated differently, Problem 4.1 consists in finding all the formations $\{\hat{x}_1 = ((0\ \ 0)^T, 0), \hat{x}_2 \ldots, \hat{x}_n\}$ that are compatible with the given observations, i.e., satisfy (4.1). In general, a solution to Problem 4.1 may exist or not. In the following, we assume that each observation P_i, $i = 1, \ldots, n$, has been gathered by robot $\mathcal{R}_i$ with reference to the same spatial arrangement of the group. This is sufficient to claim that Problem 4.1 admits at least one solution.

With respect to the observation model 2 we have the following problem.

Problem 4.2 (Mutual Localization with Anonymous Position Measures – Generic measuring graph)**.** *Given n observations $P_1, \ldots, P_n$, with $|P_i| = m_i \leq n$, find all the sets $\hat{P}_1 \supset P_1$ such that $|\hat{P}_1| = n$ and all the possible pairs of functions*

$$\hat{p} : \{2, \ldots, n\} \to \hat{P}_1 \backslash (0\ 0)^T$$
$$\hat{\theta} : \{2, \ldots, n\} \to [0, 2\pi)$$

with $\hat{p}$ bijective, such that

$$\hat{P}_1 \ominus \hat{x}_i \supset P_i \qquad i = 2, \ldots, n \tag{4.2}$$

where $\hat{x}_i := (\hat{p}(i), \hat{\theta}(i))$.

With respect to the observation model 3 we have the following problem.

Problem 4.3 (Mutual Localization with Anonymous Position Measures – False positives)**.** *Given n observations $P_1, \ldots, P_n$, find all the sets $\hat{P}_1 \supset P_1$ such that $|P_1| \geq n$ all the possible pairs of functions*

$$\hat{p} : \{2, \ldots, n\} \to \hat{P}_1 \backslash (0\ 0)^T$$
$$\hat{\theta} : \{2, \ldots, n\} \to [0, 2\pi)$$

with $\hat{p}$ injective, such that

$$\hat{P}_1 \ominus \hat{x}_i \supset P_i \qquad i = 2, \ldots, n \tag{4.3}$$

where $\hat{x}_i := (\hat{p}(i), \hat{\theta}(i))$.

With respect to the observation model 4 we have the following problem.

Problem 4.4 (Mutual Localization with Anonymous Position Measures – Noysy measurement)**.** *The correct statement of this problem is still an open issue.*

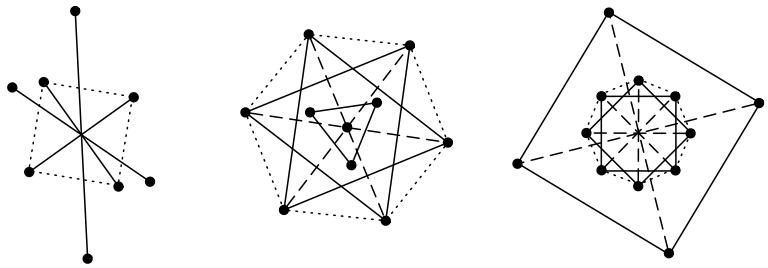

Figure 4.4: Three rotational symmetric sets of points. From left to right, the associated proper symmetric groups are respectively C_2, C_3 and C_4. Note that only the second set contains its centroid. Solid line segments join points that belonging to the same set of the rotational symmetric partition. Dotted line segments show the presence of partial higher-degree symmetries which are not relevant for the analysis: from left to right, they identify respectively a square, an hexagon and an octagon. Dashed line segments meet at the centroid of each set.

4.3 Unique solvability and structure of solutions in case of complete measuring graph

In this Section we give a necessary and sufficient condition for the unique solvability (i.e., the uniqueness of the solution) of Problem 4.1 (Proposition 4.1), an associated test (Proposition 4.2), and a quantitative and qualitative characterization of the solutions (Propositions 4.3 and 4.4). In particular, we show that the problem is uniquely solvable if and only if the set of points represented by observation P_1 does not have a rotational symmetry (remember that all observations are the same up to roto-translations). Furthermore, we show that in the case of non-unique solvability the number of solutions has a factorial trend with respect to n, the number of robots. To establish these results, we first recall a few basic concepts on rotational symmetry.

4.3.1 A brush-up on rotational symmetry

Consider a set of n points $P \subset \mathbb{R}^2$. Let $\mathcal{S}_P$ denote the *proper symmetry* group of P, i.e., the subgroup of its orientation-preserving isometries (roto-translations) under which it is invariant. It is known from symmetry group theory [66] that, since P is a bounded set, $\mathcal{S}_P$ can be represented as a subgroup of $SO(2)$ (the group of planar rotations), by choosing the origin

to be its fixed point, i.e., the centroid[2] of P. In particular, there exists a positive integer l such that $\mathcal{S}_P = C_l$, where C_l is the *cyclic group of order* l, whose generator is the rotation of $2\pi/l$. P is said to be *rotational symmetric* if $\mathcal{S}_P \neq C_1$, where C_1 is the trivial group containing only the identity operation.

Assume that $\mathcal{S}_P = C_l$ and let c be the centroid of P. Denote by $q_\phi = (c - R(\phi)c, \phi)$ the rotation by an angle ϕ around c, and in particular by

$$q_k := (c - R(2k\pi/l)c, 2k\pi/l), \tag{4.4}$$

the rotation by $2k\pi/l$, for $k = 0, 1, \ldots, l-1$. We have then

$$P = P \ominus q_k = q_k \oplus P,$$

for $k = 0, 1, \ldots, l-1$. Note that rotational symmetry is invariant under isometries: if P is rotational symmetric, also $P \ominus x$ is rotational symmetric, for any $x \in SE(2)$. Some examples of rotational symmetric sets of points are shown in Fig. 4.4.

The following Lemma establishes an important property which is valid for any finite set of points and has an important role in the study of the unique solvability of Problem 4.1.

Lemma 4.1 (Rotational Symmetric Partition)**.** *Given a set of n points P such that $\mathcal{S}_P = C_l$, there exists a partition $\mathcal{E}_P = \{E_1, \ldots, E_m\}$ of P such that E_j, with $j = 1, \ldots, m$, is invariant under any rotation in C_l around the centroid c, i.e.,*

$$E_j = E_j \ominus q_k\,, \quad k = 0, 1 \ldots, l-1.$$

If $c \notin P$, then l divides n, $m = n/l$, and the cardinality of each subset of the partition $\mathcal{E}_P$ is l. If $c \in P$, then l divides $n-1$, $m = 1 + (n-1)/l$ and the cardinality of each subset in $\mathcal{E}_P \backslash \{c\}$ is l.

Proof. Suppose w.l.o.g. that c is the origin. Chosen a point $p \in P\backslash\{c\}$, the set $E(p)$ of all points obtained applying an element of C_l to p is a subset of P by definition. Clearly, $E(p)$ has cardinality l and is invariant under C_l. Now choose a point p' in $P\backslash E(p)$, repeat the above construction to obtain $E(p')$, and proceed as before. If $c \notin P$, the collection of all the distinct sets $E(p)$ for all $p \in P$ gives the subsets $E_1, \ldots, E_m$ of the partition $\mathcal{E}_P$, with $m = n/l$. On the other hand, if $c \in P$ then set $E(c)$ is a singleton and must be added to the previous collection, which consists in this case of $(n-1)/l$ subsets. □

[2]In fact, since after any rotation in $\mathcal{S}_P$ the set of points P remains the same, also the centroid remains the same, hence the centroid is the fixed point.

n	possible values of l																		
1	1																		
2	1	2																	
3	1	2	3																
4	1	2	3	4															
5	1	2		4	5														
6	1	2	3		5	6													
7	1	2	3			6	7												
8	1	2		4			7	8											
9	1	2	3	4				8	9										
10	1	2	3		5				9	10									
11	1	2	3		5					10	11								
12	1	2	3			6					11	12							
13	1	2	3			6						12	13						
14	1	2					7						13	14					
15	1	2	3		5		7							14	15				
16	1	2	3	4	5			8							15	16			
17	1	2	3	4				8								16	17		
18	1	2	3						9								17	18	
19	1	2	3						9									18	19
$\vdots$	$\vdots$																		$\ddots$

Figure 4.5: The possible values of the integer l for the cyclic groups C_l that can be the proper symmetry groups of a set P of n points. Note that, since P can always be non-rotational symmetric, $l = 1$ is always present. Also, $l = 2$ is always possible since for any odd value of n one point can be always placed in the centroid.

Figure 4.4 shows the partitions for three different rotational symmetric set of points, while in Fig. 4.5 the possible values of l are tabulated for sets of $n = 1, \ldots, 10$ points. Limit cases are found when $l = 1$ (the set of points is not rotational symmetric, and the partition consists of n singletons) and when $l = n$ (the set of points may be a regular n-gon, and the partition consists if a single set containing all the points in P).

4.3.2 Unique solvability of the problem 4.1

In the rest of this Section, it is assumed that $\mathcal{S}_{P_1} = C_l$ and that c denotes the centroid of P_1. The role of rotational symmetry in the unique solvability of Problem 4.1 is clarified by the following result.

Proposition 4.1 (Unique Solvability)**.** *Assume that Problem 4.1 admits a solution. The solution is unique if and only if P_1 is not rotational symmetric.*

Proof. Assume that Problem 4.1 admits at least two solutions. Then there exists i and two poses $\hat{x}'_i$ and $\hat{x}''_i \neq \hat{x}'_i$ such that $P_1 \ominus \hat{x}'_i = P_i$ and $P_1 \ominus \hat{x}''_i = P_i$. Then $P_1 = \hat{x}''_i \oplus (P_1 \ominus \hat{x}'_i) = [x''_i \oplus (\mathbf{0} \ominus \hat{x}'_i)] \oplus P_1$, i.e., there exists a non-zero roto-translation which transforms P_1 in itself; this means that P_1 is rotational symmetric. On the other hand, assume that P_1 is rotational symmetric. A solution $\{x_1, \ldots, x_n\}$ exists, i.e., $P_1 \ominus x_i = P_i$, $i = 1, \ldots, n$, and there exists a non-zero roto-translation x which transforms P_1 in itself, i.e., $P_1 = P_1 \ominus x$. This means that $\{x \ominus x_1, \ldots, x \ominus x_n\}$ is also a solution. □

Remark Proposition 4.1 implies that the number of solutions to Problem 4.1 is invariant with respect to changes in the orientations of the robots in the formation (in spite of the fact that the observations change).

Unique solvability may be tested with the aid of the following result.

Proposition 4.2 (Unique Solvability Test)**.** *Denote with $P_1(\phi)$ the set of points obtained by rotating the observation P_1 by an angle ϕ around its centroid c, i.e.:*

$$P_1(\phi) := \{R(\phi)(p - c) + c \mid p \in P_1\}. \tag{4.5}$$

If $c \notin P$, Problem 4.1 has a unique solution if and only if

$$P_1 \neq P_1(2\pi/m), \quad \forall m \text{ prime factor of } n. \tag{4.6}$$

If $c \in P$, in (4.6) n must be replaced by $n - 1$.

Proof. Since P_1 has n points, its proper-symmetry group $\mathcal{S}_{P_1}$ can only be one of the cyclic groups $C_1, \ldots, C_n$. In addition, since C_l, with $2 \leq l \leq n$, also belongs to any C_m with m prime factor of l, and l can only be a divisor of n (if $c \notin P_1$) or $n - 1$ (if $c \in P_1$), it is sufficient to check the rotations that are generators of the cyclic groups C_m, with m prime factor of n or $n - 1$. □

Assume $c \notin P_1$. Since (4.6) requires n checks for any value of m, the complexity of the test is $O(n \cdot \pi(n))$, where the *prime-counting function* $\pi(n)$ can be approximated by $n/\log(n)$. If $c \in P_1$, the complexity is $O((n-1) \cdot \pi(n-1))$.

4.3.3 Structure and number of multiple solutions

We now turn our attention to the case when there are multiple solutions to Problem 4.1.

Proposition 4.3 (Structure of the Solutions)**.** *Let $i = 2, \ldots, n$. If $\hat{x}_i$ is a feasible pose for $\mathcal{R}_i$, in the sense that $\hat{x}_i = (\hat{p}_i, \hat{\theta}_i)$ satisfies (4.1), then all the non-zero poses obtained as $q_k \oplus \hat{x}_i$, with $k = 0, 1, \ldots, l-1$ and q_k defined by (4.4), are feasible for $\mathcal{R}_i$, and vice versa.*

Proof. Being $P_1 \ominus \hat{x}_i = P_i$ and $P_1 = P_1 \ominus q_k$, we have $(q_k \oplus P_1) \ominus \hat{x}_i = P_i$. Developing the pose compositions for an element p of P_1 we have that

$$\begin{aligned}(q_k \oplus p) \ominus \hat{x}_i &= (c_k + R(\phi_k)p) \ominus \hat{x}_i \\ &= R(-\hat{\theta}_i)(c_k + R(\phi_k)p - \hat{p}_i) \\ &= R(-\hat{\theta}_i)R(\phi_k)(p - R(-\phi_k)(\hat{p}_i - c_k)) \\ &= p \ominus (\hat{x}_i \ominus q_k) \end{aligned} \tag{4.7}$$

Hence $(q_k \oplus P_1) \ominus \hat{x}_i = P_1 \ominus (\hat{x}_i \ominus q_k)$ and $\hat{x}_i \ominus q_k$ is a feasible solution, for any $k = 0, 1, \ldots, l-1$, which is equivalent to say that $q_k \oplus \hat{x}_i$ is a feasible solution, for any $k = 0, 1, \ldots, l-1$. Similarly, it is simple to show that for any other feasible pose $x' \oplus \hat{x}_i$, x' must belong to $\{q_k\}_{k=0,1,\ldots,l-1}$. □

Proposition 4.3 essentially states that if the observations of Problem 4.1 are generated by a formation $\{x_1, \ldots, x_n\}$, then $\mathcal{R}_i$ can be assigned to position p_i as well as to all the other positions of the subset of $\mathcal{E}_{P_1}$ which contains p_i. This leads to the following results.

Proposition 4.4 (Number of Solutions)**.** *The number of solutions to Problem 4.1 is*

$$(l-1)! \cdot (l!)^{\frac{n}{l}-1} \qquad \text{if } c \notin P_1 \tag{4.8}$$

$$(l!)^{\frac{n-1}{l}} \qquad \text{if } c \in P_1 \tag{4.9}$$

Proof. First remember that in all solutions $\mathcal{R}_1$ is placed in $(0\ 0)^T$. If $c \notin P_1$ then $\mathcal{E}_{P_1}$ has n/l sets each composed by l positions. Each set of $\mathcal{E}_{P_1}$ has l robots associated, and, in each solution, each of these robots (except for $\mathcal{R}_1$) can be placed in any position of the set, provided that this position is not occupied by another robot. Hence, $(l-1)!$ possible permutations correspond to the set of $\mathcal{E}_{P_1}$ associated to $\mathcal{R}_1$, and $l!$ possible permutations correspond to the remaining $n/l-1$ sets of $\mathcal{E}_{P_1}$. Multiplying these possibilities we obtain (4.8). If $c \in P_1$, noticing that robot $\mathcal{R}_i$ associated to the set $\{c\}$ of $\mathcal{E}_{P_1}$ has l possible poses if $i \neq 1$, a similar analysis leads to (4.9). □

Corollary 4.1. *For a given n, the maximum number of possible solutions to Problem 4.1 is $(n-1)!$. This number is actually reached when P_1 is a regular n-gon if $c \notin P_1$, and when $P_1 \backslash c$ is a regular $(n-1)$-gon if $c \in P_1$.*

Proof. If $c \in P_1$ and $l = n-1$ then $(l!)^{\frac{n-1}{l}} = (n-1)!$ and $P_1 \backslash c$ is a regular $(n-1)$-gon. If $l < n-1$, then l is a factor of $n-1$ and $m = (n-1)/l \in \mathbb{N}$. In the fraction

$$r = \frac{(l!)^m}{(n-1)!} = \frac{(l!)^{m-1}}{(n-1)(n-2)\dots(l+1)}$$

both numerator and denominator are products of $l(m-1)$ factors and the smallest factor of the denominator is larger than the largest factor of the numerator. Then $r < 1$ holds, and we can write $(l!)^{\frac{n-1}{l}} < (n-1)!$. For $c \notin P_1$, a similar demonstration leads to $(l-1!)(l!)^{\frac{n}{l}-1} < (n-1)!$ if $l < n$, while if $l = n$ the number of solutions is $(n-1)!$ and P_1 is a regular n-gon. □

Summarizing, each point of the observation P_1 can be assigned to one and only one subset of partition $\mathcal{E}_{P_1}$. Assuming, for a moment, that $c \notin P_1$, in view of Lemma 4.1, each subset of $\mathcal{E}_{P_1}$ has l positions and l robots assigned to it. Conversely, each robot can assume l different poses which correspond to all the l positions in its subset, with l different orientations. Notice that the l orientations differ by a multiple of $2\pi/l$. The robots associated to the set to which is associated also the fixed robot $\mathcal{R}_1$, have only $l-1$ possible poses instead of l. Note that all the robots associated to the same set have the observations equal up to a (pure) rotation. If $c \in P_1$ then the robot $\mathcal{R}_i$ associated to $\{c\}$, provided that $i \neq 1$, has l different possible poses with the same position.

All the solutions can be generated by independently permuting the possible poses of each robot, with the constraint that two robots can not occupy the same position. This means that the set of solutions of Problem 4.1 is implicitly represented by the knowledge of: (1) the set P_1, (2) the partition $\mathcal{E}_{P_1}$ of P_1, (3) the association between each robot $\mathcal{R}_i$, $i = 1, \dots, n$, and the corresponding set of $\mathcal{E}_{P_1}$. Using this representation of the solutions of Problem 4.1, is useful to improve the complexity of MultiReg, an algorithm described in [26], which explicitly computes all the solutions of Problem 4.1. In fact, the partition and the association can be computed in a polynomial time using P_1 and the other observations.

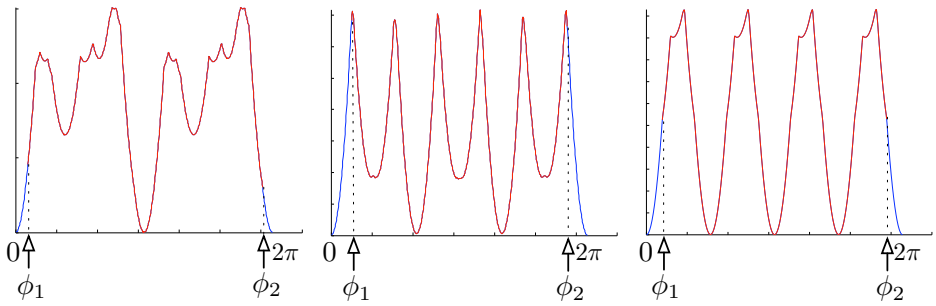

Figure 4.6: The symmetry metric function γ for the three set of points of Fig. 4.4, in the same order from left to right.

4.4 An anti-symmetry control law

Assume that a team of robots must perform a collaborative task which requires mutual localization, and that only anonymous position measures are available. If the robots are initially arranged in a formation resulting in observations that are rotational symmetric, mutual localization will be computationally heavier and will not provide a single solution. In the stochastic case, as mentioned in the introduction, problems will arise whenever the observations are *close* to being rotational symmetric. For this reason, we introduce in this Section a continuous function that measures the distance of sets of points from rotational symmetry. This will be used to design a control law aimed at keeping the solution to Problem 4.1 unique. We mention that the symmetry distance function proposed in [67] is not practical for our purposes because its computation cannot be executed in real time.

Given the set of points P_1 and an angle $\phi \in [0, 2\pi)$, define the *symmetry metric function*

$$\gamma_P(\phi) := e(P_1, P_1(\phi)).$$

where $P_1(\phi)$ is defined in (4.5) and

$$e(P', P'') := \sum_{p' \in P'} \min_{p'' \in P''} \|p' - p''\|^2.$$

is the *closest point metric* between P' and P''.

Proposition 4.5 (Properties of γ_P)**.** *The following statements are true:*

1. $\gamma_{P_1}(0) = 0$.

2. *γ_{P_1} is zero only at $\{2k\pi/l, k = 0, \ldots, l-1\}$, where l is the integer such that $\mathcal{S}_P = C_l$.*

3. *P_1 is rotational symmetric if and only if γ_{P_1} is zero for some ϕ other than 0.*

4. *there exist ϕ_1, ϕ_2, with $0 < \phi_1 < \phi_2 < 2\pi$, such that γ_{P_1} is strictly increasing in $[0, \phi_1)$ and strictly decreasing in $(\phi_1, 2\pi)$.*

Proof. 1) is true by definition. Moreover, $\gamma(\phi) = 0$ if and only if for any $p' \in P_1$ exists $p'' \in P_1(\phi)$ s.t. $p' = p''$. Hence, $P_1 = P_1(\phi)$, i.e., the rotation $R(\phi)$ belongs to C_l. This implies 2). In addition, 2) implies 3). Finally, consider the function $\hat{\gamma}_P(\phi) = \sum_{p \in P_1} \|(p-c) - R(\phi)(p-c)\|^2$, which equal to $\sum_{p \in P_1} (2(p-c)\sin(\phi/2))^2$, that is monotonically increasing in $[0, \pi]$ and monotonically decreasing in $[\pi, 2\pi]$. For each $p \in P_1$ there is a neighborhood of $\phi = 0$ in which $\min_{p' \in P_1(\phi)} \|p - p'\|^2 = \|(p-c) - R(\phi)(p-c)\|^2$, i.e., in which $\gamma_P(\phi) = \hat{\gamma}_P(\phi)$. We let $\Phi \subset [0, 2\pi)$ denote the set in which $\gamma_P(\phi) = \hat{\gamma}_P(\phi)$. Then, 4) is proven taking $\phi_1 = \max_{\Phi \cap [0,\pi]} \phi$ and $\phi_2 = \min_{\Phi \cap [\pi, 2\pi]} \phi$. □

As stated in the proof we define $\phi_1 = \max_{\Phi \cap [0,\pi]} \phi$ and $\phi_2 = \min_{\Phi \cap [\pi, 2\pi]} \phi$. According to Prop. 4.5, the minimum value of function γ_{P_1} in the interval $[\phi_1, \phi_2]$ (also called *internal* minimum value in the following) is a continuous measure of the distance of P_1 from being rotational symmetric. If the minimum is actually zero, P_1 is actually symmetric. Therefore, a control action aimed at keeping Problem 4.1 uniquely solvable can be based on the strategy of increasing such minimum value.

In particular, assume for simplicity that the position of each robot obeys an omnidirectional kinematic model:

$$\dot{p}_i = u_i, \qquad i = 1, \ldots, n,$$

where u_i is the two-dimensional vector of velocity inputs for $\mathcal{R}_i$. Consider the following *anti-symmetry* control law

$$u_i = \alpha \frac{\bar{p}_i - p_i}{\|\bar{p}_i - p_i\|} \qquad i = 1, \ldots, n, \tag{4.10}$$

where α is a positive gain and

$$\bar{p}_i := \operatorname*{argmin}_{p \in P_1(\bar{\phi})} \|p_i - p\|$$
$$\bar{\phi} := \operatorname*{argmin}_{\phi \in [\phi_1, \phi_2]} \gamma_{P_1}(\phi).$$

The above control law has a simple interpretation. Once the rotation angle $\bar{\phi}$ that minimizes γ_{P_1} in $[\phi_1, \phi_2]$ has been identified (e.g., numerically), $P_1(\bar{\phi})$ is built by rotating P_1 by $\bar{\phi}$. The closest point $\bar{p}_i \in P_1(\bar{\phi})$ is found for any $p_i \in P_1$, and the velocity input is chosen so as push $\mathcal{R}_i$ away from $\bar{p}_i$ along the segment $p_i\bar{p}_i$. This will clearly lead to an increase of $\gamma_{P_1}(\bar{\phi})$.

Notice that (4.10) is undefined if P_1 is rotational symmetric. In this case a simple randomized control for the small time sufficient to break the symmetry is used.

4.5 Simulations

We have validated the results of Sections 4.3 and 4.4 through extensive simulations of the anti-symmetry control law.

The results of the first simulation are shown in Fig. 4.7, above. The 9-robot system starts in a lattice formation whose proper symmetry group is C_4, and moves under the action of the anti-symmetry control. Symmetry is readily broken, as shown by change in symmetry metric function γ_{P_1}, which has 3 internal zeros at start. As the simulation proceeds, the internal minimum values of γ_{P_1} increase.

Figure 4.7, above, also shows the consequence of measurement noise on the accuracy of the estimated solution in the neighborhood of the initial rotational symmetric formation. To compute the solutions of Problem 4.1, we have used MultiReg, a probabilistic robust estimation algorithm that performs a multiple registration among a set of noisy observations (see Section 4.6). At each step, we have obtained multiple sets of noisy observations by adding a gaussian noise to the observations of the current arrangement. The figure shows all the possible poses of the circled robot as estimated by MultiReg on the basis of these data.

It can be observed that at the start, when the formation is rotational symmetric, the estimated solutions are evenly distributed in 4 clusters of poses. The clusters are centered on all the feasible positions of a single subset of the partition $\mathcal{E}_{P_1}$, as predicted by Proposition 4.3. The number of solutions (576) found by MultiReg matches with the one theoretically derived in Proposition 4.4. When the symmetry is completely broken, as for $t = 4.0$ s , the estimates have a gaussian distribution centered on the real pose and a covariance comparable to that of the additive noise.

In the intermediate frames, in which the formation is close to being rotational symmetric, the solutions of MultiReg are distributed in more than one cluster, but not evenly. The largest cluster is centered on the real pose of the estimated robot. The other clusters, with less solutions, become feasible

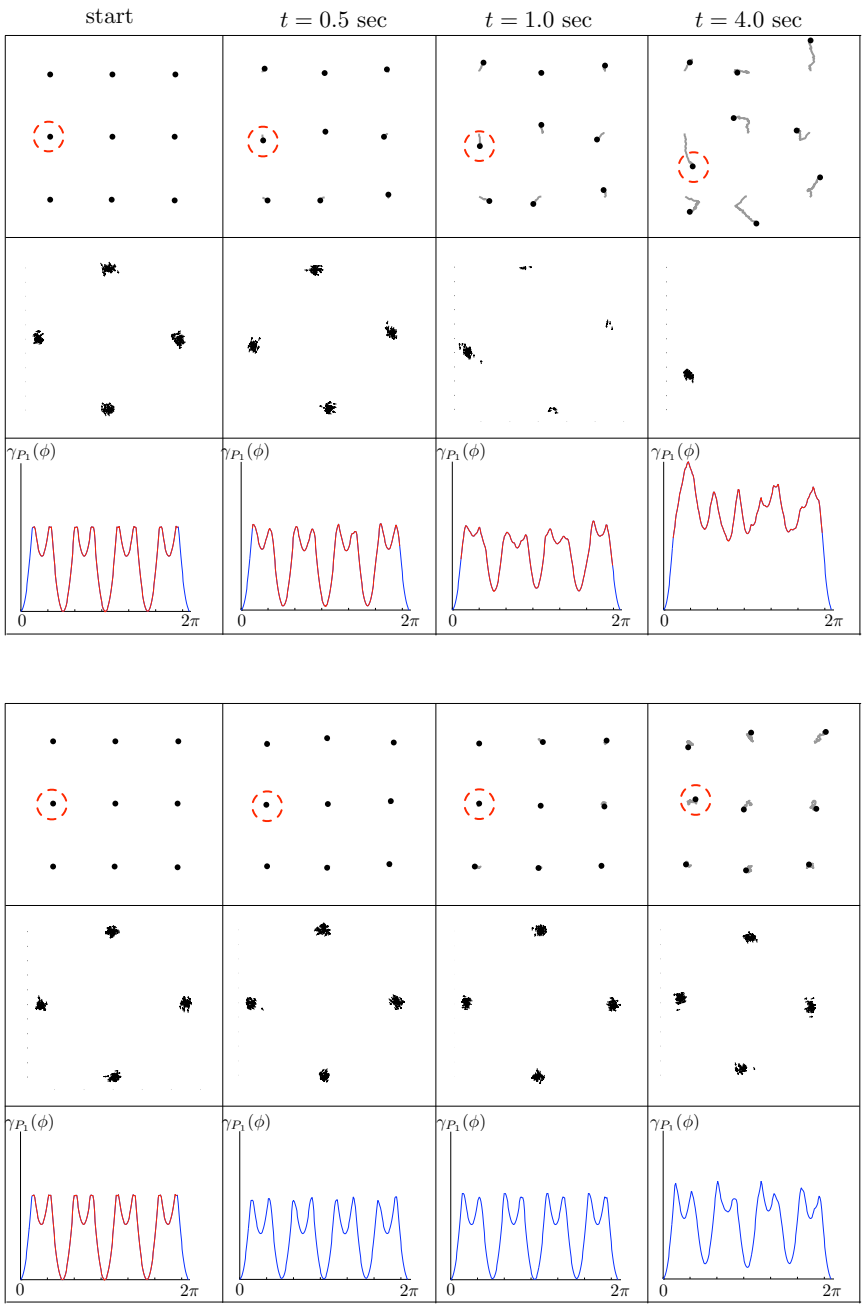

Figure 4.7: Above: the use of the anti-symmetry control law to break up a 9-robot lattice formation whose proper symmetry group is C_4; 4 snapshots of the formation (top), the estimated positions for the circled robot (center), the symmetry metric function γ_{P_1} (bottom). Below: the same results with a random control law.

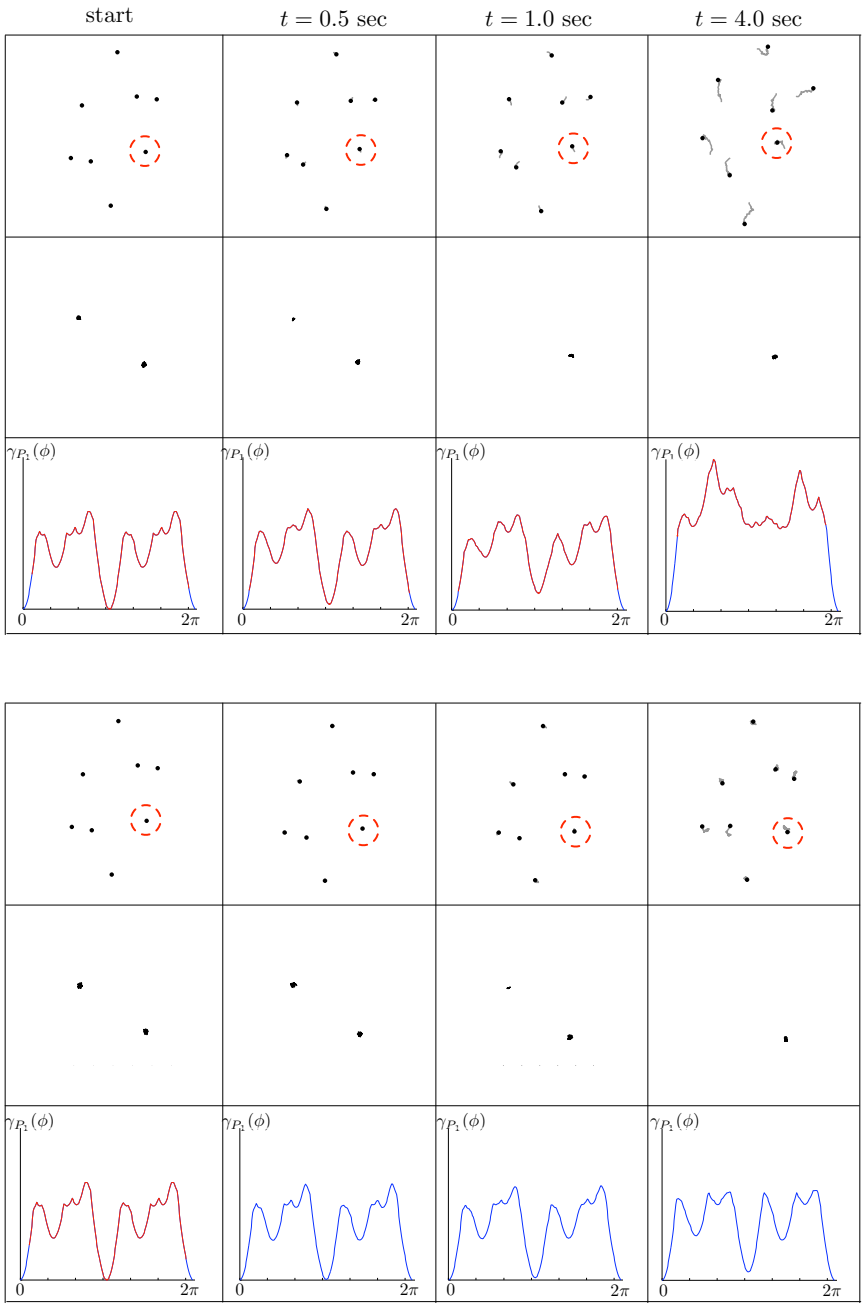

Figure 4.8: As in Fig. 4.7 for a formation with proper symmetry group C_2.

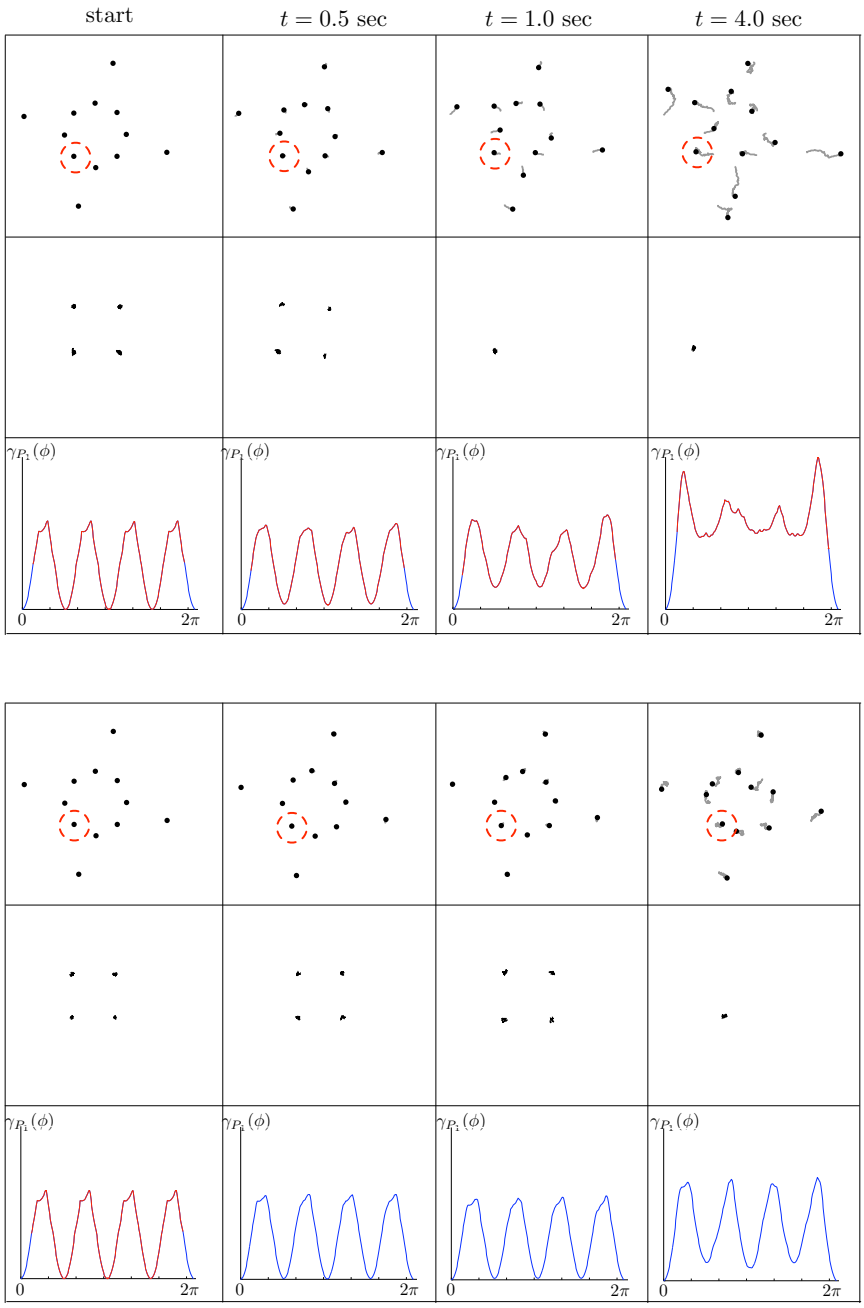

Figure 4.9: As in Fig. 4.7 for a formation with proper symmetry group C_4.

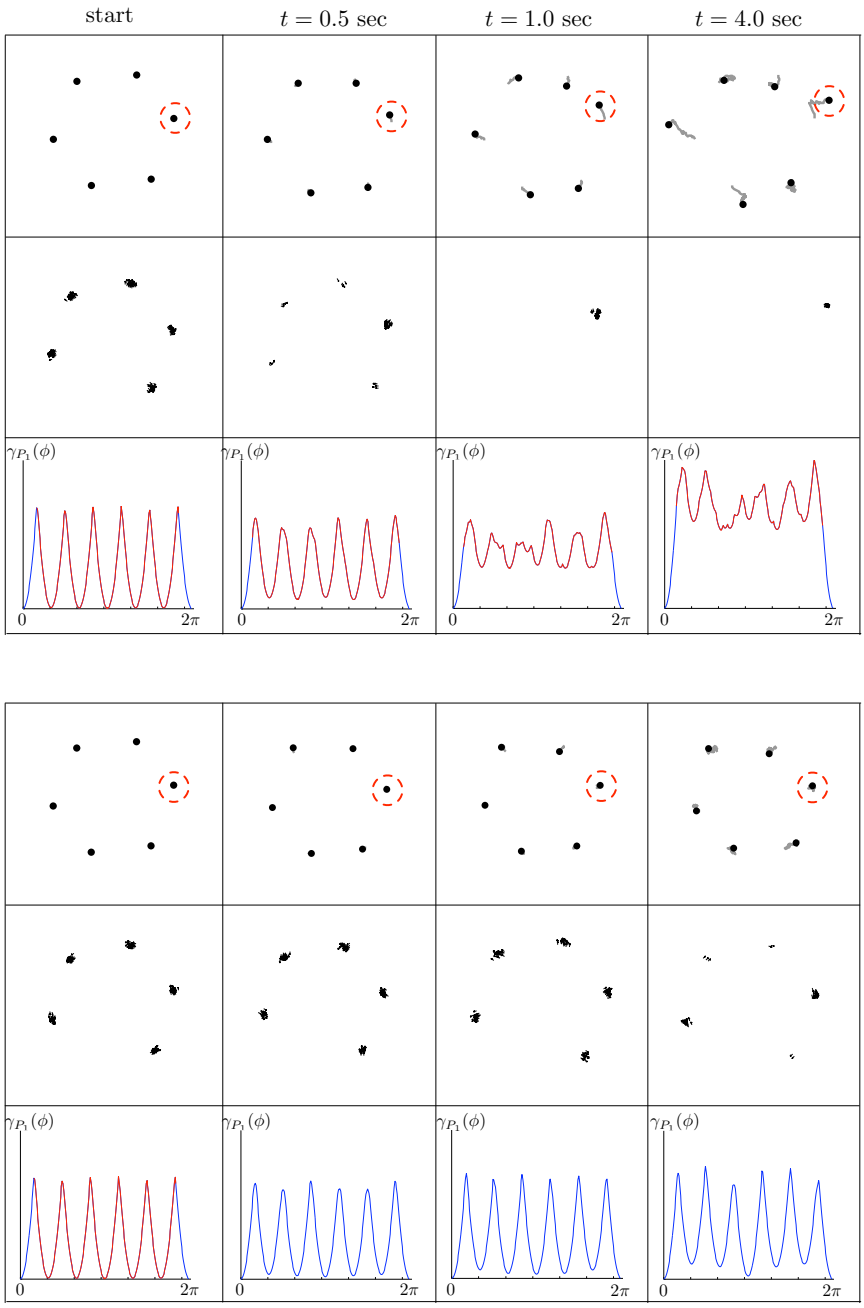

Figure 4.10: As in Fig. 4.7 for a formation with proper symmetry group C_6.

configurations only when the additive noise on the observations restores the rotational symmetry.

For comparison, we have also simulated the same 9-robot system under the action of a random control law (Figure 4.7, below). In fact, since the subset of symmetric configurations has zero measure in the configuration space, a random control law can also be expected to break the symmetry. However, the results show that the anti-symmetry control is much more effective in doing this than the random control. In fact, the increase of the internal minima of γ_{P_1} with the random control is slower and non-monotonous. Correspondingly, the multiple clusters of the estimation do not disappear.

See also `http://www.dis.uniroma1.it/labrob/research/mutLoc.html` for other simulations.

4.6 Multiple registration with MultiReg

In this Section we present an algorithm called *MultiReg* which solve the Problem 4.2.

A *multi-observation* is the main data structure used by MultiReg, it is denoted by O and is an extension of the observation concept defined in Section 4.2. It consists in a set of distinct points $F \in \mathbb{R}^2$, one of which is always the origin, and a *functional relation*[3] $A \subset F \times \{1, \ldots, n\}$ in which at least the origin is associated to an index. The relation A represents a partial labeling of the set of points. An observation is a special case of a multi-observation in which $A = \{((0\ 0)^T, i)\}$, where i is the index of the measuring robot. The points of F are called *features*.

In particular, given $f \in F$, we denote with $A(f) := \{i \in \{1, \ldots, n\} : (f, i) \in A\}$ the robot index (if any) associated to the feature f; given $i \in \{1, \ldots, n\}$, we denote with $A(i) := \{f \in F : \ (f, i) \in A\}$ the feature (if any) associated to the i-th robot (see Fig. 5.3a). Also, denote by $A(F) := \cup_{f \in F} A(f)$ the set of robot indexes appearing in O and by $A(\{1, \ldots, n\}) := \cup_{i \in \{1,\ldots,n\}} A(i)$ the set of features of O that are associated to some robot. A feature f is called *anonymous* when $A(f) = \emptyset$, i.e., $f \notin A(\{1, \ldots, n\})$. Two multi-observations $O_1 = (F_1, A_1)$ and $O_2 = (F_2, A_2)$ expressed in the same robot frame are said to be *irreconcilable* if:

a) two different features are associated to the same robot, i.e., $\exists f_1 \in A_1(\{1, \ldots, n\}), \exists f_2 \in A_2(\{1, \ldots, n\})$, with $f_1 \neq f_2$, such that $A_1(f_1) = A_2(f_2)$ (see Fig. 5.3b), or if

[3] This means that $|A(f)| \leq 1$, where $|\cdot|$ denotes the cardinality of a set.

b) two different robots are associated to the same feature, i.e., $\exists f \in A_1(\{1,\dots,n\}) \cap A_2(\{1,\dots,n\})$ such that $A_1(f) \neq A_2(f)$ (see Fig. 5.3c).

4.6.1 Binary registration

MultiReg uses binary registration as the basic tool. Given two observations $O_1 = (F_1, A_1)$ and $O_2 = (F_2, A_2)$ such that $A_1(F_1) \cap A_2(F_2) = \emptyset$ (always satisfied in MultiReg, see Sect. 4.6.2), consider a candidate change of coordinates T between the two associated frames. Letting $T(F) = \{q \in \mathbb{R}^2 | \exists f \in F : T(f) = q\}$, a binary relation $B \subset F_1 \times F_2$ is associated to T as follows: $(f_1, f_2) \in B \Leftrightarrow \|f_1 - T(f_2)\| \leq \delta$, where δ is a given *fitting threshold*. The elements of $B(F_1)$ and $B(F_2)$ are the *inliers* of F_2 and F_1, respectively. The cardinality of B, denoted by $|B|$, is the *number of inliers*.

Given $\delta > 0$ and $\mu > 0$, performing a *binary registration* of O_1, O_2 means finding a change of coordinates T such that the associated B is left- and right-unique, and satisfies: *i)* $|B| \geq \mu$ and *ii)* $|A(f_1) \cup A_2(f_2)| \in \{0, 1\}$ $\forall (f_1, f_2) \in B$. The first condition is a constraint on the *minimum number of inliers* (note that $|B| = |B(F_1)| = |B(F_2)|$). The second requires that, for any pair of features $(f_1,\ f_2)$ that are related by B, either f_1 or f_2 (or both) must be anonymous. In fact, being $A_1(F_1) \cap A_2(F_2) = \emptyset$, a 'double' assignment would certainly represent a conflict.

Once T has been determined, a new observation $O_{12} = (F_{12}, A_{12})$ is generated, where $F_{12} = F_1 \cup T(F_2)$ and $A_{12} \subset F_{12} \times \{1,\dots,n\}$ is such that for any $f \in F_{12}$ it is

$$A_{12}(f) = \begin{cases} A_1(f) & \text{if } f \in A_1(\{1,\dots,n\}) \\ A_2(f^*) & \text{if } f \in T(A_2(\{1,\dots,n\})) \\ A_2(B(f)) & \text{if } f \in B(F_2) \backslash A_1(\{1,\dots,n\}) \\ A_1(B(f^*)) & \text{if } f \in T(B(F_1)) \backslash T(A_2(\{1,\dots,n\})) \\ \emptyset & \text{otherwise} \end{cases}$$

where $f^* := T^{-1}(f)$ (see Fig. 4.11). For our purposes, the output of the binary registration (called *solution* in the following) is the triple $r(O_1, O_2) = (T, O_{12}, |B|)$. Clearly, for a given pair of observations there may exist multiple changes of coordinates that satisfy the above conditions, and therefore multiple solutions. We call two solutions *irreconcilable* if their corresponding observations are irreconcilable. In the following, we assume that the binary registration algorithm returns a finite set of irreconcilable solutions.

The combinatorial essence of the problem suggests the use of probabilistic techniques, while the presence of outliers (i.e., features observed by only one robot) calls for a robust estimation paradigm. We chose RANSAC [68]

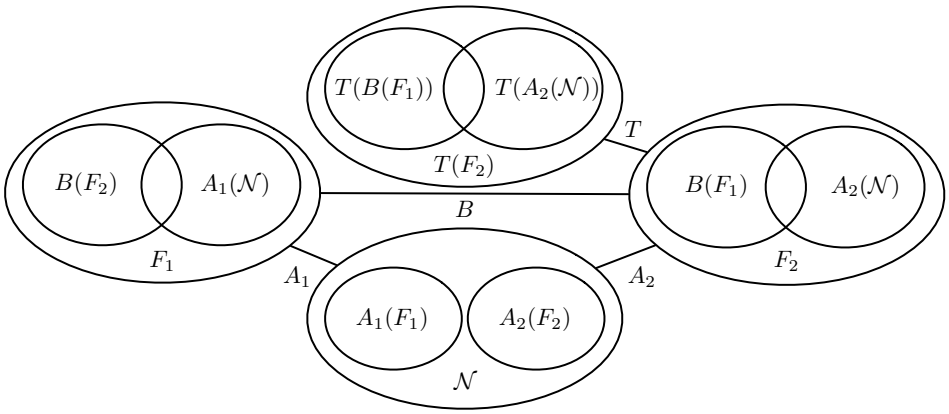

Figure 4.11: Sets of indexes/features involved in a binary registration with the associated relations.

Algorithm 8: RANSAC-based binary registration algorithm

input : $O_1 = (F_1, A_1), O_2 = (F_2, A_2)$ (2 observations)
parameters: I (max. number of iterations), δ (fitting threshold), μ (min. number of inliers)
variables : T (change of coordinates), O_{12} (observation), B (relation of inliers), D (equidistant pairs of features), $(f_1^a, f_1^b, f_2^a, f_2^b)$ (pairs of features)
output : $r(O_1, O_2) = \{\ldots, (O_{12j}, T_j, v_j), \ldots\}$ (set of solutions)

1 $r(O_1, O_2) = \emptyset$;
2 $D = \{(q, r, s, t) \in F_1 \times F_1 \times F_2 \times F_2 : \|q - s\| - \|r - t\| \leq 2\delta\}$;
3 **while** $h \leq I$ **and** $D \neq \emptyset$ **do**
4 extract randomly and without repetitions $(f_1^a, f_1^b, f_2^a, f_2^b)$ from D;
5 compute the change of coordinates T that aligns the segment $f_2^a f_2^b$ with the segment $f_1^a f_1^b$ and overlaps their middle points;
6 compute relation B from T and δ;
7 **if** $|B| \geq \mu$ **then**
8 compute observation O_{12} from B and T;
9 add $(T, O_{12}, |B|)$ to $r(O_1, O_2)$;
10 **return** $r(O_1, O_2)$

for binary registration because it has both this properties. Our implementation follows from the algorithm presented in [69] for a binary lidar scan registration and can be found in [70].

Algorithm 9: MultiReg algorithm for the i-th robot

input : $\Omega = \{\ldots, O_j, \ldots\}$, with $O_j = (F_j, A_j)$ ($|C_i| + 1$ raw observations)

variables: ${}^i\bar{t}_{jl} = (\ldots, {}^i\bar{t}_{jhl}, \ldots)$ (partial solution at the l-th iteration), $\tilde{O}_l$ (partial registered observation at the l-th iteration), $U_l \subseteq \{1, \ldots, n\}_\Omega$ (indexes of the unregistered observations at the l-th iteration)

output : $X(\Omega) = \{\ldots, {}^i\bar{t}_{js}, \ldots\}$ (set of solutions, in shared memory)

1 $\tilde{O}_1 = O_i$, $U_1 = \{1, \ldots, n\}_i$, $X(\Omega) = \emptyset$;

2 **for** $l \in \{1, \ldots, |C_i|\}$ **do**

3 $\quad \Gamma = \cup_{O \in \Omega_{U_l}} r(\tilde{O}_l, O)$;

4 $\quad \Gamma^* = \{\gamma = (T_\gamma, O_\gamma, |B_\gamma|) \in \Gamma \mid |B_\gamma| = \max_{|B|} \Gamma\} \subseteq \Gamma$ (this is the subset of Γ of elements that maximize the number of inliers);

5 $\quad$ Compute a maximal subset of irreconcilable solutions $\tilde{\Gamma} \subseteq \Gamma^*$;

6 $\quad$ Perform a least square estimation for every $\gamma \in \tilde{\Gamma}$, substituting T_γ with that minimizing the mean square error among the inliers and recomputing O_γ accordingly;

7 $\quad$ **for** $\gamma \in \tilde{\Gamma}$ **do**

8 $\quad\quad$ Fork the algorithm with $\tilde{O}_{l+1} = O_\gamma$, ${}^i\bar{t}_{jh} = ({}^i\bar{t}_{2(l-1)}, \ldots, {}^i\bar{t}_{|C_i|(l-1)})$, ${}^i\bar{t}_{hl} = t_\gamma$, $U_{l+1} = U_l \backslash \{s\}$ (where s is the index of the raw observation added in γ);

9 put the solution in the set of solutions $X(\Omega)$;

10 **return** $X(\Omega)$

4.6.2 Multiple registration

At each step k, $\mathcal{R}_i$ executes MultiReg on the set Ω made by its own raw observation $O_i = \{F_i, A_i\}$ and the raw observations $O_j = \{F_j, A_j\}$, for $j \in C_i[k]$. Let $\{1, \ldots, n\}_\Omega$ be the set of the indexes of the robots whose observations are in Ω. Since MultiReg is a memoryless algorithm, we drop k in the following. As stated before, it is $|A_j(F_j)| = 1$, $\forall j \in \{1, \ldots, n\}_\Omega$, and $\cap_{j \in \{1,\ldots,n\}_\Omega} A_j(F_j) = \emptyset$. We set $\Omega_j = \Omega \backslash O_j$ and, for any $\{1, \ldots, n\} \subseteq \{1, \ldots, n\}_\Omega$, we set $\Omega_{\{1,\ldots,n\}} = \Omega \backslash \{O_j : j \in \{1, \ldots, n\}\}$. The output is the set $X(\Omega) = \{{}^i\bar{t}_j, \}_{j \in \{1,\ldots,n\}_\Omega}$ where ${}^i\bar{t}_j = \{\ldots, {}^i\bar{t}_{jh}, \ldots\}_{j \in \{1,\ldots,n\}_\Omega}$, whose generic element ${}^i\bar{t}_{jh}$ is a estimate of ${}^it_j$.

A pseudocode description of MultiReg is given in Table 9. MultiReg executes $|C_i|$ iterations. Step 9 initializes the first iteration. At step 9, during the l-th iteration, $|C_i| - l$ binary registrations $r(\tilde{O}_l, O)$ are performed between

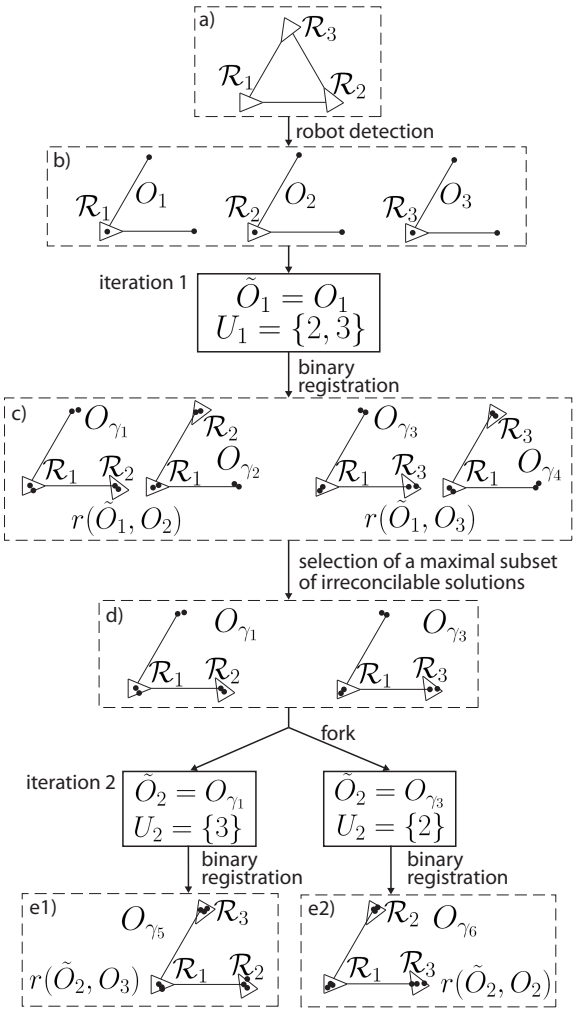

Figure 4.12: An example of MultiReg execution in a simple ambiguous situation: a) actual configuration b) raw observations of the robots c) results of the binary registrations between $\tilde{O}_1$ and O_2, O_3 d) selection of a maximal subset of irreconcilable solutions of Γ^* e1) result of the binary registration for the first branch e2) result of the binary registration for the second branch.

the observation so far obtained, $\tilde{O}_l$, and the raw observations $O \in \Omega_{U_l}$. Their solutions γ are stored in Γ. At step 9, the solutions with the maximum number of inliers are stored in Γ^*, and at step 9 $\tilde{\Gamma}$ is computed as a maximal subset of irreconcilable solutions of Γ^*. At step 9, the estimated change of coordinates T_γ in γ is tuned, for each $\gamma \in \tilde{\Gamma}$, by minimizing the mean square error of the inliers pairs using the algorithm in [71]. At step 9, MultiReg forks in $|\tilde{\Gamma}|$ branches, one for each $\gamma \in \tilde{\Gamma}$. If $\gamma = \{T_\gamma, O_\gamma, |B_\gamma|\} \in \tilde{\Gamma}$ is a solution given by the registration of $\tilde{O}_l$ with O_s, the new iteration of the branch starting from γ is initialized with $\tilde{O}_{l+1} = O_\gamma$ as partial registered observation and $U_{l+1} = U_l \backslash \{s\}$ as set of indexes of unregistered observations. A branch is terminated with no solution if the set Γ becomes empty. Otherwise, at the $|C_i|$-th iteration the branch contains a solution that is irreconcilable with the solutions of all the other branches. The algorithm returns as output the set of all solutions found in all branches.

An example of MultiReg execution is shown in Fig. 4.12 for a simple ambiguous configuration (Fig. 4.12a). The objective of the algorithm is the multiple registration of the raw observations O_1, O_2, O_3 (Fig. 4.12b). The MultiReg instance on $\mathcal{R}_1$ performs $|C_1| = 2$ iterations. At first iteration, the raw observation O_1 is chosen as partial registered observation $\tilde{O}_1$, and the indexes of the other observations are put in the set of the unregistered observations U_1. Then, two binary registrations are performed between $\tilde{O}_1$ and O_2, O_3 respectively (Fig. 4.12c). The results are put in $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$, and $\Gamma^* = \Gamma$ is selected as the subset of elements of Γ that maximize the number of inliers. Then, a maximal subset $\{\gamma_1, \gamma_3\} = \tilde{\Gamma} \subseteq \Gamma^*$ of irreconcilable solutions in Γ^* is computed (Fig. 4.12d). In the second iteration, for each $\gamma_i \in \tilde{\Gamma}$ $(i = 1, 3)$ the algorithm forks, initializing a new branch with $\tilde{O}_2 = O_{\gamma_i}$, and deleting the index of the registered robot from U_1: in particular in the first branch we set $U_2 = \{3\}$ and in the second $U_2 = \{2\}$. The first branch executes the binary registration between $\tilde{O}_2$ and O_3 (Fig. 4.12e1), finding the solution γ_5, and the second executes a binary registration between $\tilde{O}_2$ and O_2 (Fig. 4.12e2), finding the solution γ_6.

4.7 Open issues

In this chapter we have theoretically formulated and investigated a novel problem called Mutual Localization with Anonymous Position Measures. This is an extension of Mutual Localization with Position Measures, with the additional assumption that the identities of the measured robots are not known. Through the introduction of the concept of rotational symmetry in $\mathbb{R}^2$, we have demonstrated a necessary and sufficient condition

for the uniqueness of the solution, providing also a unique solvability test. Furthermore, we have studied the structure of multiple solutions, classifying the solutions in a number of equivalence classes which is linear with n. The identification of the relationship between the number of robots and the number of possible solutions allows to upper bound the maximum number of possible solutions to Problem 4.1 to $(n-1)!$.

Through the introduction of a continuous function that measures the distance of sets of points from rotational symmetry, we have designed a control law aimed at keeping the solution to Problem 4.1 unique. Its effectiveness is demonstrated through extensive simulations, comparing its performances with that a random control. The application of MultiReg corroborates the theoretical results about number and structure of the solutions and suggests some interesting probabilistic considerations.

Future works are the extension of a similar result to the whole problem addressed in [26], i.e., a non complete measuring graph with the presence of false positives and negatives, and a the design of a decentralized anti-symmetry control.

Chapter 5

A Method for the Absolute Mutual Localization Problem with Anonymous Relative Position Measures

This chapter, which directly follows Chapter 4 describes a method, partially developed in [26], which addresses the absolute mutual localization problem for a multi-robot system, under the assumption that each robot is equipped with a sensor that provides a measure of the relative position of nearby robots without their identity. Anonymity generates a combinatorial ambiguity in the inversion of the measure equations, leading to a multiplicity of admissible relative pose hypotheses. To solve the problem, we propose a two-stage localization system based on MultiReg, an the algorithm described in Section 4.6, that computes on-line all the possible relative pose hypotheses, whose output is processed by a data associator and a multiple EKF to isolate and refine the best estimates. The performance of the mutual localization system is analyzed through experiments, proving the effectiveness of the method and, in particular, its robustness with respect to false positives (objects that look like robots) and false negatives (robots that are not detected) of the measure process.

5.1 Introduction

This chapter deals with *mutual localization* (ML) in multi-robot systems. ML problems are of great importance in performing decentralized tasks that require data fusion, such as cooperative map-building and formation control.

Clearly, the accuracy of the localization can significantly affect the quality of the task execution.

In a multi-robot system, we refer to *relative mutual localization* (RML) as the problem of estimating the relative poses (position and orientation) among the moving frames attached to the robots. Assuming that each robot also has its own fixed frame, one can in addition define *absolute mutual localization* (AML) as the problem of estimating the relative poses among the various fixed frames. If each robot is self-localized with respect to its own fixed frame, the solution of RML can be obtained in principle from the solution of AML (and vice versa) by simple changes of coordinates.

To the best of our knowledge, no researchers have so far investigated the AML problem directly. Previous works have addressed either the RML problem or the problem of cooperative localization (CL) of a multi-robot system in a common[1] fixed frame. Roughly speaking, two different classes of approaches emerge: *filter-based* and *geometry-based*. The former use Extended Kalman (EK) or particle filters to estimate relative poses from measures, while the latter perform an instantaneous inversion of the mapping between relative poses and measures.

In most of the early filter-based approaches, such as [72, 73, 74, 75], the RML problem is solved by filtering out the noise from the output of a vision-based sensor that directly measures the relative poses between robots; at the same time, the filter provides a solution to the CL problem. In other works, the filter was also used to reconstruct the non-measured part of the change of coordinates; examples include [76], where relative range-only measures obtained by a combined RF/ultrasonic sensor are used; [77], where an extension of [72] is presented for different sensing equipments; and [78], where a detailed analysis is performed for range-only measurement.

As for geometry-based approaches, the problem of estimating the relative positions of robots in a formation by range-only measures or bearing-only measures has been investigated in [79, 80]. In the case of position (bearing plus range) relative measures, it is possible to obtain the relative pose of a robot respect to another by simply processing two bearing and one range measure [81].

A possible limitation of all the above methods is the assumption that the relative measures also provide the identity of the robots. In fact, interesting situations that may arise in practice are: *i)* the identities of the detected robots is not known (anonymous measures), *ii)* false positives (false detected robots) and *iii)* false negatives (undetected robots) occur in the relative

[1] The idea of a common fixed frame presumes a certain degree of centralization, because it is necessary that robots share some information at the beginning of the task.

position measurement process. The first and the second situation fit, for example, robot measurement systems based on a feature extraction module that looks for characteristics that are common to all robots and may also be found in other objects: for example, this happens when the robots and some obstacle in the environment have the same size, color, or shape, either by chance or by hostile camouflage. The third situation accounts for the fact that robots within the sensing range may not be detected, e.g., due to occlusions.

A pioneering work that addresses the anonymous RML problem is [82], in which an algorithm based on geometrical arguments is proposed to obtain relative pose estimates from anonymous bearing measurements. However, the method does not take into account false positives or false negatives, preventing its application to the aforementioned situations.

In this chapter, we address RML and AML problems with anonymous measures affected by false positives and negatives, as formalized in Sect. 5.2. The proposed two-stage localization system is described in Sect. 5.3. The first stage is a multiple registration algorithm that generates on-line all the geometrically admissible RML solutions (Sect. **??**). With the aid of self-localization data, these are used to solve the AML problem via data association and multiple EK filtering (Sect. 5.4). Experimental results are presented in Sect. 5.5.

5.2 Problem formulation

We take the following assumptions (refer to Figs. 5.1 and 5.2).

1. The multi-robot system includes n robots $\mathcal{R}_1, \ldots, \mathcal{R}_n$, where $n \geq 2$ is unknown. The robots move in $\mathbb{R}^2$. Let $\mathcal{N} = \{1, \ldots, n\}$ be the robot index set.

2. Each $\mathcal{R}_i$ ($i \in \mathcal{N}$) has two associated frames: a *fixed* frame $\mathcal{F}_i^\star$ and a *moving* frame $\mathcal{F}_i$ (see Fig. 5.1). The latter is rigidly attached to a representative point of the robot. Given $i, j \in \mathcal{N}$, denote by ${}^i t_j$ and ${}^i t_j^\star$ the 3-vectors describing the position and orientation (*pose*) respectively of $\mathcal{F}_j$ with respect to $\mathcal{F}_i$, and of $\mathcal{F}_j^\star$ with respect to $\mathcal{F}_i^\star$. Given ${}^i t_j$, it is immediate to build the change of coordinates ${}^i T_j$ from $\mathcal{F}_j$ to $\mathcal{F}_i$. The configuration of robot $\mathcal{R}_i$ is the pose of $\mathcal{F}_i$ with respect to $\mathcal{F}_i^\star$ and is indicated by $x_i^\star = (p_i^{\star T}\, \theta_i^\star)^T$ (see Fig. 5.1).

3. Each $\mathcal{R}_i$ comes with an independent *self-localization module* that provides an estimate $\hat{x}_i^\star$ of $x_i^\star$, i.e., the pose of the robot $\mathcal{R}_i$ in the frame $\mathcal{F}_i^\star$.

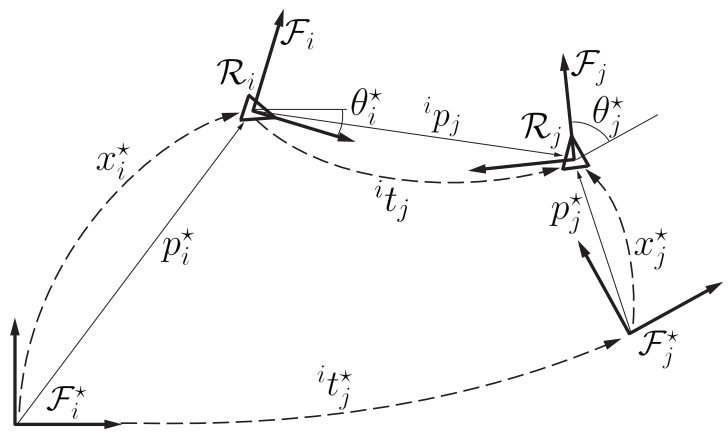

Figure 5.1: Geometric setting for mutual localization problems. Triangles are robots, solid arrows are position vectors and dashed arrows are pose vectors.

4. Each $\mathcal{R}_i$ is equipped with a *robot detector*, a sensor device that measures the *relative position* ${}^ip_j$ of other robots, provided that they fall in a perception set D_p that is rigidly attached to $\mathcal{F}_i$ (see Fig. 5.2). Note that no assumption is taken on the shape of D_p, in particular the robot detector does not need to be omnidirectional.

5. Each $\mathcal{R}_i$ has a *communication module* that can send/receive data to/from any other robot $\mathcal{R}_j$ contained in a communication set D_c (see Fig. 5.2). We assume that $D_p \subseteq D_c$, so that if $\mathcal{R}_i$ can detect $\mathcal{R}_j$ it can also communicate with it. Each message sent by $\mathcal{R}_i$ contains: (1) the robot index i (2) the estimate $\hat{x}_i^\star$ as provided by the self-localization module (3) the measures coming from the robot detector.

The relative position measures provided by the robot detector are *anonymous*, in the sense that they do not include the index j of the detected robot. This is true, for example, when the detection process relies on features that are common to all the robots. A consequence of anonymity is the existence of *ambiguous* situations (such as that in Fig. 4.12a), where the same set of measures is obtained for different configurations of the multi-robot system. As shown in Fig. 5.2, the robot detector is also prone to *false positives* (it can be deceived by objects that look like robots) and *false negatives* (robots belonging to D_p which are not detected, e.g., due to line-of-sight occlusions). Hence, the measures coming from the robot detector will be generically referred to as *features* – they might or not represent actual robots. False

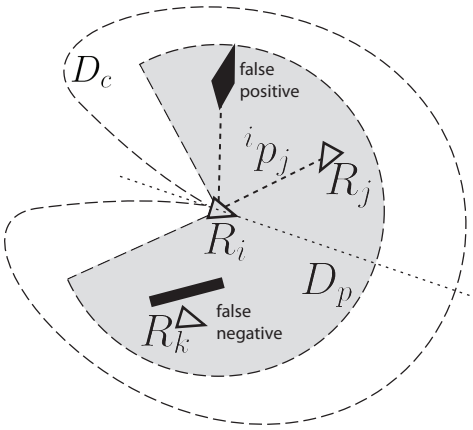

Figure 5.2: Robot detection and communication. Triangles are robots, black polygons are occluding objects, and the grey region is the perception set.

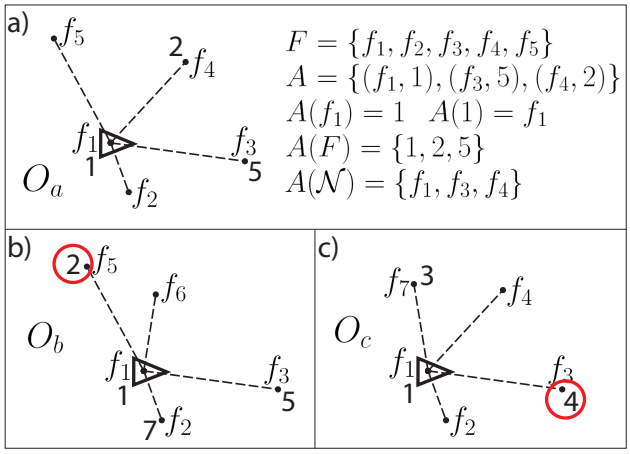

Figure 5.3: The structure of an observation (triangles are robots, points are features): a) An example observation O_a by $\mathcal{R}_1$; b) O_b is irreconcilable with O_a because robot $\mathcal{R}_2$ is associated to a different feature; c) O_c is irreconcilable with O_a because feature f_3 is associated to a different robot.

negatives (robot belonging to D_c that do not receive messages) may also affect the communication, whereas false positives may be easily avoided by appropriate message coding.

Our objective is to solve the *absolute mutual localization* problem for the generic i-th robot, i.e., to estimate ${}^i t_j^\star$, for $j \neq i$. As a byproduct, this will also solve the *relative mutual localization* problem, i.e, provide an estimate

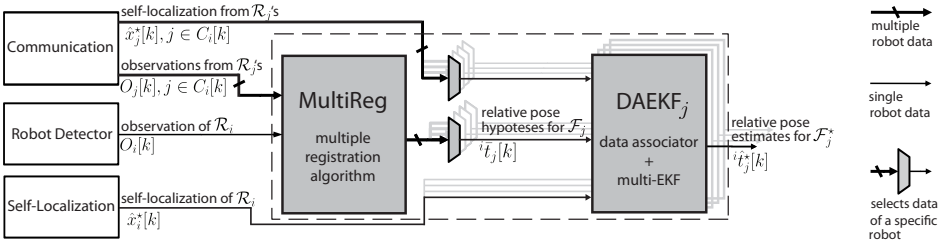

Figure 5.4: A scheme of the mutual localization system that runs on $\mathcal{R}_i$.

of ${}^it_j$, for $j \neq i$. Note that, if the anonymity assumption is removed, the geometric computation of ${}^it_j^\star$ from $x_i^\star$, $x_j^\star$ (estimated by the self-localization modules) and ${}^ip_j$, ${}^jp_i$ (measured by the robot detectors) becomes a simple exercise.

Each robot detector provides an observation in which all features are anonymous, except for the feature at the origin which is associated to the index of the measuring robot; this is called a *raw observation*.

5.3 The mutual localization system

The mutual localization system running on $\mathcal{R}_i$ is composed by a cascade of two subsystems, as shown in Fig. 5.4. The first subsystem is a memoryless registration algorithm called *MultiReg*. Denote by $C_i[k] \subset \mathcal{N}$ the set of (indexes of) robots from which $\mathcal{R}_i$ receives data in the time interval $[t_{k-1}, t_k)$. At each step k, the inputs of MultiReg are a set of observations: one is provided directly by the robot detector, while the others come from the robots in $C_i[k]$ through the communication module. The output of MultiReg is a set ${}^i\bar{t}_j$ of hypotheses on the relative poses ${}^it_j$, for each $j \in C_i[k]$.

The second subsystem, called *DAEKF*, is a variable-size array of components, DAEKF$_j$, $j \in \cup_{h=1}^k C_i[h]$, each consisting of a *data associator* and a *multi-EKF*. The input of DAEKF$_j$, at step k, is the set ${}^i\bar{t}_j$ of current hypotheses on ${}^it_j$. At the same time, each DAEKF$_j$ also receives the estimates $\hat{x}_i^\star$ and $\hat{x}_j^\star$, $j \in C_i[k]$, respectively from the self-localization and the communication module. The output of DAEKF$_j$ is a set ${}^i\hat{t}_j^\star$ of estimates of ${}^it_j^\star$, for each $j \in \cup_{h=1}^k C_i[h]$. In the following, we detail the structure of MultiReg and DAEKF.

At each step k, the generic robot $\mathcal{R}_i$ executes the MultiReg algorithm to perform a memoryless *multiple registration* among the observations coming from $\mathcal{R}_i$ and all the $\mathcal{R}_j$'s, $j \in C_i[k]$; in our context, this means computing all the possible relative poses ${}^it_j$ of the frames attached to the $\mathcal{R}_j$'s using

purely geometric arguments.

5.4 Data association and EK filtering

At the k-th step, the DAEKF subsystem running on robot $\mathcal{R}_i$ is composed by an array of $|\cup_{h=1}^{k} C_i[h]|$ components (see Fig. 5.4). Each component is associated to a robot $\mathcal{R}_j \in \cup_{h=1}^{k} C_i[h]$ and provides estimates of ${}^{i}t_j^{\star}$. At step k its inputs are:

1. The estimates provided by the self-localization modules

$$\begin{aligned} \hat{x}_j^{\star}[k] &= x_j^{\star}[k] + w_j^{\star}[k] \\ \hat{x}_i^{\star}[k] &= x_i^{\star}[k] + w_i^{\star}[k] \,, \end{aligned}$$

   where $w_j^{\star}[k]$ and $w_i^{\star}[k]$ are gaussian noises with zero mean and covariances $\widehat{Q}_j^{\star}[k]$ and $\widehat{Q}_i^{\star}[k]$. Note that $\hat{x}_j^{\star}[k]$ is available provided that $\mathcal{R}_j \in C_i[k]$;

2. The set of hypotheses about the relative pose ${}^{i}t_j[k]$

$${}^{i}\bar{t}_j[k] = \{\ldots, {}^{i}\bar{t}_{jh}[k], \ldots\} \,,$$

   provided by MultiReg. Here, ${}^{i}\bar{t}_{jh}[k]$ is a gaussian random variable with unknown mean and covariance ${}^{i}\bar{Q}_{jh}[k]$.

Each DAEKF$_j$ (see Fig. 5.5) is composed by a *data associator* (DA$_j$) and a variable-size *multi-EKF* (EKF$_j$ $= \{\ldots, \text{EKF}_{jl}, \ldots\}$). The data associator is a 'nearest neighbor-like' memoryless algorithm [83] in charge of dispatching each relative pose hypothesis ${}^{i}\bar{t}_{jh}[k]$ produced by MultiReg to the appropriate EKF$_{jl}$ of the array. This EKF$_{jl}$, taking $\hat{x}_j^{\star}[k]$, $\hat{x}_i^{\star}[k]$ and ${}^{i}\bar{t}_{jh}[k]$ as inputs, produces an estimate ${}^{i}\hat{t}_{jl}^{\star}[k]$ of ${}^{i}t_j^{\star}$, together with its covariance matrix.

At step k, DA$_j$ converts each ${}^{i}\bar{t}_{jh}[k]$ to one hypothesis on ${}^{j}t_i^{\star}$, using $\hat{x}_j^{\star}[k]$, $\hat{x}_i^{\star}[k]$ and geometric computation. Then, the covariance-weighted distance of each hypothesis from the estimate of each EKF$_{jl}$ is computed. Each hypothesis is used as input for the filter with the closest estimate, provided that the distance is smaller than a *maximum distance* $d_{\max}$. For each hypothesis ${}^{i}\bar{t}_{jm}[k]$ which is not associated to any EKF$_{jl}$, a new filter is added to EKF$_j$, initialized with the triple $\{{}^{i}\bar{t}_{jm}[k], \hat{x}_i^{\star}[k], \hat{x}_j^{\star}[k]\}$.

At each step, a *mark* is associated to each EKF$_{jl}$, given by the number of *hits* (steps in which a hypothesis is associated to that filter) in the last $L[k]$ steps, where $L[k]$ is the *backward horizon*. The EKF$_{jl}$ with the highest

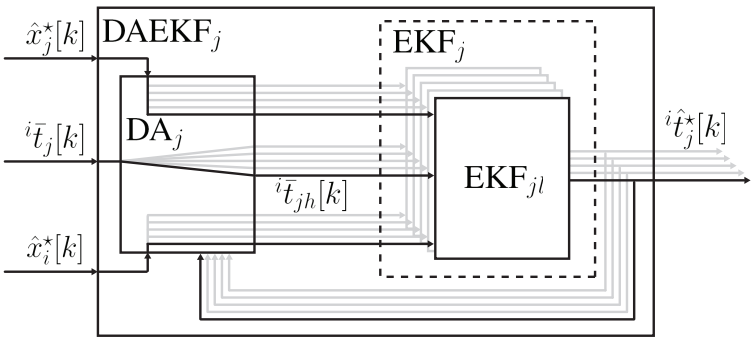

Figure 5.5: A scheme of the DAEKF$_j$ which estimates ${}^it_j^\star$.

mark provides the best current estimate of ${}^it_j^\star$. An EKF$_{jl}$ whose mark goes below a certain threshold $\mu_{\min}$ is removed from the EKF$_j$ array. The model equation of the generic EKF$_{jl}$, used to estimate the constant parameter ${}^it_j^\star$, is ${}^it_j^\star[k] = {}^it_j^\star[k-1]$. The measurement equation, the Jacobian matrix and the expression of the Kalman gain can be found in [70].

5.5 Experiments

The proposed ML method has been implemented and validated using the MIP experimental software, described in [84] and available at `http://www.dis.uniroma1.it/labrob/software/MIP`. In particular, we have simulated the robots with Player/Stage in the testing phase, and used an actual team of 5 Khepera III robots in the experimental phase (see Fig. 5.6). Each robot is equipped with a Hokuyo URG-04LX laser range finder, that has a 240° angular range and a linear range artificially limited to 2 m.

The robot detector is a simple feature extraction algorithm that inspects the laser scan searching for the indentations made by the vertical cardboard squares mounted atop each robot in the shadow zone of the range finder. Since each square can give $1-12$ cm wide indentations of the laser scan, depending on the relative orientation between the measuring and the measured robot, the detector cannot distinguish among robots and obstacles whose size is in the same range. Moreover, the squares are identical, and therefore the features are anonymous. Accurate measures of the ${}^it_j^\star$ to be used as ground truth are taken in advance by a human operator. Self-localization is obtained by simple dead reckoning.

Fig. 5.7 refers to the early steps of a 5 min experiment involving five robots (numbered 0–4) and four deceiving obstacles. The results shown are

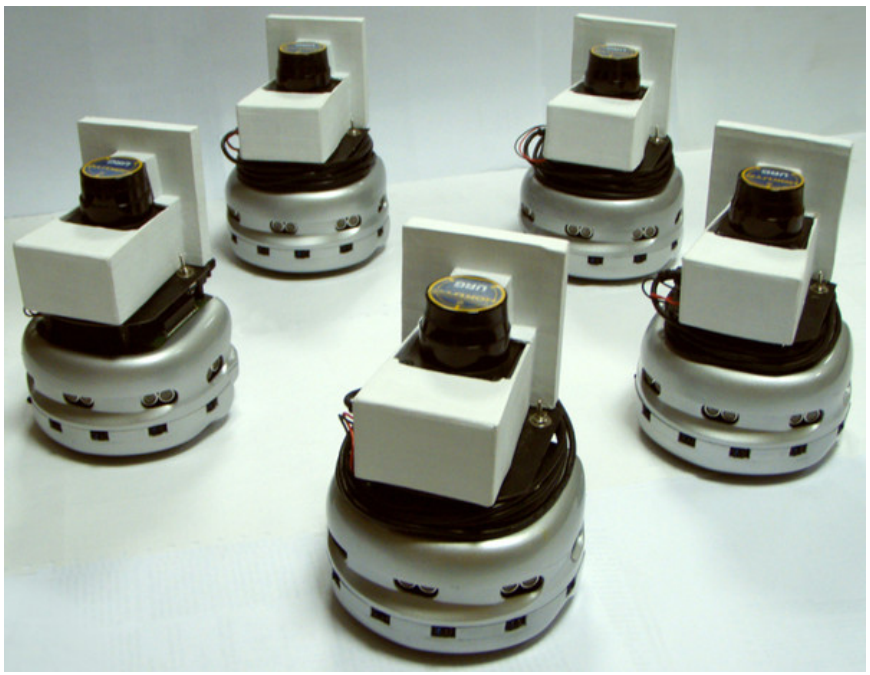

Figure 5.6: The 5 Khepera III used in our experiments. A cardboard square is placed in the shadow zone of the URG-04LX to allow robot detection.

those produced by the mutual localization system running on robot $\mathcal{R}_4$ at 10Hz, which is the same frequency of the laser range finder. Note that the initial configuration of the team is highly symmetrical, and therefore very ambiguous for MultiReg; moreover, it contains several occlusions. At the beginning the best available estimates for $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ are wrong, due to occlusions and symmetry; however, as the experiment proceeds, correct estimates are quickly identified and prevail.

Fig. 5.8 summarizes the result of the experiment in terms of estimation errors (cartesian and angular) and marks assigned by DAEKF_1 to the available hypotheses on the relative pose of the $\mathcal{R}_1$ fixed frame with respect to that of $\mathcal{R}_4$. The timescale is 150 sec. Note in particular how the best estimate, easily identifiable by the three (darker) lines that achieve the smaller errors and the higher mark, appears only 5 sec (approximately) after the beginning of the experiment.

Figures 5.9 and 5.10 refer to another experiment, whose peculiarity is that 2 robots act as deceiving movable obstacles, since they move but do not broadcast their observations. As we expect, the estimation system works well also in this case.

Figure 5.7: Above: stroboscopic motion in the early steps of the experiment (robots are numbered 0–4). Below: the best estimates for the same steps (lighter impressions indicates older estimates) and the measured features measured by the robot detectors (small dots).

Another experiment, in which two robots are used as deceiving mobile obstacles (they do not communicate their measures), is shown in `http://www.dis.uniroma1.it/labrob/research/mutLoc.html`.

5.5.1 MultiReg running time

The running time of MultiReg, which accounts for most of the cycle time of our mutual localization system, depends on the number $|\Omega|$ of raw observations it receives in input, as well as on the ambiguity of the multi-robot system configuration. Note that $\max |\Omega| = n$. In unambiguous situations, $|\Omega|(|\Omega| - 1)/2$ binary registrations are needed to produce a solution; since

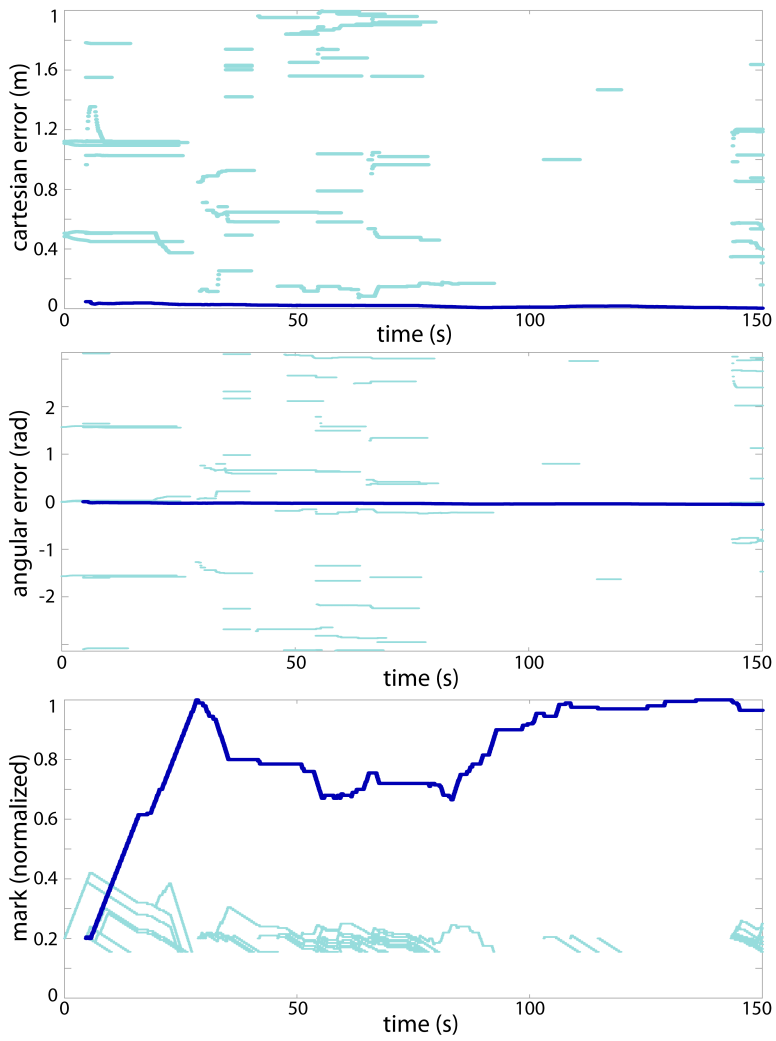

Figure 5.8: Errors and mark for the best estimate on the pose of the fixed frame of $\mathcal{R}_1$ with respect to that of $\mathcal{R}_4$. The light lines in the background refer to all other hypotheses on the estimate. Each light line represents the life of an estimate. Plotting is interrupted for estimates whose normalized mark goes below 0.15.

each binary registration requires constant time in the worst case, the time complexity of MultiReg in this case is $o(n^2)$. In ambiguous situations, as many as $(n-1)!$ irreconcilable solutions may exist, leading to an $o(n!)$ time complexity.

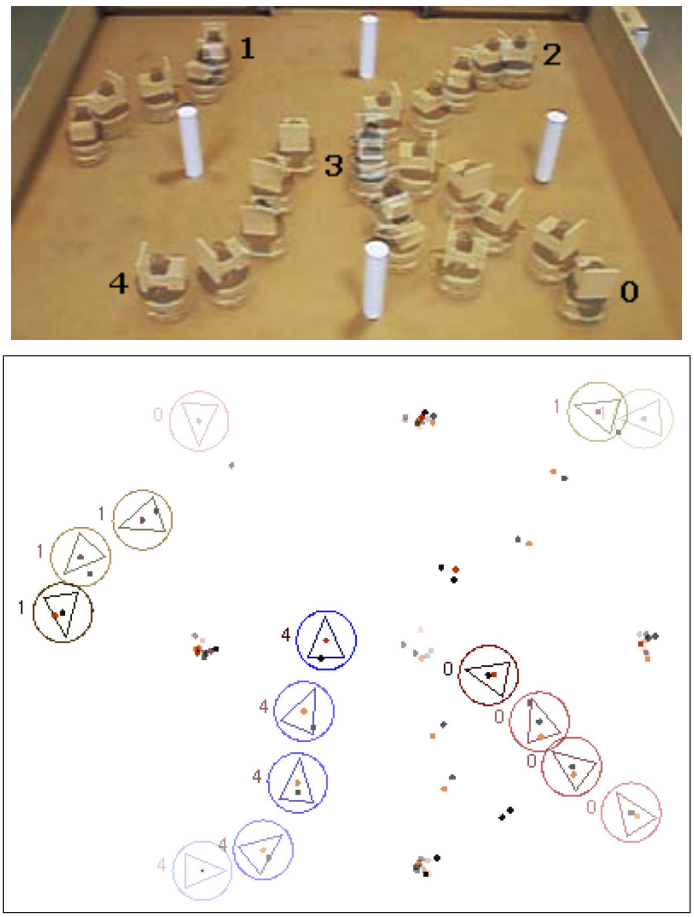

Figure 5.9: Above: stroboscopic motion of the early step of the second experiment. Below: the best estimate for the same steps (lighter robots indicates older estimates). Robots $\mathcal{R}_2$ and $\mathcal{R}_3$ acts as deceiving movable obstacles.

Fig. 5.11 reports some statistics on the running time of MultiReg as a function of $|\Omega|$ and of the number of solutions it finds. The first plot shows that the upper bound increases exponentially, the lower bound is constant and the mean value time has an approximately quadratic increasing rate. These results are consistent with the above theoretical prediction. In the second plot, the mean value increases linearly whereas upper bounds are higher for small numbers of solutions. This is due to the fact that small solution numbers were much more frequent (about 25000 samples against a few dozens). All things considered, our experiments indicate that the

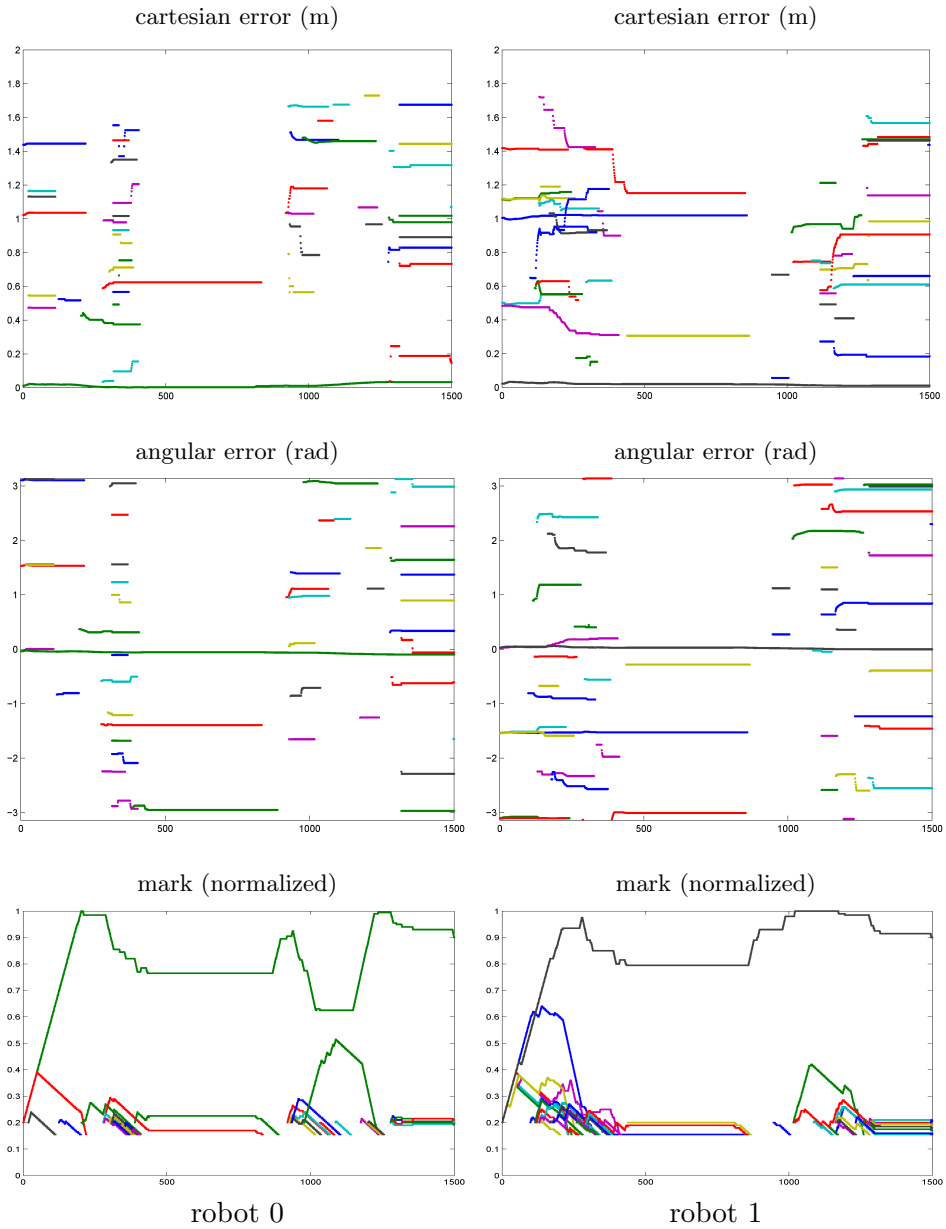

Figure 5.10: Errors and marks of the estimates generated by the 2 multi-EKFs in the second experiment. Time step is 100 ms.

proposed mutual localization system can easily run at 10 Hz for a team of five robots.

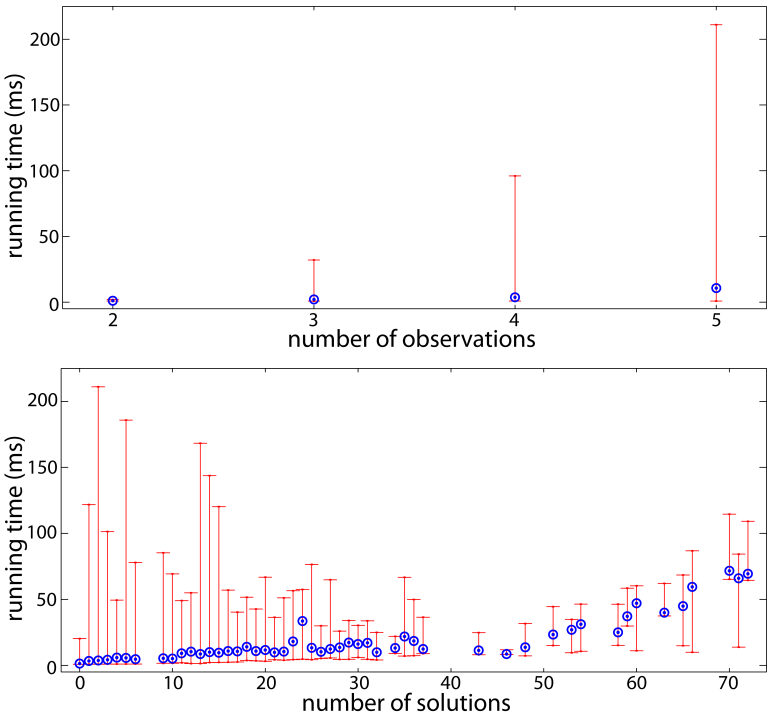

Figure 5.11: Max-min (horizontal ticks) and average (circles) values of the running times (ms) of MultiReg with respect to the number of input observations and the number of solutions. Data based on 63898 executions of MultiReg during 38 real robot experiments.

5.6 Open issues

In this chapter, we have presented a novel method for mutual localization in multi-robot systems. In particular, our technique allows to estimate the changes of coordinates among the various robot frames using relative position measures that are anonymous as well as affected by false positives and negatives. The data available to each robot are processed by the MultiReg algorithm to obtain a set of hypotheses on the relative pose of the other robots of the team. The anonymity hypothesis causes an ambiguity in the inversion of the observations, that is solved using a multi-hypotheses filter. Satisfactory performance has been obtained both in simulations and in real robot experiments, showing that the proposed localization system is applicable in practice.

One possible problem with the proposed approach is that the running time of MultiReg may increase considerably if the number of its solutions

grows. However, the case with factorial number of different solutions is obtained only in particular configurations in which a subset of the observations are roughly the same. We are currently developing a theoretical study of the ambiguity introduced by the anonymity hypothesis, aimed at reducing the number of MultiReg solutions by generating in linear time a single representative for each class of equivalent solutions. Another possible technique to reduce the complexity might be the use of some threshold on the number of the registered observations. Another improvement would be obtained by considering only solutions that are sufficiently close to the currently available estimates, so as to introduce a feedback mechanism from the DAEKF subsystem to MultiReg. Moreover, the 'nearest neighbor' policy of the data association can be avoided by resorting to a particle filter that samples also on data association, such as that developed in [64]. Work in progress also deals with the application of the developed system to decentralized tasks, such as formation control and cooperative exploration.

Chapter 6

Distributed Target Localization and Encircling with a Multi-robot System

This chapter presents an experimentally validated estimation/control system for the detection and encircling of an object by means of a team of robots. In order to accomplish their task, the robot must also equalize their radius of revolution, inter-distances and speeds. All these goals are achieved in a distributed way, using limited-range transceivers and relative-position sensors which are assumed noisy, anisotropic, and anonymous. The latter characteristic, which is distinctive of this setting, means that the sensors do not provide the identity of the measured object, as in Chapters 4 and 5. An extensive experimentation supports the theoretical analysis.

6.1 Introduction

The detection and encircling of an object by means of a team of robots is the main topic of this chapter. The great deal of applications related to this subject explains its appealing to the literature during the recent years. In fact, encircling may be used, among the others, to retrieve and merge data about an object from different viewpoints (observing), protect a member of the team from exterior agents (escorting), or prevent the motion of an alien object (entrapment). In our setting, in order to accomplish their task, the robot must also equalize their radius of revolution, inter-distances and speeds. All these goals are supposed to be achieved in a distributed way, using limited-range transceivers and relative-position sensors which are assumed noisy, anisotropic, and anonymous. The latter characteristic, which is distinctive of this setting, means that the sensors do not provide the

identity of the measured object. We refer the reader to Chapters 4 and 5 for a thorough description of the anonymous-position sensor setting, to which the commonly used estimation techniques do not apply. The main advantages of this setting are: *i)* the applicability to adverse environmental conditions (unstable light or darkness, rain, fog, etc.) which may hide the distinguishing features used for identification, *ii)* the possibility of exploit the complete resemblance of the members of the team to let the identity of some of them to remain secret for security or disguising purposes, and *iii)* the applicability in team missions requiring mimicry or stealth capabilities of the members.

Several control techniques have been proposed in literature to achieve the encircling or some related formation control objectives. The majority assume the presence of an ideal measurement system. In case of unlimited communication between holonomic agents, some of them are based on potential fields. [85] analyze the performances of a swarm approaching a goal and its cohesion in presence of attractive and repulsive profiles. [86] make use of virtual points to deform the shape of the formation for some task and take care of collision avoidance. In [60] a group of unicycles is steered at constant linear speed around a common point. In [87] an extension with communication limitations is considered. In both a Lyapunov control approach is used. The uniform spacing along the circle is achieved using the so called (M, N) *patterns*, for which a controller based on phase potential s is proposed. In this approach the controller is nested, i.e., whenever the circle has to be divided in M sectors, the moments of order $M-1$ have to be calculated. Another limitation is the constant linear speed. Strictly related to this approach is the work presented in [88] in which a group of robot is steered on a circle using a centralized vision system to obtain positions and headings. [61] propose a centralized approach where a global vision system provides the configuration of the system to a unique external controller, derived from the null-space manipulator theory, that computes the trajectories considering a set of required tasks. A good estimate of the pose of every robot with respect to a fixed frame is required to compute the formation Jacobian. In [89] a control for nonholonomic agents with directional sensors is presented using a Lyapunov approach based on the cartesian displacement between robots. The control law requires an a priori knowledge of the number of agents, hence insertion and faults are not taken into account. Some algorithms, as in [90], present control laws whose convergence is based on the topology of the communication graph. [91] consider a fixed topology algorithm based on the concept of α *stability* property. Some works, as in [92], demonstrate that agents reach a common estimate (consensus) of

global quantities under the assumption of strong connectivity of the communication graph. Extensions thi this problem have been proposed in [93] and [94]. All theses works ignore the actual presence of two graphs, the communication and the perception one. Loosely related to this problem, is the pursuit-evasion problem, addressed, among the others, in [95], where an neural networks artificial intelligence approach to the *pursuer-evader* problem is used, and in [96], where a four-phase bio-inspired cooperative strategy to confine an evader in a bounded region is demonstrated assuming unlimited sensing capabilities and the instantaneous measurement of position and velocity of the evader. In [97] a real experiment with an heterogeneous formation is performed using both aerial and ground vehicles. Such algorithms are based on a probabilistic grid map representation of the environment that is built during the experiment. Some recent works introduce noisy measurements for which distributed estimation algorithms are inserted. In [98] a formation controller with estimation filters is simulated and demonstrated using the passivity property of the system, while in [87] a formation controller based on the quality of the estimate is analyzed.

In all the previous works a direct knowledge of the quantities involved in the control laws assumed and labelled measures are assumed. When experimental results are presented, these quantities are acquired by a centralized system, typically a vision one, in which different robots are discriminated by identification tags. This architecture is usually to test the robustnes to robot faults and estimate error, and needs for a structured environment to work. In a real context it is very important to design a decentralized estimation system that uses the measures taken directly by the agents to estimate the quantities involved in the control law. For the best of our knowledge, our work is innovative because it takes into account a more realistic and minimal model of the measured quantities, achieving the encircling task in an more unstructured environment, and validating it by experiments.

6.2 Problem Formulation

Consider a group of identical mobile robots with unicycle kinematics and a target, all moving on the plane. The target may be an inanimate object, a robot, or even a living agent. The task assigned to the robots is to *encircle* the target, i.e., move around it in a regular circular formation, also called a *splay state* formation [87]. The target is assumed to be stationary for the sake of analysis, but we will experimentally show in Sect. 6.6 that 'slow' movements may be accommodated. No a priori information is available about the cardinality of the group, the initial configuration the robots, and

the position of the target. In addition, the task must be realized on the basis of local sensory information only.

Clearly, the encirclement task requires each robot of the group to be able to localize the target as well as the other robots. We may distinguish between two cases, depending on whether the appearance of the target is *identical* to or *different* from that of the robots. In the first case, the robots and the target may be detected by a single sensory device, whereas two separate detectors (a robot detector and a target detector) must be used in the second case. We shall consider in detail the first case; however, the proposed technique can be trivially extended to the second.

A target that appears to be identical to the robots may be an unfriendly agent in disguise that must be made inoffensive, or actually one of the robots that must be guarded or escorted by the others. An unfriendly target does not communicate with the other robots (*non-cooperating target*), while communication with a friendly target is obviously possible (*cooperating target*).

We assume that each robot is equipped with:

1. A *robot detector*, a sensor device that measures the relative position (not the orientation) of other robots and of the target w.r.t. the detector, provided that they fall in a perception set D_p that is rigidly attached to it. The shape of D_p is arbitrary, and in particular it may contain blind zones. The relative position measures provided by the robot detector are *anonymous*, i.e., they do not convey the specific identity of the detected robot (hence, the target is detected as a robot).

2. A *communication module* that can send/receive data to/from any other robot contained in a communication set D_c such that $D_p \subseteq D_c$.

Under the above assumptions, it may happen that one robot can detect another, but not vice versa; in other words, the robot *detection graph* is directed (and includes the target as a *sink*). However, if one robot can detect another it can also communicate with it; the *communication graph* is therefore undirected and contains the detection graph, with the exception of the target and its in-arcs. In the following, two robots that can communicate with each other are simply called *neighbors*.

6.3 System Architecture

The encirclement system installed on each robot, shown in Fig. 6.1, consists of two main modules. The *mutual localizer* is in charge of computing the

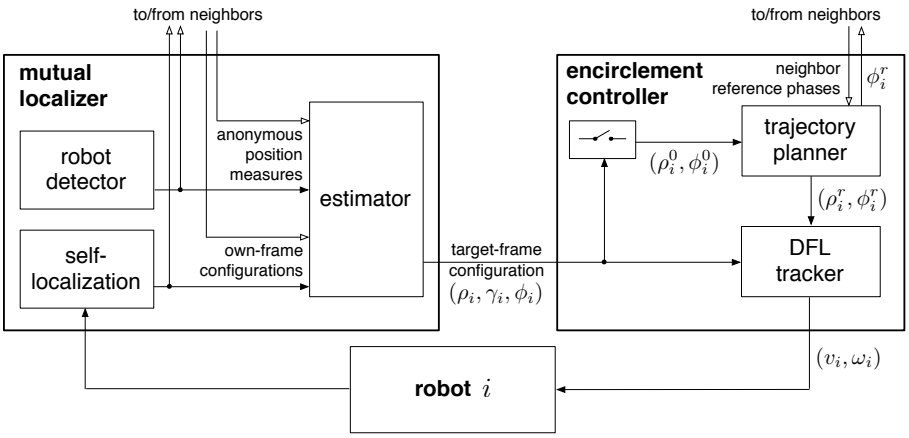

Figure 6.1: The structure of the encirclement system that runs on the i-th robot.

configuration of the robot in a reference frame centered at the target. This information is passed to the *encirclement controller*, that generates a reference trajectory for the robot and the feedback control inputs that guarantee its tracking.

The mutual localization module implements the method proposed by [26], to which the reader is referred for a detailed description. The inputs to this component are (1) the anonymous relative position measures (which include the target, if this is contained in D_p) coming from the robot detector (2) an estimate of the configuration of the robot in its own frame, computed by any self-localization (position tracking) algorithm (3) the same information (i.e., anonymous relative position measures and robot configuration in its own fixed frame) obtained via communication from each neighbor. A multiple registration algorithm followed by a multi-EKF are used to process these data and compute an accurate estimate of the configuration of each robot in a common fixed frame. While a cooperating target is directly identified and localized with this procedure, it is interesting to note that a non-cooperating target can still be singled out by the mutual localization module as the only 'robot-like' object that does not communicate its data. From the mutual localization results, each robot can directly derive an estimate of its configuration (position and orientation) with respect to a *common* target-centered frame.

The encirclement control module generates the control inputs to the robot using the target-centered configuration of the robot computed by the mutual localizer as well as information coming from the neighbor robots.

The structure of this module is discussed in detail in the following section.

6.4 Encirclement Controller

In view of the encirclement objective, it is convenient to express the configuration of the generic i-th robot in polar coordinates with respect to a reference frame centered at the target, as in Fig. 6.2. In particular, these coordinates are the distance ρ_i of the unicycle wheel center from the origin, the angle γ_i that the sagittal (forward) axis of the robot forms with the line joining the unicycle wheel center to the origin, and the angle ϕ_i between the same line and the x axis. In the following, ϕ_i is also called *phase* of the robot). The kinematic model of the unicycle is then written as [99]

$$\begin{aligned}\dot{\rho}_i &= -v_i \cos\gamma_i \\ \dot{\gamma}_i &= (\sin\gamma_i)/\rho_i - \omega_i \\ \dot{\phi}_i &= v_i(\sin\gamma_i)/\rho_i,\end{aligned}$$

where v_i and ω_i are respectively the driving and steering velocity inputs.

It is assumed that the robot index i refers to the cyclic counterclockwise ordering of the robots defined by their increasing phase angles (see Fig. 6.2). For the i-th robot, denote by $\bar{\phi}_i$ the mean between the phases of the successor (robot $i+1$) and the predecessor (robot $i-1$) of the robot. Correct execution of the encirclement task requires that

$$\lim_{t\to\infty} \rho_i(t) = R \quad \lim_{t\to\infty} \phi_i(t) = \bar{\phi}_i(t) \quad \lim_{t\to\infty} \dot{\phi}_i(t) = \Omega \quad \forall i, \tag{6.1}$$

where R and Ω are respectively the encirclement radius and angular speed, which must be the same for all robots.

The entrapment control module (see Fig. 6.1) works as follows. The initial configuration $(\rho_i^0, \gamma_i^0, \phi_i^0)$, provided by the mutual localization module[1] is used to plan a reference trajectory for the robot. In particular, such trajectory is specified by an exosystem that assigns reference evolutions ρ_i^r, ϕ_i^r to the coordinates ρ_i, ϕ_i. In fact, it is easy to verify that these two are *flat* outputs [100] for the unicycle in polar coordinates, i.e., once an evolution is assigned to them it is possible to compute algebraically the corresponding evolution γ_i^r of the remaining variable γ_i as well as the reference inputs v_i^r, ω_i^r. The reference outputs ρ_i^r, ϕ_i^r are fed to a feedback controller based

[1] This assumes that the configuration estimate is immediately reliable. In practice, it may be necessary to perform a preliminary motion of the multi-robot system aimed at improving the accuracy of the estimate. To this end, the anti-symmetry control law proposed in [55] may be used.

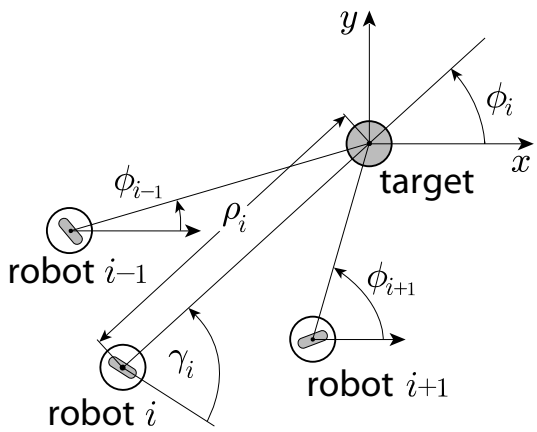

Figure 6.2: Polar coordinates for the i-th robot and the cyclic ordering defined by phases.

on Dynamic Feedback Linearization (DFL), that generates the control inputs v_i, ω_i so as to guarantee global exponential tracking of the reference trajectory [28]. It should be noted that ρ_i^r, ϕ_i^r are initialized at ρ_i^0, γ_i^0, so that the transient is extremely fast. During its operation, the DFL tracker uses the current estimate of the target-frame robot configuration $(\rho_i, \gamma_i, \phi_i)$ computed by the mutual localization module.

In the following, we consider three slightly different versions of the basic encirclement task entailed by (6.1), and give the appropriate form of the trajectory planner (exosystem). In all versions, the encirclement radius R is assigned in advance. The reference radius $\rho_i^r(t)$ is therefore always generated by

$$\dot{\rho}_i^r = K_\rho(R - \rho_i^r) \qquad \rho_i^r(0) = \rho_i^0, \tag{6.2}$$

where K_ρ is a positive gain. As a consequence, $\rho_i^r(t)$ exponentially converges to R for any initial condition. Note that $\rho_i^r(t)$ does not depend on the reference radius of any other robot.

The three versions of the encirclement task differ on the procedure used by the robots to agree on the common value of the angular speed Ω in (6.1). They are analyzed in detail below.

Encirclement – Version 1.

In the first version, the angular speed Ω is also specified in advance. The reference phase $\phi_i^r(t)$ for the i-th robot is generated by

$$\dot{\phi}_i^r = \Omega + K_\phi(\bar{\phi}_i^r - \phi_i^r) \qquad \phi_i^r(0) = \phi_i^0, \tag{6.3}$$

where K_ϕ is a positive gain and $\bar{\phi}_i^r$ is the mean between the reference phases of the predecessor and the successor (in accordance with the counterclockwise cyclical ordering of the reference phases). We have the following result (the proofs of all the propositions are in the Appendix).

Proposition 6.1. *The flow of (6.2),(6.3) yields exponential convergence of ρ_i^r to R, of ϕ_i^r to $\bar{\phi}_i^r$, and of $\dot{\phi}_i^r$ to Ω, for any assigned R, Ω and any initial ρ_i^0, ϕ_i^0.*

An example of reference robot trajectories corresponding to the flow of (6.2),(6.3) is shown in Fig. 6.3. The robots approach the circle in such a way that the 'insertion points' are almost uniformly spaced, and actually achieve the required formation very quickly.

Encirclement – Version 2.

In the second version, the robots are assigned an *escape window* s, i.e., the time interval in which a point on the circle remains unvisited at the steady state corresponding to the asymptotic conditions (6.1). Being $s = 2\pi/n\Omega$, where n is the number of robots, the robots can in principle easily compute the required value of Ω as $\Omega = 2\pi/ns$; however, since n is not known a priori, an estimate $\hat{n}$ of this number is required.

Assume that each robot instantaneously computes its own estimate as $\hat{n}_i = 2\pi/\Delta_i^r$, where $2\Delta_i^r$ is the reference phase difference between the successor and the predecessor. The required angular speed for the robot is then computed as $\Omega_i = 2\pi/\hat{n}_i s = \Delta_i^r/s$. Using this expression for Ω in (6.3) we obtain the following exosystem for the reference phase:

$$\dot{\phi}_i^r = \Delta_i^r/s + K_\phi(\bar{\phi}_i^r - \phi_i^r) \qquad \phi_i^r(0) = \phi_i^0. \tag{6.4}$$

Proposition 6.2. *The flow of (6.2),(6.4) yields exponential convergence of ρ_i^r to R, of ϕ_i^r to $\bar{\phi}_i^r$, and of $\dot{\phi}_i^r$ to $2\pi/ns$, for any assigned R, s and any initial ρ_i^0, ϕ_i^0.*

Encirclement – Version 3.

In the third version, only the radius R is assigned, and the robots must autonomously agree on a common value of the angular speed Ω. The reference phase exosystem for the i-th robot is

$$\dot{\Omega}_i^r = K_\Omega(\bar{\phi}_i^r - \phi_i^r) \qquad \Omega_i^r(0) = 0 \tag{6.5}$$

$$\dot{\phi}_i^r = \Omega_i^r + K_\phi(\bar{\phi}_i^r - \phi_i^r) + \xi_i \qquad \phi_i^r(0) = \phi_i^0 \tag{6.6}$$

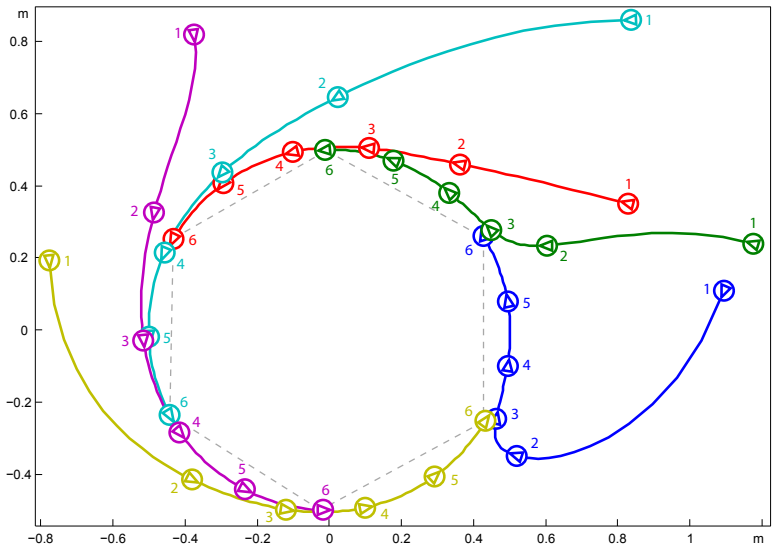

Figure 6.3: Reference trajectories corresponding to the flow of (6.2),(6.3) for a generic initial configuration of a 6-robot system, with a target located at the origin. The configuration of each reference robot along its trajectory is explicitly shown at six equispaced time instants, identified by $1, \ldots, 6$.

where ξ_i is a costant forcing term. Denote by $\bar{\xi}$ the average of the forcing terms ξ_i over the multi-robot system.

Proposition 6.3. *The flow of (6.2), (6.5–6.6) yields exponential convergence of ρ_i^r to R, of ϕ_i^r to $\bar{\phi}_i^r$, and of $\dot{\phi}_i^r$ to $\bar{\xi}$, for any assigned R and any initial ρ_i^0, ϕ_i^0.*

An interesting feature of this third scheme is that the common frequency of the phase reference trajectories can be regulated by acting on a single robot; to this end, it is sufficient to let $\xi_i = 0$ for all the robots but one.

To allow the implementation of (6.3), (6.4), or (6.5–6.6) all the robots must broadcast their current reference phase through the communication system. However, each robot computes its reference trajectory and control inputs autonomously on the basis of local information, i.e., its own configuration and data coming from the neighbors.

6.5 Conditions for Task Achievement

The proposed method will achieve the encirclement task provided that the robots can localize the target and each other. In this section, we briefly

discuss the conditions under which these two requirements are actually satisfied. Recall that the mutual localization module used in our encirclement scheme is effective within weakly connected components (simply called *subnets* in the following) of the robot detection graph, provided that $D_p \subseteq D_c$ and multi-hop communication is used [26].

The first condition may be derived from the analysis of the desired steady state of the system, in which the n robots are uniformly distributed along a circle of radius R. In this formation, the whole detection graph must be weakly connected, i.e., a single subnet must exist. In view of the circle topology, this property is certainly guaranteed if each robot can detect the target and the successor robot with respect to the cyclic phase ordering (that is actually the predecessor if Ω is positive). For example, this is true if D_p is a frontal circular sector with central angle at least $\pi + \epsilon$ wide, with ϵ any positive number, and radius at least $\max\{R, 2R \sin \pi/n\}$.

The second condition is instead obtained considering the beginning of the encirclement task. To localize the target at $t = 0$, each subnet of the detection graph must contain at least one robot that detects the target. From that moment on, all the robots will get closer to the target in view of the reference evolution (6.2) for ρ, and therefore target detection is guaranteed throughout the task (this is easy to show if D_p has the shape discussed above). In particular, all the subnets will merge into a single connected component that includes the whole graph.

Note that the first condition (on D_p) concerns the robot detector, whereas the second (on the detection graph at $t = 0$) restricts the admissible initial arrangements of the robots with respect to the target. Taken together, they are a *sufficient* condition for task achievement — less demanding requirements may be enforced (in particular, on the shape of D_p) but their efficacy would be more difficult to prove.

6.6 Experiments

We have experimentally validated our encirclement scheme with a system of five Khepera III robots. Each robot is equipped with a wi-fi card and a Hukuyo URG-04LX laser sensor with an angular range of 240° and a linear range artificially limited to 2 m. The robot detector is a simple feature extraction algorithm that inspects the laser scan searching for the indentations made by the vertical cardboard squares mounted atop each robot (in the blind zone of the range finder). Since each square can give indentations of the laser scan that are $1 \div 12$ cm wide, depending on the relative orientation between the measuring and the measured robot, the detector

cannot distinguish among robots, the target and obstacles whose size is in the same range. Odometry is used as self-localization, in fact, an accurate self-localization s not needed in our scheme. The encirclement scheme has been implemented using the MIP architecture[2] which provides a multitasking estimation/control framework, a realistic simulation environment, and allows direct porting for execution on real robots.

A typical experimental result is summarized in Fig. 6.4. Here we are considering version 1 of the encirclement task, and hence (6.2–6.3) as a trajectory planner, with $R = 0.5$ m and $\Omega = 0.06$ rad/s. One of the robot is used as a stationary (cooperating) target. At the start (snapshot 1), only three robots are active. At $t_1 = 200$ s, with the three robots rotating around the target in a regular formation, another robot is added (snapshot 2); the four robots then achieve a regular formation (snapshot 3). At $t_2 = 310$ s one of the robot is manually placed away (snapshot 4). The four robots achieve again a regular formation, since the mutual localizer quickly recovers the large odometry estimation error (not showed). At $t_3 = 490$ s a robot is switched off and removed. The three remaining robots rearrange themselves in a regular formation (snapshot 5). Finally, at $t_4 = 600$ s another robot is removed and the formation becomes a 1 m wide dipole. The evolution of the experiment is also illustrated by Fig. 6.5, that shows the plots of the distances between consecutive robots, the distances between each robot and the target, and the robot angular speeds. The fact that during each phase the correct regular formation is promptly reached shows the reactivity of the proposed encirclement scheme.

In fact, Kidnappings and deletions can be considered also as robots failures, showing the fault tolerant capabilities of the mutual localization. In fact, at t_2 the estimation of one robots has a momentary bad estimation induced by the kidnapping of the other robot, which is suddenly recovered. Note that the estimator of each robot provides also a steady relative measure of robot temporarily or definitely hidden by means of occlusions as well as the relative measure of a robot on the back which is invisible to the robot detector due to the limited cone of the laser.

We have also run experiments with moving targets, obtaining satisfactory results as long as the speed of the target remains at least one order of magnitude smaller than that of the robots. One such experiment is shown in Fig. 6.6. Video clips of the experiments are available at http://www.dis.uniroma1.it/labrob/research/encirclement.html.

[2]http://www.dis.uniroma1.it/∼labrob/software/MIP/index.html

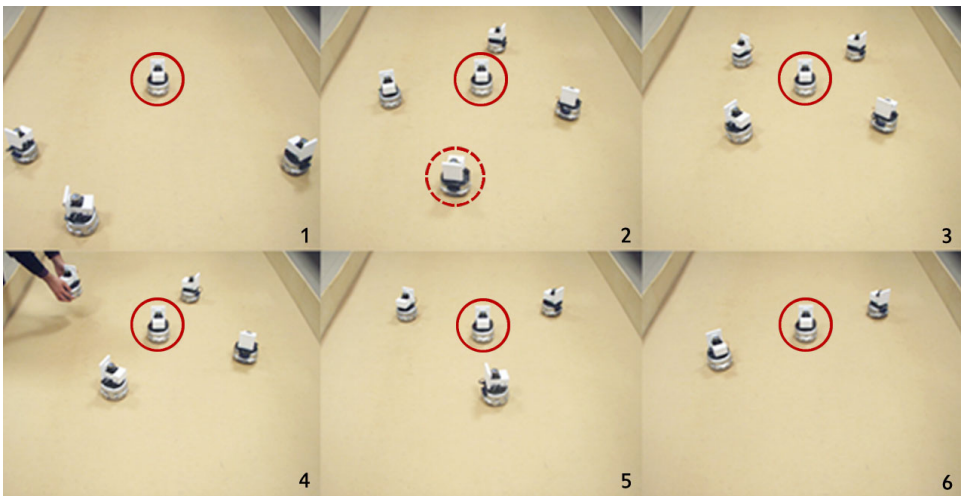

Figure 6.4: Encirclement of a stationary target (solid red circle).

6.7 Proofs

We index the robots accordingly to the ordering of the initial reference phases, and we use the following notations:

$$\begin{aligned}
\mathbf{1} &= (1 \cdots 1)^T\,, \quad \boldsymbol{b} = (-\pi\ 0 \cdots 0\ \ \pi)^T\,, \quad \boldsymbol{g} = (\pi\ 0 \cdots 0\ \pi)^T\,, \\
\boldsymbol{\phi}^r &= (\phi_1^r\ \cdots\ \phi_n^r)^T\,, \quad \bar{\boldsymbol{\phi}}^r = \boldsymbol{C}\boldsymbol{\phi}^r + \boldsymbol{b}, \quad \boldsymbol{\Delta}^r = \boldsymbol{D}\boldsymbol{\phi}^r + \boldsymbol{g}, \\
\boldsymbol{\Omega}^r &= (\Omega_1^r\ \cdots\ \Omega_n^r)^T\,, \quad \boldsymbol{\xi} = (\xi_1\ \cdots\ \xi_n)^T\,,
\end{aligned}$$

where $\boldsymbol{C}$ and $\boldsymbol{D}$ are the *circulant matrices* with first rows $(0\ \ 1/2\ \ 0\ \ \cdots\ \ 0\ \ 1/2)$, and $(0\ \ -1/2\ \ 0\ \ \cdots\ \ 0\ \ 1/2)$ respectively. The presence of $\boldsymbol{b}$ and $\boldsymbol{g}$ in the definition of $\bar{\boldsymbol{\phi}}^r$ and $\boldsymbol{\Delta}^r$ takes into account the fact that

$$\begin{aligned}
\bar{\phi}_1^r &= (\phi_2^r + \phi_n^r - 2\pi)/2, \quad \bar{\phi}_n^r = (\phi_1^r + 2\pi + \phi_{n-1}^r)/2, \\
\Delta_1^r &= (\phi_2^r - \phi_n^r + 2\pi)/2, \quad \Delta_n^r = (\phi_1^r + 2\pi - \phi_{n-1}^r)/2,
\end{aligned}$$

and allows to encode the topology of S^1 in $\mathbb{R}$. Also, denote by $\boldsymbol{e}_\phi = \bar{\boldsymbol{\phi}}^r - \boldsymbol{\phi}^r = (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{\phi}^r + \boldsymbol{b}$ and $\boldsymbol{e}_\Omega = \boldsymbol{\Omega}^r + \boldsymbol{\xi} - \bar{\xi}\mathbf{1}$ the phase and pulsation error vectors, where $\boldsymbol{I}$ is the identity.

Proof of Proposition 6.1. Writing (6.3) for all the robots we obtain

$$\dot{\boldsymbol{\phi}}^r = \Omega\mathbf{1} + K_\phi(\bar{\boldsymbol{\phi}}^r - \boldsymbol{\phi}^r).$$

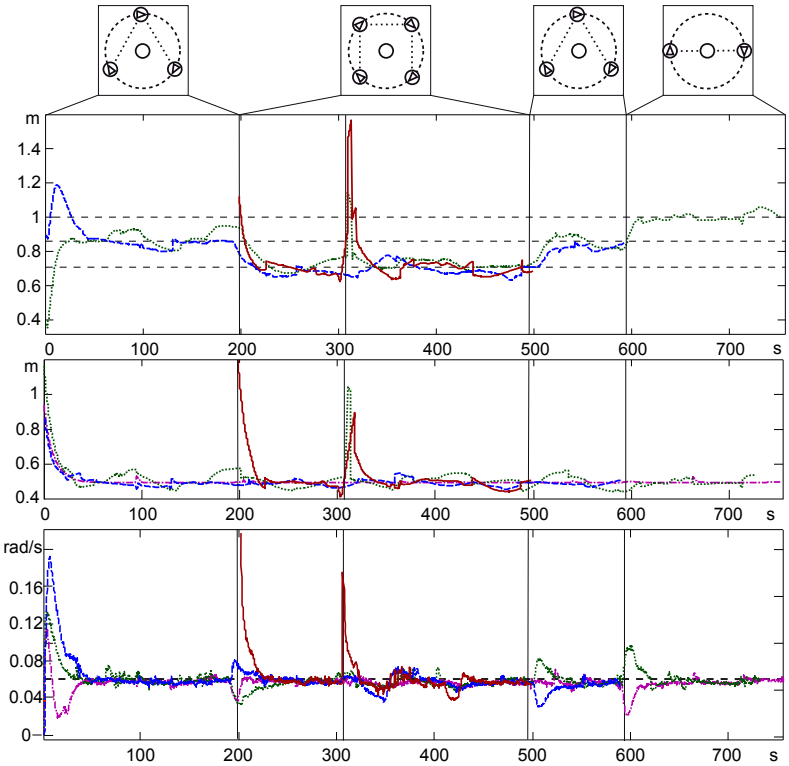

Figure 6.5: Plots for the previous experiment: distances between consecutive robots (top), distances between each robot and the target (center), angular speeds of the robots (bottom).

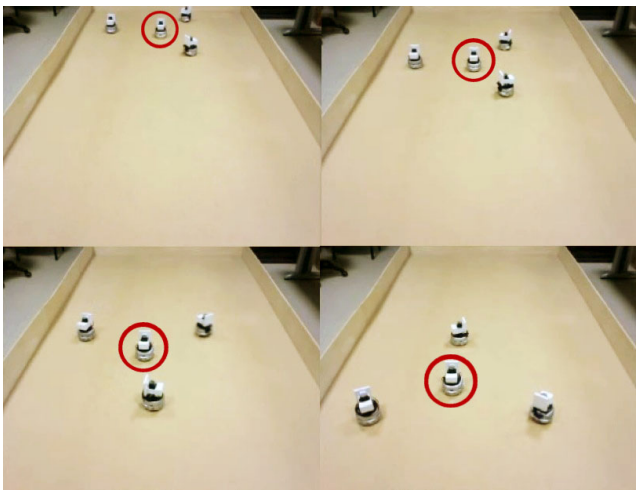

Figure 6.6: Encirclement of a moving target (solid red circle).

In order to prove the statement, it is sufficient to show that $\boldsymbol{e}_\phi$ goes to zero. Note that $\mathbf{1}^T(\boldsymbol{C}-\boldsymbol{I}) = (\boldsymbol{C}-\boldsymbol{I})\mathbf{1} = \mathbf{0}$, i.e., $\boldsymbol{C}$ is balanced; $\ker(\boldsymbol{C}-\boldsymbol{I}) = \operatorname{span}\{\mathbf{1}\}$ and the algebraic multiplicity of 0 is 1; all the other eigenvalues are negative, i.e., $\boldsymbol{C}-\boldsymbol{I}$ is negative semidefinite[3]. Writing the error dynamics we obtain

$$\dot{\boldsymbol{e}}_\phi = K_\phi(\boldsymbol{C}-\boldsymbol{I})\boldsymbol{e}_\phi + \Omega(\boldsymbol{C}-\boldsymbol{I})\mathbf{1} = K_\phi(\boldsymbol{C}-\boldsymbol{I})\boldsymbol{e}_\phi.$$

Hence, the error converges to its initial average, which is zero for any initial condition, since

$$\mathbf{1}^T\boldsymbol{e}_\phi(0) = \mathbf{1}^T(\boldsymbol{C}-\boldsymbol{I})\boldsymbol{\phi}^r(0) + \mathbf{1}^T\boldsymbol{b} = \mathbf{0}.$$

□

Proof of Proposition 6.2. Writing (6.4) for all the robots, and letting $f = 1/s$, we obtain

$$\dot{\boldsymbol{\phi}}^r = f\boldsymbol{\Delta}^r + K_\phi(\bar{\boldsymbol{\phi}}^r - \boldsymbol{\phi}^r).$$

The error dynamics in this case is

$$\dot{\boldsymbol{e}}_\phi = K_\phi(\boldsymbol{C}-\boldsymbol{I})\boldsymbol{e}_\phi + f(\boldsymbol{C}-\boldsymbol{I})(\boldsymbol{D}\boldsymbol{\phi}^r + \boldsymbol{g}).$$

[3]It is the Laplacian of the undirected ring with weights 1/2.

Since $\boldsymbol{C}-\boldsymbol{I}$ and $\boldsymbol{D}$ commute, we have

$$\dot{\boldsymbol{e}}_\phi = (K_\phi(\boldsymbol{C}-\boldsymbol{I}) + f\boldsymbol{D})\boldsymbol{e}_\phi + f((\boldsymbol{C}-\boldsymbol{I})\boldsymbol{g} - \boldsymbol{D}\boldsymbol{b}),$$

and being $(\boldsymbol{C}-\boldsymbol{I})\boldsymbol{g} - \boldsymbol{D}\boldsymbol{b} = \boldsymbol{0}$ we conclude that

$$\dot{\boldsymbol{e}}_\phi = (K_\phi(\boldsymbol{C}-\boldsymbol{I}) + f\boldsymbol{D})\boldsymbol{e}_\phi.$$

The matrix $K_\phi(\boldsymbol{C}-\boldsymbol{I}) + f\boldsymbol{D}$ has the same properties[4] of $K_\phi(\boldsymbol{C}-\boldsymbol{I})$ which was used to show the convergence of $\boldsymbol{e}_\phi$ to $\boldsymbol{0}$ in the Proof of Proposition 6.1. This implies also that $\hat{\phi}_i^r$ converges to $2\pi/n$ and $\dot{\phi}_i^r$ to $2\pi/ns$. □

For the proof of Proposition 6.3 we need a preliminary result.

Lemma 6.1. *Consider a $2n \times 2n$ matrix of the form*

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{0} & k_1\boldsymbol{I} \\ \boldsymbol{B} & k_2\boldsymbol{B} \end{pmatrix}$$

where $\boldsymbol{0}$ is the $n \times n$ null matrix, $\boldsymbol{I}$ is the $n \times n$ identity matrix, $\boldsymbol{B}$ is a $n \times n$ matrix, and k_1, k_2 are non-zero real numbers. For any eigenvalue μ of $\boldsymbol{B}$ associated to the eigenvector $\boldsymbol{u}$, the two roots of $\lambda^2 - k_2\mu\lambda - k_1\mu$, denoted with $\lambda_{1,2}$, are eigenvalues of $\boldsymbol{A}$ associated to the eigenvectors $\left(k_1\boldsymbol{u}^T \;\; \lambda_{1,2}\boldsymbol{u}^T\right)^T$.

Proof. A vector $\left(\boldsymbol{v_1}^T, \boldsymbol{v_2}^T\right)^T$ is an eigenvector of $\boldsymbol{A}$ associated to λ if

$$k_1\boldsymbol{v_2} = \lambda\boldsymbol{v_1} \tag{6.7}$$

$$\boldsymbol{B}\boldsymbol{v_1} + k_2\boldsymbol{B}\boldsymbol{v_2} = \lambda\boldsymbol{v_2}. \tag{6.8}$$

Hence, from (6.7), the eigenvectors have the structure $(k_1\boldsymbol{v} \;\; \lambda\boldsymbol{v})$. Letting $\boldsymbol{v} = \boldsymbol{u}$ in this structure, and substituting it into (6.8) we obtain $k_1\mu\boldsymbol{u} + k_2\lambda\mu\boldsymbol{u} = \lambda^2\boldsymbol{u}$, which establishes the Lemma. □

Proof of Proposition 6.3. Writing (6.5),(6.6) for all the robots we obtain

$$\dot{\boldsymbol{\Omega}}^r = K_\Omega(\bar{\boldsymbol{\phi}}^r - \boldsymbol{\phi}^r), \quad \boldsymbol{\Omega}_d(0) = \boldsymbol{0} \tag{6.9}$$

$$\dot{\boldsymbol{\phi}}^r = \boldsymbol{\Omega}^r + K_\phi(\bar{\boldsymbol{\phi}}^r - \boldsymbol{\phi}^r) + \boldsymbol{\xi}. \tag{6.10}$$

Let us consider the dynamics of the error $\boldsymbol{e} = (\boldsymbol{e}_\phi^T \; \boldsymbol{e}_\Omega^T)^T$

$$\dot{\boldsymbol{e}} = \begin{pmatrix} K_\Omega\boldsymbol{e}_\phi \\ (\boldsymbol{C}-\boldsymbol{I})(\boldsymbol{\Omega}^r + \boldsymbol{u}) + K_\phi(\boldsymbol{C}-\boldsymbol{I})\boldsymbol{e}_\phi \end{pmatrix} =$$

$$= \begin{pmatrix} \boldsymbol{0} & K_\Omega\boldsymbol{I} \\ \boldsymbol{C}-\boldsymbol{I} & K_\phi(\boldsymbol{C}-\boldsymbol{I}) \end{pmatrix} \begin{pmatrix} \boldsymbol{e}_\Omega \\ \boldsymbol{e}_\phi \end{pmatrix} = \tilde{\boldsymbol{A}}\boldsymbol{e},$$

[4] It is the Laplacian of the undirected ring with different weights.

where we have made use of the fact that $\bar{\xi} = (1/n)(\mathbf{1}^T\boldsymbol{\xi})$ and $(\boldsymbol{C}-\boldsymbol{I})\mathbf{1} = \mathbf{0}$. Recalling that $\boldsymbol{C}-\boldsymbol{I}$ is balanced, negative semi-definite, and noting that its smallest eigenvalue is -2, and applying the Lemma to $\tilde{\boldsymbol{A}}$ we can conclude that all the real parts of its eigenvalues have the same sign of the eigenvalues of $\boldsymbol{C}-\boldsymbol{I}$ (in fact, they are simply related by a 1/2 factor). Furthermore, the algebraic multiplicity of the eigenvalue 0 of $\tilde{\boldsymbol{A}}$ is 2, and its generalized eigenspace is generated by $\text{span}\{(\mathbf{1}^T\ \mathbf{0}^T)^T, (\mathbf{0}^T\ \mathbf{1}^T)^T\}$. Since, by construction, $\mathbf{1}^T\boldsymbol{e}_\Omega(0) = 0$ and $\mathbf{1}^T\boldsymbol{e}_\phi(0) = 0$, there is no evolution over this unstable eigenspace, which implies that $\boldsymbol{e}$ goes exponentially to zero. □

Note that Propositions 6.1, 6.2, and 6.3 imply that the reference phases are asymptotically in the same order as the initial reference phases. It can be proved that the same property actually holds along the *whole* duration of the trajectories of (6.3) and (6.4); the proof is lengthy and therefore omitted. Presently, this is only a (likely) conjecture for (6.5–6.6).

6.8 Conclusions

We have presented a distributed control method for encircling a target by means of a multi-robot system. The proposed scheme integrates a mutual localization module based on the developments in [26]. The theoretical proof of its effectiveness is supported by extensive experimental results.

Future work will be aimed at:

- proving that the reference trajectories never meet, so as to provide grounds for identifying a collision avoidance condition for robots of finite size;
- performing a theoretical analysys of a trajectory generation scheme based on *continuous replanning*, in which the actual robot coordinates (estimated through the mutual localization module) are used in place of their reference value (we already implemented such a variant with encouraging results);
- integrating a *consensus* mechanism among the robots on the results of the mutual localization, and especially the configuration of the target.

Conclusions

Acknowledgements

I would like to thank my advisor Prof. Giuseppe Oriolo, who provided assistance in numerous ways, and Prof. Francesco Bullo, who hosted me at UCSB. I am also grateful to my collaborator Paolo Stegagno, and to my co-authors (in alphabetic order) A. Censi, M. Di Rocco, J. Durham, L. Freda, L. Marchionni F. Pasqualetti, and M. Vendittelli.

Bibliography

[1] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration," *IEEE/ASME Trans. on Mechatronics*, vol. 14, no. 2, pp. 163–175, 2009.

[2] Y. Cao, A. Fukunaga, and A. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, no. 1, pp. 7–27, 1997.

[3] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, pp. 375–397, 1996.

[4] I. Rekletis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 7–40, 2001.

[5] D. Goldberg and M. J. Matarić, "Interference as a tool for designing and evaluating multi-robot controllers," in *Fourtheenth National Conference on Artificial Intelligence*, Providence, RI, Jul. 1997, pp. 637–642.

[6] T. Balch and R. Arkin, "Communication in reactive multiagent robotic systems," *Autonomous Robots*, vol. 1, no. 1, pp. 27–52, 1994.

[7] B. Yamauchi, "Frontier-based exploration using multiple robots," in *2nd Int. Conf. on Autonomous Agents*, Minneapolis, MN, May 1998, pp. 47–53.

[8] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. on Robotics and Automation*, vol. 1, no. 3, pp. 376–386, 2005.

[9] A. Howard, L. E. Parker, and G. S. Sukhatme, "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection," *International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 431–447, 2006.

[10] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, "A practical, decision-theoretic approach to multi-robot mapping and exploration," in *2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, Oct. 2003, pp. 3232–3238.

[11] A. Howard, M. J. Matarić, and G. S. Sukhatme, "An incremental deployment algorithm for mobile robot teams," in *2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, Oct. 2003, pp. 2849–2854.

[12] R. Zlot, A. Stenz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *2002 IEEE Int. Conf. on Robotics and Automation*, Washington, DC, May 2002, pp. 3016–3023.

[13] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Seventheenth National Conference on Artificial Intelligence*, Austin, TX, Jul. 2000, pp. 852–858.

[14] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "A randomized strategy for cooperative robot exploration," in *2007 IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 768–774.

[15] ——, "A decentralized strategy for cooperative robot exploration," in *ACM International Conference Proceeding Series, Proceedings of the 1st international conference on Robot communication and coordination*, vol. 318, no. 7, Athens, Greece, Oct. 2007.

[16] G. Oriolo, M. Vendittelli, L. Freda, and L. Troso, "The SRT method: Randomized strategies for exploration," in *2004 IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA, Apr. 2004, pp. 4688–4694.

[17] L. Freda and G. Oriolo, "Frontier-based probabilistic strategies for sensor-based exploration," in *2005 IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 3892–3898.

[18] J. C. Latombe, *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[19] R. Fagin, J. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning about Knowledge.* The MIT Press, 1995.

[20] J. Serra, *Image Analysis and Mathematical Morphology.* Academic Press, 1982.

[21] H. H. González-Baños and J. C. Latombe, "Navigation strategies for exploring indoor environments," *International Journal of Robotics Research*, vol. 21, no. 10, pp. 829–848, 2002.

[22] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* MIT Press, 2005.

[23] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1983.

[24] R. Diestel, *Graph Theory*, 2nd ed., ser. Graduate Texts in Mathematics. Springer, 2005, vol. 173.

[25] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink, *Mathematical Morphology and its Applications to Image and Signal Processing.* Kluwer Academic Publishers, 2000.

[26] A. Franchi, G. Oriolo, and P. Stegagno, "Mutual localization in a multi-robot system with anonymous relative position measures," in *2009 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, St. Louis, MO, Oct. 2009, pp. 3974–3980.

[27] T. Simeon, J. P. Laumond, and F. Lamiraux, "Move3d: A generic platform for path planning," in *4th Int. Symp. on Assembly and Task Planning*, Fukuoka, Japan, May 2001, pp. 25–30.

[28] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: Design, implementation, and experimental validation," *IEEE Trans. on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, 2002.

[29] A. Censi, L. Marchionni, and G. Oriolo, "Simultaneous maximum-likelihood calibration of robot and sensor parameters," in *2008 IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, May 2008, pp. 2098–2103.

[30] A. Censi, "An ICP variant using a point-to-line metric," in *2008 IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, May 2008, pp. 19–25.

[31] L. Freda, G. Oriolo, and F. Vecchioli, "Sensor-based exploration for general robotic systems," in *2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, France, Sep. 2008, pp. 2157–2164.

[32] J. Clark and R. Fierro, "Mobile robotic sensors for perimeter detection and tracking," *ISA Transactions*, vol. 46, no. 1, pp. 3–13, 2007.

[33] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. Mclain, S. M. Li, and R. Mehra, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *Int. Journal of Systems Sciences*, vol. 37, no. 6, pp. 351–360, 2006.

[34] S. Susca, S. Martínez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Trans. on Control Systems Technology*, vol. 16, no. 2, pp. 288–296, 2008.

[35] Y. Elmaliach, A. Shiloni, and G. A. Kaminka, "A realistic model of frequency-based multi-robot polyline patrolling," in *International Conference on Autonomous Agents*, Estoril, Portugal, May 2008, pp. 63–70.

[36] G. J. Woeginger, "Exact algorithms for NP-hard problems: A survey," in *Combinatorial Optimization – Eureka, You Shrink!*, ser. Lecture Notes in Computer Science, M. Jünger, G. Reinelt, and G. Rinaldi, Eds. Springer, 2003, vol. 2570, pp. 185–208.

[37] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *Journal of the Association for Computing Machinery*, vol. 32, no. 1, pp. 130–136, 1985.

[38] D. B. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of UAVs," *IEEE Trans. on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[39] A. Machado, G. Ramalho, J. D. Zucker, and A. Drogoul, "Multi-agent patrolling: An empirical analysis of alternative architectures," in *Multi-Agent-Based Simulation II*, ser. Lecture Notes in Computer Science, J. S. Sichman, F. Bousquet, and P. Davidsson, Eds. Springer, 2003, pp. 155–170.

[40] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," in *IEEE/WIC/ACM Int. Conf. Intelligent Agent Technology*, Beijing, China, Sep. 2004, pp. 302–308.

[41] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006, available at `http://planning.cs.uiuc.edu`.

[42] A. Ganguli, J. Cortes, and F. Bullo, "Distributed deployment of asynchronous guards in art galleries," in *2006 American Control Conference*, Minneapolis, MN, Jun. 2006, pp. 1416–1421.

[43] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009.

[44] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal on Computing*, vol. 21, no. 2, pp. 863–888, 1992.

[45] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," *International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[46] S. Sachs, S. Rajko, and S. M. LaValle, "Visibility-based pursuit-evasion in an unknown planar environment," *International Journal of Robotics Research*, vol. 23, no. 1, pp. 3–26, 2004.

[47] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *International Journal of Robotics Research*, vol. 28, no. 2, pp. 201–219, 2009.

[48] T. D. Parsons, "Pursuit-evasion in a graph," in *Theory and Applications of Graphs*, ser. Lecture Notes in Mathematics, Y. Alavi and D. Lick, Eds. Springer, 1978, vol. 642, pp. 426–441.

[49] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking, "Randomized pursuit-evasion in graphs," *Combinatorics, Probability and Computing*, vol. 12, no. 3, pp. 225–244, 2003.

[50] A. Kolling and S. Carpin, "Multi-robot surveillance: an improved algorithm for the graph-clear problem," in *2008 IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, May 2008, pp. 2360–2365.

[51] A. Howard, M. J. Matarić, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[52] L. J. Guibas, J.-C. Latombe, S. M. Lavalle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *International Journal of Computational Geometry & Applications*, vol. 9, no. 4/5, pp. 471–494, 1999.

[53] B. Gerkey and contributors, "The Player/Stage Project," `http://playerstage.sourceforge.net`, July 2009, version 2.13.

[54] A. Franchi and P. Stegagno, "Multirobot Integrated Platform," `http://www.dis.uniroma1.it/~labrob/software/MIP/`, Aug. 2009.

[55] A. Franchi, G. Oriolo, and P. Stegagno, "On the solvability of the mutual localization problem with anonymous position measures," in *2010 IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, May 2010, pp. 3193–3199.

[56] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur, "A theory of network localization," *IEEE Trans. on Mobile Computing*, vol. 5, no. 12, pp. 1663–1678, 2006.

[57] T. Eren, W. Whiteley, and P. Belhumeur, "Using angle of arrival (bearing) information in network localization," in *45th IEEE Conf. on Decision and Control*, San Diego, CA, Jan. 2006, pp. 4676–4681.

[58] G. Piovan, I. Shames, B. Fidan, F. Bullo, and B. Anderson, "On frame and orientation localization for relative sensing networks," in *47th IEEE Conf. on Decision and Control*, Cancun, Mexico, Dec. 2008, pp. 2326–2331.

[59] M. Batalin and G. S. Sukhatme, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *IEEE Trans. on Robotics*, vol. 23, no. 4, pp. 661–675, Aug 2007.

[60] R. Sepulchre, D. A. Paley, and N. E. Leonard, "Stabilization of planar collective motion: All-to-all communication," *IEEE Trans. on Automatic Control*, vol. 52, no. 5, pp. 811–824, May 2007.

[61] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The entrapment/escorting mission: An experimental study using a multirobot system," *IEEE Robotics & Automation Magazine*, vol. 15, no. 1, pp. 22–29, 2008.

[62] X. S. Zhou and S. I. Roumeliotis, "Robot-to-robot relative pose estimation from range measurements," *IEEE Trans. on Robotics*, vol. 24, no. 6, pp. 1379–1393, 2008.

[63] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[64] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using fastslam," in *2003 IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, Sep. 2003, pp. 1985–1991.

[65] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainly," *International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[66] W. Miller, *Symmetry Groups and Their Applications.* Academic Press, 1972.

[67] H. Zabrodsky, S. Peleg, and D. Avnir, "Symmetry as a continuous feature," *IEEE Trans. on Pattern Analysis & Machine Intelligence*, vol. 17, no. 12, pp. 1154–1166, 1995.

[68] 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, 1981.

[69] D. Fontanelli, L. Ricciato, and S. Soatto, "A fast ransac-based registration algorithm for accurate localization in unknown environments using lidar measurements," in *2007 IEEE Int. Conf. on Automation Science and Engineering*, 2007, pp. 597–602.

[70] A. Franchi, G. Oriolo, and P. Stegagno, "Mutual localization of a multi-robot team with anonymous relative position measures," Department of Computer and System Sciences, Tech. Rep. 1, January 2009. [Online]. Available: http://padis2.uniroma1.it:81/ojs/index.php/DIS_TechnicalReports/issue/view/157

[71] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[72] S. I. Roumeliotis and G. A. Bekey, "Distributed multirobot localization," *IEEE Trans. on Robotics*, vol. 18, no. 5, pp. 781–795, 2002.

[73] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "Collaborative multi-robot localization," in *23rd Annual German Conf. on Artificial Intelligence*, 1999, pp. 255–266.

[74] D. Fox, W. Burgard, H. Kruppa, and T. S., "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[75] A. Howard, M. Mataric, and G. Sukhatme, "Cooperative relative localization for mobile robot teams: an ego-centric approach," in *Naval Research Lab. Workshop on Multi-Robot Systems*, 2003, pp. 65–76.

[76] R. Grabowski, L. Navarro-Serment, C. Paredis, and P. Khosla, "Heterogeneous teams of modular robots for mapping and exploration," *Autonomous Robots*, vol. 8, no. 3, pp. 43–52, 2000.

[77] A. Martinelli and R. Siegwart, "Observability analysis for mobile robot localization," in *2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, Canada, Aug. 2005, pp. 1471–1476.

[78] X. Zhou and S. Roumeliotis, "Determining the robot-to-robot relative pose using range-only measurements," in *2007 IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 4025–4031.

[79] T. Eren, P. Belhumeur, and A. Morse, "Closing ranks in vehicle formations based on rigidity," in *41st IEEE Conf. on Decision and Control*, vol. 3, 2002, pp. 2959–2964.

[80] T. Eren, W. Whiteley, A. Morse, P. Belhumeur, and B. Anderson, "Sensor and network topologies of formations with direction, bearing, and angle information between agents," in *42nd IEEE Conf. on Decision and Control*, vol. 3, 2003, pp. 3064–3069.

[81] X. Zhou and S. Roumeliotis, "Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[82] H. Kato, K. Ishiguro and M. Barth, "Identifying and localizing robots in a multi-robot system environment," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 2, 1999, pp. 966–971.

[83] I. Cox, "A review of statistical data association techniques for motion correspondence," *Int. Journal of Computer Vision*, vol. 10, no. 1, pp. 53–66, 1993.

[84] A. Franchi, L. Freda, L. Marchionni, G. Oriolo, and M. Vendittelli, "Decentralized cooperative exploration: Implementation and experiments," in *Intelligent Autonomous Systems 10*, 2008, pp. 348–355.

[85] V. Gazi and K. M. Passino, "Stability analysis of social foraging swarms: combined effects of attractant/repellent profiles," in *41th IEEE Conf. on Decision and Control*, Las Vegas, NV, Dec. 2002, pp. 2848–2853.

[86] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *40th IEEE Conf. on Decision and Control*, Orlando, FL, Dec. 2001, pp. 2968–2973.

[87] R. Sepulchre, D. A. Paley, and N. E. Leonard, "Stabilization of planar collective motion with limited communication," *IEEE Trans. on Automatic Control*, vol. 53, no. 3, pp. 706–719, 2008.

[88] N. Moshtagh, N. Michael, A. Jadbabaie, and K. Daniilidis, "Vision-based, distributed control laws for motion coordination of nonholonomic robots," *IEEE Trans. on Robotics*, vol. 25, no. 4, pp. 851–860, 2009.

[89] N. Ceccarelli, M. Di Marco, A. Garulli, and A. Giannitrapani, "Collective circular motion of multi-vehicle systems," *Automatica*, vol. 44, no. 12, pp. 3025–3035, 2008.

[90] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Trans. on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[91] Z. Lin, B. Francis, and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *IEEE Trans. on Automatic Control*, vol. 50, no. 1, pp. 121–127, 2005.

[92] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *2003 American Control Conference*, Denver, CO, Jun. 2003, pp. 951–956.

[93] W. Ren, "Consensus seeking in multi-vehicle systems with a time-varying reference state," in *2007 American Control Conference*, New York, NY, Jul. 2007, pp. 717–722.

[94] ——, "Consensus strategies for cooperative control of vehicle formations," *IET Control Theory & Applications*, vol. 1, no. 2, pp. 505–512, 2007.

[95] G. Nitschke, "Co-evolution of cooperation in a pursuit evasion game," in *2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, Oct. 2003, pp. 2037–2042.

[96] S. D. Bopardikar, F. Bullo, and J. P. Hespanha, "A cooperative Homicidal Chauffeur game," *Automatica*, vol. 45, no. 7, pp. 1771–1777, 2009.

[97] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 5, pp. 662–669, 2002.

[98] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Trans. on Automatic Control*, vol. 53, no. 11, pp. 2480–2496, 2008.

[99] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, "Closed loop steering of unicycle like vehicles via Lyapunov techniques," *IEEE Robotics & Automation Magazine*, vol. 2, no. 1, pp. 27–35, 1995.

[100] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[101] T. Tay and W. Whiteley, "Generating isostatic frameworks," *Structural Topology*, vol. 11, no. 1, pp. 21–69, 1985.

[102] J. C. Maxwell, "On the calculation of the equilibrium and stiffness of frames," *Philosophical Magazine*, vol. 27, pp. 294–299, 1864.

[103] G. Laman, "On graphs and rigidity of plane skeletal structures," *Journal of Engineering Mathematics*, vol. 4, no. 4, pp. 331–340, 1970.

[104] B. Hendrickson, "Conditions for graph unique realizations," *SIAM Journal on Computing*, vol. 21, no. 1, pp. 65–84, 1992.

[105] B. Jackson and T. Jordan, "Connected rigidity matroids and unique realizations of graphs," *Journal of Combinatorial Theory, Series B*, vol. 94, no. 1, pp. 1–29, 2005.

[106] R. Connelly, "Generic global rigidity," *Discrete & Computational Geometry*, vol. 33, no. 4, pp. 549–563, 2005.

[107] B. Servatius and W. Whiteley, "Constraining plane configurations in cad: Combinatorics of directions and lengths," *SIAM J. Discrete Math*, vol. 12, no. 1, pp. 136–153, Jan 1999.

[108] T. Eren, "Using angle of arrival (bearing) information for localization in robot networks," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 15, no. 2, pp. 169–186, 2007.

Appendix A

A Classification of Ordinary Mutual Localization Problems

A.1 A classification of localization problems

In this section we define and classify the variety of localization problem for planar multi-robot systems that are present in literature. All those problems share a common goal: the estimation of one or more changes of coordinates between pairs of frames, i.e., the reconstruction of a set of poses from a set of measures. Sometimes the problem are focused on the estimation of a part of the whole change of coordinates, like the translation of the origin. This reduced versions are investigated, for example, in system of robots in which the orientation is not of interest. This reduced problem is a localization problem in which the relative orientation between the frames are known, as if all robots have a compass that make possible to share a common 'north'. Also, there are problems which want to estimate the unknown quantities up to planar transformations, like rotation, translation, dilation, et cetera.

A first classification emerge considering the behavior of the unknown quantities with respect to time. In the static case the unknown changes of coordinate are constant respect to the time. These are also called identification problems. In the dynamic case the unknown quantities are time varying.

Another distinction arise considering the motion capabilities of the robots, basically we divide the case the robot are still from the case the robot can move.

Furthermore, the localization problems may be classified according to

which are the known input quantities available to the estimation system. This inputs can be divided in a priori knowledge and measures. An a priori knowledge could be, for example, the position of an object, i.e, a rigid body, with respect to a certain frame. Those kind of objects are commonly called beacons.

The measures can be divided in two big classes. A first kind of measures are given by exteroceptive sensors, they are angles or distances between a pair of objects, one of which is the sensor. The second object could be another robot or a beacon. Interesting aspect concerning exteroceptive measures are (1) the kind of measurement provided by the sensors (distance, bearing, heading, et cetera) and (2) the relation between the physical state of the robotic system and the actual availability of the measures, represented by a graph called the measuring graph. More than one graphs may be used to describe case in which each robot carries more than one sensor. Those graphs could be directed or undirected depending on the problem assumptions.

Another kind of measures is provided by proprioceptive sensors, like the odometry, which is in turn an estimator producing the change of coordinates between any two mobile frames at different times of a moving robot. Usually if the robots are still the measures refers to time constant quantities while they refers to time varying quantities if the robot are moving.

The measures can be noiseless (they have the exact value of the measured quantity) or noisy, like a random variable, with the possibility of having false negatives and false positives (outliers).

Also, the measures can be ordered or anonymous. In the first case the correspondence between the measured quantity and the measure is known, while in the second case the measures comes without any given correspondence and they are a simple collection of unordered values.

The localization problems can also be classified with respect to the localizability property, a feature which depends on both the unknown quantities and measures. In particular there are problems requiring statical (istantaneous) localizability, i.e., problems in which the set of measures taken in account is rich enough to allow the problem solution without the robot motion. Also, there are problems requiring dynamical localizability, i.e., problems in which the set of measures taken in account is rich enough to allow the problem solution with the aid of the robot motion and the proprioceptive measures. Statical implies dynamical localizability but not vice versa.

Since the each measure is normally directly available only to a single robot, a further distinction can be made on the basis of the communication

model. The main communication aspects are: (1) which is the relationship between the communication topology, i.e, the communication graph, and the physical state of the system; (2) which kind of data can be transmitted among the robots (raw measures or local estimates); (3) the availability of the identity of the transmitter; (4) the reliability of the channel.

Finally, the problem can be divided in those requiring a decentralized versus a centralized solution. In the first case is in general assumed that the communication graph is not complete and the transmitted data are the local estimates.

A summary of the classification criteria follows, notice that many possibilities can coexists, and that the criteria are not completely independent:

1. Unknown quantity to estimate w.r.t. time:
   (a) static localization problems (identification),
   (b) time varying localization problems.
2. Robot motion capabilities:
   (a) still robots (canonic sensor network),
   (b) moving robots.
3. Kind of available inputs:
   (a) a priori knowledge (beacon positions),
   (b) measures:
      i. exteroceptive measures:
         A. referred to beacon,
         B. referred to robots.
      ii. proprioceptive measures (only if moving robots).
4. Kind of measures:
   (a) distances,
   (b) bearings,
   (c) orientations.
5. Measuring graphs:
   (a) fixed,
   (b) physically state dependent:

        i. isotropic (undirected graphs),
        ii. anisotropic (directed graphs).

6. Measured quantities w.r.t. time:
    (a) constant measured quantities,
    (b) time varying measured quantities (moving robots).
7. Certainty on measures:
    (a) noiseless measures,
    (b) noisy measures
        i. random variable
        ii. false positives
        iii. false negatives
8. Identity of measures:
    (a) ordered measures,
    (b) anonymous measures.
9. Requested localizability:
    (a) statical (istantaneous) localizability,
    (b) dynamical localizability.
10. Communication topology:
    (a) physical state dependent topology
    (b) fixed topology
11. Communication data used:
    (a) raw measures,
    (b) local estimate.
12. Communication identity:
    (a) know sender,
    (b) anonymous sender.
13. Communication reliability:
    (a) reliable channel,

(b) unreliable channel.

14. Requested solution:

    (a) centralized,

    (b) decentralized.

In the following we introduce some terminology to present the main class of problems in a formal way. First of all we always assume that $i, j \in \{1, \ldots, n\}$. We denote with $\mathcal{A}_i$ the i-th robot, with $\mathcal{F}$ the world frame, with with $\mathcal{F}_i$ and $\mathcal{M}_i(t)$, respectively, the fixed frame of $\mathcal{A}_i$ and mobile attached frame of $\mathcal{A}_i$ at time t. We indicate with ${}^{\mathcal{F}_j}T_{\mathcal{F}_i}$ the change of coordinates from $\mathcal{F}_i$ to $\mathcal{F}_j$, with ${}^{\mathcal{M}_j}T_{\mathcal{M}_i}(t)$ the change of coordinates from $\mathcal{M}_i(t)$ to $\mathcal{M}_j(t)$ and finally with ${}^{\mathcal{F}_j}T_{\mathcal{M}_i}(t)$ the change of coordinates from $\mathcal{M}_i(t)$ to $\mathcal{F}_j$.

We generically use the letter q with some indexes and apexes to denote the pose of a robot expressed in a certain frame, we use the words 'pose' and 'configuration' as synonyms. Using the same indexes and apexes, we split q using the letters p for the position, and θ for the angle. Similarly we split p using x and y to indicate the cartesian coordinates. Note that, instead of the cartesian coordinates, the polar coordinates $\rho = \sqrt{x^2 + y^2}$, and $\phi = \text{atan2}(y, x)$, may be also used. In particular, we denote with ${}^{\mathcal{F}}q_i(t)$ the pose of $\mathcal{A}_i$ at time t expressed in $\mathcal{F}$. Also, we denote with ${}^{\mathcal{F}_j}q_i(t)$ the pose of $\mathcal{A}_i$ at time t expressed in $\mathcal{F}_j$, and with ${}^{\mathcal{M}_j}q_i(t)$ the configuration of $\mathcal{A}_i$ at time t expressed in $\mathcal{M}_j(t)$. We assume that ${}^{\mathcal{M}_i}q_i(t)$ is known and equal to $(0\ 0\ 0)$, hence the information provided by ${}^{\mathcal{M}_j}q_i(t)$ is equivalent to ${}^{\mathcal{M}_j}T_{\mathcal{M}_i}(t)$, also, the information provided by ${}^{\mathcal{F}_j}q_i(t)$ is equivalent to ${}^{\mathcal{F}_j}T_{\mathcal{M}_i}(t)$.

The estimation of ${}^{\mathcal{F}_j}q_i(t)$ is simply called localization. If the knowledge of part of ${}^{\mathcal{M}_j}q_i(t)$ is used to improve the estimation, then is called cooperative localization (CL). The estimation of ${}^{\mathcal{F}_j}T_{\mathcal{F}_i}$ is called absolute mutual localization (AML), since the quantity to estimate is constant with respect the time, it is an identification problem. The estimation of ${}^{\mathcal{M}_j}T_{\mathcal{M}_i}(t)$ is called relative mutual localization (RML).

The exteroceptive measures considered in mutual localization problems are always measures of quantities referred to the mobile attached frames. With respect to a pair of robots $\mathcal{A}_i$ and $\mathcal{A}_i$ the most considered quantities are: the distance $d_{ij}(t)$ between the origin of $\mathcal{M}_j$ and the origin of $\mathcal{M}_i$ at time t; the angle difference θ_{ij} between of the x axis of $\mathcal{M}_j$ with respect to the x axis of $\mathcal{M}_i$ at time t; the bearing angle $\phi_{ij}(t)$ of the origin of $\mathcal{M}_j$ with respect to the x axis of $\mathcal{M}_i$, the bearing angle $\phi_{ji}(t)$ of the origin of $\mathcal{M}_i$ with respect to the x axis of $\mathcal{M}_j$.

A typical choice is to consider $\mathcal{F}_i = \mathcal{M}_i(t_0)$.

A.2 Localization problems requiring instantaneous localizability

This kind of problems aim at reconstructing the unknown quantities at time t using only a snapshot of the measures at time t. For this reason the time evolution of the system is not important, and the problem can be solved considering the robots still. On the other hand, the problems requiring dynamical localizability can be considered of this class if the time is discretized and we consider simultaneously all measures taken by the same robot at different times. Depending on the measured quantities the unknown quantities can be at most reconstructed up to certain transformations, like translations, congruences, homotheties. Also, in this section the measures are considered noiseless.

A.2.1 Position localization with distance information

This problem is commonly referred as "network localization problem with distance information", and it is stated in [56]. It deals with the reconstruction of the robot positions knowing some of the inter-distances between them and the position of some robots acting as beacons. Since the frames to be estimated are attached to the robots it is an RML, but since it applies to still robots it could be also considered an AML problem. It is a static localization problem, in which the orientation of the frames is not of interest. In fact, only the positions of the robots are considered. This model matches both with problems in which the robots are omnidirectional and with problems in which the orientation is already known.

Since it is requested the statical (or instantaneous) localizability of the system, the robot motion is not of interest, and the robots are w.l.o.g. considered still. The presence of beacons is admitted, the exteroceptive measures are only the distances between robots, and the odometry is not of interest. The measuring graph is fixed and undirected, and the measured quantities are constant w.r.t. time. The measures are noiseless and ordered. The communication aspects are not addressed and the requested solution is centralized.

First we describe the problem, second we give some definition that are related to the problem and some results on the solvability.

Problem A.1. *We are given m beacon positions $p_1, \ldots, p_m$ in $\mathbb{R}^d$ ($d \in \{2, 3\}$). We denote with $p_{m+1}, \ldots, p_n$ the unknown positions of $n - m$ robots. In addition we are given the distance $\|p_i - p_j\|$ for every couple (i, j) belonging to a certain link set $L \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$. Find the*

positions $\pi_1, \ldots, \pi_N$ *satisfying* $\|\pi_j - \pi_i\| = \|p_j - p_i\|$, $\forall (i,j) \in L$, *subject to the constraint that* $\pi_k = p_k$ *for* $k \in \{1, \ldots, m\}$. *(Notice that, by default,* $\{1, \ldots, m\} \times \{1, \ldots, m\} \subset L$.*)*

Problem A.2. *Consider the same setup of the Prob. A.1 without beacons, i.e., with* $m = 0$. *Find the positions* $\pi_1, \ldots, \pi_n$, *up to a congruence.*[1]

The Prob. A.2 emerge from the observation that if we apply an congruence to a vector of positions p, then all the distances remain the same. Hence a complete knowledge of all the distances between the robot positions, without any beacons, allows the reconstruction of the robot positions only up to a congruence.

Remark It is clear that the the solvability of the Prob. A.1 and A.2 requires an instantaneous localizability property of the system.

Definition A.1 (Global rigidity)**.** *A vector of* n *positions* $(p_1 \ldots p_n)$ *and an undirected graph* $G = (\{1, \ldots, n\}, L)$ *are globally rigid if it is possible to reconstruct, up to a congruence, the unknown robot positions* $p_1, \ldots, p_n$, *knowing every distance* $\|p_i - p_j\|$ *for every couple* (i,j) *belonging to* L.

The following proposition is straightforward:

Proposition A.1. *The Prob. A.1 is solvable if and only if there are at least* $d+1$ *beacons in general position,*[2] *and the positions* $p_1, \ldots, p_n$ *with the graph* $G = (\{1, \ldots, n\}, L)$ *are globally rigid. The Prob. A.2 is solvable if and only if the positions* $p_1, \ldots, p_n$ *with the graph* $G = (\{1, \ldots, n\}, L)$ *are globally rigid.*

Hence, the solvability of the Prob. A.1 and A.2 depends on the global rigidity of the pair positions-measuring graph. Instead of characterizing the globally rigid pairs of vectors of positions and graphs, we want characterize the measuring graphs that are rigid with (almost) all the vectors of positions. This kind of properties are called 'generic', and can be usually checked with a combinatorial test on the graph.

Definition A.2 (Generic global rigidity)**.** *An undirected graph* $G = (\{1, \ldots, n\}, L)$ *is generically globally rigid if, for all* p *in (an open dense subset of)* $\mathbb{R}^{dn}$, p *and* G *are globally rigid.*

To understand the necessary and sufficient condition for the generic global rigidity (the Theorem A.3) we need to define the weaker concepts

[1] A distance preserving map.

[2] Not aligned if $d = 2$, or not on the same plane if $d = 3$

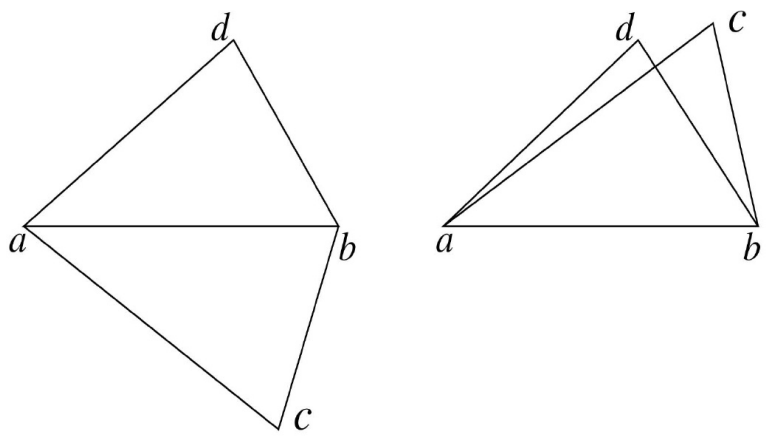

Figure A.1: A vector of positions and links which are rigid but not globally rigid. In fact, there are two isolated vectors of positions with the same links. From [56].

of infinitesimal rigidity and generic infinitesimal rigidity. We define also the rigidity concept to have a more complete view of the argument. We denote with k the number of measured links, i.e., cardinality of L. Consider the function $\delta_L : \mathbb{R}^{nd} \to \mathbb{R}^k$ mapping the vector of n positions p, to the vector of k measured distances, i.e., the function defined by:

$$\delta_{ij}^2(p) = (p_i - p_j)^T(p_i - p_j) \quad \forall (i,j) \in L, \tag{A.1}$$

The set $\delta_L^{-1}(\delta_L(p))$ is a smooth manifold, denoted with $\mathcal{D}_L(p)$, and is formed by all the vectors $q \in \mathbb{R}^{dn}$ satisfying the constraint equations:

$$(q_i - q_j)^T(q_i - q_j) = (p_i - p_j)^T(p_i - p_j) \quad \forall (i,j) \in L. \tag{A.2}$$

To check the global rigidity we want to know whether $\mathcal{D}_L(p)$ contains only points congruent with p. To factor out the these equivalent points, we take the quotient of $\mathcal{D}_L(p)$ by the group Γ of congruences. We present the following two definitions only for completeness.

Definition A.3 (Rigidity)**.** *A vector of n positions p and an undirected graph $G = (\{1, \ldots, n\}, L)$ are rigid if the vector p is isolated in the quotient topology $\mathcal{D}_L(p)/\Gamma$.*

Definition A.4 (Generic rigidity)**.** *An undirected graph $G = (\{1, \ldots, n\}, L)$ is generically rigid if G with all p in (an open dense subset of) $\mathbb{R}^{dn}$ are rigid.*

Rigidity is a difficult property to establish. For this reason the tangent space of $\mathcal{D}_L(p)$ is studied instead of $\mathcal{D}_L(p)/\Gamma$. Taking the derivative of the Eq. (A.2) we have the corresponding linear equations in the tangent space:

$$(p_i - p_j)^T(\dot{p}_i - \dot{p}_j) = 0 \quad \forall (i,j) \in L \tag{A.3}$$

The Eq. A.3 means that the difference between the velocity of two linked positions must be 'perpendicular' to the link. The Equations A.3 can be written in a matrix equation:

$$R_L(p)\dot{p} = 0, \tag{A.4}$$

where $R_L(p)$ is the $k \times nd$ Jacobian matrix of δ_L evaluated in p and is called rigidity matrix. Since the tangent vectors to the congruences must satisfy Eq. A.4, the rank of $R_L(p)$ must be less or equal $2n-3$, if $d=2$, or $3n-6$, if $d=3$.

Definition A.5 (Infinitesimal rigidity)**.** *A vector of n positions p and an undirected graph $G = (\{1,\dots,n\}, L)$ are infinitesimally (or first-order) rigid if the rank of the rigidity matrix $R_L(p)$ is $2n-3$, if $d=2$, or $3n-6$, if $d=3$.*

It is well known that infinitesimal rigidity implies rigidity, but not global rigidity, as depicted in Fig. A.1. The following definition introduces the generic property associated to infinitesimal rigidity.

Definition A.6 (Generic (infinitesimal) rigidity)**.** *An undirected graph $G = (\{1,\dots,n\}, L)$ is generically (infinitesimally) rigid if the rank of the rigidity matrix $R_L(p)$ is $2n-3$, if $d=2$, or $3n-6$, if $d=3$, for all p in (an open dense subset of) $\mathbb{R}^{dn}$.*

Theorem A.1 (Tay, Whiteley [101])**.** *Generic infinitesimal rigidity and generic rigidity are equivalent.*

The following theorem characterize the generically rigid graphs for $d=2$, no result is known for $d \geq 3$.

Theorem A.2 (Maxwell, Laman [102, 103])**.** *An undirected graph $G = (\{1,\dots,n\}, L)$ is generically (infinitesimally) rigid in $\mathbb{R}^2$ if and only if L contains a subset E consisting of $2n-3$ edges with the property that, for any nonempty subset $E' \subset E$, the number of edges in E' cannot exceed $2n'-3$, where n' is the number of vertices of G which are endpoints of edges in E'.*

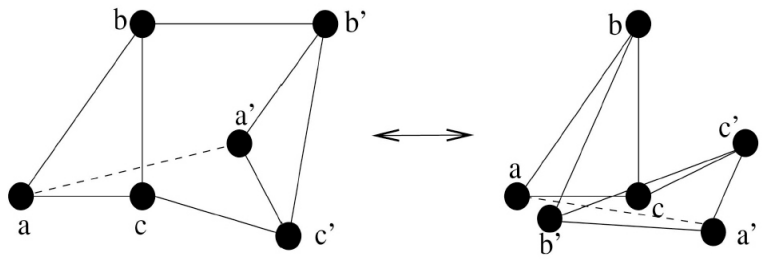

Figure A.2: If the link (a, a') is removed the graph loose the 3-connectedness and the generic global rigidity. From [56].

To conclude this section we give a theorem that completely characterize the generically globally rigid graphs for $d = 2$.

Theorem A.3 (Hendrickson, Jackson, Jordan, [104, 105])**.** *A graph G with more than 3 vertices is generically globally rigid in $\mathbb{R}^2$ if and only if it is 3-connected and the removal of any single edge results in a graph that is also generically (infinitesimally) rigid in $\mathbb{R}^2$.*

For $d \geq 3$, Thm. A.3 extends only as a necessary but not sufficient condition. Given d, we say that a graph G is redundantly rigid if the removal of any single edge results in a graph that is also generically (infinitesimally) rigid in $\mathbb{R}^d$

Theorem A.4 (Hendrickson, Connelly [104, 106])**.** *If a graph G with more than $d+1$ vertices is generically globally rigid in d-space, then G is at least $d+1$ connected and redundantly rigid. In all dimensions $d \geq 3$, there are redundantly rigid and at least $d+1$ connected graphs that are not generically globally rigid.*

A.2.2 Network localization problem with bearing information (and known heading)

This problem is defined in [107, 57, 108], and it refers to the planar case. As in the distance case, first we describe the problem, second we give some definition that are related to the problem and some results on the solvability.

Problem A.3. *We are given m beacon positions $p_1, \ldots, p_m$ in $\mathbb{R}^2$ and a unit vector e_x representing a common heading of $n - m$ robots. We denote with $p_{m+1}, \ldots, p_n$ the unknown positions of the robots. In addition we are given the angles $\angle(p_j - p_j, e_x)$ for every couple (i, j) belonging to*

a certain link set $B \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$ *(notice that, by default,* $\{1, \ldots, m\} \times \{1, \ldots, m\} \subset B$*). We want to exactly reconstruct the unknown robot positions* $p_{m+1}, \ldots, p_n$*.*

Notice that we can consider the set of links B symmetric, in fact, since $\angle(p_i - p_j, e_x) = \pi + \angle(p_j - p_i, e_x)$, the knowledge of $\angle(p_j - p_j, e_i)$ implies the knowledge of $\angle(p_i - p_i, e_x)$.

Problem A.4. *Consider the same setup of the Prob. A.3 without beacons, i.e., with* $m = 0$*. We want to reconstruct, up to a dilation,*[3] *the unknown robot positions* $p_1, \ldots, p_n$*.*

The Prob. A.2 emerge from the observation that if we apply an dilation to a vector of positions p, then all the bearings remain the same. Hence a complete knowledge of all the bearing between the robot positions, without any beacons, allows the reconstruction of the robot positions only up to a dilation.

Remark It is clear that the the solvability of the Prob. A.3 and A.4 requires an instantaneous localizability property of the system.

Definition A.7 (Global parallel rigidity)**.** *A vector of* n *positions* $(p_1 \ \ldots \ p_n)$ *and an undirected graph* $G = (\{1, \ldots, n\}, B)$ *are globally parallel rigid if it is possible to reconstruct, up to a dilation, the unknown robot positions* $p_1, \ldots, p_n$*, knowing every bearing* $\angle(p_j - p_i, e_x)$*, respect to a common heading* e_x*, for every couple* (i, j) *belonging to* B*.*

The Prob. A.3 is solvable if and only if there are at least 2 beacons in general position,[4] and the positions $p_1, \ldots, p_n$ with the graph $G = (\{1, \ldots, n\}, B)$ are globally parallel rigid. The Prob. A.4 is solvable if and only if the positions $p_1, \ldots, p_n$ with the graph $G = (\{1, \ldots, n\}, B)$ are globally parallel rigid. Hence, the solvability of the Prob. A.3 and A.4 depends on the global parallel rigidity of the pair positions-measuring graph. Instead of characterizing the globally parallel rigid pairs of vectors of positions and graphs, we want characterize the measuring graphs that are parallel rigid with (almost) all the vectors of positions. This kind of properties are called 'generic', and can be usually checked with a combinatorial test on the graph.

Definition A.8 (Generic global parallel rigidity)**.** *An undirected graph* $G = (\{1, \ldots, n\}, B)$ *is generically globally parallel rigid if, for all* p *in (an open dense subset of)* $\mathbb{R}^{dn}$*,* p *and* G *are globally parallel rigid.*

[3] A parallel line preserving map, i.e. a similarity, i.e., a translation plus an homothety.
[4] Not coincident.

We denote with k the number of measured links, i.e., cardinality of B. Consider the function $\phi_B : \mathbb{R}^{nd} \to]-\pi, \pi]^k$ mapping the vector of n positions p, to the vector of k measured bearings, i.e., the function defined by:

$$\phi_{ij}(p) = \angle(p_j - p_i, e_x) \quad \forall (i,j) \in B, \tag{A.5}$$

The set $\phi_B^{-1}(\phi_B(p))$ is a smooth manifold, denoted with $\mathcal{A}_L(p)$, and is formed by all the vectors $q \in \mathbb{R}^{dn}$ satisfying the constraint equations:

$$(p_j - p_j)^{\perp^T}(q_i - q_j) = 0 \quad \forall (i,j) \in B. \tag{A.6}$$

which force, for each couple $(i,j) \in B$, the vector $q_i - q_j$ to have the same direction of the vector $p_i - p_j$.

Taking the derivative of the Eq. (A.6) we have the corresponding linear equations in the tangent space:

$$(p_i - p_j)^{\perp^T}(\dot{q}_j - \dot{q}_i) = 0 \quad \forall (i,j) \in B \tag{A.7}$$

The Eq. A.7 means that the difference between the velocity of two linked positions must be 'parallel' to the link. The Equations A.7 can be written in a matrix equation:

$$R_B(p)\dot{q} = 0, \tag{A.8}$$

where $R_B(p)$ is called parallel rigidity matrix. Note that the Eq. A.6 also can be written with the same matrix equation:

$$R_B(p)q = 0, \tag{A.9}$$

Since the tangent vectors to the homotheties must satisfy Eq. A.8, the rank of $R_B(p)$ must be less or equal $2n - 3$.[5]

Definition A.9 ((Infinitesimal) parallel rigidity)**.** *A vector of n positions p and an undirected graph $G = (\{1, \ldots, n\}, B)$ are (infinitesimally) parallel rigid if the rank of the parallel rigidity matrix $R_B(p)$ is $2n - 3$.*

Definition A.10 (Generic (infinitesimal) parallel rigidity)**.** *An undirected graph $G = (\{1, \ldots, n\}, B)$ is generically (infinitesimally) parallel rigid if the rank of the parallel rigidity matrix $R_B(p)$ is $2n - 3$ for all p in (an open dense subset of) $\mathbb{R}^{dn}$.*

Since the this rank condition is equivalent to global parallel rigidity, in opposition with the distance case, the global concept of rigidity coincides with the infinitesimal one.

[5] A dilation, is like a congruence with an dilation instead of a rotation.

Duality Any statement for the localization problem with distances can be given for the same problem with bearing where distances are switched with bearing. This process preserves the solution space (just turning the solution by $\pi/2$).

Theorem A.5. *An undirected graph $G = (\{1, \ldots, n\}, B)$ is generically (infinitesimally) rigid in $\mathbb{R}^2$ if and only if B contains a subset B consisting of $2n - 3$ edges with the property that, for any nonempty subset $B' \subset B$, the number of edges in E' cannot exceed $2n' - 3$, where n' is the number of vertices of G which are endpoints of edges in E'.*

A.2.3 Network localization problem with distance and bearing information (and known heading)

This problem is defined in [107, 57, 108], and it refers to the planar case.

Problem A.5. *We are given m beacon positions $p_1, \ldots, p_m$ in $\mathbb{R}^2$ and a unit vector e_x representing a common heading of $n - m$ robots. We denote with $p_{m+1}, \ldots, p_n$ the unknown positions of the robots. In addition we are given the distances $|p_j - p_j|$ for every couple (i, j) belonging to a certain distance-link set $L \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$, and the angles $\angle(p_j - p_j, e_x)$ for every couple (i, j) belonging to a certain bearing-link set $B \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$ (notice that, by default, $\{1, \ldots, m\} \times \{1, \ldots, m\} \subset L$ and $\{1, \ldots, m\} \times \{1, \ldots, m\} \subset B$). We want to exactly reconstruct the unknown robot positions $p_{m+1}, \ldots, p_n$.*

Problem A.6. *Consider the same setup of the Prob. A.3 without beacons, i.e., with $m = 0$. We want to reconstruct, up to a translation, the unknown robot positions $p_1, \ldots, p_n$.*

The Prob. A.6 emerge from the observation that if we apply an translation to a vector of positions p, then all the distances and the bearings remain the same. Hence a complete knowledge of all the distances and the bearings between the robot positions, without any beacons, allows the reconstruction of the robot positions only up to a translation.

Similar definition for rigidity up to a translation.

Theorem A.6. *A graph $G = (\{1, \ldots, n\}, L, B)$ is generically rigid up to a translation in 2-space if and only if the following conditions hold:*

1. $|L \cup B| = 2n - 2$
2. *for all subsets V' of vertices:* $|L' \cup B'| \leq 2n' - 2$

3. *for all subsets V' of at least two vertices:* $|B'| \leq 2n' - 3$
4. *for all subsets V' of at least two vertices:* $|L'| \leq 2n' - 3$

No results on global rigidity. The previous hold if there is only a distance measure. And there is also a conjecture (Conjecture 3.1).

A.2.4 Network localization problem with bearing information (unknown heading)

The problem is defined in [108] but setup is not clear.

Problem A.7. *We are given m beacon positions $p_1, \ldots, p_m$ in $\mathbb{R}^2$. We denote with $p_{m+1}, \ldots, p_n$ the unknown positions of the robots, and with $e_1, \ldots, e_n$ the unknown headings of the beacons and the robots. We are given the bearings $\angle(p_j - p_j, e_i)$ for every couple (i, j) belonging to a certain link set $B \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$. We want to exactly reconstruct the unknown robot positions $p_{m+1}, \ldots, p_n$.*

Notice that in this case the set B does not implicitly contains the set $\{1, \ldots, m\} \times \{1, \ldots, m\}$ because the headings of the beacons are not known. Furthermore, notice that we can not consider the set of links B symmetric, as for problems A.1, A.2, A.3, and A.4. In fact, since $\angle(p_i - p_j, e_j) = \angle(p_j - p_j, e_i) - \angle(e_j, e_i) + \pi$, but we do not know $\angle(e_j, e_i)$, the knowledge of $\angle(p_j - p_j, e_i)$ does not imply the knowledge of $\angle(p_i - p_j, e_j)$.

Problem A.8. *Consider the same setup of the Prob. A.7 without beacons, i.e., with $m = 0$. We want to reconstruct, up to a dilation and a rotation,*[6] *the unknown robot positions $p_1, \ldots, p_n$.*

The Prob. A.2 emerge from the observation that if we apply an dilation (translation plus homothety) to a vector of positions p or the same rotation to all the headings $e_1, \ldots, e_n$, and all the positions $p_1, \ldots, p_n$, then all the bearings remain the same. Hence a complete knowledge of all the bearings between the robot positions, without any beacons, allows the reconstruction of the robot positions only up to a dilation and a rotation. Notice that the reflections are excluded, since a reflection swap the signs of bearings.

Definition A.11 (Global parallel directed rigidity)**.** *A vector of n positions $(p_1 \ldots p_n)$ and an directed graph $G = (\{1, \ldots, n\}, B)$ are globally parallel directed rigid if it is possible to reconstruct, up to a dilation and a rotation, the unknown robot positions $p_1, \ldots, p_n$, knowing every bearing $\angle(p_j - p_i, e_i)$, respect to a set of unknown headings $e_1, \ldots, e_n$, for every couple (i, j) belonging to B.*

[6] A translation plus an homothety plus a rotation.

Definition A.12 (Generic parallel directed rigidity)**.** *A directed graph* $G = (\{1, \ldots, n\}, B)$ *is generically parallel directed rigid in* $\mathbb{R}^2$ *if is parallel directed rigid with all* p *in (an open dense subset of)* $\mathbb{R}^{2n}$*.*

Theorem A.7. *(Not sure) An directed graph* $G = (\{1, \ldots, n\}, B)$ *is Generic parallel directed rigidity in* $\mathbb{R}^2$ *if and only if the following holds:*

1. $\forall i \in \{1, \ldots, n\}$, $\exists j, k$ *with* $j \neq k$ *such that* $(k, i) \in B$ *and* $(j, i) \in B$
2. *the underlying undirected link set of* B *contains a subset* B' *consisting of* $2n-3$ *edges with the for any nonempty subset* $B'' \subset B'$*, the number of edges in* E'' *cannot exceed* $2n''-3$*, where* n'' *is the number of vertices of* G *which are endpoints of edges in* E''*.*

A.2.5 Orientation and Frame network localization problems with undirected bearing information

These problems are defined in [58].

Problem A.9 (Heading)**.** *We are given a beacon position* p_1 *in* $\mathbb{R}^2$ *and its heading* e_1*. We denote with* $p_2, \ldots, p_n$ *the unknown positions of the robots, and with* $e_2, \ldots, e_n$ *the unknown headings. In addition we are given the bearings* $\angle(p_j - p_j, e_i)$ *for every couple* (i, j) *belonging to a certain symmetric*[7] *link set* $B \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$*. We want to exactly reconstruct the unknown headings* $e_2, \ldots, e_n$*.*

Problem A.10. *Consider the same setup of the Prob. A.9 without beacon, i.e., with unknown* p_1 *and* e_1*. We want to reconstruct, up to a rotation, the unknown robot headings* $e_1, \ldots, e_n$*.*

The Prob. A.10 emerge from the observation that if we apply rotation to all the positions $p_1, \ldots, p_n$ and all the headings $e_1, \ldots, e_n$, then all the bearings remain the same. Hence a complete knowledge of all the bearings between the robot positions, without any beacons, allows the reconstruction of the robot headings only up to a rotation.

The following result is straightforward [58]:

Proposition A.2. *The Problems A.9 and A.10 are solvable if and only if the graph* $G = (\{1, \ldots, n\}, B)$ *is connected.*

Since the knowledge of all the headings is equivalent to the knowledge of a common heading, problems A.5 and A.9 can be merged into the following.

[7] $(i, j) \in B \Leftrightarrow (j, i) \in B$.

Problem A.11 (Frame)**.** *We are given a beacon position p_1 in $\mathbb{R}^2$ and its heading e_1. We denote with $p_2, \ldots, p_n$ the unknown positions of the robots, and with $e_2, \ldots, e_n$ the unknown headings. In addition we are given the distances $|p_i - p_j|$ for every couple (i, j) belonging to a certain distance-link set $L \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$, and the angles $\angle(p_j - p_j, e_x)$ for every couple (i, j) belonging to a certain symmetric bearing-link set $B \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$ We want to exactly reconstruct the unknown headings $e_2, \ldots, e_n$ and positions $p_2, \ldots, p_n$.*

Notice that [58] the Prob.A.11 is not solvable if $B = \emptyset$. In fact, no heading information can be retrieved from the distance measurements. In addition, the following result is also straightforward [58]:

Proposition A.3. *If the graph $G = (\{1, \ldots, n\}, B)$ is generically rigid and $|L| \geq 1$ then the Prob. A.11 is solvable.*

Appendix B

Multi-robot Integrated Platform

We have developed the Multi-robot Integrated Platform (MIP), a C++ software aimed to develop control and estimation robotics algorithms. A good level of modularity, the use of abstracted low-level robot interfaces and the fact that a different instance of the same MIP executable controls each robot guaranties software reusability and easiness of embedding .

MIP provides both an inter-robot and an intra-robot IP-based communication module. They are first mandatory for multi-robot applications and for splitted istantiations of the robot control process.

MIP has the following Components :

Baselib basic library for general purpuse and robotics functionalities, e.g., pose, laser scan, IP communication, class serialization, multithreading, file managment, user option managment, etc...

Algorithms class collection of robotics algorithms, e.g., geometric and sensor data processing (Voronoi diagrams, feature extraction,...), estimate (Kalman filetring, particle filtering,...), control (trajectory control, obstacle avoidance,...).

Resources classes derived from the Resource class providing interface modules respect to the hardware or the MIP platform facilities (motors, sensors, communication modules, keyboard, logging/tracing, 2D/3D display,...).

Tasks classes derived from the Task class which actualy perform the robot activities that must be execute in parallel, glueing algorithms and resources. Example of activities are: tracking, deployment, target navi-

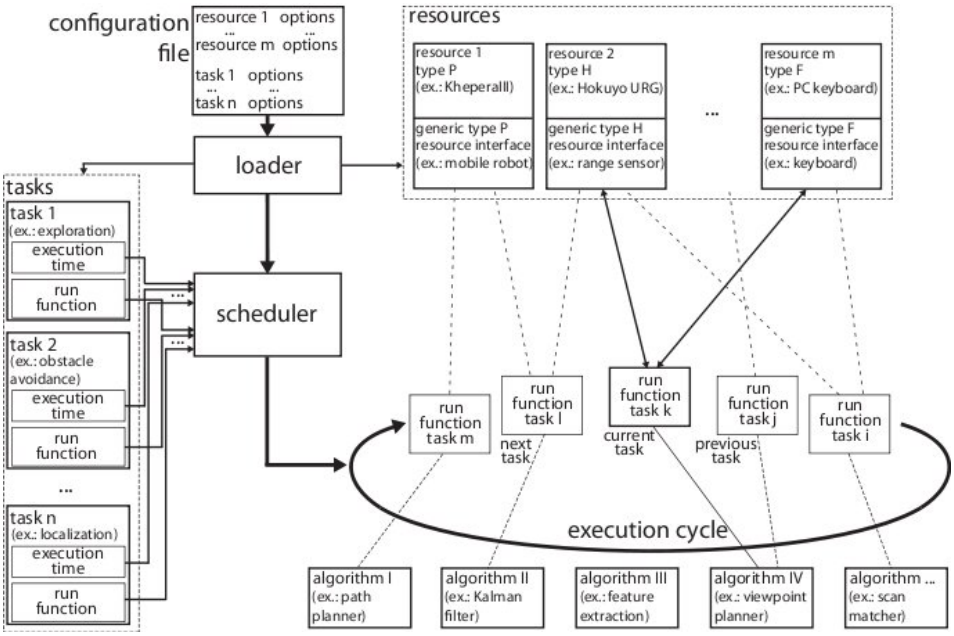

Figure B.1: A block scheme of MIP.

gation, mutual localization, entrapment, exploration,... Each task is a finite state machine that uses the algorithms and the resources to gain its objective. A task can benefit of outputs or provide inputs from/to other concurrent scheduled tasks. The inter-task data exchange pass through the resources, for this reason the resources are also a shared memory for tasks.

Main main of the program. Here is created and launched the Scheduler. The scheduler executes ciclically a list of task, checking the timing correctness and managing the frequency of execution, as requested from every task. The Scheduler is not preemptive.

In general a MIP program acts, as depicted in Fig. B.1,in the following way:

1. The scheduler instantiates resources needed by the tasks (specified in a configuration file).
2. The scheduler executes cyclically the tasks.
3. Some task gets sensorial and communication data provided by the resources, processes them by mean the algorithms.

4. Some task uses the resources to interact with the uman operator , e.g.,a visual feedback or a keyboard input.
5. Some task executes the control law sending commands to the resources, e.g, to the motor module.

It has been used, among the others, for the following works:

- Mapping
- ScanMatching
- Mutual Localization
- Joystick
- Deployment
- Line Following
- Entrapment
- SRG Demo
- Particle Filter Mutual Localization
- Tracciatore
- Quadrotor
- Goal-based navigation

For a complete documentation refer to [54].

Rome, Italy
December 8, 2010.