2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

A Randomized Strategy for Cooperative Robot Exploration

Antonio Franchi Luigi Freda Giuseppe Oriolo Marilena Vendittelli
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Eudossiana 18, 00184 Roma, Italy
{franchi,freda,oriolo,venditt}@dis.uniroma1.it

Abstract— **We present a cooperative exploration strategy for mobile robots. The method is based on the randomized incremental generation of a collection of data structures called Sensor-based Random Trees, each representing a roadmap of an explored area with an associated safe region. Decentralized cooperation and coordination mechanisms are introduced so as to improve the exploration efficiency and to avoid conflicts. Simulations in various environments are presented to show the performance of the proposed technique.**

I. INTRODUCTION

Exploration is the basic task by which a mobile robot covers an unknown area, typically learning a model of the environment at the same time. Possible applications include automated surveillance, search-and-rescue operations in hostile areas, map building and planetary missions.

The use of a multi-robot system brings in general many advantages [1], [2]. In exploration, it aims at significantly reducing the time required to complete the task. If a map is to be acquired, the redundant information provided by multiple robots can be also used to increase the final map accuracy and the quality of the localization [3]. In order to achieve these objectives, some sort of task decomposition and allocation are required. In practice, strategies to conveniently distribute robots over the environment should be accurately devised in order to reduce the occurrence of spatial conflicts [4] and actually reap the benefits of a multi-robot architecture. Clearly, communication plays a crucial role in achieving a cooperative behavior with improved performance [5].

In most exploration strategies, the boundary between known and unknown territory (the *frontier*) is approached in order to maximize the information gain. For the multi-robot case, a pioneering work in this direction is [6]: the robots merge the acquired information in a global gridmap of the environment, from which the frontier is extracted and used to plan the individual robot motions. While this basic scheme lacks an arbitration mechanism preventing robots from approaching the same frontier region, in [7] 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. In [8], the utility of a particular frontier region from the viewpoint of relative robot localization (and hence, of the accuracy of map merging) is also considered. In the incremental deployment algorithm of [9], 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 [10], whereas [11] proposes a centralized approach which uses a frontier-based search and a bidding protocol assign frontier targets to the robots.

This paper presents a randomized strategy for cooperative exploration which is the outgrowth of the SRT method, designed for a single robot and presented in [12], [13]. The basic tool therein is the Sensor-based Random Tree (SRT), a compact data structure representing a roadmap of the explored area, which can be seen as a sensor-based version of the RRT concept [14]. In particular, each node of an SRT contains a configuration assumed by the robot and the Local Safe Region (LSR) perceived from that location, while an arc between two nodes represents a collision-free path between the two configurations. The SRT is incrementally built by using a randomized local planner which privileges the frontier of the LSR, i.e., the directions that lead from the LSR to unexplored areas. This mechanism automatically realizes a trade-off between information gain and navigation cost when choosing the next robot configuration.

Our cooperative exploration strategy is essentially a parallelization of the single-robot SRT method, with the addition of three functionalities: *(i)* cooperation, for increasing efficiency *(ii)* coordination, to avoid conflicts *(iii)* communication, as a fundamental tool to cooperate and coordinate. Each robot of the team builds an SRT, taking into account the presence of other robots through an appropriate redefinition of the concept of local frontier, and planning its motion toward areas which appear to be unexplored by itself as well as the rest of the team. In addition to this local cooperation mechanism, there is a simple coordination algorithm that guarantees safe collective motion. Once a robot has completed its SRT, it makes itself available for supporting other robots in their expansion; this introduces a form of global cooperation. A key feature of the proposed strategy is that it is completely decentralized and can be implemented with a limited communication range.

The paper is organized as follows. After stating the working assumptions, we outline in Sect. III the basic steps of the exploration algorithm running on each robot. In particular, the construction of an SRT with the associated local cooperation and coordination strategies is described in Sect. IV, while the supportive phase is detailed in Sect. V. Simulation results in different environments are reported and discussed in Sect. VI. Possible extensions of the present work are mentioned in the concluding section.

II. Problem setting

The cooperative SRT-based exploration method is presented under the following assumptions.

1) All the robots in the team are identical.

2) The robots move in a planar workspace, i.e., $\mathbb{R}^2$ or a connected subset of it.

3) Each robot is a disk which can move in any direction. The configuration q of the robot is therefore the position of the disk center.

4) Each robot knows its configuration.

5) Each robot is equipped with a sensory system which provides the *Local Safe Region* (LSR) $\mathcal{S}(q)$, a (possibly conservative) description of the free space surrounding the robot at q. The LSR is a star-shaped[1] subset of $\mathbb{R}^2$, whose maximum radius is bounded by the robot perception range R_p. Each LSR is stored in the robot memory with an associated *timestamp*, i.e., the time at which it was gathered.

6) Each robot knows its *ID number*.

7) Each robot can broadcast within a communication range R_c the information stored in its memory (or relevant portions of it) 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 inside R_c.

Many of these assumptions are only taken for simplicity and can be relaxed. First, the robots do not need to be identical: for example, the sensory systems may very well differ in nature and perception range. 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 3 is only taken for ease of presentation: the proposed method is readily applicable to robots with arbitrary shape, possibly subject to nonholonomic constraints. As for Assumption 4, it can be eliminated by incorporating a localization module in the SRT method as described in [15]. See the concluding section for additional comments.

III. Cooperative SRT-based exploration

The design of our cooperative exploration strategy proceeds from the parallelization of the basic SRT method for a single robot described in [12], [13], to which we essentially add three functionalities: cooperation, for increasing efficiency; coordination, to avoid conflicts; and communication as a fundamental tool to cooperate and coordinate.

In the cooperative SRT-based exploration method, each robot builds one or more partial maps of the environment, organized in a collection of Sensor-Based Random Trees (SRTs). Each node of an SRT represents a configuration q which was visited by at least one robot, together with the associated Local Safe Region $\mathcal{S}(q)$. An arc between two nodes represents a collision-free path between the two configurations. The tree is incrementally built by extending the structure in the most promising direction via a biased random mechanism. The presence of other robots in the vicinity is taken into account at this stage in order to maximize the information gain and guarantee collision avoidance.

[1]This means that $\mathcal{S}(q)$ is homeomorphic to the closed unit disk and a line segment from the robot center to any point in $\mathcal{S}$ is completely in $\mathcal{S}$.

```
COOPERATIVE_SRT_BASED_EXPLORATION($q_{\rm init}$)
1  $\mathcal{T}$.init($q_{\rm init}$);
2  BUILD_SRT($q_{\rm init}$,$\mathcal{T}$);
3  do                                    %start the supportive phase
4     for $i = 1$ to $n$
5        if OPEN_FRONTIER($\mathcal{T}_{\rm i}$) $\geq \bar{F}$·ACTIVE_AGENTS($\mathcal{T}_{\rm i}$)
6           ADD($\mathcal{I}, \mathcal{T}_{\rm i}$);
7     $\mathcal{T}_{\rm s} \leftarrow$ SELECT($\mathcal{I}$);
8     if $\mathcal{T}_{\rm s} \neq$ NULL
9        $q_{\rm curr} \leftarrow$ TRANSFER_TO($\mathcal{T}_{\rm s}$.root);
10       BUILD_SRT($q_{\rm curr}$,$\mathcal{T}_{\rm s}$);
11       $q_{\rm curr} \leftarrow$ TRANSFER_TO($\mathcal{T}$.root);
12 while EXPLORATION_RUNNING()
```

Fig. 1. Pseudocode of the SRT-based cooperative exploration algorithm.

The exploration algorithm running on each robot is shown in Fig. 1. At first, the robot builds its own SRT $\mathcal{T}$ rooted at its starting configuration $q_{\rm init}$ through the BUILD_SRT procedure (see Sect. IV). This procedure terminates when the robot is unable to further expand $\mathcal{T}$. At this point, the robot enters the do-while cycle (lines 3–12). This cycle identifies the *supportive phase* in which the robot contributes to the expansion of SRTs that have been initiated by other robots (see Sect. V): this is achieved executing other instances of the BUILD_SRT procedure starting from the roots of these trees. When the supportive phase ends, the robot has returned to the root of its own tree and its exploration is over.

In the above exploration algorithm, only perception, planning and motion functionalities are made explicit. A parallel communication thread runs on each robot, not described here for compactness [16]. In principle, two approaches are possible. In the first, which is chosen for our presentation, each robot continuously broadcasts all its knowledge, including that derived from other robots: this means that the relevant information for cooperation and coordination with other agents inside the communication range R_c is immediately available to the robot. The second solution, aimed at reducing bandwidth consumption, would be to establish robot-to-robot communications only when needed.

IV. The BUILD_SRT procedure

The pseudocode of the BUILD_SRT procedure is shown in Fig. 2. We first give a quick commentary of the main steps, and then discuss their structure in some detail.

At each iteration of the SRT construction, the robot uses all the available information (partly gathered on its own and partly acquired through communication with other robots) to identify 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 (BUILD_AND_WAIT_GPA). Then, the robot gathers

```
BUILD_SRT($q_{\text{init}}$,$\mathcal{T}$)
$q_{\text{curr}} \leftarrow q_{\text{init}}$;
do
    BUILD_AND_WAIT_GPA();
    $\mathcal{S}(q_{\text{curr}})$← PERCEIVE($q_{\text{curr}}$);
    ADD($\mathcal{T}$,($q_{\text{curr}}$, $\mathcal{S}(q_{\text{curr}})$));
    $\mathcal{G}$ ←BUILD_GEA();
    $\mathcal{F}(q_{\text{curr}})$ ← LOCAL_FRONTIER($q_{\text{curr}}$,$\mathcal{S}(q_{curr})$,$\mathcal{T}$,$\bigcup \mathcal{T}_i$);
    $q_{\text{target}}$ ← PLAN($q_{\text{curr}}$,$\mathcal{F}(q_{\text{curr}})$, $q_{\text{init}}$);
    if $q_{\text{target}} \neq$ NULL
      if $|\mathcal{G}| > 1$
          $(\mathcal{G}_f, \mathcal{G}_u)$ ← CHECK_ FEASIBILITY($\mathcal{G}$);
          if $\mathcal{G}_u \neq \emptyset$
              $q_{\text{target}}$ ← COORDINATE($\mathcal{G}_f, \mathcal{G}_u$);
      $q_{\text{curr}}$ ← MOVE_TO($q_{\text{target}}$);
while $q_{\text{target}} \neq$ NULL
```

Fig. 2. Pseudocode of the BUILD_SRT procedure.

data through its sensory systems, builds the current LSR (PERCEIVE) and accordingly updates its own tree $\mathcal{T}$. The actual GEA can now be built (BUILD_GEA). At this point the robot computes its *local frontier* (i.e., the portion of its current LSR boundary leading to areas which appear to be still unexplored) on the basis of $\mathcal{T}$ as well as of any other tree $\mathcal{T}_i$ acquired through communication and stored in its memory (LOCAL_FRONTIER).

If the local frontier is non-empty, the robot generates a random configuration contained in the current LSR and directed towards the local frontier; otherwise, the target configuration can be set either to the parent node (backtracking) or to a child node (if any) provided that local frontiers are present in the subtree of $\mathcal{T}$ starting from that child (PLAN).

If the GEA is composed only of the robot itself, the robot directly moves to its target. Otherwise, the prospective paths of the GEA robots are checked for collisions, and classified in feasible and unfeasible paths (CHECK_FEASIBILITY). If the subset $\mathcal{G}_u$ of robots with unfeasible paths is non-empty, a coordination phase takes place which may either confirm or modify the current target of the robot (COORDINATE). In particular, the robot's move may be simply forbidden by resetting the target to its current configuration. The MOVE function then transfers the robot to the target.

The main loop is repeated until the target configuration is set to NULL by the planner: that is, when the robot is unable to further expand the tree $\mathcal{T}$ (no local frontiers remain) and therefore it has backtracked to the root of its SRT (*homing*).

A. GPA/GEA construction

At the start of BUILD_SRT, the robot is stationary and needs to identify other robots whose LSRs may overlap with its own, in order to cooperate (optimize the exploration) and coordinate (avoid conflicts) 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.

We say that two robots are *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 (Fig. 3, left). To achieve synchronization, the GPA is computed and updated until all its members are stationary (BUILD_AND_WAIT_GPA).

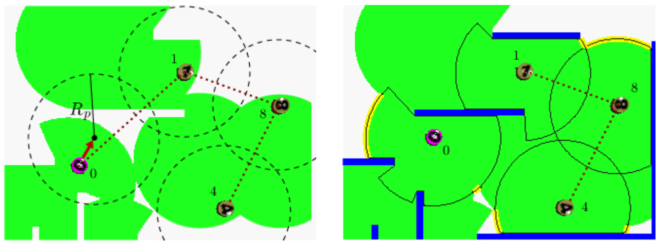

Fig. 3. An example of GPA/GEA construction *Left*: The GPA of robot 1 consists of robots 0,1,4,8: robot 0 is still moving towards its target point, while robots 1,4,8 are stationary. The perception areas of the robots (prospective in the case of robot 0) overlap in pairs. *Right*: Once the LSR have been computed, only robots 1,4,8 belong to the GEA of robot 1 since their LSRs overlap in pairs.

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 $3R_p$ (the other robot may still be moving to its target, which however cannot be farther than R_p from the current configuration), it is sufficient to assume $R_c \geq 3R_p$ 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, has perceived the LSR and updated its SRT, 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 (Fig. 3, right) is composed by all the GPA robots to which it can be connected through a chain of GEA couplings (BUILD_GEA). Synchronization guarantees that all the GPA robots are still 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. A remarkable feature is that the GEA can be built with a limited communication range ($R_c \geq 3R_p$).

B. Frontier extraction

Once the robot has built its GEA, the LOCAL_FRONTIER procedure is invoked. This computes the portion of the boundary of the LSR $\mathcal{S}(q_{\text{curr}})$ leading to areas which appear unexplored according to the available information. To this end, the robot uses its own tree $\mathcal{T}$ as well as any other tree $\mathcal{T}_i$ stored in its memory and received through communications.

To find promising directions in $\mathcal{S}(q_{\text{curr}})$, its boundary is divided in *obstacle*, *free* and *frontier* arcs (Fig. 4). Obstacle arcs are located along the directions where obstacles have been detected, while free arcs fall within other known LSRs (either belonging to $\mathcal{T}$ or to $\bigcup \mathcal{T}_i$). Any arc which is neither obstacle nor free is a frontier arc, and by construction identifies the boundary between $\mathcal{S}(q_{\text{curr}})$ and an unexplored region. LOCAL_FRONTIER identifies the frontier arcs of $\mathcal{S}(q_{\text{curr}})$ directly from the range reading profile [13], and collects them in the local frontier $\mathcal{F}(q_{\text{curr}})$.

The above definition, according to which a frontier arc cannot belong to trees being built by other robots, implements a simple decentralized cooperation mechanism aimed at optimizing the exploration performance of the team. Such

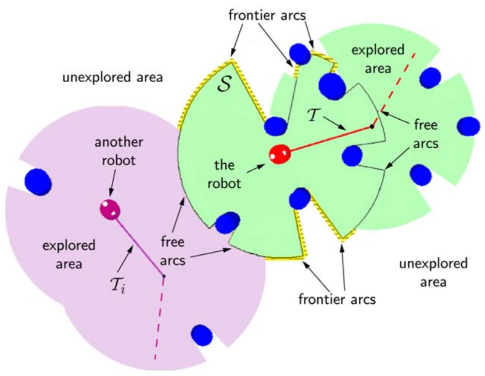

Fig. 4. Frontier, free and obstacle arcs in the Local Safe Region $\mathcal{S}$.

mechanism is inherently local and contingent because it relies on communication between the robots.

C. Planner

The planner (Fig. 5) determines the new target configuration on the basis of the local and open frontiers associated to the current node. We define the *open frontier* of a tree (subtree) as the sum of the lengths of the local frontiers associated to its nodes.

If the local frontier $\mathcal{F}(q_{\text{curr}})$ of the current LSR is nonempty, the planner generates a random configuration in the direction of $\mathcal{F}(q_{\text{curr}})$. In particular, RANDOM_DIR selects an exploration direction ϕ_{rand} by first choosing one of the frontier arcs, using a probability proportional to the arc length, and then by generating ϕ_{rand} on the basis of a normal distribution with mean value ϕ_{m} (the orientation of the bisectrix of the arc) and standard deviation $\gamma/6$ (γ being the arc angular width). The DISPLACE function generates a new target configuration q_{target} moving from q_{curr} in the direction of ϕ_{rand} with a certain stepsize (a fixed percentage of the LSR extension in that direction).

If the local frontier is empty, two possibilities occur. Let $\mathcal{T}(q_{\text{curr}})$ be the subtree of $\mathcal{T}$ rooted at q_{curr}. If the open frontier of $\mathcal{T}(q_{\text{curr}})$ is nonzero, the robot invokes the RANDOM_CHILD function. This function performs a random selection among the children of the current node q_{curr}, using a probability proportional to the open frontier of the corresponding subtrees, and sets the target to the chosen child. On the other hand, if the open frontier of $\mathcal{T}(q_{\text{curr}})$ is zero, the planner sets the target configuration to the parent node, i.e., it forces the robot to backtrack. In the latter case, when q_{curr} equals q_{init}, the robot is at the root of the currently built tree and is unable to further expand it: hence, the planner sets the target configuration to NULL and BUILD_SRT is exited.

Note that, for simplicity, we have referred to a point robot in explaining the planner. The finite size (and possibly the non-disk shape) of the robot is readily taken into account by mapping frontier arcs to the configuration space, where actual planning takes place. The possibility that the robot is subject to nonholonomic constraints would be considered by the MOVE_TO function, which is in charge of generating feasible paths. Controllability of the robot guarantees that any target configuration in the LSR can be reached by a

```
PLAN($q_{\text{curr}}$,$\mathcal{F}(q_{\text{curr}})$, $q_{\text{init}}$)
1   $q_{\text{target}} \leftarrow$ NULL;
2   if $\mathcal{F}(q_{\text{curr}}) \neq \emptyset$
3       $\phi_{\text{rand}} \leftarrow$ RANDOM_DIR($\mathcal{F}(q_{\text{curr}})$);
4       $q_{\text{target}} \leftarrow$ DISPLACE($q_{\text{curr}}$, $\phi_{\text{rand}}$);
5   else if OPEN_FRONTIER($\mathcal{T}(q_{\text{curr}})$) $\neq 0$
6           $q_{\text{target}} \leftarrow$ RANDOM_CHILD($q_{\text{curr}}$);
7       else if $q_{\text{curr}} \neq q_{\text{init}}$
8           $q_{\text{target}} \leftarrow q_{\text{curr}}$.parent;
9   return $q_{\text{target}}$;
```

Fig. 5. Pseudocode of the random planner.

feasible path which stays in the LSR.

Thanks to the synchronization phase performed by BUILD_AND_WAIT_GPA, all the robots in a GEA plan at the same time, and therefore the cooperation mechanism intrinsic to the local frontier definition is enforced on all the GEA. This 'agreement of intents' is realized without any centralized decision module.

D. GEA path feasibility check

Although the local frontier of the robot cannot belong to the LSR of another robot of the GEA (Fig. 4), the two prospective paths may still intersect. Figure 6 is an example of the conflict that may arise. Hence, the CHECK_FEASIBILITY function verifies whether the prospective paths of robots in the GEA $\mathcal{G}$ are all simultaneously feasible[2] or not. To this end, all the pairs of paths that intersect with each other 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 $O(|\mathcal{G}|^2)$.

E. Coordination

If the subset $\mathcal{G}_u$ of robots with unfeasible paths is nonempty, the coordination function is invoked (Fig. 7). The first step is to elect a *master robot* within $\mathcal{G}$. 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 invokes the ORGANIZE function, whose task is to rearrange the vector $\mathcal{Q}_{\mathcal{G}}$ containing the targets of the GEA robots 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 versions of the function, i.e., ORGANIZE_1 and ORGANIZE_2 (see below).

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

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

1) Organization via arbitration: ORGANIZE_1 implements a simple arbitration mechanism on $\mathcal{G}$. All the robots contained in the feasible subset $\mathcal{G}_f$ are allowed to move (their

[2]Collision check is performed in configuration space. Not taking into account the possibility of velocity scaling along the paths to avoid collisions. Paths that intersect are not allowed in our approach.

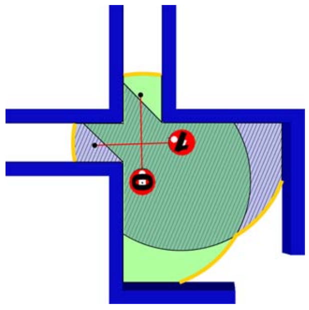

Fig. 6. The prospective paths of robots belonging to the GEA may intersect.

target configuration is left unchanged). The robots in the unfeasible subset $\mathcal{G}_u$ are not allowed to move (their target configuration is reset to the current configuration) with the exception of a single one whose motion is authorized (by construction, this strategy does not 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 local frontier is empty: these are robots whose target is either their parent node or one of their children nodes. 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 local frontier is non-empty, to which the random planner may propose a different target in the next cycle).

An antithetical criterion would be to choose randomly among the robots in $\mathcal{G}_u$, using a probability proportional to the extension of their local frontier.

2) Organization via replanning: ORGANIZE_2 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 target configurations $\mathcal{Q}_\mathcal{G}$ associated to the GEA $\mathcal{G}$. Two target configurations 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 [17], [18].

The ideal objective of the ORGANIZE_2 function is to modify the set of target configurations $\mathcal{Q}_\mathcal{G}$ 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 towards the local frontier of the robot (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. A description of the game is given in [16].

```
COORDINATE($\mathcal{G}_f$,$\mathcal{G}_u$)
1   master_id ← MASTER_ELECTION($\mathcal{G}$);
2   **if** my_id = master_id
3       $\mathcal{Q}_\mathcal{G}$ ← ORGANIZE($\mathcal{G}_f$,$\mathcal{G}_u$);
4   **else**
5       WAIT;
6   **return** $\mathcal{Q}_\mathcal{G}$(my_id);
```

Fig. 7. Pseudocode of the coordination function.

V. The Supportive Phase

The supportive phase (Fig. 1, lines 3–12) can be divided in two main subphases, which are repeated over and over: *(i)* the robot chooses which other robot to support in its exploration, or, more precisely, which other tree to help expand (there may be more than one robot acting on a tree); *(ii)* the selected tree is reached and the robot tries to extend it by attaching subtrees built via the BUILD_SRT procedure.

The do-while loop is repeated until the robot has received confirmation that all the other robots have finished their exploration. At first, the robot collects in a set $\mathcal{I}$ the trees belonging to $\bigcup \mathcal{T}_i$ that may need support for expansion. In particular, a tree $\mathcal{T}_i$ is put in $\mathcal{I}$ if its open frontier is at least equal to a constant $\bar{F}$ multiplied by the number of robots that are active on $\mathcal{T}_i$ according to the most recent available information (ACTIVE_AGENTS). If $\mathcal{I}$ is non-empty, the robot selects a particular tree $\mathcal{T}_\mathrm{s}$ from $\mathcal{I}$ according to some criteria (e.g., the tree with the closest root, or with the most recent update time), and travels to its root. Once there, the robot begins a subtree expansion using the BUILD_SRT procedure. During this process, the robot keeps on trying to add subtrees to $\mathcal{T}_\mathrm{s}$ until it has returned to the root of $\mathcal{T}_\mathrm{s}$ and its open frontier is zero. At this point, the robot goes back to the root of its own tree (its start configuration) and makes itself available for supporting the expansion of other trees.

VI. Simulations

In this section we present simulations of the proposed SRT-based cooperative exploration method in Move3D [19], a software platform developed at LAAS-CNRS and dedicated to motion planning[3]. The team is composed of a varying number of MagellanPro carrying a 360° laser range finder, with a perception range of four times the robot diameter.

The performance of the method is evaluated in terms of *exploration time* (the time required by the last robot of the team to return home) and *total traveled distance* (the sum of the distances traveled by each robot). These values are respectively expressed as a percentage of the exploration time and traveled distance obtained with a team composed by a single robot. Environment coverage is not reported because it was complete in all our simulations. In view of the randomized nature of our method, numerical results for each situation are averaged over ten simulation runs.

Two different environments have been used. The first is a square region with a garden-like layout, where each area can be reached from different access points. The second,

[3]Move3D is at the origin of the product KineoWorks currently marketed by the company Kineo CAM (www.kineocam.com).

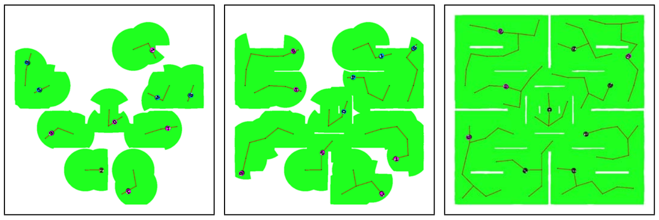

Fig. 8. Garden: SRT-based cooperative exploration with scattered start.

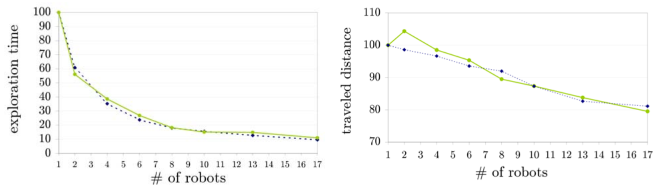

Fig. 9. Garden exploration with scattered start: exploration time and total traveled distance with teams of different cardinality for unlimited (dotted line, blue) and limited (solid line, green) communication range.

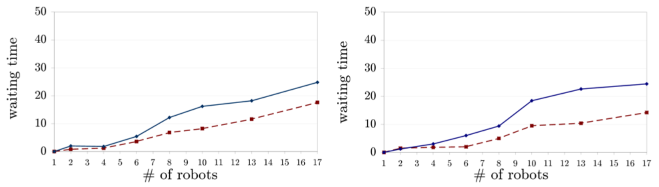

Fig. 10. Garden exploration with scattered start: waiting time with the ORGANIZE_1 (solid line, blue) and ORGANIZE_2 (dashed line, red) coordination strategies. Left: limited communication range. Right: unlimited communication range.

similar to an office, extends mainly along one dimension and contains areas (rooms) which can be accessed by one passage (door) only.

We consider two possible 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.

Figures 8–10 refer to results obtained in the garden environment with a scattered start. In particular, Fig. 8 shows the exploration progress with a team of 10 robots in the case of limited communication range $R_c = 3R_p$. Here, ORGANIZE_2 is used for coordination. Note how each robot builds its own SRT and never enters the supportive phase. We have found this behavior to be common when the robots are evenly distributed at the start.

Exploration time and traveled distance for teams of different cardinality are shown in Fig. 9, both in the case of limited and unlimited communication range. Again, ORGANIZE_2 is used for coordination (the performance with ORGANIZE_1 is similar). As the number of robots in the team increases, the exploration time quickly decreases and tends asymptotically to zero (consider that an increment of 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 total traveled distance; however, in the case of a limited communication range, this parameter increases for a two-dimensional team, due to the fact that two robots which are far apart at the start can exchange very little information during the exploration. As the number of robots

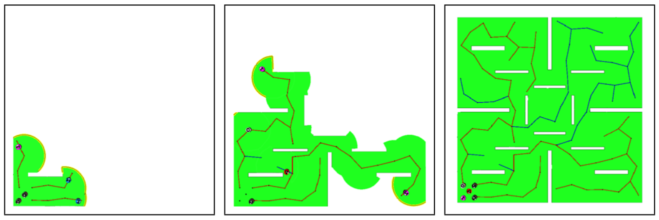

Fig. 11. Garden: SRT-based cooperative exploration with clustered start.

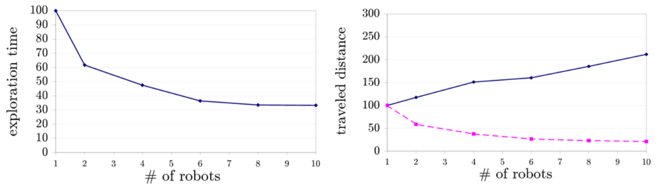

Fig. 12. Garden exploration with clustered start: exploration time (left) and total traveled distance (right, solid blue line) with teams of different cardinality for limited communication range. Also shown is the average distance traveled by each robot (right, dashed violet line).

increases, communication chains are easily formed and the total distance decreases.

The difference between the coordination strategies ORGANIZE_1 and ORGANIZE_2 can be appreciated from Fig. 10, which plots the *waiting time* (i.e., the average percentage of the exploration time that a robot spends waiting due to coordination) both in the case of limited and unlimited communication range. The inactivity period is always smaller with ORGANIZE_2. Note also that the waiting time grows with the number of robots, due to the corresponding increased importance of the coordination phase.

A typical exploration progress obtained in the garden environment with a team of 5 robots with a clustered initial deployment is shown in Fig. 11. Here, ORGANIZE_2 is used for coordination and the communication range is limited to $R_c = 3R_p$. In this case, three of the robots begin expanding their own SRT, while the other two see no local frontier and therefore wait until they can support the others (i.e., until there is a sufficient amount of open frontier in one of the trees), joined by one of the three whose SRT expansion terminates very soon. At the end, the environment has been completely explored and three SRTs have been built.

Figure 12 summarizes the performance 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. The total traveled distance increases with the cardinality of the team because more robots try to support the others in their expansion. Not surprisingly, the average distance traveled by each robot tends to zero. The progress of the cooperative exploration in the first two simulations is shown in the video clip attachment to the paper.

The last simulation deals with the exploration of the office environment with limited communication range. In particular, Fig. 13 shows the progress of the exploration for a team of 6 robots with clustered start, while Fig. 14 shows the influence of the number of robots on the performance.

Video clips of these as well as of other simu-

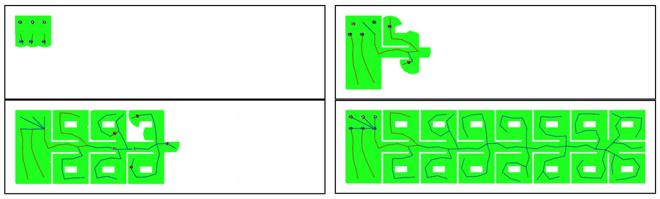

Fig. 13. Office: SRT-based cooperative exploration with clustered start.

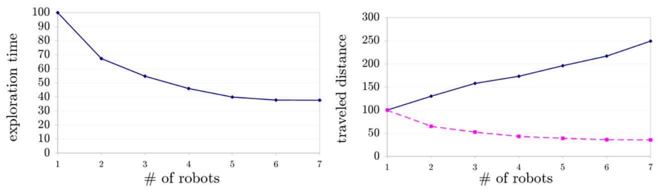

Fig. 14. Office exploration with clustered start: exploration time (left) and total traveled distance (right, solid blue line) with teams of different cardinality for limited communication range. Also shown is the average distance traveled by each robot (right, dashed violet line).

lations of our method are available at the webpage `http://www.dis.uniroma1.it/~labrob/research/multiSRT.html` .

VII. Conclusions

We have presented a randomized strategy for cooperative exploration based on the SRT (Sensor-based Random Tree) concept. The method entails two decentralized cooperation mechanisms at different levels. The first simply 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. The second allows a robot that has completed its individual exploration phase to support the others in their task. Local coordination guarantees that the collective motion of the team is feasible from the collision viewpoint. Simulation results have shown the satisfactory performance of the method, even in the case of limited communication radius.

In our view, the main feature of the proposed method is that no central supervision is needed, and no task decomposition/allocation is performed. The selection of exploration actions by each robot is spontaneous and made on the basis of the available information. This is expected to provide robustness with respect to possible robot breakdowns as well as communication failures.

Future work will address several aspects, i.e., the integration of a localization module along the lines of [15], the extension to heterogeneous robots, and an experimental validation so as to assess the robustness of the proposed method to sensor and communication failures. Two further aspects that will be reconsidered are memory usage and communication procedures. In particular, the ‘broadcasting’ approach taken in this paper may be troublesome in the case of limited bandwidth. One-to-one communication procedures including only the most recent data can be designed to alleviate this problem.

Finally, we are currently working to remove the separation of the supportive phase in the current algorithm (first expand your tree, then help others). Additional edges, called *bridges*, can be used to join pairs of nodes which belong to different SRTs and have overlapping LSRs. In this way, all the SRTs can be connected in a single super-graph in which the bridges can be used by the robots as collision-free local paths to transfer from one SRT to another. This super-graph can be incrementally updated by each robot every time a new node is added to an SRT. Once this graph is available, the planner can consider as possible target configurations also the nodes of different SRTs which present promising open frontiers and are reachable with a local bridge from the current configuration. This modification would allow the robots to share, at all times, all the currently built SRTs in a more effective way, avoiding the separation between phases and root-to-root transfers.

References

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

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

[3] 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.

[4] D. Goldberg and M. Mataric, “Interference as a tool for designing and evaluating multi-robot controllers,” in *14th AAAI/9th IAAI*, 1997, pp. 637–642.

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

[6] B. Yamauchi, “Frontier-based exploration using multiple robots,” in *2nd Int. Conf. on Autonomous Agents*, 1998, pp. 47–53.

[7] 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.

[8] 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*, 2003, pp. 3232–3238.

[9] A. Howard, M. Mataric, and S. Sukhatme, “An incremental deployment algorithm for mobile robot teams,” in *2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003, pp. 2849–2854.

[10] 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*, 2002, pp. 3016–3023.

[11] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, “Coordination for multi-robot exploration and mapping,” in *17th AAAI/12th IAAI*, 2000, pp. 852–858.

[12] 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*, 2004, pp. 4688–4694.

[13] L. Freda and G. Oriolo, “Frontier-based probabilistic strategies for sensor-based exploration,” in *2005 IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 3892–3898.

[14] S. M. LaValle and J. J. Kuffner, “Rapidly-exploring random trees: Progress and prospects,” in *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Eds. Wellesley, MA: A K Peters, 2001, ch. 10, pp. 293–308.

[15] L. Freda, F. Loiudice, and G. Oriolo, “A randomized method for integrated exploration,” *2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[16] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, “A randomized strategy for cooperative robot exploration,” DIS, Università di Roma “La Sapienza”, Tech. Rep., 2006.

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

[18] F. Harary, *Graph Theory*. Reading, MA: Addison-Wesley, 1994.

[19] T. Simeon, J.-P. Laumond, and F. Lamiraux, “Move3d: A generic platform for path planning,” in *4th Int. Symp. on Assembly and Task Planning*, 2001, pp. 25–30.