Auton Robot
DOI 10.1007/s10514-016-9578-9

Decentralized simultaneous multi-target exploration using a connected network of multiple robots

Thomas Nestmeyer[1] · **Paolo Robuffo Giordano**[2] · **Heinrich H. Bülthoff**[3] · **Antonio Franchi**[4,5]

Received: 24 October 2015 / Accepted: 19 May 2016

Abstract This paper presents a novel decentralized control strategy for a multi-robot system that enables parallel multi-target exploration while ensuring a time-varying connected topology in cluttered 3D environments. Flexible continuous connectivity is guaranteed by building upon a recent connectivity maintenance method, in which limited range, line-of-sight visibility, and collision avoidance are taken into account at the same time. Completeness of the decentralized multi-target exploration algorithm is guaranteed by dynamically assigning the robots with different motion behaviors during the exploration task. One major group is subject to a suitable downscaling of the main traveling force based on the traveling efficiency of the current leader and the direction alignment between traveling and connectivity force. This supports the leader in always reaching its current target and, on a larger time horizon, that the whole team realizes the overall task in finite time. Extensive Monte Carlo simulations with a group of several quadrotor UAVs show the scalability and effectiveness of the proposed method and experiments validate its practicability.

Keywords Multi-robot coordination · Path-planning · Decentralized exploration · Connectivity maintenance

Simulations and experiments were performed while the authors were at the MPI for Biological Cybernetics, Tübingen, Germany.

Electronic supplementary material The online version of this article (doi:10.1007/s10514-016-9578-9) contains supplementary material, which is available to authorized users.

✉ Antonio Franchi
antonio.franchi@laas.fr

Thomas Nestmeyer
tnestmeyer@tue.mpg.de

Paolo Robuffo Giordano
prg@irisa.fr

Heinrich H. Bülthoff
hhb@tuebingen.mpg.de

1 Max Planck Institute for Intelligent Systems, Spemannstraße 41, 72076 Tübingen, Germany

2 CNRS at Irisa and Inria Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes cedex, France

3 Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany

4 CNRS, LAAS, 7 Av. du Colonel Roche, 31400 Toulouse, France

5 Univ de Toulouse, LAAS, 31400 Toulouse, France

1 Introduction

Success of multi-robot systems is based on their ability of parallelizing the execution of several small tasks composing a larger complex mission such as, for instance, the inspection of a certain number of locations either generated off- or online during the robot motion (e.g., exploration, data collection, surveillance, large-scale medical supply or search and rescue (Howard et al. 2006; Franchi et al. 2009; Pasqualetti et al. 2012; Murphy et al. 2008; Faigl and Hollinger 2014)). In all these cases, a fundamental difference between a group of many single robots and a multi-robot system is the ability to communicate (either explicitly or implicitly) in order to then cooperate together towards a common objective. Another distinctive characteristic in multi-robot systems is the absence of central planning units, as well as all-to-all communication infrastructures, leading to a *decentralized* approach for algorithmic design and implementations (Lynch 1997). While communication of a robot with every other robot in the group (via multiple hops) would still be possible as long as the group stays connected, in a decentralized approach each robot is only assumed able to communicate with the robots in its 1-hop neighborhood (i.e., typically the ones spatially close by). This brings the advantage of scalability in communication and computation complexity when considering groups of many robots.

The possibility for every robot to share information (via, possibly, multiple hops/iterations) with any other robot in the group is a basic requirement for typical multi-robot algorithms and, as well-known, it is directly related to the connectivity of the underlying *graph* modeling inter-robot interactions. Graph connectivity is a prerequisite to properly fuse the information collected by each robot, e.g., for mapping, localization, and for deciding the next actions to be taken. Additionally, many distributed algorithms like consensus (Olfati-Saber and Murray 2004) and flooding (Lim and Kim 2001) require a connected graph for their successful convergence. Preserving graph connectivity during the robot motion is, thus, a fundamental requirement; however, connectivity maintenance may not be a trivial task in many situations, e.g., because of limited capabilities of onboard sensing/communication devices which can be hindered by constraints such as occlusions or maximum range. Given the cardinal role of communication for the successful operation of a multi-robot team, it is then not surprising that a substantial effort has been spent over the last years for devising strategies able to preserve graph connectivity despite constraints in the inter-robot sensing/communication possibilities, see, e.g., Antonelli et al. (2005), Stump et al. (2008, 2011), Pei and Mutka (2012), Robuffo Giordano et al. (2013). In general, *fixed topology* methods represent conservative strategies that achieve connectivity maintenance by restraining any pairwise link of the interaction graph to be broken during the task execution. A different possibility is to aim for *periodical connectivity* strategies, where each robot can remain separated from the group during some period of time for then rejoining when necessary. *Continuous connectivity* methods instead try to obtain maximum flexibility (links can be continuously broken and restored unlike in the fixed topology cases) while preserving at any time the fundamental ability for any two nodes in the group to share information via a (possibly multi-hop) path (unlike in periodical connectivity methods).

With respect to this state-of-the-art, the problem tackled by this paper is the design of a *multi-target* exploration/visiting strategy for a team of mobile robots in a cluttered environment able to (*i*) allow visiting multiple targets at once (for increasing the efficiency of the exploration), while (*ii*) always guaranteeing connectivity maintenance of the group despite some typical sensing/communication constraints representative of real-world situations, (*iii*) without requiring presence of central nodes or processing units (thus, developing a fully *decentralized* architecture), and (*iv*) without requiring that all the targets are known at the beginning of the task (thus, considering *online target generation*).

Designing a decentralized strategy that combines multi-target exploration and continuous connectivity maintenance is not trivial as these two goals impose often antithetical constraints. Several attempts have indeed been presented in the previous literature: a *fixed-topology* and centralized method is presented in Antonelli et al. (2005), which, using a virtual chain of mobile antennas, is able to maintain the communication link between a ground station and a single mobile robot visiting a given sequence of target points. The method is further refined in Antonelli et al. (2006). A similar problem is addressed in Stump et al. (2008) by resorting to a partially centralized method where a linear programming problem is solved at every step of motion in order to mix the derivative of the second smallest eigenvalue of a weighted Laplacian (also known as algebraic connectivity, or Fiedler eigenvalue) and the k-connectivity of the system. A line-of-sight communication model is considered in Stump et al. (2011), where a centralized approach, based on polygonal decomposition of the known environment, is used to address the problem of deploying a group of roving robots while achieving *periodical connectivity*. The case of periodical connectivity is also considered in Pasqualetti et al. (2012) and Hollinger and Singh (2012). The first paper optimally solves the problem of patrolling a set of points to be visited as often as possible. The second presents a heuristic algorithm exploiting the concept of implicit coordination. *Continuous connectivity* between a group of robots exploring an unknown 2D environment and a single base station is considered in Pei et al. (2010). The proposed exploration methodology, similar to the one presented in Franchi et al. (2009), is integrated with a centralized algorithm running on the base station and solving a variant of the Steiner Minimum Tree Problem. An extension of this approach to heterogeneous teams is presented in Pei and Mutka (2012). Zavlanos and Pappas (2007) exploit a potential field approach to keep the second smallest eigenvalue of the Laplacian positive. The method is tested with ground robots in an empty environment and assumes that each robot has access to the whole formation for computing the connectivity eigenvalue and the associated potential field. It is therefore not scalable, because the strength of all links has to be broadcasted to all robots in the group. Continuous connectivity achieved by suitable mission planning is described in Mosteo et al. (2008), although this work does not allow for parallel exploration. Another method providing flexible connectivity based on a spring-damper system, but not able to handle significant obstacles, is reported in Tardioli et al. (2010).

A decentralized strategy addressing the problem of continuous connectivity maintenance for a multi-robot team is considered in Robuffo Giordano et al. (2013). In this latter work, the introduction of a sensor-based weighted Laplacian allows to distributively and analytically compute the anti-gradient of a generalized Fiedler eigenvalue. The connectivity maintenance action is further embedded with additional constraints and requirements such as inter-robot and obstacle collision avoidance, and a stability guarantee of the whole system, when perturbed by external control inputs for steering the whole formation, is also provided. Finally, apart for Robuffo Giordano et al. (2013), all the previously mentioned continuous connectivity methods have only been applied to 2D-environment models.

In this work, we leverage upon the general decentralized strategy for connectivity maintenance of Robuffo Giordano et al. (2013) for proposing a solution to the aforementioned problem of decentralized *multi-target exploration* while coping with the (possibly opposing) constraints of continuous connectivity maintenance in a cluttered 3D environment. The main contributions of this paper and features of the proposed algorithm can then be summarized as follows: *(i)* decentralized and continuous maintenance of connectivity, *(ii)* guarantee of collision avoidance with obstacles and among robots, *(iii)* possibility to take into account non-trivial sensing/communication models, including maximum range and line-of-sight visibility in 3D, *(iv)* stability of the overall multi-robot dynamical system, *(v)* decentralized exploration capability, *(vi)* possibility for more than one robot to visit different targets at the same time, *(vii)* online path planning without the need for any (centralized) pre-planning phase, *(viii)* applicability to both 2D and 3D cluttered environments, and finally *(ix)* completeness of the multi-target exploration (i.e., all robots are guaranteed to reach all their targets in a finite time). The items *(i)*–*(iv)* have already been tackled in Robuffo Giordano et al. (2013) and are here taken as a basis for our work. On the other hand, the combination of *(i)*–*(iv)* with the items *(v)*–*(ix)* is a novel contribution: to the best of our knowledge, our work is then the first attempt to propose a decentralized multi-target exploration algorithm possessing all the mentioned features altogether.

The rest of the paper is organized as follows: Sect. 2 provides a formal description of the problem under consideration. The proposed algorithm is then thoroughly illustrated in Sect. 3. In Sect. 4, we report the results of extensive Monte Carlo simulations and experiments with real quadrotors, and Sect. 5 concludes the paper. In the Appendix, we recap the main features of the decentralized continuous connectivity method presented in Robuffo Giordano et al. (2013) which is extensively exploited in this paper. We finally note that a preliminary version of our work has been presented in Nestmeyer et al. (2013a, b).

2 System model and problem setting

We consider a group of N robots operating in a 3D obstacle-populated environment and denote with $q_i \in \mathbb{R}^3$ the position of a reference point of the i-th robot, $i = 1, \ldots, N$, in an inertial world frame. We also let $\mathcal{O}$ be the set of obstacle points in the environment. Each robot i is assumed to be endowed with an omnidirectional sensor able to measure the relative position $q_j - q_i$ of another robot j provided that:

1. $\|q_j - q_i\| < R_s$, where $R_s > 0$ is the maximum sensing range of the sensor, and
2. $\min_{\varsigma\in[0,1],o\in\mathcal{O}} \|q_i + \varsigma(q_j - q_i) - o\| \geq R_o$, i.e., the line segment connecting q_i to q_j is at least at distance $R_o > 0$ away from any obstacle point.

These two conditions account for two common characteristics of exteroceptive sensors, namely, presence of a limited sensing range R_s, and the need for a non-occluded line-of-sight visibility.[1] We further assume that if the i-th robot can measure $q_j - q_i$ then it can also communicate with the j-th robot with negligible delays, that is, the sensing and communication graphs are taken coincident. This assumption is justified by the fact that communication typically relies on wireless technology, thus with a broader range than sensing and without the need for direct visibility to operate. The neighbors of the i-th robot are denoted with $\mathcal{N}_i(t)$, i.e., the (time-varying) set of robots whose relative position can be measured by the i-th robot at time t.

Each robot i is also endowed with a sensor that measures the relative position $o - q_i$ of every obstacle point $o \in \mathcal{O}$ such that $\|o - q_i\| < R_m$, where $R_m > 0$ is the maximum sensing range of this sensor.

Consider the time-varying (undirected) *interaction graph* defined as $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, where $\mathcal{V} = \{1, \ldots, N\}$ and $\mathcal{E}(t) = \{(i, j) \mid j \in \mathcal{N}_i(t)\}$. Preserving connectivity of $\mathcal{G}(t)$ for all t, allows every robot to communicate *at any time* with any other robot in the network by means of a suitable multi-hop routing strategy, although due to efficiency and scalability reasons, it is always preferred to use one-hop communication when possible.

As previously stated, decentralized continuous connectivity maintenance is guaranteed by exploiting the method described in Robuffo Giordano et al. (2013), which is based on a gradient-descent action that keeps the second smallest eigenvalue λ_2 of the *sensor-based* weighted graph Laplacian positive (Fiedler 1973) (see Appendix for a formal definition).

[1] More complex sensing models could also be taken into account, see Robuffo Giordano et al. (2013) for a discussion in this sense.

Each i-th robot is finally endowed with a local motion controller able to let q_i track any arbitrary desired $\bar{\mathcal{C}}^2$ trajectory $q_i(t)$ with a sufficiently small tracking error. This is again a well-justified assumption for almost all mobile robotic platforms of interest, and its validity will also be supported by the experimental results of Sect. 4.2. Following the control framework introduced in Robuffo Giordano et al. (2013), the dynamics of q_i is then modeled as the following second order system

$$\Sigma : \begin{cases} M_i \dot{v}_i - f_i^B - f_i^\lambda = f_i \\ \dot{q}_i = v_i \end{cases} \qquad i = 1, \ldots, N \tag{1}$$

where $v_i \in \mathbb{R}^3$ is the robot velocity, $M_i \in \mathbb{R}^{3\times 3}$ is its positive definite inertia matrix, and:

1. $f_i^B = -B_i v_i \in \mathbb{R}^3$ is the *damping force* (with $B_i \in \mathbb{R}^{3\times 3}$ being a positive definite damping matrix) meant to represent both typical friction phenomena (e.g., wind/atmosphere drag in the case of aerial robots) and/or a stabilizing control action;
2. $f_i^\lambda \in \mathbb{R}^3$ is the *generalized connectivity force* whose decentralized computation and properties are thoroughly described in Robuffo Giordano et al. (2013) (a short recap is provided in Appendix);
3. $f_i \in \mathbb{R}^3$ is the *traveling force* used to actually steer the robot motion in order to execute the given task. An appropriate design of f_i is the main goal of this work. As will be clear in the following, special care must be taken in the design of f_i to avoid, for instance, deadlocks situations in which the robot group 'gets stuck'.

The following fact, shown in Robuffo Giordano et al. (2013) and recalled in Appendix, holds:

Fact 1 *As long as f_i keeps bounded, the action of the generalized connectivity force f_i^λ will always ensure obstacle and inter-robot collision avoidance* and *continuous connectivity maintenance for the graph $\mathcal{G}(t)$ despite the various sensing/communication constraints (in the worst case, by completely dominating the bounded f_i).*

To summarize, each robot has *(i)* an accurate enough measurement of its own location, *(ii)* an omnidirectional sensor able to measure relative positions of other robots and obstacles in its close proximity, *(iii)* negligible (compared to the time scale of the robot motion) communication delays with all robots that it can sense/communicate, *(iv)* the ability to accurately track a smooth path with a force controller.

2.1 Multi-target exploration problem

We consider the broad class of problems in which each robot runs a black-boxed algorithm that produces *online*[2] a continually adjustable list of targets that have to be visited by the robot in the presented order. We refer to this algorithm as the *target generator* of the i-th robot, and we also assume that the portion of the map needed to reach the next location from the current position q_i is known to robot i. The target generator may represent a large variety of algorithms, such as pursuit-evasion (Durham et al. 2012), patrolling (Pasqualetti et al. 2012), exploration/mapping (Franchi et al. 2009; Burgard et al. 2005), mobile-ad-hoc-networking (Antonelli et al. 2005), and active localization (Jensfelt and Kristensen 2001). It might be a cooperative algorithm, or each robot could have a target generator with objectives that are independent from the other target generators. Another possibilty is to appoint a human supervisor as the target generator.

Depending on the particular application, the locations in the lists provided online by the target generators may, e.g., represent:

1. View-points from where to perform the sensorial acquisitions,
2. Coordinates of objects that have to be picked up or dropped down,
3. Positions of some base stations located in the environment.

We formally denote with $(z_i^1, \ldots, z_i^{m_i}) \in \mathbb{R}^{3\times m_i}$ the list of m_i locations provided by the i-th target generator. Additionally, we consider the possibility, for the target generator, to specify a time duration $\Delta t_i^k < \infty$ for which the i-th robot is required to stay close to the point z_i^k, with $k = 1, \ldots, m_i$. This quantity may represent, with reference to the previous examples, the time

1. Needed to perform a full sensorial acquisition,
2. Necessary to pick up/drop down an object,
3. Required to upload/download some information from a base station,

and can also possibly be adjusted at runtime during the execution of the respective task.

Finally, we also introduce the concept of a *cruise speed* $v_i^{\text{cruise}} > 0$ that should be maintained by the i-th robot during the transfer phase from a point to the next one.

[2] By *online* we mean that the targets are generated at runtime, thus precluding the presence of a preliminary phase in which the robots may *plan in advance* the multi-target exploration action. Indeed, if all the targets are known beforehand, one could still apply our method but other planning strategies might potentially lead to better solutions.

Given these modeling assumptions, the problem addressed in this paper can be formulated as follows:

Problem 1 *Given a sequence of targets $z_i^1, \ldots, z_i^{m_i}$ (presented online) for every robot $i = 1, \ldots, N$, together with the corresponding sequence of time durations $\Delta t_i^1, \ldots, \Delta t_i^{m_i}$ and a radius R_z, design, for every $i = 1, \ldots, N$, a decentralized feedback control law f_i (i.e., a function using only information locally and 1-hop available to the i-th robot) for system (1) which is bounded and such that, for the closed-loop trajectory $q_i(t, f_{i,[0,t)})$, there exists a time sequence $0 < t_i^1 < \ldots < t_i^{m_i} < \infty$ so that for all $k = 1 \ldots m_i$, robot i remains for the duration Δt_i^k within a ball of radius R_z centered at z_i^k, formally $\forall t \in [t_i^k, t_i^k + \Delta t_i^k] : \|q_i(t) - z_i^k\| < R_z$.*

3 Decentralized algorithm

In this section, we describe the proposed distributed algorithm aimed at generating a traveling force f_i that solves Problem 1. We note that the design of such an autonomous distributed algorithm requires special care: When added to the generalized connectivity force in (1), the traveling force f_i should fully exploit the group capabilities to concurrently visit the targets of all robots whenever possible and, at the same time, should not lead to '*local minima*', where the robots get stuck, due to the simultaneous presence of the hard connectivity constraint. While Robuffo Giordano et al. (2013) already gave an exact description of f_i^B and f_i^λ, an application of f_i was kept open. The main focus of this work is to define f_i in such a way that the above mentioned challenges are properly addressed.

In order to provide an overview of the several variables used in Sects. 2 and 3, we included Table 1 for the reader's convenience.

Table 1 Meaning of the variable names

Variable	Meaning
N	Number of robots
q_i	Position of i-th robot
v_i	Velocity of i-th robot
$\mathcal{O}$	Set of obstacle points
R_s	Maximum sensing range
R_o	Minimum distance to obstacle
R_c	Minimum inter-robot distance
$\mathcal{N}_i$	Neighbors of i-th robot
$\mathcal{G}$	Interaction graph
λ_2	Second smallest eigenvalue of the sensor-based weighted graph Laplacian
f_i^λ	Generalized connectivity force
f_i^B	Damping force
f_i	Traveling force
z_i^k	k-th target of i-th robot
Δt_i^k	Amount of time to stay close to target z_i^k
R_z	Maximum distance to target when anchored
v_i^{cruise}	Maximum cruise speed
γ_i	Path to current target, starting from position of robot at time of computation
q_i^γ	Closest point of path from current position
d_i^γ	Length of remaining path
R_γ	Distance to path at which it should be re-planned
α_Λ	Weighting of position vs. velocity error
e_i	Absolute tracking error of i-th robot along path
(x_c, x_M)	Tracking error bounds for the traveling efficiency
Λ_i	Traveling efficiency of i-th robot (i.e., tracking error nonlinearly scaled to [0, 1] based on x_c, x_M)
$\hat{\Lambda}_p^i$	Estimation of the traveling efficiency of the 'prime traveler' by the i-th robot
Θ_i	Force direction alignment between connectivity and traveling force of i-th robot
σ	Weighting between the force direction alignment and the 'prime traveler' traveling efficiency
ρ_i	Downscaling factor of a 'secondary traveler', dependent on $\hat{\Lambda}_p^i$, Θ_i and σ

3.1 Notation and algorithm overview

As in any distributed design, several instances of the proposed algorithms run separately on each robot and locally exchange information with the 'neighboring' instances via communication. Each instance is split into two concurrent routines: a *planning algorithm* and a *motion control algorithm* whose pseudocodes are given in Algorithms 1 and 2, respectively. The planning algorithm acts at a higher level and performs the following actions:

- it processes the targets provided by the target generator,
- it generates the desired path to the current target, and
- it selects an appropriate motion control behavior (see later).

The motion control algorithm acts at a lower level by specifying the traveling force f_i as a function of the behavior and the planned path selected by the planning algorithm.[3]

The two algorithms have access to the same variables which are formally introduced as follows (see Fig. 1 for a graphical representation of some of these variables): the variable targetQueue$_i$ is filled online by the target generator and contains a list of future targets to be visited by the i-th

[3] The two routines can run at two different frequencies, typically slower for the planning loop and faster for the motion control loop.

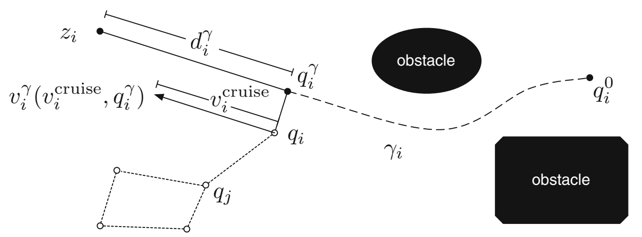

Fig. 1 Position q_i and path γ_i followed by a traveler from the point q_i^0 to the current target z_i. The *solid part* of the path represents the *remaining path* which starts at the closest point on the path q_i^γ and whose length is denoted by d_i^γ

robot. During the overall running time of the algorithms, the target generator of robot i has access to the whole list targetQueue$_i$ (which can also be changed online if needed). The current target for the i-th robot (i.e., the last target extracted from the first entry in targetQueue$_i$) is denoted with z_i. Variable γ_i is a $\bar{\mathcal{C}}^2$ geometric path that leads from the current position q_i of the i-th robot to the target z_i. In our implementation, we used B-splines (Biagiotti and Melchiorri 2008) in order to get a parameterized smooth path, but any other $\bar{\mathcal{C}}^2$ path would be appropriate. If the robot is not traveling towards any target, then γ_i is set to `null`.

With reference to Fig. 1, we also denote with q_i^γ the closest point of the path γ_i to q_i, i.e., the solution of arg $\min_{p \in \gamma_i} \|p - q_i\|$. In case of multiple solutions, we choose the one with the largest arc-length, i.e., the one nearest to the target along the path. Therefore, the closest point q_i^γ can be considered as unique in the following. The portion of the path γ_i from q_i^γ to z_i is referred to as the *remaining path*, and its length is denoted with d_i^γ.

The motion behavior of the i-th robot is determined by the variable state$_i$ that can take four possible values:

– `connector`
– `prime-traveler`
– `secondary-traveler`
– `anchor`.

The following provides a qualitative illustration of these motion behaviors, while a functional description is given in the next sections:

- *'connector'*: a robot in this state is not assigned any target by the target generator and therefore, its only goal is to help keeping the graph $\mathcal{G}$ connected. For this reason f_i is set to zero and hence the robot is subject solely to the damping and generalized connectivity force f_i^λ in (1);
- *'prime traveler'*: a robot in this state travels towards its current target z_i along the path γ_i thanks to the force f_i. At the same time, the robot distributively broadcasts to every other robot a non-negative real number, denoted with Λ_i, that represents its *traveling efficiency*, i.e., a measure how well it is able to follow its desired path while being influenced by the other robots in the group via the generalized connectivity force f_i^λ (which is described in more detail later). It is essential for the algorithm that only one 'prime traveler' exists in the group at any time. Every other robot with an assigned target needs to be a 'secondary traveler' or 'anchor'. This feature will allow one robot (the 'prime traveler') to reach its target with a high priority, while the other robots will only be allowed to reach their own targets as long as this action does not hinder the 'prime traveler' goal.
- *'secondary traveler'*: a robot in this state travels towards its current target z_i along the path γ_i thanks to the force f_i. The robot keeps an internal estimation $\hat{\Lambda}_p^i$ of the traveling efficiency of the current 'prime traveler', and it scales down the intensity of its traveling force f_i by an *adaptive gain* ρ_i whenever the action of f_i is 'too conflicting' w.r.t. that of f_i^λ, or the 'prime traveler' $\hat{\Lambda}_p^i$ drops lower than a given threshold.
- *'anchor'*: a robot in this state has reached the proximity of the target z_i. The force f_i is then exploited in order to keep q_i within a circle of radius R_z centered at z_i (i.e., the robot is 'anchored' to the target), while waiting for the associated time Δt_i to elapse.

In order to obtain a better intuition of the roles of the robots, we suggest the reader to watch the "Empty Space" video available in the attached multimedia material.

To summarize this qualitative description, these behaviors are designed in such a way that the single 'prime traveler' approaches its target with the highest priority, the 'secondary travelers' approach their targets as long as they have enough spatial freedom by the generalized connectivity force, the 'anchors' stay close to the target until their task is completed, and the 'connectors' help the 'secondary travelers' in providing as much spatial freedom as possible while preserving the connectivity of the graph.

Whenever a robot moves, it may indirectly exert a certain generalized connectivity force on all its neighbors because of the properties of f_i^λ (i.e., for retaining generalized connectivity of the graph $\mathcal{G}$ (see Robuffo Giordano et al. (2013) and Appendix). This connectivity action can possibly conflict with the traveling force f_i, and also prevent, in the worst case, fulfilment of the multi-target exploration task (e.g., the group falls in a local minimum because two robots start traveling in opposite directions over too large distances, thus threatening connectivity maintenance).

Since the 'connectors' implement $f_i = 0$ by definition, they cannot directly hinder the 'prime traveler' motion. In other words, a group made by all 'connectors' and one 'prime traveler' would always allow the 'prime traveler' to reach its target. Presence of 'anchors' can instead block the 'prime

Procedure Start-up for Robot i

```
1  if targetQueue_i is empty then
2      $\gamma_i \leftarrow$ null ;
3      state_i $\leftarrow$ connector
4  else
5      Extract first target from targetQueue_i and save it as $z_i$ ;
6      $\gamma_i \leftarrow$ Shortest obstacle-free path from $q_i$ to $z_i$
7      Enroll in the list of Candidates to take part in the first
       distributed 'prime traveler' election
8      if $i = \arg\min_{j \in Candidates} d_j^{\gamma}$ then
9          state_i $\leftarrow$ prime-traveler
10     else
11         state_i $\leftarrow$ secondary-traveler
12 $\hat{\Lambda}_p^i \leftarrow 0$ ;
13 Run Algorithm 1 and Algorithm 2 in parallel;
```

traveler' because of the anchoring force which prevents them to move away from their targets. Nevertheless the anchoring phase can only last for a finite time Δt_i^k after which the 'anchor' changes state and is again free to move.

No such mechanism is instead present for the 'secondary travelers' which would constantly attempt to move along their paths with a ρ_i set to 1. As explained, if many robots are simultaneously traveling in arbitrary directions inside a cluttered environment, while also maintaining connectivity of $\mathcal{G}$, the overall group motion can potentially (and quite easily) fall into a local minimum. The idea behind the gain ρ_i is to then adaptively scale down the traveling force f_i of the 'secondary travelers' whenever either (i) the direction f_i deviates too much from the connectivity force f_i^{λ}, or (ii) the 'prime traveler' motion is nevertheless too obstructed by the actions of the other 'secondary travelers' in the group. Consequently, this gain $\rho_i \in [0, 1]$ is chosen so that the current 'prime traveler' can always reach its target, no matter the motion planned by the 'secondary travelers' in the group. A formal description of this concept will be given in Sect. 3.7.

3.2 Start-up phase

The Procedure 'Start-up for Robot i' performs the distributed initialization of the planning and motion control algorithms. Its pseudocode is quickly commented in the following.

At the beginning, if targetQueue$_i$ is empty then the path γ_i is set to `null` and state$_i$ to `connector` (line 3). Otherwise the first target from targetQueue$_i$ is extracted and saved in z_i. Then, the robot i computes a $\bar{\mathcal{C}}^2$ shortest and obstacle-free path γ_i that connects its current position q_i with z_i (line 6). This path is generated with a two-step optimization method: first, the known portion of the map is discretized into an equally spaced grid in 3D with a cell size of R_{grid}. A cell is marked as occupied whenever an obstacle lies inside a radius of R_{grid} around the cell. On this grid, a shortest path is found via A^*. Then, the waypoints obtained from A^* are approximated with a B-spline (Biagiotti and Melchiorri 2008) in order to remove corners from the path. We note that, depending on the smoothing parameter, this approximation is not guaranteed to leave enough clearance from surrounding obstacles. Obstacle avoidance is nevertheless ensured thanks to the presence of the generalized connectivity force that prevents any possible collisions by (possibly) locally adjusting the planned path when needed. As an alternative, one could also rely on the method proposed in Masone et al. (2012) for directly generating a smooth path with enough clearance from obstacles.

Subsequently, the robot takes part in the distributed election of the first 'prime traveler' (see Sect. 3.3). Depending on the outcome of this election, state$_i$ is set either to `prime-traveler` or `secondary-traveler`.

At the end of the initialization procedure, the estimate $\hat{\Lambda}_p^i$ of the traveling efficiency of the current 'prime traveler' is initialized to zero (line 12) for all robots, and the planning and motion control algorithms are both started (line 13).

3.3 Election of the 'prime traveler'

In a general election of a new 'prime traveler', the current 'prime traveler' triggers the election process (line 14 of Algorithm 1), to which every 'secondary traveler' replies with its index and remaining path length, in order to be taken into the list of candidates (line 23). Since this election is a low-frequency event, we chose to implement it via a simple flooding algorithm (Lim and Kim 2001). Although this solution complies with the requirement of being decentralized, one could also resort to'smarter' distributed techniques such as (Lynch 1997). The 'prime traveler' then waits for $2(N-1)$ steps to collect these replies, being $2(N-1)$ the maximum number of steps needed to reach every robot with flooding and obtain a reply. The winner of this election is then the robot with the shortest remaining path length d_i^{γ}, i.e., the robot solving $\arg\min_{j \in \text{Candidates}} d_j^{\gamma}$. In the unlikely event of two (or more) robots having exactly the same remaining path length, the one with the lower index is elected. During the whole election process, the 'prime traveler' keeps its role and only upon decision it abdicates by switching into the 'anchor' state. After announcing the winner, no 'prime traveler' exists in the short time interval (at most $N-1$ steps) until the announcement reaches the winning 'secondary traveler'. This winning robot then switches into the 'prime traveler' behavior. This mechanism makes sure that at most one 'prime traveler' exists at any given time.

The *first* election in the Start-up phase (see Sect. 3.2 and line 7 in Procedure 'Start-up for Robot i') is handled slightly differently. Instead of the current 'prime traveler' organizing the election, robot 1 is always assigned the role of host and, instead of the only 'secondary travelers' replying, every robot

Algorithm 1: Planning for Robot i

```
while true do
    switch state_i do
        case connector
            if targetQueue_i is not empty then
                Extract the next target from targetQueue_i and
                save it as $z_i$
                $\gamma_i \leftarrow$ Shortest obstacle-free path from $q_i$ to $z_i$
                if There is no 'prime traveler' in the group then
                    state_i ←prime-traveler
                else
                    state_i ←secondary-traveler
        case prime-traveler
            if $\|q_i - z_i\| < R_z$ then
                $\gamma_i \leftarrow$ null
                Permit 'prime traveler' candidacy within timeout
                state_i ← anchor
        case secondary-traveler
            if $\|q_i - q_i^\gamma\| > R_\gamma$ then
                $\gamma_i \leftarrow$ Shortest obstacle-free path from $q_i$ to $z_i$ ;
            if $\|q_i - z_i\| < R_z$ then
                $\gamma_i \leftarrow$ null
                state_i ← anchor
            else if 'prime traveler' candidacy is allowed then
                Enroll in the list of Candidates to take part in the
                distributed 'prime traveler' election
                if $i = \arg\min_{j \in \text{Candidates}} d_j^\gamma$ then
                    state_i ← prime-traveler
        case anchor
            if task at target $z_i$ is completed then
                state_i ← connector
```

with an assigned target replies with its index and remaining path length (including robot 1 if it has an assigned target).

3.4 Planning algorithm

In this section, we describe in detail the execution of Algorithm 1 running on the i-th robot, whose logical flow is provided in Figure 2 as a graphical representation. The algorithm consists of a continuous loop where different decisions are taken according to the value of state$_i$ and according to the following different behaviors:

3.4.1 Case `connector`

If state$_i$ is set to `connector` then targetQueue$_i$ is checked. In case of an empty queue, state$_i$ remains `connector`, otherwise the next target is extracted from the queue and saved in z_i (line 5). Then the i-th robot computes a $\bar{C}^2$ shortest and obstacle-free path γ_i connecting the current robot position q_i with z_i (line 6) implementing what was previously described in the start-up procedure. Finally, the robot

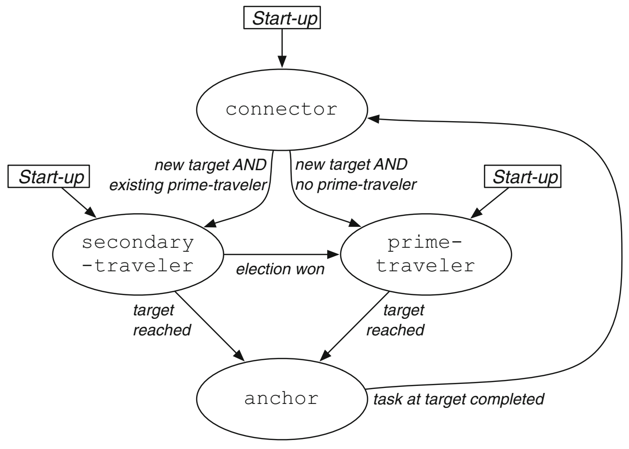

Fig. 2 State machine of the Algorithm 1

changes the value of state$_i$ in order to track γ_i. In particular, if no 'prime traveler' is present in the group, then state$_i$ is set to `prime-traveler` (line 8). Otherwise, state$_i$ is set to `secondary-traveler` (line 10).[4]

3.4.2 Case `prime-traveler`

When state$_i$ is set to `prime-traveler` (line 11) and the current position q_i is closer than R_z to the target z_i (line 12), the following actions are performed:

- the path γ_i is reset to `null` (line 13),
- a new distributed 'prime traveler' election as described in Sect. 3.3 is announced (line 14),
- the robot abdicates the role of 'prime traveler' and state$_i$ is set to `anchor` (line 15).

If, otherwise, z_i is still far from the current robot position q_i, then state$_i$ remains unchanged and the robot continues to travel towards its target.

3.4.3 Case `secondary-traveler`

When state$_i$ is `secondary-traveler` (line 16) the distance $\|q_i^\gamma - q_i\|$ to the (closest point on the) path is checked (line 17). If this distance is larger than the threshold R_γ, the robot replans a path from its current position q_i (line 18). This re-planning phase is necessary since a 'secondary traveler' could be arbitrarily far from the previously planned γ_i because of the 'dragging action' of the current 'prime traveler'. Section 3.6 will elaborate more on this point. Subsequently, if q_i is closer than R_z to the target z_i (line 19), the path γ_i is reset to `null` (line 20) and state$_i$ is set to `anchor`

[4] Presence of a 'prime traveler' can be easily assessed in a distributed way by, e.g., flooding (Lim and Kim 2001) on a low frequency.

(line 21). Otherwise, if the target is still far away, the robot checks whether the 'prime traveler' abdicated and announced an election of a new 'prime traveler' (line 22). If this was the case, the robot takes part in the election (line 23) as described in Sect. 3.3. If the robot wins the election (line 24), state_i is set to `prime-traveler` (line 25) otherwise it remains set to `secondary-traveler`.

3.4.4 Case `anchor`

The last case of Algorithm 1 is when state_i is `anchor` (line 26). The robot remains in this state until the task at target z_i is completed (line 27), after which state_i is set to `connector`.

3.5 Completeness of the planning algorithm

Before illustrating the *motion control algorithm*, we state some important properties that hold during the whole execution of the planning algorithm.

Proposition 1 *If there exists at least one target in one of the* $\mathsf{targetQueue}_i$, *then exactly one 'prime traveler' will be elected at the beginning of the operation. Furthermore, this 'prime traveler' will keep its state until being closer than* R_z *to its assigned target. In the meantime no other robot can become 'prime traveler'.*

Proof The start-up procedure guarantees that, if there exists at least one target in at least one of the $\mathsf{targetQueue}_i$, the group of robots includes exactly one 'prime traveler' and no 'anchor' at the beginning of the task. Any other robot is either a 'connector' or 'secondary traveler' depending on the corresponding availability of targets. During the execution of Algorithm 1, a robot can only switch into 'prime traveler' when being a 'connector' or a 'secondary traveler'. As long as there exists a 'prime traveler' in the group, a 'connector' cannot become a 'prime traveler'. Furthermore, a 'secondary traveler' becomes a 'prime traveler' only if it wins the election announced by the 'prime traveler'. Since the 'prime traveler' allows for this election only when in the vicinity of its target (within the radius R_z), the claim directly follows. □

Using this result, the following proposition shows that Algorithm 1 is actually guaranteed to complete the multi-target exploration in the following sense: when presented with a finite amount of targets, all targets of all robots are guaranteed to be visited in a finite amount of time. In order to show this result, an assumption on the robot motion controller is needed.

Assumption 1 In a group of robots with exactly one 'prime traveler', the adopted motion controller is such that the 'prime traveler' is able to arrive closer than R_z to its target in a finite amount of time regardless of the location of the targets assigned to the other robots.

In Sect. 3.7 we discuss in detail how the motion controller introduced in the next Sect. 3.6 meets Assumption 1.

Proposition 2 *Given a finite number of targets and a motion controller fulfilling Assumption 1, the whole multi-target exploration task is completed in a finite amount of time as long as the local tasks at every target can be completed in finite time.*

Proof In the trivial case of no targets, the multi-target exploration task is immediately completed. Let us then assume presence of at least one target. Proposition 1 guarantees existence of exactly one 'prime traveler' at the beginning of the planning algorithm, and that such a 'prime traveler' will keep its role until reaching its target, an event that, by virtue of Assumption 1, happens in finite time. At this point, assuming as a *worst case* that no 'secondary traveler' has reached and cleared its own target in the meantime, one of the following situations may arise:

1. There is at least one 'secondary traveler'. The 'secondary traveler' closest to its target becomes the new 'prime traveler' in the triggered election, and it then starts traveling towards its newly assigned target until reaching it in finite time (Proposition 1 and Assumption 1).
2. There is no 'secondary traveler' and no other 'anchor' besides the former 'prime traveler'. In this case, no other robot has targets in its queue as, otherwise, at least one 'secondary traveler' would exist. Therefore, after completing its task at the target location (a finite duration), the former 'prime traveler' and now 'anchor' becomes 'connector' and, in case of additional targets present for this robot, it switches back into being a 'prime traveler' and travels towards the new targets in a finite amount of time as in case 1.
3. There is no 'secondary traveler', but at least one other 'anchor'. This situation can be split again into two subcases:
   (a) there exists at least one 'anchor' with a future target in its queue. Then, after a finite time, this 'anchor' becomes 'secondary traveler' and case 1 holds;
   (b) there is no 'anchor' with a future target in its queue. Then, after a finite time, all 'anchors' have completed their local tasks and case 2 holds.

In all cases, therefore, one target is visited in finite time by the current 'prime traveler'. Repeating this loop finitely many times, for all the (finite number of) targets, allows to conclude that *all* targets will be visited in a finite amount of time, thus showing the completeness of the planning algorithm.

Algorithm 2: Motion Control for Robot i

```
1  while $true$ do
2      switch $state_i$ do
3          case connector
4              Update $\hat{\Lambda}^i_p$ using $\dot{\hat{\Lambda}}^i_p = k_\Lambda \sum_{j \in \mathcal{N}_i} (\hat{\Lambda}^j_p - \hat{\Lambda}^i_p)$
5              $f_i \leftarrow 0$
6          case prime-traveler
7              $\hat{\Lambda}^i_p \leftarrow \Lambda_i$ using (9)
8              $f_i \leftarrow f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}})$, using (3)
9          case secondary-traveler
10             Update $\hat{\Lambda}^i_p$ using $\dot{\hat{\Lambda}}^i_p = k_\Lambda \sum_{j \in \mathcal{N}_i} (\hat{\Lambda}^j_p - \hat{\Lambda}^i_p)$
11             $f_i \leftarrow \rho_i f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}})$, using (3) and (15)
12         case anchor
13             Update $\hat{\Lambda}^i_p$ using $\dot{\hat{\Lambda}}^i_p = k_\Lambda \sum_{j \in \mathcal{N}_i} (\hat{\Lambda}^j_p - \hat{\Lambda}^i_p)$
14             $f_i \leftarrow f_{\text{anchor}}(q_i, z_i, R_z)$, as per (5)
```

If a 'secondary traveler' already reaches its target while the 'prime traveler' is active, the aforementioned worst case assumption is not valid anymore. But since, in this case, the target of the 'secondary traveler' is already cleared, the total number of iterations is even smaller than in the previous worst case, thus still resulting in a finite completion time. □

3.6 Motion control algorithm

With reference to Algorithm 2, we now describe the motion control algorithm that runs in parallel to the planning algorithm on the i-th robot, and whose goal is to determine a traveling force f_i that can meet Assumption 1. The algorithm consists of a continuous loop, as before, in which the force f_i is computed according to the behavior encoded in the variable state_i determined by Algorithm 1:

3.6.1 Case `connector`

If state_i is set to `connector`, the estimate $\hat{\Lambda}^i_p$ of the traveling efficiency of the current 'prime traveler' is updated with a consensus-like algorithm (line 4) that will be described in the next Sect. 3.7. The traveling force f_i is in this case simply set to 0 (line 5). It is worth mentioning that $f_i = 0$ does not mean the i-th robot will not move, since a 'connector' is still dragged by the other travelers via the generalized connectivity force (according to (1), the 'connectors' are still subject to f_i^λ and f_i^B).

3.6.2 Case `prime-traveler`

If state_i is set to `prime-traveler`, the estimate $\hat{\Lambda}^i_p$ is set to the true traveling efficiency Λ_i, defined by (9) (line 7). Afterwards (line 8) the robot sets

$$f_i = f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}}), \tag{2}$$

where $f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}}) \in \mathbb{R}^3$ is a proportional, derivative and feedforward controller meant to travel along γ_i at a given cruise speed v_i^{cruise}:

$$\begin{aligned} f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}}) = a_i^\gamma(v_i^{\text{cruise}}, q_i^\gamma) \\ + k_v(v_i^\gamma(v_i^{\text{cruise}}, q_i^\gamma) - \dot{q}_i) \\ + k_p(q_i^\gamma - q_i). \end{aligned} \tag{3}$$

Here, k_p and k_v are positive gains, q_i^γ is the point on γ_i closest to q_i (see Fig. 1), $v_i^\gamma(v_i^{\text{cruise}}, q_i^\gamma)$ is the velocity vector of a virtual point traveling along γ_i and passing at q_i^γ with tangential speed v_i^{cruise}, and $a_i^\gamma(v_i^{\text{cruise}}, q_i^\gamma)$ is the acceleration vector of the same point. It is straightforward to analytically compute both the velocity and the acceleration from v_i^{cruise}, given the spline representation of the curve (Biagiotti and Melchiorri 2008).

3.6.3 Case `secondary-traveler`

If state_i is set to `secondary-traveler`, the estimate $\hat{\Lambda}^i_p$ is updated with a consensus-like protocol (line 10). Then (line 11) the robot sets

$$f_i = \rho_i f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}}), \tag{4}$$

where f_{travel} is defined as in (3) and $\rho_i \in [0, 1]$ is an adaptive gain meant to scale down the intensity of the action of $f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}})$ whenever *(i)* its alignment is too conflicting with the generalized connectivity force f_i^λ or *(ii)* the 'prime traveler' is not able to efficiently travel along its path because its reached speed is too low compared to its desired cruise speed. Section 3.7 is dedicated to provide details on choosing an effective ρ_i.

3.6.4 Case `anchor`

If state_i is set to `anchor`, the estimate $\hat{\Lambda}^i_p$ is again updated using a consensus-like protocol (line 13). Then (line 14) the force f_i is set as

$$f_i = f_{\text{anchor}}(q_i, z_i, R_z) = -\frac{\partial V^{R_z}_{\text{anchor}}(\|q_i - z_i\|)}{\partial q_i} \tag{5}$$

where $V^{R_z}_{\text{anchor}} : [0, R_z) \rightarrow [0, \infty)$ is a monotonically increasing potential function of the distance $\ell_i = \|q_i - z_i\|$ between the robot position q_i and the target z_i, and such that $V^{R_z}_{\text{anchor}}(0) = 0$ and $\lim_{\ell_i \nearrow R_z} V^{R_z}_{\text{anchor}}(\ell_i) = \infty$. Under the action of $f_{\text{anchor}}(q_i, z_i, R_z)$ the position q_i is then guaranteed to remain confined within a sphere of radius R_z centered at

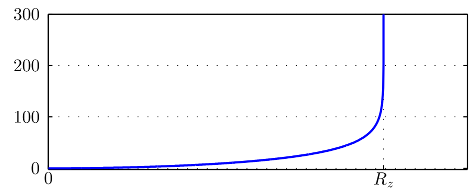

Fig. 3 Shape of the function $V^{R_z}_{\text{anchor}}(\ell_i)$ defined in (6) that is 0 on the target z_i itself ($\ell_i = 0$) and grows unbounded at the border of a sphere with radius R_z

z_i until the local task at the target location is completed. In our simulations and experiments we employed

$$V^{R_z}_{\text{anchor}}(\ell_i) = -k_z \frac{2R_z}{\pi} \ln\left(\cos\left(\frac{\ell_i \pi}{2R_z}\right)\right) \tag{6}$$

where k_z is an arbitrary positive constant. The shape of this function is shown in Fig. 3 and the associated f_{anchor} is

$$f_{\text{anchor}}(q_i, z_i, R_z) = -k_z \tan\left(\frac{\ell_i \pi}{2R_z}\right) \frac{q_i - z_i}{\ell_i}\,. \tag{7}$$

3.7 Traveling efficiency, force alignment and adaptive gain

We now describe how the estimation of the *traveling efficiency* Λ_i of all robots and the *adaptive gain* ρ_i of a ‘secondary traveler’, used in Algorithm 2, are actually computed. Remember that the idea behind the gain ρ_i is to adaptively scale down the traveling force f_i of a ‘secondary traveler’ whenever *(i)* the alignment of f_i and the generalized connectivity force f_i^λ is too different, or *(ii)* the traveling efficiency of the ‘prime traveler’ is too low.

Therefore the design of ρ_i aims at guaranteeing that the current ‘prime traveler’ can always reach its target, whatever the motion planned by the other robots in the group are, and thus in fact enforces Assumption 1.

We recall that we provide a compendium of all important variables in Table 1.

In order to implement the desired behavior we introduce two functions:

$$\Theta : \mathbb{R}^3 \times \mathbb{R}^3 \to [0, 1]$$
$$\Lambda : \mathbb{R}^+_0 \times K^* \to [0, 1]$$

where $K^* = \{(x_c, x_M) \in \mathbb{R}^2 \mid 0 \le x_c < x_M\}$, defined as:

$$\Theta(x, y) = \begin{cases} \frac{1}{2}\left(1 + \frac{x^T y}{\|x\|\|y\|}\right) & x \neq 0, y \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

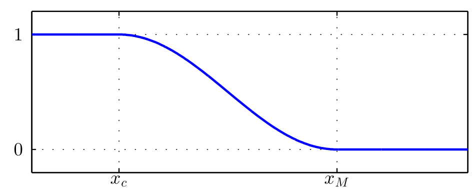

Fig. 4 Sketch of the function $\Lambda(x, x_c, x_M)$ for fixed x_c and x_M

$$\Lambda(x, x_c, x_M) = \begin{cases} 1 & x \in [0, x_c] \\ \frac{1}{2} + \frac{\cos\left(\frac{x - x_c}{x_M - x_c}\pi\right)}{2} & x \in (x_c, x_M) \\ 0 & x \in [x_M, \infty). \end{cases} \tag{9}$$

Function $\Theta(x, y)$ represents a ‘measure’ of the direction alignment of the two non-zero 3D vectors x and y. In particular, $\Theta(x, y)$ is 1 if x and y are parallel with the same direction, $\frac{1}{2}$ if they are orthogonal, and 0 if they are parallel with opposite direction. Note that $\Theta(x, y)$ is equivalent to $\frac{1}{2}(1 + \cos\theta)$ with θ being the angle between vectors x and y.

Function $\Lambda(x, x_c, x_M)$ ‘measures’ how small x is. If $x \le x_c$ then x is considered ‘small enough’ and, therefore, $\Lambda = 1$. If $x \in (x_c, x_M)$ then Λ strictly monotonically varies from 1 to 0. If $x \ge x_M$, then $\Lambda = 0$. The shape of Λ is depicted in Fig. 4.

Having introduced these functions, we now define the *force direction alignment* of the i-th robot as

$$\Theta_i = \Theta(f_i^\lambda, f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}})), \tag{10}$$

and note that Θ_i can be locally computed by the i-th robot. The quantity Θ_i thus represents an index in [0, 1] measuring the degree of conflict among the directions of the generalized connectivity force and the traveling force.

When $\gamma_i \neq \texttt{null}$, we also define the absolute tracking error as

$$e_i = (1 - \alpha_\Lambda)\|v_i^\gamma(v_i^{\text{cruise}}, q_i^\gamma) - v_i\| + \alpha_\Lambda\|q_i^\gamma - q_i\|, \tag{11}$$

with $\alpha_\Lambda \in [0, 1]$ being a constant parameter modulating the importance of the velocity tracking error w.r.t. the position tracking error. The *traveling efficiency* is then defined as

$$\Lambda_i = \Lambda\left(e_i, x_c, x_M\right), \tag{12}$$

where $0 \le x_c < x_M < \infty$ are two user-defined thresholds representing the point at which the traveling efficiency Λ_i starts to decrease and the maximum tolerated error after which the traveling efficiency vanishes. In this way it is possible to evaluate how well a traveler can follow its desired planned path according to a suitable combination of velocity and position accuracy. It is important to note that the value $\Lambda_i = 1$ does not imply an exact tracking of the path, but it still allows a small tracking tolerance (dependent on the parameter x_c). Similarly, the value $\Lambda_i = 0$ does not imply a complete loss of path tracking, but it represents the possibility of a tracking error higher than a maximum threshold (dependent on x_M).

In order to meet Assumption 1, we are only interested in the traveling efficiency of the current 'prime traveler' for monitoring whether (and how much) its exploration task is held back by the presence/motion of the 'secondary travelers'. From now on we then denote this value as Λ_p, where

$$p = i \quad s.t. \;\; \text{state}_i = \texttt{prime-traveler}.$$

This quantity is not in general locally available to every robot in the group, and therefore a simple decentralized algorithm is used for its propagation to avoid a flooding step. Among many possible choices we opted for using the following well-known consensus-based propagation (Olfati-Saber and Murray 2003):

$$\begin{aligned} \dot{\hat{\Lambda}}{}^i_p &= k_\Lambda \sum_{j \in \mathcal{N}_i} (\hat{\Lambda}^j_p - \hat{\Lambda}^i_p) && \text{if } i \neq p \\ \hat{\Lambda}^i_p &= \Lambda_i && \text{if } i = p. \end{aligned} \tag{13}$$

This distributed estimator lets $\hat{\Lambda}^i_p$ track Λ_p for all i that hold $\text{state}_i \neq \texttt{prime-traveler}$ with an accuracy depending on the chosen gain k_Λ. Notice that, for a constant Λ_p, the convergence of this estimation scheme is exact. Furthermore, since $\Lambda_p \in [0, 1]$, $\hat{\Lambda}^i_p$ is then saturated so as to remain in the allowed interval despite the possible transient oscillations of the estimator. Instead of this simple consensus, one could also resort to a PI average consensus estimator (Freeman et al. 2006) to cope with presence of a time-varying signal. However, for simplicity we relied on a simple consensus law with less parameters to be tuned, and with, nevertheless, a satisfying performance as extensively shown in our simulation and experimental results.

Hence, every 'secondary traveler' can locally compute Θ_i and build an estimation $\hat{\Lambda}^i_p$ of Λ_p. In order to consolidate these two quantities into a single value, we define the function $\rho : [0, 1] \times [0, 1] \times [1, \infty) \to [0, 1]$ as:

$$\rho(x, y, \sigma) = (1 - x)\, y^\sigma + x \left(1 - (1 - y)^\sigma\right), \tag{14}$$

where $1 \le \sigma < \infty$ is a constant parameter. Gain ρ_i is then obtained from Θ_i and $\hat{\Lambda}^i_p$ as

$$\rho_i = \rho(\Theta_i, \hat{\Lambda}^i_p, \sigma) \tag{15}$$

with $1 \le \sigma < \infty$ being a tunable parameter.

The reasons motivating this design of gain ρ_i are as follows: ρ_i is a smooth function of Θ_i and $\hat{\Lambda}^i_p$ possessing the following desired properties (see also Fig. 5)

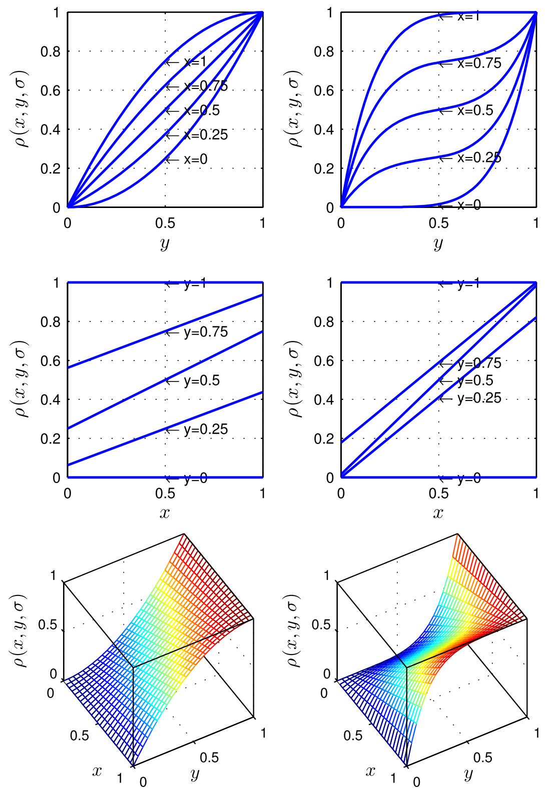

Fig. 5 Function ρ for $\sigma = 2$ (*left side*) and $\sigma = 6$ (*right side*). The motion controller exploits this function by plugging the force direction alignment in the x argument, and the estimate of the traveling efficiency of the current 'prime traveler' in the y argument

1. $\hat{\Lambda}^i_p = 1 \Rightarrow \rho_i = 1$: if the traveling efficiency of the 'prime traveler' is 1 then every 'secondary traveler' sets $f_i = f_{\text{travel}}(q_i, \gamma_i, v_i^{\text{cruise}})$;
2. $\hat{\Lambda}^i_p = 0 \Rightarrow \rho_i = 0$: if the traveling efficiency of the 'prime traveler' is 0 then every 'secondary traveler' sets $f_i = 0$;
3. ρ_i monotonically increases w.r.t. $\hat{\Lambda}^i_p$ for any Θ_i and σ in their domains;
4. ρ_i constantly increases w.r.t. Θ_i for any $\hat{\Lambda}^i_p \in (0, 1)$ and $\sigma > 1$;
5. if $\sigma = 1$ then $\rho_i = \hat{\Lambda}^i_p$ for any $\Theta_i \in [0, 1]$
6. if $\sigma \to \infty$ then $\rho_i \to \Theta_i$ for any $\hat{\Lambda}^i_p \in (0, 1)$.

Summarizing, gain ρ_i mixes the information of both the force direction alignment and the traveling efficiency of the 'prime traveler', with more emphasis on the first or the second term depending on the value of the parameter σ. Nevertheless, the traveling efficiency $\hat{\Lambda}^i_p$ is always predominant at its boundary values (0 and 1) regardless of the value of σ. This means that, whenever the estimated travel efficiency of the 'prime traveler' is $\hat{\Lambda}^i_p = 0$ and robot i is a 'secondary traveler', its traveling force is scaled to zero and, therefore, robot i only becomes subject to the connectivity and damping force. Therefore, in this situation the motion of all 'secondary travelers' results dominated by the 'prime traveler', which is then able to execute its planned path towards its target location. On the other hand, when $\hat{\Lambda}^i_p = 1$, the 'prime traveler' has a sufficiently high traveling efficiency despite the 'secondary traveler' motions. Therefore, every 'secondary traveler' is free to travel along its own planned path regardless of the direction alignment between traveling and connectivity force.

We conclude noting that the main goal of the machinery defined in Sects. 3.6 and 3.7 is to ensure that the motion controller meets the requirements defined in Assumption 1. Although some of the steps involved in the design of the traveling force f_i have a 'heuristic' nature, the proposed algorithm is quite effective in solving the multi-target exploration task (in a decentralized way) under the constraint of connectivity maintenance, as proven by the several simulation and experimental results reported in the next section.

4 Simulations and experiments

In this section, we report the results of an extensive simulative and experimental campaign meant to illustrate and validate the proposed method. The videos of the simulations and experiments can be watched in the attached material and on http://homepages.laas.fr/afranchi/videos/multi_exp_conn.html.

All the simulation (and experimental) results were run in 3D environments, although only a 2D perspective is reported in the videos for the simulated cases (therefore, robots that may look as 'colliding' are actually flying at different heights, since their generalized connectivity force prevents any possible inter-robot collision).

As robotic platform in both simulations and experiments we used small quadrotor Unmanned Aerial Vehicles (UAVs) with a diameter of 0.5 m. This choice is motivated by the versatility and construction simplicity of these platforms, and also because of the good match with our assumption of being able to track any sufficiently smooth linear trajectory in 3D space.

We further made use of the SwarmSimX environment (Lächele et al. 2012), a physically-realistic simulation software. The simulated quadrotors are highly detailed models of the real quadrotors later employed in the experiments. The physical behavior of the robots itself and their interaction with the environment is simulated in real-time using PhysX[5]

For the experiments, we opted for a highly customized version of the MK-Quadro[6] We implemented a software on the onboard microcontroller able to control the orientation of the robot by relying on the integrated inertial measurement unit. The desired orientation is provided via a serial connection by a position controller implemented within the ROS framework[7] that can run on any generic GNU-Linux machine. The machine can be either mounted onboard or acting as a base-station. In the latter case a wireless serial connection with XBees[8] is used. We opted for the separate base station in order to extend the flight time thanks to the reduction of the onboard weight. The current UAV position used by the controller is retrieved from a motion capturing system,[9] while obstacles are defined statically before the task execution.

To abstract from simulations and experiments, we used the TeleKyb software framework, which is thoroughly described in Grabe et al. (2013). Finally, the desired trajectory (consisting of position, velocity and acceleration) is generated by our decentralized control algorithm implemented using Simulink[10] running in real-time at 1 kHz.

4.1 Monte Carlo simulations

The proposed method has been extensively evaluated through randomized experiments in three significantly different scenarios. The first scenario is an obstacle free 3D space and three snapshots of the evolution of the proposed algorithm are presented in Fig. 6. The second, a more complex, scenario includes a part of a town and is reported in Fig. 7. The third is an office-like environment shown in Fig. 8. The size of the environments is 50 m × 70 m for both the empty space and the town, and about 10 m × 15 m for the office. Since the first two environments are outdoor scenarios and the office-like environment is indoor, two different sets of parameters were employed in the simulations. The values of the main parameters are listed in Table 2.

The number of robots varied from 10 to 35. In every trial 3 targets are sequentially assigned to 5 robots and 2 targets are sequentially assigned to other 5 robots, for a total of 25 targets per trial. The remaining robots are given no targets (i.e., they act always as 'connectors').

The configuration of the given targets is randomized across the different trials. The same random configurations are repeated for every different number of robots in order to allow for a fair comparison among the results. In the following we refer to the robots with at least one target assigned during a trial as 'explorers'.

To summarize, we simulated a total number of 1800 trials arranged in the following way: in each of the 3 scenes, and for each of the 100 target configurations in each scene, we ran

5 http://www.geforce.com/hardware/technology/physx.

6 http://www.mikrokopter.com.

7 http://www.ros.org.

8 http://www.digi.com/lp/xbee.

9 http://www.vicon.com.

10 http://www.mathworks.com/products/simulink/.

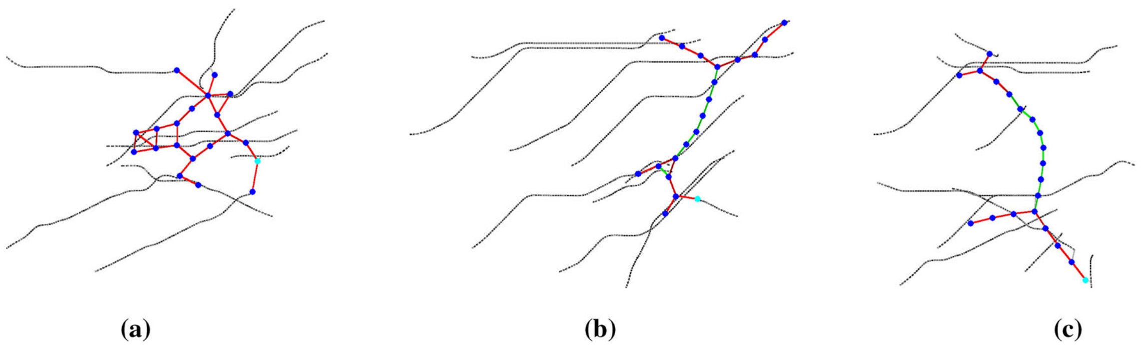

Fig. 6 Snapshots of a simulation with 20 UAVs in empty space in three different consecutive time instants. The *dotted black curves* represent the planned path γ_i to the current target for each robot i (if it has a current target). *Blue dots* are the robots, the turquoise dot is the current 'prime traveler'. *Line segments* represent the presence of a connection link between a pair of robots with the following color coding: *green*—well connected, *red*—close to disconnection. The robots are able to concurrently explore the given targets and continuously maintain the connectivity of the interaction graph

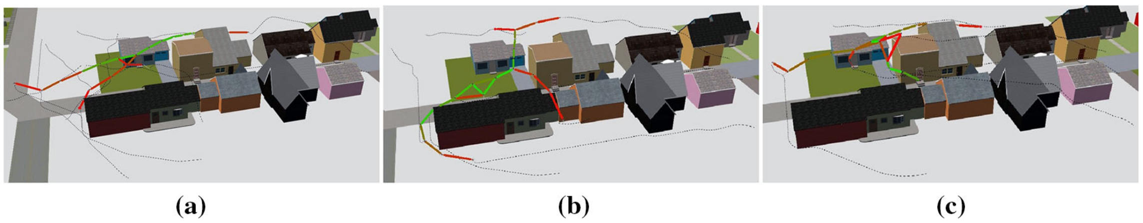

Fig. 7 Three snapshots of consecutive time instants of a simulation in the town environment. Graphical notation similar to Fig. 6

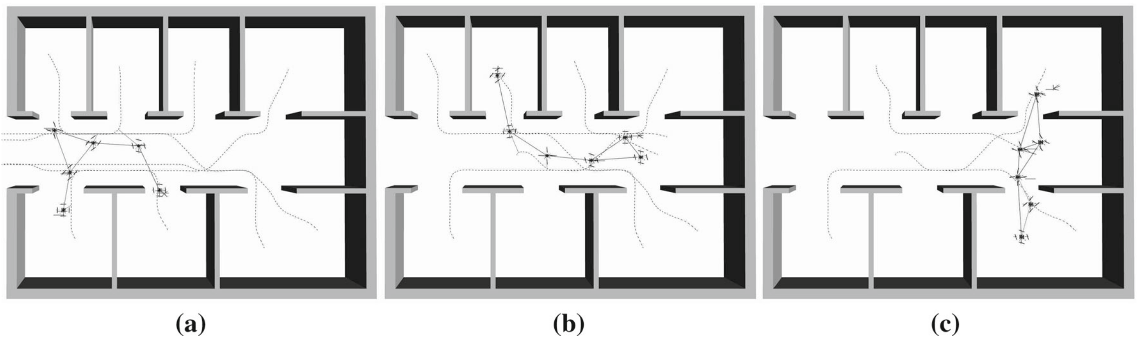

Fig. 8 Snapshots of a simulation in the office-like environment in three consecutive time instants. Graphical notation similar to Fig. 6

Table 2 Main parameters of the algorithm used in the 1800 randomized simulative trials in the different scenarios

Parameter	Empty space and town	Office
(R'_s, R_s)	(2.5 m, 6 m)	(1.1 m, 2.5 m)
(R_o, R'_o)	(0.75 m, 1.75 m)	(0.25 m, 0.6 m)
(R_c, R'_c)	(1 m, 2.5 m)	(0.8 m, 1.1 m)
$(\lambda_2^{\min}, \lambda_2^{\text{null}})$	(0, 1)	(0, 1)
R_{grid}	0.75 m	0.25 m
σ	3	3
v_i^{cruise}	3 m/s	1 m/s for all i
(x_c, x_M)	$(0.1, 0.6)v_i^{\text{cruise}}$	$(0.1, 0.6)v_i^{\text{cruise}}$
Δt_i^k	3 s for all i and k	3 s for all i and k
R_z	1.8 m	1 m

a simulation with 6 different numbers of robots, namely 10, 15, 20, 25, 30, 35. We encourage the reader to also watch the attached video where some representative simulative trials are shown.

In Fig. 9 we show the evolutions of the statistical percentiles of:

- the overall completion time,
- the mean traveled distance of the 10 'explorers',
- the maximum Euclidean distance between two 'explorers'
- the average of $\lambda_2(t)$ over time along the whole trial (we recall that the larger the λ_2 the more connected is the group of robots, refer to Appendix),

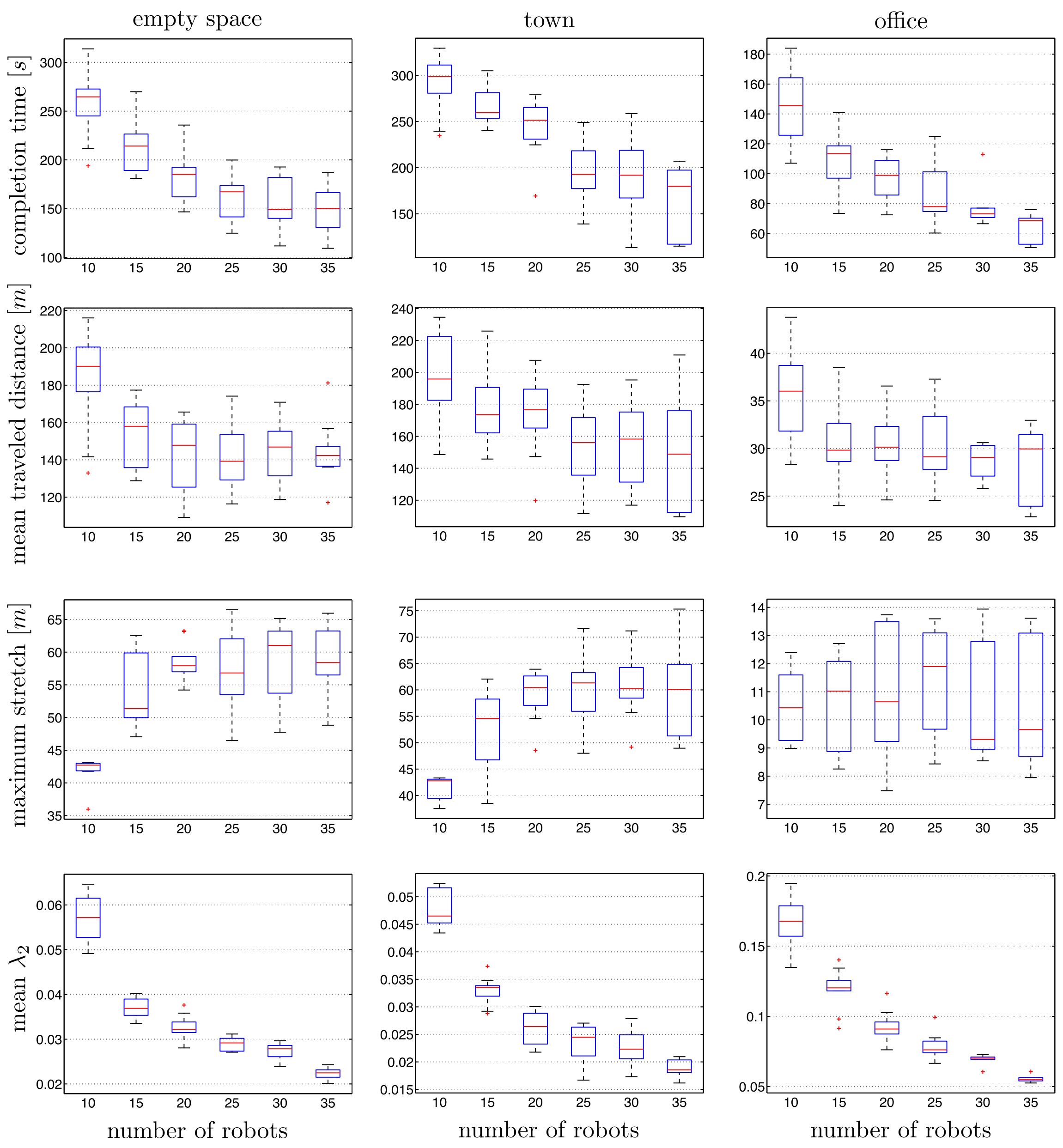

Fig. 9 Statistics of the completion times (*first row*), mean traveled distance of the traveling robots (*second row*), the maximum Euclidian distance between two traveling robots (*third row*) and mean λ_2 (*forth row*) versus the number of robots in the environments empty space (*left column*), town (*middle column*) and office (*right column*)

when the number of robots varies from 10 (i.e., no 'connectors') to 35 (i.e., 25 'connectors'). Each column refers to one of the 3 different scenarios.

An improvement with the increasing number of 'connectors' in all scenarios is obvious. The mean completion time (first row) roughly halves when comparing 0 to 25 'connectors'. Adding more than 25 connectors will likely produce only minor improvement compared to the higher cost of having more robots, since the trend becomes basically flat. For this reason we did not perform simulations with a larger number of robots.

In the second row (mean traveled distance) one can see how, by already adding a few robots, a reduced mean traveled distance is obtained. This can be explained by the fact that the 'connectors' make the 'explorers' less disturbed by other 'explorers' with, therefore, more freedom to avoid unnecessary detours in reaching their targets.

Another measure of the reduced task completion time is the maximum stretch among the 'explorers' (i.e., the maximum Euclidean distance between any two 'explorers', see third row). The more connectors, the more stretch is allowed: 'connectors' in fact provide the support needed by

Table 3 Main parameters used in the experiments

Parameter	Value
(R'_s, R_s)	$(1.4\,\text{m}, 2.5\,\text{m})$
(R_o, R'_o)	$(0.5\,\text{m}, 0.75\,\text{m})$
(R_c, R'_c)	$(1.0\,\text{m}, 1.4\,\text{m})$
$(\lambda_2^{\text{min}}, \lambda_2^{\text{null}})$	$(0, 1)$
R_{grid}	$0.2\,\text{m}$
σ	3
v_i^{cruise}	$0.5\,\text{m/s}$
(x_c, x_M)	$(0.2, 0.7)\ v_i^{\text{cruise}}$
Δt_i^k	3 s for all i and k
R_z	$0.75\,\text{m}$

the 'explorers' for keeping graph $\mathcal{G}$ connected while freely moving towards their targets. Only the office-like environment does not show this trend in the maximum stretch. This is due to the fact that the scene is relatively small and therefore the targets are not enough spread apart, so no bigger stretch is needed.

The increased freedom of the 'explorers' is also evident in the plots of the average $\lambda_2(t)$ (fourth row). These plots show how the 'connectors' are also useful to let the 'explorers' move more freely even in small environments. In fact, the larger the amount of 'connectors', the lower the mean λ_2: with more connectors the 'explorers' are more able to simultaneously travel towards their targets, thus bringing the topology of the group closer to less connected topologies (i.e., closer to tree-like topologies where the explorers would be the leaves of the tree). Clearly, this effect is independent of the maximum stretch, in fact the average λ_2 follows this decreasing trend also in the third office-like environment (third column).

4.2 Experiments

The experiments involved 6 real quadrotors and were meant to test the applicability of the algorithm in a real scenario. The parameters of the algorithm used in the experiments are reported in Table 3.

In order to obtain a $\bar{\mathcal{C}}^4$ trajectory smoother than $q_i(t)$ and, thus, better matching the dynamics capabilities of a quadrotor UAV (Mistler et al. 2001), we made use of a fourth order linear filter for each quadrotor:

$$\ddddot{q}_i^{\,f}(t) = -k_1\dddot{q}_i^{\,f}(t) - k_2\ddot{q}_i^{\,f}(t) - k_3\dot{q}_i^{\,f}(t) + k_4(q_i(t) - q_i^f(t)) \tag{16}$$

that tracks the position of the original trajectory $q_i(t)$, while keeping the velocity, acceleration, and jerk low in the filtered trajectory. The tunable gains were chosen as $k_1 = 44$, $k_2 = 707$, $k_3 = 5090$, $k_4 = 13692$ for placing the (real negative) poles at approximately $-12, -13, -14, -15$, then resulting in a settling time of about 0.3 s within a band of 5 %.

The resulting trajectory $q_i^f(t)$ is then provided in place of $q_i(t)$ as input trajectory for the robot i as defined in (1), since it results very close to $q_i(t)$ as shown in Fig. 10a. However, at the same time, it provides a much smoother reference position signal to the quadrotor by filtering off occasional abrupt motions, as can be seen in the velocity and acceleration reported in Fig. 10b and c. Figure 11 shows the norms of the UAV errors while tracking the desired trajectory $q_i^f(t)$. The average norm of all the quadrotors tracking errors during the whole experiment is 0.021 m, a few short peaks are above 0.06 m, and the highest peak is about 0.098 m.

For these experiments, we reproduced a scene similar to the office-like environment used in simulation, see Fig. 12. The UAVs with IDs '2' and '4' (called 'explorers') were given some targets, while the UAVs with IDs '1', '3', '5', and '6' ('connectors') had no target, for then a total of 6 quadrotors.

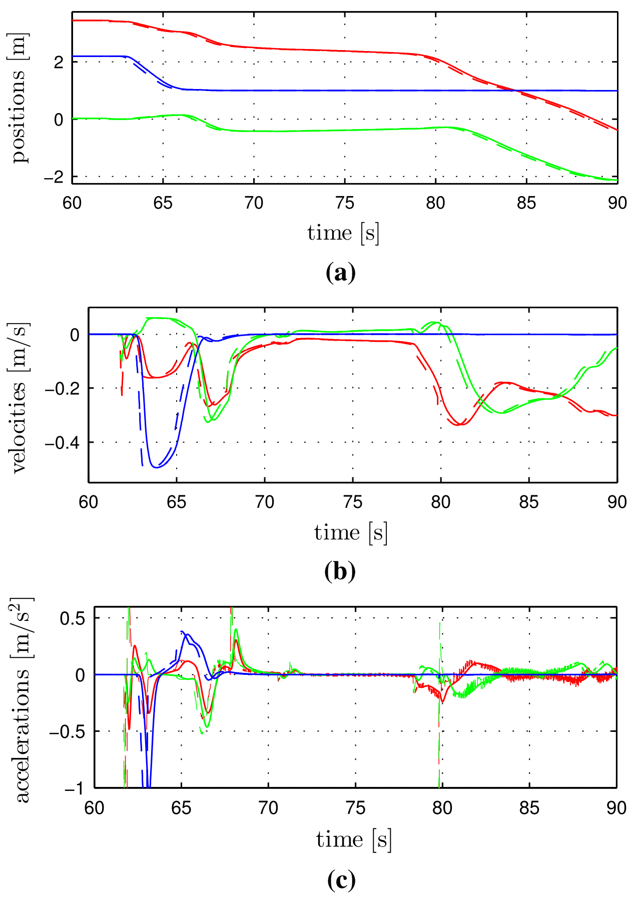

Fig. 10 Position, velocity and acceleration of 'explorer' 1 during a representative period of the experiment, where $q_i(t), \dot{q}_i(t), \ddot{q}_i(t)$ are plotted in *dash* and $q_i^f(t), \dot{q}_i^f(t), \ddot{q}_i^f(t)$ as *solid curves*. The x, y and z component is plotted in *red*, *green* and *blue* respectively

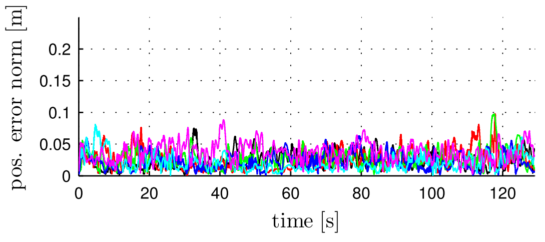

Fig. 11 Plots of the 6 norms of the position error between $q_i(t)$ and the corresponding real quadrotor trajectory, for $i = 1, \ldots, 6$. The average error norm is 0.021 m

The *'explorer' 1* (with ID '4') carries an onboard camera and has two targets in total. Whenever it reaches one of its targets it gives a human operator direct control of the vehicle in the surrounding area of the target. Then, with the help of the onboard camera, the human operator has the task of searching for an object in the environment. When the object is found by the human operator, the task at the target is considered completed, and the UAV switches back to autonomous control. In order to allow full human control of 'explorer' 1 in the anchoring behavior, the UAV is temporarily decoupled from the point q_4, which is instead kept close to the target by the action of f_{anchor} (as desired). The *'explorer' 2* (with ID '2') is instead fully autonomous and is assigned with a total of 4 targets. At the first target location, the task is to pick up an object to then be released at the second target location. The same task is subsequently repeated with targets 3 and 4. We note, however, that the pick and place action is only virtually performed since the employed quadrotors are not equipped with an onboard gripper. We also stress that all these operations are performed concurrently while keeping the topology of the group connected at all times.

A video of the experiment is present in the attached material and can be found under the given link above.

Table 4 reports and describes all the relevant events taking place during an experiment in a chronological order.

Figure 12 shows the top-view of the 'explorer' paths for five representative time periods: $T_1 = [0, 25]$ s in Fig. 12a, $T_2 = [25, 60]$ s in Fig. 12b, $T_3 = [60, 80]$ s in Fig. 12c, $T_4 = [80, 120]$ s in Fig. 12d, and finally $T_5 = [120, 129]$ s in Fig. 12e. Every plot shows the (connected) graph topology of the group at the beginning of the time interval (dashed black lines) and the paths of the 2 'explorers' (solid lines, blue for the 'explorer' 1 and red for the 'explorer' 2). The initial positions of the robots are shown with colored circles and are labeled with the IDs of the corresponding robots. The two small blue squares represent the two desired target locations of the 'explorer' 1. The two green squares and the two red squares represent the two pick positions and release positions of 'explorer' 2, respectively. Finally, the vertical walls of the environment are shown in gray. Figure 12f on the other hand shows the z-coordinate of all the six quadrotors

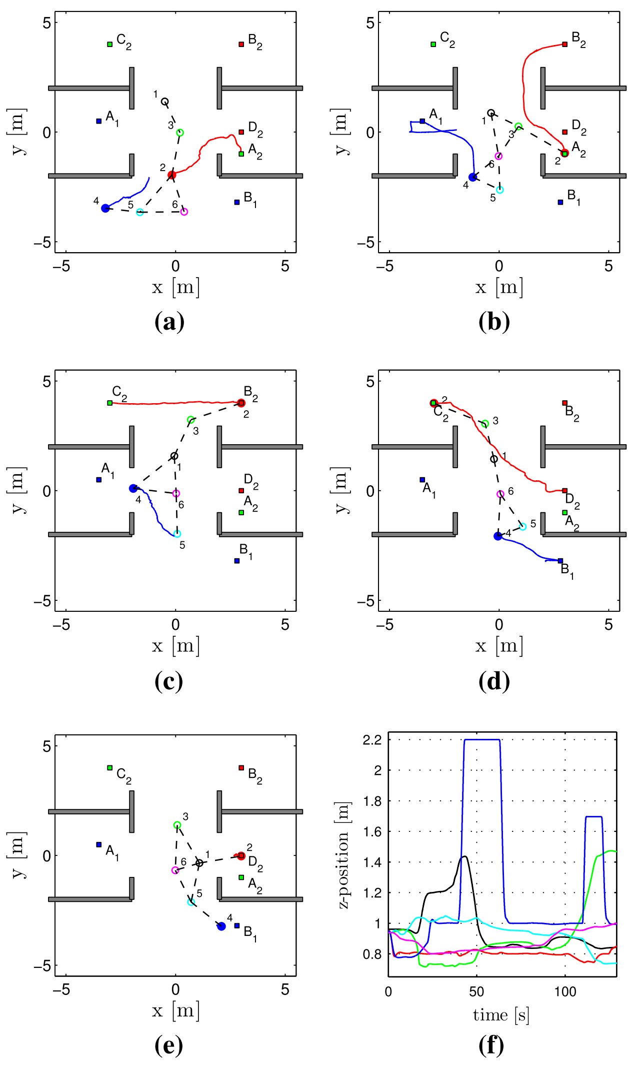

Fig. 12 **a–e** Top view of the 3D paths of the 'explorers' (*solid blue and red curves*) during the experiment in five representative time intervals. The interaction graph at the beginning of each interval is shown with *black dashed lines*. The ID of each robot is shown besides the *circle* representing the starting position of each robot at the beginning of the corresponding interval. Targets are represented with *colored squares* and walls are *gray*. The specific time intervals are: **a** $T_1 = [0, 25]$ s, **b** $T_2 = [25, 60]$ s, **c** $T_3 = [60, 80]$ s, **d** $T_4 = [80, 120]$ s and **e** $T_5 = [120, 129]$ s. **f** z-coordinate of the positions of all six quadrotors to help interpreting the 2D projection reported in the plots (and videos). The large vertical motion of 'explorer' 1 (*blue*) is due to the human operator flying this robot, while the subsequent descent is autonomously performed thanks to the proposed algorithm

in order to understand the 3D motion in the 2D projections of Fig. 12a to e.

Figure 13 shows three screenshots of the experiment: the lines between two quadrotors represent the corresponding connecting link as per graph $\mathcal{G}$.

Finally, Fig. 14 reports nine plots that capture the behavior of several quantities of interest throughout the whole experiment. As can be seen in Fig. 14a, the generalized algebraic

Table 4 Chronological list of important events in the experiment

Fig. 12	Time	Events
Fig. 12a	0 s	The experiment starts. Both ‘explorers’ are assigned a target and since ‘explorer’ 2 is closer to its goal, it becomes ‘prime traveler’, while ‘explorer’ 1 is ‘secondary traveler’
	22 s	‘Explorer’ 2 arrives at its first target, where it should pick up an object. Therefore ‘explorer’ 2 goes into ‘anchor’ and ‘explorer’ 1 becomes ‘prime traveler’
Fig. 12b	29 s	‘Explorer’ 2 has completed the pick-up action and receives the point to release the object as a new target. Since ‘explorer’ 1 is still ‘prime traveler’, ‘explorer’ 2 becomes ‘secondary traveler’
	35 s	‘Explorer’ 1 arrives at its target, where the human operator takes control of the UAV and use its camera to find a yellow picture on the wall. ‘Explorer’ 2 then becomes ‘prime traveler’
	56 s	‘Explorer’ 2 arrives at the target where it needs to release the object.
Fig. 12c	63 s	‘Explorer’ 2 has completed the releasing action and receives the next pick-up location. ‘Explorer’ 1 is still under the control of the human operator and therefore in an ‘anchor’ state, so ‘explorer’ 2 directly becomes ‘prime traveler’
	65 s	The human operator finds the picture on the wall, ‘explorer’ 1 becomes autonomous again and starts to move towards its next target as ‘secondary traveler’, since ‘explorer’ 2 is ‘prime traveler’
	78 s	‘Explorer’ 2 arrives at the location where to pick up the second object and goes to the ‘anchor’ state. Hence ‘explorer’ 1 becomes ‘prime traveler’
Fig. 12d	85 s	‘Explorer’ 2 has completed the pick-up action and starts moving towards the releasing location as ‘secondary traveler’
	100 s	‘explorer’ 1 arrives at its target, goes to ‘anchor’ state and is thus under control of the human operator, therefore ‘explorer’ 2 becomes ‘prime traveler’
	119 s	‘Explorer’ 2 arrives at its final target and switches into ‘anchor’
Fig. 12e	123 s	The human operator finds the searched object and ‘explorer’ 1 becomes ‘connector’ since it has no new target location
	126 s	‘Explorer’ 2 has completed the releasing action and becomes a ‘connector’ since it has also no new target
	129 s	No UAV has a next target and the experiment ends

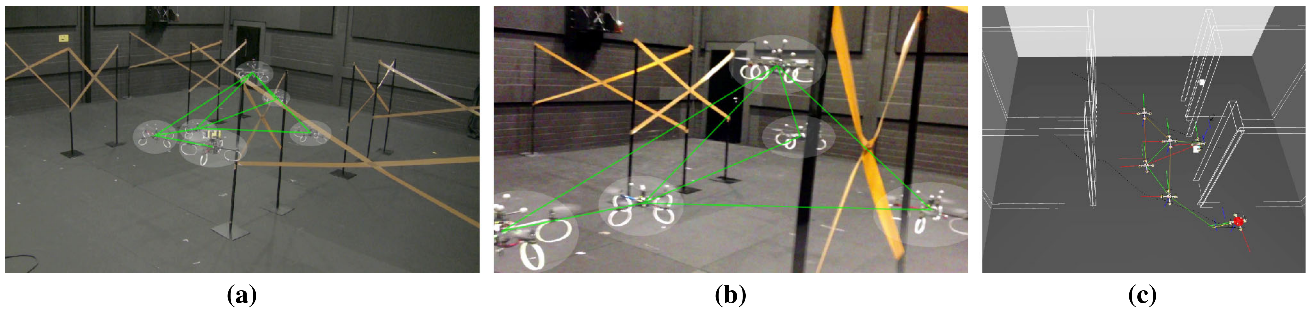

Fig. 13 Three simultaneous screenshots of the experiment described in the text: **a** shows the side view of the scene from a fixed camera. Connections between UAVs (*brightened areas*) are overlayed as *green lines*. **b** shows the view taken from the onboard camera of the ‘explorer’ 1 using the same highlighting. **c** shows a 3D synthetic reconstruction of the robot positions and connections are shown with a line given in *green* when the weight is high, *red* shortly before a connection breaks and as a gradient in between. The robot that is marked with the *red sphere* is currently decoupled and controlled by the human operator

connectivity eigenvalue $\lambda_2(t)$ (see Appendix) remains positive for any $t > 0$, thus implying continuous connectivity of the graph $\mathcal{G}$ as desired. The time-varying number of edges in Fig. 14b shows the dynamic reconfiguration of the group topology which ranges between topologies with 5 edges (the minimum for having $\mathcal{G}$ connected) and topologies with up

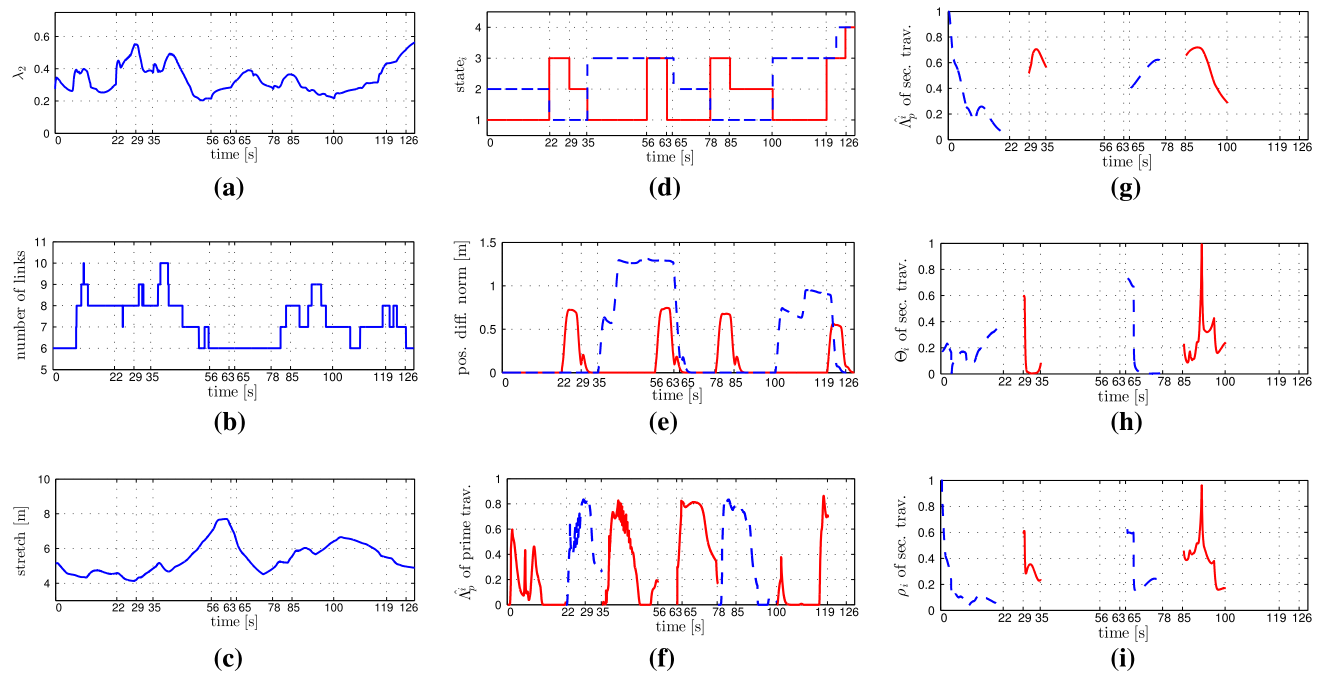

Fig. 14 Behavior of different measurements during an experiment: **a** λ_2 always keeps greater than zero, thus showing how the group remains connected at all times, **b** the number of links $|\mathcal{E}(t)|$ of the interaction graph $\mathcal{G}(t)$, **c** the stretch of the formation given by the maximum Euclidean distance between any two quadrotors over time, **d** the exploration states with the following meaning: 1: 'prime traveler', 2: 'secondary traveler', 3: 'anchor', 4: 'connector', **e** the position difference between the virtual point of the connectivity maintenance and the commanded position to the quadrotors showing the decoupling as an 'anchor', **f** the traveling efficiency of the current 'prime traveler' (see (12)), **g** the estimation of the traveling efficiency by the 'secondary travelers' (see (13)), **h** the force direction alignment for the 'secondary travelers' (see (10)), **i** the adaptive gain used by the 'secondary travelers' to scale down their traveling force (see (15))

to 10 edges. This plot clearly shows how the adopted connectivity maintenance approach can cope with time-varying graphs. In Fig. 14c, we report the stretch of the group, defined as the maximum Euclidean distance between any two robots at a given time t. One can then appreciate how this stretch varies among 3.5 and 7.5 m thus exploiting at most the allowable ranges of the experimental arena. Notice also how the stretch is in general larger when the number of links (and consequently $\lambda_2(t)$) is smaller. In fact the two peaks at about 60 s and 103 s occur when the robots are forced into a sparsely connected topology because the two 'explorers' have concurrently reached their farthest target pairs, i.e., (A_1,B_2) and (B_1,D_2).

Figure 14d shows the 'explorer' states state_2 and state_4 over time, with a dashed blue line and solid red line, respectively. In the plot, the following code is used: $1 =$ 'prime traveler', $2 =$ 'secondary traveler', $3 =$ 'anchor' and $4 =$ 'connector'. For $i = 1, 3, 5, 6$ it is $\mathsf{state}_i = 4$ for all $t \in [0, 129]$. Notice that, because of the algorithm design, at most one 'explorer' has $\mathsf{state}_i = 1$ at any given time.

The temporary decoupling of the 'explorers' from the points q_2 and q_4 during their anchoring behavior can be appreciated in Fig. 14e, where the Euclidian distance between the real robot position in the trajectory and the corresponding $q_i(t)$ is shown, for $i = 2, 4$. 'Explorer' 1 (solid red line) decouples four times in total, in correspondence of the 2 pick-and-place operations, which gives rise to 4 short peaks in the plot. 'Explorer' 2 (dashed blue line) decouples two times in total, in correspondence of the 2 human-in-the-loop operations, causing 2 long peaks in the plot.

Figure 14f shows the traveling efficiency Λ_p of the current 'prime traveler' with a dashed blue line when 'explorer' 1 is the 'prime traveler' and with a solid red line when 'explorer' 2 is the 'prime traveler'. The estimation $\hat{\Lambda}_p^i$ of this value (see (13)) by all robots that are currently not 'prime traveler' is given in Fig. 15. We chose $k_\Lambda = 1$ resulting in a relatively slow propagation to show the additional robustness of our algorithm against this parameter (and the simple adopted consensus propagation), but clearly one could easily employ higher gains. To make it easier for the reader to understand the following discussion, we show again in Fig. 14g the essential information of this last plot whenever a robot is a 'secondary traveler'. In Fig. 14h and i the force direction alignment Θ_i (see (10)) and the adaptive gain ρ_i (see (15)) of the current 'secondary traveler' are shown. In the latter three plots a dashed blue line indicates when 'explorer' 1 is the 'secondary traveler', and a solid red line when 'explorer' 2 is the 'secondary traveler'.

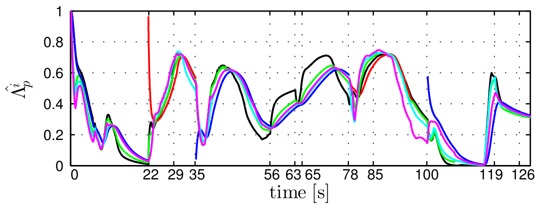

Fig. 15 Estimation of the 'prime traveler' traveling efficiency of all the six robots, whenever they are currently not a 'prime traveler' (see (13)). The color scheme for the robots is as in Fig. 12

To fully understand the important features of our method, we now give a detailed description of the time interval [0, 22] in Fig. 14f–i. A similar pattern can then be found in the rest of the experiment. In this time interval, the 'explorer' 2 is the 'prime traveler', while 'explorer' 1 is a 'secondary traveler' (and the rest are 'connectors'). Due to the initial transient of its motion controller, the 'prime traveler' starts with $\Lambda_p = 0$ and quickly reaches $\Lambda_p = 0.6$. Shortly after, the traveling efficiency decreases again since 'explorer' 2 reaches the end of the area where it can freely move and, thus, needs to 'pull' the other robots for preserving connectivity of $\mathcal{G}$. This effect is propagated to the 'explorer' 1 as shown in Fig. 14g. The force direction alignment between the traveling force and the generalized connectivity force is shown in Fig. 14h. Combining these two plots with (15) allows to understand the effect of Fig. 14i. As can be seen, the 'secondary traveler' slows down its motion to around 10 % for roughly 5 s. This enables the 'prime traveler' to travel faster again (see Fig. 14f). However, since the 'explorer' 2 needs to move around the wall (see Fig. 12a) to reach its target, it needs to 'pull' the other robots even more for preserving connectivity. Therefore, the traveling efficiency becomes zero and, although the direction alignment of the 'secondary traveler' becomes higher, the overall gain ρ_i stays very low: this makes it possible for the 'prime traveler' to eventually reach its target. We recall here that, according to Table 3 and (9), $\Lambda_p = 1$ as soon as the 'prime traveler' achieves a speed of at least 80 % of its desired cruise speed (so the error is less than 20 %), while $\Lambda_p = 0$ means a speed of less than 30 % (an error of more than 70 %), and *not necessarily* a zero velocity.

5 Conclusions

In this paper we presented a novel distributed and decentralized control strategy that enables simultaneous multi-target exploration while ensuring a time-varying connected topology in a 3D cluttered environment. We provided a detailed description of our algorithm which effectively exploits presence of *four* dynamic roles for the robots in the group. In particular, a 'connector' is a robot with no active target, an 'anchor' a robot close to its desired location, and all other robots are instead moving towards their targets. Presence of at most one 'prime traveler', holding a leader virtue, is always guaranteed. All other robots ('secondary travelers') are bound to adapt their motion plan so as to facilitate the 'prime traveler' visiting task. This feature ensures that the 'prime traveler' is always able to reach its target, and thus ultimately allows to conclude completeness of the exploration strategy. The scalability and effectiveness of the proposed method was shown by presenting a complete and extensive set of simulative results, as well as an experimental validation with real robots for further demonstrating the practical feasibility of our approach.

As future development, we plan to modify the control of the 'connectors' in order to actively improve the connectivity (e.g., moving towards the center of the group or towards the closest 'explorer') and therefore decrease the overall completion time even more. Another extension could include imposing temporal targets that expire before any robot can possibly reach them. In our framework this could be easily achieved by letting the corresponding 'prime traveler' or 'secondary traveler' switching into a 'connector' whenever a target expires, for then automatically starting to explore the next target (if any).

An important direction worth of investigation would also be the possibility to (explicitly) deal with errors or uncertainties in the relative position measurements (w.r.t. robots and obstacles) needed by the algorithm. Indeed, the presented results rely on an accurate measurement of robot and obstacle relative positions obtained by means of an external motion capture system.

Another improvement could address the distributed election of the 'prime traveler' as was already discussed in Sect. 3.3. Indeed, while the adopted flooding approach does not require presence of a centralized planning unit, it still needs to take into account information from all robots. It would obviously be preferable to only exploit information available to the robot itself and its 1-hop neighbors. This could be achieved by leveraging some (suitable variant of the) consensus algorithm as done for the decentralized propagation of the traveling efficiency of the current 'prime traveler'. More generally, it might also be beneficial to improve the election of the 'prime traveler' by considering other criteria than the Euclidean distance w.r.t. a target which may not always result in an 'optimal' group motion (e.g., when obstacles, such as a wall, are present between the next 'prime traveler' and the target). The election could for instance choose the robot with the highest chance of reducing even further the completion time, e.g., based on the current motion of the group or direction of the majority of current targets of all 'secondary travelers'.

Finally, it would be interesting to obtain an analytical upper bound of the total exploration time for our approach, although, in our opinion, deriving such a bound is unfortunately not so straightforward. Clearly, the considered multi-target exploration scenario has some analogies with the multiple traveling salesman problem (Bektas 2006), where a certain number N of agents are asked to find a set of N shortest routes through a set of m cities and return back to the start. Nevertheless, an analysis based on the multiple traveling salesman problem would not easily extend to our case because of the constraint of continuous connectivity maintenance.

6 Appendix

For the sake of completeness and readablity, we will recap here the main features of the connectivity maintenance algorithm presented in Robuffo Giordano et al. (2013) with some small changes in the variable names. We start by defining $d_{ij} = \|q_i - q_j\|$ as the distance between two robot positions q_i and q_j, and $d_{ijo} = \min_{\varsigma\in[0,1],o\in\mathcal{O}} \|q_i + \varsigma(q_j - q_i) - o\|$ as the closest distance from the line of sight between robot i and j to any obstacle.

The main conceptual steps behind the computation of f_i^λ can be summarized as follows:

1. Define an auxiliary weighted graph $\mathcal{G}^\lambda(t) = (\mathcal{V}, \mathcal{E}^\lambda, W)$, where W is a symmetric nonnegative $n \times n$ matrix whose entries W_{ij} represent the weight of the edge (i, j) and $(i, j) \in \mathcal{E}^\lambda \Leftrightarrow W_{ij} > 0$.
2. Design every weight W_{ij} as a *smooth* function of the robot positions q_i, q_j and of the obstacle points surrounding q_i and q_j, with the property that $W_{ij} = 0$ if and only if at least one of the following conditions is verified:
   (a) the maximum sensing range R_s is reached: $d_{ij} \geq R_s$,
   (b) the minimum desired distance to obstacles R_o is reached (where $R_o < R_m$): $d_{ijo} \leq R_o$;
   (c) the minimum desired inter-robot distance R_c is reached: $d_{ik} \leq R_c$ for at least one $k \neq i$.
3. Compute f_i^λ as the negative gradient of a potential function $V^\lambda(\lambda_2)$ that grows unbounded when $\lambda_2 \to \lambda_2^{\min}$ from above, where λ_2 is the second smallest eigenvalue of the (symmetric and positive semi-definite) Laplacian matrix $L = \mathrm{diag}_{i=1}^n(\sum_{j=1}^n W_{ij}) - W$, and $\lambda_2^{\min}$ is a non-negative parameter. This eigenvalue λ_2 is often also called Fiedler eigenvalue.

It is known from graph theory that a graph is connected if and only if the Fiedler eigenvalue of its Laplacian is positive (Fiedler 1973). If $\mathcal{G}^\lambda(0)$ is connected, and in particular $\lambda_2(0) > \lambda_2^{\min}$, then under the action of f_i^λ the value of $\lambda_2(t)$ can never decrease below $\lambda_2^{\min}$ and therefore $\mathcal{G}^\lambda(t)$ always stays connected.

From a formal point of view the anti-gradient of V^λ for the i-th robot takes the form

$$f_i^\lambda = -\frac{\partial V^\lambda(\lambda_2)}{\partial q_i} = -\frac{dV^\lambda}{d\lambda_2}\frac{\partial \lambda_2}{\partial q_i}. \tag{17}$$

Moreover, if the formal expression of V^λ and W are known then (17) can be analytically computed via the expression (Yang et al. 2010),

$$\frac{\partial \lambda_2}{\partial q_i} = \sum_{j\in\mathcal{N}_i} \frac{\partial W_{ij}}{\partial q_i}(\nu_{2_i} - \nu_{2_j})^2, \tag{18}$$

where ν_{2_i} is the i-th component of the normalized eigenvector of L associated to λ_2.

In order to have a fully decentralized computation of f_i^λ, the robots perform a distributed estimation of both $\lambda_2(t)$ and $\nu_{2_i}(t)$, for all $i = 1, \ldots, N$, as shown in Yang et al. (2010). In Robuffo Giordano et al. (2013) the authors finally prove the passivity (and then the stability) of the system w.r.t. the pair (f_i, v_i) for all $i = 1, \ldots, N$, as well as the possibility to compute the connectivity force f_i^λ in (17) in a completely decentralized way.

References

Antonelli, G., Arrichiello, F., Chiaverini, S., & Setola, R. (Aug 2005). A self-configuring MANET for coverage area adaptation through kinematic control of a platoon of mobile robots. In *2005 IEEE/RSJ international conference on intelligent robots and systems* (pp. 1332–1337), Edmonton, Canada.

Antonelli, G., Arrichiello, F., Chiaverini, S., & Setola, R. (Mar 2006). Coordinated control of mobile antennas for ad-hoc networks in cluttered environments. In *9th International conference on intelligent autonomous systems*, Tokyo, Japan.

Bektas, T. (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, *34*(3), 209–219.

Biagiotti, L., & Melchiorri, C. (2008) *Trajectory planning for automatic machines and robots*. Springer. ISBN 978-3540856283.

Burgard, W., Moors, M., Stachniss, C., & Schneider, F. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics and Automation*, *21*(3), 376–386.

Durham, J. W., Franchi, A., & Bullo, F. (2012). Distributed pursuit-evasion without global localization via local frontiers. *Autonomous Robots*, *32*(1), 81–95.

Faigl, J., & Hollinger, G. A. (Sep 2014). Unifying multi-goal path planning for autonomous data collection. In *2013 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2937–2942), Chicago, IL.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, *23*(98), 298–305.

Franchi, A., Freda, L., Oriolo, G., & Vendittelli, M. (2009). The sensor-based random graph method for cooperative robot exploration. *IEEE/ASME Transactions on Mechatronics*, *14*(2), 163–175.

Freeman, R. A., Yang, P., & Lynch, K. M. (Jan 2006). Stability and convergenceproperties of dynamic average consensus estimators. In *45thIEEE conference on decision and control* (pp. 338–343), San Diego, CA.

Grabe, V., Riedel, M., Bülthoff, H. H., Robuffo Giordano, P., & Franchi, A. (Sep 2013). The TeleKyb framework for a modular and extendible ROS-based quadrotor control. In *6th European conference on mobile robots* (pp. 19–25), Barcelona, Spain.

Hollinger, G., & Singh, S. (2012). Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics*, *28*(4), 967–973.

Howard, A., Parker, L. E., & Sukhatme, G. S. (2006). Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *The International Journal of Robotics Research*, *25*(5–6), 431–447.

Jensfelt, P., & Kristensen, S. (2001). Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*, *17*(5), 748–760.

Lächele, J., Franchi, A., Bülthoff, H. H., & Robuffo Giordano, P. (2012). SwarmSimX: Real-time simulation environment for multi-robot systems. In I. Noda, N. Ando, D. Brugali, & J. J. Kuffner (Eds.), *3rd international conference on simulation, modeling, and programming for autonomous robots*. *Lecture Notes in Computer Science* (Vol. 7628, pp. 375–387). Springer.

Lim, H., & Kim, C. (2001). Flooding in wireless ad hoc networks. *Computer Communications*, *24*(3–4), 353–363.

Lynch, N. A. (1997). *Distributed algorithms*. Morgan Kaufmann, ISBN 1558603484.

Masone, C., Franchi, A., Bülthoff, H. H., & Robuffo Giordano, P. (Oct 2012) Interactive planning of persistent trajectories for human-assisted navigation of mobile robots. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2641–2648), Vilamoura, Portugal.

Mistler, V., Benallegue, A., & M'Sirdi, N. K. (Sep 2001). Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In *10th IEEE internationl symposium on robots and human interactive communications* (pp. 586–593), Bordeaux, Paris, France.

Mosteo, A. R., Montano, L., & Lagoudakis, M. G. (Nov 2008). Guaranteed-performance multi-robot routing under limited communication range. In *9th international symposium on distributed autonomous robotic systems* (pp. 491–502), Tsukuba, Japan.

Murphy, R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., & Erkmen, A. (2008). Search and rescue robotics. In B. Siciliano & O. Khatib (Eds.), *Springer handbook of robotics* (pp. 1151–1173). Springer.

Nestmeyer, T., Robuffo Giordano, P., & Franchi, A. (May 2013a). Multi-target simultaneous exploration with continual connectivity.In *2nd international work. on crossing the reality gap—From single tomulti- to many robot systems, at 2013 IEEE international conference on roboticsand automation*, Karlsruhe, Germany.

Nestmeyer, T., Robuffo Giordano, P., & Franchi, A. (Oct 2013b). Human-assisted parallel multi-target visiting in a connected topology. In *6th International work. on human-friendly robotics*, Rome, Italy.

Olfati-Saber, R., & Murray, R. M. (Jun 2003). Consensus protocols for networks of dynamic agents. In *2003 American control conference* (pp. 951–956), Denver, CO.

Olfati-Saber, R., & Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, *49*(9), 1520–1533.

Pasqualetti, F., Franchi, A., & Bullo, F. (2012). On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *IEEE Transactions on Robotics*, *28*(3), 592–606.

Pei, Y., & Mutka, M. W. (May 2012). Steiner traveler: Relay deployment for remote sensing in heterogeneous multi-robot exploration. In *2012 IEEE international conference on robotics and automation* (pp. 1551–1556), St. Paul, MN.

Pei, Y., Mutka, M. W., & Xi, N. (May 2010). Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness. In *2010 IEEE international conference on robotics and automation* (pp. 5460–5465), Anchorage, AK.

Robuffo Giordano, P., Franchi, A., Secchi, C., & Bülthoff, H. H. (2013). A passivity-based decentralized strategy for generalized connectivity maintenance. *The International Journal of Robotics Research*, *32*(3), 299–323.

Stump, E., Jadbabaie, A., & Kumar, V. (May 2008) Connectivity management in mobile robot teams. In *2008 IEEE international conference on robotics and automation* (pp. 1525–1530), Pasadena, CA.

Stump, E., Michael, N., Kumar, V., & Isler, V. (2011) Visibility-based deployment of robot formations for communication maintenance. In *2011 IEEE international conference on robotics and automation* (pp. 4489–4505), Shanghai, China.

Tardioli, D., Mosteo, A. R., Riazuelo, L., Villarroel, J. L., & Montano, L. (2010). Enforcing network connectivity in robot team missions. *The International Journal of Robotics Research*, *29*(4), 460–480.

Yang, P., Freeman, R. A., Gordon, G. J., Lynch, K. M., Srinivasa, S. S., & Sukthankar, R. (2010). Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica*, *46*(2), 390–396.

Zavlanos, M. M., & Pappas, G. J. (2007). Potential fields for maintaining connectivity of mobile networks. *IEEE Transaction on Robotics*, *23*(4), 812–816.

Thomas Nestmeyer received his diploma in computer science from the Eberhard Karls University in Tübingen, Germany in 2013 after being a visiting student at the University of Otago in Dunedin, New Zealand in 2010. From 2012–2014 he worked on autonomous multi-robot exploration and tele-operation with haptic feedback at the Max Planck Institute for Biological Cybernetics in Tübingen Germany. Since 2014 he is a Ph.D. student at the Max Planck Institute for Intelligent Systems in Tübingen, Germany, where he focuses on computer vision and especially the inference of physical properties of objects in an image.

Paolo Robuffo Giordano received his M.Sc. degree in Computer Science Engineering in 2001, and his Ph.D. degree in Systems Engineering in 2008, both from the University of Rome "La Sapienza". In 2007 and 2008 he spent one year as a PostDoc at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR), and from 2008 to 2012 he was Senior Research Scientist at the Max Planck Institute for Biological Cybernetics in Tübingen, Germany. Since 2012 he is a CNRS researcher in the Lagadic group at Irisa and Inria in Rennes, France. Dr. Robuffo Giordano is currently Associate Editor of the IEEE Transactions on Robotics. His research interests span nonlinear control, robotics, planning, haptics and VR applications.

Heinrich H. Bülthoff received the Ph.D. in natural sciences from the Eberhard Karls Universität in Tübingen, Germany in 1980. He is director of the Human Perception, Cognition and Action Department at the Max Planck Insti- tute for Biological Cybernetics in Tübingen from 1993. He is also Honorary Professor at the Eberhard Karls Universität in Tübingen. He was a research scientist at the MPI for Biological Cybernetics and Massachusetts Institute of Technology (1980-1988) and Professor of Cognitive Science at Brown University from 1988-1993. His research interests include object recognition and categorization, perception and action in virtual environments, human-robot interaction and perception.

Antonio Franchi received the Ph.D. degree in System Engineering from Sapienza University of Rome, Italy, in 2010. In 2009 he was a Visiting Scholar at the University of California at Santa Barbara. From 2010–2012 he was a Research Scientist in the Human-Robot Interaction group at the Max Planck Institute for Biological Cybernetics (MPI-KYB) in Tübingen, Germany. From 2012–2013 he was a Senior Research Scientist and the Project Leader of the Autonomous Robotics and Human-Machine Systems group at the MPI-KYB. Since 2014 he has been a Permanent CNRS Researcher in the RIS group at LAAS-CNRS, Toulouse, France. His main research interests include autonomous systems and robotics, with a special regard to control, planning, estimation, human-machine systems, haptics, and hardware/software architectures. His main areas of expertise are multiple-robot systems and aerial robotics. He published more than 70 papers in peer-reviewed international journals and conferences. In 2010 he was awarded the IEEE RAS ICYA Best Paper Award. He is an Associate Editor of the IEEE Robotics & Automation Magazine. He is the co-founder and co-chair of the IEEE RAS Technical Committee on Multiple Robot Systems.