J Intell Robot Syst (2018) 89:69–85
DOI 10.1007/s10846-017-0516-7

Multi-agent Rapidly-exploring Pseudo-random Tree

Armando Alves Neto · Douglas G. Macharet · Mario F. M. Campos

Received: 10 June 2016 / Accepted: 10 February 2017 / Published online: 7 March 2017

Abstract Real-time motion planning and control for groups of heterogeneous and under-actuated robots subject to disturbances and uncertainties in cluttered constrained environments is the key problem addressed in this paper. Here we present the Multi-agent Rapidly-exploring Pseudo-random Tree (MRPT), a novel technique based on a classical Probabilistic Road Map (PRM) algorithm for application in robot team cooperation. Our main contribution lies in the proposal of an extension of a probabilistic approach to be used as a deterministic planner in distributed complex multi-agent systems, keeping the main advantages of PRM strategies like simplicity, fast convergence, and probabilistic completeness. Our methodology is fully distributed, addressing missions with multi-robot teams represented by high nonlinear models and a great number of Degrees of Freedom (DoFs), endowing each agent with the ability of coordinating its own movement with other agents while avoiding collisions with obstacles. The inference of the entire team's behavior at each time instant by each individual agent is the main improvement of our method. This scheme, which is behavioral in nature, also makes the system less susceptible to failures due to intensive traffic communication among robots. We evaluate the time complexity of our method and show its applicability in planning and executing search and rescue missions for a group of robots in $\mathbb{SE}3$ outdoor scenarios and present both simulated and real-world results.

Keywords Multi-agent systems · Heterogeneous teams · Underactuated robots · Path planning and control · Rapidly-exploring random trees

1 Introduction

A fundamental problem to be faced by the robotics research community in the near future is the cooperation of heterogeneous and underactuated agents in constrained cluttered environments. Technological advancements in the present have shown that, soon, large numbers of robots will become increasingly ubiquitous in our daily lives and, consequently, they will demand distributed control techniques with low computational cost. It is well-known that groups of robots with different capabilities and physical constraints working collaboratively may be applied to solve complex tasks more efficiently than teams with homogeneous characteristics. Although there are techniques in the state-of-the-art literature dealing with

A. Alves Neto (✉)
Department of Electronic Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil
e-mail: aaneto@cpdee.ufmg.br

D. G. Macharet · M. F. M. Campos
Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil

specific parts of this problem, the results are still below expectations.

Therefore, in this paper, we address the problem of coordinate a team of heterogeneous robots navigating in three-dimensional environments with static obstacles, and subject to disturbances and measurement noise. More specifically, we propose a set of algorithms for fully decentralized control and real-time trajectory planning to guide the robot team from a initial to a final configuration, defined by its mission, subject to cooperative and collision constraints.

Our method extends a random planning algorithm to consider multi-robot cooperation scenarios, finding efficient solutions with linear computational cost. This challenge is faced by individually enabling agents to predict the behavior of the whole team, despite the random nature of the method, thereby allowing the coordination of the group. Here we introduce the Multi-agent Rapidly-exploring Pseudo-random Tree (MRPT), a (pseudo-random) deterministic algorithm which extends benefits of the well-known Closed Loop Rapidly-exploring Random Tree (CL-RRT) technique to multi-robot cooperative assembles of heterogeneous agents in cluttered environments, like the one composed by aerial and ground robots illustrated at Fig. 1.

We choose the CL-RRT as the basis for our methodology due to its robustness to uncertainties and its capability of real-time re-planning. Compared to current literature, our approach is capable of executing plans in a completely decentralized manner for robots with highly nonlinear models and a great number of Degrees of Freedom (DoFs). The prediction of the entire team's behavior by each of its members, in a deterministic way, substantially reduces the need for communication between agents, making the system more robust to disturbances.

Fig. 1 Real experiment of cooperative navigation using the MRPT with a team of heterogeneous (aerial and ground) under-actuated robots

The remainder of this paper is structured as follows: Section 2 presents the related work; Section 4 formalizes the problem and introduce the set of algorithms composing our methodology, and an analysis of the computational complexity of all steps, demonstrating their scalability; Section 5 shows simulated and real-world experiments, in the context of search and rescue navigation for teams of heterogeneous robots; and finally, Section 6 draw the conclusions and discuss drawbacks from using our approach, as well as avenues for future investigation.

2 Related Work

In the state-of-the-art literature, control and motion planning for teams of homogeneous robots has been the focus of several research works, and a great number of solutions has been proposed. However most of these techniques, from Artificial Potential Fields [1] to Cell Decomposition Roadmaps [2, 3], present disadvantages when dealing with large dimensional problems, e.g. multi-robot coordination. Generally speaking, deterministic methods usually present high computational cost, a major limitation regarding their applicability. Model Predictive Controller (MPC) [4], for example, generates good results for path planning problems, but its strategy is based on optimization algorithms, which can easily render impractical time complexity solutions for large problems.

On the other hand, Probabilistic Road Maps (PRMs) and Sampling-based motion algorithms have been vastly employed to single-robot scenarios, e.g. the Open Loop Rapidly-exploring Random Tree (OL-RRT), or simply RRT [3]. Introduced by Lavalle in [5], RRTs are growing graph strategies that spread trajectory trees through known cluttered environments, based on open-loop models of a system. Randomly chosen nodes are used to incrementally generate paths satisfying motion and collision constraints of the robot to led it to a given goal position. Their main advantages are simplicity, fast convergence, probabilistically completeness, and also the capability of

incorporate complex systems modeled by nonlinear equations, reasons why RRTs have been widely used in recent years. Scenarios with great number of Degrees of Freedom (DoFs) and geometric constraints can easily be treated by such approach, and that is (as we will discuss later) a very important feature in the present work.

Concerning the *open-loop* characteristic of RRTs, a disadvantage in some aspects, authors of [6] developed the Closed Loop Rapidly-exploring Random Tree (CL-RRT). Rather than acting directly in the vehicle's inputs, this method generates desired commands to feed internal controllers of the robot, then planning in a *closed-loop* form. When such controllers are well tunned, trajectories provided by the CL-RRT become feasible during the execution step. This step is given by an execution loop, where parts of the current growing tree are sent to the robot's controllers in real-time.

The authors of [7] compared the propagation error in the robot's states between the Open Loop Rapidly-exploring Random Tree (OL-RRT) and the CL-RRT under external disturbances, and conclude that CL-RRTs are more robust to uncertainties, keeping the error between the model (for the linear case) and real robot always finite and bounded.

In view of their advantages, RRTs have also been applied to multi-robot cooperation. Most of the works, however, present centralized approaches with restrictive scalability strategies. In [8], for example, the random technique is applied in a cooperative object transportation task, where humanoid robots use local information from their cameras to plan and re-plan actions by a centralized approach. In [9], Vahrenkamp et al. present a method based on RRTs to coordinate mobile humanoid robots (with twenty Degrees of Freedom (DoFs) each) in grasping tasks. Collision avoidance between agents is also ensured by a centralized verification strategy.

The authors of [10] present one of the first attempts to develop a decentralized random planning scheme, which generates paths for single or groups of robots in environments with obstacles. For the multi-robot case, the overall strategy was distributed among the agents, but the planning is synchronized, so that a robot plans its path only after receiving the paths planned by the others.

Another strategy for decentralized planning of multiple robots based on the CL-RRT is presented in [11]. Called Cooperative Decentralized Multi-Agent Rapidly-exploring Random Tree (DMA-RRT), this method also provides synchronization among agents, and uses an heuristic based on choice via auction to define the re-planning order in the team. At each bidding round, only the auction winner is allowed to re-plan in order to avoid collisions with other agents, and this new path must be broadcast to the team, so each robot will be capable of update their collision constraints. Authors of [12] also present a decentralized RRT-based approach, dealing inclusive with imperfect communication and collision checking. However, their method is limited to homogeneous groups of robots.

In [13], we have proposed a version of the classical RRT to, in a centralized form, plan trajectories for a team of fixed-wing Unmanned Aerial Vehicles (UAVs) flying at the same constant altitude in environments filled with obstacles. This technique was further extended in [14], considering a decentralized approach where robots were able to plan different movements for the entire team and propagate them, in the manner that all agents could individually decide the best among all of these plans.

Although some aforementioned techniques are completely decentralized, they are also heavily dependent on a fully connected communication network and require intensive information traffic due to re-planning and coordination (robots must always broadcast their plans to the team). In addition, changes in individual plans may end up invalidating parts of the tree as a whole, which means that processing time is wasted. The proposed technique does not heavily depend on network structure, allowing for simultaneous and distributed re-planning, without discarding parts already planned.

In this paper, we extend the CL-RRT to multi-robot systems based upon heuristics to improves the temporal coordination between each vehicle of the team, reducing errors in the plan execution and also reducing the chance of collisions among agents. Classical CL-RRT deals only with one single robot and needs to be adapted to deals with multi-robot teams. Our technique, called MRPT, mainly differs from CL-RRT at the capability of allowing coordination among multiple agents with the advantages of probabilistic planners. Our method also differs from Cooperative Decentralized Multi-Agent Rapidly-exploring Random Tree (DMA-RRT) at the capability of dealing with heterogeneous robots and the little need for

re-planning of other agents when one single agent changes its plan in real-time.

3 Notation

A and **a** indicate matrices and vectors, respectively; $\mathscr{A}$ is a set of elements; reference frames are denoted by $\{A\}$; $\vec{0}_{m\times n}$ is a null matriz with dimensions $m \times n$; diag $(\vec{a})$ is a diagonal matrix with $\vec{a}$ been the diagonal elements; $\left[\,\cdot\,\right]'$ is the vector/matrix transpose operator; $\mathscr{N}\left(\mu, \sigma^2\right)$ is a standard Gaussian distribution with mean μ and deviation σ; $\mathscr{O}(\cdot)$ is the time complexity of an algorithm, or part of it; $\mathcal{T} \leftarrow \vec{a}$ indicates a vertex $\vec{a}$ was incorporated to set $\mathcal{T}$.

4 Methodology

4.1 Problem Statement

Let us assume a previously known constrained workspace $\mathcal{W}$ and an obstacle-free area $\mathcal{W}_{\text{free}} \subset \mathcal{W}$. Inside it navigates a generic dynamic system

$$\vec{x}_{t+1} = \mathrm{f}\left(\vec{x}_t, \vec{u}_t, \vec{w}_t\right), \tag{1}$$

where $\vec{x} \in \mathbb{R}^{n_x}$ is the state vector (with initial conditions $\vec{x}_0$ known *a priori*), $\vec{u} \in \mathbb{R}^{n_u}$ is the input vector, and $\vec{w} \in \mathbb{R}^{n_w}$ an unknown disturbance and/or uncertainty matrix corrupting the system. Generally speaking, $\mathrm{f}(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \rightarrow \mathbb{R}^{n_x}$ is a nonlinear function describing the time variation of $\vec{x}$. Let us also consider the control law

$$\vec{u}_t = \mathrm{k}\left(\vec{r}_t, \vec{x}_t\right), \tag{2}$$

where $\vec{r}$ is the trajectory command and $\mathrm{k}(\cdot) : \mathbb{R}^{n_r} \times \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_u}$ is the function for process stabilization. Most of the vehicles are subject to mechanical constraints, hence, in this paper, we are mainly interested in control robots in *under-actuated conditions*, such that $n_x > n_u$ and $n_x \gg n_r$.

Now consider a team $\mathcal{R}$ composed by n heterogeneous robots sharing $\mathcal{W}_{\text{free}}$. This team is ruled by the dynamic model

$${}^n\vec{x}_{t+1} = \mathrm{F}\left({}^n\vec{x}_t, {}^n\vec{u}_t, {}^n\vec{W}_t\right), \tag{3}$$

where

$${}^n\vec{x}_t = \left[\,\vec{x}_t^1\ \vec{x}_t^2 \ldots \vec{x}_t^n\,\right]', \quad {}^n\vec{u}_t = \left[\,\vec{u}_t^1\ \vec{u}_t^2 \ldots \vec{u}_t^n\,\right]' \text{ and}$$
$${}^n\vec{W}_t = \mathrm{diag}\left(\left[\,\vec{w}_t^1\ \vec{w}_t^2\ \ldots\ \vec{w}_t^n\,\right]\right)$$

are the state vector, input vector and noise matrix of the group, respectively, all modeled as simple compositions of the corresponding vectors of each $R_i \in \mathcal{R}$. Here ${}^n\vec{W}$ is a diagonal matrix, since no correlated noise among robots is assumed. Equivalently, $\mathrm{F}(\cdot) : \mathbb{R}^{\sum {n_x}^i} \times \mathbb{R}^{\sum {n_u}^i} \times \mathbb{R}^{\sum\sum {n_w}^{ij}} \rightarrow \mathbb{R}^{\sum {n_x}^i}$ is a nonlinear function describing the dynamics of the robot ensemble. However, due to inter-agent motion constraints, like collision, this function cannot be modeled as a simple composition of $\mathrm{f}(\cdot)$ from each R_i.

Apart from motion constraints of each agent, it is also necessary take into account navigation constraints imposed by other members of the group. We represent these collision constraints by inequalities, such that

$$\mathrm{G}\left(\vec{x}_t^i, \vec{x}_t^j\right) \leq 0, \quad \forall i, j \in 1...n, i \neq j, \tag{4}$$

where $\mathrm{G}(\cdot) : \mathbb{R}^{{n_x}^i} \times \mathbb{R}^{{n_x}^j} \rightarrow \mathbb{R}$ is a function describing collision conditions between robots R_i and R_j, which is highly dependent on geometrical characteristics and sensing uncertainties of each vehicle. Hence, in a team of heterogeneous robots, there may exist many different collision functions.

Concerning these and other characteristics (e.g., network communication structure) it is possible to establish an analogy with Eq. (2), leading to

$${}^n\vec{u}_t = \mathrm{K}\left({}^n\vec{r}_t, {}^n\vec{x}_t\right), \tag{5}$$

where $\mathrm{K}(\cdot) : \mathbb{R}^{\sum {n_r}^i} \times \mathbb{R}^{\sum {n_x}^i} \rightarrow \mathbb{R}^{\sum {n_u}^i}$ is a nonlinear and decentralized multi-robot control law, function of the states of all robots and the trajectory command

$${}^n\vec{r}_t = \left[\,\vec{r}_t^1\ \ \vec{r}_t^2\ \ \ldots\ \ \vec{r}_t^n\,\right]'.$$

Finally, the specific problem addressed in this paper can be defined as follows:

Problem 1 Let $\mathcal{R}$ be a team composed of n heterogeneous robots, described by ${}^n\vec{x}_t$ in $\mathcal{W}$, compute ${}^n\vec{r}_t$ leading this team from its initial state ${}^n\vec{x}_{\text{init}} = {}^n\vec{x}_0$ at t_0 to some final state ${}^n\vec{x}_{\text{goal}}$ within a finite time $t_f > t_0$, such that ${}^n\vec{x}_t \in \mathcal{W}_{\text{free}}\ \forall t \in \left(t_0, t_f\right]$. This trajectory, ${}^n\vec{r}_t$, must be calculated in a coordinated, decentralized, collision-free and communication-constrained form.

4.2 Pseudo-Random Strategy

Although nowadays there is a variety of heuristics to guide the random choice at sampling-based algorithms, they can be grouped into two basic categories: *pseudo-quasi-random sampling* and *random sampling* approaches. Both represent deterministic functions to simulate randomness, but while pseudo-random methods generate *real* random numbers, quasi-random (also known as low-discrepancy) techniques produce totally deterministic sequence of numbers, drawn from some probability distribution.

Due to this low-discrepancy characteristic, quasi-random sequences outperform pseudo-random algorithms in motion planning problems, though, for very high dimensional systems (like our case), this difference of performance is practically negligible [15]. In fact, when multiple robots apply random planning to navigate in common environments, it becomes very difficult to use any kind of *a priori* knowledge about the group to predict its behavior and avoid inter-agent collisions. Thus, it is necessary to make extensive use of communication between robots, which must broadcast their plans to other agents whenever a change occurs [10, 11]. This characteristic reduces re-planning ability of the agents, and is likely to cause failures due to temporary loss of communication, since the methods are heavily dependent on a fully connected network.

Let us assume a scenario where a deterministic (quasi-random or pseudo-random) function is used to generate samples in the Open Loop Rapidly-exploring Random Tree (OL-RRT) or any other PRM algorithm [3, 10]. For simplicity reasons, we will refer to this henceforth as the pseudo-random sampling function

$$^{n}\vec{r}_k = \mathrm{S}(k), \tag{6}$$

where k is a natural sequence, such that $\mathrm{S}(\cdot) : \mathbb{N} \rightarrow \mathbb{R}^{\sum {n_r}^i}$. In other words, starting from a numerical sequence, this function generates reference commands to guide the closed loop multi-robot team through $\mathcal{W}_{\text{free}}$.

Let us also consider the following assumptions:

Assumption 1 *Each robot R_i knows the complete navigation model of $\mathcal{R}$, defined by the dynamic model of Eq. (3), control law of Eq. (5) and inter-agent constraints of Eq. (4), besides initial condition $^{n}\vec{x}_0$. This complete model represents the team's motion dynamics.*

Assumption 2 *All robots in $\mathcal{R}$ share the same random planning algorithm, including the randomness guaranteed by function of Eq. (6), known a priori.*

Assumption 3 *Robots communicate via a perfect and fully connected network.*

Concerning assumptions 1, 2 and 3, and disregarding possible numerical errors in the computing modules of each agent in $\mathcal{R}$, it is possible to suppose that each robot is capable of predict the behavior of the entire multi-robot system with limited uncertainty in a decentralized form. In this sense, if all agents know the random sequence, using the same random planning technique (e.g., RRT) will allow each robot to predict the behavior of all others, since the same plan is followed by them. That is the underlying idea of our method, which will be detailed in the next sections.

4.3 Multi-agent Rapidly-Exploring Pseudo-Random Tree

The Multi-agent Rapidly-exploring Pseudo-random Tree (MRPT), an extension of the classical CL-RRT to the multi-robot scenario, is presented in this section. The original CL-RRT is composed of two basic procedures: (i) the *planning module*, which promotes the expansion of two trees along the environment by eliminating nodes leading to states of collision; and (ii) the *re-planning and real-time control module*, which calculates an optimal path to be followed by the robot's controller in the trees, re-planning if necessary.

This closed loop approach makes the robot's navigation more robust to disturbances, reason why it was chosen as basis for our method. In the following sections, we discuss our version of these two modules for the multi-robot case. Both algorithms run in a deterministic and decentralized manner on each robot of the team, as presented in the framework overview of Fig. 2.

4.3.1 Trees Expansion

Similarly to CL-RRT, the first step of our algorithm is the expansion of two exploring trees throughout the environment, the *tree of reference commands* $\mathcal{T}_\mathrm{r}$ and the *tree of states* $\mathcal{T}_\mathrm{s}$ of the system. Both trees are simultaneously propagated, so that, for each node in $\mathcal{T}_\mathrm{r}$, there is a corresponding node in $\mathcal{T}_\mathrm{s}$.

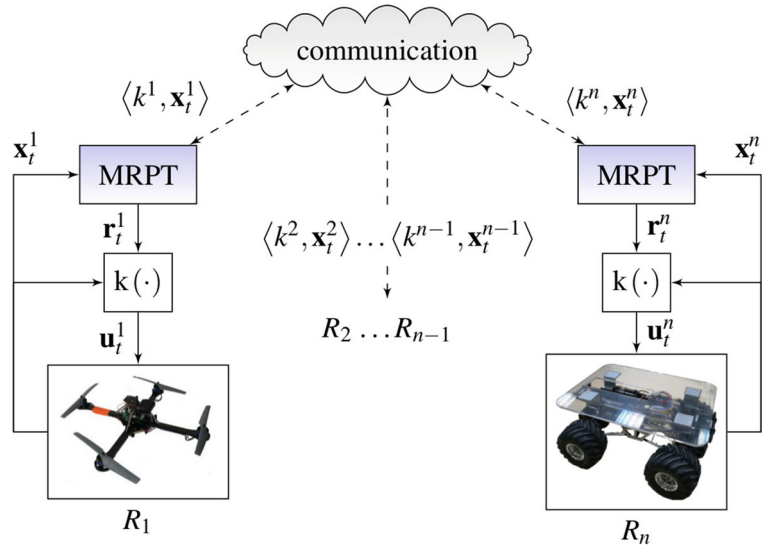

Fig. 2 Methodology framework overview: agents from 1 to n run a distributed version of the Multi-agent Rapidly-exploring Pseudo-random Tree and use communication to synchronize their current state of execution

Algorithm 1 Multi-robot trees expansion

Require: $\mathcal{T}_\mathrm{r}$, $\mathcal{T}_\mathrm{s}$, ${}^n\vec{x}_\mathrm{goal}$, t, k

1: ${}^n\vec{r}_\mathrm{samp} = \mathrm{S}(k) \in \mathcal{W}_\mathrm{free}$
2: ${}^n\vec{r}_\mathrm{near}$ = vertex in $\mathcal{T}_\mathrm{r}$ minimizing $\mathrm{d}\left({}^n\vec{r}_\mathrm{near}, {}^n\vec{r}_\mathrm{samp}\right)$
3: ${}^n\widehat{\vec{x}}_t$ = vertex in $\mathcal{T}_\mathrm{s}$ corresponding to ${}^n\vec{r}_\mathrm{near}$
4: **for** $\tau = t$ **to** $(t+\delta t)$ **do**
5: propagate ${}^n\vec{r}_\tau$ from ${}^n\vec{r}_\mathrm{near}$ to ${}^n\vec{r}_\mathrm{samp}$
6: ${}^n\widehat{\vec{u}}_\tau = \mathrm{K}\left({}^n\vec{r}_\tau, {}^n\widehat{\vec{x}}_\tau\right)$
7: ${}^n\widehat{\vec{x}}_{\tau+1} = \mathrm{F}\left({}^n\widehat{\vec{x}}_\tau, {}^n\widehat{\vec{u}}_\tau, \vec{0}_{n\times n_w}\right)$ subject to constraint of Eq. (4)
8: **if** ${}^n\widehat{\vec{x}}_{\tau+1}$ is feasible **and** ${}^n\vec{r}_\tau \in \mathcal{W}_\mathrm{free}$
9: $\mathcal{T}_\mathrm{r} \leftarrow {}^n\vec{r}_\tau$ **then**
10: $\mathcal{T}_\mathrm{s} \leftarrow {}^n\widehat{\vec{x}}_{\tau+1}$
11: **end if**
12: **if** ${}^n\vec{r}_\tau$ reaches ${}^n\vec{r}_\mathrm{samp}$ **or** ${}^n\widehat{\vec{x}}_{\tau+1}$ reaches ${}^n\vec{x}_\mathrm{goal}$ **then**
13: **break**
14: **end if**
15: **end for**

Ensure: $\mathcal{T}_\mathrm{r}$, $\mathcal{T}_\mathrm{s}$

Consider Algorithm 1, which presents a adapted version of the expansion procedure of the RRT [5] to the multi-robot case. It takes as inputs two trees to be expanded, $\mathcal{T}_\mathrm{r}$ and $\mathcal{T}_\mathrm{s}$, the target region ${}^n\vec{x}_\mathrm{goal}$, the current system time t and the pseudo-random step k, returning as output the trees after they have been expanded.

Initially, a sample ${}^n\vec{r}_\mathrm{samp} \in \mathbb{R}^{\sum {n_r}^i}$ is randomly chosen within $\mathcal{W}_\mathrm{free}$ (line 1), which will be used to propagate the team's model. Then, vertex ${}^n\vec{r}_\mathrm{near}$ belonging to $\mathcal{T}_\mathrm{r}$, and closest to ${}^n\vec{r}_\mathrm{samp}$, is selected (line 2). This closeness is given in terms of minimization of the distance function $\mathrm{d}\left({}^n\vec{r}_\mathrm{near}, {}^n\vec{r}_\mathrm{samp}\right)$, such that $\mathrm{d}(\cdot): \mathbb{R}^{\sum {n_r}^i} \times \mathbb{R}^{\sum {n_r}^i} \rightarrow \mathbb{R}$.

Afterwards, node ${}^n\widehat{\vec{x}}_t \in \mathcal{T}_\mathrm{s}$ correspondent to ${}^n\vec{r}_\mathrm{near} \in \mathcal{T}_\mathrm{r}$ is identified (line 1). From that point, at each new loop iteration, reference and state of the system are propagated towards the sample ${}^n\vec{r}_\mathrm{samp}$ within a finite time interval δt. Reference command evolves according to line 5. An estimated control input ${}^n\widehat{\vec{u}}$ is computed by Eq. (5) to minimize the error between estimated positions ${}^n\widehat{\vec{x}}_\tau$ of robots and its respective desired positions at ${}^n\vec{r}_\tau$.

Finally, an estimate of system states ${}^n\widehat{\vec{x}}$ is propagated according Eq. (3) (line 1), disregarding measurement uncertainties and disturbances by setting ${}^n\vec{W} = \vec{0}_{n\times n_w}$. Each new collision-free state ${}^n\widehat{\vec{x}}_{\tau+1}$ and its corresponding ${}^n\vec{r}_\tau$ are incorporated to $\mathcal{T}_\mathrm{s}$ and $\mathcal{T}_\mathrm{r}$, respectively, and the loop is broken when ${}^n\vec{r}_\tau$ or ${}^n\widehat{\vec{x}}_{\tau+1}$ reaches ${}^n\vec{r}_\mathrm{samp}$ or ${}^n\vec{x}_\mathrm{goal}$.

4.3.2 Real-Time Re-planning

As previously shown, multi-robot tree of states is obtained based on the prediction model of Eq. (3),

disregarding the existence of disturbances and sensory measurement uncertainties. However, real-world robots are always subject to several kinds of uncertainties and, therefore, estimates ${}^{n}\widehat{\vec{x}}$ generated by Alg.1 do not guarantee collision-free navigation nor safely achievement of ${}^{n}\vec{x}_{\text{goal}}$.

Primarily due to this feature, Closed Loop Rapidly-exploring Random Tree presents a real-time execution component which basically uses the system's motion model to propagate its trajectory through the environment from its last measurement (with uncertainties), via the best path found in the tree at the current time. This propagation evaluates the feasibility of this best path from its current position to the next time instant.

Algorithm 2, an extended version of the CL-RRT execution loop for the multi-robot case, is presented next. This modification makes the original procedure less susceptible to problems when an agent R_i does not follow correctly the plan computed for the team. The procedure receives as input only initial ${}^{n}\vec{x}_{\text{init}}$ and final ${}^{n}\vec{x}_{\text{goal}}$ state vectors of the multi-robot system and iterates until agent R_i reaches its respective goal $\vec{x}^{i}_{\text{goal}} \in {}^{n}\vec{x}_{\text{goal}}$.

Algorithm 2 Multi-robot execution loop

Require: ${}^{n}\vec{x}_{\text{init}}$, ${}^{n}\vec{x}_{\text{goal}}$

1: $t = 0, k^i = 1$
2: $\mathcal{T}_{\text{s}} \leftarrow {}^{n}\vec{x}_{\text{init}}, \mathcal{T}_{\text{r}} \leftarrow \vec{0}$
3: **while** $\vec{x}^{i}_{t} \in {}^{n}\vec{x}_{t}$ has not reached $\vec{x}^{i}_{\text{goal}} \in {}^{n}\vec{x}_{\text{goal}}$ **do**
4: communicate to update $\langle k,\ {}^{n}\widehat{\vec{x}}_{t} \rangle$
5: estimate best current path $\mathcal{P} \subset \mathcal{T}_{\text{r}}$
6: **if** $\mathcal{P}$ is empty **then**
7: apply a *safety action* and go to Line 2
8: **end if**
9: ${}^{n}\vec{r}_{t} = {}^{n}\vec{r}\,|_{\mathcal{P}} + \text{h}\left({}^{n}\widehat{\vec{x}}_{t} - {}^{n}\vec{x}_{t}\right)$
10: **if** ${}^{n}\vec{r}_{t} \notin \mathcal{W}_{\text{free}}$ **then**
11: remove infeasible parts of ${}^{n}\vec{r}_{t}$
12: **END IF**
13: apply $\vec{r}^{\,i}_{t} \in {}^{n}\vec{r}_{t}$ to the controller of robot R_i
14: **while** time remaining in ΔT **do**
15: expand $\mathcal{T}_{\text{r}}$ and $\mathcal{T}_{\text{s}}$ (Alg.1)
16: $k = k + 1$
17: **end while**
4: $t = t + \Delta T$
4: **end while**

After initializing the current time t, numerical sequence k^i, and trees $\mathcal{T}_{\text{s}}$ and $\mathcal{T}_{\text{r}}$, robot R_i begins its control loop by updating the k and states ${}^{n}\vec{x}_{t}$ of its version of the system (line 2). This is basically done by communication, as illustrated in Fig. 2, where the state of the whole system is composed by the broadcast of individual states of all agents in a fully connected network, such that ${}^{n}\vec{x}_{t} = \langle \vec{x}^{1}_{t}, \vec{x}^{2}_{t}, \ldots, \vec{x}^{n}_{t} \rangle$. Each robot also has its own k^i index, however, as they need to expand the trees in a coordinated form, k must be synchronize. Then, all robots also broadcast its index and use the current lowest value in the team. In other words, propagation speed is limited by the slowest agent.

It is also reasonable to suppose that state $\vec{x}^{i}_{t}$ of robot R_i is always available to itself (due to direct measurement), but update information about states of other agents depends upon communication. Therefore, the knowledge of ${}^{n}\vec{x}$ by one agent of $\mathcal{R}$ may differ from others, possibly being incorrect due to delays or even momentary communication failures. In consequence, as it is impossible to guarantee reliability of information at the current state of the whole system, it is necessary to consider alternatives to eliminate or minimize problems of collision and other faults in complex missions due to environmental disturbances and uncertainties.

A simple alternative to update ${}^{n}\vec{x}_{t}$ is assuming ${}^{n}\vec{x}_{t} \approx {}^{n}\widehat{\vec{x}}_{t}$. In this slight conservative approach, communication among agents would be completely unnecessary, except to determinate k. Another alternative, probably more interesting, would be the use of a state estimator (like Kalman Filter) to compute ${}^{n}\vec{x}_{t}$ based on measurement uncertainty, correcting the estimate whenever R_i receives information about other robots.

Regardless of which alternative is used at line 2, it seems reasonable to suppose, in coordinated and/or cooperative teams of robots, that the smaller the difference between ${}^{n}\vec{x}_{t}$ and ${}^{n}\widehat{\vec{x}}_{t}$ (line 1 of Alg.1), the lower the risk of collision between members of the team. I.e., if each robot R_i behaves according to its plan (${}^{n}\widehat{\vec{x}}_{t}$), greater are the chances for successful mission completion.

Inspired by assumptions in [6], used to reduce the prediction error for the single robot case, we used this information to predict states of the system, common to all robots, seeking to improve coordination in the team of real-world robots. Such an improvement is given

exactly by decreasing the error ${}^{n}\widehat{\vec{x}}_t - {}^{n}\vec{x}_t$ between prediction (or planning) and current measurement of these states.

Like in the classical CL-RRT, we choose the best path $\mathcal{P} \subset \mathcal{T}_\mathrm{r}$ to be followed in the current state (line 2). This choice is usually guided by minimizing a cost-based heuristic path to the target [16]. Euclidean distance is the most employed one, but several others have been used already [17]. Furthermore, a safety action must be automatically applied if such a path cannot be found at the time. For example, agents can stop moving (if possible) until a new path is found.

Each node of $\mathcal{P}$ corresponds to a reference vector ${}^{n}\vec{r}\,|_{\mathcal{P}}$, predicted by Alg.1, which can be decomposed into trajectory commands to control each individual robot R_i executing its version of the method. It would be the "best" signal reference if error ${}^{n}\widehat{\vec{x}}_t - {}^{n}\vec{x}_t$ was always null, however, as this is not always true, it is necessary to change the reference within the planner to reduce this error. This information must be used to compute the modified reference signal ${}^{n}\vec{r}_t$ (line 9), which will finally be applied to the real robot controller.

Function $\mathrm{h}(\cdot) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_r}$ promotes the reference modification due to the current error. It is commonly neglected in papers applying CL-RRT based approaches, since it may significantly increase the stability analysis of the system. According to [17], every time the robot's controller fails in track the reference command satisfactorily due to disturbances, bad tunning or modeling errors, the planner can adjust $\vec{r}_t$, so that the agent achieves the original plan. However, this additional feedback loop can potentially lead the overall system to an instability condition. In fact, when two modulus, planner and controller, try to decrease the same error, they can overcompensate or negate effects of each other. In our extension to multirobots, the incorporation of $\mathrm{h}(\cdot)$ in the control loop may cause instability in the final MRPT, and the choice of this function has to be made after careful analysis. A good strategy is to design a controller that presents certain robustness in relation to such effects, as discussed in the next section.

Next, the vehicle model is propagated from its current state using the new modified reference $\vec{r}_t^{\,i} \in {}^{n}\vec{r}_t$, and parts leading the system to collision states are removed from $\mathcal{P}$. In the original CL-RRT algorithm, this is called *lazy check approach*, and is used to avoid collision check throughout the entire tree, which can be very costly. The collision-free $\vec{r}_t^{\,i}$ is then applied to the robot's controller. Finally, the algorithm promotes the expansion of the planning trees during the rest of the execution time interval ΔT (line 2). When that time ends, a new iteration of the main loop begins, and this goes on until robot R_i reaches its individual goal.

4.4 Performance Analysis

It is quite difficult to evaluate computational cost of PRM algorithms. Generally speaking, some assumptions and approximations must be done to analyze such methodologies from this point. Concerning classical RRT approaches, the authors of [18] have determined its time complexity as $\mathcal{O}\left(v^2\right)$, where v represents the number of vertexes added to $\mathcal{T}_\mathrm{s}$. In this context, other papers in the literature have concentrated efforts in optimize RRT algorithms by reducing the number of vertexes necessary to reach the goal position.

Concerning Alg.1, which is basically an extension of the classical RRT for multi-robots, we focus on critical parts to evaluate effects of n in our method. It is quite easy to see, for each node added to $\mathcal{T}_\mathrm{r}$ and $\mathcal{T}_\mathrm{s}$, that each robot must propagate the reference signal – Eq. (6) – and calculate the dynamic model and control functions – Eqs.(3) and (5) – for all n of the team in a sequential form. Verify collision inequality of Eq. (4), however, will demand (for the worst-case) an evaluation between two agents of the team, which lead us to an $\mathcal{O}\left(n \log n\right)$ time complexity, as described in [19]. The highest cost at the RRT is exactly at this expansion step, then, we can set Alg.1 as having an $\mathcal{O}\left(v^2 n\right)$ time complexity in general, and $\mathcal{O}\left(v^2 n \log n\right)$ for the worst-case.

Now, concerning Alg. 2, there are 4 steps to evaluate. The communication update is basically an inference of k and ${}^{n}\vec{x}_t$ based upon messages sent by n agents in our perfect communication channel. Even by using a Kalman Filter estimator, it is possible to keep its time complexity linear with respect to n [20]. Estimate the best current path depends upon finding the vertex nearest to the goal, which occurs in $\mathcal{O}\left(v\right)$. Calculate and correct ${}^{n}\vec{r}_t$ commands are $\mathcal{O}\left(n\right)$ operations, since they only depend upon the evaluation of functions with linear behavior concerning n. Finally, inputting the reference signal to the agent's controller is an $\mathcal{O}\left(1\right)$, since it is only applied to the robot itself,

and the time cost of expand the trees was previously discussed.

Therefore, the time complexity of our proposed method can be defined as $\mathcal{O}(n) + \mathcal{O}(v) + \mathcal{O}(n) + \mathcal{O}(1) + \mathcal{O}\left(v^2 n\right) \approx \mathcal{O}\left(v^2 n\right)$ generally speaking, and $\mathcal{O}(n) + \mathcal{O}(v) + \mathcal{O}(n) + \mathcal{O}(1) + \mathcal{O}\left(v^2 n \log n\right) \approx \mathcal{O}\left(v^2 n \log n\right)$ for the worst-case. We will validate this analysis in the next section by measuring the execution time of the MRPT for teams with different number of members.

5 Experiments

In this section we apply the MRPT to a very challenging scenario: air-ground cooperation in outdoor environments. Approximate dynamic models are used to represent the heterogeneous vehicles and a set of simple control laws are employed to stabilize them. Simplified collision models between agents were also applied, specially to avoid position conflicts. Both simulated and real-world experiments are discussed next.

5.1 Quadrotor Dynamic Model

In our experiments we used the Hummingbird UAV from Asctec[1]. The mathematical model, based on Newton-Euler formalism, was adapted from [21], where authors use the same UAV in $\mathcal{W} \equiv \mathbb{R}^3$. We consider the state vector $\vec{x} \in \mathbb{R}^{12}$ and the input vector $\vec{u} \in \mathbb{R}^4$, such that

$$\vec{x} = \begin{bmatrix} \vec{p} & \vec{v} & \psi & \vec{q} \end{bmatrix}' \quad \text{and} \quad \vec{u} = \begin{bmatrix} \delta\omega^T & \delta\omega^\phi & \delta\omega^\theta & \delta\omega^\psi \end{bmatrix}',$$

where $\vec{p} = \begin{bmatrix} x & y & z \end{bmatrix}'$ (in meters) and $\vec{v} = \begin{bmatrix} u & v & w \end{bmatrix}'$ (in meters per second) are linear positions and velocities, respectively, of the robot's center of mass related to the world reference frame $\{\mathcal{W}\} \in \mathbb{R}^3$, $\psi = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}'$ (in radians) is the orientation vector in $\mathbb{SO}(3)$, also related to $\{\mathcal{W}\}$, and $\vec{q} = \begin{bmatrix} p & q & r \end{bmatrix}'$ (in radians per second) is the angular velocity vector with respect to the body reference frame $\{\mathcal{B}\}$ attached to the quadrotor's center of mass. Here, $\delta\omega^T$ represents the nominal thrust speed of the UAV rotors ω, and $\delta\omega^\phi$, $\delta\omega^\theta$ and $\delta\omega^\psi$ are deviations on rotor speeds that causes roll, pitch and yaw angular moments, respectively.

Figure 3 represents output commands of the quadrotor, and its relative pose between world and body reference frames. We use this representation as basis for the modeling stage, presented next.

Time varying linear motion of this robot can be written as

$$\vec{p}_{t+1} = \vec{v}_t, \tag{7}$$

$$m\vec{v}_{t+1} + \vec{C}|\vec{v}_t|\vec{v}_t + m\vec{g} = \lambda\, \delta\omega^T{}_t \begin{bmatrix} \cos\phi_t \sin\theta_t \\ -\sin\phi_t \\ \cos\phi_t \cos\theta_t \end{bmatrix} + \vec{D}, \tag{8}$$

where m represents the quadrotor mass, $\vec{C}$ the drag coefficients matrix, $\vec{g}$ the gravity vector related to $\{\mathcal{W}\}$ (such that $\|\vec{g}\| \approx 9.78\text{m/sec}^2$), λ a positive rotor gain value, and $\vec{D}$ the wind disturbance function.

Similarly, time varying angular motion can be described by

$$\psi_{t+1} = \vec{B}\vec{q}_t, \tag{9}$$

$$\vec{J}\vec{q}_{t+1} + \vec{C}|\vec{q}_t|\vec{q}_t = l\lambda \begin{bmatrix} \delta\omega^\phi{}_t \\ \delta\omega^\theta{}_t \\ \delta\omega^\psi{}_t \end{bmatrix}, \tag{10}$$

where $\vec{J}$ represents the robot's inertial tensor (diagonal due to the vehicle's symmetry), l the wing span and $\vec{B}$ the transformation matrix from $\{\mathcal{B}\}$ to $\{\mathcal{W}\}$, described as

$$\vec{B} = \begin{bmatrix} 1 & \tan\theta_t \sin\phi_t & \tan\theta_t \cos\phi_t \\ 0 & \cos\phi_t & -\sin\phi_t \\ 0 & \dfrac{\sin\phi_t}{\cos\theta_t} & \dfrac{\cos\phi_t}{\cos\theta_t} \end{bmatrix}.$$

Reference commands are given by trajectory vector $\vec{r} = \begin{bmatrix} x^d & y^d & z^d & \psi^d \end{bmatrix}'$, such that desired state $\vec{x}_t^d = \begin{bmatrix} x^d{}_t & y^d{}_t & z^d{}_t & \vec{0}_{5\times1} & \psi^d{}_t & \vec{0}_{3\times1} \end{bmatrix}'$, where x^d, y^d and z^d are desired positions and ψ^d the desired yaw. Then, we use the control law

$$\vec{u}_t = \vec{K}_q \left(\vec{x}_t^d - \vec{x}_t\right) \tag{11}$$

to stabilize the UAV, where $\vec{K}_q$ represents a gain matrix obtained, for example, by fuzzy robust analysis [22]. Finally, rotor speeds are given by

$$\omega_t = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \vec{u}_t. \tag{12}$$

[1] http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-hummingbird/, accessed in March 4, 2017.

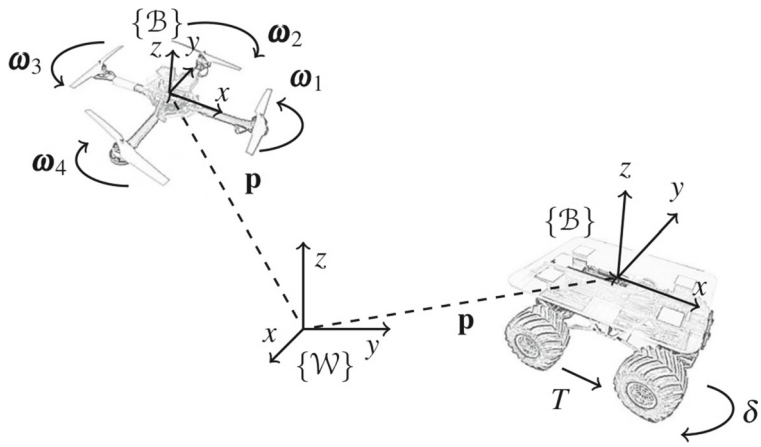

Fig. 3 Reference frames for aerial and ground robots

5.2 Car-Like Dynamic Model

As terrestrial robots, we used in our experiments car-like (Ackerman) UGV platforms with mathematical models also based on Newton-Euler formalism, defined in $\mathcal{W} \equiv \mathbb{R}^2$ with state vector $\vec{x} \in \mathbb{R}^5$ and input vector $\vec{u} \in \mathbb{R}^2$, such that

$$\vec{x} = \begin{bmatrix} \vec{p} & v & \psi & r \end{bmatrix}' \qquad \text{and} \qquad \vec{u} = \begin{bmatrix} T & \delta \end{bmatrix}',$$

where $\vec{p} = \begin{bmatrix} x & y \end{bmatrix}'$ (in meters) and v (in meters per second) are linear positions and velocity variables, respectively, of the robot's center of mass related to the world reference frame $\{\mathcal{W}\} \in \mathbb{R}^2$, ψ (in radians) is the yaw angle in $\mathbb{SO}(2)$, also related to $\{\mathcal{W}\}$, and r (in radians per second) is the angular speed. Here T represents the nominal thrust speed of the UGV and δ the steering angle of front wheels, as presented in Fig. 3.

Time varying linear motion of this robot can be written as

$$\vec{p}_{t+1} = v_t \begin{bmatrix} \cos\psi_t \\ \sin\psi_t \end{bmatrix}, \tag{13}$$

$$m v_{t+1} + c|v_t|v_t = T_t, \tag{14}$$

where m is the mass and c the friction coefficient.

Similarly, time varying angular motion can be described by

$$\psi_{t+1} = r_t, \tag{15}$$

$$J r_{t+1} + c|r_t|r_t = b\, T_t \sin\delta_t, \tag{16}$$

where J represents the robot's inertia and b the distance between front wheels and back wheels axis.

Reference commands are finally given by $\vec{r} = \begin{bmatrix} x^d & y^d & \psi^d \end{bmatrix}$, such that desired state $\vec{x}_t^d = \begin{bmatrix} x^d{}_t & y^d{}_t & 0 & \psi^d{}_t & 0 \end{bmatrix}$, where x^d and y^d are desired positions and ψ^d the desired orientation. Then, we use the control law

$$\vec{u}_t = \vec{K}_c \left(\vec{x}_t^d - \vec{x}_t \right) \tag{17}$$

to stabilize the UGV, where $\vec{K}_c$ represents positive gain matrix.

5.3 MRPT Settings

Since, in our experiments, we are dealing with teams of aerial and ground robots (with $\mathbb{R}^3$ and $\mathbb{R}^2$ workspaces, respectively), it is reasonable to assume that only similar robots can collide with each other. For collisions among ground agents, we represent the constraint function by the following inequality

$$\begin{aligned} \mathrm{G}\left(\vec{x}_t^i, \vec{x}_t^j\right) &\equiv \mathrm{G}\left(\vec{p}_t^{\,i}, \vec{p}_t^{\,j}\right) \leq 0, \\ &\equiv \left\{ \vec{p}_t^{\,i}, \vec{p}_t^{\,j} \in \mathbb{R}^2 : \left\| \vec{p}_t^{\,i} - \vec{p}_t^{\,j} \right\| - \rho \leq 0 \right\}, \end{aligned} \tag{18}$$

where ρ is the minimum radius surrounding each UGV. Without loss of generality, for collisions between UAVs, we can used a similar equation,

$$\mathrm{G}\left(\vec{x}_t^i, \vec{x}_t^j\right) \equiv \left\{ \vec{p}_t^{\,i}, \vec{p}_t^{\,j} \in \mathbb{R}^3 : \left\| \vec{p}_t^{\,i} - \vec{p}_t^{\,j} \right\| - \rho \leq 0 \right\}. \tag{19}$$

In order to apply the MRPT to a given scenario, it is necessary to set others of its features. In Alg.1, line 2, for example, one must define the $\mathrm{d}(\cdot)$ function that gives the distance between ${}^n\vec{r}_{\text{near}}$, an node inside $\mathcal{T}_\text{r}$, and ${}^n\vec{r}_{\text{samp}}$, the randomly chosen sample. Here, we use the summation of the Euclidean distance between all $\vec{r}^{\,i}$ reference commands, such that

$$\mathrm{d}\left({}^n\vec{r}_{\text{near}}, {}^n\vec{r}_{\text{samp}}\right) = \sum_{i=1}^{n} \left\| \vec{r}^{\,i}_{\text{near}} - \vec{r}^{\,i}_{\text{rand}} \right\|. \tag{20}$$

Still concerning Alg.1, at line 5, it is necessary to provide a model by which ${}^n\vec{r}_\tau$ can be propagated from ${}^n\vec{r}_{\text{near}}$ towards ${}^n\vec{r}_{\text{samp}}$. For simplicity reasons, we establish circular formations for the teams of terrestrial and aerial robots separately, and propagate the center of the circle by a first order model $\dot{\vec{x}} = \vec{u}$, leading ${}^n\vec{r}_{\text{near}}$ to move towards ${}^n\vec{r}_{\text{samp}}$. It is also possible to randomly vary some parameters of the circular formation (like radius) in order to avoid collisions with obstacles, but we keep them constant in our experiments.

Finally, correction function $\mathrm{h}(\cdot)$ must be chosen to decrease the influence of disturbance in the system. As previously discussed, this step was basically introduced to reduce the navigation error ${}^n\widehat{\vec{x}}_t - {}^n\vec{x}_t$, increasing the chances of successful accomplish of the mission. However, some instability can be introduced into the whole system, and this problem can't be solved with simple analysis, since it depends upon the coordination of heterogeneous agents. In order to test our approach, we have arbitrarily set this function as an action proportional to the error, such that

$$\mathrm{h}\left({}^n\widehat{\vec{x}}_t - {}^n\vec{x}_t\right) = \alpha\Big[\ \mathrm{P}\left(\widehat{\vec{x}}_t^1 - \vec{x}_t^1\right)\ \ \mathrm{P}\left(\widehat{\vec{x}}_t^2 - \vec{x}_t^2\right)\ \ \ldots\ \ \mathrm{P}\left(\widehat{\vec{x}}_t^n - \vec{x}_t^n\right)\ \Big]', \tag{21}$$

where $\mathrm{P}(\cdot) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_r}$ is a function projecting the robot's state space at the reference space, basically extracting reference variables from $\vec{x}$. Also, $\widehat{\vec{x}}^i$ is the estimated state of robot R_i, extracted from ${}^n\widehat{\vec{x}}$, and α is some positive gain empirically adjusted. Since line 9 of Alg.2 represents a vector summation, here α can be used to adjust the impact of the error correction in the system. For all simulated and real-world experiments presented next, we have set a low value to this gain, $\alpha = 0.1$, to avoid disturb the stability of our multirobot team. Thought, as previously discussed, there is no guarantee about stability, we have observed experimentally that, for $\alpha > 1$, the convergence is compromised.

It is necessary to say that our method, like the classical CL-RRT approach, can also work without such error correction ($\alpha = 0$), but our results will demonstrate the advantageous impact of this function on the completion time of the mission, even when robust control laws are used in the low level controllers of the robots.

Finally, as both kind of platforms can stop moving during navigation (quadrotors can hover at any $\vec{p}_t$), we set the "stopping command" to be the *safety action* described at line 7 of Alg.2.

5.4 Simulated Experiments

As previously explained, we are mainly interested in air-ground cooperation missions, whose complexity we are addressing with our methodology. Our task is relatively simple: a subgroup of UAVs must navigate through a three-dimensional cluttered environment to a goal position, where a subgroup of UGVs waits for aerial coverage to return to a safety area from where the aerial ensemble first left. Our framework was entire built in *Matlab R2014a*, running over a notebook with *Core I5* processor, 4GB memory and *Ubuntu 14.04* OS.

Noise was added to the state vector of our robots to make simulation more realistic and to evaluate effects of uncertainty in our proposed approach. For UAVs, we have corrupted the orientation vector ψ and angular speed vector $\vec{q}$ with standard Gaussian noises $\mathcal{N}(0, , 0.07)$ (in radians) and $\mathcal{N}(0, , 0.28)$ (radians per second), respectively. Position and speed vectors, $\vec{p}$ and $\vec{v}$, were corrupted by *random-walk* signals with about 5meters of uncertainty. Same noise was applied to the corresponding variables of UGVs. In both cases, for aerial and ground vehicles, no estimation filter was used to decrease uncertainty, only linear filters were applied to smooth the measurements. UAVs were also corrupted by a disturbance matrix $\vec{D} \approx \vec{C}\left[\ 5\ 0\ 0\ \right]'$, corresponding to a lateral wind of 5m/s along X axis.

In our first simulated experiment, we defined a team composed by 1 UAV and 4 UGVs. We choose a $\mathbb{R}^3$ space with dimensions 30m × 40m × 5m, cluttered with rectangular obstacles, as shown in Fig. 4.

Figure 4a presents initial configuration of the multirobot system. Aware of UGVs positions, the UAV (whose trajectory is represented by the continuous red line) initiates the MRPT, growing its trees and following the current best path (the black dotted line) towards the other robots (confined to the red circle), as shown in Fig. 4b. The blue circle represents the

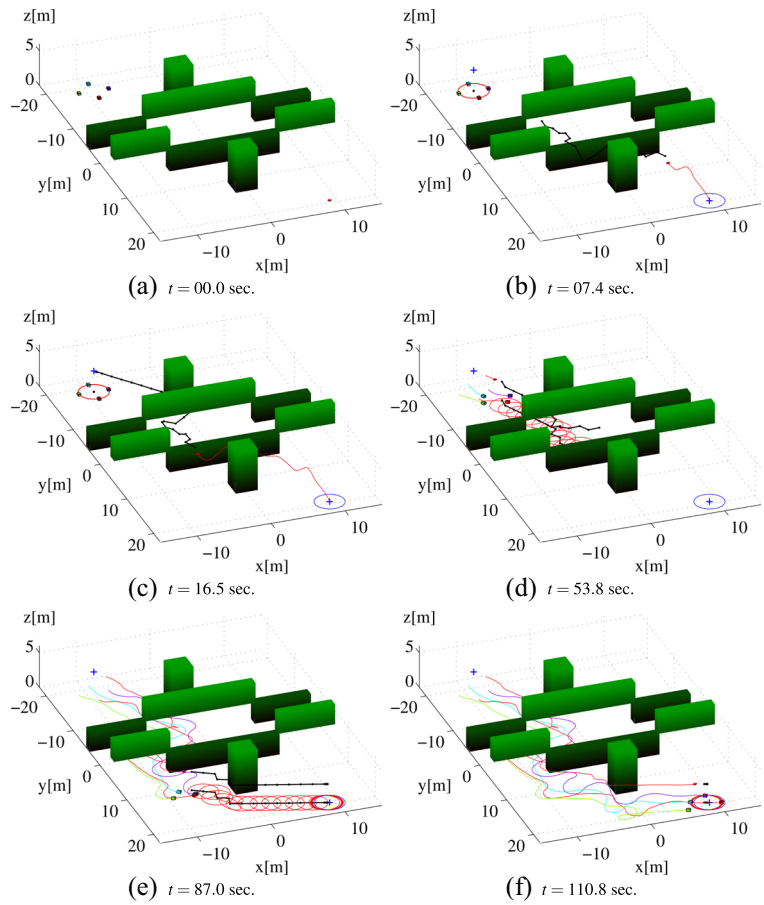

Fig. 4 Simulated experiment: **a** initial configuration of $\mathcal{R}$ with 1 UAV and 4 UGVs; **b** UAV begins the search step; **c** $\mathcal{T}_r$ reaches the first goal; **d** the team begins the coverage/rescue step; **e** $\mathcal{T}_r$ reaches the second safety area; and **f** the team completes the mission

mission basis, to where the team must return at the end of the task.

The algorithm rapidly reaches its first goal at Fig. 4c, after 16,5s, and the mission changes its state from "search" to "rescue" mode, with the subgroup of UGVs been escorted by the aerial agent. Figure 4d shows the growing trees and the team navigating back to the blue circle. The team navigates back in air-ground formation, while its trees reach the final goal, at Fig. 4e, after 87,0s. Mission is successfully accomplished at 110,8s, when all agents arrives in the safety area, as seen in Fig. 4f.

In a second experiment, we now consider a team of 8 UAVs and 16 UGVs, executing the search and rescue task at the same scenario, to demonstrate the applicability to groups with higher numbers of members. Figure 5 shows a top-view of the execution. Once more we can see the trees expanding and the groups of robots executing the two-steps coverage mission. Figure 5a and b present the "search stage", where UAVs move in coordinated formation to meet the UGVs. Figures 5c and d show the "rescue stage", with the entire team navigating back to the basis over aerial-ground coordination.

Next, we have also realized a set of experiments to evaluate time consumption of our method over a large number of robots. To do that, we measured the time spent by one robot of the team in the execution of the main loop described in Alg.2. Repeating the same mission described before, we ran 40 trials with the number of agents varying from 2 to 200 (equally distributed between UAVs and UGVs), and registered all time intervals during each real-time loop. Figure 6 shows the mean and standard deviation of the measurements. Red curve of Fig. 6a represents the time of iterations for the "search stage" when the only agents considered are UAVs, while blue curve of Fig. 6b shows the time for the "rescue stage" when the team have UAVs and UGVs together. Here, it is possible to see that both curves have an approximately linear behavior as

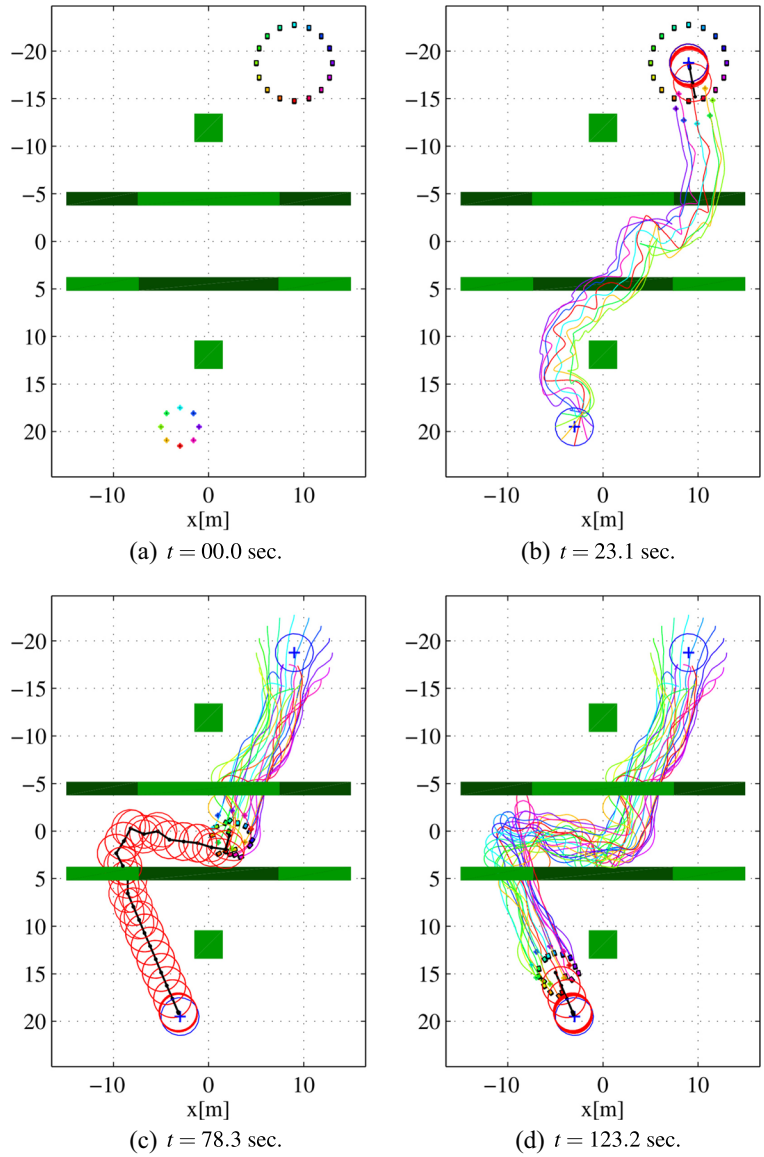

Fig. 5 **a** initial configuration of $\mathcal{R}$ with 8 UAVs and 16 UGVs; **b** aerial robots find ground robots; **c** team returns to the goal; and **d** mission is completed

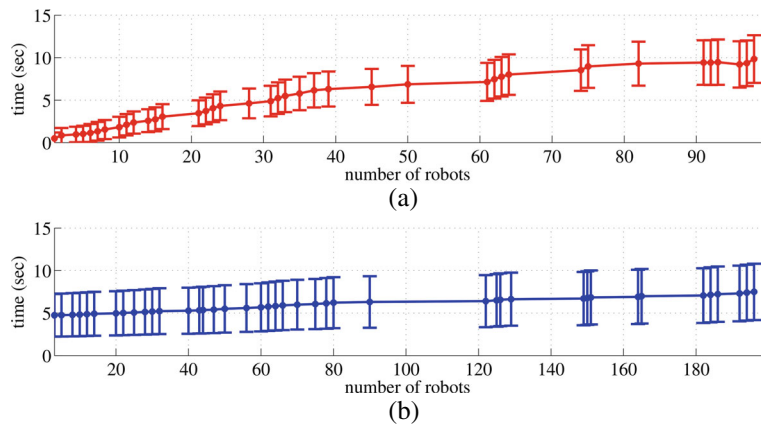

Fig. 6 Time consumption for iterations of Alg.2: **a** *red curve* represents the "search stage", when only UAVs are considered; **b** *blue curve* shows the time for the "rescue stage", with UAVs and UGVs together

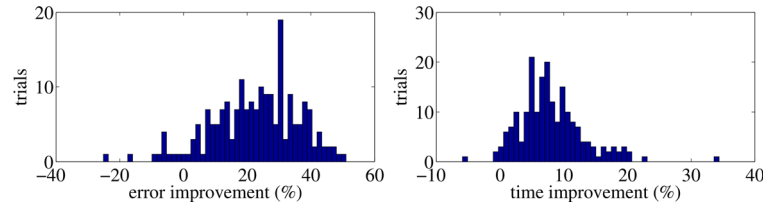

Fig. 7 Impact of the error correction function described at Alg. 2 (line 9): **a** for all evaluated experiments, the error fell 25 % on average for more than 95 % of the cases; and **b** the mission completion time fell 7 % on average for more than 98 % of the cases

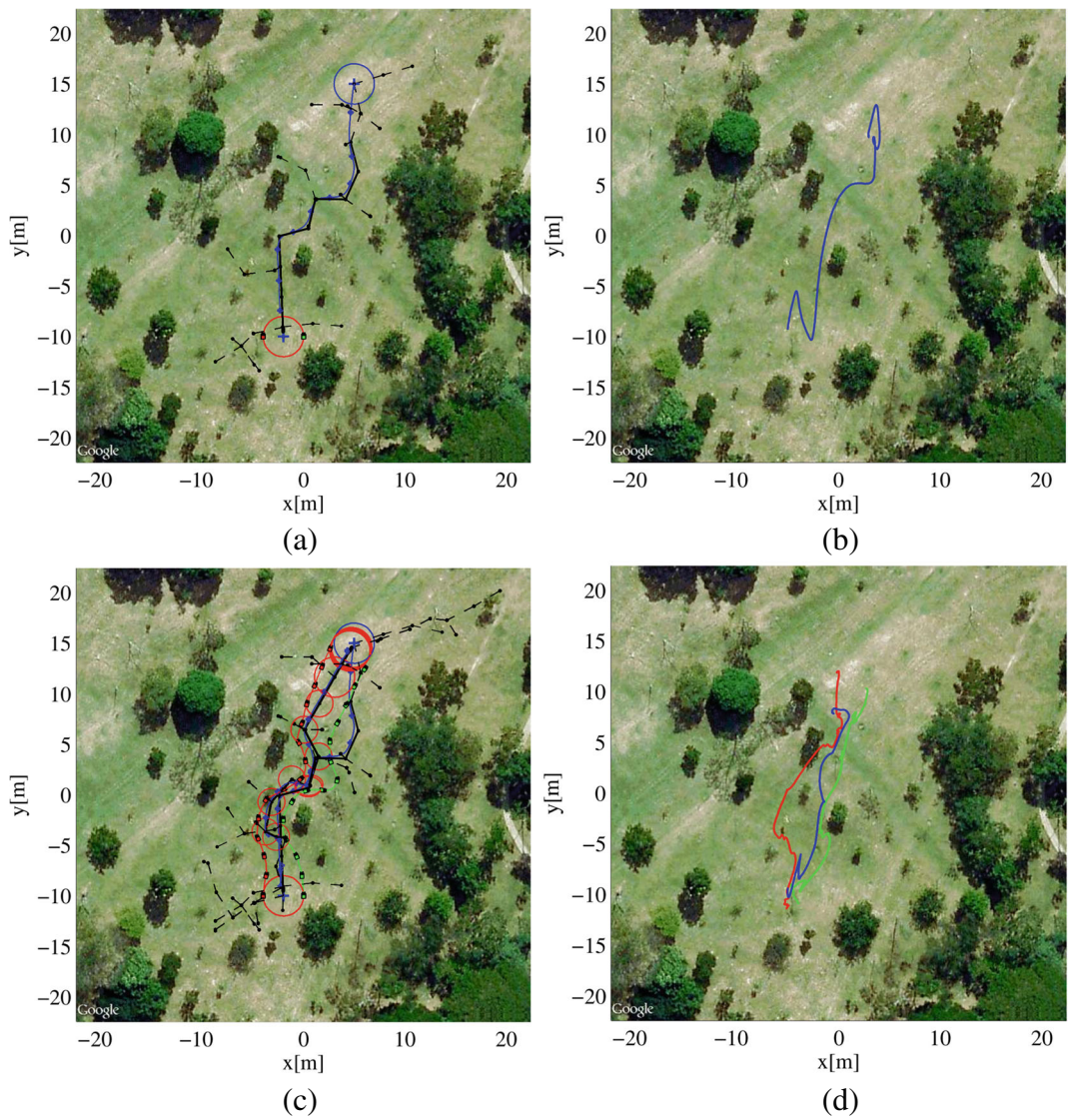

Fig. 8 Real-world experiment with a team of heterogeneous robots from VeRLab (figures based on *Google Maps* images): **a** expansion of $\mathcal{T}_r$ (in *black*) and $\mathcal{P}$ (in *blue*) for the UAV in the "search" mode; **b** real trajectory via GPS localization; **c** expansion of $\mathcal{T}_r$ (in *black*) and $\mathcal{P}$ (in *red* and *green* for the UGVs and blue for the UAV) in the "rescue" mode; and **d** real trajectories, via GPS localization

the number of agents grows, like expected given our performance analysis.

Finally, to evaluate effects of the correction function on our method, we have ran about 400 simulated experiments, with 200 applying our complete proposed methodology and other 200 (using the same initial configuration of previous ones) without correction step of Alg. 2, line 9. We have then compared both set of experiments to verify the error and the mission completion time concerning this difference. Figure 7a shows the histogram of the error improvement, given by the percentage difference between $\|{}^n\widehat{\vec{x}}_t - {}^n\vec{x}_t\|$ for cases with and without the error correction. Here it is possible to see the use of the correction function reduced the error in more than 95 % of the cases, with an average value of approximately 25 %. On the other hand, the time spent on the mission was also smaller in more than 98 % of the trials, with a average value of approximately 7 % as can be seen in Fig. 7b. This was also given by the percentage difference between cases with and without the error correction.

5.5 Real-World Experiment

As a proof of concept, one UAV and two UGVs were used in a real-world experiment, as presented in Fig. 1. Hummingbird quadrotor platform is equipped with on-board sensors for autonomous navigation, like triple-axial magnetic compass, barometric sensor, inertial measurement unit (IMU) module consisting of three gyros and three accelerometers (all with 1 kHz operating rate), global positioning system (GPS) receiver, ZigBee wireless serial link for communication purposes (with 10 Hz operating rate), and two on-board CPUs (one of which is fully programmable).

UGVs, on the other hand, were adapted in-house at the Computer Vision and Robotics Laboratory (VeRLab)/UFMG, based on the *Tamiya TXT-1 Monster Truck* platform. They are equipped with IMU, PWM servo controllers, ZigBee link, and GPS receptor. All planning, communication and control modules were implemented in *Python* language and run on an ASUS Netbook with Ubuntu 11.10 OS at 10 Hz. The minimum curvature radius of the vehicle is approximately 3 m and the control speed was adjusted to the constant velocity of 1 m/s.

The MRPT was calculated in a decentralized manner, on-board for the ground robots and off-board for the aerial robot, due to hardware constraints. The same mission was proposed for this group, as it can be seen in Figs. 8a and b, for the "search" step, and in Figs. 8c and d, for the "rescue" step.[2]

The experiment have demonstrated the feasibility of applying our pseudo-random multi-robot planner to groups of real-world robots, subject to uncertainties and disturbances in cluttered environments.

6 Conclusion and Future Work

A novel approach for distributed motion planning and control of groups with heterogeneous and underactuated robots subject to uncertainties in cluttered constrained environments was presented in this paper. In our framework, a decentralized version of the well-known CL-RRT algorithm allows each agent to navigate through the environment avoiding collisions with obstacles and other agents of the team, following a plan. In both, simulated and real-world experiments, coordination among agents was obtained by predicting the team's behavior and by decreasing the prediction error of the model.

A disadvantage of our method is related to scalability. Due to the fact that each robot has to calculate the motion prediction of the entire team, in order to execute its own planned trajectory without collision, the number of team members shall not be very large. In fact, as previously demonstrated, the time complexity of our method grows linearly with the number of agents, presenting an $\mathcal{O}\left(v^2 n\right)$ cost for common cases and $\mathcal{O}\left(v^2 n \log n\right)$ for the worst-case, where v is the number of vertexes added to $\mathcal{T}_s$. In problems considering, for example, robot swarms, i.e., teams with hundreds or even thousands agents, it would lead to the need of more computing capacity on each robot, breaking paradigms of hardware simplicity considered in this specific scope. However, we believe the continuous technological advances will make it possible to increase computational capacity of such systems, such that communication may become a more serious hindrance as the number of robots in a team increases.

As a perfect network was assumed, another important conclusion is that send only vector state informations to other robots is better than send the entire tree

[2] A video of the complete execution is available at: https://youtu.be/pge9fNxQwPk.

(as done in [11]), since the first case only depends upon n and the second depends upon v, and $v \gg n$.

A possible solution to minimize the computational cost of our technique in large groups may be the application of a hierarchy of priority among robots, so that each agent will only need to compute the predictive model of its nearest neighbors. We are currently investigating the use of our method to a broader class of missions, beyond the formation and navigation control in complex environments. We also believe the ${}^{n}\vec{r}$ can easily be generalized to tasks using robot teams without formation constraints, but for whom navigation is a critical problem to the coordination process. Another important topic is to guarantee stability in the whole team navigation. This is quite simple when dealing with homogeneous groups of robots, but very complex in heterogeneous teams. Here, only local stability (for each robot) can be observed, but in the future we can concentrate at this issue to ensure feasibility to the movement of the team as a whole.

Acknowledgments This work was developed with support of Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG).

References

1. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1), 90–98 (1986)
2. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (June 2005)
3. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Illinois (2006)
4. Lee, S.M., Kim, H., Myung, H., Yao, X.: Cooperative coevolutionary algorithm-based model predictive control guaranteeing stability of multirobot formation. IEEE Trans. Control Syst. Technol. **23**(1), 37–51 (2015)
5. LaValle, S., Kuffner J., Jr: Randomized kinodynamic planning. IEEE Int. Conf. Robot. Autom. **1**, 473–479 (1999)
6. Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E., How, J.P.: Motion planning in complex environments using closed-loop prediction. In: AIAA Guidance, Navigation, and Control Conference and Exhibit (2008)
7. Luders, B.D., Karaman, S., Frazzoli, E., How, J.P.: Bounds on tracking error using closed-loop rapidly-exploring random trees. In: American Control Conference (2010)
8. Kamio, S., Iba, H.: Cooperative object transport with humanoid robots using RRT path planning and re-planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2006)
9. Vahrenkamp, N., Kuhn, E., Asfour, T., Dillmann, R.: Planning Multi-Robot Grasping Motions. In: IEEE/RAS International Conference on Humanoid Robots, Nashville (2010)
10. Frazzoli, E.: Quasi-random algorithms for real-time spacecraft motion planning and coordination. Acta Astronaut. **53**(4-10), 485–495 (2003)
11. Desaraju, V.R., How, J.P.: Decentralized path planning for multi-agent teams with complex constraints. Auton. Robot. **32**(4), 385–403 (2012)
12. Otte, M., Bialkowski, J., Frazzoli, E.: Any-com collision checking: Sharing certificates in decentralized multi-robot teams. In: IEEE International Conference on Robotics and Automation, pp. 563–570 (2014)
13. Alves Neto, A., Macharet, D.G., Campos, M.F.M.: On the generation of trajectories for multiple uavs in environments with obstacles. J. Intell. Robot. Syst. **57**(4), 123–141 (2010)
14. Alves Neto, A., Macharet, D.G., Chaimowicz, L., Campos, M.F.M.: Path planning with multiple rapidly-exploring random trees for teams of robots. in IEEE International Conference on Advanced Robotics. Montevideo, Uruguay (2013)
15. LaValle, S.M., Branicky, M.S., Lindemann, S.R.: On the relationship between classical grid search and probabilistic roadmaps. Int. J. Robot. Res. **23**(7–8), 673–692 (2004)
16. Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. AIAA J. Guid. Control. Dyn. **25**, 116–129 (2002)
17. Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., How, J.P.: Real-time motion planning with applications to autonomous urban driving. IEEE Trans. Control Syst. Technol. **17**(5), 1105–1118 (2009)
18. Svenstrup, M., Bak, T., Andersen, H.J.: Minimising computational complexity of the rrt algorithm a practical approach. In: IEEE International Conference on Robotics and Automation, pp. 5602–5607. (2011)
19. Vemuri, B.C., Cao, Y., Chen, L.: Fast collision detection algorithms with applications to particle flow. Comput. Graph. Forum **17**(2), 121–134 (1998)
20. Pnevmatikakis, E.A., Rad, K.R., Huggins, J., Paninski, L.: Fast kalman filtering and forward-backward smoothing via a low-rank perturbative approach. J. Comput. Graph. Stat. **23**(2), 316–339 (2014)
21. Michael, N., Mellinger, D., Lindsey, Q., Kumar, V.: The grasp multiple micro-uav testbed. IEEE Robot. Autom. Mag. **17**(3), 56–65 (2010)
22. Mozelli, L.A., Alves Neto, A., Campos, M.F.M.: Attitude of quadrotor-like vehicles: Fuzzy modeling and control with prescribed rate of convergence. In: IEEE International Conference on Robotics and Automation, pp. 1710–1715. (2015)

Armando Alves Neto is Assistant Professor at the Department of Electronic Engineering at Universidade Federal de Minas Gerais. He received the B.S.E. degree in Automation and Control Engineering from the UFMG in 2006, and S.M. and Ph.D. degrees in Computer Science from UFMG in 2008 and 2012, respectively. Research interests include real-time motion planning, multi-agent control, robust control, and collision avoidance strategies.

Douglas G. Macharet is currently an Assistant Professor at the Department of Computer Science (DCC) at the Federal University of Minas Gerais (UFMG). He received M.Sc. and D.Sc. degree in Computer Science from the same university in 2009 and 2013, respectively. He is with the Computer Vision and Robotics Laboratory (VeRLab), and his main research interests are in mobile robotics, focusing on localization, mapping, path planning for nonholonomic robots, multi-robots systems and human-robot interaction.

Mario F. M. Campos received B.S. degrees in Engineering and the M.S. degree in Computer Science from the Universidade Federal de Minas Gerais, Belo Horizonte, Brazil and the Ph.D. degree in Computer and Information Science from the University of Pennsylvania, Philadelphia. He is Full Professor of Computer Vision and Robotics in the Department of Computer Science at the Federal University of Minas Gerais (UFMG). Research interests include cooperative robotics, robot vision, and sensor information processing. His main contributions are in haptics, multi-robot cooperation, and robot vision. He is the Founder and Director of the Vision and Robotics Lab (VeRLab), UFMG.