Autonomous Robots
https://doi.org/10.1007/s10514-018-9783-9

Distributed multi-robot formation control in dynamic environments

Javier Alonso-Mora[1] · Eduardo Montijano[2] · Tobias Nägeli[3] · Otmar Hilliges[3] · Mac Schwager[4] · Daniela Rus[5]

Received: 15 February 2017 / Accepted: 10 July 2018

Abstract
This paper presents a *distributed* method for formation control of a homogeneous team of aerial or ground mobile robots navigating in environments with static and dynamic obstacles. Each robot in the team has a finite communication and visibility radius and shares information with its neighbors to coordinate. Our approach leverages both constrained optimization and multi-robot consensus to compute the parameters of the multi-robot formation. This ensures that the robots make progress and avoid collisions with static and moving obstacles. In particular, via distributed consensus, the robots compute (a) the convex hull of the robot positions, (b) the desired direction of movement and (c) a large convex region embedded in the four dimensional position-time free space. The robots then compute, via sequential convex programming, the locally optimal parameters for the formation to remain within the convex neighborhood of the robots. The method allows for reconfiguration. Each robot then navigates towards its assigned position in the target collision-free formation via an individual controller that accounts for its dynamics. This approach is efficient and scalable with the number of robots. We present an extensive evaluation of the communication requirements and verify the method in simulations with up to sixteen quadrotors. Lastly, we present experiments with four real quadrotors flying in formation in an environment with one moving human.

Keywords Multi-robot systems · Distributed robotics · Formation control · Dynamic environments · Collision avoidance · Unmanned aerial vehicles · Drones · Micro air vehicles

1 Introduction

Multi-robot systems will be ubiquitous to perform many tasks, such as surveillance (Schwager et al. 2011), inspection (Suzuki et al. 2000), factory automation (Alonso-Mora et al. 2015a), logistics (Wurman et al. 2008) or cinematography (Nägeli et al. 2017b). While some of these problems require team navigation in a rigid pattern, other scenarios, such as cooperative manipulation of deformable objects or transportation of cable-suspended loads, allow for more flexibility, yet requiring certain level of coordination. This is also the case, for example, for a team of robots that fly through narrow canyons while preserving inter-robot communication

This work was supported in part by pDOT ONR N00014-12-1-1000, the Boeing Company, the MIT-Singapore Alliance on Research and Technology under the Future of Urban Mobility, Spanish Project DPI2015-69376-R (MINECO/FEDER), Microsoft Research and the Netherlands Organisation for Scientific Research NWO-TTW Veni 15916. We are grateful for their support.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10514-018-9783-9) contains supplementary material, which is available to authorized users.

✉ Javier Alonso-Mora
j.alonsomora@tudelft.nl

Eduardo Montijano
emonti@unizar.es

Tobias Nägeli
naegelit@ethz.ch

Mac Schwager
schwager@stanford.edu

Daniela Rus
rus@mit.edu

[1] Department of Cognitive Robotics, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

[2] Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, Zaragoza, Spain

[3] Department of Computer Science, ETH Zurich, Zürich, Switzerland

[4] Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA

[5] Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, 32 Vassar St, Cambridge, MA 02139, USA

Published online: 20 July 2018

or visibility. In this paper we present a method for formation control that is ideally suited for these kind of flexible multi-robot formations, since our approach is capable of adjusting several parameters of the formation dynamically to avoid collisions with the environment.

Multi-robot navigation in formation has received extensive attention in the past, with many works considering obstacle-free scenarios. In this work we leverage efficient constrained optimization, multi-robot consensus and geometric reasoning to achieve *distributed* formation control in environments with static and moving obstacles. In contrast to our previous work (Alonso-Mora et al. 2017), we consider the case where robots are no more centrally controlled, but instead have a limited field of view and communicate with their immediate neighbors to coordinate.

Given a set of target formation shapes, our method optimizes the parameters (such as position, orientation and size) of the multi-robot formation in a neighborhood of the robots. The method guarantees that the team of robots remains collision-free by rearranging its formation, see Fig. 1 for an example with four quadrotors in a square formation. A simplified global planner can use this method to navigate the group of robots from an initial location to a final location. This global planner may consist of a series of waypoints for the formation center. A human may also provide the global path for the formation, or a desired velocity, and the robots will adapt their configuration automatically.

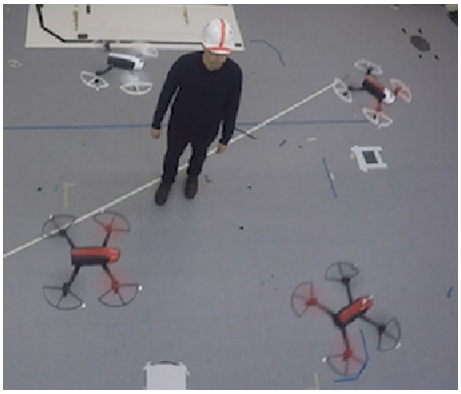


Fig. 1 Four drones in a square formation avoid a walking human

1.1 Related works

A large part of the literature in multi-robot navigation with obstacles considers solutions designed for ground robots operating on the plane. These techniques include using a set of reactive behaviors (Balch and Arkin 1998), potential fields (Balch and Hybinette 2000; Sabattini et al. 2011), abstractions (Michael et al. 2008; Ayanian and Kumar 2010a), decentralized feedback laws with graph theory (Desai et al. 2001), proximity constraints (Ayanian and Kumar 2010b) and stochastic planning (Urcola et al. 2017), to name a few. In contrast, our method automatically optimizes for the formation parameters natively in three-dimensional dynamic environments.

The use of distributed consensus algorithms (Ren and Beard 2008), where each robot only needs to interact with nearby team mates, has also led to a wide variety of formation control strategies, as shown in the survey by Oh et al. (2015). Regarding the robot dynamics, Lin et al. (2005) considered unicycle robots, Dong et al. (2015) considered aerial vehicles and Hatanaka et al. (2012) considered motion in SO(3). Although our method does not directly model the robot dynamics in the computation of the formation parameters, it relays on low-level controllers to drive each robot towards its individual position in the formation while respecting its dynamic constraints. We show experiments with a team of quadrotors. In terms of the sensing model, Franchi et al. (2012) considered relative bearing measurements, Oh and Ahn (2011) considered inter-robot distances and Mostagh et al. (2009) and Montijano et al. (2016) employed explicit vision measurements to estimate the relative positions of neighboring robots and reach the formation. A common assumption in these approaches is the lack of obstacles in the environment, focusing on the design of low-level controllers for each robot to reach the desired formation pattern. Our method is different, in the sense that we exploit the consensus algorithm to agree upon high-level navigation concepts, such as a large convex region reachable by the robots and which does not intersect any of the observed obstacles.

Decentralized solutions with local sensing and communication in environments with obstacles were treated by Mosteo et al. (2008) and Nestmeyer et al. (2017). Both approaches dealt with the problem of task assignment for the team of robots under the assumption of a known map, taking into account connectivity constraints in the plan execution. Compared to them, in our solution we do not make any assumption about structure or knowledge of the map, although we consider a global target goal for all the robots, to which the team needs to move in formation.

Our approach relies on convex and non-convex optimization methods to obtain the locally optimal state of the formation. Several approaches have formulated the navigation of teams of robots as an optimization problem. In particular, convex optimization frameworks for navigating in formation include semidefinite programming (Derenick et al. 2010), which considers only 2D circular obstacles; distributed quadratic optimization (Alonso-Mora et al. 2015a), without global coordination; distributed optimization with discrete-time communications by Kia et al. (2016), which considers a global function defined by the sum of individual costs; and second order cone programming (Derenick and Spletzer 2007), which triangulates the free 2D space to compute the optimal motion in formation. Our method applies to

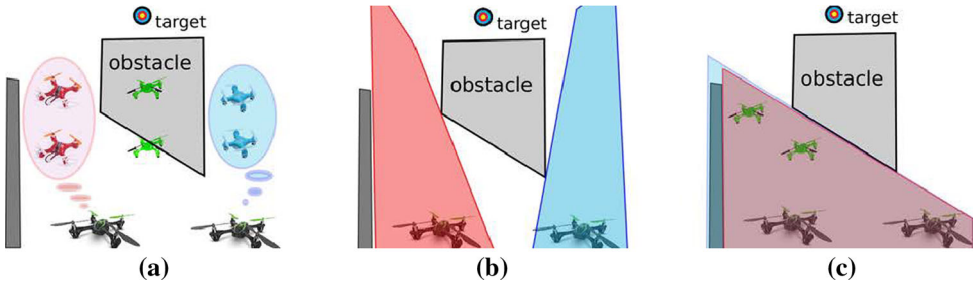


Fig. 2 Example of three approaches for distributed formation planning with obstacles. **a** Each robot independently computes a target formation (red/blue). Consensus on the formation's parameters (green) could lead to a formation in collision with the obstacle. **b** Each robot computes an obstacle-free region, but their intersection could be empty. **c** Our approach, see Sect. 2.4, with target formation after consensus (green). The target formation is computed within the intersection of convex regions, which contain all the robots (Color figure online)

polygonal obstacles and does not require a triangulation of the environment.

Centralized non-convex optimizations include a mixed integer approach by Kushleyev et al. (2013) and a discretized linear temporal logic approach by Saha et al. (2014). Both require high computational effort and can only be applied offline to precompute trajectories. Our goal is to have real-time capability for online computation. Online sequential convex programming has been employed by Augugliaro et al. (2012) and Chen et al. (2015) to compute collision-free trajectories for multiple Micro Air Vehicles (MAVs), but without considering formations. The assignment of robots to the target positions in the formation is another optimization problem that was solved with a centralized algorithm by Turpin et al. (2014) or with a distributed algorithm, albeit in environments without obstacles, by Montijano and Mosteo (2014) and Morgan et al. (2016). Building upon the centralized, yet online, method by Alonso-Mora et al. (2017), we propose an optimization and consensus based approach to reconfigure the formation in dynamic environments, which is distributed and online.

Several works have proposed distributed constrained optimization approaches to maintain a formation, based on Model Predictive Control (MPC). In particular, Keviczky et al. (2008) computed a set of inputs for a team of aerial vehicles navigating in, or towards, a given formation, and Kuriki and Namerikawa (2015) relied on a leader to compute the formation configuration. Our approach is leaderless and can adjust the size of the formation to avoid obstacles.

1.2 Contribution

The main contribution of this paper is a *distributed* method for formation control. The method enables a team of ground or aerial robots to navigate in a dynamic environment while reconfiguring their formation to avoid collisions with static and moving obstacles. A descriptive idea of the method is shown in Fig. 2.

We present a holistic method where we rely on convex optimization techniques to compute a convex region in free position-time space and on non-convex optimization techniques to compute the configuration for the team of robots. We introduce distributed consensus algorithms to obtain:

- The convex hull of the robot's positions.
- The preferred direction of motion.
- A convex region in free position-time space, given by the intersection of individual regions.

We provide a formal analysis with convergence guarantees of the distributed algorithms composing the holistic approach, simulations with teams of robots and experiments with four quadrotors avoiding a human. Our algorithms are easy to implement, require small computational efforts and scale well with the number of robots, as opposed to equivalent flooding methods.

An earlier version of this paper was published by Alonso-Mora et al. (2016). In this version we extend the method with an additional consensus round to compute the preferred direction of motion and a validation of the method in experiments with four real drones navigating in an environment with a human.

Our proposed method is intended for local motion planning and therefore, deadlocks may arise. To avoid deadlocks, our method can be employed in combination with a global planner, in a manner similar to the work on centralized formation control by Alonso-Mora et al. (2017). In this work we have introduced an additional step in the method, where robots compute the preferred direction of motion. This additional step is intended to avoid disagreements in the case that each robot computes an independent global path, and to coordinate the intentions of all robots. This is done in a max-min consensus step where the best direction of movement is chosen with respect to all the robots.

2 Preliminaries

In this section we provide the needed definitions, the problem formulation for distributed formation control in dynamic environments and an overview of the proposed method.

2.1 Definitions

2.1.1 Robots

Consider a team of robots navigating in formation. For each robot $i \in \mathcal{I} = \{1, \ldots, n\} \subset \mathbb{N}$, its position at time t is denoted by $\mathbf{p}_i(t) \in \mathbb{R}^3$. In the following, we consider all robots to have the same dynamic model and cylindrical non-rotating shape of radius r and height $2h$ in the vertical dimension. Denote the volume occupied by a robot at position $\mathbf{p}$ by $\mathcal{A}(\mathbf{p}) \subset \mathbb{R}^3$.

2.1.2 Communication

Let $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ be the communication graph associated to the team of robots. Each edge in the graph, $(i, j) \in \mathcal{E}$, denotes the possibility of robots i and j to directly communicate with each other. The set of neighbors of robot i is denoted by $\mathcal{N}_i$, i.e., $\mathcal{N}_i = \{j \in \mathcal{I} \mid (i, j) \in \mathcal{E}\}$. We assume ideal communications, i.e., noise-free and without packet losses, and that $\mathcal{G}$ is connected, i.e., for every pair of robots i, j there exists a path of one or more edges in $\mathcal{E}$ that links robot i to robot j. We denote by d the diameter of $\mathcal{G}$, which is the longest among all the shortest paths between any pair of robots.

2.1.3 Field of view

We consider that each robot i has a limited field of view, typically a sphere of given radius centered at the robot's position. We denote it by $\mathcal{B}_i \subset \mathbb{R}^3$.

2.1.4 Static obstacles

Consider a set of static obstacles $\mathcal{O} \subset \mathbb{R}^3$ defining the global map, and $\mathcal{O}_i = \mathcal{B}_i \bigcap \mathcal{O}$ the set of obstacles seen by robot i. Further denote by $\bar{\mathcal{O}}_i$ the set $\mathcal{O}_i$ dilated by half of the robot's volume, i.e., the positions for which the robot of cylindrical shape would be in collision with any of the obstacles within its visibility radius, formally

$$\bar{\mathcal{O}}_i = \{\mathbf{p} \in \mathbb{R}^3 | \mathcal{A}(\mathbf{p}) \cap \mathcal{O}_i \neq \emptyset\}. \tag{1}$$

2.1.5 Moving obstacles

Moving, or dynamic, obstacles within the field of view of robot i can be accounted for. Consider $j \in \mathcal{J}_i = \{1, \ldots, n_{DO,i}\} \subset \mathbb{N}$ the list of observed moving obstacles of shape $\mathcal{D}_j \subset \mathbb{R}^3$ by robot i. We denote by $\mathcal{D}_{j|i}(t)$ the volume occupied by the dynamic obstacle j at time t, as seen by robot i and

$$\bar{\mathcal{D}}_{j|i}(t) = \{\mathbf{p} \in \mathbb{R}^3 | \mathcal{A}(\mathbf{p}) \cap \mathcal{D}_j(t) \neq \emptyset\}, \tag{2}$$

its dilation by half of the robot's volume. In our implementation we assume that dynamic obstacles maintain a constant velocity.

2.1.6 Position-time obstacles

We rely on the notion of position-time space, where the time dimension is added to the workspace to account for moving obstacles. This is similar to the concept of configuration-time space introduced by Erdmann and Lozano-Perez (1987), but differs in that it is embedded in $\mathbb{R}^4$ instead of in the potentially large high-dimensional space. We denote the current time by t_0 and consider a time horizon τ, typically a few seconds in the future. For robot i and current time t_0, we denote the union of static and dynamic obstacles seen by robot i by

$$\hat{\mathcal{O}}_i(t_0) = \bar{\mathcal{O}}_i \times [0, \tau] \cup \bigcup_{\substack{t \in [0,\tau] \\ j \in \mathcal{J}_i}} \bar{\mathcal{D}}_{j|i}(t_0 + t) \times t \subset \mathbb{R}^4,$$

where $\times$ denotes the Cartesian product of two spaces, in this case the workspace and the time dimension.

2.1.7 Position-time free space

For robot i and current time t_0, the free space in the position-time space is then

$$\bar{\mathcal{F}}_i(t_0) = \mathbb{R}^3 \times [0, \tau] \setminus \hat{\mathcal{O}}_i(t_0) \subset \mathbb{R}^4. \tag{3}$$

This set represents the positions at which the robot does not collide with any static or moving obstacle at a given time within the time horizon τ.

2.1.8 Motion planning

This work presents an approach for local navigation. We consider that a desired goal position for the team of robots is given, and known by all robots. This global position could be given by a human operator or a standard sampling based approach for global planning, and is outside the scope of this work. Denote by $\mathbf{g}(t) \in \mathbb{R}^3$ the goal position for the centroid of the formation at time t. Our *distributed* local planner then computes the configuration state of the target formation and the required motion of the robots for a given time horizon $\tau > 0$, which must be longer than the required time to stop. We denote $t_1 = t_0 + \tau$.

2.2 Definition of the formation

We consider a pre-defined set of $m \in \mathbb{N}$ template formations, such as a square or a line. See Fig. 3 for an example. Each template formation $f \in \mathcal{I}_f = [1, m]$ is given by a set of robot positions $\{\mathbf{r}^f{}_{0,1}, \ldots, \mathbf{r}^f{}_{0,n}\}$ and a set of outer vertices $\{\mathbf{w}^f{}_1, \ldots, \mathbf{w}^f{}_{n_f}\}$ relative to the center of rotation (typically the centroid) of the formation, where n_f denotes the number of outer vertices defining formation f. The set of vertices represents the convex hull of the robot's positions in the formation, thus reducing the complexity for formations with a large number of robots.

Further denote by d_f the minimum distance between any given pair of robots in the template formation f. Template formations can be defined by a human designer or automatically computed for optimal representation of a target shape as showed by Schoch et al. (2014).

A formation is then defined by an isomorphic transformation, which includes the size $s \in \mathbb{R}_+$, a translation $\mathbf{t} \in \mathbb{R}^3$ and a rotation $R(\mathbf{q})$ described by a unit quaternion $\mathbf{q} \in SO(3)$, its conjugate denoted by $\bar{\mathbf{q}}$. With this formation definition, the configuration state for the team of robots is fully defined by $\mathbf{z} = [\mathbf{t}, s, \mathbf{q}] \in \mathbb{R}^3 \times \mathbb{R}_+ \times SO(3)$.

Given the configuration state $\mathbf{z}$, and template formation number f, the robot positions and outer vertices of the resulting formation are computed by an isomorphic transformation,

$$\begin{aligned} \mathbf{r}^f_i &= \mathbf{t} + s\, R(\mathbf{q})\mathbf{r}^f_{0,i},\ \forall i \in [1, n], \\ \mathbf{v}^f_j &= \mathbf{t} + s\, R(\mathbf{q})\mathbf{w}^f_j,\ \forall j \in [1, n_f], \end{aligned} \tag{4}$$

where the rotation in $SO(3)$ is given by the quaternion operation

$$\left[0,\ R(\mathbf{q})\mathbf{w}^f_j\right]^T = \mathbf{q} \times \left[0,\ \mathbf{w}^f_j\right]^T \times \bar{\mathbf{q}}. \tag{5}$$

For template formation f and configuration state $\mathbf{z}$ we denote the set of outer vertices by

$$\mathcal{V}(\mathbf{z}, f) = [\mathbf{v}^f_1, \ldots, \mathbf{v}^f_{n_f}]. \tag{6}$$

In this paper we rely on this definition for the formation. The method is general and could be applied to alternative definitions, such as for a team of mobile manipulators carrying a rigid object, as shown by Alonso-Mora et al. (2017) for *centralized* formation control.

2.3 Problem formulation

Consider a team of n mobile robots, each robot i with a limited field of view $\mathcal{B}_i$ and a communication graph $\mathcal{G}$. Consider also a set of m template formations known by all the robots in the team. For a template formation, the configuration state $\mathbf{z}$ fully defines the positions of the robots in the formation.

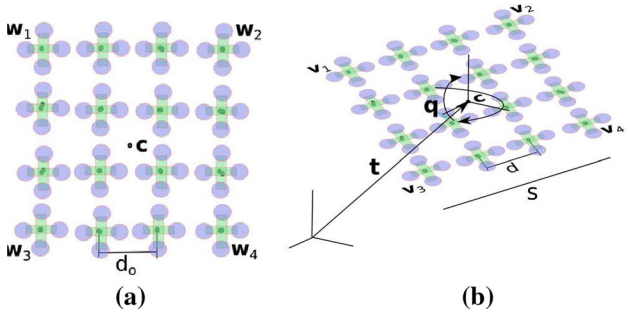


Fig. 3 **a** Example of a template square formation with sixteen MAVs with the four vertices defining the convex hull. **b** The formation can be transformed with a translation $\mathbf{t}$, a 3D rotation $\mathbf{q}$ and a size s isomorphic transformation

Consider also a set of static and moving obstacles seen by the robots and a prediction of their future positions for a time horizon τ. From this information, each robot individually computes the free position-time space $\bar{\mathcal{F}}_i(t_0)$.

Our method solves the following two problems jointly.

Problem 1 (*Optimal target configuration*) At the current time t_0, obtain a goal configuration $\mathbf{z}^*$ and formation index $f^* \in \mathcal{I}_f$ for time $t_1 = t_0 + \tau$ such that the deviation between the robot team's centroid and a desired position $\mathbf{g}$ is minimized, and the robot positions in the target formation are collision free with respect to all observed obstacles, that is $\mathcal{V}(\mathbf{z}^*, f^*) \times t_1 \subset \bigcup_{i \in \mathcal{I}} \bar{\mathcal{F}}_i(t_0)$.

Problem 2 (*Collision-free motion*) Given the current position at time t_0 of all robots, ensure that the transition from their current positions, $\mathbf{p}_1(t_0) \ldots \mathbf{p}_n(t_0)$, to their assigned positions, $\mathbf{r}_1(t_1) \ldots \mathbf{r}_n(t_1)$, in the target formation is collision free at all time instances until t_1, i.e., for all robot $i \in \mathcal{I}$ and time $t \in [t_0, t_1]$ its position satisfies $\mathbf{p}_i(t) \times t \subset \bigcup_{i \in \mathcal{I}} \bar{\mathcal{F}}_i(t_0)$.

In the following, and for clarity, we will drop the time index whenever it is self-evident and denote $\mathbf{p}_i(t_0)$ by $\mathbf{p}_i$.

2.4 Method overview

In the following we give an intuitive idea of the method followed by a detailed method overview.

2.4.1 Idea

Consider a team of robots, each of them with a limited field of view, and a communication topology. A naive approach to solve Problem 1 could be that each robot computes a target formation and then all robots perform consensus on the formation parameters. Unfortunately, this can lead to a formation in collision with an obstacle, as shown in Fig. 2a. If

all the robots first agree on a convex obstacle-free region, and then compute a target formation therein, then this problem would not appear any more. An approach to compute this common obstacle-free region could be that each robot computes an obstacle-free region with respect to its limited field of view and then the robots collaboratively compute the intersection of all regions. Nonetheless, this could lead to an empty intersection as shown in Fig. 2b. This second problem can be solved by imposing (a) that the robots first agree on a common direction of motion and (b) that the convex obstacle-free region computed by each robot accounts for the robots' positions. The latter is equivalent to imposing that the convex obstacle-free region includes the convex hull of the robots' positions. See Fig. 2c for an example.

Following this line of thought, the proposed method consists of the following steps, which are detailed in Fig. 4.

2.4.2 Formation control

The team of robots computes a target formation and the assigned position of each robot with the following distributed procedure.

(a) Robots perform distributed consensus to compute the convex hull of the robots' positions.
(b) Robots perform a distributed min/max consensus to agree on the preferred direction of movement for the team.
(c) Each robot computes a large convex region in obstacle-free space, grown from the convex hull of the robots' positions and directed in the preferred direction of motion.
(d) Robots perform distributed consensus to compute the intersection of the individual convex regions.
(e) Each robot computes the optimal target formation within the resulting convex volume. At this stage all the robots execute the same optimization and with identical initial conditions, variables, cost function and constraints (these are computed from the intersection of convex regions, the convex hull of the robot positions and the preferred direction of motion). Therefore, we assume that they reach the same solution. If not, an additional consensus round would be required.
(f) Robots are assigned, with a distributed optimization, to target positions within the desired formation.

2.4.3 Low level control

Each robot navigates towards its assigned position within the target formation with a high-frequency control loop. They locally avoid collisions with their neighbors and remain within the convex region in free position-time space to avoid collisions with perceived static and moving obstacles. The target positions are updated as soon as a new configuration state of the team's formation is obtained.

3 Method

In this section we explain all the steps of the distributed navigation algorithm, discussing which information the robots need to communicate to their neighbors and which steps are executed locally. The proposed algorithm accounts for the limited visibility and communication capabilities of all the robots, exploiting the good properties of a distributed consensus scheme. To avoid confusions in the notation, throughout the section we denote discrete-time communication rounds using the index k and remove the continuous time dependency of the previous section.

We assume that the final time t_1 is longer than the amount of time required for the distributed algorithm to compute the formation and that the robots update their local information used in the consensus rounds only at the beginning of each execution of the algorithm. In addition, during this time for simplicity we consider that the communication network, $\mathcal{G}$, remains fixed. In a separate process, and at high frequency, each robot does update its information continuously for local collision avoidance and control towards the assigned position in the most recent target configuration for the team. See Fig. 4.

3.1 Convex hull of the robots' positions

In the first step the robots need to compute the convex hull, $\mathcal{C}$, of their positions. The convex hull of a set of points can be computed trivially via a function that we denote by *convhull*. To compute $\mathcal{C}$ in a distributed manner we let each robot handle

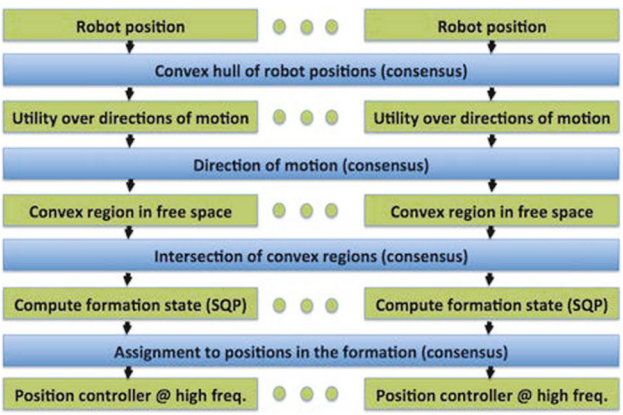


Fig. 4 Schema of the method which includes several consensus rounds to compute an obstacle-free convex region and the parameters of the formation. The position control includes local collision avoidance and is executed in a separate process with high update rate, controlling towards the newest formation

a local estimation of the convex hull, $\mathcal{C}_i$, which is initialized containing exclusively the robot's position, i.e., $\mathcal{C}_i(0) = \{\mathbf{p}_i\}$.

The robots execute an iterative process where, at each iteration, the local estimations are grown using the convex hull estimations obtained from direct neighbors in the communication graph. Then, the robots communicate to their neighbors only the new points that are part of their convex hull estimation, $\bar{\mathcal{C}}_i(k) = \mathcal{C}_i(k) \setminus \mathcal{C}_i(k-1)$. The whole process is repeated for a number of communication rounds equal to the diameter of $\mathcal{G}$, d. This method is synthesized in Algorithm 1.

Algorithm 1 Distributed Convex Hull - Robot i

1: $\mathcal{C}_i(-1) = \emptyset,\ \mathcal{C}_i(0) = \{\mathbf{p}_i\}$
2: **for** $k = 0 \dots d-1$ **do**
3: Send $\bar{\mathcal{C}}_i(k) = \mathcal{C}_i(k) \setminus \mathcal{C}_i(k-1)$ to all $j \in \mathcal{N}_i$
4: Receive $\bar{\mathcal{C}}_j(k)$ from all $j \in \mathcal{N}_i$
5: $\mathcal{C}_i(k+1)$ =convhull$(\mathcal{C}_i(k), \bar{\mathcal{C}}_j(k))$
6: **end for**

Proposition 1 *The execution of Algorithm 1 makes the local estimation of all the robots converge to the actual convex hull of the whole team in no more than d communication rounds. That is,*

$$\mathcal{C}_i(d) = \mathcal{C},\ \forall i \in \mathcal{I}. \tag{7}$$

Proof In order to show that (7) holds, we first show by induction that

$$\mathcal{C}_i(k+1) = \text{convhull}(\mathcal{C}_i(k), \mathcal{C}_j(k)), \tag{8}$$

for all $i \in \mathcal{I},\ j \in \mathcal{N}_i$ and $k \geq 0$.

Equation (8) holds for $k = 1$ because $\bar{\mathcal{C}}_i(0) = \mathcal{C}_i(0) = \{\mathbf{p}_i\}$ and, therefore, for all i, $\mathcal{C}_i(1)$ =convhull$(\mathcal{C}_i(0), \mathcal{C}_j(0))$. Assume now that Eq. (8) is also true up to some other $k > 0$. Thus,

$$\begin{aligned}\mathcal{C}_i(k+1) &= \text{convhull}(\mathcal{C}_i(k), \bar{\mathcal{C}}_j(k)) \\ &= \text{convhull}(\text{convhull}(\mathcal{C}_i(k-1), \mathcal{C}_j(k-1)), \\ &\quad \mathcal{C}_j(k) \setminus \mathcal{C}_j(k-1)) \\ &= \text{convhull}(\mathcal{C}_i(k), \mathcal{C}_j(k)),\end{aligned}$$

where in the last equality we have accounted that all the points that are not sent by robot j are already contained in the convex hull at the previous step of robot i.

Now let $\mathcal{N}_i(k), k \geq 0$, be the set of robots that are reachable from robot i after k propagation steps. That is, for $k = 1$, $\mathcal{N}_i(1) = \mathcal{N}_i$, whereas for $k = 2$, $\mathcal{N}_i(k)$ contains the neighbors of robot i and the neighbors of its neighbors. In a second step we show that

$$\mathcal{C}_i(k) = \text{convhull}(\mathbf{p}_i, \mathbf{p}_j),\ j \in \mathcal{N}_i(k), \tag{9}$$

for all $k \geq 0$. Clearly Eq. (9) is true for $k = 0$ and $k = 1$. Assume that it is also true for some other k. Using Eq. (8),

$$\begin{aligned}\mathcal{C}_i(k+1) &= \text{convhull}(\mathcal{C}_i(k), \mathcal{C}_j(k)), \\ &= \text{convhull}(\mathbf{p}_i, \mathbf{p}_j),\ j \in \mathcal{N}_i(k) \cup \mathcal{N}_j(k) \\ &= \text{convhull}(\mathbf{p}_i, \mathbf{p}_j),\ j \in \mathcal{N}_i(k+1).\end{aligned}$$

By induction, since the communication graph is assumed to be connected, $\mathcal{N}_i(d) = \mathcal{I}$ and Eq. (7) holds. □

We analyze now the communication cost of Algorithm 1. Note that, in the worst case, where the convex hull contains the positions of all the robots, our algorithm presents a communication cost equal to that of flooding all the positions to all the robots. Nevertheless, even in such case, there are practical advantages of using this procedure instead of pure flooding. Besides the likely savings in communications from positions that are not relayed because they do not belong to the convex hull, with our procedure there is no need for a specific identification of which position corresponds to a particular robot, making it better suited for pure broadcast implementations.

Remark 1 (Unknown d) If the diameter, d, is unknown, the consensus runs until convergence for all robots. Since only new points are transmitted at each iteration, the convergence of the algorithm can be detected using a timeout when no new messages are received.

Remark 2 (Algorithm complexity) In terms of computational demands, our algorithm requires the computation of d convex hulls for each robot, instead of a single computation. On the other hand, each convex hull computation will potentially contain fewer points, and the information from previous rounds could also be exploited for efficiency. Nevertheless, existing algorithms to compute the convex hull of a set of points are already fast enough not to consider the additional computations an issue of the algorithm.

3.2 Preferred direction of motion

The next step of the algorithm consists in computing the direction in which the team of robots needs to move. Denote by $\mathbf{g} \in \mathbb{R}^3$ the goal position for the robot formation and consider it known by all robots. A priori the preferred direction of motion, $\theta^* \in \mathbb{R}^3$, for the team of robots is given by the vector from $\mathbf{c} \in \mathbb{R}^3$, the centroid of $\mathcal{C}$, to the goal position, $\mathbf{g} \in \mathbb{R}^3$. Note that the centroid of the convex hull can be computed by all the robots locally without the need of additional information, as opposed to the centroid of the robots' positions, which would require further information and possibly

asymptotic consensus methods. However, it may happen that an obstacle is in the way of such direction, which might be seen only by a subset of the robots. Thus, we introduce an optional step in the algorithm in which the robots agree upon the best direction to compute the goal formation.

Our algorithm considers a discrete set, $\Theta = \{\theta_1, \ldots, \theta_\kappa\}$, containing κ different possible directions of motion. We assume that a common orientation frame is available to all the robots, with origin defined by the vector $\mathbf{g} - \mathbf{c}$. For each $\theta \in \Theta$, each robot computes a utility value, $u_i : \Theta \to \mathbb{R}^+$, that describes how good is that direction. The utility function can be defined for example, as the distance to an obstacle in that direction, based on the local perception of that robot.

Since different robots may have different utility values for the same angle, the global utility of the angle is given by the worst utility among all the robots, i.e.,

$$u(\theta) = \min_{i \in \mathcal{I}} u_i(\theta). \tag{10}$$

Therefore, the objective for the team is to find the angle that gives the best global utility,

$$\theta^* = \arg\max_{\theta \in \Theta} u(\theta) = \arg\max_{\theta \in \Theta} \min_{i \in \mathcal{I}} u_i(\theta). \tag{11}$$

In order to compute θ^* in a distributed fashion, each robot handles a vector $\mathbf{u}_i \in \mathbb{R}^\kappa$ to compute the global utility for all the angles in Θ. The vector is initialized as $\mathbf{u}_i(0) = [u_i(\theta_1), \ldots, u_i(\theta_\kappa)]$, i.e., initially each robot considers that its own utility is the global utility. Then, all the robots execute the following iterative rule,

$$\mathbf{u}_i(k+1) = \min_{j \in \mathcal{N}_i} (\mathbf{u}_i(k), \mathbf{u}_j(k)), \tag{12}$$

where the minimum is taken component wise in the vectors. This rule corresponds to κ distributed leader-election algorithms run in parallel (Lynch 1997), which is a well known algorithm that converges for all the robots to the minimum of the initial conditions in d iterations. Therefore, once the algorithm has converged, each robot knows the global utility for all the angles and can locally select the maximum over $\mathbf{u}_i(d)$, which will correspond to θ^*. In case of a tie between two directions, for simplicity we let the robots choose the one with minimum index in Θ. The procedure is summarized in Algorithm 2.

Remark 3 (Bandwidth reduction) Using the above rule, the total bandwidth used in the network will be equal to $\kappa n d$ units of information, obtained from d communication rounds, each one of them requiring n robots to transmit a vector of dimension κ. In order to reduce this quantity, at each communication round each robot only sends the components of the vector that have changed after executing (12). The messages contain then segments of the vector, determined by the initial component, the length of the segment and the data. While in the worst case the bandwidth usage of this methodology raises to $\frac{3}{2}\kappa n d$, we will show empirically that in practice this approach brings substantial savings.

Algorithm 2 Distributed Direction of Motion - Robot i

1: $\mathbf{u}_i(0) = [u_i(\theta_1), \ldots, u_i(\theta_\kappa)]$
2: **for** $k = 0 \ldots d-1$ **do**
3: Send $\mathbf{u}_i(k)$ to all $j \in \mathcal{N}_i$
4: Receive $\mathbf{u}_j(k)$ from all $j \in \mathcal{N}_i$
5: $\mathbf{u}_i(k+1) = \min_{j \in \mathcal{N}_i}(\mathbf{u}_i(k), \mathbf{u}_j(k))$ component-wise
6: **end for**
7: $\theta^* = \arg\max \mathbf{u}_i(d)$

3.3 Obstacle-free convex region

Recall that, from Sect. 3.1, all robots have knowledge of the convex hull $\mathcal{C}$ of the robots' positions and from Sect. 3.2 they share a preferred direction of motion. With this common information, but different obstacle maps due to the limited field of view, each robot computes an obstacle-free convex region embedded in position-time space, denoted $\mathcal{P}_i \subset \mathbb{R}^3 \times [0, \tau]$. If the step in Sect. 3.2 is omitted, the robots will use by default the angle θ^* defined by the vector $\mathbf{g} - \mathbf{c}$ as preferred direction.

To compute the convex regions $\mathcal{P}_i$ we follow our previous work for centralized formation control (Alonso-Mora et al. 2017), which relies on the iterative optimization by Deits and Tedrake (2014). Given a small initial ellipsoid in free space we compute (1) the separating hyperplanes between the ellipsoid and the obstacles and (2) the largest ellipsoid contained in the resulting convex polytope. These two steps are formulated as convex programs and are repeated iteratively until convergence to a large convex region in free space and as long as a set of points are contained in the convex region. The initial ellipsoid can be generated by two points biasing the growth of the convex polytope. We note though that the distributed formation control method described in this paper is agnostic to the underlaying algorithm to compute convex polytopes in free space.

For each robot i we can consider two polytopes, see Fig. 5, where each polytope is computed with the aforementioned procedure. These are:

- $\mathcal{P}_i^{\mathcal{C}}$, a convex polytope that contains the convex hull $\mathcal{C}$ of the robot positions, is computed towards a point $\chi = \mathbf{c} + \theta^* \tau$ in the preferred direction of motion, and is embedded in the free position-time space as seen by the robot. This polytope $\mathcal{P}_i^{\mathcal{C}} = \mathcal{P}_{\mathcal{C}\times 0}^{\chi \times \tau}(\bar{\mathcal{F}}_i(t_0))$ verifies that $\mathcal{C} \times 0 \subset \mathcal{P}_i^{\mathcal{C}} \subset \bar{\mathcal{F}}_i(t_0)$.

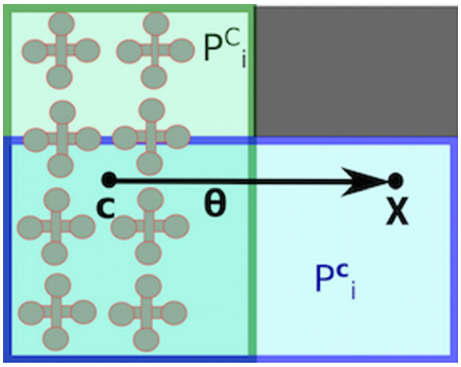


Fig. 5 Eight drones in formation and one static obstacle (black). The preferred direction of motion θ is shown with an arrow from the centroid $\mathbf{c}$ of the convex hull $\mathcal{C}$ of the robot positions. The convex region $\mathcal{P}_i^{\mathcal{C}}$, which contains all the robots, is shown on the left (green sides), and the convex region $\mathcal{P}_i^{\mathbf{c}}$, which contains the centroid $\mathbf{c}$, is shown on the bottom (blue sides) (Color figure online)

– $\mathcal{P}_i^{\mathbf{c}}$, a convex polytope that, in contrast to $\mathcal{P}_i^{\mathcal{C}}$, only contains the centroid $\mathbf{c}$ of the convex hull. This polytope $\mathcal{P}_i^{\mathbf{c}} = \mathcal{P}_{\mathbf{c}\times 0}^{\chi\times\tau}(\bar{\mathcal{F}}_i(t_0))$ verifies that $\mathbf{c}\times 0 \subset \mathcal{P}_i^{\mathbf{c}} \subset \bar{\mathcal{F}}_i(t_0)$.

We then define $\mathcal{P}_i$ as the intersection of both polytopes, i.e., $\mathcal{P}_i = \mathcal{P}_i^{\mathcal{C}} \bigcap \mathcal{P}_i^{\mathbf{c}}$. The former polytope (thanks to its convexity) guarantees that the robots can move towards the target configuration following collision-free trajectories. The latter polytope guides the team towards the goal.

However, due to the local visibility of the robots, some of these regions may intersect some obstacles that a particular robot has not seen. Additionally, these regions might not be equal for all robots, which, if used without further agreement, would lead to different target formations. Thus, the robots need to agree upon a common region that is globally free of obstacles. For that purpose, we next propose a distributed algorithm that computes the intersection of all the regions, $\mathcal{P} = \bigcap_{i\in\mathcal{I}} \mathcal{P}_i$.

As in Algorithm 1, each robot handles a local estimation of the region of interest. We denote $\mathcal{P}_i(k)$ the region of robot i at iteration k. This region is initialized with the value provided by the local optimizer, $\mathcal{P}_i(0) = \mathcal{P}_i$. At each iteration the individual regions are shrunk by computing local intersections with the regions received from the neighbors in the communication graph. The algorithm finishes after d iterations, as shown in Algorithm 3.

Algorithm 3 Distributed Obstacle-Free Region - Robot i

1: $\mathcal{P}_i(0) = \mathcal{P}_i$
2: **for** $k = 0 \ldots d-1$ **do**
3: Send $\mathcal{P}_i(k)$ to all $j \in \mathcal{N}_i$
4: Receive $\mathcal{P}_j(k)$ from all $j \in \mathcal{N}_i$
5: $\mathcal{P}_i(k+1) = \mathcal{P}_i(k) \cap \mathcal{P}_j(k)$
6: **end for**

Proposition 2 *When executing Algorithm 3 the regions of all the robots converge to a common region, equal to the intersection of the initial regions, in no more than d communication rounds. That is,*

$$\mathcal{P}_i(d) = \mathcal{P} = \bigcap_{j\in\mathcal{I}} \mathcal{P}_j(0), \quad \forall i \in \mathcal{I}. \tag{13}$$

Proof Similarly to the proof of Proposition 1, we let $\mathcal{N}_i(k)$, $k \geq 0$, be the set of robots that are reachable from robot i after k propagation steps. We show by induction that

$$\mathcal{P}_i(k) = \bigcap_{j\in\mathcal{N}_i(k)} \mathcal{P}_j, \tag{14}$$

for all $k \geq 0$. Clearly Eq. (14) is true for $k = 0$ and $k = 1$. Assuming that it is also true for some k, using the associative and distributive properties of the intersection with respect to the intersection it is straightforward to show that it also holds for $k+1$. Therefore, by the connectedness of $\mathcal{G}$, Eq. (13) holds for $k = d$. □

To compute the intersections, we rely on a representation of the obstacle-free convex polytope $\mathcal{P}$ given by its equivalent set of linear constraints

$$\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^4 | A\mathbf{x} \leq \mathbf{b}, \quad \text{for } A \in \mathbb{R}^{n_l\times 4}, \quad \mathbf{b} \in \mathbb{R}^{n_l}\}, \tag{15}$$

where n_l denotes the number of faces of $\mathcal{P}$. This leads to messages of size equal to $n_l \times 4$.

Let us note that any face that belongs to both $\mathcal{P}_i(k)$ and $\mathcal{P}_i(k+1)$ will yield the same linear constraints in both polytopes. This implies that, similarly to Algorithm 1, robots do not need to send all the constraints at each communication round, but only those that are new, and consequently more restrictive than in the previous round. In particular, robots send at each communication round the new linear constraints that have appeared after computing the intersection in Line 5 of Algorithm 3, instead of all the linear constraints at each round, as we originally considered in Alonso-Mora et al. (2016). This modification leads to substantial communication savings in the Algorithm, specially when compared to a pure flooding approach. In addition, it allows us to define a solid stop criterion for the algorithm in case of unknown value of d, as in the case of Remark 1.

Proposition 3 *The resulting convex region $\mathcal{P}$ is a convex polytope.*

Proof The intersection of convex regions is also convex. □

Proposition 4 *The resulting convex region $\mathcal{P}$ does not intersect with any obstacle seen by the robots in the team for the time period $[t_0, t_1]$, i.e., it is fully contained in the free position-time space.*

Proof For each robot i, its individual convex region $\mathcal{P}_i(0)$ is fully contained in its observed free position-time space by construction, i.e., $\mathcal{P}_i(0) \subset \bar{\mathcal{F}}_i(t_0)$.

In each consensus round, the new polytope is given by the intersection of the previous polytope with the received ones, therefore $\mathcal{P}_i(k+1) \subset \mathcal{P}_i(k)$. This implies that, after convergence, $\mathcal{P} = \mathcal{P}_i(d) \subset \mathcal{P}_i(0)$.

From these two set inclusions we then have that $\mathcal{P} \subset \mathcal{P}_i(0) \subset \bar{\mathcal{F}}_i(t_0)$ for all robot i. Therefore, the following holds true

$$\mathcal{P} \subset \bigcap_{i\in\mathcal{I}} \bar{\mathcal{F}}_i(t_0) = \mathbb{R}^3 \times [0,\tau] \setminus \bigcup_{i\in\mathcal{I}} \hat{\mathcal{O}}_i(t_0). \tag{16}$$

Which guarantees that the convex region $\mathcal{P}$ does not intersect any obstacle within the time horizon. □

If $\mathcal{P} = \emptyset$, an alternative convex region $\mathcal{P}_i$ can be selected by each robot as described by Alonso-Mora et al. (2017)—Sect. III-C, and consensus on the intersection is repeated.

3.4 Optimal formation

Given the convex set $\mathcal{P}$, and recalling Sect. 2.2, each robot i then computes the configuration state $\mathbf{z}^*$ of a locally optimal formation for the team. For a template formation $f \in \mathcal{I}_f$ the optimal configuration state $\mathbf{z}_f^*$ is found by solving the non-linear optimization

$$\begin{aligned} \mathbf{z}_f^* = \arg\min_{\mathbf{z}} \quad & J_f(\mathbf{z}) \\ s.t. \quad & \mathcal{V}(\mathbf{z}, f) \times t_1 \subset \mathcal{P} \qquad \text{(collision-free)} \\ & s \geq 2\tfrac{\max(r,h)}{d_f} \qquad \text{(minimum size)} \end{aligned} \tag{17}$$

where $J_f(\mathbf{z})$ is a cost function penalizing the weighted deviation to the goal $\mathbf{g}$, to a preferred size $\bar{s}$ and to a preferred orientation $\bar{\mathbf{q}}$. The first constraints impose that all vertices are within the convex region $\mathcal{P}$. The second constraint imposes that no two robots within the formation are in collision.

Recalling that $\mathbf{z} = [\mathbf{t}, s, \mathbf{q}]$, we employ the cost function

$$\begin{aligned} J_f(\mathbf{z}) = {} & w_t ||\mathbf{t} - \mathbf{g}(t_1)||^2 + w_s ||s - \bar{s}||^2 \\ & + w_q ||\mathbf{q} - \bar{\mathbf{q}}||^2 + c_f, \end{aligned} \tag{18}$$

where w_t, w_s, w_q are design weights, and c_f is the predefined cost for formation type $f \in \mathcal{I}_f$.

This constrained optimization was first introduced by Alonso-Mora et al. (2017) and can be solved with state of the art Sequential Convex Programming solvers. We employ the non-linear solver SNOPT by Gill et al. (2002). If multiple template formations exist, the best one f^* is obtained by solving m constrained optimizations. Thanks to the previous consensus rounds, all robots execute this optimization with the same parameters (the template formations are known by all robots and the convex region $\mathcal{P}$ was agreed in the consensus round). Therefore, even when the optimization is solved individually by each robot, we assume that they all obtain the same values for the target formation.

3.5 Robot assignment to positions in the formation

The result of the computation of Sect. 3.4 is a target formation f^* and configuration state $\mathbf{z}^*$, from which its associated set of target robot positions $\{\mathbf{r}_1{}^*, \ldots, \mathbf{r}_n{}^*\}$ can be computed from Eq. (4).

Robots are assigned to the goal positions with the objective of minimizing the sum of squared travelled distances, i.e., find the permutation matrix, $\mathbf{X} : \mathcal{I} \to \mathcal{I}$, minimizer of

$$\min_{\mathbf{X}} \quad \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}} x_{ij} ||\mathbf{p}_i - \mathbf{r}_j^*||^2. \tag{19}$$

There exist several distributed algorithms based on local interactions that are able to find the optimal solution to the above linear program, such as the distributed simplex proposed by Burger et al. (2012). The algorithm has a bounded communication cost per iteration and proven finite-time termination.

Proposition 5 *The robots can transition to their assigned positions in the target formation with collision-free paths.*

Proof Under the assumption of holonomic motion model, the proposition is guaranteed if, for every robot, the straight line from the current position to the assigned position is collision free within the position-time space.

Recall Sect. 3.3 and let us denote by $\mathcal{P}^{\mathcal{C}}$ the intersection $\mathcal{P}^{\mathcal{C}} = \bigcap_{i\in\mathcal{I}} \mathcal{P}_i^{\mathcal{C}}$, which contains the convex hull of robot positions, i.e., $\mathcal{C} \subset \mathcal{P}^{\mathcal{C}}$, and the consensus polytope $\mathcal{P}$, i.e, $\mathcal{P} \subset \mathcal{P}^{\mathcal{C}}$, by construction. It also does not intersect any of the seen obstacles, i.e., $\mathcal{P}^{\mathcal{C}} \subset \bigcap_{i\in\mathcal{I}} \bar{\mathcal{F}}_i(t_0)$.

From Sect. 3.3, we have that the current robot position $\mathbf{p}_i$ is inside the convex region $\mathcal{P}^{\mathcal{C}}$, since $\mathbf{p}_i \times 0 \in \mathcal{C} \times 0 \subset \mathcal{P}^{\mathcal{C}}$. Furthermore, the optimization problem of Eq. (17) guarantees that the target position $\mathbf{r}^*_{\sigma(i)}$ is within the same convex region, since $\mathbf{r}^*_{\sigma(i)} \times \tau \in \mathcal{P} \subset \mathcal{P}^{\mathcal{C}}$. Therefore, the path from the current position to the target position is within a convex polytope $\mathcal{P}^{\mathcal{C}}$ which does not intersect any of the seen obstacles. □

3.6 Real-time control

Consider $\mathbf{r}_i^*$ to be the target position assigned to robot i, which is updated as soon as a new target formation is computed. In a high frequency control loop each robot individually navigates towards its target position avoiding collisions with static obstacles, moving obstacles and other robots locally. For this we compute a collision-free local motion via a state of the art receding horizon controller which accounts for the

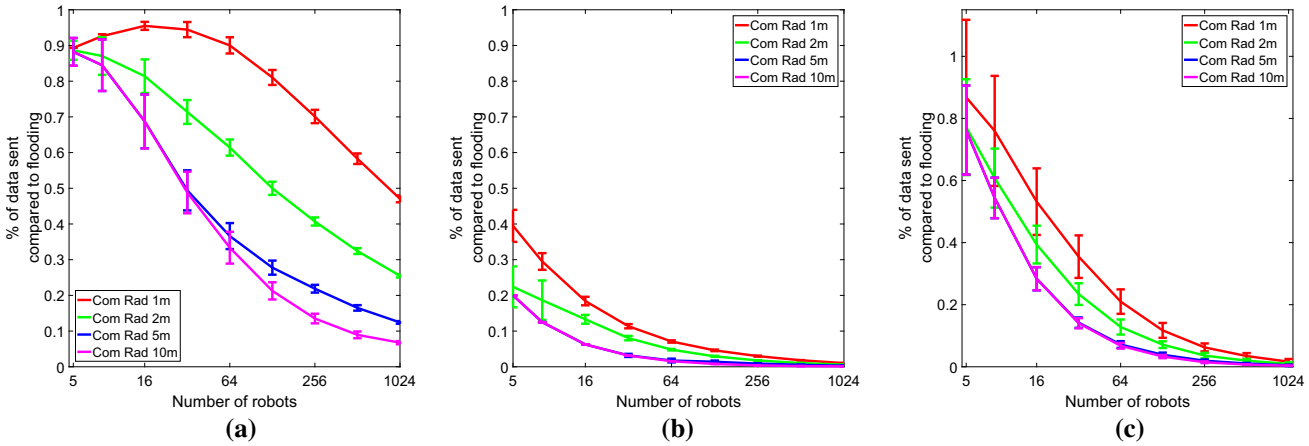


Fig. 6 Communication cost of Algorithm 1 (Distributed Convex Hull) (**a**), Algorithm 2 (Distributed Direction of Motion) (**b**) and Algorithm 3 (Distributed Obstacle-Free Region) (**c**) relative to flooding. The plots show the mean and standard deviation over 100 trials for different numbers of robots and communication radii. In the majority of cases the three algorithms require less bandwidth than pure flooding in the same number of communication rounds and present a good scalability with the number of robots

dynamical model of the robot. Two suitable methods are the distributed reciprocal velocity obstacles with motion constraints for aerial vehicles by Alonso-Mora et al. (2015b) and the receding horizon controller by Nägeli et al. (2017a).

4 Simulation results

4.1 Performance of the consensus strategies

In this section we present simulation results using Monte Carlo experiments to analyze the distributed Algorithms 1, 2 and 3. In particular, we are interested in comparing the communication demands of our algorithms with a solution consisting on flooding the information of all the robots to the whole network, i.e., a centralized solution under the assumption of limited communication. Since the final solution and the number of communication rounds are equivalent to those of the centralized solution, we do not analyze these parameters in the simulation.

4.1.1 Convex hull

In Algorithm 1 we considered different group sizes, from $n = 5$ to $n = 1024$ robots. For each value of n we have considered 100 different initial conditions, where the robots have been randomly placed in a 3 dimensional space, with minimum inter-robot distance equal to 0.5 m, forcing the connectedness of the communication graph for a communication radius of one meter. Then, for each configuration we have considered four different communication radii, $CR = \{1, 2, 5, 10\}$ and we have run the algorithm.

The amount of information exchanged over the network, relative to the amount required when using flooding, is shown in Fig. 6a. The plot shows the mean and standard deviation over the 100 trials for each scenario. First of all, it should be noted that the total bandwidth requirements over the network will actually increment as we increase the number of robots, because the number of communication rounds and possibly the size of the convex hull will increase accordingly. Thus, the objective of the plots is not to analyze the total bandwidth but to compare how much better (or worse) is one solution relative to the other, understanding that there might be other limitations for the algorithms depending on the size of the network and its configuration. With this in mind, the first observation is that in all the cases our algorithm requires less communication than pure flooding of all the positions because the relative cost is always less than one. The algorithm also shows the scalability with the number of robots. As n increases, the amount of positions that do not belong to the convex hull is also increased, resulting in fewer information exchanges for any communication radius. In a similar fashion, by increasing the communication radius, the relative communication cost is also decreased. This happens because at each communication round, the robots are able to discard more points from their local convex hull estimations, since they have information from more neighbors available. Overall, taking into account that the number of communication rounds of our algorithm is the same as the one for flooding, we conclude that our distributed solution is always a better choice.

4.1.2 Direction of movement

We have also analyzed Algorithm 2 (Distributed Direction of Motion) using the same simulation parameters, i.e., 100 trials for each different number of robots and communication

radii. We have measured the relative cost to a pure flooding algorithm for vectors with $\kappa = 100$ utilities, using the implementation described in Remark 3. The results of this experiment are depicted in Fig. 6b, where we can observe that for this particular algorithm our solution is by large more efficient than flooding. As in the convex hull, the algorithm also performs better for densely connected networks and large values of n. We have also analyzed the influence of the size of κ, observing that the relative cost, compared to flooding, was basically the same for the different sizes. Therefore, since this parameter can be chosen arbitrarily, the design choice should be made according to the capacity of the network to find a good balance between the degree of accuracy in the orientation of the direction and the absolute bandwidth requirements.

4.1.3 Intersection of convex regions

Finally, in order to analyze Algorithm 3 we have considered again the same number of robots and communication radii, as well as 100 random initial configurations for each pair of values. The initial regions $\mathcal{P}_i$ have been created using the following procedure: first we have created a random polytope composed by 20 three dimensional vertices. Then, for each robot we have randomly changed 5% of the vertices and included perturbations on another 15% of the vertices. These parameters have been designed taking into account the properties of the polytopes obtained in the full simulations containing real obstacles described in Sect. 4.2. The results of these experiments are depicted in Fig. 6c.

The plot shows a similar behavior to the one in Fig. 6a, with decreasing bandwidth requirements, relative to a flooding procedure, as n and the communication radius are increased. Only for small teams of robots, some executions of the algorithm will require the exchange of more information than flooding. This happens because for the flooding we send the 3-dimensional vertices of the associated obstacle-free polytope instead of the 4-dimensional constraints to reduce the bandwidth. When n is small, the savings from sending the new constraints at each iteration are not enough to compensate the increase in the dimension of the information exchanged, given that $d \simeq n$. Nevertheless, in the rest of cases our algorithm outperforms this algorithm to levels where we only require to send a small fraction of the information. Considering the extra routing control mechanisms that flooding would require make our solution a much better choice. Besides, the solution sending only the new constraints improves over our original approach in Alonso-Mora et al. (2016), where on average for small teams the cost of our algorithm was much bigger.

In summary, our algorithms require in (almost) all cases less bandwidth than equivalent flooding approaches using the same number of communication rounds. However, it should be noted that the number of communication rounds of these algorithms will increase with the diameter of the network, which will also grow with the number of robots. Nevertheless, even if the number of rounds increases with n, the size of the messages will remain more or less constant for arbitrarily large numbers of robots. The reason for this is that, while the number of robots can grow, the number of points that define the convex hull (or similarly the direction of motion and the points in the obstacle free region) will remain approximately constant.

4.2 Multi-drone formation control

We present simulations with teams of quadrotor MAVs, where we employ the same nonlinear dynamical model and LQR controller employed by Alonso-Mora et al. (2015b), which was verified with real quadrotors. We use SNOPT by Gill et al. (2002) to solve the non-linear program, a goal-directed version of IRIS by Deits and Tedrake (2014) to compute large convex regions and the Drake toolbox from MIT[1] to handle quaternions, constraints and interface with SNOPT.

In our simulations a time horizon $\tau = 4$ s is considered for the experiments with 4 robots and of $\tau = 10$ s for the experiments with 16 robots, due to the large size of the formation and the scenario. In all cases a new formation is computed every 2 s. The individual collision avoidance planners run at 5 Hz and the quadrotors have a preferred speed of 1.5 m/s. Both the visibility distance and the communication radius are set to 3 m and sensing and actuation noise are neglected.

We test the distributed algorithm described in this paper in two scenarios previously introduced in Alonso-Mora et al. (2017) for the centralized case. This provides a direct comparison and evaluation.

4.2.1 Four robots

Figure 7 shows snapshots and trajectories of four quadrotors tracking a circular trajectory while locally avoiding three static obstacles and a dynamic obstacle. Three default formations are considered: square (1st preference), diamond (2nd preference) and line. The optimal parameters are computed with the distributed consensus algorithm and non-linear optimizaiton, allowing rotation in 3D (flat horizontal orientation preferred) and reconfiguration.

The four quadrotors start from the horizontal square and slightly tilt it (11 s) to avoid the incoming dynamic obstacle. To fully clear it while avoiding the obstacle in the lower corner, they shortly switch to a vertical line, and then back to the preferred square formation (20 s). To pass through the next narrow opening they switch back to the line formation (30

[1] http://drake.mit.edu.

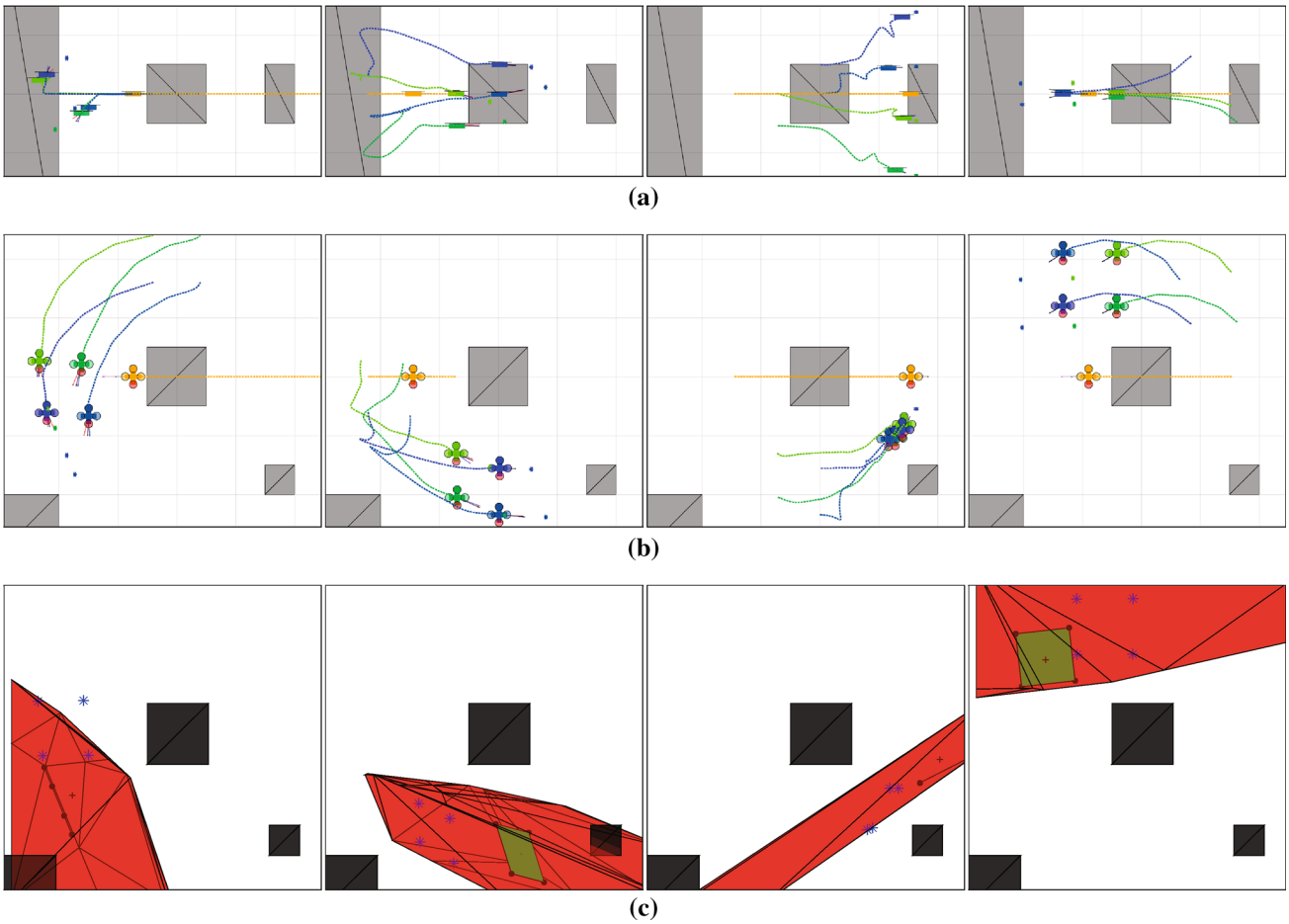


Fig. 7 Four quadrotors (green-blue) navigate in a $12 \times 12 \times 6$ m^3 scenario with three static obstacles (grey) and a dynamic obstacle (yellow). The four quadrotors track a circular motion around the central obstacle and locally reconfigure the formation to avoid collisions and make progress. **a** Side view. From left to right, snapshots at 11, 20, 30 and 45 s, and paths of the robots in-between. **b** Top view. From left to right, snapshots at 11, 20, 30 and 45 s, and paths of the robots in-between. **c** Projection (red) of the convex obstacle-free polytope $\mathcal{P}$ onto the 2D top view at approximately the same time instances of the snapshots. The projection can overlap with obstacles that are not in the field of view of the robots. The projection of the optimized formation is shown in green and the convex hull $\mathcal{C}$ of the robot positions at the time of computation with blue stars (recall that the optimized formation is only recomputed every 2 s). In two frames the projection of the optimized formation has negligible size and is shown by a line (Color figure online)

s). Once the obstacles are cleared they return to the preferred horizontal square formation (45 s).

4.2.2 Sixteen robots

Figure 8 shows the paths of 16 quadrotors moving along a corridor of three different widths. Three default formations are considered: $4 \times 4 \times 1$ defined by four vertices (preferred), $4 \times 2 \times 2$ defined by eight vertices and $8 \times 2 \times 1$ defined by four vertices. At each time step the method computes the optimal parameters for each of the three and selects the one with lowest cost upon them. Between times 75 and 110 s the method successfully rotates the formation by 90° to avoid collisions (the default formations were horizontal, which is also preferred in the cost function).

5 Experimental results

In this section we describe experimental results with a team of four quadrotors.

5.1 Implementation details

Our experiments are conducted with two standard laptops (Quadcore Intel i7 CPU@2.8 GHz). The person and drones are tracked with a external motion capture system that provides precise position information at a high update rate and move in an environment of approximately 4m (W) $\times$ 5m (L) $\times$ 3m (H). In order to guarantee connectedness of the communication graph, the communication radius of the drones is simulated at 3 m. The visibility has also been set at 3m, looking for a compromise between safety and allowing different perceptions of the environment. The physical diam-

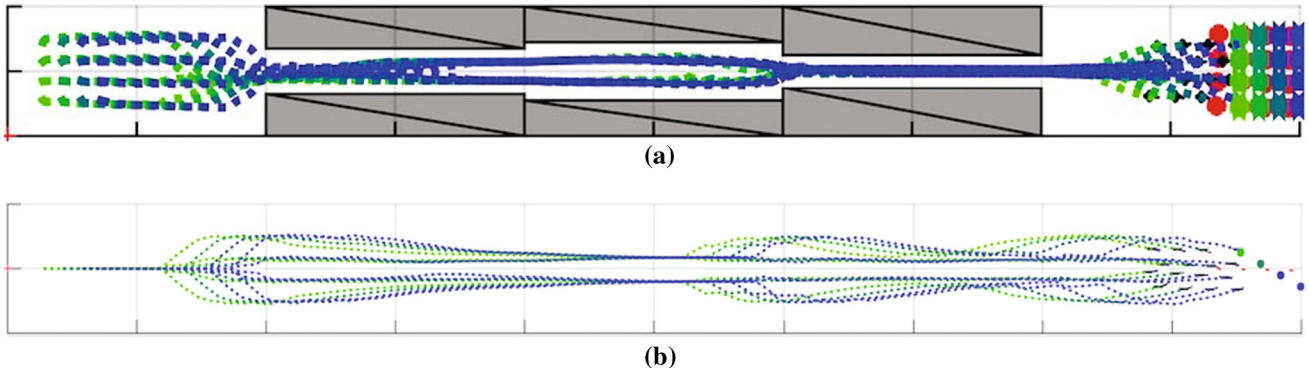


Fig. 8 Sixteen quadrotors navigate along a $100 \times 10 \times 10$ m^3 corridor, with obstacles shown in grey (top view). The quadrotors locally adapt the formation to remain collision free. The robots start in the preferred horizontal $4 \times 4 \times 1$ formation and tilt it to vertical, to pass trough the narrow corridors. In the wider middle region they transform to a $4 \times 2 \times 2$ formation, which has lower cost than the vertical $4 \times 4 \times 1$. They finally transition towards $4 \times 4 \times 1$. **a** Top view (X–Y) with robot paths. Sixteen simulated quadrotors move from left to right. **b** Side view (X–Z) with robot paths. Sixteen simulated quadrotors move from left to right

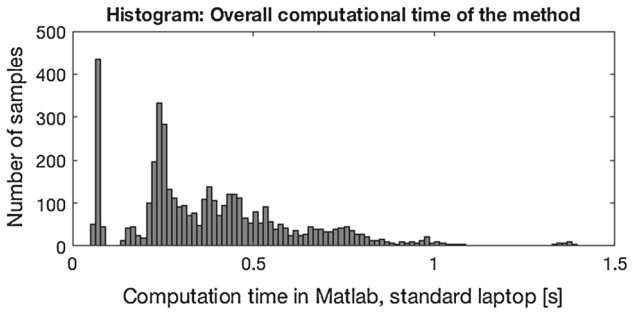


Fig. 9 Histogram of computation time of the distributed formation control approach, for a sequential implementation in Matlab, with all the data from the drone experiments

eter of each drone is, approximately, 0.3 m. When computing the configuration of the formation, we impose a minimum distance of 1 m between drones, to avoid collisions and aerodynamic disturbances.

In one laptop we receive the current state of the drones and obstacles and execute the method of this paper. In particular, we compute convex obstacle-free regions, perform the consensus rounds, optimize the configuration of the formation and compute the goal position for each robot in that formation. These computations are performed in a continuous manner, as soon as one execution is finished, we recompute. The computations are performed in Matlab and the communication is handled with ROS. In practice, we observe some variability in the computation time, with the median at 0.35 s. See Fig. 9 for a histogram of the computation time over all experiments. We observe that many instances took in the order of 0.25 s, with a large number of them below 0.5 s, and most of them below 1 s. Very few instances took longer than 1 s. These are due to longer computational time in the optimization and search for convex regions, for which we did not set strict real-time bounds. Furthermore, our current implementation is sequential, with several for-loops over all robots, and thus a parallelized implementation may reduce the computation time.

In the second laptop, we receive the current state of the drones and obstacles at a high frequency and send input commands to the drones. We also receive the target positions for the drones, computed in the first laptop. This laptop controls the position of the drones at a high, and approximately constant, frequency of 20 Hz. We implement a slightly modified version of the Model Predictive Controller introduced by Nägeli et al. (2017a) and extended to multiple drones by Nägeli et al. (2017b), where we remove all the cost terms for videography. This controller minimizes the deviation from the assigned position of the drone in the formation, subject to collision avoidance, state and input constraints. We run a controller for each drone and exchange the planned trajectories sequentially. We employ a horizon of $M = 20$ steps, at 0.05 s each, and we solve the MPC problem with FORCES Pro (Domahidi et al. 2012; Domahidi and Jerez 2016), which generates fast solver code, exploiting the special structure in the NLP problem. MPC methods have also been employed with onboard sensing of obstacles, for example by Odelga et al. (2016).

5.2 Quadrotor hardware

We use unmodified Parrot Bebop2 quadrotors in all our experiments. The communication between the drones and the host PC is handled via ROS (Quigley et al. 2009) and we directly send the control inputs from the first time step, *without* an additional feedback controller for trajectory tracking on the drone.

The state of the quadrotor is given by its position $\mathbf{p} \in \mathbb{R}^3$, its velocity $\dot{\mathbf{p}} = [\dot{\mathbf{p}}_{x,y}, \dot{p}_{q_z}] \in \mathbb{R}^3$ and its orientation, i.e. roll Φ_q, pitch Θ_q and yaw ψ_q.

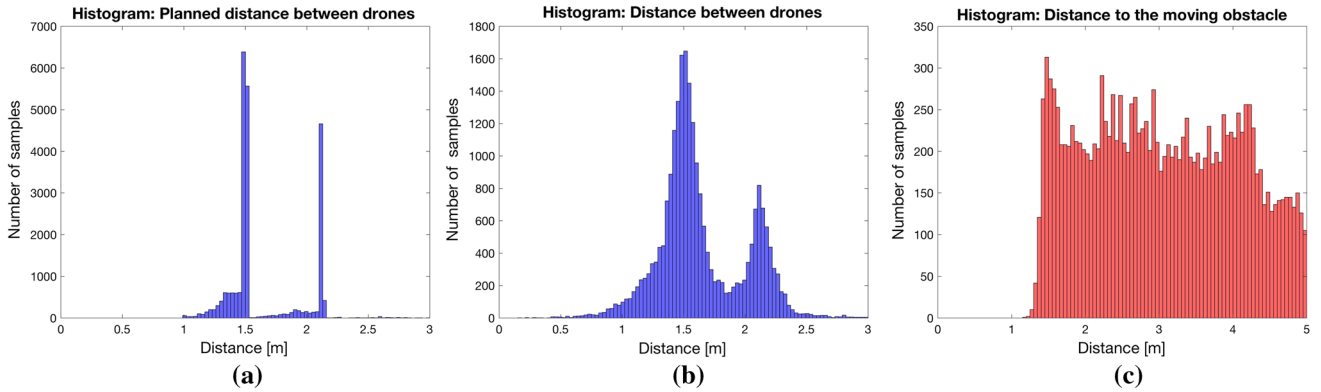


Fig. 10 Histograms with robot–robot and robot–human distance from cumulated data of all 20 experiments. Although the planned formations maintained the desired separation of one meter between drones, in a few instances the drones got closer due to their dynamics and uncertainties, see Sect. 5.3 for a discussion. The robots always maintained a separation of over one meter with the moving human. **a** Planned inter-drone distance. **b** Measured inter-drone distance. **c** Distance to moving obstacle

The control inputs to the system are: the velocity of the quadrotor in the body-z axis $v_z = \dot{p}_{q_z}$, the desired roll angle ϕ_q and the desired pitch angle θ_q for the quadrotor and its angular speed around the body-z axis ω_{q_z}. The horizontal velocities are not directly controlled.

The NMPC by Nägeli et al. (2017b) accounts for the internal constraints of the Parrot Bebop 2 (e.g., maximal vertical and horizontal velocities, maximal roll and pitch angles). The limits are described in the documentation of the Parrot SDK.[2]

5.3 Results

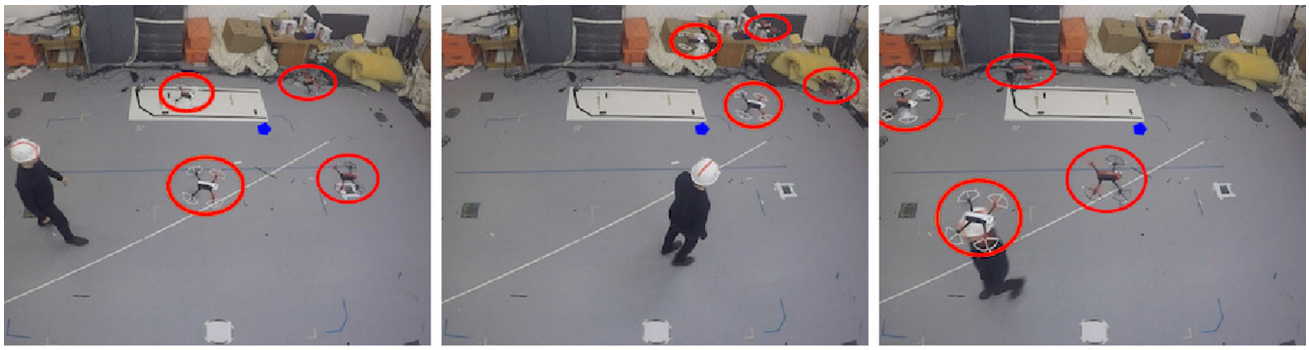
We have performed a total of twenty experiments, all of them with one moving obstacle (walking and running) and three to four drones. In each experiment the robots are tasked with either maintaining the centroid of the formation as close as possible to the center of the room or with tracking a circular, constant velocity motion. For the experiments with three drones we consider both a line and a triangle formation. For the experiments with four drones we consider both a line and a square formation. In all cases, the preferred size of the formation is set to 1.5 m between consecutive robots (side of the triangle and square), and its minimum size is set to 1 m between robots.

In Fig. 10 we show three histograms with cumulated data over all twenty experiments, for a total of about half an hour of flight time. In Fig. 10a we cumulate the distance between the planned position of all drones in the formation, i.e. their goal positions as computed by the method of this paper. We observe two distinct peaks, one at 1.5 m, the preferred inter-robot distance, and one at 2.1 m, the diagonal in the square formation. The variability in distance is due to the method adjusting the size of the formation to avoid collisions, while maintaining a minimum distance of 1 m between robots.

In Fig. 10b we cumulate the distance between all drones during the experiments. Again, we observe the two distinct peaks at the planned 1.5 and 2.1 m. Yet, we also observe a much larger variability in the inter-robot distance. In most instances, the drones maintain a separation greater than 1 m, as planned by the formation control module. In a few instances, two drones were between the planned separation of 1 m and the collision distance of 0.5 m. In extremely few instances, the separation between two drones was below 0.5 m and a collision could occur, but did not happen in our experiments. These instances occurred in cases where the human is running, the drones are pushed towards the walls of the room and have to fly over in a narrow space and short period of time in order to avoid a collision. Recalling our system architecture, see Sect. 5.1, we note that the set-point tracking is the responsibility of the low-level collision avoidance, which in our implementation was a sequential non-linear model predictive controller (NMPC). The reduction of inter-drone distance, below the predefined setpoint, was due to several factors: (a) the drone dynamics were not perfectly identified in the model employed, (b) delays in the control and communication framework were not modeled, (c) higher weight was given to drone–human collision avoidance than to drone–drone collision avoidance, and (d) slack variables were used in the NMPC optimization framework. We recall that the contribution of this paper is the formation control method—which generated collision-free setpoints for the drones—and not the low level single-drone controller.

Finally, in Fig. 10c we cumulate the distance between each drone and the moving human. In all instances a minimum separation of 1 m was achieved and therefore collision with the human were avoided. The approach successfully adapted the configuration of the formation to avoid the moving human, whose motion ranged from walking to running.

[2] http://developer.parrot.com/.



(a) (b) (c)

Fig. 11 Four drones navigate in a square planar formation (drones are circled in red). The formation is allowed to change its position and rotate around the vertical axis to avoid the moving person and the walls of the room. The team of robots minimizes the deviation between its centroid and the center of the room (blue hexagon). We show a slightly tilted top view. **a** Preferred formation at goal position. **b** Avoidance to the side. **c** Avoidance over the person (Color figure online)

We now present experimental results for four distinct scenarios. In each scenario we describe a distinct capability of the method. A video illustrating the results accompanies this paper and can be found at https://youtu.be/khzM54Qk1QQ.

5.3.1 Single planar formation and static setpoint

In this first scenario four drones navigate in a square planar formation. The team of robots is allowed to change the position of the centroid of the formation and the orientation of the formation around the vertical axis, in order to avoid the moving person and the walls of the room. The team of robots minimizes the deviation between its centroid and the center of the room. We show three representative frames in Fig. 11. The left one shows the team of robots at their preferred position and orientation (squared formation in the center of the room). In the middle image the team of robots moves to the side to let the person pass. In the right image the team of robots flies up to let the person pass below. The approach showed good performance.

5.3.2 Single planar formation and moving setpoint

In this second scenario four drones navigate in the square planar formation of the previous experiment. As before, the formation is allowed to change its position and rotate around the vertical axis to avoid the moving person and the walls of the room. In this scenario, the centroid of the team of robots tracks a circular trajectory of radius 2 m and speed 0.3 m/s while avoiding the moving person. In four representative frames, see Fig. 12, we show a full avoidance maneuver where the team of robots lifts to avoid the person and then continues tracking the specified trajectory.

5.3.3 Single formation (with free 3D rotation) and static setpoint

In this third scenario four drones navigate again in the square formation, which can now rotate freely in 3D to avoid the moving person and the walls of the room. The team of robots minimizes the deviation between its centroid and the center of the room and tilts the formation to avoid the human. In Fig. 13 we show the robots at their preferred position and orientation and three examples of the team of robots tilting the formation to avoid the moving person. In these set of experiments we observed that while the method successfully computes tilted configurations for the team of robots, which are safe, their execution was not always robust due to the turbulences created by the drones and their height sensors (sonar), which produced interferences between the drones when they were very near.

5.3.4 Multiple formations and moving setpoint

In the fourth scenario three drones navigate in a triangle formation and can reconfigure to a line formation if advantageous. The team of robots optimizes (a) the formation type (triangle -preferred- or line), (b) the centroid of the formation and (c) the orientation around the vertical axis. This is done to avoid the moving person and the walls of the room and to let the centroid of the team of robots track a circular trajectory of radius 2 m and speed 0.5 m/s. In Fig. 14 we show two sequences of three images each. The three drones are first shown in their preferred formation type (triangle) tracking the circular motion (Fig. 14a). When the person traps them against a side, they have to switch to a line formation (Fig. 14b), which then goes up to fly over the person (Fig. 14c), before returning the preferred triangle formation once the area is clear (Fig. 14d). If the person runs towards the drones,

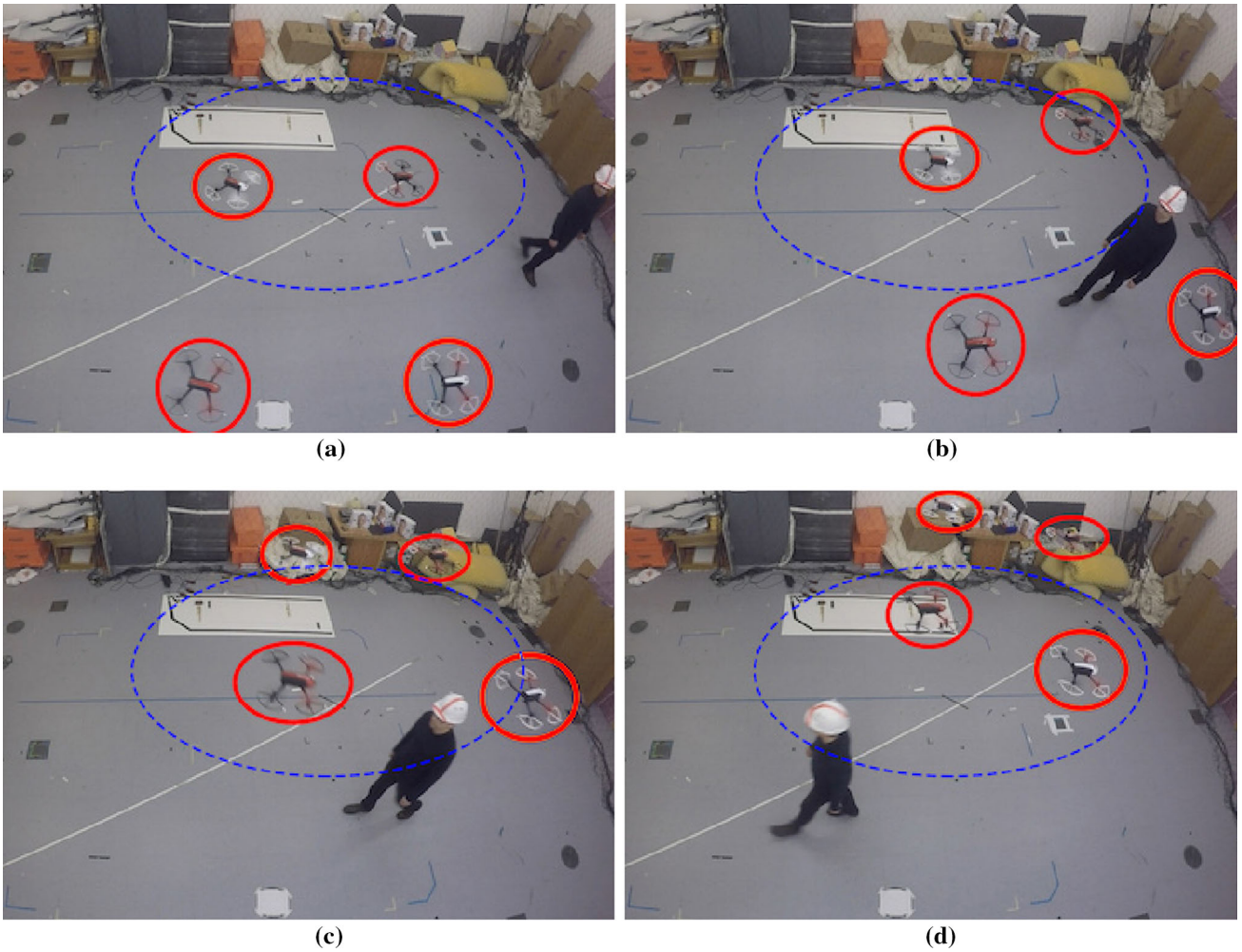


Fig. 12 Four drones navigate in a square planar formation (drones are circled in red). The formation is allowed to change its position and rotate around the vertical axis to avoid the moving person and the walls of the room. The centroid of the team of robots tracks a circular trajectory (blue dashed line) of radius 2 m. We show a slightly tilted top view. **a** The team moves right towards the person. **b** The team slightly rotates in the plane to avoid the person. **c** The team backs-up and flies over the person. **d** The team continues tracking the circular motion (Color figure online)

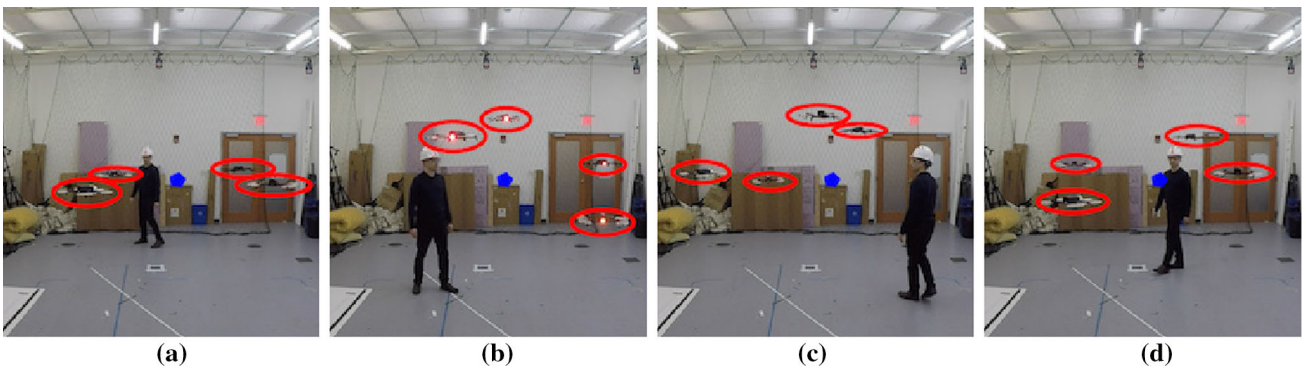


Fig. 13 Four drones navigate in a square formation. The team of robots is allowed to change its position and 3D orientation freely to avoid the moving person and the walls of the room. The team of robots minimizes the deviation between its centroid and the center of the room (blue hexagon) and tilts the formation to avoid the human. We show a side view. **a** Preferred formation at goal position. **b** The team tilts the formation to avoid the person. **c** The team tilts the formation to avoid the person. **d** The team tilts the formation to avoid the person (Color figure online)

Fig. 14 Three drones navigate in a triangle formation and can reconfigure to a line formation if advantageous (drones are circled in red). The team of robots optimizes the **a** the formation type (triangle- preferred- or line), **b** the centroid of the formation and **c** the orientation around the vertical axis. This is done to avoid the moving person and the walls of the room and to let the centroid of the team of robots track a circular trajectory (blue dashed line) of radius 2 m and speed 0.5 m/s. We show a slightly tilted top view. **a** Preferred triangle formation. **b** Switches to line to avoid person. **c** Team flies over the person. **d** The team returns to the triangle formation. **e** The team flies up to avoid the person. **f** The team flies up to avoid the person (Color figure online)

and they have enough room, they may quickly go up to pass over the person (Fig. 14e) and continue tracking the circular trajectory (Fig. 14f).

5.3.5 Discussion on limitations

In multiple experiments we have observed the following.

The approach is safe under predictable movement of the human. If the human walks in the environment or runs with constant speed, the formation control method updates the parameters of the formation to successfully avoid the moving person. If the human makes abrupt changes in speed while running, then the formation control method can result in an unfeasible optimization due to the constant velocity assumption and the computation delay. If this happens, the individual low level controllers, based on NMPC, avoid collisions with the moving person and the formation is recovered as soon as the optimization becomes feasible again, typically in below a second.

Although the median computation time of the approach was 0.35 s, several instances took over one second to compute, see Fig. 9 and Sect. 5.1 for a discussion. This delay was noticeable at high obstacle speeds. A faster and bounded update rate is desirable for more fluid performance and shorter reaction times.

Higher robustness is achieved when in the cost function of the optimization, Eq. (18), we set a lower weight for vertical deviations from the setpoint than for deviations in the horizontal plane. In this way we give preference to avoiding the human by lifting the formation, rather than a sideways avoidance. This helps in avoiding situations where the robots are trapped against a wall and they have to quickly fly up to avoid the collision with the moving human.

The formation control method takes polytopes as obstacles. The volume occupied by the human was enclosed by a convex polytope. We chose an hexagonal prism with the sides slightly tilted, i.e. the upper face was slightly smaller than the lower face. This serves two purposes: (a) it provides larger clearance around the body of the person and the legs and (b) it helps in biasing the free-space convex polytope to have more clearance as height increases.

Overall, the approach performed very well and was able to safely adapt the configuration of the formation in real time.

6 Conclusion

In this paper we considered a team of networked robots in which each robot only communicates with its neighbors. We showed that navigation of distributed teams of robots in formation among static and dynamic obstacles can be achieved via a constrained non-linear optimization combined with consensus. The robots first compute an obstacle-free convex region and then optimize the formation parameters. In particular, non-convex environments can be handled. Thanks to the consensus on convex obstacle-free regions, the robots do not need to exchange the position of all the obstacles. Instead they compute, and exchange, the joint free space. This approach may present lower computational cost, specially in scenarios with many obstacles, and requires substantially fewer communication messages than flooding for consensus.

In simulations with up to sixteen drones, and in experiments with up to four drones, we showed successful navigation in formation. The robots were able to reconfigure the formation when required, in order to avoid collisions with static and moving obstacles, and to make progress. Last, but not least, the approach is general and could be adapted to other formation definitions and applications, such as collaborative transportation with mobile manipulations, as long as the formation can be defined by a set of equations that determine the outer vertices and position of the robots.

Since the approach is local, deadlocks may still occur. Yet, the consensus round on the best direction of motion could be extended to account for global planning performed by the individual robots. Another avenue of future work is splitting and merging into smaller and larger subteams, which also navigate in formation. Future works should also look at the integration of planning and sensing in real environments and at joint low-level and high-level planning.

References

Alonso-Mora, J., Knepper, R. A., Siegwart, R., & Rus, D. (2015a). Local motion planning for collaborative multi-robot manipulation of deformable objects. In: *IEEE international conference on robotics and automation*.

Alonso-Mora, J., Montijano, E., Schwager, M., & Rus, D. (2016). Distributed multi-robot navigation in formation among obstacles: A geometric and optimization approach with consensus. In *IEEE international conference on robotics and automation*.

Alonso-Mora, J., Baker, S., & Rus, D. (2017). Multi-robot formation control and object transport in dynamic environments via constrained optimization. *The International Journal of Robotics Research*, *36*(9), 1000–1021.

Alonso-Mora, J., Nägeli, T., Siegwart, R., & Beardsley, P. (2015b). Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots*, *39*(1), 101–121.

Augugliaro, F., Schoellig, A. P., & D'Andrea, R. (2012). Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In: *IEEE/RSJ international conference on intelligent robots and systems*.

Ayanian, N., & Kumar, V. (2010a). Abstractions and controllers for groups of robots in environments with obstacles. In: *IEEE inter-*

national conference on robotics and automation, Anchorage, AK (pp. 3537–3542).
Ayanian, N., & Kumar, V. (2010b). Decentralized feedback controllers for multi-agent teams in environments with obstacles. *IEEE Transactions on Robotics*, *26*(5), 878–887.
Balch, T., & Hybinette, M. (2000). Social potentials for scalable multi-robot formations. In: *IEEE international conference on robotics and automation*.
Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE Transaction on Robotics and Automation*, *14*(6), 926–939.
Burger, M., Notarstefano, G., Allgower, F., & Bullo, F. (2012). A distributed simplex algorithm for degenerate linear programs and multi-agent assignments. *Automatica*, *48*(9), 2298–2304.
Chen, Y., Cutler, M., & How, J. P. (2015). Decoupled multiagent path planning via incremental sequential convex programming. In *IEEE international conference on robotics and automation* (ICRA).
Deits, R., & Tedrake, R. (2014). Computing large convex regions of obstacle-free space through semidefinite programming. In *Workshop on the algorithmic fundamentals of robotics*.
Derenick, J., Spletzer, J., & Kumar, V.(2010). A semidefinite programming framework for controlling multi-robot systems in dynamic environments. In: *IEEE conference on decision and control*.
Derenick, J. C., & Spletzer, J. R. (2007). Convex optimization strategies for coordinating large-scale robot formations. *IEEE Transaction on Robotics*, *23*, 1252–1259.
Desai, J. P., Ostrowski, J. P., & Kumar, V. (2001). Modeling and control of formations of nonholonomic mobile robots. *IEEE Transaction on Robotics and Automation*, *17*(6), 905–908.
Domahidi, A., & Jerez, J. (2016). FORCES Pro: Code generation for embedded optimization. https://www.embotech.com/FORCES-Pro.
Domahidi, A., Zgraggen, A. U., Zeilinger, M. N., Morari, M., & Jones, C. N. (2012). Efficient interior point methods for multistage problems arising in receding horizon control. In: *47th IEEE conference on decision and control, 2008. CDC 2008* (pp. 668–674). IEEE.
Dong, X., Yu, B., Shi, Z., & Zhong, Y. (2015). Time-varying formation control for unmanned aerial vehicles: Theories and applications. *IEEE Transactions on Control Systems Technology*, *23*(1), 340–348.
Erdmann, M., & Lozano-Perez, T. (1987). On multiple moving objects. *Algorithmica*, *2*, 477–521.
Franchi, A., Masone, C., Grabe, V., Ryll, M., Bulfhoff, H. H., & Giordano, P. R. (2012). Modeling and control of UAV bearing formations with bilateral high-level steering. *International Journal of Robotics Research*, *31*, 1504–1525.
Gill, P. E., Murray, W., & Saunders, M. A. (2002). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, *12*(4), 979–1006.
Hatanaka, T., Igarashi, Y., Fujita, M., & Spong, M. W. (2012). Passivity-based pose synchronization in three dimensions. *IEEE Transactions on Automatic Control*, *57*(2), 360–375.
Keviczky, T., Borrelli, F., Fregene, K., Godbole, D., & Balas, G. J. (2008). Decentralized receding horizon control and coordination of autonomous vehicle formations. *IEEE Transactions on Control Systems Technology*, *16*(1), 19–33.
Kia, S. S., Cortes, J., & Martinez, S. (2016). Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication. *Automatica*, *55*(5), 254–264.
Kuriki, Y., & Namerikawa, T. (2015). Formation control with collision avoidance for a multi-UAV system using decentralized mpc and consensus-based control. *SICE Journal of Control, Measurement, and System Integration*, *8*(4), 285–294.
Kushleyev, A., Mellinger, D., Powers, C., & Kumar, V. (2013). Towards a swarm of agile micro quadrotors. *Autonomous Robots*, *35*(4), 287–300.
Lin, Z., Francis, B., & Maggiore, M. (2005). Necessary and sufficient graphical conditions for formation control of unicycles. *IEEE Transaction on Automatic Control*, *50*(1), 540–545.
Lynch, N. (1997). *Distributed algorithms*. Burlington: Morgan Kaufmann publishers.
Michael, N., Zavlanos, M. M., Kumar, V., & Pappas, G. J. (2008). Distributed multi-robot task assignment and formation control. In: *IEEE international conference on robotics and automation*.
Montijano, E., & Mosteo, A. R. (2014). Efficient multi-robot formations using distributed optimization. In: *IEEE 53th conference on decision and control*.
Montijano, E., Cristofalo, E., Zhou, D., Schwager, M., & Sagues, C. (2016). Vision-based distributed formation control without an external positioning system. *IEEE Transactions on Robotics*, *32*(2), 339351.
Morgan, D., Subramanian, G. P., Chung, S.-J., & Hadaegh, F. Y. (2016). Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research*, *35*(10), 1261–1285.
Mostagh, N., Michael, N., Jadbabaie, A., & Daniilidis, K. (2009). Vision-based, distributed control laws for motion coordination of nonholonomic robots. *IEEE Transactions on Robotics*, *25*(4), 851860.
Mosteo, A. R., Montano, L., & Lagoudakis, M. G. (2008). Guaranteed-performance multi-robot routing under limited communication range. In: *Distributed autonomous robotic systems* (pp. 491–502).
Nägeli, T., Meier, L., Domahidi, A., Alonso-Mora, J., & Hilliges, O. (2017b). Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)*, *36*(4), 132.
Nägeli, T., Alonso-Mora, J., Domahidi, A., Rus, D., & Hilliges, O. (2017a). Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters*, *2*(3), 1696–1703.
Nestmeyer, T., Robuffo Giordano, P., Blthoff, H. H., & Franchi, A. (2017). Decentralized simultaneous multi-target exploration using a connected network of multiple robots. *Autonomous Robots*, *41*(4), 989–1011.
Odelga, M., Stegagno, P., & Bülthoff, HH. (2016). Obstacle detection, tracking and avoidance for a teleoperated UAV. In: *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 2984–2990). IEEE.
Oh, K.-K., & Ahn, H.-S. (2011). Formation control of mobile agents based on inter-agent distance dynamics. *Automatica*, *47*(10), 2306–2312.
Oh, K. K., Park, M. C., & Ahn, H. S. (2015). A survey of multi-agent formation control. *Automatica*, *53*(3), 424–440.
Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). Ros: An open-source robot operating system. In: *IEEE ICRA workshop on open source software*.
Ren, W., & Beard, R. W. (2008). *Distributed consensus in multi-vehicle cooperative control. Communications and control engineering*. London: Springer.
Sabattini, L., Secchi, C., & Fantuzzi, C. (2011). Arbitrarily shaped formations of mobile robots: Artificial potential fields and coordinate transformation. *Autonomous Robots*, *30*, 385–397.
Saha, I., Ramaithitima, R., Kumar, V., Pappas, G. J., & Seshia, S. A. (2014). Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In: *IEEE/RSJ international conference on intelligent robots and systems*.
Schoch, M., Alonso-Mora, J., Siegwart, R., & Beardsley, P. (2014). Viewpoint and trajectory optimization for animation display with aerial vehicles. In: *2010 IEEE international conference on robotics and automation (ICRA)* (pp. 4711–4716). IEEE.

Schwager, M., Julian, B. J., Angermann, M., & Rus, D. (2011). Eyes in the sky: Decentralized control for the deployment of robotic camera networks. *Proceedings of the IEEE*, *99*(9), 1541–1561.

Suzuki, T., Sekine, T., Fujii, T., Asama, H., & Endo, I. (2000). Cooperative formation among multiple mobile robot teleoperation in inspection task. In *Proceedings of the 39th IEEE Conference on Decision and Control, 2000* (Vol. 1, pp. 358–363). IEEE.

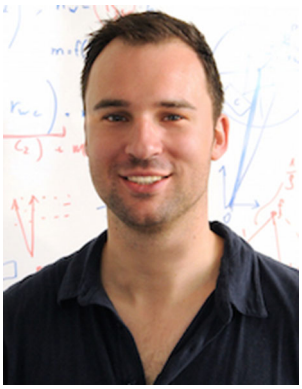
Turpin, M., Mohta, K., Michael, N., & Kumar, V. (2014). Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots*, *37*(4), 401–415.

Urcola, P., Lazaro, M. T., Castellanos, J. A., & Montano, L. (2017). Cooperative minimum expected length planning for robot formations in stochastic maps. *Robotics and Autonomous Systems*, *87*, 3850.

Wurman, P. R., D'Andrea, R., & Mountz, M. (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, *29*(1), 9.

Javier Alonso-Mora is an Assistant Professor at the Delft University of Technology. Until October 2016 he was a Postdoctoral Associate at the Computer Science and Artificial Intelligence Lab CSAIL of MIT, working in the Distributed Robotics Lab. He received his Ph.D. degree in robotics from ETH Zurich, working in the Autonomous Systems Lab (2014). He holds a M.Sc. from ETH Zurich, a Diploma in Engineering and a Diploma in Mathematics from the Technical University of Catalonia. He was also a member of Disney Research Zurich. His main research interests are in navigation, decision-making, motion planning and control of autonomous mobile robots, with a special emphasis in multi-robot systems and robots that interact with other robots and humans. He received a NWO Veni award in 2017.

Eduardo Montijano is an assistant professor at Universidad de Zaragoza, Spain. He received the M.Sc. and Ph.D. degrees from the Universidad de Zaragoza, Spain, in 2008 and 2012 respectively, obtaining the extraordinary award of the Universidad de Zaragoza in the 2012–2013 academic year. He has been a visiting scholar at University of California San Diego, University of California Berkeley and Boston University in the United States and at Royal Institute of Technology, in Stockholm, Sweden. He has also been a faculty member at Centro Universitario de la Defensa, Zaragoza, Spain, between 2012 and 2016. His main research interests include distributed algorithms, cooperative control and computer vision.



Tobias Nägeli is currently a Ph.D. student in the Advanced Interactive Technologies group of the Institute of Pervasive Computing at the Swiss Federal Institute of Technology Zürich (ETH). His Ph.D. advisor is Prof. Dr. Otmar Hilliges. In 2013, he received his MSc from ETH Zurich in Electrical Engineering with a main focus on Control and Estimation.

Otmar Hilliges is originally from Munich, Germany where he was born and raised. Currently he is an Assistant Professor of Computer Science (tenure-track) at ETH Zurich. He leads the AIT lab affiliated with the Institute of Pervasive Computing and the Institute of Visual Computing. Prior to joining ETH he was Researcher at Microsoft Research Cambridge, in the I3D group (2010–2013). He was awarded a Diplom (equiv. MSc) in Computer Science from Technische Universität München, Germany (Summa Cum Laude 2004) and a PhD in Computer Science from LMU München, Germany (Summa Cum Laude 2009). Following his studies, he spent two years as a postdoc at Microsoft Research Cambridge (2010–2012).

Mac Schwager is an assistant professor with the Aeronautics and Astronautics Department at Stanford University. He obtained his BS degree in 2000 from Stanford University, his MS degree from MIT in 2005, and his PhD degree from MIT in 2009. He was a postdoctoral researcher working jointly in the GRASP lab at the University of Pennsylvania and CSAIL at MIT from 2010 to 2012, and was an assistant professor at Boston University from 2012 to 2015. His research interests are in distributed algorithms for control, perception, and learning in groups of robots and animals. He received the NSF CAREER award in 2014.

Daniela Rus is the Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science and Director of the Computer Science and Artificial Intelligence Laboratory (CSAIL) at MIT. Rus's research interests are in robotics, mobile computing, and big data. The key focus of her research is to develop the science of networked/distributed /collaborative robotics, by asking: how can many machines collaborate to achieve a common goal? Rus is a Class of 2002 MacArthur Fellow, a fellow of ACM, AAAI and IEEE, and a member of the National Academy of Engineering. She earned her PhD in Computer Science from Cornell University. Prior to joining MIT, Rus was a professor in the Computer Science Department at Dartmouth College.