Distributed Multi-Robot Formation Control among Obstacles: A Geometric and Optimization Approach with Consensus

Javier Alonso-Mora*, Eduardo Montijano†, Mac Schwager‡ and Daniela Rus*

Abstract— **This paper presents a *distributed* method for navigating a team of robots in formation in 2D and 3D environments with static and dynamic obstacles. The robots are assumed to have a reduced communication and visibility radius and share information with their neighbors. Via distributed consensus the robots compute (a) the convex hull of the robot positions and (b) the largest convex region within free space. The robots then compute, via sequential convex programming, the locally optimal parameters for the formation within this convex neighborhood of the robots. Reconfiguration is allowed, when required, by considering a set of target formations. The robots navigate towards the target collision-free formation with individual local planners that account for their dynamics. The approach is efficient and scalable with the number of robots and performs well in simulations with up to sixteen quadrotors.**

I. Introduction

Muti-robot teams can be employed for various tasks, such as surveillance, inspection, or automated factories. In these scenarios, robots may be required to navigate in formation, for example for maintaining a communication network, for collaboratively handling an object, for surveilling an area or to improve navigation.

Multi-robot navigation in formation has received extensive attention in the past, with many works considering obstacle-free scenarios. In [1] we leveraged efficient optimization techniques, namely quadratic programming, semi-definite programming and (non-linear) sequential quadratic programming to devise a *centralized* method for local navigation in formation. These techniques provided good computational efficiency, local guarantees and generality, and were employed at different stages of the method for local motion planning in formation among static and dynamic obstacles, albeit centralized.

In this work we combine the optimization concepts presented in [1] with distributed consensus and geometric reasoning to achieve similar results in a distributed schema, where robots are no more centrally controlled, but instead have a limited field of view, and communicate with their immediate neighbors.

* The authors are at CSAIL-MIT, 32 Vassar St, 02139 Cambridge MA, USA {jalonsom,rus}@csail.mit.edu

† The author is with Centro Universitario de la Defensa, Zaragoza, Spain, emonti@unizar.es

‡ The author is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA, schwager@stanford.edu

*This work was supported in part by pDOT ONR N00014-12-1-1000, ARL Grant W911NF-08-2-0004, the Boeing Company, the MIT-Singapore Alliance on Research and Technology under the Future of Urban Mobility and by Spanish projects DPI2012-32100, DPI2015-69376-R, CUD2013-05 and grant CAS14/00205. We are grateful for their support.

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. A simplified global planner, only waypoints for the formation center are required, can use this method to navigate the group of robots from an initial location to a final location. But a human may also provide the global path for the formation, or a desired velocity, and the robots will adapt their configuration automatically.

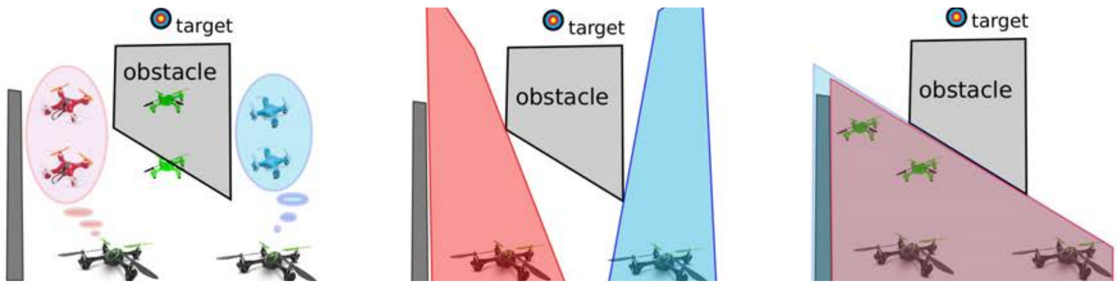
A. Related works

Extensive work exists for real-time navigation of multiple robots in formation. These techniques include using a set of reactive behaviors [2], potential fields [3], navigation functions [4] and decentralized feedback laws with graph theory [5]. These have been mostly shown in 2D environments and may require extensive tuning for the particular formation and environment. In contrast, our method automatically optimizes for the formation parameters natively in a 2D and 3D dynamic environment. On the down side, our method does not directly model the agent dynamics in the optimization, although it includes them in the individual local planners.

Distributed formation control solutions are also abundant in the literature [6]. The restriction of having a central planner can be easily lifted using nearest neighbors information [7]. These type of solutions usually rely on consensus-type controllers, such as [8], [9], or optimization methods [10], and assume the environment is obstacle-free. Compared to them, we exploit consensus algorithms to obtain all the necessary information to compute the optimal formation in the presence of obstacles.

Convex optimization frameworks for navigating in formation include semidefinite programming [11] which considers only 2D circular obstacles, distributed quadratic optimization [12] without global coordination and limited adaptation of the formation, and second order cone programming [13] which triangulates the free 2D space to compute the optimal motion in formation. In contrast, we propose a more general optimization plus consensus based approach.

Offline centralized non-convex optimizations include a mixed integer approach [14] and a discretized linear temporal logic approach [15]. They provide global guarantees but scale poorly with the number of robots. The second one is further limited in the definition of the formation. In contrast, we aim at on-line computation, albeit local, by solving a non-linear program via sequential convex programming. This technique



(a) Independent target formations (b) Independent obstacle-free regions (c) Our approach

Fig. 1. Example of three approaches for distributed formation planning with obstacles as discussed in Sec. II. (a) Each robot independently computes a target formation (red/blue). Consensus on the formation's parameters (green) would lead to a formation in collision with the obstacle. (b) Each robot computes an obstacle-free region, but their intersection is empty. (c) Our approach, see Sec. II, with target formation, after consensus, in green. Note that, in this example, the left most obstacle is not within the field of view of the robot in the right (blue region), but seen by the one in the left (red region).

has been employed [16] to compute collision-free trajectories for multiple UAVs, but without considering formations.

B. Contribution

The main contribution of this paper is a *distributed* method for navigation of a team of robots while reconfiguring their formation to avoid collisions. The method applies to robots navigating in 2D and 3D workspace among static and moving obstacles.

As part of our holistic method, we present distributed consensus methods to compute (a) the convex hull of the robot's positions and (b) the intersection of convex regions. We further rely on convex and non-convex optimization techniques first introduced in [1].

We provide a formal analysis with convergence guarantees of the distributed algorithms composing the holistic approach and simulations with teams of robots.

C. Organization

Sec. II provides a high level overview of the whole approach presented in the paper. Sec. III describes the notation and the centralized solution to the problem. Sec. IV describes the main algorithm and Sec. V discusses experimental results. Finally, the conclusions of the paper are in Sec. VI.

II. Algorithm overview

Consider a team of robots, each with a limited field of view, and a communication topology.

A naive approach 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 Figure 1(a). This problem can be solved if all the robots compute a new formation in a common obstacle-free region which is convex.

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 Figure 1(b).

This second problem can be solved by imposing that the convex obstacle-free region computed by each robot accounts for the robots' positions, which is equivalent to it accounting for the convex hull of the robots' positions. See Figure 1(c) for an example.

Following this line of thought, the proposed method consists of the following steps.

1) Distributed computation of target formation:

a) Robots perform distributed consensus to compute the convex hull of the robots' positions.
b) Each robot computes the largest convex region in obstacle-free space, grown from the convex hull of the robots' positions and which is directed in the preferred direction of motion.
c) Robots perform distributed consensus to compute the intersection of the individual convex regions.
d) Each robot computes the optimal target formation within the resulting convex volume.
e) Robots are assigned, with a distributed optimization, to target positions within the target formation.

2) Collision-free motion towards the target formation: Robots, at a higher update rate, navigate towards their assigned goals within the target formation. They locally avoid collisions with their neighbors.

III. Preliminaries

A. Definitions

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 $\mathbf{p}_i(t) \in \mathbb{R}^3$. 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., $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$. We assume 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. In the following we consider all robots to have the same shape (cylinders). But the method is not strictly limited to this case.

2) 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 potentially known by all robots. This can be given by a human operator or a standard sampling based approach, 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. The *distributed* local planner presented in this work computes a 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. Denote the current time by t_o and $t_f = t_o + \tau$.

3) Static obstacles and field of view: For each robot i, its field of view, typically a sphere of given radius centered at the robot's position, is denoted by $\mathcal{B}_i \subset \mathbb{R}^3$. 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. 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.

4) Moving obstacles: Moving obstacles within the field of view of robot i can be accounted for. Consider $j \in \mathcal{J}_i = \{1, \dots, n_{DO,i}\} \subset \mathbb{N}$ the list of observed moving obstacles of shape $\mathcal{D}_j \subset \mathbb{R}^3$. We denote by $\mathcal{D}_j(t)$ the volume occupied by the dynamic obstacle j at time t and $\bar{\mathcal{D}}_j(t)$ its dilation by half of robot i's volume. For predicted future positions we employ the constant velocity assumption.

5) Position-time workspace: For robot i and current time t_o denote the union of static and dynamic obstacles seen by robot i by

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

The position-time workspace for the robot is then

$$\bar{\mathcal{W}}_i(t_o) = \mathbb{R}^3 \times [0, \tau] \setminus \hat{\mathcal{O}}_i(t_o) \subset \mathbb{R}^4. \tag{1}$$

B. Formation definition

We consider a pre-defined set of $f \in \mathbb{N}$ default formations, such as square, line or T and known by all robots in the team. Denote by $\mathcal{F}_0^i$, $1 \le i \le f$, one such default formation. Formation $\mathcal{F}_0^i$ is given by a set of robot positions $\{\mathbf{r}_{0,1}^i, \dots, \mathbf{r}_{0,n}^i\}$ and a set of vertices $\{\mathbf{f}_{0,1}^i, \dots, \mathbf{f}_{0,n_i}^i\}$ relative to the center of rotation (typically the centroid) of the formation. 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. Denote by d_0^i the minimum distance between any given pair of robots in the default formation $\mathcal{F}_0^i$. See Fig. 2 for an example.

A formation is then defined by an isomorphic transformation, which includes an expansion $s \in \mathbb{R}_+$, a translation $\mathbf{t} \in \mathbb{R}^3$ and a rotation represented by a unit quaternion $\mathbf{q} \in SO(3)$, its conjugate denoted by $\bar{\mathbf{q}}$. The vector of optimization variables is denoted by $\mathbf{x} = [\mathbf{t}, s, \mathbf{q}] \in \mathbb{R}^8$ and the vertices and robot positions of the resulting formation

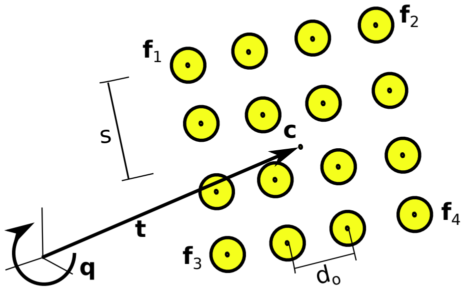


Fig. 2. Example of a square formation with sixteen disk robots and transformed by $\mathbf{x} = [\mathbf{t}, s, \mathbf{q}]$. The convex hull is given by vertices $\mathbf{f}_j$.

$\mathcal{F}^i(\mathbf{x})$ are given by

$$\begin{aligned} \mathbf{r}_j^i &= \mathbf{t} + s\,\mathrm{rot}(\mathbf{q}, \mathbf{r}_{0,j}^i), \quad \forall j \in [1, n], \\ \mathbf{f}_j^i &= \mathbf{t} + s\,\mathrm{rot}(\mathbf{q}, \mathbf{f}_{0,j}^i), \quad \forall j \in [1, n_i], \end{aligned} \tag{2}$$

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

$$\begin{bmatrix} 0 & \mathrm{rot}(\mathbf{q}, \mathbf{x}) \end{bmatrix}^T = \mathbf{q} \times \begin{bmatrix} 0 & \mathbf{x} \end{bmatrix}^T \times \bar{\mathbf{q}}. \tag{3}$$

In the exposition of the method we rely on this definition for the formation, but the method is more general and can be applied to alternative definitions, such as a team of mobile manipulators carrying a rigid object. For that description of the formation, refer to Sec. V of the centralized method [1].

C. Centralized formation planning

The original algorithm for *centralized* local formation planning [1] consists of the following steps.

First, compute the largest convex polytope $\mathcal{P}$ in free space, grown from the current robot positions, $\mathbf{p}_i(t_o) \in \mathcal{P}$, $\forall i \in \mathcal{I}$, and that is directed towards the goal $\mathbf{g}(t_f)$.

Second, compute the optimal formation $\mathcal{F}(t_f)$ contained within $\mathcal{P}$ and minimizing the distance between the formation's centroid and the goal $\mathbf{g}(t_f)$. The parameters of the formation are optimized subject to a set of constraints via a centralized sequential convex optimization. In this computation the robot's dynamics are ignored.

Third, in a faster loop, the robots are optimally assigned to target positions of the formation $\mathcal{F}(t_f)$ and move towards them employing a low level local planner [17] that generates collision-free inputs that respect the robot's dynamics.

In the case that no feasible formation exists, the robots navigate independently towards the goal.

We extend this to *distributed* navigation in formation.

IV. Distributed Algorithm

In this section we present the distributed algorithm to compute the obstacle-free target formation. The algorithm accounts for the limited visibility and communication capabilities of all the robots by iterative message exchange using a consensus-type 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_f is longer than the amount of time required for the distributed algorithm to compute the formation.

A. *Convex hull of the robots' positions*

The first step the robots need to address to compute the target formation is the computation of the convex hull, $\mathcal{C}$, of their positions. While this is a trivial problem in centralized scenarios, the same is not true in the current context of limited communications because each robot only has access to partial information. To overcome this limitation we propose a distributed algorithm that allows all the robots to obtain the convex hull of their positions using only local interactions. In the algorithm we assume that there is a function, *convhull*, that computes the convex hull spanned by a given set of points and that there are no pose variations of the robots during its execution. Under these assumptions, we let each robot handle a local estimation of the convex hull, $\mathcal{C}_i$, that is initialized containing exclusively the robot's position, i.e., $\mathcal{C}_i(0) = \mathbf{p}_i$.

After that, the robots execute an iterative process where at each iteration the local estimations are grown using the convex hull estimations obtained by 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 $k = 0 \ldots d-1$ 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,

$$\mathcal{C}_i(d) = \mathcal{C},\ \forall i \in \mathcal{I}. \tag{4}$$

Proof: In order to show that (4) holds, we first show by induction that

$$\mathcal{C}_i(k+1) = \text{convhull}(\mathcal{C}_i(k), \mathcal{C}_j(k)), \tag{5}$$

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

Equation (5) 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. (5) is also true up to some other $k > 0$. Thus,

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

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

$$\mathcal{C}_i(k) = \text{convhull}(\mathbf{p}_i, \mathbf{p}_j), j \in \mathcal{N}_i(k), \tag{6}$$

for all $k \geq 0$. Clearly Eq. (6) is true for $k = 0$ and $k = 1$. Assume that it is also true for some other k. Using (5),

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

By induction, since the communication graph is assumed to be connected, $\mathcal{N}_i(d) = \mathcal{I}$ and (4) holds. ■

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 rather than 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.

B. *Obstacle-free convex region*

Denote by $\mathbf{g} \in \mathbb{R}^3$ the goal position for the robot formation and consider it known by all robots. Recall that, from the previous step, all robots have knowledge of the convex hull $\mathcal{C}$ of the robots' positions. With this common information, but different obstacle map due to the limited field of view, each robot computes an obstacle-free convex region embedded in position - time space, denoted $\mathcal{P}_i \in \mathbb{R}^3 \times [0, \tau]$.

Analogously to the derivations in [1], $\mathcal{P}_i$ is given by the intersection of two convex polytopes, both directed towards the goal $\mathbf{g}$, the first one containing $\mathcal{C}$ and the second one containing only the centroid of $\mathcal{C}$, denoted by $\mathbf{c} \in \mathbb{R}^3$. Following the notation therein, the convex polytope is then given by $\mathcal{P}_i = \mathcal{P}_{f_o \to g|i} \cap \mathcal{P}_{o \to g|i} = \mathcal{P}^{[\mathbf{g},\tau]}_{\mathcal{C}\times 0}(\bar{\mathcal{W}}_i(t_o)) \cap \mathcal{P}^{[\mathbf{g},\tau]}_{\mathbf{c}\times 0}(\bar{\mathcal{W}}_i(t_o)) \subset \bar{\mathcal{W}}_i(t_o)$.

$\mathcal{P}_i$ guarantees that the transition to the new formation, and the transition, will be obstacle-free (from $[\mathcal{C} \times 0] \subset \mathcal{P}_{f_o \to g|i}$ and $\mathcal{P}_i \subset \mathcal{P}_{f_o \to g|i}$) and is likely to make progress in future iterations ($\mathcal{P}_i \subset \mathcal{P}_{o \to g}$). The convex regions are grown in the direction towards the goal $\mathbf{g}$ following an iterative optimization [18] as described in [1].

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. With a slight abuse of notation, 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 regions are shrunk computing local intersections with those regions received from neighbors in the communication graph. The algorithm finishes after d iterations, as shown in Algorithm 2.

Algorithm 2 Distributed Obstacle-Free Region - Robot i

```
1: $\mathcal{P}_i(0) = \mathcal{P}_i$
2: for $k = 0 \dots d-1$ 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:* The execution of Algorithm 2 makes 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,

$$\mathcal{P}_i(d) = \mathcal{P} = \bigcap_{j\in\mathcal{I}} \mathcal{P}_j(0), \ \forall i \in \mathcal{I}. \tag{7}$$

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

$$\mathcal{P}_i(k) = \bigcap_{j\in\mathcal{N}_i(k)} \mathcal{P}_j, \tag{8}$$

for all $k \geq 0$. Clearly Eq. (8) 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. (7) 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

$$\mathcal{P} = \{\mathbf{z} \in \mathbb{R}^4 \mid A\mathbf{z} \leq \mathbf{b}, \text{ for } A \in \mathbb{R}^{n_l \times 4}, \mathbf{b} \in \mathbb{R}^{n_l}\}, \tag{9}$$

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

Compared to Algorithm 1, in this algorithm the robots need to send all the linear constraints at each iteration. In the worst possible scenario this can lead to bigger communication demands than pure flooding if the number of faces of the partial intersections is bigger than each of the individual polytopes. However, in practice that is not the case, usually obtaining a similar number of faces, or even smaller. As a consequence, in most cases the total communication demands of this algorithm is smaller than the cost of flooding, besides keeping the size of messages bounded at all iterations. In addition, the computational cost of computing multiple intersections of fewer constraints is in general smaller than the cost of computing one intersection with a large number of constraints. A similar modification to that of Algorithm 1 can be done to apply Remark 2 to this algorithm, simply by not sending the new region if it is equal to that of the previous iteration.

If $\mathcal{P} = \emptyset$, an alternative convex region $\mathcal{P}_i$ is selected by each robot as described in [1] - Sec. III-C, and consensus on the intersection is repeated. The alternative regions are (a) $\mathcal{P}_i = \mathcal{P}_{f_o\to g|i}$, (b) $\mathcal{P}_i = \mathcal{P}_{o\to g|i}$ and (c) $\mathcal{P}_i = \mathcal{P}_{g|i}$.

***Remark** 2 (Unknown d):* If the diameter, d, is unknown, the consensus runs until convergence for all robots. Convergence for robot i can be checked by computing the maximum distance between $\mathcal{P}_i(k)$ and $\mathcal{P}_i(k+1)$ [1].

C. Optimal formation

Recalling Sec. III-B and following [1], each robot i can compute the optimal formation $\mathcal{F}^* = \mathcal{F}^{l^*}(\mathbf{x}^*)$, via the non-linear optimization defined by

$$\begin{aligned} \underset{\substack{l\in\{1,\dots,f\}\\ \mathbf{x}=[\mathbf{t},s,\mathbf{q}]}}{\arg\min} \quad & ||\mathbf{t}-\mathbf{g}||^2 + w_s||s-\bar{s}||^2 + w_q||\mathbf{q}-\bar{\mathbf{q}}||^2 + c_l \\ s.t. \quad & \{A(\mathbf{t} + s\,\mathrm{rot}(\mathbf{q}, \mathbf{f}^l_{0,j})) \leq \mathbf{b}\} \quad \forall j \in \{1,\dots,n_l\} \\ & \{s\ d^l_0 \geq 2\max(r,h)\} \\ & \{||\mathbf{q}||^2 = 1\}. \end{aligned} \tag{10}$$

Where the deviation to the goal $\mathbf{g}$, a preferred size $\bar{s}$ and orientation $\bar{\mathbf{q}}$ is minimized. The first contraints impose that all vertices are within the convex region $\mathcal{P}$. The second constraint that no two robots within the formation are in collision and the third one that the quaternion has unit length.

To solve this non-convex optimization we employ the non-linear solver SNOPT [19], which internally executes sparse Sequential Convex Programming. Note that all robots execute this optimization with the same parameters, since the template formations are known by all and the convex region $\mathcal{P}$ is computed via consensus. Therefore, even when the optimization is solved individually by each robot, they all obtain the same values for the target formation.

D. Robot assignment to positions in the formation

The result of the computation of Sec. IV-C is a target formation $\mathcal{F}^*$ and its associated set of target robot positions $\{\mathbf{r}^*_1, \dots, \mathbf{r}^*_n\}$ computed with Eq. (2).

Robots are assigned to the goal positions with the objective of minimizing the sum of squared travelled distances, with assignment $\sigma : \mathcal{I} \to \mathcal{I}$ minimizing

$$\min_{\sigma} \quad \sum_{i\in\mathcal{I}} ||\mathbf{p}_i - \mathbf{r}_{\sigma(i)}||^2. \tag{11}$$

There exists several distributed algorithms based on local interactions that are able to find the optimal solution to the

[1] We compute this distance as $\max(\mathrm{dist}(\mathcal{P}_i(k)|\mathcal{P}_i(k+1)), \mathrm{dist}(\mathcal{P}_i(k+1)|\mathcal{P}_i(k)))$, where $\mathrm{dist}(P|Q) = \max(||A\mathbf{v} - b||_\infty$, for $\mathbf{v}$ vertex of Q and $A\mathbf{z} \leq b$ linear constraint representation of P).

above linear program. In particular, in our implementation we make use of the distributed simplex proposed in [20]. The algorithm has a bounded communication cost per iteration and proven finite-time termination.

E. Real-time control

Consider $\mathbf{r}_i^*$ to be the goal position assigned to robot i. To compute a collision-free local motion towards the goal, we employ the recent work on distributed reciprocal velocity obstacles with motion constraints for aerial vehicles [17] and in particular its extension to account for static obstacles, as described in [1]. This approach is able to adapt to changes in the environment and moving obstacles in real-time and respects the dynamics of the robot.

This low level controller drives the robot towards its goal within the target formation at a higher update frequency than that of computing a new target formation.

V. Results

A. Consensus performance

In this section we present simulation results using Monte Carlo experiments to analyze the distributed algorithms 1 and 2. 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.

1) Convex hull: For Algorithm 1 we have considered different number of participating robots, 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. 3 (a). The plot shows the mean and standard deviation over the 100 trials for each scenario.

First of all, it is observed 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 solutions is always a better choice.

2) Intersection of convex regions: In order to analyze Algorithm 2 we have considered again the same number of robots and communication radii, as well as 100 random initial configurations. 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 Section V-B. The results of these experiments are depicted in Fig. 3 (b).

The plot shows a similar behavior to the one in Fig. 3 (a), with smaller demands as n and the communication radius are increased. In most of the cases our algorithm also performs better than flooding. However, in this case when n is small the relative communication cost is greater than one, which means that if the team is small and the network is sparse, it might seem that a better solution is to simply exchange all the constraints and compute a global intersection individually. Nevertheless, even in such case the extra routing control mechanisms and storage capabilities required for flooding make our solution an appealing alternative.

B. Simulation results

We present simulations with teams of quadrotor UAVs, where we employ the same dynamical model and controller of [17], which was verified with real quadrotors. A video illustrating the results accompanies this paper.

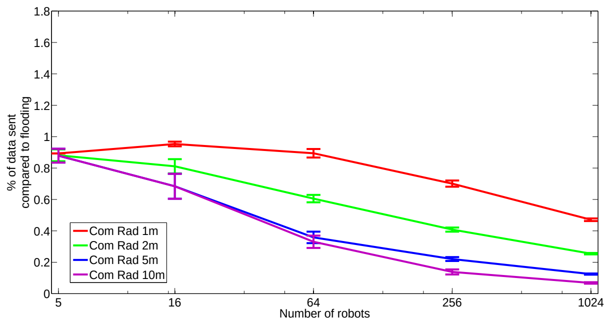
We use SNOPT [19] to solve the non-linear program, a goal-directed version of IRIS [18] to compute the largest convex regions and the Drake toolbox from MIT [2] to handle quaternions, constraints and interface with SNOPT.

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.

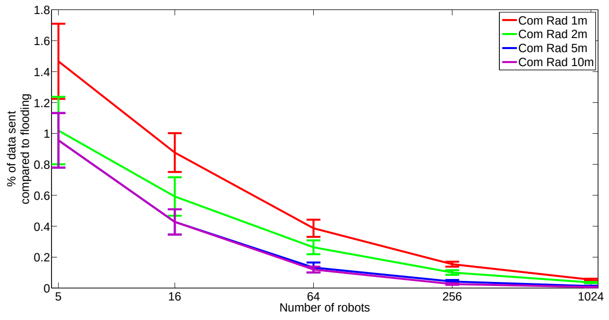
We test the distributed algorithm described in this paper in two scenarios previously introduced in [1]. This provides a direct comparison and evaluation.

1) Four robots: Fig. 4 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.

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



(a) Consensus: convex hull



(b) Consensus: obstacle-free region

Fig. 3. Communication cost of Algorithm 1 (Distributed Convex Hull) (left) and Algorithm 2 (Distributed Obstacle-Free Region) (right) relative to flooding. The plots show the mean and standard deviation over 100 trials for different numbers of robots and communication radii. The convex hull computation always requires less bandwidth than pure flooding in the same number of communication rounds. The collision free region computation performs worse for a small number of robots but as the communication radius and n increase, it overperforms flooding.

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 s) and then to the preferred square, tilted to avoid the dynamic obstacle (37 s). Once the obstacles are cleared they return to the preferred horizontal square formation (45 s).

2) Sixteen robots: Fig. 5 shows the paths of 16 quadrotors moving along a corridor of three different widths. Three default formations are considered: 4x4x1 defined by four vertices (preferred), 4x2x2 defined by eight vertices and 8x2x1 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. Between times 75 s and 110 s the method successfully rotates the formation by 90^o for it to be collision free (the default formations were horizontal, which is also preferred in the cost function).

VI. Conclusion

In this paper we considered a team of networked robots in which each robot only communicates with its close 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. In this work we consider known obstacle locations, within the field of view of the robot, but we can rely on sensing to detect them. Note that, thanks to the consensus on the convex obstacle-free region, the robots do not need to exchange the position of the static obstacles. This approach presents low computational cost and requires substantially fewer communication messages than flooding for consensus.

In several simulations we showed successful navigation in formation where robots may reconfigure the formation as required to avoid collisions and make progress. In fact, the results are comparable to those obtained with our previous centralized approach. Last, but not least, the approach is general and can be adapted to other formation definitions and applications, such as collaborative transportation with mobile manipulations.

Since the approach is local, deadlocks may still occur. In future works we are looking at adding a consensus step to agree on the direction of movement too, which is a distributed max-min problem.

References

[1] J. Alonso-Mora, S. Baker, and R. Siegwart, "Multi-Robot Navigation in Formation Via Sequential Convex Programming," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[2] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[3] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *IEEE Int. Conf. Robotics and Automation*, 2000.

[4] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *of the IEEE Int. Conf. on Robotics and Automation*, 2008.

[5] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 6, pp. 905–908, 2001.

[6] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, March 2015.

[7] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules," *IEEE Transactions on Automatic Control*, vol. 48, pp. 988–1001, June 2003.

[8] J. Cortés, "Global and robust formation-shape stabilization of relative sensing networks," *Automatica*, vol. 45, pp. 2754–2762, Dec 2009.

[9] E. Montijano and A. R. Mosteo, "Efficient multi-robot formations using distributed optimization," in *IEEE 53th Conference on Decision and Control*, 2014.

[10] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and Animation Display with Multiple Robots," *Int. Journal of Robotics Research*, vol. 31, pp. 753–773, May 2012.

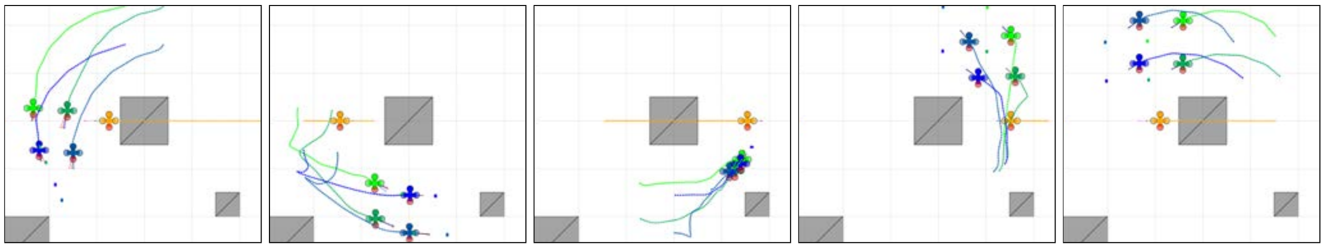
[11] J. Derenick, J. Spletzer, and V. Kumar, "A semidefinite programming framework for controlling multi-robot systems in dynamic environments," in *IEEE Conf. on Decision and Control*, 2010.

[12] J. Alonso-Mora, R. A. Knepper, R. Siegwart, and D. Rus, "Local Motion Planning for Collaborative Multi-Robot Manipulation of Deformable Objects," in *IEEE Int. Conf. Robotics and Automation*, 2015.

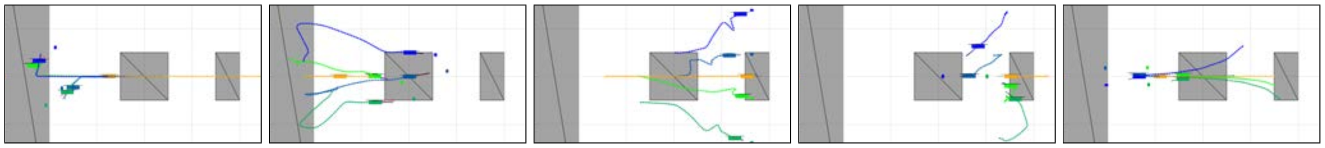
[13] J. C. Derenick and J. R. Spletzer, "Convex optimization strategies for coordinating large-scale robot formations," *IEEE Trans. on Robotics*, vol. 23, Dec. 2007.

[14] A. Kushleyev, D. Mellinger, and V. Kumar, "Towards a swarm of agile micro quadrotors," in *Robotics: Science and Systems*, 2012.

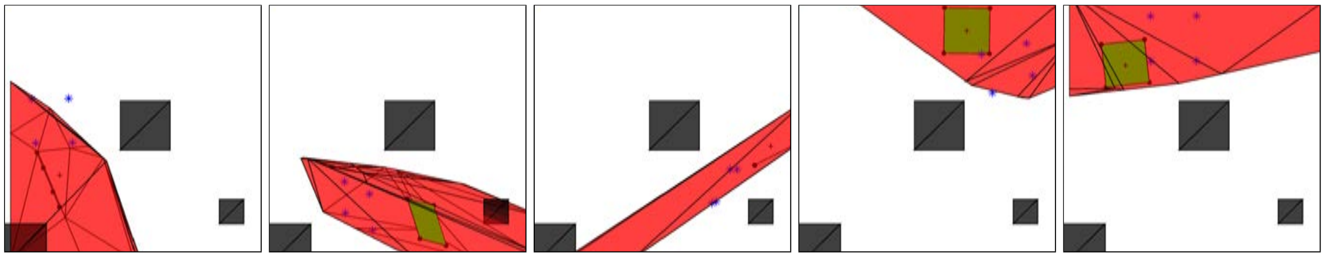
[15] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, "Automated Composition of Motion Primitives for Multi-Robot Systems from Safe LTL Specifications," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.



(a) Top view. From left to right, snapshots at 11 s, 20 s, 30 s, 37 s and 45 s, and paths of the robots in-between.

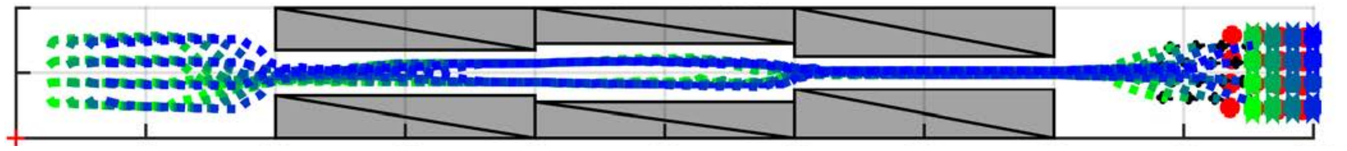


(b) Side view. From left to right, snapshots at 11 s, 20 s, 30 s, 37 s and 45 s, and paths of the robots in-between.

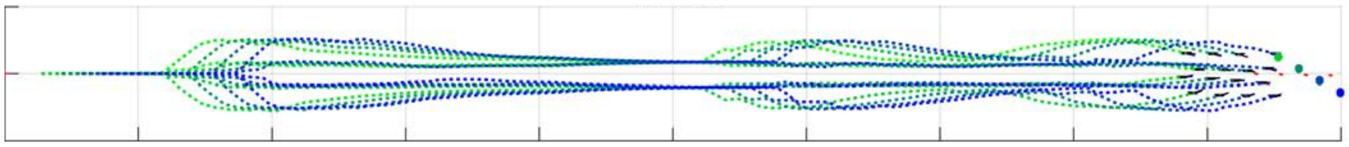


(c) Projection (red) of polytope $\mathcal{P}$ onto 2D top view. It can overlap with obstacles that are not in the field of view of the robots. The projection of the target formation is shown in green and the convex hull $\mathcal{C}$ of the robot positions with blue stars.

Fig. 4. Four quadrotors (green-blue) navigate in a 12 x 12 x 6 m^3 scenario with three static obstacles (grey) and a dynamic obstacle (yellow). The four quadrotors track a circular motion and locally reconfigure the formation to avoid collisions and make progress.



(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. 5. Sixteen quadrotors navigate along a 100 x 10 x 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 4x4x1 formation and tilt it to vertical, to pass trough the narrow corridors. In the wider middle region they transform to a 4x2x2 formation, which has lower cost than the vertical 4x4x1. They finally transition towards 4x4x1.

[16] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012.

[17] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Auton. Robot.*, Jan. 2015.

[18] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," *Workshop on the Algorithmic Fundamentals of Robotics*, 2014.

[19] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM journal on optimization*, vol. 12, no. 4, pp. 979–1006, 2002.

[20] M. Burger, G. Notarstefano, F. Allgower, and F. Bullo, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.