Distributed multi-robot formation splitting and merging in dynamic environments

Zhu, Hai; Juhl, Jelle; Ferranti, Laura; Alonso-Mora, Javier

DOI
10.1109/ICRA.2019.8793765

Publication date
2019

Document Version
Final published version

Published in
Proceedings of the International Conference on Robotics and Automation, ICRA 2019

2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Distributed Multi-Robot Formation Splitting and Merging in Dynamic Environments

Hai Zhu, Jelle Juhl, Laura Ferranti and Javier Alonso-Mora

Abstract— **This paper presents a distributed method for splitting and merging of multi-robot formations in dynamic environments with static and moving obstacles. Splitting and merging actions rely on distributed consensus and can be performed to avoid obstacles. Our method accounts for the limited communication range and visibility radius of the robots and relies on the communication of obstacle-free convex regions and the computation of an intersection graph. In addition, our method is able to detect and recover from (permanent and temporary) communication and motion faults. Finally, we demonstrate the applicability and scalability of the proposed method in simulations with up to sixteen quadrotors and real-world experiments with a team of four quadrotors.**

I. INTRODUCTION

Multi-robot systems will be employed in several application domains, such as surveillance [1], area converge [2], and collaborative manipulation [3]. In particular, these applications require a team of robots to move in *formation* to maintain a communication network, monitor an area, or cooperatively carry an object.

In dynamic environments with static and moving obstacles, the presence of obstacles would require strategies to *split* (for example to avoid the obstacle or in case of failure of one of the robots in the formation) and subsequently *merge* (for example, to restore the original formation after a splitting or once the faulty robot recovers from the fault) the formation. These strategies are nontrivial given *(i)* the nonconvexity of the workspace in which the formation is moving, *(ii)* the limited sensing and communication resources of the robots, and *(iii)* the presence of possible communication or stuck faults.

A. Related Works

Extensive work exists for multi-robot formation control [4]. These approaches include reactive methods [5], navigation functions [6], potential fields [7] and controller syntheses [8]. These formation control approaches, however, are limited to either *obstacle-free* environments or *planar* environments with static obstacles. In contrast, our method allows one to consider *three-dimensional dynamic* environments with both static and moving obstacles via formation splitting and merging.

Several approaches rely on optimization techniques to solve the navigation of teams of robots. For example, the problem can be formulated using convex optimization techniques [9]. Alternatively, the navigation can be formulated as a centralized non-convex problem solved either offline [10], [11] or online [12], [13], [14]. Finally, the formation control can be formulated using distributed optimization techniques [3], [15]. Our design rely on distributed and online optimization to solve the splitting and merging of a formation in dynamic environments (with static and moving obstacles of polygonal shape).

Several methods use splitting and merging for formation control with obstacles. In [16], splitting and merging actions are performed by adding additional repulsive forces among the robots and using the flocking algorithm [17]. In [18], the actions are performed by combining artificial potentials with a leader-follower formation control scheme. In [19] the actions are performed by using a switching control law between leader mode and follower mode. Compared to these approaches, we take into account the limited sensing and communication range of robots and employ consensus on convex regions to compute the formation splitting and merging actions in a distributed fashion.

Our work builds on a distributed method for local multi-robot formation control among obstacles [20]. In [20], the target formation parameters are optimized in obstacle-free convex regions. If navigation in formation is not possible, the formation breaks into individuals and each robot navigates to the goal independently. Differently, our method allows the group formation to split into smaller subteams which also navigate in formation for collision avoidance purposes and merge back into a larger formation after avoiding the obstacles. Compared to [20], our design builds a graph from the intersection of the obstacle-free convex regions of each robot. Then, our algorithm performs splitting/merging actions using a graph partition algorithm [21]. Furthermore, our strategy allows one to detect and handle communication and stuck faults of individual robots.

B. Contribution

The main contribution of this paper is a real-time distributed method to split and merge a team of robots navigating in dynamic environments with static and moving obstacles. The method relies on distributed consensus (using network flooding) to split/merge the formation and to achieve scalability with the number of robots (as also shown in simulation). The proposed algorithm builds a graph based on the intersection of the obstacle-free convex regions of each robot. Then, our method decides on formation splitting/merging by using a graph partition algorithm [21]. Limited sensing and

This work is supported by NWO Veni grant 15916.

The authors are with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD, Delft, The Netherlands {h.zhu; l.ferranti; j.alonsomora}@tudelft.nl

communication range of robots are considered, as well as robustness to faults.

II. PRELIMINARIES

In the following, we introduce some useful definitions and assumptions.

Robots: Similar to [20], we assume that all robots have the same dynamic model and cylindrical non-rotating shape of radius r and height $2h$ in the vertical dimension. The position of the i-th robot at time t is denoted by $\mathbf{p}_i(t) \in \mathbb{R}^3$, $i \in I = \{1, \dots, n\} \subset \mathbb{N}$.

Sensing and communication range: We consider that each robot has a limited field of view, which is modeled as a sphere of given radius centered at the robot's position. The field of view of the i-th robot is denoted by $B_i \subset \mathbb{R}^3$, $i \in I$. To account for the limited communication range of the robots, let $G = (I, E)$ be the communication graph associated with the team of robots, where I indicates the set of vertices of the graph, and E the set of edges. In particular, each edge $(i, j) \in E$, denotes the possibility of robots i and j to directly communicate with each other. The set of neighbors of robot i is denoted by $N_i = \{j \in I \mid (i, j) \in E\}$. We assume that G (the graph associated with the team of robots) is connected, that is, for each pair of robots (i, j) there exists a path that links the two robots. We denote by d the diameter of G, which is the longest among all the shortest paths between any pair of robots in the team.

Formations: As in [20], we consider a predefined set of default formations, such as square, line, or T formations. This predefined set of formations is known by all robots in the team. Denote by $\mathbf{p}_c(t)$ the position of the center (typically the centroid) of a team in formation at time t. We assume that a global reference center position $\mathbf{p}_r$ for the team is given and is known by all robots. The reference position could be an input given by the operator, or the output of a global planning algorithm.

Obstacles: We consider static and moving obstacles:
Static obstacles. Let $O \subset \mathbb{R}^3$ define the global map of obstacles and $O_i = B_i \cap O$ the set of static obstacles seen by Robot i. Let $\tilde{O}_i$ be the set O_i dilated by half of the robot's volume. $\tilde{O}_i$ defines the set of positions for which Robot i would be in collision with the static obstacles within its visibility radius.
Moving obstacles. Let $J_i \subset \mathbb{N}$ be the list of observed moving obstacles of Robot i. For a moving obstacle $j \in J_i$ at time t, let $D_j(t) \subset \mathbb{R}^3$ be the volume it occupies, and $\tilde{D}_j$ its dilation by half of the robot's volume. The constant velocity assumption is employed (but the method is not restricted to it) to predict the future positions of moving obstacles.

Obstacle-free workspace: At current time t_0, the set of static and dynamic obstacles seen by Robot i within a time horizon τ is defined as follows:

$$\hat{O}_i(t_0) := \tilde{O}_i \times [0, \tau] \bigcup_{t \in [0,\tau], j \in J_i} \tilde{D}_{i|j}(t_0 + t) \times t \subset \mathbb{R}^4. \quad (1)$$

where $\times$ denotes the Cartesian product of two sets. Consequently, the position-time obstacle-free workspace for Robot i is defined as follows:

$$W_i := \mathbb{R}^3 \times [0, \tau] \setminus \hat{O}_i(t_0) \subset \mathbb{R}^4. \quad (2)$$

Obstacle-free convex region: Denote by $P_i \in \mathbb{R}^3 \times [0, \tau]$ the obstacle-free convex region computed by Robot i, which is embedded in position-time space. P_i guarantees that the transition of the robot to the new formation will be obstacle-free and is likely to make progress in future iterations. The computation of P_i is based on a fast iterative method as described in detail in [20]. Note that due the to limited field of view, each robot i in a team can computes a different P_i.

III. FORMATION CONTROL

Our work relies on the method presented in [20]. We summurize the main results in the remainder of this section.

Given an initial formation configuration and a reference goal (the center of the formation), [20] computes the locally optimal target formation parameters and navigates all robots to the target formation, while avoiding static and dynamic obstacles. The local formation control method mainly consists of the following steps. First, all robots agree on a common obstacle-free convex region, and then compute a target formation therein. To compute the common obstacle-free region, each robot computes an obstacle-free region with respect to its limited field of view. Then, the robots collaboratively compute the intersection of all regions. Second, robots are assigned, with a distributed optimization algorithm, to target positions within the target formation. Third, in a faster loop, each robot navigates towards its assigned goal within the target formation by employing a low level local planner [22] that generates collision-free inputs according to the robots dynamics.

Note that the common obstacle-free region (first step) might be empty or not large enough. In this case, no feasible target formation exists and the algorithm breaks the formation into individual robots. In this paper, instead of breaking the formation into individuals, we present a method that relies on a graph partition algorithm to split the formation into smaller subteams (which also navigate in formation). The subteams can merge into larger teams in formation when certain conditions are satisfied. As the distributed formation control approach in [20], our proposed formation splitting and merging method is also distributed. We assume that one splitting action can lead to multiple sub-formations, but one merging action only involves two sub-formations to be merged.

IV. PROPOSED METHOD

A. Overview

We now present our method to compute splitting and merging actions for a multi-robot team. Fig. 1 provides an overview of the proposed method. In particular, the method consists of the following steps:

1) *Distributed splitting/merging decision making and computation:*

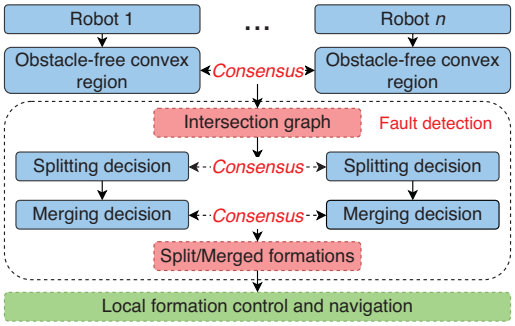


Fig. 1: Overview of the proposed method. The components in blue depict the local steps, while the components in red highlight the consensus steps.

- Each robot computes an obstacle-free convex region in position-time space P_i [20].
- All the robots perform distributed consensus (network flooding) to compute the intersection of inter-robot obstacle-free convex regions $P_{ij} = P_i \cap P_j$ and $P = \bigcap_{i \in I} P_i$ $(i, j \in I, i \neq j)$ and construct an intersection graph G_{int}.
- When some conditions are satisfied (Section IV-B), all the robots in a team decide to split the formation into sub-formations using a graph partition algorithm [21].
- When some conditions are satisfied (Section IV-C), the robots in neighboring teams merge into a larger formation.

2) *Local formation control and navigation:*
   - Once the splitting/merging decisions are made, the team or sub-teams of robots are controlled to navigate to their goals using a local formation controller. The formation control algorithm is summurized in Section III.

3) *Fault detection*
   - Fault detection of (temporary and permanent) communication loss and (temporary and permanent) stuck fault of an individually robot in the formation is performed when computing splitting/merging actions.

B. Formation Splitting

When the common convex region P of a team is empty or too small (i.e., the team does not fit), for example due to nearby obstacles, a feasible target formation within the obstacle-free workspace for the team may not exist. Our design overcomes this issue by splitting the team of robots into smaller formations to avoid the obstacles.

Algorithm 1 details the method to compute formation splitting actions. The algorithm consists of three main steps:

1) Computation of convex regions and network flooding: Steps 1-5.
   - Each robot $i \in I$ in the team computes its corresponding obstacle-free convex region P_i.

Algorithm 1 Formation Splitting

1: **for** Each robot $i \in I$ **do**
2: Compute obstacle-free convex region P_i
3: Send P_i to all $j \in N_i$
4: Receive P_j and P_k from all $j \in N_i$ and $k \in N_j$
5: **end for**
6: Compute $P_{ij} = P_i \cap P_j$ between Robots $i, j \in I$
7: Compute the size of the intersection region $A_a(i,j) = V(P_{ij}) = \det(P_{ij})$
8: **if** $\exists i, j \in I : A_a(i,j) < V_{\min}$ **then**
9: `split` $= 1$
10: Split the team using `graph_partition`(A_a)
11: **else**
12: `split` $= 0$
13: **end if**

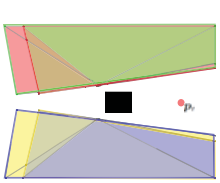
   - Then, P_j $(j \in I, j \neq i)$ is obtained by using a network flooding algorithm that takes into account the robots' limited communication range.

2) Computation of intersection graph: Steps 6-7.
   - After obtaining P_j $(j \in I, j \neq i)$, the algorithm builds an intersection graph G_{int}. The intersection graph is an undirected graph in which each vertex represents a robot in the team, and each edge is $P_{ij} := P_i \cap P_j$. The intersection graph is represented by a symmetric adjacency matrix A_a in which $A_a(i,j) = V(P_{ij}) = \det(P_{ij})$. In general, we denote by $V(P) := \det(P)$ the size of a convex region P and by $V_{\min}$ the smallest size that allows for a template formation.

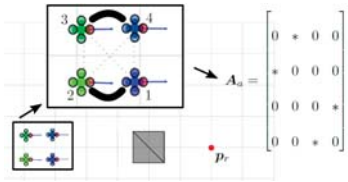
3) Splitting decision making and action computation: Steps 8-13.
   - The splitting actions are computed based on A_a. If there exists an edge in A_a such that $A_a(i,j) < V_{\min}$, then it indicates that there is no feasible target formation within the obstacle-free workspace for the team. Thus, a decision is made to split the team, as follow.

Our design relies on a graph partition algorithm (step 10 `graph_partition`(A_a)). The objective when splitting a large team is to minimize the size of region intersections P_{ij} corresponding to robot pairs (i,j) that belong to different teams after splitting. In other words, the sum of the cut edges is supposed to be minimized while partitioning the intersection graph A_a into subgraphs. Several algorithms exist to solve the graph partition problem [23]. Our design uses [21] that is based on spectral factorization combined with a k-means clustering and can be implemented very efficiently. Fig. 2 illustrates the relationship between obstacle-free convex regions and the corresponding intersection graph for a team of four quadrotors. Graph partition results in the team splitting into two line sub-formations to avoid the obstacles.

Remark 1. Note that all the robots in the team execute the intersection graph construction and partition with the same



(a) Top views of the obstacle-free convex regions of each robot. The goal of the team is $\mathbf{p}_r$ and the black blocks are obstacles.



(b) Intersection graph and graph partition. The solid line represents the size of the intersection P_{ij}, the dotted line represents empty intersection, and the $*$ in A_a represent non-zero elements.

Fig. 2: Scheme of progress from the obstacle-free convex regions to the intersection graph for a team of four quadrotors.

parameters, given that the template formation information is known by all the robots in the team. Furthermore, their convex regions P_i are communicated via network flooding. Therefore, even if the splitting action is computed individually by each robot, they all obtain the same splitting results.

C. Formation Merging

The conditions to be satisfied to merge two sub-formations k and l into one larger formation are defined as follows:

- The distance of the centers of the two sub-formations satisfies $\mathbf{p}^{kl} = \left\|\mathbf{p}_c^k - \mathbf{p}_c^l\right\| \leq \mathbf{p}_{\max}$, where $\mathbf{p}_{\max}$ is the maximum allowed distance for merging.
- The size of intersection of the common obstacle convex regions of the two sub-formations satisfies $V(P^{kl}) = V(P^k \cap P^l) \geq V_{\min}$, where $P^k = \bigcap_{i \in I^k} P_i$ and $P^l = \bigcap_{i \in I^l} P_i$.
- The robots are within communication range.

The superscripts k, l denote the numbered sub-teams split from a large formation and the subscripts i, j denote the numbered robots in a sub-team. Algorithm 2 describes how to compute formation merging actions for one of the sub-teams k. The algorithm consists of three main steps:

1) Checking merging conditions: Steps 1-9.
   - Each robot in a sub-team k contacts robot j of another sub-team l within its communication range requesting P^l and the position of center $\mathbf{p}_c^l$. Then, Robot i in sub-team k checks if the conditions for merging defined above are satisfied. Me_i denotes the preferred sub-team in which Robot i in sub-team k is more likely to merge with.
2) Agreement on merging request: Steps 10-14.
   - Locally, if Robot i meets the merging requirements with multiple other sub-teams, it communicates which team it would like to merge with. In particular, its preference is the sub-team with the larger region intersection $V(P^{kl})$. At sub-team level, the robots agree on the sub-team to merge with via distributed consensus. Note that d^k indicates the number of communication rounds for Team k.

Algorithm 2 Formation Merging

1: **for** Each robot $i \in I^k$ **do**
2: Send P^k and $\mathbf{p}_c^k$ to one $j \in I^l, j \in N_i$ of all other sub-teams l
3: Receive P^l and $\mathbf{p}_c^l$ from one $j \in I^l, j \in N_i$ of all other sub-teams l
4: **if** $\left\|\mathbf{p}_c^k - \mathbf{p}_c^l\right\| \leq p_{\max}$ and $V(P^k \cap P^l) \geq V\min$ **then**
5: $Me_i(0) = l$
6: **else**
7: $Me_i(0) = 0$
8: **end if**
9: **end for**
10: **for** $m = 0, \ldots, d^k - 1$ **do**
11: Send $Me_i(m), V(P^k \cap P^{Me_i(m)})$ to all $j \in I^k, j \in N_i$
12: Receive $Me_j(m), V(P^k \cap P^{Me_j(k)})$ from all $j \in I^k, j \in N_i$
13: $Me_i(m+1) = Me_i(m)$ or $Me_j(m)$ that maximize $V(P^k \cap P^{Me_i(m+1)})$
14: **end for**

Finally, Sub-team k exchanges its merging request with sub-team l. If sub-team l also requests to merge with k, both sub-teams merge into a larger team.

D. Fault Detection

Formation splitting and merging can occur when there is a failure with keeping the formation. We consider two types of faults:

1) *Communication loss*. One of the robots in the formation is unable to communicate with the other robots in the team. Typical causes of communication loss are related, for example, to failures in the robot's communication module. The fault can be permanent or temporary. Our design takes both occurrences into account. Due to the loss of communication, the communication graph G of the team becomes unconnected. We proceed as follows to overcome this issue. First, we assume that the information in the consensus steps are labeled according to its creator. Second, we know that in case of a connected communication graph, the consensus steps converge in at most d communication rounds [20]. This information is used to detect which robot in the team stops communicating. After d communication rounds, the non-communicating robots will be discarded from the team. In this way, the new reduced formation can proceed towards its goal without the faulty robots. If the fault in the communication unit of Robot i is only temporary, the robot tries to communicate with other neighboring teams and merge into one of them using Algorithm 2.
2) *Stuck faults*. One of the robots might remain stuck. A robot stops moving towards the goal of the formation, for example, due to faulty actuators or internal problems. Stuck faults can be permanent or temporary. Our algorithm should be able to detect

these faults to prevent the whole formation to stop. We consider both permanent and temporary actuator faults. Let $\|\Delta\mathbf{p}_i\|$ be the position progress towards the goal between two consecutive time steps of Robot i. Let $\Delta\mathbf{p}_{\min}$ be a predefined minimum allowable progress distance $\Delta p_{\min}$ in n_{dead} consecutive time steps. If $\sum_{n_{\text{dead}}} \|\Delta\mathbf{p}_i\| \leq \Delta\mathbf{p}_{\min}$, our algorithm considers the robot as stuck. Once a robot finds itself stuck, it notifies the other robots in the team and leaves the formation. If the robot starts moving again after the detection of the stuck fault (temporary fault), it tries to communicate with other neighboring teams and merge into one of them using Algorithm 2.

Remark 2. Note that d, $\Delta p_{\min}$, and n_{dead} are tuning parameters. Their selection is based on a trade-off between performance of the detection and to avoid misdetections. Note that our fault-detection strategy is able to recover from misdetections and allows the *faulty* robots to re-join a formation.

V. RESULTS

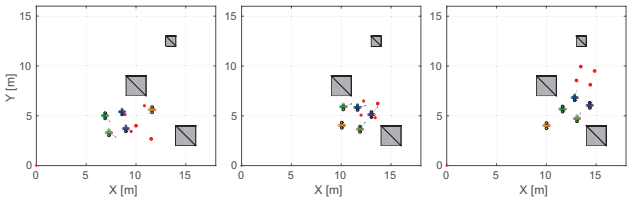
In this section, we illustrate the effectiveness of the proposed method in both simulations and real experiments with teams of quadrotors. A video demonstrating the results accompanies this paper. For the quadrotors we employ the same dynamical model and controller of [22]. We preformed the computation with two standard computers (Quadcore Intel i7 CPU@2.8 GHz). In one computer we executed the method of this paper and gave the target position for each quadrotor. In particular, we computed obstacle-free convex regions, performed consensus rounds, determined splitting/merging actions, and computed the target positions for each quadrotor. These computations are performed in a continuous manner, that is, as soon as one execution is finished we recompute. In the second computer we received the current state of the quadrotors and obstacles and each quadrotor's target position obtained from the first computer. Then we ran the local collision avoidance planner and sent input commands to the quadrotors. The computations are performed in MATLAB and the communication is handled with ROS.

A. Simulation Results

Simulation setup: We tested our approach in the following scenarios:

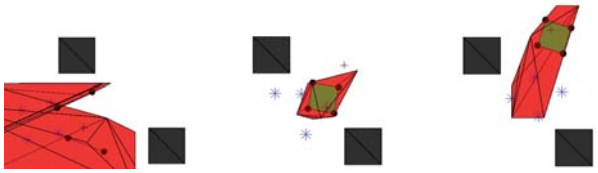
Scenario 1. This scenario considers four quadrotors operating in dynamic environments, with three static obstacles and one moving obstacle. The quadrotors have to track a circular trajectory, while avoiding the obstacles. In this scenario, we set the time horizon $\tau = 3$ sec. Furthermore, each formation checks whether to perform splitting and merging actions every $f_f = 1$ sec. The visibility distance and communication range of robots are set as $r_B = 6$ m and $r_C = 3$ m respectively.

Scenario 2. This scenario considers sixteen quadrotors operating in a static environment. The quadrotors cooperate to pass a static obstacle and a narrow corridor.



(a) t = 7 sec (b) t = 15 sec (c) t = 18 sec

Fig. 3: Snapshots of robots and target positions (red dots).



(a) t = 7 sec (b) t = 15 sec (c) t = 18 sec

Fig. 4: Snapshots of common obstacle-free convex regions (red), target formation (black dots) and robot positions (blue stars).

The individual collision avoidance planners run at 5 Hz and each quadrotor has a preferred speed of 1.5 m/s. In the figures, gray cuboids represent static obstacles and the yellow quadrotors represent dynamic obstacles. The red dots represent the target position of each drone.

Results: Fig. 3 and Fig. 4 present the results obtained for Scenario 1. Fig. 3 shows four snapshots of the four quadrotors at (starting from the left) 7 sec, 15 sec and 18 sec. Fig. 4 shows the obstacle-free convex regions, target formation, and robot positions at the same time instances. The four quadrotors start from a horizontal square formation. At $t = 7$ sec, the formation starts to split into two sub-formations, since the common convex region is too small for a target square configuration. Each sub-formation has two quadrotors in line to avoid the dynamic obstacle. At $t = 15$ sec, the two sub-formations merge back into a square one, given that there exists a common obstacle-free convex region between the two teams. At $t = 18$ sec, the formation continues to move in a square formation while tracking the circular reference trajectory.

Table I provides the computation time for each quadrotor for our MATLAB implementation. Splitting checking takes a longer time due to the required consensus action among all robots to compute the intersection graph. Graph partition also takes a comparatively large part of the computation time. However, Note that the graph partition is only executed if the team must split. The total mean computation time is 238.6

TABLE I: Computation time [ms] of the implementation

Computations	Min.	Mean.	Max.	Std.
Convex regions	19.4	45.5	163.6	22.4
Splitting checking	60.9	72.7	120.0	13.6
Graph partition	68.6	68.6	68.6	0.0
Merging decisions	4.1	11.4	17.8	4.7
Local formation control	14.9	40.4	128.4	23.8

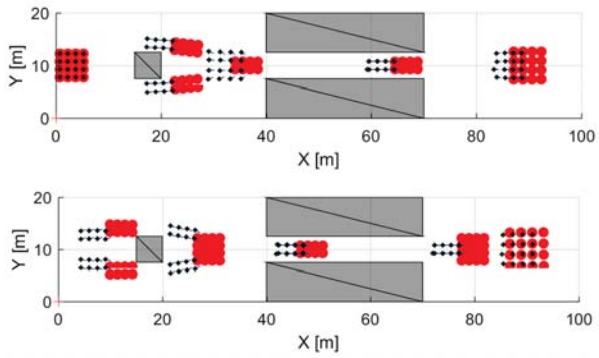


Fig. 5: Alternating top view of snapshots of quadrotors (black/blue) and target positions (red) between t = [0,100] s.

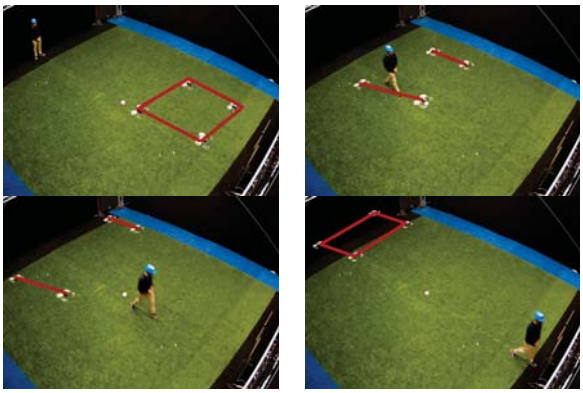


Fig. 6: Isometric view, from left to right. Snapshots of the quadrotor team avoiding a moving human by splitting and afterwards merging.

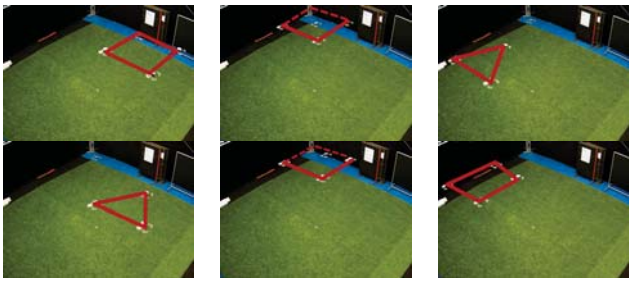


Fig. 7: Isometric view, from left to right, top to bottom. Snapshots of the quadrotor team tracking a circular trajectory while an individual robot stops moving.

ms, which shows a real time performance of our proposed approach.

Fig. 5 presents the results obtained for Scenario 2. The quadrotor team starts from a preferred $4\times4\times1$ formation (top row). When the first static obstacle is encountered, the large formation splits into two sub-formations (bottom row), each containing eight quadrotors with a $4 \times 2 \times 1$ configuration (10 sec). The obstacle is then avoided by each formation individually. The two sub-formations re-merge into a large one (top row) as soon as they get into communication range and reconfigure into a $4\times2\times2$ configuration defined by eight vertices to squeeze through the narrow corridor (40 sec). After leaving the narrow corridor (bottom row), the formation returns to its preferred $4\times4\times1$ configuration (top row).

B. Experimental Results

Experimental setup: Our experimental platform is the Parrot Bebop 2 quadrotor. The pose of each quadrotor and obstacle (human) is obtained using an external motion capture system (OptiTrack) at a high update rate. The human's velocity is obtained via a standard Kalman Filter. The local collision avoidance planner is run at 10 Hz. The communication and visibility radius of the quadrotors are both set as 3 m to guarantee connectedness of the communication graph.

First experiment: A human serves as a dynamic obstacle and walks towards a team of four quadrotors in a square formation. Fig. 6 shows four snapshots of the experiment. The large square formation splits into two line sub-formations successfully when encountering the approaching human. Then, both sub-formations avoid the human safely, while progressing towards their goal. After avoiding the human, the two sub-formations merge into a larger team in the square formation.

Second experiment: A team of four quadrotors are tracking a circular trajectory while one of the quadrotors stops moving temporarily. Fig. 7 shows six snapshots of the experiment. The four quadrotors start in a square formation and reconfigure into a triangle formation after one of them stops moving and leaves the team, according to our fault-detection strategy. The triangle formation keeps tracking the circular trajectory. When the faulty quadrotor recovers from the fault, it successfully merges into the team and the formation returns back to the original square configuration.

VI. CONCLUSIONS AND FUTURE WORK

We presented a novel splitting and merging strategy for a team of networked robots navigating in formation among static and dynamic obstacles using distributed consensus and graph partitioning. First, the robots construct an intersection graph and decide whether to split the formation by checking the edges of the graph. Then, two sub-formations merge into one when the intersection of the obstacle-free convex regions of two sub-formations contains both formations (i.e., it allows for a larger formation). A fault-detection strategy is also proposed to account for communication and stuck faults. Our design accounts for limited communication and visibility range of the robots. The proposed method is scalable to large teams of robots, works in real time, and is robust to faults as both our simulations and real-world experiments show.

Our approach is a local planner and deadlocks may still occur. Hence, as part of our future work we plan to incorporate a global planner to prevent these issues. Alternatively, detailed robustness analysis of the method on the impact of unforeseen disturbances and state uncertainties of the robots should be further investigated in the future.

REFERENCES

[1] A. Jahn, R. J. Alitappeh, D. Saldaa, L. C. A. Pimenta, A. G. Santos, and M. F. M. Campos, "Distributed multi-robot coordination for dynamic perimeter surveillance in uncertain environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 273–278.

[2] A. Breitenmoser and A. Martinoli, "On combining multi-robot coverage and reciprocal collision avoidance," in *Proc. Int. Symp. Distrib. Autom. Robot. Syst.*, 2016, pp. 49–64.

[3] J. Alonso-Mora, R. Knepper, R. Siegwart, and D. Rus, "Local motion planning for collaborative multi-robot manipulation of deformable objects," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5495–5502.

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

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

[6] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 128–133.

[7] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, pp. 73–80.

[8] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized controllers for shape generation with robotic swarms," *Robotica*, vol. 26, no. 5, pp. 691–701, 2008.

[9] J. Derenick, J. Spletzer, and V. Kumar, "A semidefinite programming framework for controlling multi-robot systems in dynamic environments," in *Proc. IEEE Conf. Decis. Control.*, 2010, pp. 7172–7177.

[10] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Auton. Robot.*, vol. 35, no. 4, pp. 287–300, 2013.

[11] 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 *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2014, pp. 1525–1532.

[12] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2012, pp. 1917–1922.

[13] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5954–5961.

[14] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Auton. Robot.*, vol. 37, no. 4, pp. 401–415, 2014.

[15] S. S. Kia, J. Cortés, and S. Martínez, "Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication," *Automatica*, vol. 55, pp. 254–264, 2015.

[16] Z. Chen, T. Chu, and J. Zhang, "Swarm splitting and multiple targets seeking in multi-agent dynamic systems," in *Proc. IEEE Conf. Decis. Control.*, 2010, pp. 4577–4582.

[17] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Trans. Autom. Control.*, vol. 51, no. 3, pp. 401–420, 2006.

[18] K. Raghuwaiya, J. Vanualailai, and B. Sharma, "Formation splitting and merging," in *Proc. Int. Conf. Swarm. Intell.*, 2016, pp. 461–469.

[19] P. Ogren, "Split and join of vehicle formations doing obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 1951–1955.

[20] J. Alonso-Mora, E. Montijano, T. Nageli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Auton. Robot.*, Jul 2018. [Online]. Available: https://doi.org/10.1007/s10514-018-9783-9

[21] J. P. Hespanha, "An efficient MATLAB algorithm for graph partitioning," University of California, Tech. Rep., 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.6722&rep=rep1&type=pdf

[22] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Auton. Robot.*, vol. 39, no. 1, pp. 101–121, 2015.

[23] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, "Recent advances in graph partitioning," in *Algorithm. Eng.* Springer, Cham, 2016, pp. 117–158.