Distributed Formation Control for Multi-Vehicle Systems with Splitting and Merging Capability

Szilárd Novoth, Qian Zhang, *Member, IEEE,* Kang Ji, Dingli Yu, *Member, IEEE,*

***Abstract*—This work develops a novel strategy for splitting and merging of agents travelling in formation. The method converts the formation control problem into an optimization problem, which is solved among the agents in a distributed fashion. The proposed control strategy is one type of Distributed Model Predictive Control (DMPC) which allows the system to cope with disturbances and dynamic environments. A modified Alternating Direction Method of Multipliers (ADMM) is designed to solve the trajectory optimization problem and achieve formation scaling. Furthermore, a mechanism is designed to implement path homotopy in splitting and merging of the formation, which examines the H-signature of the generated trajectories. Simulation shows that, by using the proposed method, the formation is able to automatically resize and dynamically split to better avoid obstacles, even in the case of losing communication among agents. Upon splitting the newly formed groups proceed and merge again when it becomes possible.**

***Index Terms*—Distributed control, networked control systems, multi-agent systems, homology classes, trajectory optimization**

I. Introduction

RECENT improvements regarding micro-controllers and communication technologies have led to increased attention towards multi-vehicle systems (MVS). In comparison to systems consisting of a single vehicle, MVSs have the capability to accomplish a task more efficiently and reliably.

One significant research direction about MVSs is formation control of swarm autonomous systems. Vehicles moving in formation can be applied to cooperatively transport loads or to monitor or survey the environment. In formation control, four major problems can be identified: 1) maintaining the formation, 2) formation reconfiguration, 3) formation scaling and 4) splitting and merging. Many studies focus on 1), a few works on 2) and 3) and only a handful considers 4) due to its high complexity. However, splitting or merging of formations is an important topic. In environments occupied by static and dynamic obstacles, splitting the formation could be highly beneficial in order to better avoid obstacles or to prevent communication loss due to potentially signal-blocking obstacles, like a radio-tower.

In this work, we tackle multiple problems in formation control by proposing a new Distributed Model Predictive Control (DMPC) strategy, which combines a modified Alternating Direction Method of Multipliers (ADMM) with the path homotopy concept. First, formation keeping is considered in a dynamic environment. Second, the proposed algorithm enables the scaling of the formation to fit through narrow corridors or to better avoid obstacles. Third, the proposed method based on complex analysis enables splitting and merging of formation in a practical way. This work is a very early attempt to achieve flexible adaptation of formation, considering potential loss of communication among agents. It is the first time to apply the path homotopy concept to facilitate the formation splitting and merging.

S. Novoth, Q. Zhang, K. Ji and D. Yu are with the Department of Electronics and Electrical Engineering, Liverpool John Moores University, Liverpool L3 3AF, U.K. (email: s.novoth@2018.ljmu.ac.uk, q.zhang@ljmu.ac.uk, k.ji@2017.ljmu.ac.uk, d.yu@ljmu.ac.uk)

II. Related Work

State-of-the-art formation control strategies can be divided into three main categories: behavioral approaches [1], leader-follower approaches [2] and virtual structure based approaches [3]. These methods usually focus on keeping the formation while traveling on the prescribed path. Defining this path is commonly handled by an external computational unit which generates it either in real-time or beforehand. For the calculation of the path, a well established control method called Model Predictive Control (MPC) can be used.

The basis of MPC is to transform an original control problem into an optimization problem which is then solved in every sampling time. The advantage of MPC lies in its ability to handle multi-variable interactions, constraints on the input/output variables and nonlinear dynamics.

Centralized MPC is not robust against failure of the main computational unit. Despite significant improvements in processing power, they still cannot cope with large network sizes. DMPC therefore has seen a surge in the number of applications. It breaks down a large optimization problem into smaller sub-problems, which are then solved by individual agents and so the system becomes scalable. Furthermore, the inherent modularity of the system allows for simple formulation of the individual objectives and an easy way to add or discard agents from the group. For this reason, distributed systems are less prone to failure, hence any misbehaving unit can be removed and potentially replaced if needed.

Some works have been published that apply DMPC for the control of multi-agent systems to move in a prescribed formation. In [4] so-called "assumed" states are exchanged between the vehicles and each agent solves its optimization problem taking these values in consideration. More recently another method, ADMM has surfaced to solve consensus problems for multi-vehicle systems. It is an augmented Lagrangian method that was developed in the 1970s and later reviewed in [5]. In [6] ADMM was used for a distributed flocking control strategy which allowed the vehicles to follow a prescribed trajectory while avoiding collisions between agents and obstacles. ADMM was used in [7] to drive a fleet of vehicles

represented with double integrator dynamics. Similarly [8] also used ADMM to control a MVS with linear dynamics through a dynamic environment. In a later work the method was extended for systems possessing non-linear dynamics [9]. The optimization method presented in this paper is a modified version of the ideas used in [8], [9].

The research relating to splitting and merging of autonomous systems is still at a very early stage. One important contribution to the topic can be found in [10] and [11], where the intersection (or the lack thereof) of the obstacle-free convex regions is the indicator whether splitting or merging of groups should be executed. In [12] Homotopy Continuation Method (HCM) was used to generate various collision free paths in 2D and 3D environments. In [13] distinct paths were generated using Path Guided Optimization (PGO). Each of these paths was then optimized and the least cost one was implemented.

III. Trajectory optimization

The goal in trajectory optimization is to find an optimal state and input trajectory $\{\mathbf{x}^*(t), \mathbf{u}^*(t)\}\ \forall t \in [0, T]$, that brings a dynamical system from its initial state to its final state. The trajectory is optimal in the sense that it minimizes some cost functions while satisfying all the equality and inequality constraints. In the centralized case the trajectory optimization problem minimizes the sum of all agents' objective functions J_i while satisfying constraints on the states and inputs. The state $\mathbf{x}_i$ has to belong to the set $\mathcal{X}_i$ which usually represents the system dynamics and boundaries on position, speed and acceleration. Furthermore it also includes the initial and final conditions for the states. Similarly, the input $\mathbf{u}_i$ has to belong to the set $\mathcal{U}_i$. This set usually represents boundaries on the inputs caused by actuator limitations. The number of agents is represented by M. In the following we represent the current agent using i and the neighbouring agent using j, which is a member of the set $\mathcal{N}_i$. Interaction among neighbouring agents is defined by the equality constraint $g_{ij}(\cdot) = 0$. Finding the desired trajectory can be achieved by solving the optimization problem as follows:

$$\begin{aligned} \underset{\forall i: \mathbf{x}_i(\cdot), \mathbf{u}_i(\cdot)}{\text{minimize}} \quad & \sum_{i=1}^{M} J_i(\mathbf{x}_i, \mathbf{u}_i) \\ \text{subject to} \quad & \mathbf{x}_i(k) \in \mathcal{X}_i \quad \forall i \\ & \mathbf{u}_i(k) \in \mathcal{U}_i \quad \forall i \\ & g_{ij}(\mathbf{x}_i(k), \mathbf{x}_j(k)) = 0 \quad \forall i, \forall j \in \mathcal{N}_i \\ & k = 0, 1, ..., N \end{aligned} \tag{1}$$

where k represents the discrete time instance up to N.

The state vector $\mathbf{x}$ of the i-th vehicle is composed of position values $p \in \mathbb{R}^n$ and velocity values $v \in \mathbb{R}^n$. As such, the system dynamics can be described by using the equations (2):

$$\begin{aligned} p(k+1) &= p(k) + T_s v(k) \quad \forall k = 0, 1, ..., N \\ v(k+1) &= v(k) + T_s a(k) \quad \forall k = 0, 1, ..., N \end{aligned} \tag{2}$$

where T_s represents the sampling time and $a \in \mathbb{R}^n$ denotes the acceleration.

Collision avoidance between agents and obstacles is achieved by using the separating hyperplane theorem, presented in [14]. If two nonempty sets C and D are disjoint and convex, then there exists a hyperplane $\{x | a_n^T x = b\}$ which separates the two sets. In other words, $a_n^T x \geq b$ on one set and $a_n^T x \leq b$ on the other. Given a rectangle with $\iota = 4$ corners, the collision avoidance constraint at time instance k assuming a vehicle with radius r_{veh} becomes (3):

$$\begin{aligned} a_n(k)^T \nu_\iota(k) - b(k) &\geq 0, \quad \iota = 1, ..., 4 \\ a_n(k)^T y(k) - b(k) &\leq -r_{veh} \\ ||a_n(k)||_2 &\leq 1 \\ \forall k &\in [0, N] \end{aligned} \tag{3}$$

where $a_n(k)$ and $b(k)$ are the normal vector and the offset for the hyperplane respectively, $y(k)$ represents the motion trajectory and $\nu_\iota(k)$ are the individual corners of the ι-th rectangle.

The formation constraint for each vehicle is expressed with respect to the fixed coordinate system. The constraint can be expressed as follows:

$$g_{ij}(\mathbf{x}_i(k), \mathbf{x}_j(k)) = \mathbf{x}_i(k) - \mathbf{x}_j(k) - \Delta\mathbf{x}_{ij} \ \ \forall k = 0, 1, ..., N \tag{4}$$

where $\Delta\mathbf{x}_{ij}$ is a constant, describing a user-defined relative position between the two agents.

IV. Distributed problem formulation

ADMM [5] is a first-order optimization method that combines the advantages of two distinctive algorithms. It inherits decomposability from the dual ascent and the good convergence property from the method of multipliers. It solves problems of the form:

$$\begin{aligned} \underset{x,z}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & x - z = 0 \end{aligned} \tag{5}$$

where f and g are two objectives, separated by using the variable z which is a duplicate of the original decision variable x.

Based on (5) the augmented Lagrangian can be formulated:

$$\mathcal{L}_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^T(x - z) + \frac{\rho}{2}||x - z||_2^2 \tag{6}$$

where λ is the dual variable and $\rho > 0$ is the parameter of the penalty term that alters the convergence speed. ADMM is a minimization scheme which solves the augmented Lagrangian in an alternating fashion. First, $\mathcal{L}_\rho$ is minimized with respect to x, then the updated value x^+ is used to minimize $\mathcal{L}_\rho$ with respect to z. Finally, a gradient step is performed to update the dual variable λ.

Using the above method the centralized optimization problem of (1) can be distributed among the agents. Analogous to [15] we duplicate the state trajectories in order to decouple the only coupling constraint $g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ between the agents. As such the duplicate of $\mathbf{x}_i$ becomes $\mathbf{z}_i$. The duplicate of the j-th agent's state in the eyes of the i-th agent becomes $z_{i \to j}$. This value can be considered as a proposal from agent i to agent j. Using these notations the problem formulation can be expressed as:

$$\begin{aligned}
\underset{\forall i:\mathbf{x}_i(\cdot),\mathbf{u}_i(\cdot)}{\text{minimize}} \quad & \sum_{i=1}^{M} J_i(\mathbf{x}_i,\mathbf{u}_i) \\
\text{subject to} \quad & \mathbf{x}_i(k)\in\mathcal{X}_i \quad \forall i \\
& \mathbf{u}_i(k)\in\mathcal{U}_i \quad \forall i \\
& g_{ij}(\mathbf{z}_i(k),\mathbf{z}_{i\to j}(k))=0 \quad \forall i,\forall j\in\mathcal{N}_i \\
& \mathbf{x}_i=\mathbf{z}_i \quad \forall i \\
& \mathbf{x}_{ij}=\mathbf{z}_{i\to j} \quad \forall j\in\mathcal{N}_i \\
& k=0,1,...,N
\end{aligned} \tag{7}$$

Based on (7) the augmented Lagrangian can be formulated as:

$$\begin{aligned}
\mathcal{L}_\rho &= \sum_{i=1}^{M}\Big(J_i(x_i)+\lambda_i^T(x_i-z_i)+\frac{\rho}{2}||x_i-z_i||_2^2 \\
&\qquad \sum_{j\in\mathcal{N}_i}(\lambda_{i\to j}^T(x_j-z_{i\to j})+\frac{\rho}{2}||x_j-z_{i\to j}||_2^2)\Big) \\
&= \sum_{i=1}^{M}\Big(\mathcal{L}_{\rho,i}(x_i,z_i,\lambda_i)+\sum_{j\in\mathcal{N}_i}\mathcal{L}_{\rho,i\to j}(x_j,z_{i\to j},\lambda_{i\to j})\Big) \\
&= \sum_{i=1}^{M}\Big(\mathcal{L}_{\rho,i}(x_i,z_i,\lambda_i)+\sum_{j\in\mathcal{N}_i}\mathcal{L}_{\rho,j\to i}(x_i,z_{j\to i},\lambda_{j\to i})\Big)
\end{aligned} \tag{8}$$

In the last line of (8), the indices i and j are flipped. This change is appropriate if bi-directional interaction between the agents is assumed ($j\in\mathcal{N}_i \Leftrightarrow i\in\mathcal{N}_j$) [8].

To find the optimum, gradient ascent is employed to maximize the dual problem (9):

$$q(\lambda_i,\lambda_{i\to j}) = \inf_{\substack{\forall i:\ x\in\mathcal{X}_i,u\in\mathcal{U}_i \\ \forall i,\forall j\in\mathcal{N}_i:\ g_{ij}(z_i,z_{i\to j})=0}} \mathcal{L}_\rho \tag{9}$$

The variable x_i and the variables z_i, $z_{i\to j}$ are updated in two consecutive steps. Then, using the updated values x_i^+, z_i^+, $z_{i\to j}^+$ the gradient becomes (10). This is followed by a gradient step, updating the dual variables λ_i and $\lambda_{i\to j}$.

$$\begin{aligned}
\nabla_{\lambda_i} q &= x_i^+ - z_i^+ \\
\nabla_{\lambda_{i\to j}} q &= x_j^+ - z_{i\to j}^+
\end{aligned} \tag{10}$$

V. Path homotopy

Homotopy classes of trajectories were investigated in previous research [16]–[18] to plan trajectories for a single agent, while this paper extends their application to formation planning and control. Different trajectories belong to the same homotopy class if they can be smoothly transformed by bending and stretching without colliding with any obstacle. For 2D cases [16] uses complex analysis and the Cauchy Integral Theorem.

Trajectories with different homotopy classes may arise in the presence of obstacles. Two trajectories τ_1 and τ_2 connecting the same start points z_s and end points z_f are homotopic iff one can be smoothly deformed into the other without intersecting obstacles [16]. The paths that are homotopic to each other constitute one homotopic class.

With the help of homotopic classes one can decide whether the communication between the agents can be retained throughout the travel. The computation of these classes is difficult however. Instead, [18] suggests comparing the homology of the trajectories in question. This is easier to calculate and in most robotic applications homotopy and homology can be assumed to be equivalent. As defined in [17], two trajectories τ_1 and τ_2 connecting the same start and end points are homologous iff τ_1 together with τ_2 form the complete boundary of a two-dimensional manifold embedded in the configuration space and not containing or intersecting any obstacle.

In [17] a functional $\mathcal{H}(\tau)$ is defined (termed the H-signature of the trajectory), which can be used to differentiate separate homology classes. It assigns a unique complex number to various paths and upon comparing the number, a path's homology class can be determined.

The H-signature is defined as an integral for a continuous path τ starting at $\tau(t=0)=z_s$ and ending at $\tau(t=T)=z_f$ [17]:

$$\mathcal{H}(\tau)=\int_\tau \mathcal{F}(z)dz \tag{11}$$

where $\mathcal{F}(z)$ denotes the obstacle marker function, defined as:

$$\mathcal{F}(z)=\frac{f_0(z)}{(z-\xi_1)(z-\xi_2)...(z-\xi_{N_{obst}})} \tag{12}$$

where $\xi_l\in\mathbb{Z}$ is an arbitrarily chosen point from inside an obstacle $\mathcal{O}_l, \forall l=1,...,N_{obst}$. In the current work f_0 was chosen to equal $(z-BL)^2(z-TR)^2$, following the suggestion in [16], where BL and TR are the complex representation of the bottom-left and top-right points of the environment.

VI. Proposed control method

In this section we propose some important modifications to ADMM to achieve smooth and flexible adaptation of formation in dynamic environments. The path homotopy idea is also implemented to reflect the situation of losing communication in travelling and help adjust formation objective and constraint to achieve robust trajectory planning in the splitting and merging cases.

A. New formation constraint

While the formation constraint presented in (4) is a straightforward choice to retain the desired formation, it doesn't leave enough flexibility for the formation to change its size. To this end we propose a constraint that enforces the group to retain the shape of the formation and also have the flexibility to grow or contract. To achieve this, instead of restricting the relative position of the position vectors, we restrict their crossproduct with the constant Δx_{ij}, which describes a user-defined relative position between two agents. This change essentially requires these vectors to point into the same direction. The updated constraint can be seen in (13):

$$g_{ij}(\mathbf{z}_i(k),\mathbf{z}_{i\to j}(k))=(\mathbf{z}_i(k)-\mathbf{z}_{i\to j}(k))\times\Delta\mathbf{x}_{ij} \quad \forall j\in\mathcal{N}_i \tag{13}$$

To balance the effect of the modified formation constraint, an extra cost function $\hat{J}_i$ is added to the optimization problem in Step 3 of Algorithm 1.

$$\hat{J}_i(\mathbf{z}_i,\mathbf{z}_{i\to j})=\sum_{j\in\mathcal{N}_i}\gamma_z(\mathbf{z}_i(k)-\mathbf{z}_{i\to j}(k)-\Delta\mathbf{x}_{ij}) \tag{14}$$

where γ_z is a predefined constant. Equation (14) effectively turns the original formation constraint of (3) into a soft constraint. Equation (13) in conjunction with (14) allows the group to keep the shape of the desired formation and also allows for smooth growing or contracting when fitting through tight corridors or opening up to avoid small obstacles. During experiments, the neighbourhood set $\mathcal{N}_i$ is set to be the four closest neighbours that communicate with agent i.

B. Individual objectives

The individual objective of an agent is designed as (15):

$$J_i(\mathbf{u}_i) = \sum_{k=1}^{N} \eta \cdot u_i^2(k) \cdot k^2 \tag{15}$$

where $\eta > 0$ is a constant, u_i is the input. The input is multiplied by k^2 so that the system is encouraged to arrive to the destination as soon as possible.

C. Modified ADMM algorithm

Algorithm 1 Distributed trajectory optimization

update $\mathcal{N}_i$ and **initialize** $\lambda_i, z_i, \{\lambda_{i\to j}, \lambda_{j\to i}, z_{j\to i}\}_{j\in\mathcal{N}_i}$

repeat

1. *Prediction*: update x_i

$$\begin{aligned} \underset{x_i,u_i}{\text{argmin}} \quad & J_i(u_i) + \lambda_i^T(x_i - z_i) + \frac{\rho}{2}||x_i - z_i||_2^2 \\ & \sum_{j\in\mathcal{N}_i} (\lambda_{j\to i}^T(x_i - z_{j\to i}) + \frac{\rho}{2}||x_i - z_{j\to i}||_2^2) \\ \text{subject to} \quad & x_i \in \mathcal{X}_i \end{aligned}$$

2. *Communication* with agent j, $\forall j \in \mathcal{N}_i$

$$\begin{aligned} \text{send} \quad & \to x_i^+ \\ \text{receive} \quad & \leftarrow \{x_j^+\}_{\forall j\in\mathcal{N}_i} \end{aligned}$$

3. *Coordination*: updating z_i, $z_{i\to j}$

$$\begin{aligned} \underset{z_i,z_{i\to j}}{\text{argmin}} \quad & \hat{J}_i(z_i, z_{i\to j}) + \lambda_i^T(x_i - z_i) + \frac{\rho}{2}||x_i - z_i||_2^2 \\ & \sum_{j\in\mathcal{N}_i} (\lambda_{i\to j}^T(x_j - z_{i\to j}) + \frac{\rho}{2}||x_j - z_{i\to j}||_2^2) \\ \text{subject to} \quad & g_{ij}(z_i, z_{i\to j}) = 0 \quad \forall j \in \mathcal{N}_i \end{aligned}$$

4. *Dual variable update*: computing λ_i, $\lambda_{i\to j}$

$$\lambda_i^+ = \lambda_i + \rho(x_i - z_i)$$
$$\lambda_{i\to j}^+ = \lambda_{i\to j} + \rho(x_j - z_{i\to j}) \quad \forall j \in \mathcal{N}_i$$

5. *Communication* with agent j, $\forall j \in \mathcal{N}_i$

$$\begin{aligned} \text{send} \quad & \to (\lambda_{i\to j}^+, z_{i\to j}^+) \\ \text{receive} \quad & \leftarrow \{\lambda_{j\to i}^+, z_{j\to i}^+\}_{\forall j\in\mathcal{N}_i} \end{aligned}$$

until convergence

The optimization problem (7) can be solved in a distributed fashion if all agents perform a modified ADMM algorithm, as described in Algorithm 1. In the first, *prediction* step of Algorithm 1 each agent plans a collision-free trajectory that obeys its system dynamics, minimizes its objective and drives the agent to the goal position. The objective function driving each vehicle to its destination was described in (15). In the second, *communication* step the agents exchange their updated trajectories which will be used in the next optimization step. In the third, *coordination* step the duplicate variables z_i and $z_{i\to j}$ are updated. These values are chosen so, that they stay close to the originally proposed trajectories. This is followed by the *update* of the dual variables λ_i and $\lambda_{i\to j}$. Finally, the updated dual variables $\lambda_{i\to j}^+$ and suggested trajectories $z_{i\to j}^+$ are *communicated* to all agents j, $j \in \mathcal{N}_i$.

D. Implementation of path homotopy

The homotopy classification of trajectories from different vehicles is not self-evident, because naturally the conditions of starting from the same position and ending in the same position cannot be met. For this reason, in this work we propose virtually connecting the starting point of each vehicle's trajectory to the current center of the formation. Similarly, its ending point should be connected by a straight line to the desired formation center at the ending position. As such, the conditions for comparing homotopy classes of trajectories are met.

E. Proposed DMPC scheme for formation control

To coordinate the agents to their desired destination, a distributed MPC scheme is used. The strategy uses Algorithm 1 in every control step to calculate the optimal trajectories. However, letting the optimization algorithm run until convergence would mean too high of a computational load for the system. For this reason, only a small number of ADMM iterations are executed every control step and the time-shifted results of the previous iteration are used to warm-start the optimizer in the next control step.

The summary of the proposed control scheme can be found in Algorithm 2. First, n optimization iterations are performed to initialize the relevant variables. Thereafter starts the control loop which is repeated until the agents arrive to their destination. Each agent follows the currently prescribed trajectory. Meanwhile, the future position is estimated and a time-shifted version of the previously calculated trajectory is used to warm-start the optimizer. The optimization loop is performed to update the variables. Each agent then calculates the path homotopy of its trajectory and compares them with the visible agents. If the difference in values is less than a predefined value ϵ, the given agents will form the same group. The value of ϵ should be tailored to the specific implementation, but in general, it may be a fairly large value, because the H-signature of paths that are not in the same homotopy class tends to give significantly different results. If a new group is formed, the agents receive updated final states and formation constraints from a higher level controller and continue to make progress towards the goal.

VII. Experimental results

The proposed DMPC strategy is an online algorithm capable of scaling, splitting and merging formation. In this example

Algorithm 2 Control algorithm for agent i

Perform Algorithm 1 for n iterations, obtain:
$x_i^0, x_j^0, z_i^0, z_{i\to j}^0, \lambda_i^0, \lambda_{i\to j}^0$
repeat every ΔT until destination reached
  follow trajectory x_i^k
  estimate future position $\hat{x}_i(k+1)$
  transform $x_i^k, x_j^k, z_i^k, z_{i\to j}^k, \lambda_i^k, \lambda_{i\to j}^k$ to warm
    start the optimizer
  perform Algorithm 1 for m iterations
  calculate the center of the group
  calculate $\mathcal{H}(x_i)$ using equation (11)
  communicate $\mathcal{H}(x_i)$ with all visible agent $j_{visible}$
  form a group with all $\{j \mid \ |\mathcal{H}(x_i) - \mathcal{H}(x_j)| < \epsilon\}$
  if no change in j
    continue
  else
    update final destination and formation constraint
  end if
end repeat

twelve two-dimensional holonomic vehicles receive a task to travel in a circular formation. On their way they have to overcome challenges like a narrow corridor and a dynamic obstacle moving towards the group. The simulation was implemented in Matlab using CasADi [19] to formulate the optimization problem and to interface the optimizer IPOPT [20]. The host platform was a Windows PC with an Intel i7-8550H CPU @2.6 Ghz.

Figure 1 shows the agents agreeing on paths that squeeze the formation through a narrow corridor. Furthermore, the system can avoid dynamic obstacles by being able to predict the obstacles' future position, therefore the planned trajectories open up for the obstacle coming from the right. The planned trajectories of the groups are represented with coloured curves and the already travelled paths are shown in grey. The trajectories with the same color belong to the agents of the same group. Figure 2 shows the path homotopy algorithm organizing the trajectories in two separate classes, indicated by the green and purple colors. The agents then form two new groups and receive updated final destinations and updated formation constraints, to reflect the situation that the communication among agents is to be blocked. Following this, they start to move towards their new final position keeping a smaller circular formation, as shown in Figure 3.

Lastly, after leaving behind the obstacle that blocked the communication, the planned trajectories of the agents again form the same homotopy and the two groups are allowed to merge. The updated final destinations and formation constraints are sent to all agents and the group is able to plan trajectories that will lead to the desired formation after a few iterations.

Separating the agents into groups has important advantages in real-world scenarios. Once the communication between agents is blocked, the agents have no information about the agents on the other side of the obstacle. The non-visible agents may follow their trajectories as planned but they might also get stuck or have to take different routes. In a dynamic

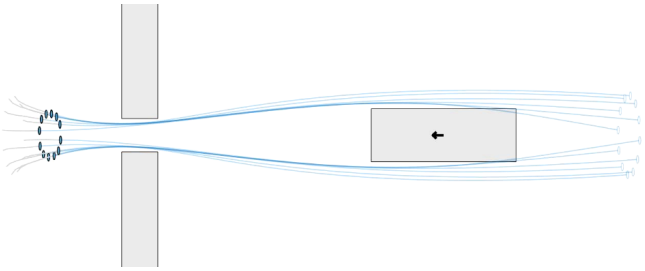

Fig. 1: Squeezing through a narrow corridor

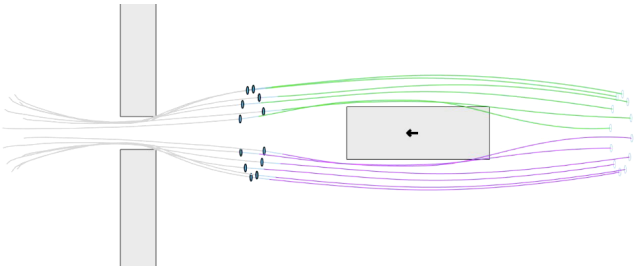

Fig. 2: Trajectories with different homotopy classes, triggered by a nearby obstacle

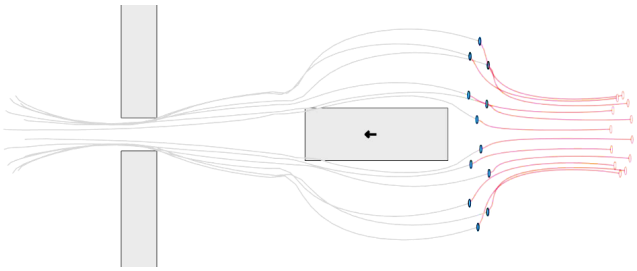

Fig. 3: Two groups keeping circular formation in loss of communication, and merging thereafter

environment there is no guarantee that all the agents can merge in the future. For this reason it is suggested to split the agents into independent groups which perform their individual tasks. The groups can merge when the communication can be re-established.

To evaluate the performance of the algorithm, a fitness value is calculated to measure, how well the formation is kept. In one control update, each agent calculates the angle between two vectors, one pointing from the agent to its neighbour and the other pointing from the agent to the neighbour's desired position. This angle is calculated by each agent for each of its neighbours at the current time. The sum of these angle values becomes the fitness value for the current control loop.

The performance of the proposed control strategy is greatly affected by the warm-starting mechanism in optimization and some pivotal parameters, such as the number of optimization iterations per control update m and ρ in Algorithm 1. We have carried out a set of experiments to study their effectiveness and/or settings, which are introduced in the following paragraphs.

Table I shows how m affects the overall performance and the computational time. We can see a clear correlation between

TABLE I: The Effects of the Number of Optimization Iterations per Control Update and the Warm-Starting Mechanism

no. of ADMM iterations	1	2	3	5	7	10	$\hat{1}$
Overall fitness	912	802	787	775	766	747	1109
Time per control update (s)	0.5	1.7	2.6	4.7	6.2	8.5	1.4

* $\hat{1}$ stands for the case without optimizer warm-starting.

TABLE II: The Effect of ρ on Simulation Performance

ρ value	0.4	0.6	0.8	1	1.5	2
Overall fitness	1131	1009	977	912	873	836
Time to reach dest. (s)	45	48	51	53	60	65
Required control effort	132	131	128	127	121	119

m and the overall fitness value. The table shows, that performing more optimization iterations per control update can be beneficial because it improves on the fitness value. However, the computational burden also increases but to a much greater extent. Based on the above findings, it is suggested to utilize only a few optimization iterations, such as 1 or 2, to ensure a fast but still good performing algorithm. In the table, $\hat{1}$ stands for the case, where the optimizer was not warm started. The computational time and overall fitness in this case has both greatly increased, which verifies the effectiveness of the warm-starting mechanism.

Another important parameter used to alter how closely the formation is kept is ρ in Algorithm 1. This constant influences how strictly the optimization algorithm alters the desired trajectory of an agent to conform to the formation constraint, set by the position of the neighbouring agents. As shown in Table II, by increasing the value of ρ, the fitness value can be improved. This indicates that the agents achieve the desired formation faster, but the time to reach the goal and the overall control effort increase. For this reason, the designer has to consider the benefits and the trade-offs when choosing an appropriate ρ value for a specific application.

VIII. Conclusion

In this work a distributed formation control algorithm was developed that allows agents to avoid dynamic obstacles and adapt the formation according to different environment and obstacles. The novelty of the work lies in achieving scaling, splitting and merging of formation simultaneously, by designing an improved ADMM and applying path homotopy techniques into formation adaptation for the first time. In the proposed method, agents are able to detect obstacles that may prevent inter-agent communication and in such cases they split and form new groups based on different homotopy classes. Similarly, if two groups find themselves in the same homotopy class, they smoothly merge to form a new larger group. Simulation results verified that the proposed method is able to dynamically resize, split and merge the formation in the case involving loss of communication among agents.

References

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 21, no. 4. New York, NY, USA: ACM Press, 1987, pp. 25–34.

[2] R. Cui, S. S. Ge, B. V. E. How, and Y. S. Choo, "Leader–follower formation control of underactuated autonomous underwater vehicles," *Ocean Engineering*, vol. 37, no. 17-18, pp. 1491–1502, Dec. 2010.

[3] D. Zhou, Z. Wang, and M. Schwager, "Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 916–923, Aug. 2018.

[4] L. Dai, Q. Cao, Y. Xia, and Y. Gao, "Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance," *Journal of the Franklin Institute*, vol. 354, no. 4, pp. 2068–2085, Mar. 2017.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[6] Y. Lyu, J. Hu, B. M. Chen, C. Zhao, and Q. Pan, "Multivehicle flocking with collision avoidance via distributed model predictive control," *IEEE Transactions on Cybernetics*, Early Access.

[7] T. H. Summers and J. Lygeros, "Distributed model predictive consensus via the alternating direction method of multipliers," in *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*. Monticello, IL, USA: IEEE, Oct. 2012, pp. 79–84.

[8] R. Van Parys and G. Pipeleers, "Online distributed motion planning for multi-vehicle systems," in *Proceedings of 2016 European Control Conference*. Aalborg, Denmark: IEEE, Jun. 2016, pp. 1580–1585.

[9] ——, "Distributed MPC for multi-vehicle systems moving in formation," *Robotics and Autonomous Systems*, vol. 97, pp. 144–152, Nov. 2017.

[10] J. Juhl, "Splitting and Merging of Quadrotor Teams Flying in Formation," Master's Thesis, Delft University of Technology, Delft, Netherlands, 2017.

[11] H. Zhu, J. Juhl, L. Ferranti, and J. Alonso-Mora, "Distributed multi-robot formation splitting and merging in dynamic environments," in *Proceedings of 2019 International Conference on Robotics and Automation*. Montreal, QC, Canada: IEEE, May 2019, pp. 9080–9086.

[12] H. Vazquez-Leal, A. Marin-Hernandez, Y. Khan, A. Yildirim, U. Filobello-Nino, R. Castaneda-Sheissa, and V. M. Jimenez-Fernandez, "Exploring collision-free path planning by using homotopy continuation methods," *Applied Mathematics and Computation*, vol. 219, no. 14, pp. 7514–7532, Mar. 2013.

[13] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time UAV replanning using guided gradient-based optimization and topological paths," in *Proceedings of 2020 International Conference on Robotics and Automation*. Early Access.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[15] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully decentralized ADMM for coordination and collision avoidance," in *Proceedings of 2018 European Control Conference*. Limassol, Cyprus: IEEE, Jun. 2018, pp. 825–830.

[16] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Proceedings of the National Conference on Artificial Intelligence*. Atlanta, GA, USA: AAAI Press, Jul. 2010, pp. 1230–1237.

[17] S. Bhattacharya, M. Likhachev, and V. Kumar, "Search-based path planning with homotopy class constraints in 3D," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. Toronto, ON, Canada: AAAI Press, Jul. 2012, pp. 2097–2099.

[18] ——, "Identification and representation of homotopy classes of trajectories for search-based path planning in 3D," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, Jun. 2011, pp. 9–16.

[19] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 2013.

[20] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006.