Online B-spline based trajectory planning for swarm of agents using distributed model predictive control

Cong Khanh Dinh, Ionela Prodan, Florin Stoican

Online B-spline based trajectory planning for swarm of agents using distributed model predictive control

Cong Khanh Dinh[1], Ionela Prodan[1] and Florin Stoican[2]

***Abstract*— This paper deals with the motion planing in real-time for swarm of agents using distributed Model Predictive Control (DMPC). Optimization-based control helps both to stabilize the motion dynamics of the robots and to enforce the system's constraints. However, the price is a significant increase in complexity as the scale of the system surges. To improve the scalability for a large formation, we propose a DMPC framework that considers B-spline parameterizations for the agents' trajectories. The proposed approach has great promise for multi-drones system, as illustrated with simulation examples using a state of the art quadcopter model.**

***Index Terms*– Motion planning, Collision avoidance, B-splines parametrization, Multi-agent dynamical systems, Distributed Model Predictive Control (DMPC).**

I. Introduction

Generating trajectories for a group of dynamic agents is a challenging problem. This issue is becoming increasingly prominent in today's world as advancements in sensor, network, and information technologies raise a substantial need for effectively managing a huge number of interconnected systems. Examples include formation control of swarms of UAVs to perform cooperative tasks [1], coordination of multi-robots in warehouses [2], [3]. Among these, smart navigation of multiple UAVs, inspired by the biological behavior of swarms, has recently received extensive attention from both academics and professionals [4], [5]. Owing to their flexibility and autonomy, they have widely been applied in various civil and military fields, such as 3D inspection of constructions [6], [7], [8], collecting data in agriculture [9], [10], search and rescue mission in danger zones [11].

One of the primary difficulties is handling the associated constraints among several robots simultaneously [12], [13]. This is not readily accomplished by a centralized method where all data is gathered by a single central processor which sends the control decisions for the whole network. Since the number of agents rises, the solution time quickly becomes impractical. Nonetheless, the decentralized control appears to address the issue of speed by integrating parallel processing units to reduce computation time [14], [15]. However, since there is no communication, the constraints are not guaranteed. As a compromise, the distributed controller proves to be of great use as it can ensure scalability over a large number of agents in addition to guaranteeing the constraints.

Our goal in this paper is to design a reliable distributed trajectory generation algorithm that can handle multiple UAVs on a transition task while avoiding collisions and respecting operating constraints.

It is fundamental that the on-line trajectory generation mechanism takes into account collision avoidance conditions. The key for real-time planning is to execute the tasks quickly in dynamic and shared environment. A wide variety of techniques are currently proposed to deal with the target tracking problem with collision avoidance. First, direct methods that generate trajectories with no collision for point-to-point flying scenarios are thoroughly studied and come with many complete general solutions. One classical approach uses Mixed Integer Programming (MIP), where non-convex operation space are modeled through binary variables [16], [17]. This method is difficult to solve in real time due to high computational effort, hence not suitable for a large group of agents. Second, artificial potential field technique [18] assumes a repulsive and attractive field exists around the bodies so that it creates forces that prevent collision. In addition, control barrier functions (CBFs) are frequently employed in safety-critical scenarios to enhance safety by enabling robots to make decisions early, thereby avoiding getting too near to collisions [19], [20], [21]. Subsequently, learning-based methods [22], [23] are increasingly employed in solving trajectory planning problems that take into account interactions among robots. To generate appropriate behaviors, however, a significant amount of high-quality training data is needed, and this is difficult to generalize for different system sizes. Sometimes all the above methods are included in a distributed setting of Model Predictive Control (MPC) [24], known for its ability to handle complicated constraints.

Furthermore, diverse geometrical aspects are taken into account by numerous discrete and continuous planning strategies when implementing the collision algorithm for multi-agent systems in real time. The On-demand(OD) collision avoidance replans the trajectory when a future collision is detected as proposed in [25]. This very fast planning strategy for multiple quadcopters task assignment with high success rate is compared to the well-studied Buffered Voronoi Cell method [26]. Another method is to parameterize the control flat output describing the agent model with multiple polynomials concatenated [25], [27] using Bézier representation. It

[1]C. K. Dinh and I. Prodan are with Univ. Grenoble Alpes, Grenoble INP†, LCIS, F-26000, Valence, France, (email:{cong-khanh.dinh, ionela.prodan}@lcis.grenoble-inp.fr), † Institute of Engineering and Management Univ. Grenoble Alpes.

[2]F. Stoican is with the Faculty of Automation Control and Computer Science, University Politehnica of Bucharest, Romania, (e-mail: florin.stoican@upb.ro).

The work of the first two authors is supported by La Région Auvergne-Rhône-Alpes, Pack Ambition Recherche 2021 - PlanMAV, RECPLAMAL-CIR and Ambition Internationale 2023, Horizon-TA C7H-REG24A10, France. The third author is supported by a grant from the National Program for Research of the National Association of Technical Universities - GNAC ARUT 2023; Project ID: 207, UNSTPB, Romania.

has the advantage of smoothness of the trajectory and that the constraints of input and its derivatives can be imposed with ease using the control points describing the Bézier curve.

Leveraging the idea of control point space mapping, we build upon the properties of the B-splines [28] and distributed MPC for collision and constraints handling. Hence, in this paper, we propose the following contributions:

i) a novel distributed algorithm that addresses the need for real-time trajectory generation for multiple agents systems while maintaining a balance between the rapidity and reliability of decentralized and centralized approaches;
ii) B-spline parameterization for distributed MPC that offers enhanced control over the decision variable and decreases the complexity associated with enforcing continuity constraints on the optimization problem, compared to conventional power basis functions (e.g., polynomials);
iii) a fast collision avoidance formulation based on Buffered Voronoi Cell (BVC) that can be possibly implemented online using the B-spline framework.

The remain of this paper is organized as follows. In section II, we introduce the model and theories about B-spline and Buffered Voronoi Cell (BVC). Section III provides the distributed algorithm under the MPC framework. We present several simulation results and analysis in Section IV followed by conclusions in Section V.

Notation

Vectors are represented by bold letters. Capital letters in bold represent the matrices. $\mathbf{I}_n$ and $\mathbf{0}_n$ are identity and zeros matrix with dimension $n \times n$. $||x||_{\mathbf{Q}} \triangleq \sqrt{\mathbf{x}^\intercal \mathbf{Q} \mathbf{x}}$. Otherwise, if not specified in the subscript, $\| \blacksquare \|$ represents the Euclidean norm.

With notation $\hat{(\blacksquare)}[k|k_t]$ denoting the predicted value of $(\blacksquare)$ at time step $k + k_t$ from information known at time k_t, an agent's prediction model used inside the MPC problem is $\hat{\mathbf{x}}_i[k+1|k_t] = \mathbf{A}\hat{\mathbf{x}}_i[k|k_t] + \mathbf{B}\hat{\mathbf{u}}_i[k|k_t]$, where $\hat{\mathbf{x}}_i[0|k_t] \leftarrow \mathbf{x}_i[k_t]$ denotes the vector of measured states.

II. Preliminaries

A. Agents dynamics

While the approach followed in the rest of the paper works for arbitrary dynamics, for illustrations purposes, hereinafter the agent dynamics are those of a quadcopter. Such a model exhibits strong nonlinearities which are canceled[1] through a flatness-based linearization as in [29]. After a discretization step, we arrive at the decoupled stacking of three double integrators:

$$\mathbf{x}_i[k+1] = \mathbf{A}\mathbf{x}_i[k] + \mathbf{B}\mathbf{u}_i[k], \tag{1}$$

where $\mathbf{A} = \begin{bmatrix} \mathbf{I}_3 & h\mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} h^2\mathbf{I}_3/2 \\ h\mathbf{I}_3 \end{bmatrix}$ and h is the discretization time. The state of the i^{th} agent $\mathbf{x}_i[k]$ is defined by the position and velocity of the vehicle, i.e $\mathbf{x}_i[k] = (\mathbf{p}_i^\intercal[k], \mathbf{v}_i^\intercal[k])^\top \in \mathbb{R}^6$, the control input $\mathbf{u}_i[k] = \mathbf{a}_i[k] \in \mathbb{R}^3$ is the acceleration of the vehicle. In case of different agent nonlinear dynamics, we can obtain its linear model either by feedback linearization or Taylor approximation.

[1]The price of linearizing the dynamics in the flat output space is the strongly nonlinear coupling of the input constraints. See [29] for further details.

B. B-spline definition and properties

The use of B-splines for trajectory generation has became popular due to the many properties among which, continuity and locality ensure smooth curve generation and fast reconfiguration, [28]. Convexity and end point interpolation also play important roles in constraint validation. Specifically, a family of B-spline basis functions $\{B_{i,p}(t)\}$ with $i \in \{0, \dots N\}$ characterized by cardinality N, order p and knot-vector $\tau = \{\tau_1, \dots, \tau_m\}$ weighted by control points $\{P_i\}$ gives the curve

$$\mathbf{z}(t) = \sum_{i=0}^{N} P_i B_{i,p}(t) = \mathbf{P}\mathbf{B}_p(t), \quad \forall t \in [\tau_1, \tau_m]. \tag{2}$$

In (2), the column vector $\mathbf{B}_p(t)$ and control matrix $\mathbf{P}$ come from stacking the B-spline basis functions $\{B_i(t)\}$ vertically and, respectively, the control points $\{P_i\}$ as columns.

Considering the dynamics (1), $\mathbf{z}(t) \in \mathbb{R}^{3\times 1}$ denotes the reference position which has to be tracked by the position component of the state, i.e., $Cx_i[k] = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix} x_i[k]$.

Without detailing all the steps (see [28] for further details) note that derivatives of (2) may be written as B-spline curves of lower order (order $p - r$ for the r-th order derivative of $\mathbf{z}^{(r)}(t)$) weighted by new control points which depend linearly on the original ones. For example, we may write

$$\mathbf{z}^{(2)}(t) = \mathbf{P}^{(2)}\mathbf{B}_{p-2}(t), \tag{3}$$

where $\mathbf{P}^{(2)} := \mathbf{P}\mathbf{M}_2$ and $\mathbf{M}_2$ is a matrix suitably computed (see [30]).

C. Buffered Voronoi Cell partitioning

Voronoi decomposition is often used in collision-free path finding for robotics [31]. By partitioning the space into regions based on proximity to given points (the "centers"), Voronoi diagrams help create a road map for motion planning. It is particularly beneficial for scenarios where adaptability to a changing environments is crucial.

We recall here some key definitions of Buffered Voronoi Cell (BVC) which allow accounting for physical dimensions of the agent, [26].

Definition 1 ([31]): For a group of N_a agents with position $\{\mathbf{p}_i\}_{i\in N_a} \in \mathbb{R}^3$, the general Voronoi cell for i-th agent is defined as:

$$\mathcal{V}_i = \{\mathbf{p} \in \mathbb{R}^n \mid\mid \| \mathbf{p} - \mathbf{p}_i \| \leqslant \| \mathbf{p} - \mathbf{p}_j \|, \forall j \neq i\} \tag{4}$$

Equivalently, (4) can be rewritten as

$$\mathcal{V}_i = \{\mathbf{p} \in \mathbb{R}^n \mid \left(\mathbf{p} - \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\right)^\top \mathbf{d}_{ij} \leqslant 0, \forall j \neq i\} \tag{5}$$

where $\mathbf{d}_{ij} = \mathbf{p}_j - \mathbf{p}_i$.

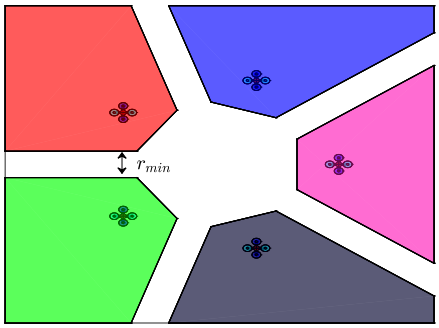

Fig. 1: Buffered Voronoi Cell in 2-D plane

Furthermore, to incorporate a safety distance between agents, denoted by r_{min} (e.g., corresponding to their physical sizes), the notion of BVC is employed.

Definition 2 (Buffered Voronoi Cell): For groups of N_a agents with position $\{\mathbf{p}_i\}_{i\in N_a} \in \mathbb{R}^3$, the Buffered Voronoi Cell (BVC) of the i-th agent is defined as:

$$\bar{\mathcal{V}}_i = \{\mathbf{p} \in \mathbb{R}^3 \mid \left(\mathbf{p} - \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\right)^\top \mathbf{d}_{ij} + \frac{r_{min}}{2}||\mathbf{d}_{ij}|| \leqslant 0, \forall j \neq i\}. \quad (6)$$

Note that in (6), the agents are confined to stay within their individual Buffered Voronoi Cell $\bar{\mathcal{V}}_i$.

We utilize a BVC based on the definition provided in reference [26], enhanced by the incorporation of the scaling matrix $\Theta \in \mathbb{R}^{3\times 3}$ as described in [25].

$$\bar{\mathcal{V}}_i = \{\mathbf{p} \in \mathbb{R}^3 \mid \frac{(\mathbf{p}_i - \mathbf{p}_j)^\top \Theta^{-2}(\mathbf{p} - \mathbf{p}_i)}{||\mathbf{d}_{ij}||} \geqslant \frac{r_{min} - ||\mathbf{d}_{ij}||}{2}, \forall j \neq i\} \quad (7)$$

where $\mathbf{d}_{ij} = \Theta^{-1}(\mathbf{p}_j - \mathbf{p}_i)$, $\mathbf{p}_i$ and $\mathbf{p}_j$ denote the coordinates of agents i and j respectively.

When considering the collision avoidance constraints among the agents using BVC, we can obtain linear matrix inequalities as showed in (7). This will be further integrated in our motion planning algorithm denoted as a constrained quadratic programming problem (see, later on, (9g)).

III. Online distributed architecture for MPC

Due to the inherently high computational complexity of real-time centralized controllers' implementation, we propose in this section a distributed MPC architecture which accounts for collision avoidance.

A. Distributed MPC optimization

Note that, in the control loop, the optimization problems of every agent are solved in parallel using the preceding prediction data broadcast among the agents. The following cost function is minimized:

$$\min_{\mathbf{P}_i, \{\Phi_{i,j}\}_{j\in 1,\ldots,N_a}} \sum_{s=0}^{N_p-1} \underbrace{\| \hat{\mathbf{p}}_i[s|k] - \mathbf{p}_i^f \|_{\mathbf{Q}_p}}_{\text{Tracking error cost}} \quad (8a)$$

$$+ \underbrace{\| \hat{\mathbf{u}}_i[s|k] \|_{\mathbf{R}}}_{\text{Control effort cost}} \quad (8b)$$

$$+ \underbrace{||\hat{\mathbf{p}}_i[N_p|k] - \mathbf{p}_i^f||_{\mathbf{P}}}_{\text{Terminal cost}} \quad (8c)$$

$$+ \underbrace{\sum_{j=1}^{N_a} \left(\eta_2 \| \Phi_{i,j} \|^2 - \eta_1 \Phi_{i,j}\right)}_{\text{Augmented cost in collision avoidance case}}, \quad (8d)$$

such that the following constraints are respected:

$$\hat{\mathbf{u}}_i[s|k] = \mathbf{P}_i^{(2)} \mathbf{B}_{p-2}(t_s), \quad (9a)$$

$$\hat{\mathbf{v}}_i[s|k] = \mathbf{P}_i^{(1)} \mathbf{B}_{p-1}(t_s), \quad (9b)$$

$$\hat{\mathbf{p}}_i[s|k] = \mathbf{P}_i \mathbf{B}_p(t_s) \quad (9c)$$

$$u_{min} \leqslant \hat{\mathbf{u}}_i[s|k] \leqslant u_{max}, \quad (9d)$$

$$v_{min} \leqslant \hat{\mathbf{v}}_i[s|k] \leqslant v_{max}, \quad (9e)$$

$$p_{min} \leqslant \hat{\mathbf{p}}_i[s|k] \leqslant p_{max} \quad (9f)$$

$$A_{coll}\hat{\mathbf{p}}_i[\ell+1|k] + \frac{||\mathbf{d}_{ij}^\ell||}{2}\Phi_{i,j} \leqslant b_{coll} \quad (9g)$$

where:

- ℓ is the nearest prediction step where the collision occurs;
- $||\mathbf{d}_{ij}^\ell||$ is the distance between agents i, j at time step ℓ;
- (A_{coll}, b_{coll}) denote the inequality constraint tuple considered for collision avoidance;
- $\Phi_{i,j}$ is the penalty term for the soft constraint: $\begin{cases} \Phi_{i,j} = 0 & \text{if } j \notin \mathcal{W}_i^\ell; \\ \Phi_{i,j} < 0 & \text{if } j \in \mathcal{W}_i^\ell; \end{cases}$
- $\mathcal{W}_i^\ell$ denotes the set of neighbours of agent i when a collision is detected at time step ℓ.

In the above optimization problem, several terms are incorporated into the cost function (8) to address different aspects of the problem and achieve the desired objectives:

i) (8a) aims to penalize the tracking error. This is particularly relevant in target tracking scenarios, where the drones need to closely follow a desired trajectory or maintain proximity to a moving target. By minimizing the tracking error, the optimization problem ensures accurate and precise tracking performance.

ii) (8b), known as the control effort cost, is introduced to encourage energy efficiency. By minimizing this term, the optimization problem forces the drones to use their control inputs optimally, thus reducing unnecessary energy consumption. This leads to improved energy efficiency and longer flight times.

iii) (8c) serves to penalize deviations from stability in the final position. By including this term, the optimization problem tends to steer the drones to reach and maintain the desired final position with greater stability. This is

particularly useful when it is essential for the drones to remain stationary or maintain a specific pose at the end of their trajectory.

To enforce collision avoidance, the optimization problem includes linear constraints. If over the prediction horizon N_p, the agent i-th detects a collision at the nearest time step ℓ,

$$||\hat{\mathbf{p}}_i[\ell+1|k] - \hat{\mathbf{p}}_j[\ell+2|k-1])|| < r_{min}$$

we construct a Buffered Voronoi Cell $\bar{\mathcal{V}}_i$ as defined in (7) around the i-th robot with its neighbouring agents $\mathcal{W}_i^\ell$ at time $k+\ell$.

$$\begin{aligned}\bar{\mathcal{V}}_i = \{\mathbf{p} \in \mathbb{R}^3 \mid \\ (\hat{\mathbf{p}}_i[\ell|k] - \hat{\mathbf{p}}_j[\ell+1|k-1])^\top \Theta^{-2} (\hat{\mathbf{p}}_i[\ell+1|k] - \hat{\mathbf{p}}_i[\ell|k]) \\ \geqslant ||\mathbf{d}_{ij}^\ell|| \frac{r_{min} - ||\mathbf{d}_{ij}^\ell||}{2}, \forall j \in \mathcal{W}_i^\ell\} \quad (10)\end{aligned}$$

To deal with infeasibility when resolving the optimization problem, we introduce a relaxation term for collision avoidance $\Phi_{i,j}$ so that

$$||\Theta^{-1}(\hat{\mathbf{p}}_i[\ell+1|k] - \hat{\mathbf{p}}_j[\ell+1|k-1])|| \geqslant r_{min} + \Phi_{i,j} \quad (11)$$

The constraint in (11) can be obtained using BVC formulation as in (10), which lead to the final constraint in the QP problem above (9). The matrices of collision constraints A_{coll} and b_{coll} are defined as:

$$A_{coll} = -(\Theta^{-2}(\hat{\mathbf{p}}_i[\ell|k] - \hat{\mathbf{p}}_j[\ell+1|k-1]))^\top, \quad (12a)$$

$$b_{coll} = A_{coll}\hat{\mathbf{p}}_i[\ell|k] - \frac{(r_{min} - ||\mathbf{d}_{ij}^\ell||)||\mathbf{d}_{ij}^\ell||}{2}. \quad (12b)$$

The relaxation variable $\Phi_{i,j}$ in the collision soft constraint is then minimized in the optimization as an augmented cost term (8d). This additional term assists in handling collision avoidance while reducing the complexity of the Quadratic Programming (QP) problem. By incorporating the augmented cost, the optimization problem adds flexibility, thus ensuring safe and efficient navigation in the presence of potential collisions.

The main decision variables in this QP are the control points $\mathbf{P}_i$ of the B-spline trajectory. By formulating the prediction model solely based on these control points, collision detection and path reconfiguration can be performed in continuous time, enabling faster response and avoidance of collisions. In the real implementation, we may implement a two-layer sampling: large sample time, h, for the MPC problem update, and small sampling time, t_s for the low-level UAV control.

B. Asynchronous algorithm

The goal of this part is to introduce the asynchronous optimization-based distributed scheme collision avoidance strategy. Every agent will broadcast its previously estimated trajectory to all of its neighbor agents at a given discrete time step, k. The information can then be used by others agents to establish an alternative path in the next step(see Fig. 2).

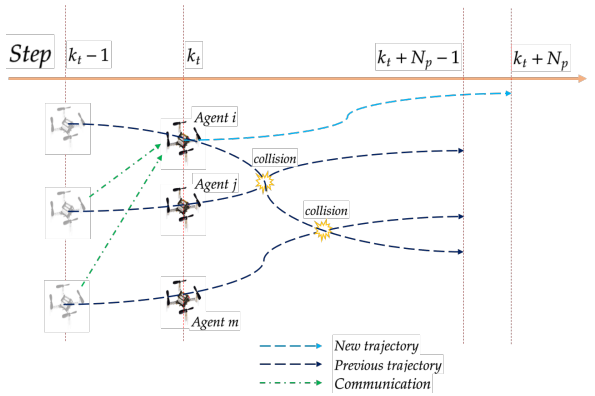

Fig. 2: Distributed MPC strategy for collision avoidance

At discrete time step k, each agent execute the subsequent operations in order to generate a set of B-spline control points that parameterizes the control input:

1) Evaluate potential future collisions by referencing the most recent predicted positions of neighboring objects from the prior time step $k-1$.
2) Formulate an optimization problem, selectively incorporating collision avoidance constraints as needed.
3) Determine the subsequent optimal sequence and apply its initial element to the model.
4) Distribute the anticipated states among the agents.

Algorithm 1 Distributed MPC for target tracking

Require: $\bar{\mathbf{x}}^0$ - initial states, $\mathbf{p}^f$ - final positions $\{\bar{\mathbf{x}}_i^0\}_{i\in\mathcal{S}}, \{\mathbf{p}_i^f\}_{i\in\mathcal{S}}, \mathcal{S} = 1, ..., N_a$
Ensure: Control input trajectories $\{u_i(t)\}_{i\in\mathcal{S}}$

```
k ← 1
while k ⩽ N_sim and not ReachTarget do
    resetTargetLocation(p^f)
    Σ_{k-1} ← BroadcastLastPrediction()
    for all agent i ∈ S = 1, ..., N_a do
        u_i[k] ← ResetInput(x̄_i[k], Σ_{k-1})
        if T_i^l ∩ T_j^l ≠ ∅ then
            W_i^l ← FindNeighbour()
            BuildCollisionConstraint(W_i^l)
            P_i[k], Φ_{i,j}[k] ← SolveDMPC(L̄_i(P_i, {Φ_{i,j}}))
        else
            P_i[k] ←P_main SolveDMPC(L̄_i(P_i))
        end if
        ū_i[k] ← UpdateControlInput(P_i[k])
    end for
    ReachTarget ← CheckArrival(x̄[k], p^f)
    k ← k + 1
end while
```

In Fig. 3, the two UAVs detect collision at step $\ell = 4$ in the future based on the information from previous prediction step $k-1$ from their neighbour. The BVC is thus constructed specifically to the positions of them on one step ahead at $\ell - 1$ and collision constraints are imposed on the collision step ℓ in the planning algorithm. By generating the BVC

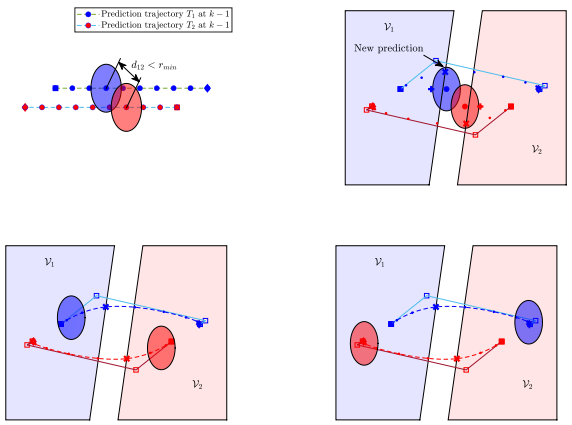

Fig. 3: 2-agent transition scenario in 2D using Algorithm 1. The circles in red and blue represent the safety region of the UAV with radius r_{min}. The diamonds and squares represent initial and final positions respectively. The dashed lines represent the prediction trajectory over the horizon.

at only one step over the prediction horizon with relaxation constraints, the proposed planning strategy becomes less conservative compared to traditional BVC method which reduce significantly computational burden in the optimization-based scheme.

IV. Simulation Scenarios and Performance Analysis

A. Simulation scenarios

To demonstrate the applicability of the proposed distributed algorithm, we show the simulation results for scenarios of multiple UAVs performing transition tasks in a limited space. The implementation was conducted in MATLAB 2023a and executed on a PC with Intel Core i7 CPU with 16 cores and 16 GB of RAM. The model of quadcopter used is based on the Crazyflie 2.0 nano-drone using the safety distance $r_{min} = 0.2$ corresponding to its physical size. The scaling matrix is chosen as $\Theta = diag([1, 1, 1])$.

Example 1 (UAVs swapping positions): On the first scenario, we examine the swapping position task of 4 drones using our distributed algorithm. Each drone starts at a symmetrical position on a circle of radius $r_c = 1m$ at altitude $h = 1m$. The two opposite drones swap their positions while avoiding collision. The UAVs are constrained to stay inside the bounded volume with dimension $3.6 \times 3.6 \times 2m$. The simulation results and parameters setup is specified in the Fig. 4 and Table. I respectively.

As can be seen in Fig 5, the minimum distance between the UAVs is about $d_{min} = 0.3m$ which indicates that our algorithm are able to plan the trajectory on the fly with no collision. We count the flying task as successful if all the drones reach their targets within a distance of $0.1m$.

Example 2 (Transition task for large group of UAVs): In this scenario, we want to evaluate the feasibility of the distributed planning algorithm for a group of 10 UAVs

TABLE I: Tuning parameters for 4 drones swapping scenario

Sampling time h(s)	0.2
Degree of B-spline p	3
Control points N	5
No. of agents N_a	4
Prediction horizon N_p	15
Safety distance r_{min}	0.2
$\mathbf{Q}_p$	$diag(100, 100, 100)$
$\mathbf{R}$	$diag(50, 50, 200)$
$\mathbf{P}$	$diag(50, 50, 50)$
(η_1, η_2)	$(50, 100)$
Transition time (s)	5.8
Computation time (s)	3.2751
Average QP solving time (s)	0.039

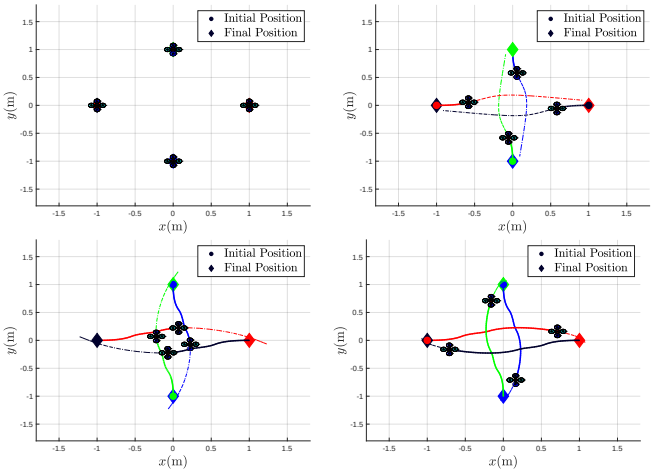

Fig. 4: 4 drones swapping position online

spawned at random positions in a bounded volume defined as in Example 1. The specifications and setup parameters for simulations are detailed in Table II. We observed that the agents can navigate safely, as shown in Fig. 7.

With an increasing number of UAVs, the algorithm's feasibility becomes expectedly more challenging due to the limited workspace. One can notice that even when the prediction horizon of the MPC scheme is changed, the number of decision variables remains.

TABLE II: Tuning parameters for 10 drones in transition task

Sampling time h(s)	0.2
Degree of B-spline p	3
Control points N	5
No. of agents N_a	10
Prediction horizon N_p	10
Safety distance r_{min}	0.2
$\mathbf{Q}_p$	$diag(100, 100, 100)$
$\mathbf{R}$	$diag(50, 50, 200)$
$\mathbf{P}$	$diag(50, 50, 50)$
(η_1, η_2)	$(50, 100)$
Transition time (s)	6.4
Computation time (s)	5.4604
Average QP solving time (s)	0.0427

B. Analysis of the performance of the proposed method

A Monte Carlo simulation is conducted for various approaches to compare the efficiency in terms of probability

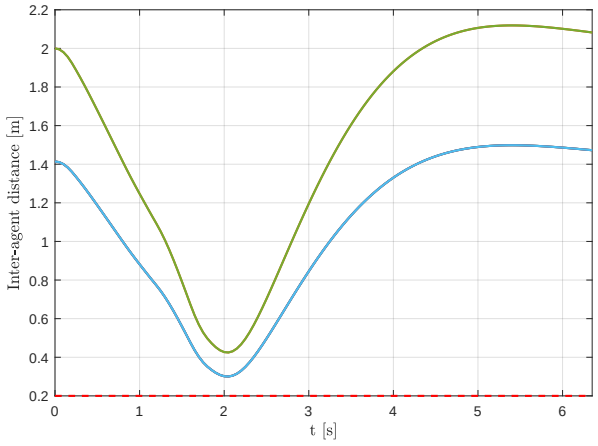

Fig. 5: Inter-UAV distance of 4 drones with safety distance $r_{min} = 0.2m$

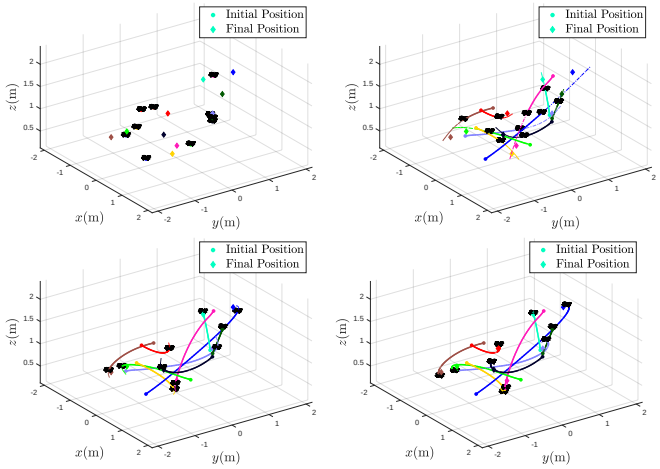

Fig. 6: 10 drones tracking the target position online

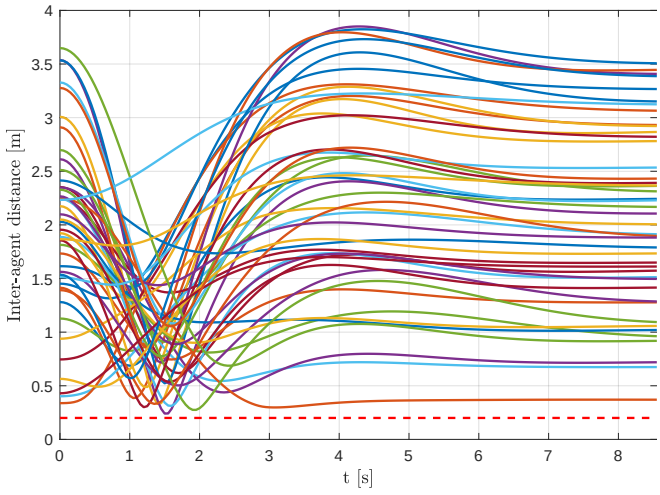

Fig. 7: Inter-agent distances among 10 agents

of success and transition time. The following situations are considered. In a fixed area of $26m^3(3.6 \times 3.6 \times 2.0)$, we generate 50 different random test cases for various swarm sizes from 10 to 60 and compute their average. The following control algorithms are investigated in simulation:

1) Our approach: Soft constraint using B-spline in Distributed MPC: Soft-B-DMPC
2) Soft constraint using B-spline in Centralized MPC: Soft-B-CMPC
3) Bspline parameterization in Distributed MPC: B-DMPC
4) On Demand collision avoidance using Bézier in Distributed MPC [25]

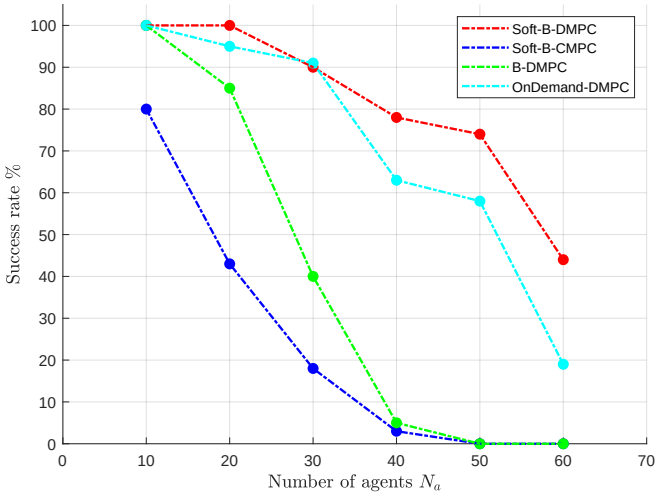

Fig. 8: Success rate in transition tasks

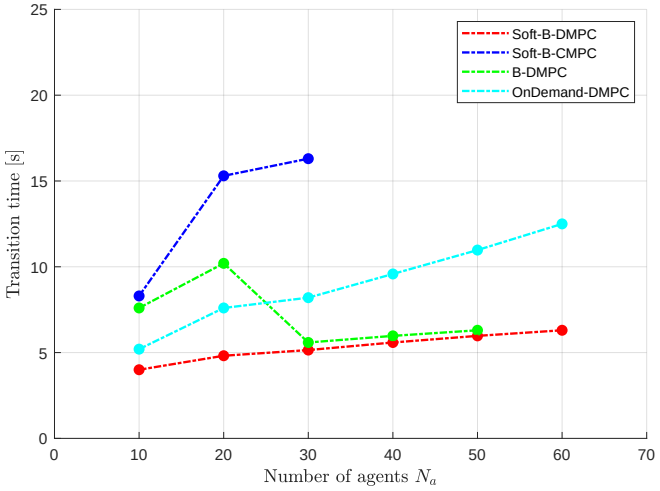

Fig. 9: Transition time of different methods of trajectory planning

To find out how the environment's density can affect the algorithms, various swarm sizes are examined. When no UAV collides and every target is tracked in less than 20 seconds, we consider the simulation to be terminated.

The performance achieved through each technique is demonstrated. In Fig. 8, the success probability for each swarm size is highlighted. With an increasing number of agents, the effectiveness of centralized method notably decreases. While soft constraints offer advantages, their lack of

scalability limits their applicability in large UAV systems.

With our approach using the idea of B-spline and distributed MPC, the performance is shows improvements when testing for swarm size larger than 30. This is a promising solution for fast online trajectory generation as the number of control points required for the problem decrease thanks to the continuity property.

V. Conclusions

This paper introduced a distributed MPC (Model Predictive Control) algorithm for trajectory generation of multi-agent systems by using B-spline parameterization. The collision avoidance constraints involving multiple agents are resolved with a relaxed BVC (Buffered Voronoi Cell) implementation. Simulation examples and performance analysis are conducted to validate the applicability in real time for multiple drones. The undergoing work concentrates on the experimental implementation of the proposed algorithm.

References

[1] J. J. Roldán-Gómez, E. González-Gironda, and A. Barrientos, "A Survey on Robotic Technologies for Forest Firefighting: Applying Drone Swarms to Improve Firefighters' Efficiency and Safety," *Applied Sciences*, vol. 11, no. 1, p. 363, Jan. 2021.

[2] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, and S. Koenig, "Lifelong Multi-Agent Path Finding in Large-Scale Warehouses," *AAAI*, vol. 35, no. 13, pp. 11 272–11 281, May 2021.

[3] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[4] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019, publisher: IEEE.

[5] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm Robotic Behaviors and Current Applications," *Front. Robot. AI*, vol. 7, p. 36, Apr. 2020.

[6] S. Siebert and J. Teizer, "Mobile 3d mapping for surveying earthwork projects using an unmanned aerial vehicle (uav) system," *Automation in construction*, vol. 41, pp. 1–14, 2014.

[7] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3d exploration and surface inspection," *Autonomous Robots*, vol. 42, pp. 291–306, 2018.

[8] M. Petrlík, T. Báča, D. Heřt, M. Vrba, T. Krajník, and M. Saska, "A robust uav system for operations in a constrained environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2169–2176, 2020.

[9] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in uav enabled wireless sensor network," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 328–331, 2017.

[10] Z. Wang, R. Liu, Q. Liu, J. S. Thompson, and M. Kadoch, "Energy-efficient data collection and device positioning in uav-assisted iot," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1122–1139, 2019.

[11] N. Zhao, W. Lu, M. Sheng, Y. Chen, J. Tang, F. R. Yu, and K.-K. Wong, "Uav-assisted emergency networks in disasters," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 45–51, 2019.

[12] P. Di Lillo, F. Pierri, G. Antonelli, F. Caccavale, and A. Ollero, "A framework for set-based kinematic control of multi-robot systems," *Control Engineering Practice*, vol. 106, p. 104669, 2021.

[13] G. Notomista, S. Mayya, M. Selvaggio, M. Santos, and C. Secchi, "A set-theoretic approach to multi-task execution and prioritization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9873–9879.

[14] G. Demesure, M. Defoort, A. Bekrar, D. Trentesaux, and M. Djemai, "Decentralized motion planning and scheduling of agvs in an fms," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1744–1752, 2017.

[15] I. Prodan, S. Olaru, C. Stoica, and S.-I. Niculescu, "Predictive control for trajectory tracking and decentralized navigation of multi-agent formations," *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 1, pp. 91–102, 2013.

[16] R. J. Afonso, R. K. Galvão, G. A. Souza, M. R. Maximo, and A. Caregnato-Neto, "Linear constraints for ensuring k-hop connectivity using mixed-integer programming for multi-agent systems," *International Journal of Robust and Nonlinear Control*, vol. 34, no. 2, pp. 1433–1447, 2024.

[17] I. Prodan, F. Stoican, S. Olaru, and S.-I. Niculescu, *Mixed-integer representations in control design: Mathematical foundations and applications*. Springer, 2016.

[18] N.-Q.-H. Tran, I. Prodan, E. Grøtli, and L. Lefèvre, "Potential-field constructions in an mpc framework: application for safe navigation in a variable coastal environment," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 307–312, 2018.

[19] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3882–3889.

[20] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "Amswarm: An alternating minimization approach for safe motion planning of quadrotor swarms in cluttered environments," *arXiv preprint arXiv:2303.04856*, 2023.

[21] V. Freire and X. Xu, "Flatness-based quadcopter trajectory planning and tracking with continuous-time safety guarantees," *IEEE Transactions on Control Systems Technology*, 2023.

[22] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[23] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of reinforcement learning and control*, pp. 321–384, 2021.

[24] A. Grancharova, E. I. Grøtli, D.-T. Ho, and T. A. Johansen, "Uavs trajectory planning by distributed mpc under radio communication path loss constraints," *Journal of Intelligent & Robotic Systems*, vol. 79, pp. 115–134, 2015.

[25] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[26] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, online collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[27] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[28] I. Prodan, F. Stoican, and C. Louembet, "Necessary and sufficient lmi conditions for constraints satisfaction within a b-spline framework," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 8061–8066.

[29] H.-T. Do, I. Prodan, and F. Stoican, "Analysis of alternative flat representations of a uav for trajectory generation and tracking," in *2021 25th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2021, pp. 58–63.

[30] L. Piegl and W. Tiller, *The NURBS book*. Springer Science & Business Media, 1996.

[31] B. Boots, K. Sugihara, S. N. Chiu, and A. Okabe, "Spatial tessellations: concepts and applications of voronoi diagrams," 2009.