Distributed MPC for motion planning of multiple drones

Cong Khanh DINH [1] Ionela PRODAN [1] Florin STOICAN [2]

[1]Univ. Grenoble Alpes, Grenoble INP[†], LCIS, F-26000, Valence, France
[†]Institute of Engineering and Management Univ. Grenoble Alpes
Email: {cong-khanh.dinh, ionela.prodan}@lcis.grenoble-inp.fr

[2]Faculty of Automation Control and Computer Science, University Politehnica of Bucharest, Romania,
Email: florin.stoican@upb.ro

Context

We address the motion planning problem for multiple drones and on-the-fly trajectories update.

- Provide motion planning mechanism including trajectory generation and tracking which ensures that the agents can reach the a priori given targets, avoid collision and minimize energy costs.
- Apply distributed control strategies for the agents task assignment.
- Experimentally test the algorithms over the Crazyflie nanoquadcopters.

Problem statement

Given N_a agents with known dynamics, compute input for each agent such that:

- The agents do not collide
- The agents remain within working space
- After a time T_f, the agents are sufficiently close to their desired position

Model of the agents

We consider the model of the nano-quadcopter Crazyflie which is naturally non-linear. By applying the flatness-based linearization approach in **do2021analysis**, we can obtain an exact linearized model under the form of a double integrator dynamics:

$$\mathbf{x}_i[k+1] = \mathbf{A}\mathbf{x}_i[k] + \mathbf{B}\mathbf{u}_i[k] \quad (1)$$

where $A = \begin{bmatrix} \mathbf{I}_3 & h\mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}$ and $B = \begin{bmatrix} \frac{h^2}{2}\mathbf{I}_3 \\ h\mathbf{I}_3 \end{bmatrix}$.

The state of the i^{th} agent $\mathbf{x}_i[k]$ is defined by the position and velocity of the vehicle, i.e $\mathbf{x}_i[k] = (\mathbf{p}_i^\intercal[k], \mathbf{v}_i^\intercal[k])^\intercal \in \mathbb{R}^6$, the control input $\mathbf{u}_i[k] = \mathbf{a}_i[k] \in \mathbb{R}^3$ is the acceleration of the vehicle.

Prediction model for the MPC (Model Predictive Control):

Let us consider the $\hat{(.)}[k|k_t]$ denotes the predicted value of $(.)[k+k_t]$ with information available at k_t. Hence, the prediction model is given by:

$$\hat{\mathbf{x}}_i[k+1|k_t] = \mathbf{A}_i\hat{\mathbf{x}}_i[k|k_t] + \mathbf{B}_i\hat{\mathbf{u}}_i[k|k_t] \quad (2)$$

with $k \in \{0, ..., N_p - 1\}$ where N_p is the prediction horizon.

Trajectory parameterization using B-splines

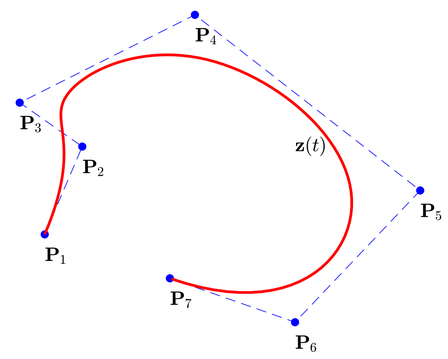
B-splines have useful properties (such as **convexity, local support, differentiability**), allowing to obtain smooth trajectories and fast reconfigurations **prodan2019necessary**.
Given open knot-vector $\tau = \{\tau_1, \tau_2, ..., \tau_m\}$, the B-spline curve $\mathbf{z}(t)$ is characterized as following:

$$\mathbf{z}(t) = \mathbf{P}\mathbf{B}_p(t), \quad \forall\, t \in [\tau_1, \tau_m]$$

where: $\mathbf{P} = [P_0, P_1, ..., P_N]$ collects $N+1$ control points,
$\mathbf{B}_p(t) = [B_{0,p}(t), B_{1,p}(t), ..., B_{N,p}(t)]$ is the associated set of basis functions.
In our framework the control input is expressed in terms of the control points:

$$\mathbf{u}(t) = \mathbf{P}^{(2)}B_{p-2}(t)$$

where $\mathbf{P}^{(2)} = \mathbf{P}\mathbf{M}_2$ is the control points of the r^{th} derivatives of the curve.



Distributed MPC algorithm

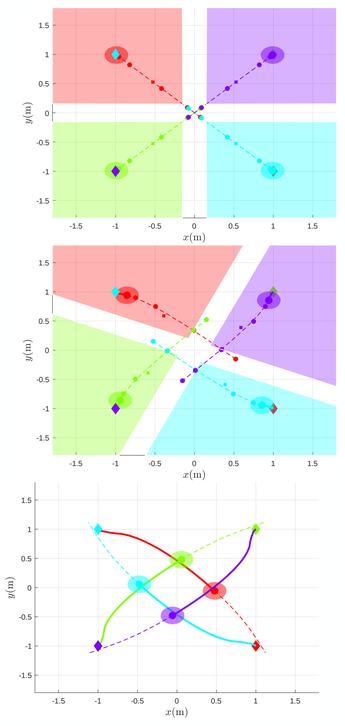
In the control loop, the optimization problems of every agent are solved in parallel using the preceding prediction data broadcast among the agents. The following cost function is minimized:

$$\min_{\mathbf{P}_i, \{\Phi_{i,j}\}_{j\in 1,...,N_a}} \bar{\mathcal{L}}_i(\mathbf{P}_i, \Phi_{i,1}, \Phi_{i,2}, ..., \Phi_{i,N_a})$$

S.t.
- Dynamical model (1)(2)
- $\mathbf{u}_i[k|k_t] = \mathbf{P}_i^{(2)}\mathbf{B}_{p-2}(t_k)$
- $u_{min} \leq \hat{\mathbf{u}}_i[k|k_t] \leq u_{max}$
- $p_{min} \leq \hat{\mathbf{p}}_i[k|k_t] \leq p_{max}$
- Soft constraints for collision avoidance **dinh2024online**: $\| \hat{\mathbf{p}}_i[k|k_t] - \hat{\mathbf{p}}_j[k] \|_2^2 \geq r_{min} + \Phi_{i,j}$

where:

$$\begin{aligned}\bar{\mathcal{L}}_i(\mathbf{P}_i, \Phi_{i,j}) &= \mathcal{L}_i(\mathbf{P}_i) + \mathcal{C}_i(\{\Phi_{i,j}\}) \\ &= \sum_{k=1}^{N_p} \| \hat{\mathbf{p}}_i[k|k_t] - \mathbf{p}_i^f \|_{\mathbf{Q}_k} \\ &+ q_a \sum_{l=0}^{N+2}\sum_{m=0}^{N+2} \left[\mathbf{P}_{i,l}^{(2)}\right]^\intercal \left(\int_0^{t_h} \mathbf{B}_{l,p-2}(t)\mathbf{B}_{m,p-2}(t) \right) \left[\mathbf{P}_{i,m}^{(2)}\right] \\ &+ \sum_{j=1}^{N_a} \eta \| \Phi_{i,j} \|^2\end{aligned}$$



Synchronous Algorithm for collision avoidance :

The proposed approach is based on synchronous distributed MPC, where the agents share their previously predicted state sequence with their neighbours before simultaneously solving the next optimization problem.
At discrete time step k_t, each agent simultaneously computes new input sequence over the horizon following these steps:

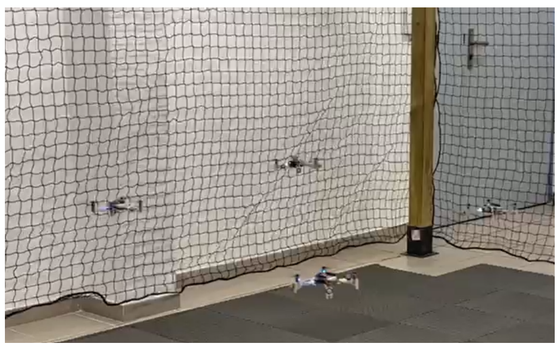
1. Check for future collision using the latest predicted states of neighbours(at time step $k_t - 1$)
2. Build the optimization problem where the constraints for collision avoidance are introduced *only if collision is detected*.
3. After obtaining the next optimal sequence, the first element is applied to the model and the agents move one step ahead. All the states predicted will be shared among the agents.

Simulation specifications

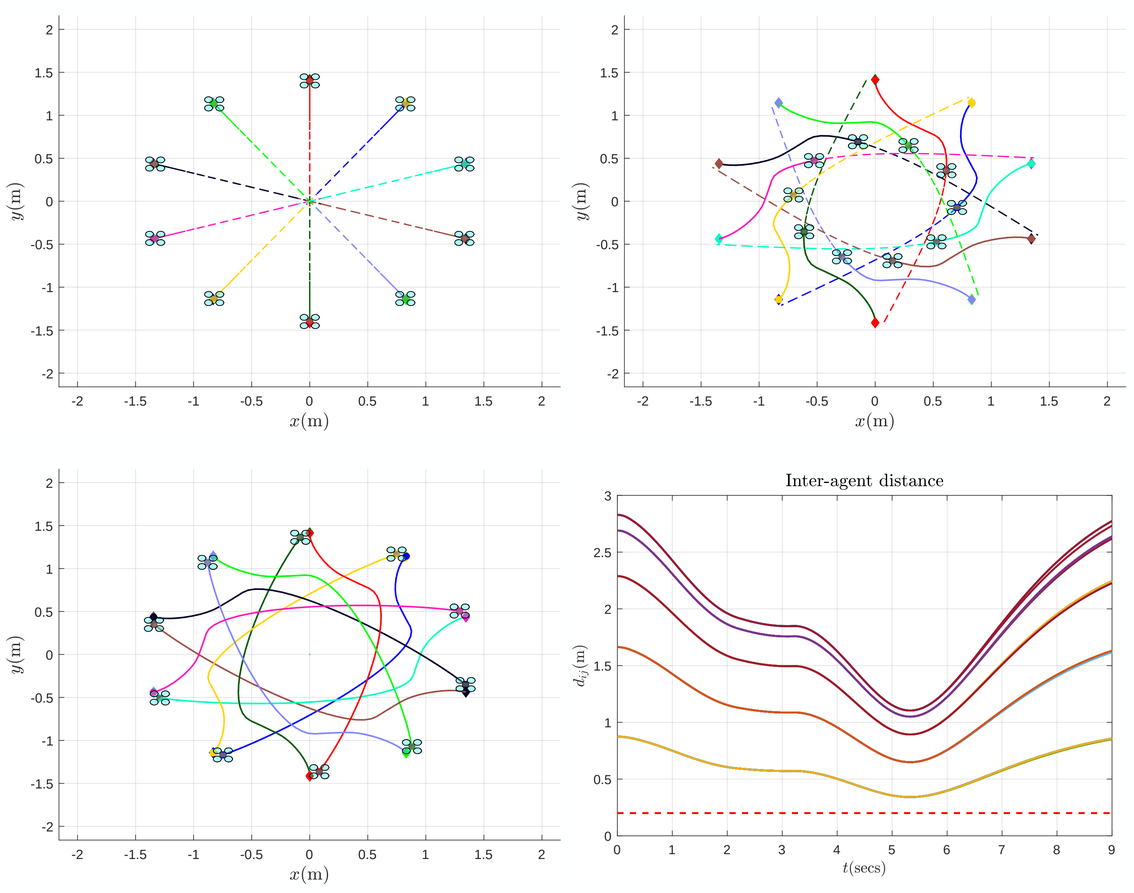
Table 1. Tuning parameters

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	5
Safety distance r_{min}	0.2
$\mathbf{Q}_p$	$diag(100, 100, 100)$
$\mathbf{R}$	$diag(50, 50, 200)$
η	100
Transition time (s)	9.0
Computation time (s)	6.4581
Average QP solving time (s)	0.014039

Figure 2. Crazyflie drones at Esisarium

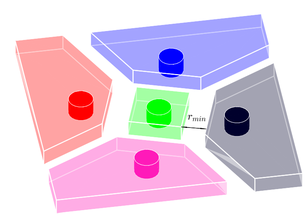


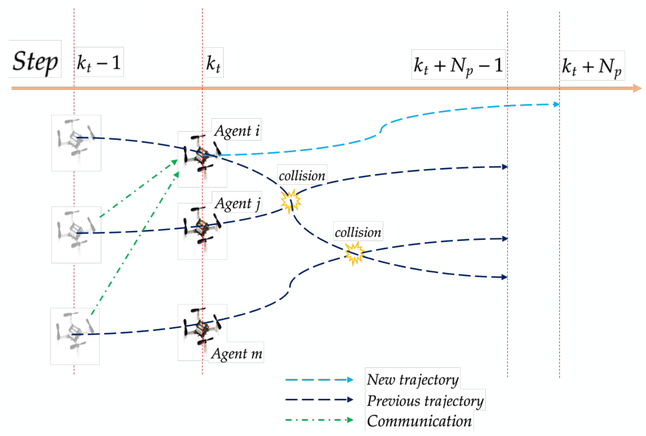
Simulation results



Conclusion and future work

- Exploit the B-spline parameterization for solving the trajectory generation problem.
- Consider collision avoidance constraints online.
- Future work will focus on adding safety guarantees of the collision avoidance constraints and testing the algorithms in experiments.





References