Towards Scalable Online Trajectory Generation for Multi-robot Systems

Carlos E. Luis, Marijan Vukosavljev, and Angela P. Schoellig

***Abstract*— We present a distributed model predictive control (DMPC) algorithm to generate trajectories in real-time for multiple robots, taking into account their trajectory tracking dynamics and actuation limits. An event-triggered replanning strategy is proposed to account for disturbances in the system. We adopted the *on-demand collision avoidance* method presented in previous work to efficiently compute non-colliding trajectories in transition tasks. Preliminary results in simulation show a higher success rate than previous online methods based on Buffered Voronoi Cells (BVC), while maintaining computational tractability for real-time operation.**

I. Introduction

Online trajectory generation is key to execute missions in dynamic or unknown environments. In particular, multi-robot tasks are especially challenging due to a high number of decision-making agents sharing the same space. In such settings, the planning algorithms must compute collision-free and goal-oriented trajectories, taking into account the state of the environment and neighbouring agents.

There exists a wide variety of techniques to tackle the multi-robot trajectory generation problem. Firstly, optimization-based techniques such as Sequential Convex Programming (SCP) [1] and Distributed Model Predictive Control (DMPC) [2] have had success in generating the trajectories offline. Secondly, discrete planning strategies like Rapidly-exploring Random Trees (RRT) have been extended to the multi-agent case [3]. Thirdly, a combination of discrete planning and continuous optimization has been developed to coordinate multiple robots in cluttered environments [4].

Despite the advances in scalability and safety of the algorithms, online trajectory generation for large groups of robots remains a challenge. Optimal Reciprocal Collision Avoidance (ORCA) and all its variants have pushed towards real-time trajectory generation [5], with convincing results in various robotic platforms in planar environments [6]. A similar approach achieves collision avoidance through the concept of Buffered Voronoi Cells (BVC) [7], showing initial results of online trajectory generation in 2D with multiple quadrotors operating at a fixed height. The BVC concept has been recently used in tandem with discrete planners [8], primarily to avoid deadlocks in scenarios where plain BVC would get trapped and fail the task.

In this work we propose an alternative framework for online trajectory generation of robot teams that must transition between two configurations. As such, our framework provides an essential functionality for higher level planners that specify complex team missions in terms of goal locations to be visited by the agents.

All the authors are with the Dynamic Systems Lab (www.dynsyslab.org) at the University of Toronto Institute for Aerospace Studies (UTIAS), Canada. Email: carlos.luis@robotics.utias.utoronto.ca, mario.vukosavljev@robotics.utias.utoronto.ca, schoellig@utias.utoronto.ca

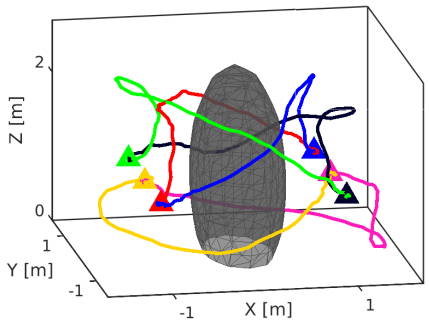


Fig. 1. Simulation of six quadrotors performing an antipodal exchange task with a static obstacle in the middle, using our online trajectory generation strategy. Coloured triangles represent the initial location of each agent.

The framework builds upon our previous DMPC algorithm for offline computation [9]. Our approach contrasts from current online methods (e.g., [8]) in that:

- It does not require any pre-computed trajectories or a discrete planner that generates collision-free solutions for the optimizer.
- It does not assume perfect trajectory tracking. Instead, the algorithm leverages knowledge on the dynamics of the system.
- It uses *on-demand collision avoidance* as in [9].

Our preliminary results show that our strategy avoids deadlocks better than state-of-the-art methods in distributed online motion planning based on BVC [7], without the need of a high-level discrete planner as in [8]. Also, our method takes into account the dynamics of the agents to generate collision-free motion, as opposed to previous methods that ignored the dynamics and only computed non-colliding reference signals.

This brief focuses on the main aspects of our method and preliminary simulation results. Experiments with a quadrotor swarm will come in the future.

II. Approach

Our formulation tackles the point-to-point trajectory generation problem for N labelled agents. Each agent is assigned a goal location which must be reached without collisions. We assume each agent i obeys some trajectory tracking dynamics given by a discrete linear system:

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

For instance, we may consider the system (1) to represent a quadrotor with an underlying position controller,

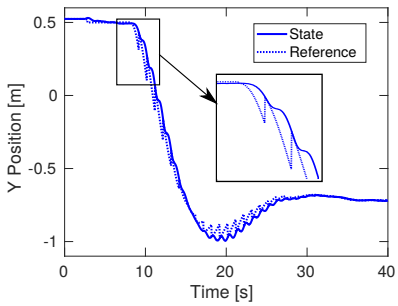


Fig. 2. Experimental data of a quadrotor flight when using online trajectory generation based on DMPC [9] with replanning every second. The discontinuities in the reference signal causes undesired behaviour.

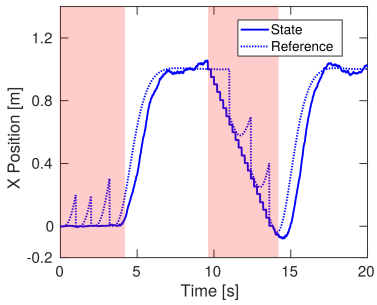


Fig. 3. Trajectory generation for a simulated quadrotor using event-triggered replanning with the activation function in (2). The robot was perturbed during the highlighted segments in red.

for which the inputs are position reference signals and the states are the position and velocity of the vehicle, i.e., $\mathbf{x}_i[k] = (\mathbf{p}_i[k], \mathbf{v}_i[k])$.

The approach is based on receding horizon control, meaning that every t_p seconds we recompute the input sequence to be applied over a finite horizon. We parameterize the continuous input signal $\mathbf{u}_i(t)$ for $t \in [t_0, t_0 + t_p]$ as a concatenation of l Bézier curves, similar to [4].

To compute the next optimal input sequence for each agent we follow these steps:

1) Based on the current states of the agent, decide what should be the initial condition for the input sequence.
2) Check for future collisions using the latest predicted states of neighbouring robots.
3) Build and solve a convex optimization problem, where the decision variables are the control points of the l Bézier curves describing the input $\mathbf{u}_i(t)$.

A. Event-triggered Replanning

Choosing the initial condition for the input to be equal to the current state of the robot was proposed in [8], but it has certain limitations. Firstly, if we require C^r-continuity on the inputs, then we need to reliably measure the r-th derivative of the robot's position. Secondly, for imperfect trajectory tracking this replanning strategy constantly causes (potentially big) discontinuities of the input to match the state of the robot, as shown in Fig. 2. Such discontinuities cause undesired jittering in the robot and slows down its progress to complete the task.

To address these concerns, we propose an event-triggered replanning strategy, in which we reset the input to match the states of the agent only whenever we detect the agent has been perturbed. To detect such an event, we designed a heuristic activation function that we threshold to detect disturbances to the agent. An example of such an activation function for second-order tracking dynamics is:

$$\boldsymbol{f}[k] = \frac{(\mathbf{p}_i[k] - \mathbf{u}_i[k])^5}{-(\mathbf{v}_i[k] + sgn(\mathbf{v}_i[k])\varepsilon)}, \tag{2}$$

where the term $(\mathbf{p}_i[k] - \mathbf{u}_i[k])$ is the trajectory tracking error, and the term $sgn(\mathbf{v}_i[k])\varepsilon$ with a small scalar $\varepsilon << 1$ is used to avoid singularities in $\boldsymbol{f}[k]$. We assume $|\mathbf{v}_i[k]| > 0$, which is realistic in real-world operation.

The intuition behind (2) is that we want to reset our reference signal whenever the tracking error grows large. However, designing an appropriate threshold value for the tracking error is tricky due to its high variability during execution. Instead, $\boldsymbol{f}[k]$ is designed to detect whenever the error is growing but the velocity is either small or growing in the opposite direction of the error. To detect these scenarios, we define the robot is operating normally if the inequality

$$\boldsymbol{f}_{\min} < \boldsymbol{f}[k] < \boldsymbol{f}_{\max} \tag{3}$$

holds for every element of $\boldsymbol{f}[k]$. The values of $\boldsymbol{f}_{\min}$ and $\boldsymbol{f}_{\max}$ must be chosen by extracting the extrema of $\boldsymbol{f}[k]$ under normal operation. If (3) does not hold, then the agent is being disturbed and we set the initial position and velocity of the Bézier curve to match the states of the vehicle, while setting higher-order derivatives to zero.

An example of our strategy is shown in Fig. 3, where a simulated quadrotor is tasked to reach an x coordinate equal to 1 meter. During the first red segment of the trajectory the robot is unable to move, and during the second red segment it is being pushed away by an external force. In both cases the condition in (3) is eventually violated and the replanning is triggered. Under normal operation (white segments) the replanning is not required and we avoid the shortcomings observed in Fig. 2.

B. On-demand Collision Avoidance

We adopted the on-demand collision avoidance method of [9], where collision constraints are used only if we predict future collisions on the horizon. This strategy assumes communicative agents that share with nearby neighbours a representation of their future actions. More precisely, after every input update the agents share a sampled representation of their future states, as predicted by their known dynamics. Leveraging this information, the agents are able to predict and avoid collisions.

With imperfect trajectory tracking, we cannot guarantee that non-colliding reference signals will generate non-colliding motion for the dynamic agents. Therefore, our avoidance strategy is aware of the trajectory tracking dynamics by imposing collision constraints on the *states* of the agents, rather than the inputs of the system.

As shown in Fig. 1, preliminary results in simulation show responsive 3D collision avoidance capabilities in environments with dynamic agents and static obstacles.

C. Convex Optimization

To update their optimal input sequence, each agent solves a convex optimization problem with the following components:

1) Equality Constraints: they ensure C^r-continuity between the concatenated segments of the Bézier curve representing the input of the system.

2) Inequality Constraints: with them we wish to limit the input of the system to obey physical boundaries, for instance, workspace dimensions and actuation limits.

One option to implement these inequality constraints is to exploit the convex hull property of Bézier curves. If we limit the control points of the curve to lie within a convex region, we know the curve will be contained within that region. This may, however, impose overly conservative bounds [10]. A second option, as suggested in [8], is to not impose the constraints at all and check afterwards if the solution satisfies the constraints; if it does not, the problem needs to be resolved one more time to guarantee constraint satisfaction.

To overcome the above mentioned limitations, we imposed constraints on discrete samples of the Bézier curve and its derivatives. We built a set of matrices $\mathbf{\Phi}^r$ that transform the decision vector into a sampled representation of the r-th derivative of the Bézier curve along the horizon. With this formulation we can limit more precisely the input and its derivatives, without having to perform a post-solve check for dynamic feasibility.

Additionally, if collisions are predicted on the horizon, we include separating hyperplane constraints on the position of the agents (not their reference signals), following the construction in [9]. The constraints are relaxed (soft constraints) to avoid infeasibilities of the optimization problem.

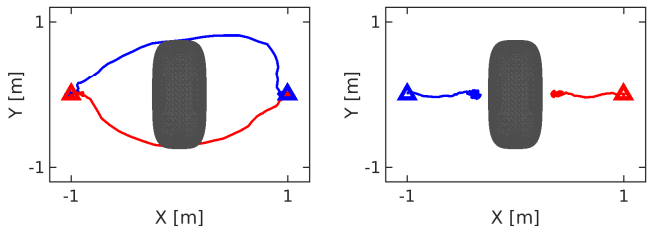
3) Cost Function: we search to i) minimize the error between the agents' positions and their respective goals, ii) minimize a measure of energy along the trajectory (e.g., minimum snap for quadrotors) and iii) if collisions are predicted, we want to minimize the constraint violation (as soft constraints). Each term is weighted and added together to obtain a scalar cost.

The end result is a convex optimization problem that is solved simultaneously by each agent, leading to efficient distributed computation. Each optimization has linear equality and inequality constraints and a quadratic cost function, which can be efficiently solved with a QP solver.

D. Robust Safety Layer

Our optimization does not guarantee collision-free trajectories. Instead, we search to minimize the magnitude of the collision constraint violation given the actuation limits of the agents. Although this has proven to be effective as an offline method, during real-time execution we are striving for resilient collision avoidance with our framework.

On-going work is focused on local strategies to deal with collisions robustly, in the presence of constraint violation or disturbances. Our first attempt was to force the agents to repel each other whenever they were trespassing their safety distance. The way we implemented this strategy was to temporarily change the goal location of the colliding agents to be in opposite directions of their distance vector, creating a repel effect. We tested the method in a wide range of transition tasks and the results are inconclusive: it may avoid collisions in certain scenarios, but in others it may indirectly induce future collisions. However, more strategically chosen temporary goal locations may have the potential to be an effective method for resolving collisions.



(a) On-demand avoidance (b) BVC

Fig. 4. A position exchange task with 2 agents and an obstacle in the middle. The triangles symbolise the initial location of each agent. The BVC method gets trapped in a deadlock while on-demand collision avoidance successfully completes the task, without any centrally pre-planned trajectory.

III. Results

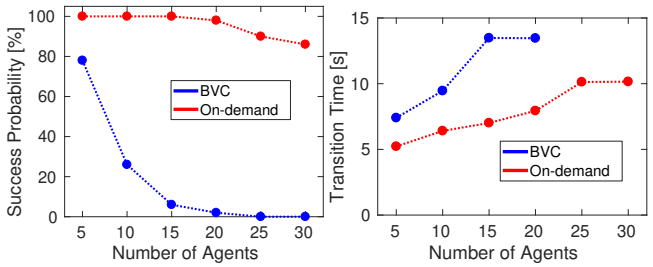
We created a simulation environment in MATLAB where the agents were modeled after the Crazyflie 2.0 quadrotor. The trajectory tracking dynamics were identified by fitting a second-order model to experimental data from the step response of the system (quadrotor + on-board position controller). For more realistic simulations, we added sensor noise to the states resulting from applying input signals to the identified model.

For the input sequence we chose Bézier curves of fifth degree with $l = 3$ and $t_p = 3\,\text{s}$, where each segment had a fixed duration of 1 second. Additionally, we imposed continuity constraints up to the third derivative of the curve.

A. Performance in Transition Tasks

We implemented and compared our on-demand collision avoidance strategy with the BVC method proposed on [8]. To study the collision avoidance capabilities of both methods, we first tested a simple scenario with two quadrotors in 2D. In Fig. 4 we show the trajectories of the two agents exchanging their positions with an obstacle in the middle. The on-demand strategy successfully finds a solution to the problem, while the BVC method is unable to circumvent the obstacle. Our method includes less hyperplane constraints per iteration, which helps avoid the deadlock situation.

We thoroughly tested both methods on simulated transition tasks with random initial and final locations for each agent. A trial was declared successful if all agents were able to reach their goals without collisions, within a maximum time of 20 seconds in a $3\times3\times2\,\text{m}$ volume. In Fig. 5a we highlight how the success rate using the BVC method quickly degrades



(a) (b)

Fig. 5. Simulation performance comparison of our on-demand collision avoidance strategy and the BVC method. We considered different number of agents in a fixed volume of $3 \times 3 \times 2$ m. For each swarm size, 50 different random test cases were generated.

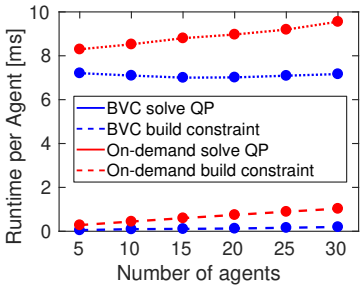


Fig. 6. Comparison of the average runtime per agent using our on-demand collision avoidance and the BVC method. The data shown is the average over 50 randomly generated tests for each swarm size considered.

as the number of agents increases. More specifically, as the density of the agents in the environment increases, the BVC method tends to fail in completing the transition within the required time. As for our method, it has more than 85% success rate with up to 30 vehicles, which represents a density of 1.67 agents per cubic meter. The main reason of failure of our method is due to collisions (12 out of 13 failures); further study is required to reduce the collision scenarios observed in these experiments.

We also compared the average time required to complete the transitions, as shown in Fig. 5b. On average, the BVC method takes longer to complete the task than our on-demand strategy. These results suggest that the collision avoidance method in BVC reduces the mobility of agents, which leads to slower transition times and in some scenarios to deadlocks (as in Fig. 4). Instead, our paradigm based on communicative agents achieves higher coordination, which ultimately reduces the transition times.

B. Runtime Benchmark

We compared the computation time per agent to update their input sequence. In Fig. 6 the results are presented, where we divided the runtime into building the collision constraint and solving the associated QP.

To formally analyze the scaling of both algorithms, define K to be the number of time steps in the planning horizon, and $n_{i,k}$ to be the number of nearby neighbours of agent i to be considered for collision avoidance at time step k. The constraint building time for agent i using BVC has a complexity of $\mathcal{O}(n_{i,k})$ whereas the complexity using on-demand collision avoidance is $\mathcal{O}(K(N + n_{i,k}))$. This is supported by the linear scaling observed for the building time of our method as N increases. Since $n_{i,k}$ is relatively constant, the BVC curve has low fluctuations.

As for solving the QP, the amount of inequality constraints on both methods scales with $\mathcal{O}(n_{i,k})$, but our method adds an additional $n_{i,k}$ decision variables to the problem. The plots in Fig. 6 suggest a quasi-constant scaling using BVC and a linear growth using on-demand collision avoidance. The average QP solving time of our method increases with the agent density, since collisions must be constantly avoided, and with potentially more neighbours.

To summarize, the computational cost of the proposed method is higher than BVC, but with an overall better performance in transition tasks (Fig. 5).

IV. Conclusion and Future Work

We presented a framework for multi-robot online trajectory generation based on DMPC. Our method takes into account the trajectory tracking dynamics of the agents which adds robustness to the system. The on-demand collision avoidance strategy showed higher success rate than BVC to solve a wide variety of transition tasks, while keeping a runtime well-suited for real-time implementation.

Future work will focus on testing the framework with our Crazyflie 2.0 swarm testbed and minimizing the probability of collisions.

References

[1] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1917–1922.

[2] R. Van Parys and G. Pipeleers, "Distributed model predictive formation control with inter-vehicle collision avoidance," in *Asian Control Conference (ACC)*, 2017.

[3] M. Čáp, P. Novák, J. Vokrínek, and M. Pěchouček, "Multi-agent rrt: sampling-based cooperative pathfinding," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1263–1264.

[4] 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.

[5] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *International Conference on Robotics and Automation*, 2011, pp. 3475–3482.

[6] J. Alonso-Mora, P. Beardsley, and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404–420, 2018.

[7] 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.

[8] B. Şenbaşlar, W. Hönig, and N. Ayanian, "Robust trajectory execution for multi-robot teams using distributed real-time replanning," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 167–181.

[9] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.

[10] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, no. 99, pp. 1–8, 2017.