Trajectory Generation for Multiagent Point-to-Point Transitions via Distributed Model Predictive Control

Carlos E. Luis

***Abstract*— Here goes the abstract**

I. Introduction

Generating collision-free trajectories when dealing with multiagent systems is a safety-critical task. In missions that require cooperation of multiple agents, such as warehouse management [1], it is often desirable to safely drive the agents from a starting location to a final location. Solving this problem, known as multiagent point-to-point transitions, is therefore an integral objective of any robust multiagent system.

There are two main variations of the multiagent point-to-point transition problem: the labelled or the unlabelled agent problems. In the former, each agent has a fixed final position that cannot be exchanged with other agents [2], [3]; in the latter, the algorithm is free to assign the goals to the agents, as to ease the complexity of the trajectories [4]. This paper will focus on the labelled agent version of the problem.

A natural way to formulate the problem is as an optimization problem. One of the first approaches proposed the use if Mixed Integer Linear Programming (MILP), modelling collision constraints as binary variables [2]. These methods, although viable, are computationally heavy and not suited for large teams.

More recently, techniques based on Sequential Convex Programming (SCP) [5] proved to improve the computational tractability of the algorithms. In [3], SCP is used to solve the optimal-fuel trajectories of quadrotor teams midflight. A decoupled version of that algorithm (dec-iSCP) is proposed in [6], which has shorter computation time at the cost of suboptimal solutions. As later seen in Sec bla bla, even in the decoupled formulation, computation time and feasibility of the problem does not scale well with the number of agents, primarily because the problem itself is harder to solve.

A new solution to the problem is proposed in this paper, based on distributed model predictive control (DMPC) [7]. Synchronous implementation of DMPC [8], allows the agents to solve their optimization problem in parallel, which significantly reduces computation time with respect to previous methods. Previous work on DMPC show its capabilities in multiagent tasks, such as formation control [4], [9], but not explicitly for point-to-point transitions. Moreover, the prediction horizon enables the agents to foresee future collisions and replan their trajectories, making the algorithm robust to infeasibility issues in previous decoupled techniques such as dec-iSCP.

ieurgeruigher

The author is with the Dynamic Systems Lab at the University of Toronto Institute for Aerospace Studies (UTIAS), Canada. Email: carlos.luis@robotics.utias.utoronto.ca

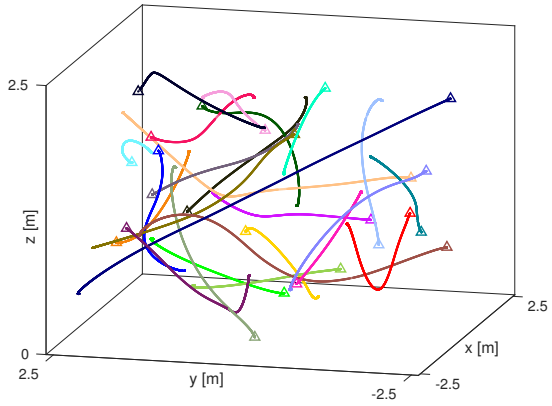

Fig. 1. A 25-vehicle point-to-point transition solved using our DMPC algorithm. The initial and final positions were chosen randomly within a convex volume of $5 \times 5 \times 2$ m. The agents were required to maintain a minimum distance of 0.75m of each other throughout the whole trajectory. Even in this highly cluttered dynamic environment, the proposed algorithm is capable of quickly finding a collision-free trajectory for all agents.

II. Problem Statement

This section introduces the different components of the optimization problem to be solved. The formulation must accomplish the following criteria:

1) Drive all agents from their starting position to their desired final positions.
2) Respect a minimum pair-wise distance at each time step of the trajectory (collision constraint).
3) The position and acceleration of the agents must be within specified boundaries.

A. The model

The model used for each individual agent is a simple kinematic model of a unit mass in $\mathbb{R}^3$. For simplicity, we use an abbreviated matrix representation as if the vectors were one-dimensional:

$$\begin{bmatrix} p[k+1] \\ v[k+1] \end{bmatrix} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p[k] \\ v[k] \end{bmatrix} + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} a[k] \qquad (1)$$

where vectors p, v and a are the discretized position, velocity and acceleration of the mass at time step $k = 0, 1, \ldots, K-1$, with K being the length of the horizon and

h the time step duration in seconds. A compact representation is the following:

$$x[k+1] = Ax[k] + Bu[k] \tag{2}$$

where $x \in \mathbb{R}^6$, $A \in \mathbb{R}^{6\times6}$, $B \in \mathbb{R}^{6\times3}$ and $u[k] \in \mathbb{R}^3$. Furthermore, a closer look into each time step gives the following set of equations:

$$\begin{aligned} x[1] &= Ax[0] + Bu[0] \\ x[2] &= A^2x[0] + ABu[0] + Bu[1] \\ &\vdots \\ x[K] &= A^Kx[0] + A^{K-1}Bu[0] + \ldots + Bu[K-1] \end{aligned} \tag{3}$$

From Eqn. (3), we can write the position sequence $P \in \mathbb{R}^{3K}$ as an affine function of the input sequence $U \in \mathbb{R}^{3K}$.

$$P = A_0X_0 + \Lambda U \tag{4}$$

where $X_0 \in \mathbb{R}^{6K}$ is the repeated sequence of initial states, and $\Lambda \in \mathbb{R}^{3K\times3K}$ is defined as follows

$$\Lambda = \begin{bmatrix} \Psi B & 0 & \ldots & 0 \\ \Psi AB & \Psi B & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \Psi A^{K-1}B & \Psi A^{K-2}B & \ldots & \Psi B \end{bmatrix} \tag{5}$$

with matrix $\Psi = \begin{bmatrix} I_3 & 0 \end{bmatrix}$ selecting the first three rows of the matrix products (those corresponding the position states). Lastly, $A_0 \in \mathbb{R}^{3K\times6}$ is the propagation of the initial states

$$A_0 = \begin{bmatrix} \Psi A & \Psi A^2 & \ldots & \Psi A^K \end{bmatrix}^\top \tag{6}$$

B. Objective function

The objective function has three main components: trajectory error, control effort and input variation.

1) Trajectory error penalty: this term drives the agents towards its waypoint. We want to minimize the error between the position at the final time step of the prediction horizon $p[K]$, and the desired final position p_d. The error term is given by

$$e = \|p[K] - p_d\|_2 \tag{7}$$

Which can be casted into a quadratic cost function of the input sequence, using Eqn. (4):

$$J_e = U^\top(\Lambda^\top\tilde{Q}\Lambda)U - 2(P_d^\top\tilde{Q}\Lambda - A_0X_0\tilde{Q}\Lambda)U \tag{8}$$

Matrix $\tilde{Q} \in \mathbb{R}^{3K\times3K}$ is a positive semidefinite matrix that weights the error at each time step. Given that we want to enforce the trajectory to *end* at the desired reference, then

$$\tilde{Q} = \begin{bmatrix} 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & Q \end{bmatrix} \tag{9}$$

where $Q \in \mathbb{R}^{3\times3}$ is chosen as a diagonal matrix that weights each coordinate of the position vector at the last time step of the horizon.

2) Control effort penalty: we also want to minimize the total control effort throughout the trajectory, through the following quadratic cost function:

$$J_u = U^\top\tilde{R}U \tag{10}$$

Similarly, $\tilde{R} \in \mathbb{R}^{3K\times3K}$ is positive semidefinite and weights the penalty on control effort, and is usually selected with the block diagonal form

$$\tilde{R} = \begin{bmatrix} R & 0 & \ldots & 0 \\ 0 & R & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & R \end{bmatrix} \tag{11}$$

with $R \in \mathbb{R}^{3\times3}$, chosen to be a diagonal positive definite matrix.

3) Input variation penalty: this term is used to minimize variations of the acceleration (jerk), which leads to smooth trajectories in position. First we define the quadratic cost

$$J_\delta = \sum_{k=0}^{K-1} \|u[k] - u[k-1]\|^2 \tag{12}$$

To transform Eqn. (12) into a quadratic form, define $\Delta \in \mathbb{R}^{3K\times3K}$

$$\Delta = \begin{bmatrix} I_3 & 0 & 0 & \ldots & 0 & 0 \\ -I_3 & I_3 & 0 & \ldots & 0 & 0 \\ 0 & -I_3 & I_3 & \ldots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -I_3 & I_3 \end{bmatrix} \tag{13}$$

and define a vector $U_{-1} \in \mathbb{R}^{3K}$ to include the term $u[-1]$ (which is a known constant at iteration k)

$$U_{-1} = \begin{bmatrix} u[-1] & 0 & \ldots & 0 \end{bmatrix}^\top \tag{14}$$

Finally, we write Eqn. (12) in quadratic form

$$J_\delta = U^\top(\Delta^\top\tilde{S}\Delta)U - 2(U_{-1}^\top\tilde{S}\Delta)U \tag{15}$$

where $\tilde{S} \in \mathbb{R}^{3K\times3K}$ is positive definite of the form

$$\tilde{S} = \begin{bmatrix} S & 0 & \ldots & 0 \\ 0 & S & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & S \end{bmatrix} \tag{16}$$

with $S \in \mathbb{R}^{3\times3}$, chosen to be a diagonal positive definite matrix.

The total cost function $\mathcal{J}$ is obtained by combining Eqns. (8,10,15)

$$\begin{aligned} \mathcal{J}(U) = U^\top(\Lambda^\top\tilde{Q}\Lambda + \tilde{R} + \Delta^\top\tilde{S}\Delta)U \\ - 2(P_d^\top\tilde{Q}\Lambda - A_0X_0\tilde{Q}\Lambda + U_{-1}^\top\tilde{S}\Delta)U \end{aligned} \tag{17}$$

C. Collision avoidance

To procure collision avoidance, each agent can access information of the most up-to-date prediction horizon of every other agent (p_j). When agent i predicts a future collision at time step k of the current prediction horizon,

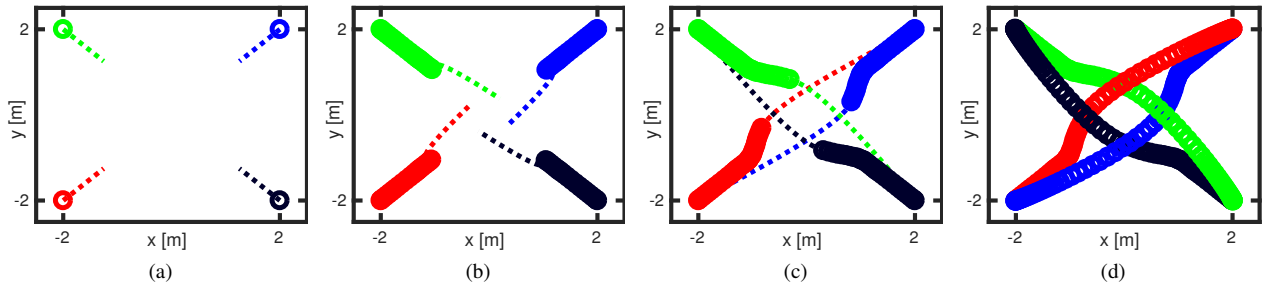

Fig. 2. The caption goes here

then agent i must solve the optimization problem with the following inequality constraint.

$$\|p_i[k] - p_j[k]\|_2 \geq r_{min} \quad \forall i, j = 1, 2, \ldots, N \quad i \neq j \quad (18)$$

The non-convex constraint in Eqn. (18) must be first linearized. For that, we use a Taylor series expansion about the previous iterate q of the predicted position horizon of agent i, p_i^q. Note that this is different than the linearization proposed in [6], in that the linearization is done about a small portion of the total trajectory, given by the previous prediction horizon.

$$\|p_i^q[k] - p_j[k]\|_2 + \frac{(p_i^q[k] - p_j[k])^\top}{\|p_i^q[k] - p_j[k]\|_2}(p_i[k] - p_i^q[k]) \geq r_{min} \quad (19)$$

Note that in Eqn. (19), the only variable is $p_i[k]$, hence we can simplify the expression

$$(p_i^q[k] - p_j[k])^\top p_i[k] \geq \rho_j \quad (20)$$

with

$$\rho_j = r_{min}\|p_i^q[k] - p_j[k]\|_2 - \|p_i^q[k] - p_j[k]\|_2^2 + (p_i^q[k] - p_j[k])^\top p_i^q[k] \quad (21)$$

Now, we cast Eqn. (20) into an affine inequality of the input sequence U:

$$\Upsilon_j \Lambda U_i \geq \rho_j - \Upsilon_j A_0 x_i[0] \quad (22)$$

where $\Upsilon_j \in \mathbb{R}^{1\times 3K}$ is

$$\Upsilon_j = \begin{bmatrix} 0_{1\times 3k} & (p_i^q[k] - p_j[k])^\top & 0_{1\times 3(K-1-k)} \end{bmatrix} \quad (23)$$

Finally, we define $A_{coll,j} \in \mathbb{R}^{1\times 3K}$ and $b_{coll,j} \in \mathbb{R}$

$$A_{coll,j} = \Upsilon_j \Lambda; \quad b_{coll,j} = \rho_j - \Upsilon_j A_0 x_i[0] \quad (24)$$

The complete collision constraint matrices for agent i, $A_{coll} \in \mathbb{R}^{(N-1)\times 3K}$ and $b_{coll} \in \mathbb{R}^{(N-1)}$, can be obtained by stacking vertically the matrices in Eqn. (24) for all the $N-1$ neighbours of agent i.

D. Physical Limits

The agents must remain within a specified volume and respecting acceleration limits, hence inequality constraints must be enforced both in the position and acceleration at each time step of the prediction horizon. Let $p_{min}, p_{max}, a_{min}, a_{max} \in \mathbb{R}^3$ be the physical limits of the system, define $P_{min}, P_{max}, U_{min}, U_{max} \in \mathbb{R}^{3K}$:

$$\begin{aligned} P_{min} &= [p_{min} \ldots p_{min}]^\top; \quad P_{max} = [p_{max} \ldots p_{max}]^\top \\ U_{min} &= [a_{min} \ldots a_{min}]^\top; \quad U_{max} = [a_{max} \ldots a_{max}]^\top \end{aligned} \quad (25)$$

Then, the physical limits constraints are formulated as

$$\begin{aligned} P_{min} \leq \Lambda U \leq P_{max} \\ U_{min} \leq U \leq U_{max} \end{aligned} \quad (26)$$

E. Convex optimization problem

The complete inequality constraints for agent i are obtained by stacking vertically the collision avoidance and physical limits constraints. At each time step, agent i solves the following convex optimization problem to update its prediction horizon

$$\begin{aligned} \underset{U}{\text{minimize}} \quad & \mathcal{J}(U) \\ \text{subject to} \quad & A_{in} U \leq b_{in} \end{aligned} \quad (27)$$

with $A_{in} \in \mathbb{R}^{(6K+N-1)\times 3K}$ and $b_{in} \in \mathbb{R}^{6K+N-1}$. The formulation in Eqn. (27) results in a Quadratic Programming problem, that can be computed efficiently by QP solvers.

III. Distributed Model Predictive Control Algorithm

The proposed DMPC algorithm for point-to-point transitions is outlined in Alg. 1. It requires as input several parameters used in Section II: h, K, N, Q, R, S, p_{min}, p_{max}, a_{min}, a_{max}, r_{min}. Also, a simulation duration T, in seconds, needs to be specified. In line 1, the algorithm initializes a list of prediction horizons for all vehicles by specifying linear trajectories from initial positions to final positions. While it hasn't converged or the maximum number of iterations is not exceeded, the algorithm goes through each time step of the trajectory ($k_T = 1, ..., K_T$ with $K_T = T/h+1$), and through every agent, solving the corresponding QP as derived in Eqn. (27). If a solution is found, then it executes a state propagation by applying the first input of the optimization variable U, to the model in Eqn. (1) (lines 6-8). If a solution was not found due to infeasibility of the problem (or exceeding the maximum number of iterations of the QP solver), then a heuristic approach was introduced to retry solving the problem.

Algorithm 1: DMPC for Point-to-Point Transitions

Input : Initial and final positions, DMPC parameters
Output: Position, velocity and acceleration trajectories for all vehicles

1 $l \leftarrow$ Initialize predictions
2 **while** *trials* $<$ *max and not converged* **do**
3 **foreach** *trajectory time step* $k_T = 1, ..., K_T - 1$ **do**
4 **foreach** *agent* $i = 1, ..., N$ **do**
5 $a_i[k_T] \leftarrow$ Solve QP - Eqn. (27)
6 **if** *QP solved* **then**
7 $p_i[k_T], v_i[k_T] \leftarrow$ Propagate States
8 $l \leftarrow$ Update Prediction Horizon
9 **else**
10 exit loop
11 **if** *QP not solved* **then**
12 $Q \leftarrow$ Increase Q
13 $trials \leftarrow trials + 1$
14 exit loop
15 **if** *QP solved and all vehicles reached goal* **then**
16 converged $\leftarrow$ true;
17 **else if** *QP solved and not all vehicles reached goal* **then**
18 $Q \leftarrow$ Increase Q
19 $trials \leftarrow trials + 1$
20 **if** *converged* **then**
21 $[p, v, a] \leftarrow$ Interpolate for higher resolution
22 **else**
23 $[p, v, a] \leftarrow \emptyset$
24 **return** *[p,v,a]*

A. Heuristic approach for improved feasibility

The heuristic approach considers two different scenarios for choosing the values of the Q matrix.

1) No collisions are detected in the horizon: the agent has a more aggressive behaviour towards the goal. Uses matrix Q_{agg} with high enough value to procure good trajectory tracking.
2) Collisions in the horizon: a more conservative behaviour is required to safely circumvent future collisions. Use matrix Q_{coll} with reduced diagonal values with respect to Q_{agg}.

Although this approach helps in some cases, in others the conservativeness of the agents is counterproductive to avoid collisions. In those cases, (lines 11-14) a more aggressive behaviour during predicted collisions is enforced, and the algorithm tries to solve the problem with the new matrix Q_{coll}. In some other cases, the conservative behaviour can avoid collisions but is unable to reach the waypoint within the simulation duration T, in those cases a more aggressive behaviour is also induced to the agents and the problem is resolved (lines 17-19).

Convergence of Alg. 1 is declared when the trajectories for all agents respect all the constraints, and the point-to-point transition was completed within T seconds. If the algorithm coverged, the last step (lines 20-21) is to interpolate the resulting trajectory to obtain a better resolution (say, time step $T_s = 0.01s$).

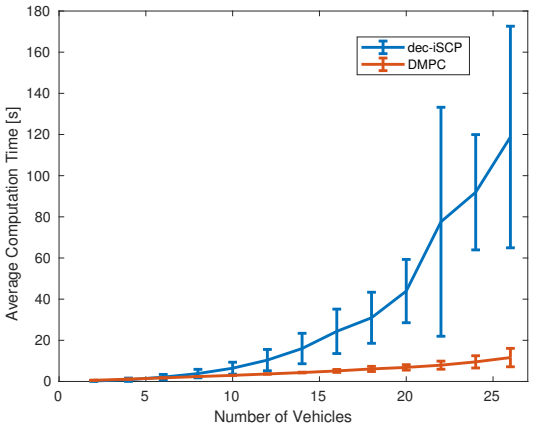

Fig. 3. bla bla

IV. Main Results

In this section, simulation analysis of the DMPC algorithm is provided. All the algorithms were implemented in MATLAB2017A and executed on a PC with an Intel Xeon CPU and 16GB of RAM. Table 1 shows the simulation parameters.

A. Four corner exchange

To illustrate how DMPC is able to manage colliding trajectories.

V. Conclusions

References

[1] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE spectrum*, vol. 45, no. 7, pp. 26–34, 2008.
[2] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *European Control Conference (ECC)*. IEEE, 2001, pp. 2603–2608.
[3] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 1917–1922.
[4] M. Turpin, N. Michael, and V. Kumar, "Decentralized formation control with variable shapes for aerial robots," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 23–30.
[5] S. Boyd, "Sequential convex programming," *Lecture Notes, Stanford University*, 2008.
[6] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5954–5961.
[7] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems*, vol. 22, no. 1, pp. 44–52, 2002.
[8] 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, 2017.
[9] R. Van Parys and G. Pipeleers, "Distributed model predictive formation control with inter-vehicle collision avoidance," in *Proceedings of the 2017 Asian Control Conference*, 2017.