Robust Trajectory Execution for Multi-Robot Teams Using Distributed Real-time Replanning

Baskın Şenbaşlar Wolfgang Hönig Nora Ayanian

I. INTRODUCTION

Motion planning for multi-robot systems is particularly important in cases where many robots must interact with each other in confined spaces, potentially with many obstacles. Modern planning algorithms can find trajectories that effectively coordinate hundreds of robots while approximately optimizing objectives such as total energy used [2]; however, all such solutions assume that the resulting trajectories can be executed nearly perfectly, which is an unrealistic assumption.

To compensate for the changes in the environment or imperfect execution, one might apply cooperative collision avoidance strategies, such as ORCA [3], at runtime. However, such algorithms often operate locally and do not take the pre-planned trajectories or dynamic limits of the robots into account.

We propose an algorithm for robust trajectory execution that compensates for a variety of dynamic changes, including newly appearing obstacles, robots breaking down, imperfect motion execution, and external disturbances. Consider the example in Fig. 1(a), where two robots must swap positions. The pre-planned trajectories are collision-free, but they do not consider the newly introduced obstacle and the blue robot does not start at its correct location. However, the pre-planned trajectories can be used as guidance for replanning. In this example, robots can get stuck if a local cooperative collision avoidance strategy is applied. Using our method, the robots can successfully swap positions, while staying as close as possible to the pre-planned trajectories, as in Fig. 1(b). Our method is fully distributed and requires no communication. The robots only need to know their own trajectories and be able to sense other robots' positions and the obstacles around them.

II. PROBLEM FORMULATION

Consider a group of m robots. Each robot i is given the following:

$\mathbf{o}_i(t)$: original trajectory of i^{th} robot where $t \in [0, T_i]$,
c : order of derivative up to which smoothness is required,

This paper is an extended abstract of work accepted at DARS 2018 [1]. This research was supported in part by Office of Naval Research grant N00014-14-1-073 and National Science Foundation grant 1724399. B. Şenbaşlar gratefully acknowledges the support from the Fulbright program sponsored by U.S. Department of State. All authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA. Email: {baskin.senbaslar, whoenig, ayanian}@usc.edu

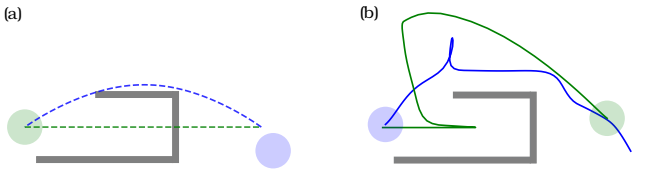

Fig. 1. (a) Two robots (green and blue circles) are tasked with following their pre-planned trajectories (green and blue dashed lines). The initial plans were created without the knowledge of the obstacle (gray) and the blue robot does not start at its planned initial position. (b) Our approach computes smooth trajectories in real-time, avoiding both the new obstacle and other robots while staying close to the pre-planned trajectory.

$R(\mathbf{p})$: convex collision shape of any robot at position $\mathbf{p}$,
γ_k : dynamic limit of the robot for the k^{th} derivative of its trajectory.

Each robot i can sense the positions $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$ of the other robots as well as the current occupied space $\mathcal{O}_i$ around it. Robots are unaware of the other robots' planned trajectories, and cannot communicate with each other. Each robot i must execute a trajectory $\mathbf{f}_i(t)$, where $\mathbf{f}_i(t)$ is a solution to the following optimization problem:

$$\text{minimize} \int_0^{T_i} \|\mathbf{f}_i(t) - \mathbf{o}_i(t)\|^2 \, dt \tag{1}$$

subject to

$$\mathbf{f}_i(t) \text{ is continuous up to degree } c,$$
$$\frac{d^j \mathbf{f}_i}{dt^j}(0) = \frac{d^j \mathbf{p}_i}{dt^j}(0) \text{ for } j \in \{0, 1, ..., c\}$$
$$\mathbf{f}_i(t) \text{ is collision-free, and}$$
$$\left\| \frac{d^k \mathbf{f}_i(t)}{dt^k} \right\| \leq \gamma_k \text{ for all desired } k,$$

where $t \in [0, T_i]$.

III. APPROACH

We solve this problem approximately, using a dynamic receding horizon approach iteratively. At every iteration K, robot i plans a trajectory $\mathbf{f}_i^K(t)$ that starts at the robots' current position and is safe to execute up to the user-provided period δt.

Replanning is done at a fixed period of δt. In each iteration K, we sense the other robots' positions to compute the buffered Voronoi cell $\mathcal{V}_i$ [4], update our current representation of the occupied space ($\mathcal{O}_i$) and compute support vector machine hyperspaces to define safe space according to obstacles, and compute a trajectory $\mathbf{f}_i^K(t)$.

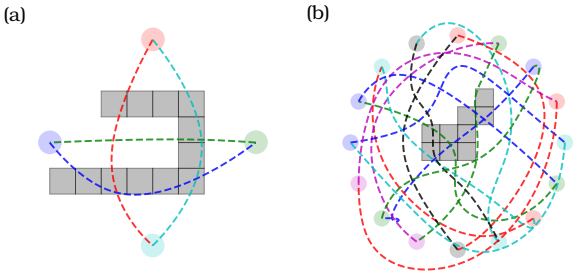

Fig. 2. The original trajectories and the occupancy grids in some of our simulation setups.

TABLE I
COMPARISON OF OUR METHOD, ORCA, AND DS+ORCA WITH RESPECT TO AVERAGE COMPUTATION TIME (t_{AVG}), AND THE NUMBER OF ROBOTS THAT REACH THEIR DESTINATIONS (s). m DENOTES THE NUMBER OF ROBOTS, AND θ DENOTES THE NUMBER OF OCCUPIED CELLS IN THE ENVIRONMENT.

		ORCA		DS+ORCA		Our Method	
m	θ	t_{avg} [ms]	s	t_{avg} [ms]	s	t_{avg} [ms]	s
2	4	< 1	0	< 1	2	7	2
4	12	< 1	0	< 1	4	10	4
8	30	< 1	4	< 1	8	13	8
16	9	< 1	13	< 1	16	12	16
32	30	< 1	23	< 1	32	16	32

We execute the following three major components iteratively: *discrete planning* that is used to efficiently plan around new obstacles and robots, *trajectory optimization* that is formulated as a convex QP to generate smooth and collision-free trajectories, and *temporal rescaling* to enforce the dynamic limits of the robot.

Given a user defined time horizon τ, we execute discrete planning if i) the original trajectory is not collision free for the time horizon τ, or ii) the previously planned trajectory is outside of the robot's buffered Voronoi cell for the period δt, or iii) the previously planned trajectory is not collision free for the time horizon τ. The first condition handles cases where previously unknown obstacles block the pre-planned trajectory of a robot. The second condition handles cases where previously unknown robots appear and cases where robots are close and moving towards each other. The third condition handles dynamic obstacles.

The discrete planning component is used to give a good initial guess for trajectory optimization. If it is not used, the previous plan is used as the initial guess. During trajectory optimization, we plan for a piecewise Bézier curve such that the curve is inside the robot's buffered Voronoi cell for up to time δt, is inside the safe-space defined by support vector machine hyperspaces calculated between obstacles and trajectory pieces, continuous up to a user defined degree of derivative, and minimizes a cost function that balances energy usage and deviation from the original trajectory. When each robot obeys to the rule that the trajectory stays inside the robot's buffered Voronoi cell up to time δt, no robot to robot collision can occur for that period. This is the cooperative aspect of our approach.

During temporal rescaling, we check if the trajectory violates the dynamic limits of the robot, and if it does, we increase the duration of the curve, and re-run trajectory optimization.

At the end of each iteration, each robot has its trajectory $\mathbf{f}_i^K(t)$ that is guaranteed to be collision-free up to time δt; is continuous up to user defined c^{th} derivative; obeys the dynamic limits of the robot; tries to stay close to the original trajectory; and is a good starting point for the next iteration.

IV. EVALUATION

First, we analyze our approach in simulation using differential drive robots in environments that contain unknown obstacles. Some of our simulation setups are given in Fig. 2. We run our simulations on a laptop computer (i7-4700MQ 2.4 GHz, 16 GB) with Ubuntu 16.04 as operating system. Assuming 10 Hz as replanning frequency, the results of our scalability tests suggest that our algorithm: i) runs in realtime when using up to 12 trajectory pieces (we found that 4 pieces are sufficient in all our test cases), ii) is not significantly affected by the number of robots, and iii) can handle several hundreds of occupied cells. We also compare our method with two variants of ORCA. In the first variant, we use ORCA as is. In the second variant, which is denoted as DS+ORCA, we combine ORCA with our discrete search method. As Table I suggests, all robots using our method or DS+ORCA reach their destinations. Our method takes more time in computation compared to both ORCA variants, but produces continuous curves up to a user defined derivative degree while ORCA and DS+ORCA are continuous only in position.

Second, we implement our approach on six differential drive robots (iRobot Create2) that are equipped with one of ODROID C1+ or ODROID XU4 single-board computers. We conduct several experiments and add an additional obstacle, change the robots' initial positions, disturb the robots during run-time, or artificially stop one of the robots. In all cases robots successfully avoid collisions and in many cases they reach their final destination within the originally planned durations. The setup for physical experiments can be found in our supplemental video at `https://youtu.be/LbWRvLfdwTA`.

An implementation of our algorithm is available online at `http://github.com/baskinburak/mrtrajreplan-dars2018`. Currently, we are working on implementing our approach for UAVs in 3D environments.

REFERENCES

[1] 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.
[2] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE T-RO, Special Issue on Aerial Swarm Robotics*, 2018.
[3] J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha, "Reciprocal *n*-body collision avoidance," in *ISRR*, 2009, pp. 3–19.
[4] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE RA-L*, vol. 2, no. 2, pp. 1047–1054, 2017.