Decentralized Multi-Agent Trajectory Planning in Dynamic Environments with Spatiotemporal Occupancy Grid Maps

Siyuan Wu, Gang Chen, Moji Shi, and Javier Alonso-Mora

Abstract— This paper proposes a decentralized trajectory planning framework for the collision avoidance problem of multiple micro aerial vehicles (MAVs) in environments with static and dynamic obstacles. The framework utilizes spatiotemporal occupancy grid maps (SOGM), which forecast the occupancy status of neighboring space in the near future, as the environment representation. Based on this representation, we extend the kinodynamic A* and the corridor-constrained trajectory optimization algorithms to efficiently tackle static and dynamic obstacles with arbitrary shapes. Collision avoidance between communicating robots is integrated by sharing planned trajectories and projecting them onto the SOGM. The simulation results show that our method achieves competitive performance against state-of-the-art methods in dynamic environments with different numbers and shapes of obstacles. Finally, the proposed method is validated in real experiments.

I. INTRODUCTION

Decentralized multi-agent trajectory planning (MATP) problem in dynamic environments remains a challenging problem for years. To solve this problem, the robot must avoid collisions with static and dynamic obstacles during planning, as well as with other MAVs. A common approach to planning in dynamic environments inherits the pipeline used in static environments with modifications to account for dynamic obstacles [1], [2]. These methods rely on an Occupancy Grid Map (OGM) to represent static obstacles and generate a collision-free trajectory for each MAV [3]. The trajectory is then further optimized to avoid dynamic obstacles and other robots, which are tracked and represented separately with fixed-shape models, e.g., ellipsoids [3], [4], columns [5] and axis-aligned bounding boxes [6].

However, those methods require distinct pipelines for static and dynamic obstacles and other robots, often leading to trajectories that avoid dynamic obstacles but conflict with static ones or other robots. To address this issue, we use Spatiotemporal Occupancy Grid Maps (SOGM) [7], [8] to capture both the current and future occupancy status of arbitrary-shaped static and dynamic obstacles in the environment. Moreover, we incorporate the occupied space of other robots within the same time window to achieve a unified representation of both obstacles and other robots. Based on
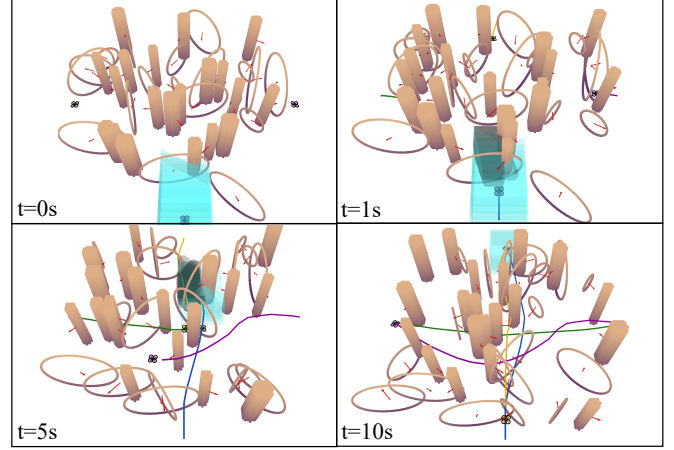
Fig. 1: 4 robots navigate in a dynamic environment with 20 columns and 20 circles. All robots perform independent planning and coordinate with trajectory sharing. The spatiotemporal corridors are indicated in blue, and obstacle velocities are marked by red arrows.

this representation, we employ spatio-temporal safety corridors to identify areas free from obstacles and other robots, effectively solving the MATP in dynamic environments.

In this work, we assume that the SOGM of each robot is given, and extend a corridor-based trajectory optimization method to MATP in dynamic environments. First, a kinodynamically feasible path is searched within the map, and spatio-temporal corridors are generated to maximize the free space. Next, trajectories are generated by solving a minimum jerk trajectory optimization problem with the spatio-temporal corridors as constraints. To prevent collisions between robots, other robots are treated as moving obstacles with known shapes, and their trajectories are shared via a communication network and represented simultaneously in the SOGM. For dynamic obstacles, such as robots that do not communicate, we employ a constant velocity prediction. Compared to existing MATP methods, our planner is capable of safely navigating multiple MAVs in complex environments with arbitrarily shaped static and dynamic obstacles. Our contributions are summarized as:

1) A decentralized multi-robot collision avoidance framework that extends the corridor-based methods used in static environments to dynamic environments.
2) An efficient planning approach that generates safe trajectories in the SOGM and is capable of avoiding static and dynamic obstacles with arbitrary shapes.
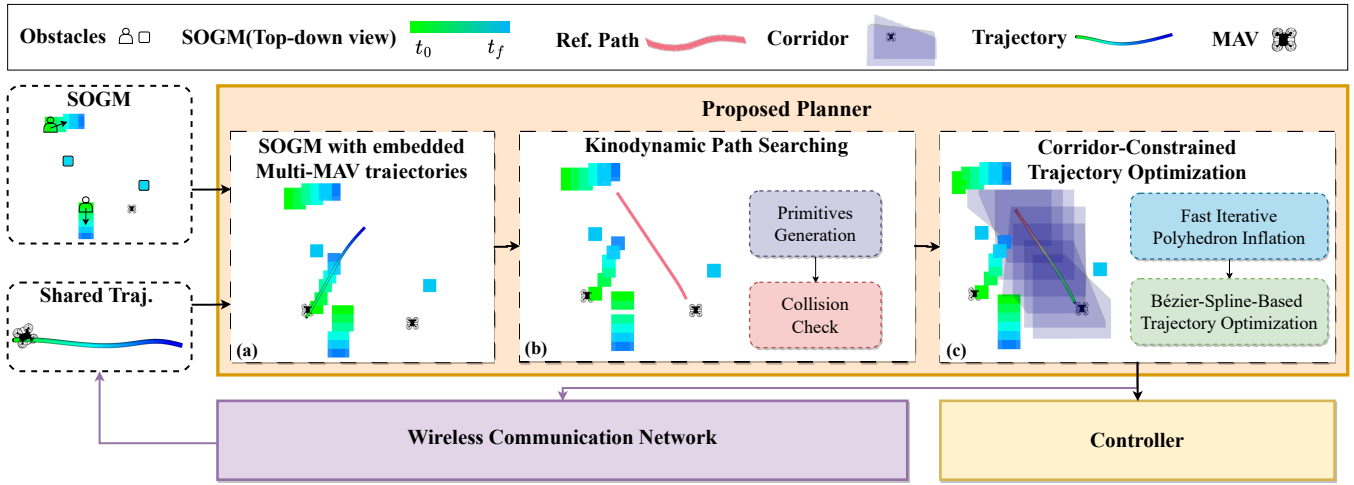3) A paradigm that integrates shared trajectories into

Fig. 2: Overview of the proposed framework for decentralized multi-robot trajectory planning.

occupancy grids to build corridors free from obstacles and other robots.

We will release our code for the community's reference.

II. RELATED WORK

A. MAV Planning in Dynamic Environment

MAV trajectory planning methods in dynamic environments can be divided into two categories, depending on whether they consider static and dynamic obstacles separately or jointly. The first category [3], [5], [9] usually employs an OGM [10], [11] to model static obstacles, taking advantage of its ability to represent arbitrary 3D shapes [12]. However, dynamic obstacles are separately represented using a fixed-shape model with predicted trajectories, including ellipsoids [13], [4], [5], spheres [14], or bounding boxes [6]. Their future trajectories are normally predicted by polynomial fitting methods [15] or the constant velocity model [16]. This representation of dynamic obstacles narrows the free space and increases the difficulty of trajectory generation by introducing non-convex collision avoidance constraints. Static and dynamic obstacles can be represented jointly. [17], [2] represents static and dynamic obstacles with a particle-based occupancy map [8] and uses a sampling-based method to find a trajectory that encounters fewer particles. Further in [18], risk-aware spatio-temporal corridors are generated on the map for trajectory optimization. The trajectory generation method in [18] is limited to cubic corridors, which can be overly conservative, especially in environments with arbitrarily shaped obstacles. Inspired by [18], this work aims to extend these corridor-based methods to solve the MATP problem by addressing the aforementioned limitations.

B. Decentralized Multi-Agent Trajectory Planning

Multi-agent trajectory planning in complex environments is a challenging task in aerial robotics. A few methods have been proposed to solve this problem in a decentralized manner for scalability and robustness [19], [20]. In these methods, robots usually coordinate with each other by sharing their current states or planned trajectories. With the shared states, collision-free trajectories can be generated by constructing Buffered Voronoi Cells [20] or distributed MPC [21], which provide better safety guarantees but inhibit flight speed. When using shared trajectories, it becomes possible to plan more agile routes. For example, MADER [6] and Robust-MADER [22] broadcast planned trajectories through a communication network, and propose a check-recheck paradigm to avoid collisions with other robots. EGO-Swarm [3] uses a reciprocal barrier cost to generate collision-free trajectories by solving unconstrained optimization problems. In [23], shared trajectories are used to generate linear relative safety corridors as hard constraints. In this work, we depict the shared trajectories in SOGM, facilitating simultaneous collision avoidance for robots and obstacles with the spatio-temporal safety corridors.

III. METHODOLOGY

This section first briefly introduces how we use the SOGM in the MATP problem and describes the paradigm that integrates shared trajectories into SOGM (Sec. III-A). Then a kinodynamic path searching method designed for the SOGM is presented (Sec. III-B). Finally, we describe the method that generates corridors from the reference path and introduce the trajectory optimization method to generate collision-free trajectories within these corridors (Sec. III-C).

A. SOGM with Shared Trajectories Embedded

Various methods have been proposed to construct SOGMs from the point cloud, such as filtering-based methods [8] and learning-based methods [7], [24]. The resulting SOGM is a 4D OGM with spatial resolution r_s and temporal resolution r_τ. In this work, we assume that the local SOGM denoted by $\mathbb{M}_k$ in the frame of robot $k \in \{1, 2, \ldots, K\}$ with T-step predictions, which is generated by the perception algorithm in real-time (Fig. 2a). $\mathbb{M}_k(i)$ denotes the i-th frame of the SOGM $\mathbb{M}_k$, which is an OGM between t_{i-1} and t_i.

In the MATP problem, collision avoidance between non-cooperative obstacles and other cooperative robots is challenging. The difficulty primarily arises from the distinct representations of the two types of obstacles: Non-cooperative obstacles are represented by the SOGM, while cooperative robots are characterized by their trajectories. It requires two independent pipelines to achieve collision avoidance, which is inefficient and sometimes intractable because a collision-free trajectory optimized in one pipeline might be infeasible in another pipeline. Therefore, the published trajectories of cooperative robots are projected in the SOGM by accounting for the occupied grids of the robots from one time step to the next. Assume that robot k receives shared trajectories $\boldsymbol{p}(t)_{k'}$ and occupancy $\mathcal{O}_{k'}$ from other robots $k' \in \{1, 2, \ldots, K\} \backslash k$. Then the occupancy of robot k' traveled from t_{i-1} to t_i is represented as

$$\mathcal{O}_{k,k'}[t_{i-1}, t_i] = \int_{t_{i-1}}^{t_i} \mathcal{O}_{k'} \oplus \boldsymbol{p}(t)_{k'} dt \quad (1)$$

which the corresponding 3D spaces occupied in the SOGM $\mathbb{M}_k$ as $\mathcal{O}_{k'}$ moves along trajectory $\boldsymbol{p}(t)_{k'}$ in the local frame of robot k, as shown in Fig. 2a. Therefore, the SOGM considering cooperative robots can be represented as

$$\mathcal{M}_k(i) = \left(\bigcup_{k' \in \{1,2,\ldots,K\}\backslash k} \mathcal{O}_{k,k'}[t_{i-1}, t_i] \right) \cup \mathbb{M}_k(i). \quad (2)$$

$\mathcal{M}_k(i)$ will be updated applying the recent $\boldsymbol{p}(t)_{k'}$ once $\mathbb{M}_k(i)$ is updated. For the initial robot, $\mathcal{M}_1(i) = \mathbb{M}_1(i)$ for all $i \in \{1, 2, \ldots, T\}$ since no other trajectories are published.

B. Kinodynamic Path Searching in SOGMs

We develop a path-searching method based on the kinodynamic algorithm A* [25] to find a kinodynamically feasible reference path (as shown in Fig. 2b). Similar to [25], we select the double integrator as the approximate model and generate a set of primitives by discretizing the control inputs of the robot. The primitive duration is set to the temporal resolution r_τ of the SOGM. The cost function is defined as the cumulative square of control inputs over the time horizon. The heuristic function is computed by applying Pontryagin's minimum principle [25] which yields the optimal control input to reach the goal state. Since this function is both admissible and consistent, optimality is guaranteed. In many path-searching methods, obstacles are pre-inflated by the robot radius to facilitate feasibility checks. Due to the high dimension of the SOGM, inflating the obstacles with the robot radius directly on the map is not efficient. Instead, we first inflate the primitive with the shape of the robot and then check if the inflated primitive collides with obstacles in the corresponding temporal frame $\mathcal{M}_k(i)$.

C. Corridor Constrained Trajectory Generation

We develop a trajectory optimization method for the MATP problem to generate collision-free trajectories constrained by spatio-temporal corridors inspired by [18].

1) Spatiotemporal Corridor Generation: Upon determining a collision-free reference path in the SOGM, our objective is to delineate the obstacle-free region along this path before proceeding with trajectory generation. Due to the dynamic nature of the environment, the safety region of the path may change with time, so the static corridor introduced in [26] is no longer applicable. A natural approach is to locate the safety region at each temporal frame $\mathbb{M}_k(i)$ of the SOGM. In this work, the spatio-temporal corridor is defined as a convex polyhedron $\mathcal{P} = \{x \in \mathbb{R}^n | \boldsymbol{A}_{\mathcal{P}} x \leq \boldsymbol{b}_{\mathcal{P}}\}$ that encloses the safety region with the time window $[t_{i-1}, t_i]$. The time duration of each corridor is determined by the time resolution r_τ of the SOGM.

The problem of finding the obstacle-free convex corridor constraints for robot R_k at temporal frame $\mathbb{M}_k(i)$ is formulated as the following optimization problem:

$$\max_{\boldsymbol{A}_{\mathcal{P}}, \boldsymbol{b}_{\mathcal{P}}} \text{vol}(\mathcal{P}), \text{ s.t. } R_k \subset \mathcal{P}, \mathcal{O}_{k,i} \subset \mathbb{M}_k(i) \backslash \mathcal{P}, \quad (3)$$

To efficiently solve the optimization problem, we apply Fast Iterative Region Inflation (FIRI) [27]. FIRI approximates the maximum volume polyhedron $\mathcal{P}$ by iteratively and monotonically increasing the volume of its inscribed ellipsoid. It formulates the problem as an equivalent second-order conic programming (SOCP) problem to decrease the dimension of the problem and reduce the computational cost. Thanks to its efficiency, we can generate a sequence of feasible spatiotemporal corridors at $\mathbb{M}_k$ in less than 5ms.

Given that the corridor produced by FIRI solely encompasses the obstacle-free region without accounting for the robot's size, it is essential to shrink the corridor's size to guarantee that the robot can navigate through. This reduction is achieved by inwardly adjusting all the corridor edges by the robot's radius, d. Following this adjustment, a linear programming check is employed to confirm the corridor remains feasible and that consecutive corridors are interconnected.

2) Minimum Jerk Trajectory Optimization: We employ the Bézier spline to parametrize the trajectory. A Bézier spline is a n-th order piecewise polynomial defined as:

$$\boldsymbol{p}_j(t) = \sum_{i=0}^{n} \boldsymbol{c}_j^i b_n^i(t - t_{j-1}), \quad t \in [t_{j-1}, t_j), \quad (4)$$

where $\boldsymbol{c}_j^i$ denotes the i-th control point at the j-th piece of Bézier spline. $b_n^i(t) = \binom{n}{i} t^i (1-t)^{n-i}$ is the Bernstein basis. The Bézier spline carries an important property: the convex hull property, which guarantees that the trajectory is entirely inside the convex hull of the control points. Therefore, we can confine the j-th piece of trajectory $\boldsymbol{p}_j(t)$ within the spatio-temporal corridors $\mathcal{P}_j$ by constraining the corresponding control points $\{\boldsymbol{c}_j^0, \boldsymbol{c}_j^1, \cdots, \boldsymbol{c}_j^n\}$ as follows:

$$\boldsymbol{A}_{\mathcal{P}_j} \boldsymbol{c}_j^i \leq \boldsymbol{b}_{\mathcal{P}_j}, \quad i \in \{0, 1, \cdots, N\}, \quad (5)$$

By taking the above properties, we can formulate a minimum jerk trajectory optimization problem with spatiotemporal corridor constraints as follows:

$$\min_{\boldsymbol{c}} \sum_{j=1}^{T} \int_{t_{j-1}}^{t_j} \left\| \frac{d^3 \boldsymbol{p}_j(t)}{dt^3} \right\|^2 dt \quad (6a)$$

$$\text{s.t.} \quad \boldsymbol{p}(t_0) = \boldsymbol{p}_0, \quad \boldsymbol{p}(t_T) = \boldsymbol{p}_T \tag{6b}$$

$$\boldsymbol{p}_j(t) \in \mathcal{P}_j, \forall t \in [t_{j-1}, t_j], \quad j = 1, 2, \cdots, T \tag{6c}$$

$$\boldsymbol{p}_j(t_j) = \boldsymbol{p}_{j+1}(t_j), \quad j = 1, 2, \cdots, T \tag{6d}$$

$$\boldsymbol{p}_j^{(1)}(t) \le \boldsymbol{p}_{max}^{(1)}, \ \boldsymbol{p}_j^{(2)}(t) \le \boldsymbol{p}_{max}^{(2)}, \ \forall t \in [t_0, t_T], \tag{6e}$$

in which $\boldsymbol{p}_0$ and $\boldsymbol{p}_T$ denote the initial and terminal position of the trajectory, and $\boldsymbol{p}_{max}^{(1)}$ and $\boldsymbol{p}_{max}^{(2)}$ give the upper bound of the velocity and acceleration. Note this is a quadratic programming (QP) problem with linear constraints, which can be solved efficiently by modern QP solvers, e.g. OSQP.

In our method, the time allocation for trajectories is established during the path search phase, as opposed to emerging from the backend optimization. This strategy is crucial to ensure the safety of the generated trajectories. As a result, our backend optimization prioritizes spatial optimization with the temporal allocation fixed. We intentionally avoid a comprehensive spatio-temporal optimization to avoid the computational burden of solving a non-convex optimization problem. By employing this approach, we not only address the collision avoidance requirements but also achieve a more lightweight backend optimization.

D. Multi-Agent Deconfliction

The deconfliction is considered to trigger the replanning of each agent. Due to the Bézier spline's convex hull property, we check the linear separability between the control points of the optimized trajectory and those of the broadcasted trajectory. Specifically, if the check step identifies a potential collision, the agent will update the SOGM and re-optimize a new trajectory; Conversely, if no collision is detected, the agent will broadcast and execute the current trajectory.

IV. RESULTS

We compared the performance of the proposed method with MADER [6] and EGO-Swarm [3] in simulations of 4 MAVs varying in dynamic obstacles' density and shape. We then conducted real-world experiments with 2 MAVs in a complex environment with static obstacles and moving pedestrians to demonstrate the effectiveness of our approach.

A. Simulation Experiment

The 3D simulation experiment was conducted on a laptop equipped with an AMD R7-5800H CPU. The simulation environment is a 16m×16m×4m 3D space with randomly generated moving columns and circles, implemented based on a quadrotor simulator in [25]. Dynamic obstacles are modeled as columns and circles of random sizes, initial positions, and velocities. The widths of the columns and the radius of the circles are uniformly sampled from 0.5 to 1.0 m and 0.7 to 2.5 m, respectively. The height of all the columns is 4.0 m, and the width of the circles is fixed at 0.1 m. The velocities of the obstacles are randomly sampled from 0 to 1.0 m/s, and the directions are uniformly sampled from 0 to 2π. Obstacles move with constant velocities and directions, and they rebound when they reach the boundary of the map space. All MAVs are initialized in the hovering state at a

height of 1.0 m, outside the obstacle space. Their objectives are reaching their individual goals without any collision.

The experiments are divided into two types of environments with varying difficulty. The first type is a **mixed environment** with columns and circles (Fig. 3d–3f), while the second type is a **pure column environment** (Fig. 3g–3i) similar to that used in MADER [6]. Fig. 3a–3c presents a bird-eye view of these settings in a mixed environment with 20 columns and 20 circles. This setup aims to create a more cluttered environment by breaking the convexity and isotropy of the obstacles. Robots can choose either side to avoid columns; however, for circles, they must plan a 3D trajectory to avoid the edges. Note that robots are allowed to pass through circles from center to center. To quantify the difficulty, Table I compares the average obstacle density in both environments that varies in the number of obstacles. The obstacle density is defined as the ratio of the total volume occupied by obstacles to the volume of the space.

We compare our proposed method against two state-of-the-art baselines, MADER [6] and EGO-Swarm [3], across three distinct position swap tasks: bilateral (Fig. 3a), unilateral (Fig. 3b), and cross (Fig. 3c and Fig. 5). Unlike MADER and EGO-Swarm, which necessitate precise shapes and exact trajectories of obstacles, our method employs SOGMs. To ensure a fair comparison, all methods are confined to using local obstacle information, namely the ground truth positions and velocities within a 5.0 m radius over a time horizon of 2.0 s. SOGMs are specifically generated for our approach. The maximum velocity and acceleration are set to 2.0 m/s and 6.0 m/s^2, respectively. We assume the perfect tracking control in simulation. More than 50 trials are conducted in each setting. Average results are reported in Fig. 3.

The performance is compared in terms of the failure rate and average time to complete the task. As shown in Fig. 3d-3i, the proposed planner presents 95.3% and 88.5% average success rates in the mixed and pure environments when the number of obstacles is less than 30. This result is comparable to the 87.3% and 90.8% success rates of EGO-Swarm, and better than the 87.5% and 84.8% success rates of MADER. In mixed environments with more than 30 obstacles, our method achieves a success rate of 73.0% between all tasks compared to 45.2% and 21.0% for EGO-Swarm and MADER, respectively. However, in pure environments with more obstacles, our method performs worse than both baselines in success rate and flight time.

To better understand the reason for the failure, we distinguish failures into two different types: collision and deadlock. Collision only occurs when the agent is following a planned trajectory. Deadlock is defined as a situation where a dynamic obstacle hits an agent after the agent has stopped due to the inability to find a feasible trajectory. It reflects the inability to find a feasible trajectory when the

Num. of Obstacles	10	20	30	40	50
Pure column environment (%)	1.73	3.45	5.18	7.00	8.62
Mixed environment (%)	0.86	1.73	3.71	4.53	5.76

TABLE I: Average obstacle density in different environments

(a) Task 1: Bilateral swap   (b) Task 2: Unilateral swap   (c) Task 3: Cross swap

(d) Bilateral swap in mixed environments   (e) Unilateral swap in mixed environments   (f) Cross swap in mixed environments

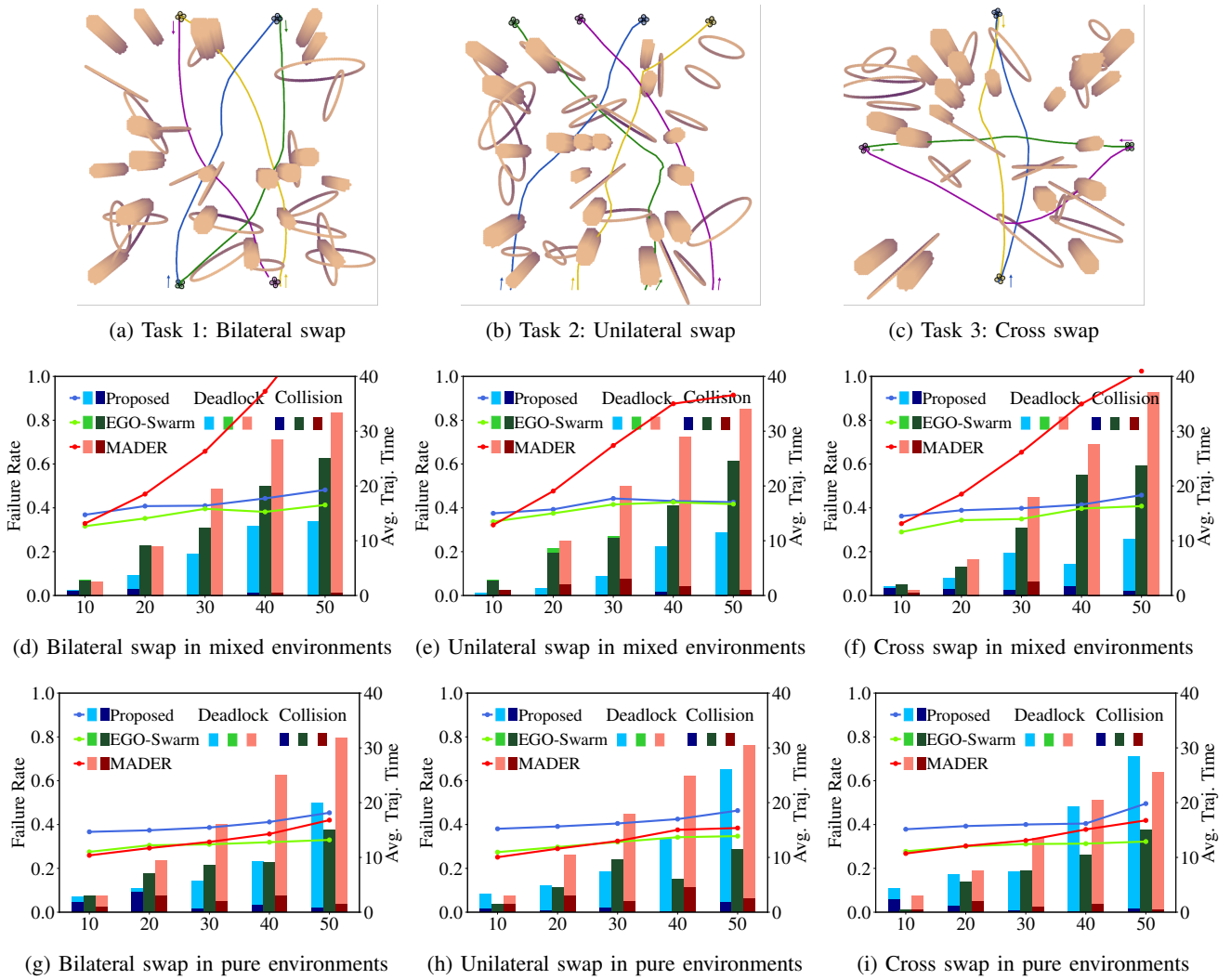(g) Bilateral swap in pure environments   (h) Unilateral swap in pure environments   (i) Cross swap in pure environments

Fig. 3: Performance comparison between our proposed method (Ours), and two baselines (MADER and EGO-Swarm) in 3 different tasks (Fig. 3a–3c) and 2 different obstacle settings: mixed environments with both columns and circles (Fig. 3d–3f), and pure column environments (Fig. 3g–3i). In Fig. 3a–3c, trajectories for each MAV are displayed with distinct colors. Fig. 3d–3i evaluate the average flight time to the goal and failure rate with different numbers of obstacles, distinguishing failure types between deadlock and collisions.

search space is exhausted. In many real-world scenarios, dynamic obstacles such as pedestrians may avoid the robots in a "deadlock" situation. As shown in Fig. 3d to Fig. 3f, deadlock accounts for almost all failures in our method. In contrast, EGO-Swarm hardly encounters deadlock, but it has a higher collision rate than other methods. Safety arises because our method confines the planned trajectory within the corridors. Furthermore, our method performs better in mixed environments than in pure environments. As depicted in Fig. 3d-3g, a significant decrease in the failure rate can be seen in environments with more obstacles. These results reveal that the ability of the SOGM to represent obstacles with arbitrary shapes allows for greater flexibility in finding feasible paths in such environments. Although it is feasible to dissect these obstacles into smaller elements and represent them using AABB or ellipsoidal costs, doing so increases the

computational burden, making it challenging to achieve real-time performance. Therefore, our method is more suitable for such environments.

The run time of each step is evaluated on a laptop with an AMD R7-5800H CPU in mixed environments, shown in Fig. 4. The overall execution time of our method is 17.19 ms on average, which is significantly faster than MADER [6] (31.04 ms). The average run time of the kinodynamic path searching step is 1.93 ms, increasing with the number of obstacles. This is because kinodynamic A* algorithm will traverse more nodes to find a feasible path as the obstacle density increases as in Table I. The corridor generation step takes 3.47 ms on average. The trajectory optimization step takes an average of 11.79 ms, which is time-consuming due to the large number of polygon planes as corridor constraints.
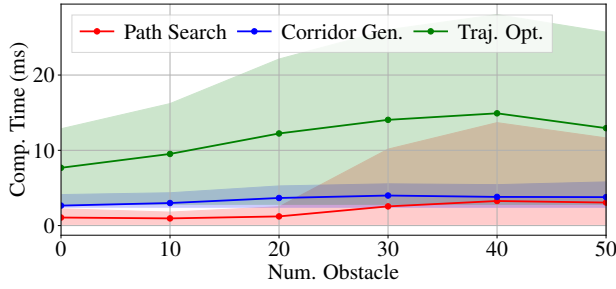
Fig. 4: The computation time of each step in our method under different obstacle settings.
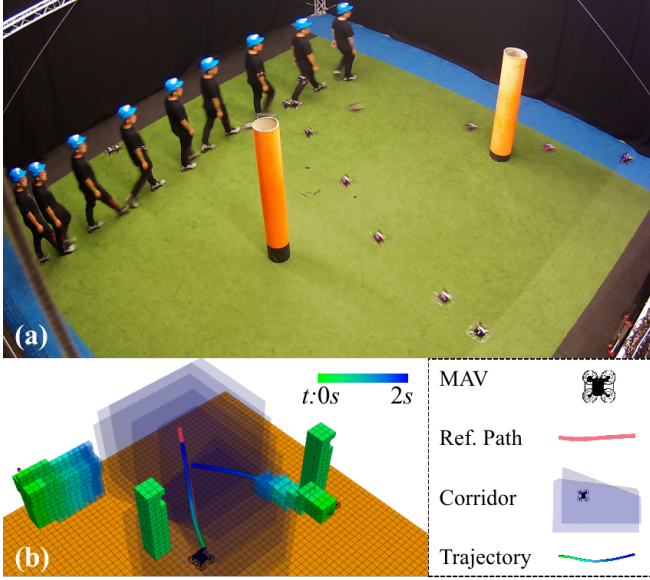

Fig. 5: (a) Composite image of two MAVs flying in an indoor dynamic environment. (b) The SOGM at $t = 2.5s$ is displayed, showing the predicted future occupancies with a gradient transition from green to blue.

B. Real-World Experiment

We conducted real-world experiments in a 10m×10m×4m indoor environment to validate the efficiency of the proposed method, shown in Fig. 6. Each robot is equipped with an NVIDIA Jetson Xavier NX for onboard computation. Communication between robots is achieved via a WiFi network. Considering the limited onboard computational resources, we rely on the OptiTrack system to provide ground-truth information on dynamic obstacles to build the local SOGM. The robot localization is provided by the OptiTrack as well, due to the limited accuracy and robustness of the onboard localization algorithm in dynamic scenes. The maximum velocity and acceleration of the robot are set to 1.0 m/s and 4.0 m/s^2. The planner runs at 10 Hz. Experiment settings and the planning results are visualized in Fig. 5 and Fig. 6. For more details, please refer to the supplementary video.

V. CONCLUSION

This paper presents a decentralized multi-agent trajectory planning framework that is able to handle environments with arbitrarily shaped static and dynamic obstacles. Our
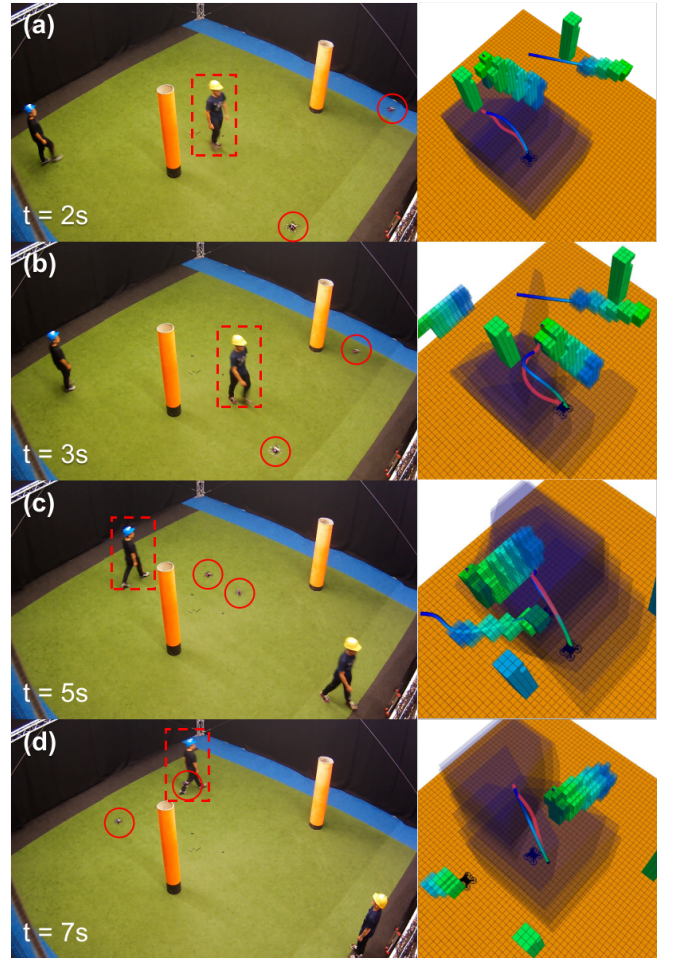

Fig. 6: Snapshot of two MAVs flying in a dynamic environment with two pedestrians. Images on the right visualize the SOGM, safety corridors, and planned trajectories.

planner adopts the kinodynamic A* path-searching algorithm on the SOGM, allowing the robots to find a feasible path in such environments. The corridor generation and trajectory optimization methods efficiently help generate safe and dynamically feasible trajectories. Simulation results show that our method achieves more than 88.5% success rate in dynamic environments with less than 20 obstacles, which is comparable to both baselines. In dynamic environments where obstacles cannot be represented by AABB or ellipsoids, our method outperforms the baselines by achieving a 51.9% and 27.7% lower failure rate than MADER and EGO-Swarm, respectively. By classifying the failure cases into collision and deadlock, we find that the majority of our failures are due to deadlock cases. Real-world experiments further validate the effectiveness of the proposed method. However, in environments dense with obstacles or when there are numerous robots, this method is susceptible to significant deadlock issues. This can be attributed to the absence of global guidance and the loss of topological homotopy under frequent replanning. In future research, we will focus on multi-agent path finding with topological homotopy to mitigate these deadlock issues.

REFERENCES

[1] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous Flights in Dynamic Environments with Onboard Vision," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1966–1973.

[2] G. Chen, W. Dong, X. Sheng, X. Zhu, and H. Ding, "An active sense and avoid system for flying robots in dynamic environments," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 2, pp. 668–678, 2021.

[3] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022/05/11.

[4] H. Zhu and J. Alonso-Mora, "Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 4, no. 2, pp. 776–783, 2019-04.

[5] Z. Xu, D. Deng, Y. Dong, and K. Shimada, "DPMPC-Planner: A real-time UAV trajectory planning framework for complex static environments with dynamic obstacles," in *2022 Int. Conf. Robot. Autom. ICRA*, 2022, pp. 250–256.

[6] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Trans. Robot. (T-RO)*, vol. 38, no. 1, pp. 463–476, 2022.

[7] H. Thomas, M. G. de Saint Aurin, J. Zhang, and T. D. Barfoot, "Learning spatiotemporal occupancy grid maps for lifelong navigation in dynamic scenes," in *2022 Intl. Conf. on Robot. and Autom. (ICRA)*, 2022, pp. 484–490.

[8] G. Chen, W. Dong, P. Peng, J. Alonso-Mora, and X. Zhu, "Continuous occupancy mapping in dynamic environments using particles," *arXiv preprint arXiv:2202.06273*, 2022.

[9] J. Hou, X. Zhou, Z. Gan, and F. Gao, "Enhanced decentralized autonomous aerial robot teams with group planning," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 7, no. 4, pp. 9240–9247, 2022.

[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[11] B. L'Espérance and K. Gupta, "Safety hierarchy for planning with time constraints in unknown dynamic environments," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1398–1411, 2014.

[12] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[13] J. Lin, H. Zhu, and J. Alonso-Mora, "Robust Vision-based Obstacle Avoidance for Micro Aerial Vehicles in Dynamic Environments," in *2020 Intl. Conf. on Robot. and Autom. (ICRA)*, 2020, pp. 2682–2688.

[14] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *2017 IEEE/RSJ Intl. Conf. on Intell. Robots and Syst. (IROS)*, 2017, pp. 236–243.

[15] F. Gao and S. Shen, "Quadrotor trajectory generation in dynamic environments using semi-definite relaxation on nonconvex QCQP," in *2017 Intl. Conf. on Robot. and Autom. (ICRA)*, 2017-05, pp. 6354–6361.

[16] H. Chen and P. Lu, "Real-time identification and avoidance of simultaneous static and dynamic obstacles on point cloud for UAVs navigation," *Robotics and Autonomous Systems*, vol. 154, p. 104124, 2022.

[17] G. Chen, P. Peng, P. Zhang, and W. Dong, "Risk-aware trajectory sampling for quadrotor obstacle avoidance in dynamic environments," *IEEE Transactions on Industrial Electronics*, pp. 1–10, 2023.

[18] G. Chen, S. Wu, M. Shi, W. Dong, H. Zhu, and J. Alonso-Mora, "RAST: Risk-Aware Spatio-Temporal Safety Corridors for MAV Navigation in Dynamic Uncertain Environments," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 8, no. 2, pp. 808–815, 2023.

[19] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory Planning for Quadrotor Swarms," *IEEE Trans. Robot. (T-RO)*, vol. 34, no. 4, pp. 856–869, 2018-08.

[20] H. Zhu and J. Alonso-Mora, "B-UAVC: Buffered Uncertainty-Aware Voronoi Cells for Probabilistic Multi-Robot Collision Avoidance," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2019, pp. 162–168.

[21] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 5, no. 2, pp. 604–611, 2020.

[22] K. Kondo, J. Tordesillas, R. Figueroa, J. Rached, J. Merkel, P. C. Lusk, and J. P. How, "Robust mader: Decentralized and asynchronous multiagent trajectory planner robust to communication delay," in *2023 IEEE Intl. Conf. on Robot. and Autom. (ICRA)*, 2023, pp. 1687–1693.

[23] J. Park and H. J. Kim, "Online Trajectory Planning for Multiple Quadrotors in Dynamic Environments Using Relative Safe Flight Corridor," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 6, no. 2, pp. 659–666, 2021.

[24] K. S. Mann, A. Tomy, A. Paigwar, A. Renzaglia, and C. Laugier, "Predicting Future Occupancy Grids in Dynamic Environment with Spatio-Temporal Learning," 2022.

[25] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 4, no. 4, pp. 3529–3536, 2019-10.

[26] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 2, no. 3, pp. 1688–1695, 2017.

[27] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically Constrained Trajectory Optimization for Multicopters," *IEEE Trans. Robot. (T-RO)*, pp. 1–10, 2022.