2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Safe and Distributed Multi-Agent Motion Planning under Minimum Speed Constraints

Inkyu Jang, Jungwon Park, and H. Jin Kim

***Abstract*— The motion planning problem for multiple unstoppable agents is of interest in many robotics applications, for example, autonomous traffic management for multiple fixed-wing aircraft. Unfortunately, many of the existing algorithms cannot provide safety for such agents, because they require the agents to be able to brake to a complete stop for safety and feasibility insurance. In this paper, we present a distributed multi-agent motion planner that guarantees collision avoidance and persistent feasibility, which can be applied to a team of homogeneous mobile vehicles that cannot stop. The planner is built on top of the idea that a collision-free trajectory in form of a loop can safely accommodate multiple unstoppable agents, while avoiding collisions among them and static obstacles. At every time step, in a distributed manner, the agents generate trajectory-manipulating actions that preserve the loop structure. Then, a deconfliction process selects a conflict-free subset of the generated actions, which are applied at the next time step. Through simulation using an unstoppable Dubins car model, we show that the proposed motion planner is able to provide persistent safety guarantees for such agents in obstacle-cluttered space in real-time.**

I. Introduction

In order to guarantee safety in receding-horizon multi-agent motion planning problems, not only collision avoidance but also persistent feasibility should be thoroughly considered. In cases where the robots are designed to be capable of braking to a complete stop (e.g., quadrotor drones, low-speed wheeled robots), adding a final stop constraint to the planned trajectory will provide feasibility assurance in a recursive manner [1–3]. However, for systems without such capabilities, (e.g., fixed-wing aircraft, highway traffic with minimum speed limits), this becomes no longer feasible: thus, special consideration should be taken.

The planning problem for such agents is important especially in the field of autonomous traffic management of unmanned airborne systems [4–7], since fixed-wing aircraft in general cannot hover in mid-air. Unfortunately, many of the algorithms do not provide theoretical guarantee of collision avoidance and/or feasibility, and they may fail in scenarios with heavy traffic or obstacles. Moreover, such works assume a certain class of system dynamics and cannot be applied to generic unstoppable systems.

The authors are with the Department of Aerospace Engineering, Automation and Systems Research Institute (ASRI), and Institute of Advanced Aerospace Technology (IAAT), Seoul National University, Seoul, Korea. {leplusbon, qwerty35, hjinkim}@snu.ac.kr

This research was supported by Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea(NRF) and Unmanned Vehicle Advanced Research Center(UVARC) funded by the Ministry of Science and ICT, the Republic of Korea. (NRF-2020M3C1C1A01086411)

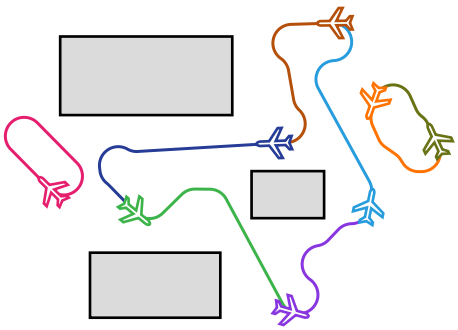


Fig. 1: A collision-free loop structure consisting of three disjoint loops. Within the loops, the agents can permanently stay without colliding with other agents. The agents and their planned trajectories are depicted using airplane icons and curves of different colors. The static obstacles are drawn as gray regions.

In this paper, we propose a safe and distributed multi-agent motion planning algorithm for a team of homogeneous unstoppable robots. The agents form a collision-free loop (see Fig. 1), within which they can permanently stay safe by following the leader. Collision avoidance and persistent feasibility of the overall planning algorithm is guaranteed through this loop, and the agents reach their goal positions by generating and committing actions that preserve the loop structure. A score-based action deconfliction process is presented, which allows the agents to select a conflict-free subset of the generated actions through communication. In a simulation using an unstoppable model, we observe that the agents can persistently avoid collision and infeasibilities while navigating through environments with obstacles.

II. Related Work

There are many existing receding-horizon multi-agent planners that consider both collision avoidance and feasibility, in which safety constraints are often linearized to enable online trajectory generation [3, 8, 9]. However, these are not suitable for unstoppable systems, as they confine themselves to a certain class of systems (e.g., linear time-invariant system), almost all of which are brakable.

The planning problem for unstoppable agents is of interest especially in the field of air traffic control (ATC), and many ATC algorithms [4–7, 10] have been proposed to solve conflicts involving more than two fixed-wing aircraft that the current traffic collision avoidance system (TCAS) cannot deal with. Algorithms that model the problem as a Markov decision process (MDP) were shown to be successful in low-density traffic scenarios [4, 5], and variants of optimal

reciprocal collision avoidance (ORCA, [11]) algorithm can be applied to robots with more general dynamics [12, 13], and even unstoppable ones [14, 15]. Unfortunately, they share a common weakness that no proof of collision avoidance or persistent feasibility is given.

The most common way of addressing the aforementioned shortcomings is to attach a loiter pattern to the planned trajectory as a final condition. The works [16–18] used circular loiter patterns for fixed-wing aircraft. In [19], funnels (forward invariant tubes calculated offline [20]) were used to form loiter patterns of arbitrary shape. In a slightly different manner, [21] constructs and maintains a directed graph of reachable nodes and explores the unknown space by only traversing through edges that are known to be viable, i.e., a path to the origin exists. These algorithms, however, may not be efficient in multi-agent settings. The loiter patterns and trajectories should be generated collision-free, and thus occupy too much space and sometimes prohibit the agents from efficiently reaching the goal in cluttered environments.

III. Preliminaries and Assumptions

A. Unstoppable Multi-Agent System

We start by assuming a team of a finite number of homogeneous mobile robots with the following nonlinear time-invariant dynamics model:

$$\dot{x}_i(t) = f(x_i(t), u_i(t)), \quad \forall i \in \mathcal{I}, \tag{1}$$

where $x_i(t) \in X$ and $u_i(t) \in U$ are the state and input variables of agent i at time t, $\mathcal{I}$ is the set of all agents. Let $W = \mathbb{R}^n$ be the workspace (two- or three-dimensional in most cases) of the robots. The differentiable projection map $\pi : X \to W$ relates the agent's state to its position in the workspace W.

We assume that the dynamics of each agent is *unstoppable*. That is, the trace of each agent's trajectory in W is constrained to a lower bound $v_{\min} > 0$, i.e., $\left\| \frac{d}{dt}\pi(x_i(t)) \right\| \geq v_{\min}$, where $\|\cdot\|$ is used to denote the 2-norm on W throughout this paper. Note that this can either be a physical constraint of the system (e.g., fixed-wing aircraft), or a rule that the agents should comply with (e.g., highway traffic with minimum-speed limit).

B. Collision Avoidance and Persistent Feasibility

We model the shape of each agent at state x as a closed ball of radius $r > 0$ centered at $p = \pi(x)$, which we denote by $B_r(p)$. Considering the ball-shaped collision model, in order to avoid colliding into a static obstacle,

$$B_r(\pi(x_i(t))) \cap O_s = \varnothing, \quad \forall i \in \mathcal{I} \tag{2}$$

should hold, where $O_s \subseteq W$ is the set of static obstacles. And to avoid collision between robots, $\left\|\pi(x_i(t)) - \pi(x_j(t))\right\| > 2r$ should hold for any pair of distinct agents i, $j \in \mathcal{I}$. In order to guarantee persistent feasibility at time t, there should be a well-defined infinite-horizon control strategy $u_{\infty,i} : [t, \infty) \to U$ for every agent $i \in \mathcal{I}$, such that

$$\left\|\pi(x_{\infty,i}(\tau)) - \pi(x_{\infty,j}(\tau))\right\| > 2r, \quad \forall j \in \mathcal{I} \setminus \{i\}, \tag{3}$$

where $x_{\infty,i} : [t, \infty) \to X$, the predicted trajectory of i, satisfies the system dynamics $\dot{x}_{\infty,i} = f(x_{\infty,i}, u_{\infty,i})$ and the initial condition $x_{\infty,i}(t) = x_i(t)$.

C. Trajectory Generation and Representation

In this work, we assume that we are equipped with a single-agent motion planner `MP`:

$$(x_p, u_p, T) = \texttt{MP}(t; x_s, x_g, O), \tag{4}$$

where t is the starting time of the planned trajectory, $x_s, x_g \in X$ are the start and goal states, $O \subseteq W$ is the set of forbidden positions, $x_p : [t, t+T] \to X$ and $u_p : [t, t+T] \to U$ are the state and input trajectories satisfying $\dot{x}_p(\tau) = f(x_p(\tau), u_p(\tau))$ and $B_r(\pi(x_p(\tau))) \cap O = \varnothing$ for all $\tau \in [t, t+T]$, $x_p(t) = x_s$, and $x_p(t+T) = x_g$. T is the duration of the planned trajectory.

The planner `MP` does not need to be complete or optimal. However, for the sake of real-time application, it should return (either success or failure) before time $t - \delta_p$, where $\delta_p > 0$ is a constant representing the maximum time needed for communication and the action deconfliction process, which will be introduced in section VI. This can be easily implemented by marking any task not terminating before $t - \delta_p$ as a failure. Additionally, we require `MP` to generate *self-collision-free* trajectories. That is, the agent does not revisit the place where it was located previously. More formally,

$$\left\|\pi(x_p(\tau_1)) - \pi(x_p(\tau_2))\right\| > 2r \tag{5}$$

for any pair of $\tau_1, \tau_2 \in [t, t+T]$ such that $|\tau_1 - \tau_2| > 2r/v_{\min}$. Most off-the-shelf motion planners are not designed to meet this requirement. Nevertheless, in most cases, a slight modification to the algorithm (i.e., considering past trajectories as obstacles) will produce self-collision-free trajectories.

IV. Loop Formation

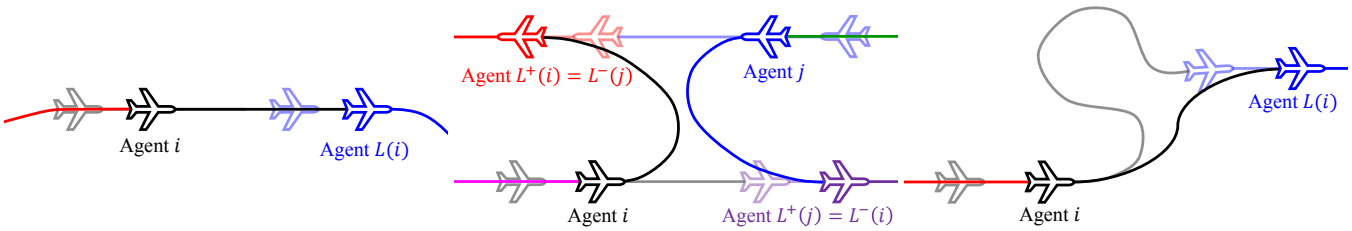
Suppose for every agent $i \in \mathcal{I}$, there exists a planned trajectory $(x_{p,i}, u_{p,i}, T_i)$ starting at time t, where $x_{p,i}$ is a (finite-horizon) trajectory which satisfies the initial condition $x_{p,i}(t) = x(t)$ and is collision-free with respect to the static obstacles. In this section, we develop a set of additional constraints for $x_{p,i}$ and $u_{p,i}$, such that collision avoidance between agents and persistent feasibility are guaranteed.

A. Leader-Follower Relationship and Persistent Feasibility

For persistent feasibility, we require every agent to follow an agent. That is, for every agent $i \in \mathcal{I}$, there exists a leading agent $L(i) \in \mathcal{I}$ such that $x_{p,i}(t+T_i) = x_{p,L(i)}(t)$. Then, we can find that

$$u_{\infty,i}(\tau) = \begin{cases} u_{p,i}(\tau) & (\tau \in [t, t+T_i)), \\ u_{p,L(i)}(\tau - T_i) & (\tau \in [t+T_i, t+T_i+T_{L(i)})), \\ \vdots & \end{cases} \tag{6}$$

is a persistently feasible control strategy for agent i to stay away from colliding with the static obstacles. Note that, it is also possible for an agent to lead itself, i.e., $L(i) = i$. We call this mapping $L : \mathcal{I} \to \mathcal{I}$ the leader-follower relationship.



(a) Hold trajectory for agent i (b) Transposition between agents i and j (c) Loop deformation for agent i

Fig. 2: The three types of trajectory manipulations that preserves the loop structure. The agents and their trajectories are drawn as airplane icons and solid lines, respectively. Each agent and its trajectory can be distinguished from others by the color. Transparent and opaque curves represent the loop structures at time t^- and t^+, respectively.

B. Conditions for Collision Avoidance between Agents

For collision avoidance between agents, a number of more conditions should be addressed. First, the leader-follower relationship $L : \mathcal{I} \to \mathcal{I}$ needs to be a bijective map, i.e., there is only one follower for an agent. Otherwise, more than two agents may be merging into a single state, which causes potential collision.

Since $\mathcal{I}$ is a finite set, L now consists of a finite number of disjoint cycles, and the trajectories form loops. If the loops are disjoint in the workspace and self-collision-free, then the multi-agent system as a whole is persistently feasible of avoiding inter-collision between agents. For that, the planned trajectories $x_{p,i}$ should be self-collision-free, and

$$o(x_{p,i}, [t, t+T_i]) \cap o(x_{p,j}, [t, t+T_j]) = \varnothing$$

should hold for any pair of distinct agents i, $j \in \mathcal{I}$. Here, $o(\cdot,\cdot)$ is the occupancy-computing function defined as

$$o(x, [t_1, t_2]) = \left(\bigcup_{\tau \in [t_1, t_2]} B_r(\pi(x(\tau))) \right) \setminus B_r(\pi(x(t_1))). \tag{7}$$

Additionally, the *time to leader* should be sufficiently long, i.e., $T_i > 2r/v_{\min}$, $\forall i \in \mathcal{I}$, to ensure that the leader is at least $2r$ (in distance) ahead, even in case where the agents are traveling at minimum speed $v_{\min}$.

If all the aforementioned conditions are met, we call the multi-agent system as a whole, written as $\chi = \big(\chi(i) = (x_{p,i}, u_{p,i}, [t, t+T_i], L(i))\big)_{i \in \mathcal{I}} \in \mathcal{X}$, a *collision-free loop structure*, where $\mathcal{X}$ is the set containing all possible collision-free loop structures. Now, as the loops are pairwise disjoint in W and self-collision-free, we conclude that the structure can be maintained through the control law (6), and thus, it is a persistently feasible and collision-free control strategy.

V. Loop-Preserving Trajectory Manipulation

We drive the agents by updating their trajectories and the leader-follower relationship L, in a way that the collision-free loop structure is preserved. In this section, three types of possible such *loop-preserving* trajectory manipulations for agent $i \in \mathcal{I}$ are listed. We also show that such update can always be made, i.e., given the existence of a valid collision-free loop structure created at time t^-, we can always find a set of new trajectories starting at time $t^+ > t^-$, which again forms a collision-free loop structure. The superscripts $(\cdot)^-$ and $(\cdot)^+$ are used to denote the quantities before and after the update, respectively. For example, the planned trajectory of agent i is updated from $x^-_{p,i} : [t^-, t^- + T_i^-] \to X$ to $x^+_{p,i} : [t^+, t^+ + T_i^+] \to X$. A graphical description of the manipulations can be found in Fig. 2.

A. List of Loop-Preserving Manipulations

1) Hold Trajectory: A trivial and always feasible way of preserving the loop structure is to continue following the leading agent without changing the pre-planned trajectory, using (6). Under this manipulation, no trajectory is newly generated, and x_i^- is reused instead. That is, $T_i^+ = T_i^-$, $L^+(i) = L^-(i)$, $x^+_{p,i}(\tau) = x^-_{\infty,i}(\tau)$, and $u^+_{p,i}(\tau) = u^-_{\infty,i}(\tau)$ for $\tau \in [t^+, t^+ + T_i^+]$.

2) Transposition: Consider a pair of distinct agents, i and j. After transposition, i and j switch leaders, i.e., $L^+ = L^- \circ \sigma_{ij}$, where the bijective map $\sigma_{ij} : \mathcal{I} \to \mathcal{I}$ swaps i and j. The two agents plan disjoint trajectories to the respective new leaders, which should be collision-free with respect to the static obstacles and other agents' trajectories. This can be done sequentially, i.e., first plan the trajectory for i using

$$(x^+_{p,i}, u^+_{p,i}, T_i^+) = \texttt{MP}(t^+; x^-_{p,i}(t^+), x^-_{p,L^-(j)}(t^+), O_{\mathrm{T},i}),$$

and then for j using

$$(x^+_{p,j}, u^+_{p,j}, T_j^+) = \texttt{MP}(t^+; x^-_{p,j}(t^+), x^-_{p,L^-(i)}(t^+), O_{\mathrm{T},j}),$$

where

$$O_{\mathrm{T},i} = \left(\bigcup_{k \in \mathcal{I} \setminus \{i,j\}} o(x^-_{\infty,k}, [t^+, t^+ + T_k^-]) \right) \cup O_s,$$
$$O_{\mathrm{T},j} = O_{\mathrm{T},i} \cup o(x^+_{p,i}, [t^+, t^+ + T_i^+]).$$

3) Loop Deformation: Through loop deformation, an agent i's trajectory $x_{p,i}$ is replaced by a newly-planned one, but the leader-follower relationship remains unchanged, i.e., $L^+ = L^-$. To avoid any conflict with the other agents' existing trajectory, the new trajectory $x^+_{p,i}$ should not coincide with

$$O_{\mathrm{LD},i} = \left(\bigcup_{j \in \mathcal{I} \setminus \{i\}} o(x^-_{\infty,j}, [t^+, t^+ + T_j^-]) \right) \cup O_s,$$

and we can obtain it by performing

$$(x^+_{p,i}, u^+_{p,i}, T^+_i) = \texttt{MP}(t^+; x^-_{p,i}(t^+), x^-_{p,L^-(i)}(t^+), O_{\mathrm{LD},i}).$$

This can also be understood as a *transposition with itself*. However, we differentiated this from transposition, since it changes only one agent's trajectory, so the implementation should be different.

B. Recursive Feasibility and Parallelizability

The trajectory updates listed above may fail either due to infeasibility or computational intractability. Nevertheless, at least one of them, i.e., hold trajectory, is always feasible, thanks to the valid infinite-horizon control strategy (6). Moreover, this does not require any heavy computation (e.g., calling MP), and can be calculated in real-time.

We also point out that the process of finding the trajectory candidates is parallelizable, provided the assumption that all agents have access to $x^-_{p,\cdot}$, $u^-_{p,\cdot}$, and L^-, and they have consensus on the update time t^+, prior to planning, since we have used only $(\cdot)^-$ and t^+ to generate new trajectories. This allows the multi-agent motion planning problem to be solved in a distributed way.

VI. Algorithm

Using the loop-preserving trajectory updates, we now present the overall multi-agent planning algorithm. The algorithm is distributed and can run in real-time with guarantees of collision-avoidance and persistent feasibility.

A. Initialization

First, we assume that the agents start in a collision-free loop structure $\chi^0 = \left((x^0_{p,i}, u^0_{p,i}, [t^0, t^0 + T^0_i], L^0(i))\right)_{i\in\mathcal{I}} \in \mathcal{X}$, where the superscript $(\cdot)^k$ denotes the quantities after the k-th update. Since finding the next update is recursively feasible, this assumption provides persistent feasibility. A simple way of finding the initial loop χ^0 from the states $\left(x_i(t^0)\right)_{i\in\mathcal{I}}$ is to construct self-loops, i.e., $L^0(i) = i$, where each agent's trajectory $x^0_{p,i}$ is a pre-defined loiter pattern, similar to [17]. If the agents are spaced sufficiently far from each other and the static obstacles at the initial step, such patterns can easily be placed without collision.

B. Trajectory Planning

Since the MPs in section V are used to generate trajectories to the leading agent (and not the goal), one cannot expect that the planning strategy will eventually drive the agent to the goal state. Therefore, we replace each MP with a slightly modified version, $\texttt{MP}_g$, where $g \in X$ is the goal state of the agent. If succeeded, $(x_p, u_p, T) = \texttt{MP}_g(t; x_s, x_g, O)$ generates a trajectory x_p that starts at $x_p(t) = x_s \in X$, first tries to head towards g in some interval $[t, t + \Delta T)$, and then reaches $x_g \in X$ at time $t + T$, as shown in Fig. 3, where $\Delta T > 0$ is a parameter. This can be implemented by calling MP twice as follows. First, we generate a trajectory that avoids $O_1 = O \cap B_R(\pi(x_s))$ and reaches g, where $R > 0$ is a parameter that defines the neighborhood range.

$$(x_1, u_1, T_1) = \texttt{MP}(t; x_s, g, O_1) \tag{8}$$

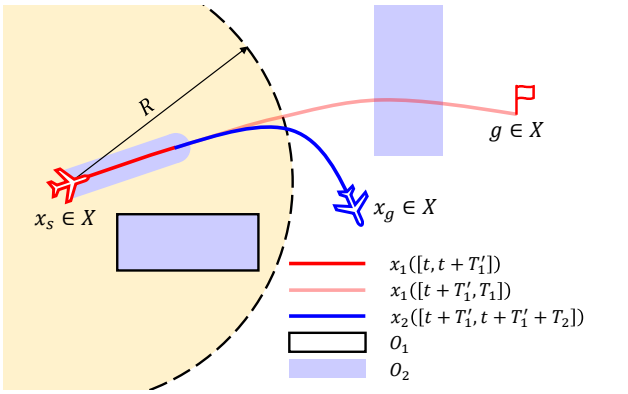


Fig. 3: The modified planner $\texttt{MP}_g$. It receives two goals g and x_g, and plans a trajectory whose beginning portion heads to g, and the rest terminates in x_g. The trajectory x_1 (red curve) is allowed to go through non-neighboring obstacles. Then, the goal-reaching trajectory x_2 (blue curve) is appended, which connects $x_1(t + T_1')$ and x_g.

Then, the x_g-reaching trajectory

$$(x_2, u_2, T_2) = \texttt{MP}(t + T_1'; x_1(t + T_1'), x_g, O_2), \tag{9}$$

where $O_2 = O \cup o(x_1, [t, t + T_1])$, is appended as follows:

$$x_p(\tau) = \begin{cases} x_1(\tau) & (\tau \in [t, t + T_1')) \\ x_2(\tau) & (\tau \in [t + T_1', t + T_1' + T_2]), \end{cases} \tag{10}$$

where $T = T_1' + T_2$, and $T_1' = \min\{T_1, \Delta T\}$. The input trajectory u_p is determined similarly. If either one of the two MPs fails, it generates a trajectory that is not collision-free, or $T \le 2r/v_{\min}$, then $\texttt{MP}_g$ is considered as a failure. It is straightforward to find out that $\texttt{MP}_g$ qualifies as a replacement of MP, i.e., it satisfies all the assumptions listed in section III-C.

C. Action Generation and Deconfliction

Now, we generate *actions* in a distributed manner. An action a is defined as $a = (\mathcal{I}_a, \chi^-_a, \chi^+_a, p_a)$, where $\mathcal{I}_a \subseteq \mathcal{I}$ is the set of *affected agents* (i.e., their trajectories change after this action is taken), $\chi^-_a \in \mathcal{X}$ and $\chi^+_a \in \mathcal{X}$ are the loop structures before and after the action, respectively. $p_a \in \mathbb{R}$ is the *score* of the action, which is used in the deconfliction step. An action contains the information of the loop structure before and after a single update, which should be one of the three types introduced in section V.

The planning algorithm does not require any specific way of selecting (or computing) the action scores p_a. In this paper, we use

$$p_a = \max_{j\in\mathcal{I}_a} \left(p(\chi^+_a(j)) - p(\chi^-_a(j))\right) \tag{11}$$

as a good rule of thumb, where the map p is defined as

$$p\left((x_{p,i}, u_{p,i}, [t, t + T_i], L(i))\right) = \begin{cases} -d(x_{p,i}(t + \min\{\rho \cdot \Delta T, T_i\}), g_i) & (T_i \le \bar{T}) \\ -\infty & (T_i > \bar{T}), \end{cases} \tag{12}$$

ΔT is from section VI-B, $\bar{T} > \Delta T$ is the upper bound for the duration of scored trajectories (used to prohibit them from

Algorithm 1 Action deconfliction

Input: The set of planned actions $\mathcal{A}^{k+1}$
Output: The set of chosen actions $\mathcal{A}_c^{k+1}$

$O \leftarrow \varnothing$, $\mathcal{A}_c^{k+1} \leftarrow \varnothing$, $\mathcal{I}_c \leftarrow \varnothing$, $\mathcal{A} \leftarrow \mathcal{A}^{k+1}$
while $\mathcal{A}$ is not empty **do**
  $a \leftarrow \arg\max_{a \in \mathcal{A}} p_a$
  $(x_{p,j}^+, u_{p,j}^+, \mathcal{T}_j^+, l_j^+) \leftarrow \chi_a^+(j), \forall j \in \mathcal{I}_a$
  Remove a from $\mathcal{A}$.
  $O_a \leftarrow \bigcup_{j \in \mathcal{I}_a} o(x_{p,j}^+, \mathcal{T}_j^+)$
  if $\mathcal{I}_a \cap \mathcal{I}_c = \varnothing$ and $O \cap O_a = \varnothing$ **then**
    $\mathcal{I}_c \leftarrow \mathcal{I}_c \cup \mathcal{I}_a$
    $O \leftarrow O \cup O_a$
    $\mathcal{A}_c^{k+1} \leftarrow \mathcal{A}_c^{k+1} \cup \{a\}$
  end if
end while

occupying excessive space), $\rho > 0$ is a parameter whose typical value lies near 1, $d : X \times X \to \mathbb{R}_{\geq 0}$ is a pseudo-metric on X, and $g_i \in X$ is the goal state of agent i. It measures how much the action *improves* the trajectory in terms of distance to the goal state, while penalizing overly long trajectories.

During the time interval $[t^k, t^{k+1} - \delta_p)$, each agent i generates actions that affects itself, i.e., $i \in \mathcal{I}_a$. After generation, at time $t^{k+1} - \delta_p$, the generated actions are posted, and all the agents now have access to the set of generated actions, which we denote using $\mathcal{A}^{k+1}$. Since the actions are generated distributively by the agents, although they are compatible with the existing loop structure, their occupancies may overlap, i.e., conflict may occur. Thus, during the interval $[t^{k+1} - \delta_p, t^{k+1})$, we construct the set of conflict-free actions $\mathcal{A}_c^{k+1} \subseteq \mathcal{A}^{k+1}$, which should satisfy the following:

1) Each agent should be assigned only one next trajectory, i.e., $\{\mathcal{I}_a \mid a \in \mathcal{A}_c^{k+1}\}$ is a partition of $\mathcal{I}$.
2) The occupancies of the new trajectories should not overlap, i.e., $o(x_{p,i}^+, \mathcal{T}_i^+)$ are disjoint, for all $a \in \mathcal{A}_c^{k+1}$, $i \in \mathcal{I}_a$, $\chi_a^+(i) = (x_{p,i}^+, u_{p,i}^+, \mathcal{T}_i^+, L^+(i))$.

Note that the generation of hold trajectory action is always feasible for every agent, and therefore $\mathcal{A}^{k+1}$ contains every possible such actions. The second condition is automatically satisfied for hold trajectory actions if the first one is met, because the new trajectories are planned to avoid the existing trajectories. Therefore, a feasible choice of $\mathcal{A}_c^{k+1}$ always exists: it is the collection only of hold trajectory actions.

The action deconfliction algorithm shown in Algorithm 1 finds such $\mathcal{A}_c^{k+1}$ while prioritizing actions with higher scores. Since the algorithm does not require heavy computation (it does not generate trajectories), the same process can be done individually by the agents within the time interval $[t^{k+1} - \delta_p, t^{k+1})$, and at time t^{k+1}, the agents can proceed to their new trajectories. For the deconfliction algorithm to produce the same results for all agents, we need an additional assumption that there is no duplicate action score p_a among all the planned actions $a \in \mathcal{A}^{k+1}$. This is achieved with probability 1 given that X is a continuous space, and the initial and goal states of the agents are not adversarially given. Even in case duplication occurs, an ad-hoc decision-making step will resolve the conflict.

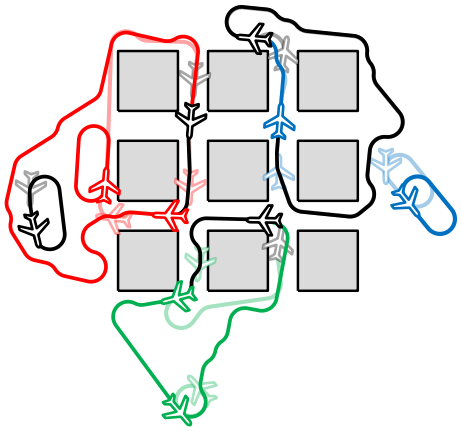


Fig. 4: A snapshot of a trajectory update in an obstacle environment. Transparent and opaque lines denote the loop structures before and after the update, respectively. Black and blue curves denote hold trajectory and loop deformation actions, respectively. Green and red colors denote transposition actions: agents with the same color swap leaders. The static obstacles are drawn as gray boxes.

Thanks to the existence of the always-feasible hold trajectory actions, it can be shown that the resulting $\mathcal{A}_c^{k+1}$ satisfies the two conditions listed above, although it is not the optimal choice. Fig. 4 shows an example of a possible trajectory update in an environment with obstacles. It can be easily found that the set of conclict-free actions $\mathcal{A}_c^{k+1}$ updates the trajectory in a way that the loop structure is preserved, i.e., every agent is inside a collision-free loop.

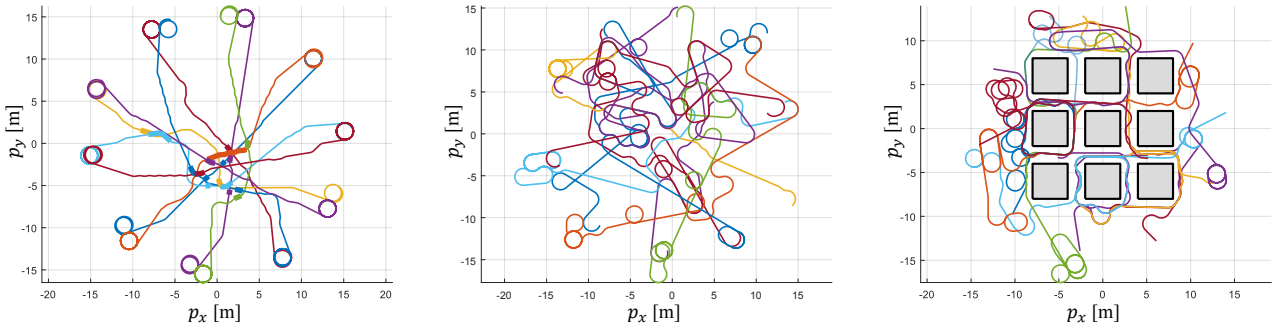
VII. Simulation Results

We simulate the proposed planning method using a simple model of fixed-wing unmanned aircraft flying on a fixed flight level with fixed speed $v \in \mathbb{R}_{>0}$, maximum trajectory curvature $\kappa_m \in \mathbb{R}_{>0}$, which can be written as

$$\dot{p}_x = v \cos\psi, \qquad \dot{p}_y = v \sin\psi, \qquad \dot{\psi} = v\kappa, \tag{13}$$

where $(p_x, p_y) \in \mathbb{R}^2$ represents the position of aircraft in the two-dimensional space, $\psi \in S^1$ is the heading angle, i.e., $x_i = (p_{x,i}, p_{y,i}, \psi_i) \in X = \mathrm{SE}(2)$ and $\pi(x_i) = (p_{x,i}, p_{y,i}) \in W = \mathbb{R}^2$. We use the signed curvature of the trajectory as the control input, i.e., $u = \kappa \in U = [-\kappa_m, \kappa_m]$. Since v is fixed to a positive value, it can be easily seen that this system is unstoppable. We point out that this two-dimensional setting (with single-dimensional U) can be more difficult than the three-dimensional one which additionally allows flight level changes, and the proposed method does not require the workspace to have a specific dimension.

The parameters used in the simulation are summarized in Table I. For the base motion planner `MP`, a simple sampling-based motion planner based on the Dubins car model [22] is implemented and used. The planner samples inputs from the set $\{-\kappa_m, 0, \kappa_m\} \subseteq U$ and creates a tree



(a) CSORCA [14] (b) Proposed, obstacle-free environment (c) Proposed, obstacle environment

Fig. 5: The trajectories accumulated for 30 seconds. The thick portions of the trajectories denote where inter-agent collisions occurred. The proposed method does not permit any collision, as shown in (b) and (c).

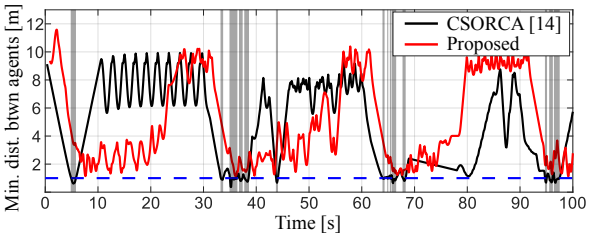


Fig. 6: The minimum among inter-agent distances in the obstacle-free scenario, plotted as a function of time. The proposed method always keeps the distance above the safety bound $2r = 1$ m (shown using a blue dashed line), while CSORCA [14] sometimes fails to prevent inter-agent collisions. CSORCA's collisions are highlighted as gray shaded regions.

TABLE I: Parameters used in the simulation

Symbol	Description	Value
r	Agent size	0.5 m
R	Neighbor range	4.5 m
v	Speed	3 m/s
κ_m	Maximum trajectory curvature	1 m^{-1}
$v_{\min}$	Minimum speed	2 m/s
ΔT	Length of the goal-pursuing trajectory	1 s
$t^{k+1} - t^k$	Update interval	1 s
$\bar{T}$	Max. scored trajectory length	6 s
δ_p	Time required for action deconfliction	0.1 s
ρ	Multiplier for score calculation	1.5

of reachable states. It terminates if a collision-free Dubins curve from a node to the goal position exists. For the sake of computational efficiency in collision checking, the obstacles and the trajectory occupancies are conservatively modeled as unions of convex polygons, and the Gilbert–Johnson–Keerthi (GJK) distance algorithm [23] is used to detect collisions between polygons.

The proposed algorithm is tested in the following two different simulation environments.

1) **Obstacle-Free Environment**: Agents start at evenly spaced initial positions along the circumference of a 15-meter-radius circle. Every 30 seconds, the agents try to switch positions with their antipodal counterparts.
2) **Obstacle Environment**: With the same initial condition, the agents are now required to avoid nine square static obstacles while traveling to the antipodal positions. This resembles an urban area with tall buildings.

In each scenario, we use 10 agents, with 1-meter-radius circles as χ^0. The planner is implemented using C++ and ROS2, and run on a desktop computer equipped with 32 GB RAM and an octacore CPU whose clock runs at 2.90 GHz. The simulation runs in real-time: the action generation and deconfliction are done within $t^{k+1} - t^k - \delta_p = 0.9$ s and $\delta_p = 0.1$ s, respectively. The generated trajectories shown in Fig. 5 confirm that the proposed planner allows the agents to avoid collisions between static obstacles and others, while always maintaining a collision-free loop structure.

In the obstacle-free scenario, we compare with constant speed ORCA (CSORCA) [14], which can be applied to the exact same Dubins car model in obstacle-free environments. As shown in Fig. 5 and Fig. 6, the minimum safety distance $2r = 1$ m is kept throughout the simulation using the proposed planner, while CSORCA sometimes fails to do so, especially when the agents gather near the origin on the way to their goals.

VIII. Concluding Remarks

In this paper, we present a safe and recursively feasible motion planner that can be applied to homogeneous multi-agent systems consisting of robots that cannot brake. For collision-avoidance, the agents are required to follow another agent through a collision-free trajectory, and only one follower is allowed for an agent. The trajectories form disjoint loops, and the agents proceed to the goals by manipulating their trajectories in a way that the loop structure is preserved. Three different types of such manipulations are introduced, at least one of which is always feasible. Thus, given an initial valid loop structure, the motion planning problem is persistently feasible. Through simulation experiments, we show that the proposed algorithm generates safe trajectories in real-time, even in obstacle-cluttered environments with dense traffic.

The proposed motion planner is not optimal as it is. Its optimality heavily depends on the base planner `MP`. Moreover, the score-based action deconfliction scheme can also be improved, since it inherits the shortcomings of greedy search algorithms. Therefore, for future work, we will look for more loop-structure-preserving actions and new action selection algorithms, which could improve the efficiency of the overall algorithm.

REFERENCES

[1] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 1934–1940.

[2] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, 2021.

[3] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, "Online distributed trajectory planning for quadrotor swarm with feasibility guarantee using linear safe corridor," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4869–4876, 2022.

[4] X. Yang and P. Wei, "Scalable multi-agent computational guidance with separation assurance for autonomous urban air mobility," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 8, pp. 1473–1486, 2020.

[5] ——, "Autonomous free flight operations in urban air mobility with computational guidance and collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5962–5975, 2021.

[6] A. Degas, E. Kaddoum, M.-P. Gleizes, F. Adreit, and A. Rantrua, "Cooperative multi-agent model for collision avoidance applied to air traffic management," *Engineering Applications of Artificial Intelligence*, vol. 102, p. 104286, 2021.

[7] P. Zhao, H. Erzberger, and Y. Liu, "Multiple-aircraft-conflict resolution under uncertainties," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 11, pp. 2031–2049, 2021.

[8] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient multi-agent trajectory planning with feasibility guarantee using relative Bernstein polynomial," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 434–440.

[9] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, "Online distributed trajectory planning for quadrotor swarm with feasibility guarantee using linear safe corridor," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4869–4876, 2022.

[10] J. Yang, X. Xu, D. Yin, Z. Ma, and L. Shen, "A space mapping based 0–1 linear model for onboard conflict resolution of heterogeneous unmanned aerial vehicles," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7455–7465, 2019.

[11] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[12] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems*. Springer, 2013, pp. 203–216.

[13] M. Rufli, J. Alonso-Mora, and R. Siegwart, "Reciprocal collision avoidance with motion continuity constraints," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 899–912, 2013.

[14] N. Durand, "Constant speed optimal reciprocal collision avoidance," *Transportation research part C: emerging technologies*, vol. 96, pp. 366–379, 2018.

[15] H. Niu, C. Ma, and P. Han, "Directional optimal reciprocal collision avoidance," *Robotics and Autonomous Systems*, vol. 136, p. 103705, 2021.

[16] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *Proceedings of the 2004 American Control Conference*, vol. 6, 2004, pp. 5576–5581 vol.6.

[17] ——, "Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 5141.

[18] D. Althoff, M. Althoff, and S. Scherer, "Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3470–3477.

[19] I. Jang, D. Lee, S. Lee, and H. J. Kim, "Robust and recursively feasible real-time trajectory planning in unknown environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1434–1441.

[20] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[21] D. Fridovich-Keil, J. F. Fisac, and C. J. Tomlin, "Safely probabilistically complete real-time planning and exploration in unknown environments," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7470–7476.

[22] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[23] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.