2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Safety-First Tracker: A Trajectory Planning Framework for Omnidirectional Robot Tracking

Yue Lin, Yang Liu, Pingping Zhang, Xin Chen, Dong Wang and Huchuan Lu

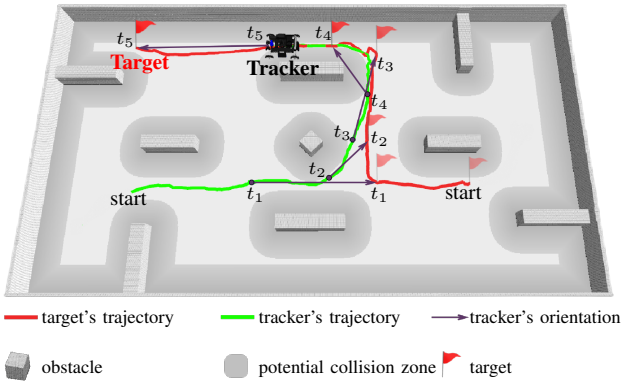
***Abstract*— This paper introduces a Safety-First Tracker (SF-Tracker) designed for omnidirectional autonomous tracking robots. The position and orientation of omnidirectional robots are decoupled for stepwise planning to ensure trajectory safety and maintain target visibility. SF-Tracker puts the trajectory safety in the first place. First, a collision-free and occlusion-free reference path is efficiently initialized by constructing a directed weighted graph. By building upon this path, safe trajectory optimization is implemented to ensure safe movement. Finally, an orientation planner is developed to achieve target visibility based on the safe trajectory. Extensive experimental evaluations in simulated environments and the real world demonstrate that the SF-Tracker outperforms state-of-the-art methods in terms trajectory safety and target visibility. Ablation experiments further demonstrate the significance of each step of the SF-Tracker. The source code and demonstration video can be found at `https://github.com/Yue-0/SF-Tracker`.**

I. INTRODUCTION

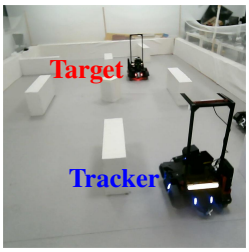
With the rise of autonomous robots, such as service robots [1] and medical robots [2], planners for autonomous tracking have attracted widespread attention. Automatic tracking is challenging, particularly in complex dynamic environments. This challenge arises from the need to balance four essential requirements within a unified task.

- Trajectory safety: The robot must avoid collisions with all obstacles and the target while tracking.
- Target visibility: The robot must maintain continuous visibility of the target within its restricted view throughout the movement process.
- Motion smoothness: The robot's movement should be smooth to prevent motion blurring of the target in view.
- Real-time processing: The planning algorithm must generate a solution in real time. This solution must adhere to dynamic constraints to handle the swift movement of the target and variations in the environment.

Some state-of-the-art autonomous trackers [3]–[5] that address the aforementioned issues have demonstrated exceptional performance. However, existing methods are prone to failure in complex scenarios due to the inherent conflict between trajectory safety and target visibility. For example, tracking frequently fails when the target maneuvers around obstacles, traverses narrow paths, or suddenly changes its direction of movement. The primary reason for such conflict is that ensuring trajectory safety necessitates trackers to select a collision-free path, while target visibility requirement often leads to aggressive paths that may overlook safety. To alleviate this dilemma, we propose a Safety-First Tracker (**SF-Tracker**) to guarantee that each planned trajectory can avoid collisions in complex dynamic scenarios while maintaining continuous visibility of the target.



(a) Trajectory history of the tracker and the target.



(b) Third perspective.

(c) Camera view at time t_4.

Fig. 1. The real-world omnidirectional robot uses our SF-Tracker to continuously track a human-controlled target. The tracker avoids all potential collision zones while maintaining visibility of the target. Experimental details are described in Section VI.

The proposed SF-Tracker puts the robot's motion safety in the first place. In particular, we decouple the position and orientation of the omnidirectional robot and formulate a step-by-step plan to prioritize trajectory safety. Then, we ensure target visibility and motion smoothness. First, we propose a method for initializing a collision-free and occlusion-free reference path by constructing a directed weighted graph. Next, we construct and solve the trajectory optimization problem to obtain a trajectory that simultaneously satisfies safety, kinematic feasibility, and motion smoothness. Finally, we design an orientation planner for the robot based on safe trajectories to ensure target visibility and smooth rotation. Fig. 2 illustrates the overall framework of the system. The framework can run in real time on edge devices.

All authors are with Dalian University of Technology, Dalian 116024, China. Corresponding author: Yang Liu. `ly@dlut.edu.cn`.

This work was supported by the National Natural Science Foundation of China under grant Nos. 62293540, 62293542, U23A20384, 62388101, and 62006037. The authors would also like to thank the DJI RoboMaster University Series and DUT0BUG Team.

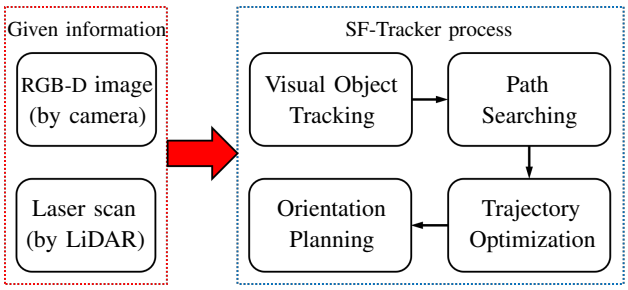


Fig. 2. System framework diagram. We implement a complete airborne system for omnidirectional robots to autonomously track targets based on laser scan data and RGB-D image information. In this paper, we propose path searching, trajectory optimization and orientation planning modules.

We have benchmarked SF-Tracker using state-of-the-art methods in simulation environment. When the target moves in a dense environment, the SF-Tracker can ensure the safety of the tracker and maintain satisfactory target visibility, whereas other methods are prone to collision or tracking failure. In addition, real-world experiments demonstrate that the SF-Tracker can control the ground omnidirectional robot to track targets, as shown in Fig. 1.

In summary, the major contributions of this paper are:

- We present the SF-Tracker to implement autonomous tracking for omnidirectional robots, effectively preventing collisions during tracking while maintaining target visibility and motion smoothness.
- We propose an efficient path search algorithm to avoid collisions and occlusions, a new safety penalty function to ensure trajectory safety, and an angle planning method to achieve target visibility.
- We conduct extensive simulations and real-world experiments to validate that our method enables safe, real-time tracking planning for omnidirectional robots.

II. RELATED WORK

Autonomous tracking has been extensively studied in robotics. Recently, learning-based trackers have achieved success in complex environments. However, these methods are either trained and fine-tuned for specific environments, lack adaptability to target and environmental changes [6], or fail to meet the demands of real-time processing and lightweight deployment [7]. To achieve target visibility, vision-based trackers [8] use the tracking error defined in image space as feedback. Although these methods can achieve real-time processing, they frequently disregard safety constraints, limiting their application to tracking scenarios only in wide open areas. In [3], a camera array is used to address target visibility constraints, while in [9], a camera with rotational degrees of freedom is utilized. However, these solutions require costly additional hardware. Vis-Planner [4] enforces target visibility by maximizing the probability of successful visual object detection, but it overlooks safety considerations. By contrast, our proposed method prioritizes trajectory safety to avoid collisions and then ensures target visibility based on safe trajectories.

The previous work [10] aimed to avoid collisions by creating safe trajectories in a principled manner. However, this method relies on the numerical optimization of entire objectives that involve complex terms, which may not guarantee satisfactory optimality. [11] and [12] developed a receding horizon motion planner to generate dynamically feasible paths in real time for dynamic situations. However, they assumed an ellipsoidal shape for obstacles, and thus, their planner was not applicable to general cases. The Elastic Tracker [5] establishes a connection between the target and the tracker through an elastic mechanism to ensure safe and stable tracking. However, this method relies on safe flight corridors, which are limited to quadcopters. [13] and [14] attempted to generate an initial path that satisfies safety and visibility by using a graph-search-based planner, then optimize the path to ensure smoothness. However, both methods rely on the assumption that the movement intention of the target is known, and thus, they are unsuitable for general situations. By contrast, our method does not require prior information and exhibits greater generalization ability.

III. SAFETY TRAJECTORY PLANNING

The SF-Tracker decouples the position and orientation of omnidirectional robots. This section presents the method for generating safe trajectories. Here, the terms *path* and *trajectory* are differentiated for clarity as described in [15]. Path refers to a geometric path, while trajectory refers to a time-parameterized path. Our method initially searches for a collision-free and occlusion-free reference path, and then optimizes this path to ensure trajectory safety, motion smoothness, and kinematic feasibility.

A. Path Searching

Given the tracker's position $\mathbf{p}_0$ and the target's best observation position $\mathbf{p}_T$, we use the A* algorithm [16] to initialize the path from $\mathbf{p}_0$ to $\mathbf{p}_T$, denoted as P_A. However, even though the initial path P_A is collision-free, it is often in close proximity to obstacles and cannot ensure absolute safety. Furthermore, it does not consider the occlusion of the target caused by obstacles, which may result in inadequate target visibility. To address these problems, we construct a directed weighted graph based on P_A to optimize it. The path P_A is a sequence of $T+1$ consecutive points, and the directional vector for each point is defined as

$$\mathbf{d}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}. \tag{1}$$

Here, $i \in \{0, 1, \cdots, T-1\}$. For two adjacent points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ in path P_A, if their direction vectors are the same (i.e., $\mathbf{d}_i = \mathbf{d}_{i+1}$), then $\mathbf{p}_{i+1}$ can be considered redundant, because it can be represented linearly by $\mathbf{p}_i$ and $\mathbf{p}_{i+2}$. Hence, the redundant points can be removed, and the directed weighted graph $\mathcal{G}$ can be constructed by the remaining points as nodes. We check for obstacles in the connection between nodes. If no obstacles exist between two nodes $\mathbf{p}_i$ and $\mathbf{p}_j$ $(i < j)$, then an edge is created from $\mathbf{p}_i$ to $\mathbf{p}_j$, with the edge weight equal to the Euclidean distance from $\mathbf{p}_i$ to $\mathbf{p}_j$. Alg. 1 illustrates this construction process.

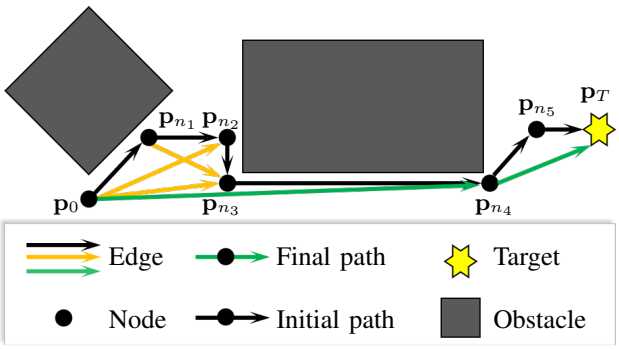


Fig. 3. An example of path searching. The initial path is $P_A = \{\mathbf{p}_0, \mathbf{p}_{n_1}, \mathbf{p}_{n_2}, \cdots, \mathbf{p}_T\}$, and the final path is optimized to the shorter path $P_D = \{\mathbf{p}_0, \mathbf{p}_{n_4}, \mathbf{p}_T\}$. The initial path P_A always follows obstacles, and the section from $\mathbf{p}_0$ to $\mathbf{p}_{n_3}$ does not meet the target visibility requirement, because the target can be obstructed by the obstacle.

We utilize Dijkstra's algorithm [17] to determine the shortest path P_D from node $\mathbf{p}_0$ to node $\mathbf{p}_T$ on the directed weighted graph $\mathcal{G}$. The number of nodes in the graph $\mathcal{G}$ is considerably less than the number of points on the initial path P_A, and thus, solving P_D will produce nearly no additional time overhead. Fig. 3 provides an example of path searching. Compared with the initial path P_A, the optimized path P_D is not only shorter in length but also positioned farther away from obstacles. Furthermore, obstacle checking between nodes improves the target visibility of optimized path P_D compared with that of the initial path P_A.

After determining the optimized path P_D, we uniformly sample $N_c - 1$ points $\{\mathbf{q}_0, \mathbf{q}_1, \cdots, \mathbf{q}_{N_c-2}\}$ on path P_D to generate a trajectory parameterized by time, where $\mathbf{q}_0 = \mathbf{p}_0$ and $\mathbf{q}_{N_c-2} = \mathbf{p}_T$.

Algorithm 1: Directed Weighted Graph Construction

Input : Path $P_A = \{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_T\}$
Output: Directed weighted graph $\mathcal{G}$

$N \leftarrow$ Number of points in path P_A;
Initialize an empty directed weighted graph $\mathcal{G}$;
Add $\mathbf{p}_0$ to the nodes of $\mathcal{G}$;
for $i \leftarrow 0$ **to** $N-2$ **do**
  if $i < N-2$ *and* $\mathbf{d}_i = \mathbf{d}_{i+1}$ **then**
    continue;
  end
  Add $\mathbf{p}_{i+1}$ to the nodes of $\mathcal{G}$;
  foreach *node* $\mathbf{p}$ *of* $\mathcal{G}$ **do**
    if $\mathbf{p} = \mathbf{p}_{i+1}$ **then**
      continue;
    end
    if *no obstacles between* $\mathbf{p}$ *and* $\mathbf{p}_{i+1}$ **then**
      Create an edge in graph $\mathcal{G}$ from node $\mathbf{p}$ to node $\mathbf{p}_{i+1}$ with weight $\|\mathbf{p} - \mathbf{p}_{i+1}\|_2$;
    end
  end
end

B. Trajectory Optimization

The trajectory of the ground omnidirectional robot is represented by $\{x(t), y(t), \psi(t)\} \in SE(2)$. We parameterize the trajectory by using a uniform B-spline curve, which is a piecewise polynomial that is uniquely determined by its degree p_b, a knot span Δt, and N_c control points $\{\mathbf{Q}_k, \Psi_k\}$. In practice, we choose $p_b = 3$, then the total duration of the trajectory is $N\Delta t$, where $N = N_c - 3$. In accordance with the properties of the third-order uniform B-spline, the relationship between $\mathbf{Q}_k$ and $\mathbf{q}_k$ is

$$\mathbf{q}_k = \frac{1}{6}\left(\mathbf{Q}_k + 4\mathbf{Q}_{k+1} + \mathbf{Q}_{k+2}\right). \tag{2}$$

The robot is omnidirectional, and thus, we decouple $\mathbf{Q}_k$ and Ψ_k. In this section, we first optimize $\mathbf{Q}_k \in T(2)$ to ensure the safety of the trajectory while satisfying motion smoothness and kinematic feasibility. For a uniform B-spline, the time interval Δt between two adjacent control points is identical, the control points of the velocity $\mathbf{V}$, acceleration $\mathbf{A}$, and jerk $\mathbf{J}$ curves are obtained by

$$\mathbf{V}_k = \frac{\mathbf{Q}_{k+1} - \mathbf{Q}_k}{\Delta t}, \mathbf{A}_k = \frac{\mathbf{V}_{k+1} - \mathbf{V}_k}{\Delta t}, \mathbf{J}_k = \frac{\mathbf{A}_{k+1} - \mathbf{A}_k}{\Delta t}. \tag{3}$$

We optimize the subset of $N_c - 2p_b + 1$ control points $\{\mathbf{Q}_{p_b}, \mathbf{Q}_{p_b+1}, \cdots, \mathbf{Q}_{N_c-p_b}\}$. The boundary state is determined by the first and last p_b control points, and thus, they should not be altered. The optimization problem is formulated as

$$\min \quad \mathcal{J} = \lambda_d \mathcal{J}_d + \lambda_s \mathcal{J}_s + \lambda_f \mathcal{J}_f, \tag{4}$$

where $\mathcal{J}_d$ is the safety penalty, $\mathcal{J}_s$ is the smoothness penalty, $\mathcal{J}_f$ is the feasibility penalty, and λ_d, λ_s and λ_f are weights for each penalty terms.

The safety penalty is designed to keep the robot away from obstacles to ensure trajectory safety. We utilize a Euclidean signed distance field (ESDF) to represent the distance from each point to obstacles. ESDF is constructed using an efficient algorithm with linear complexity [18]. We define the safety penalty $\mathcal{J}_d$ as

$$\mathcal{J}_d = \sum_{k=3}^{N} \max\left(0, \sqrt{d_s} - \sqrt{\Xi\left(\mathbf{Q}_k\right)}\right), \tag{5}$$

where $\Xi\left(\mathbf{Q}_k\right)$ is the value of $\mathbf{Q}_k$ in ESDF, and d_s is a hyper-parameter that represents the safe distance between the robot's center and obstacles.

In contrast with the commonly used square penalty, we define the safety penalty by using the radical. Compared with the square penalty, the radical penalty produces a steeper gradient and promotes rapid convergence, as depicted in Fig. 4. The gradient of $\mathcal{J}_d$ is given as

$$\frac{\partial \mathcal{J}_d}{\partial \mathbf{Q}_k} = \begin{cases} 0, & d_s < \Xi\left(\mathbf{Q}_k\right), \\ -\dfrac{1}{2\sqrt{\Xi\left(\mathbf{Q}_k\right)}} \dfrac{\mathrm{d}\,\Xi\left(\mathbf{Q}_k\right)}{\mathrm{d}\,\mathbf{Q}_k}, & d_s \geq \Xi\left(\mathbf{Q}_k\right), \end{cases} \tag{6}$$

where the gradient of $\Xi\left(\mathbf{Q}_k\right)$ can be calculated using the Sobel operator [19] and linear interpolation.

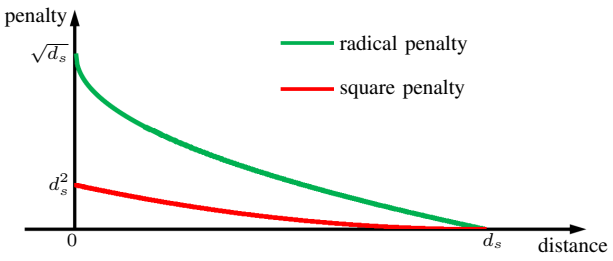


Fig. 4. We propose a radical penalty function that decreases rapidly as the distance increases and has steeper gradients than the squared penalty function described in Fast Planner [20]. In this figure, we use $d_s = 0.4$ m, which is the default value in [20].

Benefiting from the convex hull property, we follow [21] to use the high-order derivatives of the control points of the B-spline trajectory as a smoothing penalty:

$$\mathcal{J}_s = \sum_{k=1}^{N} \|\mathbf{A}_k\|_2^2 + \sum_{k=0}^{N} \|\mathbf{J}_k\|_2^2. \tag{7}$$

The purpose of the feasibility penalty is to ensure that the robot's motion trajectory is achievable. This task is accomplished by limiting velocity and acceleration:

$$\mathcal{J}_f = \sum_{k=2}^{N} f\left(\|\mathbf{V}_k\|_2^2 - v_m^2\right) + \sum_{k=1}^{N} f\left(\|\mathbf{A}_k\|_2^2 - a_m^2\right), \tag{8}$$

where v_m and a_m represent the maximum velocity and maximum acceleration of the robot, respectively, and the function $f(x) = \max(0, x)$.

IV. VISIBILITY ORIENTATION PLANNING

In this section, we plan the tracker's orientation (i.e., yaw angle) based on a safe trajectory. Assuming that a safe trajectory consists of $N+1$ points, we plan the orientation $\{\psi_1, \psi_2, \cdots, \psi_N\}$ of the last N points. The first point's orientation ψ_0 is the initial state, and thus, it cannot be altered. The optimization problem is formulated as

$$\min \quad \mathcal{L} = \mu_v \mathcal{L}_v + \mu_s \mathcal{L}_s + \mu_f \mathcal{L}_f, \tag{9}$$

where $\mathcal{L}_v$ is the visibility penalty, $\mathcal{L}_s$ is the smoothness penalty, $\mathcal{L}_f$ is the feasibility penalty, and μ_v, μ_s and μ_f are weights for each penalty terms.

Given the tracker's position (x_k, y_k) and the target's position (X, Y), the optimal orientation of the tracker to observe the target is

$$\psi_k^* = \operatorname{atan2}(Y - y_k, X - x_k). \tag{10}$$

Orientations are periodic, and thus, we design a cosine function to measure the visibility penalty:

$$\mathcal{L}_v = \sum_{k=1}^{N} \left(1 - \cos\left(\psi_k^* - \psi_k\right)\right). \tag{11}$$

In accordance with the properties of trigonometric functions, the derivative of $\mathcal{L}_v$ with respect to ψ_k is

$$\frac{\partial \mathcal{L}_v}{\partial \psi_k} = \sin\left(\psi_k - \psi_k^*\right). \tag{12}$$

In contrast with the changes in position, the orientation of the robot is periodic, and the orientation change can be expressed as

$$\Delta\psi_k = \begin{cases} \psi_{k+1} - \psi_k, & -\pi < \psi_{k+1} - \psi_k \le \pi, \\ \psi_{k+1} - \psi_k + 2\pi, & \psi_{k+1} - \psi_k \le -\pi, \\ \psi_{k+1} - \psi_k - 2\pi, & \psi_{k+1} - \psi_k > \pi. \end{cases} \tag{13}$$

Then, angular velocity ω, angular acceleration α, and angular jerk ζ are obtained by

$$\omega_k = \frac{\Delta\psi_k}{\Delta t}, \alpha_k = \frac{\omega_{k+1} - \omega_k}{\Delta t}, \zeta_k = \frac{\alpha_{k+1} - \alpha_k}{\Delta t}. \tag{14}$$

Similar to (7) and (8), the smoothness penalty and feasibility penalty are defined as

$$\mathcal{L}_s = \sum_{k=0}^{N-2} \alpha_k^2 + \sum_{k=0}^{N-3} \zeta_k^2, \tag{15}$$

$$\mathcal{L}_f = \sum_{k=0}^{N-1} f\left(\omega_k^2 - \omega_m^2\right) + \sum_{k=0}^{N-2} f\left(\alpha_k^2 - \alpha_m^2\right), \tag{16}$$

where ω_m and α_m represent the maximum angular velocity and maximum angular acceleration of the robot, respectively.

V. SIMULATIONS

A. Implementation Details

We used the Gazebo simulator to build a simulation environment that measured $8\text{ m} \times 4.5\text{ m}$ to evaluate the performance of the SF-Tracker. The robot's shape was cuboid, and its projection on the ground formed a square that measured $0.5\text{ m} \times 0.5\text{ m}$. All the simulations were performed on an Intel NUC Phantom Canyon equipped with an Intel Core i7-1165G7 CPU and an NVIDIA GeForce RTX 2060 GPU. We set the safe distance to $d_s = 0.4$ m[1], which is consistent with [20]. The other hyper-parameters are provided in Tab. I. In the simulations, we directly fed the current position of the target to the tracker. All unconstrained optimization problems were solved using the L-BFGS algorithm [22].

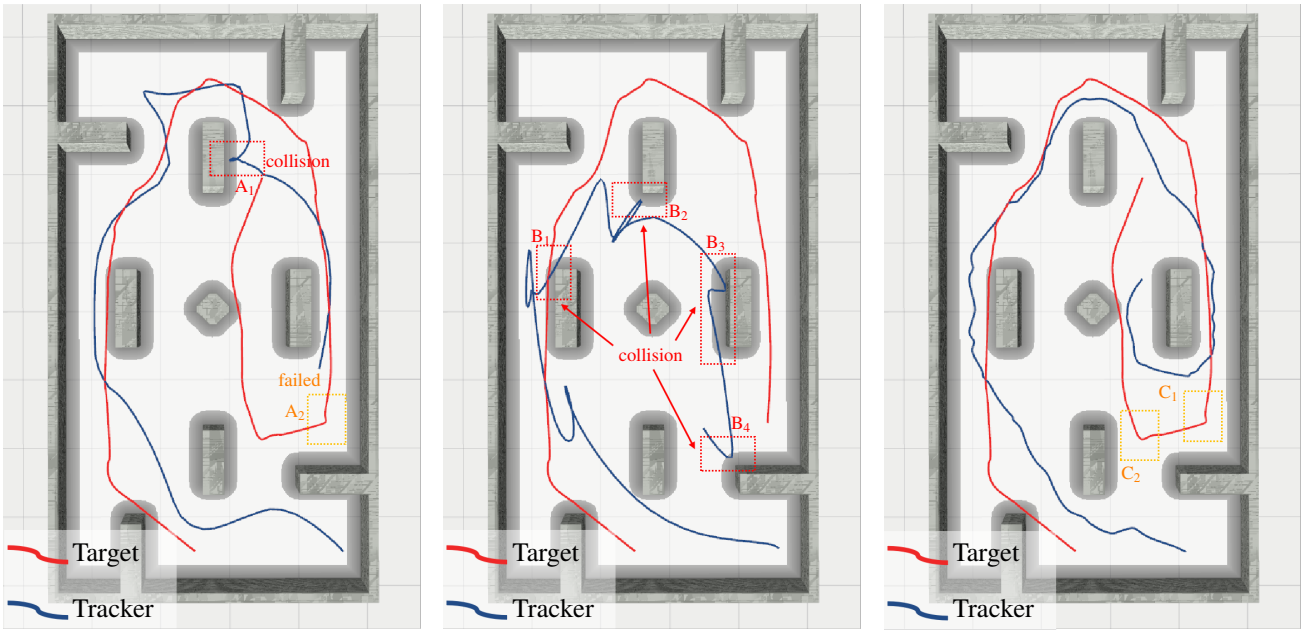
TABLE I
HYPER-PARAMETERS OF EXPERIMENTS

Name	Values
maximum velocity	$v_m = 2.5$ m/s, $\omega_m = 1.5$ rad/s
maximum acceleration	$a_m = 1.5$ m/s^2, $\alpha_m = 1.0$ rad/s^2
weights of penalty $\mathcal{J}$	$\lambda_d = 100$, $\lambda_f = 0.01$, $\lambda_s = 10^{-6}$
weights of penalty $\mathcal{L}$	$\mu_v = 100$, $\mu_f = 0.1$, $\mu_s = 0.01$

B. Benchmarks

We benchmarked the SF-Tracker by using Vis-Planner [4] and Elastic Tracker [5] in a dense simulation environment. For all the trackers, the map of the environment is unknown. The target is controlled by a human who uses a keyboard, and moves at a velocity of 1.5 m/s. We disregarded the collision volume between the target and obstacles. During the test, the

[1]The value is determined by the size of the robot. In our experiments, the maximum distance from the robot's center to obstacles when a collision occurs is $\sqrt{2} \times 0.25\text{ m} \approx 0.35\text{ m} < 0.4\text{ m}$.



(a) Vis-Planner (b) Elastic Tracker (c) SF-Tracker

Fig. 5. Comparison of tracking trajectories of three methods. The trajectories of the tracker (shown as blue curves) and the target (shown as red curves) both begin at the lower right corner. The figures display areas that are less than 0.25 m away from the obstacles in gray, and the side length of the grid is 1 m. **(a) Vis-Planner**: At A_1, Vis-Planner retreats to maintain the best observation distance from the target, but results in a collision. At A_2, the target suddenly changes direction, causing the tracking of Vis-Planner to fail. **(b) Elastic Tracker**: In our dense environment, Elastic Tracker continuously collides at B_1-B_4. At B_2 and B_3, Elastic Tracker loses target visibility, but its ability to predict target trajectory allows it to avoid tracking failure. **(c) SF-Tracker**: The proposed SF-Tracker always maintains a safe distance from obstacles during tracking. At C_1 and C_2, the SF-Tracker is able to track the target even if the target suddenly changes direction, maintaining target visibility and trajectory safety.

target moved quickly around obstacles, and the tracker had to track it while avoiding obstacles. The historical trajectories of the target and the tracker are shown in Fig. 5. The proposed SF-Tracker can avoid all the obstacles during tracking, while Vis-Planner and Elastic Tracker collide (marked with red boxes in Fig. 5). In cases wherein the target suddenly changes direction (marked with orange boxes in Fig. 5), Vis-Planner tracking fails, while SF-Tracker continues to maintain target visibility. Fig. 6 displays the distance from the robot's center to the obstacles during tracking. When the distance was close to 0.25 m, the robot collides with an obstacle. During testing, SF-Tracker consistently maintains a safe distance from the obstacles, while Vis-Planner and Elastic Tracker always collide.

Tab. II summarizes the quantitative evaluation of the trajectory safety and target visibility of the three methods. For trajectory safety, we count the proportion of the trajectory within the dangerous zones, and the distances from all points on the trajectory to obstacles. The dangerous zone is defined as the region wherein distance from the obstacles is less than the safe distance d_s. The trajectory of the SF-Tracker always steers clear of obstacles, while the trajectories of Vis-Planner and Elastic Tracker frequently enter dangerous zones. The distance between SF-Tracker's trajectory and obstacles is comparable with that of Vis-Planner, but Vis-Planner collides. Moreover, SF-Tracker exhibits the smallest standard deviation in distance, indicating the highest stability. For target visibility, we define the yaw error at the k-th point on the trajectory as $\psi_\varepsilon = |\psi_k^* - \psi_k|$, where ψ_k^* is calculated using (10), and the subtraction operation is defined like (13). The average yaw errors of the three methods are provided in the last column of Tab. II. The proposed SF-Tracker achieves a yaw error that is comparable with those of state-of-the-art methods while ensuring a more satisfactory trajectory safety.

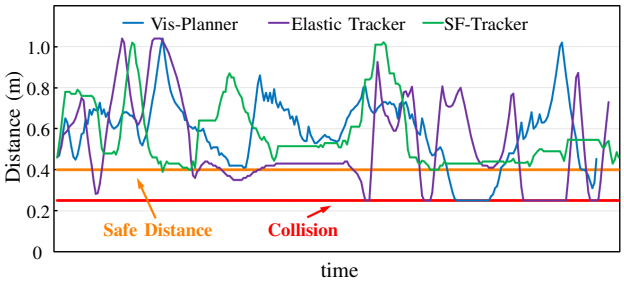


Fig. 6. Comparison of the distance from the robot's center to the obstacles. The distance between the trajectory of the SF-Tracker and the obstacles (shown in green) will always be greater than the safe distance (shown in orange straight line), satisfying trajectory safety. By contrast, the trajectories of Vis-Planner (shown in blue) and Elastic Tracker (shown in purple) frequently approach obstacles and result in collisions.

TABLE II
COMPARISON OF SAFETY AND VISIBILITY

Method	Dangerous	Distance (m)	Yaw Error (rad)
Vis-Planner	12.36%	0.58 ± 0.17	0.06π
Elastic Tracker	28.57%	0.51 ± 0.23	**0.01π**
SF-Tracker	**2.88%**	**0.57 ± 0.15**	0.04π

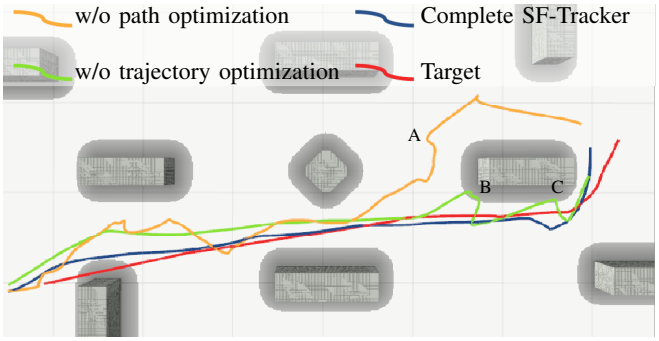


Fig. 7. Trajectories in ablation experiment. The trajectories of the tracker and the target (shown as a red curve) both begin in the lower left corner. The tracker without the path optimization module (shown as an orange curve) loses visibility of the target at A because it is obscured by the obstacle. The tracker without the trajectory optimization module (shown as a green curve) collides at B and C due to the trajectory is too close to the obstacle. In contrast, the complete SF-Tracker (shown as a blue curve) has no collisions and maintains target visibility.

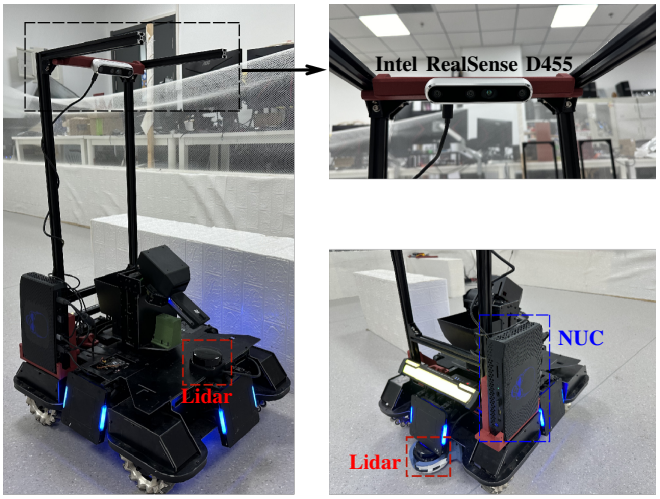


Fig. 8. Ground omnidirectional robot used in real-world experiments.

C. Ablation Studies

We conducted extensive ablation experiments to demonstrate that each module of the SF-Tracker is essential, and the absence of any one of these modules will affect the tracking performance of the SF-Tracker. First, we replaced the path search module of the SF-Tracker with a simple A* algorithm [16]. In addition, we removed the trajectory optimization module and directly used the original B-spline to represent the final trajectory. The historical trajectories of the target and the tracker in both cases are shown in Fig. 7. The reference path generated by the A* algorithm does not consider obstacles that obstruct the target, causing the tracker to lose visibility of the target. Furthermore, Removing the trajectory optimization module results in trajectories that are too close to obstacles, leading to frequent collisions and compromising trajectory safety. By contrast, the complete SF-Tracker can perform satisfactory tracking, ensuring trajectory safety and target visibility.

D. Speed Evaluation

Speed in replanning is critical due to the rapid movement of the target. Tab. III summarizes the duration of each process in the SF-Tracker. We utilized a multi-thread concurrency technology to simultaneously perform path searching and ESDF construction, because they are independent of each other. We observed that the entire pipeline of our SF-Tracker can run at an average rate of 197 Hz, demonstrating its real-time performance.

TABLE III
DURATION OF EACH PROCESS

Process	Duration (ms)	Frequency (Hz)
Path Search	1.33 ± 0.36	753
ESDF Construction	0.82 ± 0.15	1220
Trajectory Optimization	2.49 ± 1.39	400
Orientation Optimization	1.23 ± 0.74	810
Pipeline	$\mathbf{5.07 \pm 1.75}$	$\mathbf{197 \pm 73}$

VI. REAL-WORLD EXPERIMENTS

A. Implementation Details

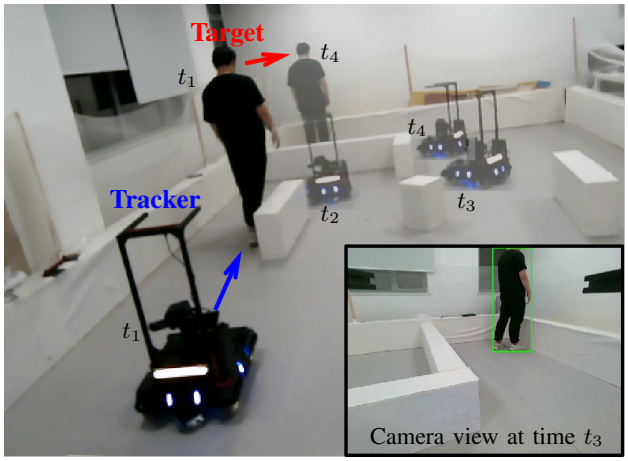
We tested the proposed SF-Tracker in unknown, dense indoor environments. Our ground omnidirectional robot (shown in Fig. 8) is equipped with an Intel RealSense D455 camera for target recognition, two single-line lidars for obstacle detection, and an Intel NUC Phantom Canyon for all computing tasks. We use the ira_laser_tools package [23] to merge data from the two lidars. To achieve visual recognition, we utilize our previous work HiT [24] to track arbitrary targets. The position of the target is determined using depth information. In real-world experiments, we set the maximum velocity of the robot to $v_m = 1.5$ m/s, the maximum angular velocity to $\omega_m = 1$ rad/s, the maximum acceleration to $a_m = 1$ m/s^2, and the maximum angular acceleration to $\alpha_m = 1$ rad/s^2. The other hyper-parameters remained consistent with those used in the simulations.

B. Test in Static Environments

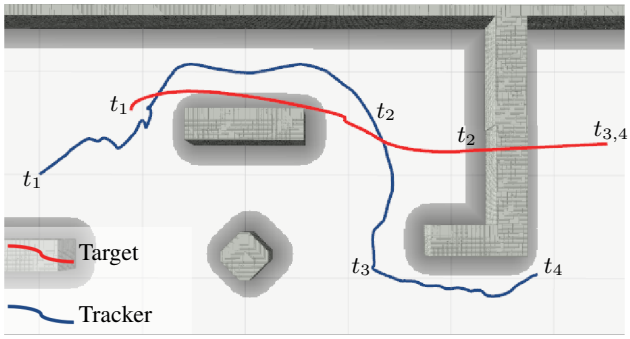
We first test our method in static environments. During the test, a human controlled the target by using a remote control, and the target moved at a maximum velocity of 1.5 m/s. The historical trajectories of the tracker and the target are shown in Fig. 1(a). The potential collision zone in the figure is defined as the region wherein the distance from the obstacles is less than half of the diagonal length of the robot's projection on the ground[2]. The target suddenly changed direction at times t_1 and t_3, and the tracker can still maintain target visibility. At time t_4, even if the target entered a narrow channel, the tracker could ensure trajectory safety and continue tracking. During the tracking process, the tracker successfully avoided all potential collision zones and maintained a clear view of the target, achieving satisfactory trajectory safety, target visibility, and motion smoothness.

[2]The projection of our real robot on the ground forms a rectangle that measuring 0.6 m $\times$ 0.45 m, with half of the diagonal length being $0.5 \times \sqrt{0.6^2 + 0.45^2}$ m $= 0.375$ m $< d_s$.

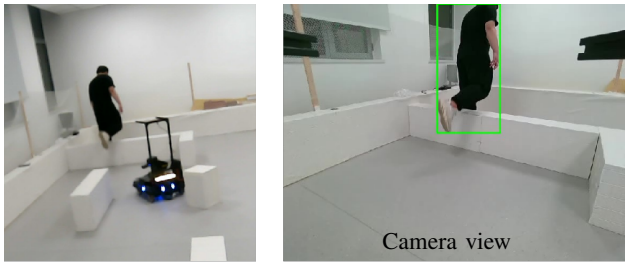
Fig. 9. The robot continuously tracks a person in a cluttered dynamic environment. At times t_1, t_3, and t_5, the person moves obstacles (marked with red circles and arrows) to obstruct the robot's tracking, posing significant safety challenges to tracking. The robot senses changes in the environment, replans a new trajectory in real time, and avoids obstacles (marked with black arrows) while maintaining target visibility. The robot continuously tracks the target from t_1 to t_6 while ensuring trajectory safety, demonstrating the practicality of the SF-Tracker in extreme situations with safety challenges.



(a) Keyframes during the tracking process.



(b) Trajectory history of the tracker and the target.



(c) The person is going directly over the obstacle.

Fig. 10. The robot tracks the person in an unknown static dense environment. Tracking begins at time t_1. The target crosses the obstacle at time t_2, and the tracker promptly navigates around the obstacle while maintaining clear target visibility. The trajectory of the target is determined by the visual object tracking model. We project it onto the map coordinate system for display.

We also tested the SF-Tracker's ability to track a person in dense environments. During the test, the person can go directly over obstacles, as shown in Fig. 10(c), while the robot must navigate around obstacles. The entire tracking process is depicted in Fig. 10. The camera view at time t_3 shows that the robot maintains clear target visibility while navigating around an obstacle. We recorded the historical trajectories of the robot and the person, as shown in Fig. 10(b). The robot can track the person and avoid all the obstacles, even if the person suddenly crosses obstacles.

C. Test in Dynamic Environment

In practical applications, the environment is always subject to dynamic changes. To demonstrate the practicality of the SF-Tracker, we tested its ability to track a person in a real cluttered dynamic environment, as illustrated in Fig. 9. During the test, the person moved obstacles to obstruct the robot's tracking. When the environment changes, our SF-Tracker can replan a new safe trajectory in real-time and continue to track the target along the new trajectory while maintaining target visibility.

VII. CONCLUSION AND LIMITATIONS

This paper introduces a new tracking framework for omni-directional robots, i.e., SF-Tracker, which effectively avoids collisions in complex dynamic environments while preserving target visibility and motion smoothness. It addresses the challenge of achieving a balance between trajectory safety and target visibility, a predicament that previous methods have found challenging, enabling the stable tracking of targets. Extensive experiments demonstrate that our SF-Tracker achieves promising tracking performance compared to state-of-the-art methods while operating in real time. We hope that this work can enhance the practical applicability of the tracker and inspire further research on the trajectory safety of tracking.

One limitation of the SF-Tracker is as follows: despite achieving competitive performance, it faces challenges in handling occlusions of the target, because the method does not incorporate an explicit occlusion-handling module. In addition, this work primarily focuses on trajectory planning, with less emphasis on predicting the motion of the target. Our future work will delve into these aspects to further enhance the performance of the SF-Tracker.

REFERENCES

[1] Z. Zhou, S. Zhu, K. Zhu, C. Cheng, and J. Gu, "A small family service robot system with uncalibrated monocular camera for visual servoing tracking fast moving family targets in short range," *Procedia Computer Science*, vol. 209, pp. 50–57, 2022.

[2] D. Cafolla, "A 3D visual tracking method for rehabilitation path planning," in *New Trends in Medical and Service Robotics: Advances in Theory and Practice*, 2019, pp. 264–272.

[3] Z. Han, R. Zhang, N. Pan, C. Xu, and F. Gao, "Fast-tracker: A robust aerial system for tracking agile target in cluttered environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 328–334.

[4] Q. Wang, Y. Gao, J. Ji, C. Xu, and F. Gao, "Visibility-aware trajectory optimization with application to aerial tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5249–5256.

[5] J. Ji, N. Pan, C. Xu, and F. Gao, "Elastic tracker: A spatio-temporal trajectory planner for flexible aerial tracking," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 47–53.

[6] S. Bhagat and P. Sujit, "UAV target tracking in urban environments using deep reinforcement learning," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 694–701.

[7] A. Maalouf, N. Jadhav, K. M. Jatavallabhula, M. Chahine, D. M. Vogt, R. J. Wood, A. Torralba, and D. Rus, "Follow anything: Open-set detection, tracking, and following in real-time," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3283–3290, 2024.

[8] H. Cheng, L. Lin, Z. Zheng, Y. Guan, and Z. Liu, "An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1732–1738.

[9] N. Pan, R. Zhang, T. Yang, C. Cui, C. Xu, and F. Gao, "Fast-tracker 2.0: Improving autonomy of aerial tracking with active vision and human location regression," *IET Cyber-Systems and Robotics*, vol. 3, no. 4, pp. 292–301, 2021.

[10] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. Scherer, "Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming," in *International Symposium on Experimental Robotics*, 2020, pp. 119–129.

[11] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, 2017.

[12] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3725–3732, 2018.

[13] B. F. Jeon and H. J. Kim, "Online trajectory generation of a MAV for chasing a moving target in 3D dense environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1115–1121.

[14] B. Jeon, Y. Lee, and H. J. Kim, "Integrated motion planner for real-time aerial videography with a drone in a dense environment," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1243–1249.

[15] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, pp. 3–27, 2015.

[16] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[17] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[18] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[19] I. Sobel, G. Feldman, *et al.*, "A 3x3 isotropic gradient operator for image processing," *a talk at the Stanford Artificial Project in*, pp. 271–272, 1968.

[20] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[21] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2020.

[22] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[23] A. L. Ballardini, S. Fontana, A. Furlan, and D. G. Sorrenti, "ira_laser_tools: a ROS laserscan manipulation toolbox," *arXiv preprint arXiv:1411.1086*, 2014.

[24] B. Kang, X. Chen, D. Wang, H. Peng, and H. Lu, "Exploring lightweight hierarchical vision transformers for efficient visual tracking," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 9612–9621.