2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# *Follow Everything:* Goal-Aware Adaptation and Graph-Based Planning Towards Arbitrary Leader Following

Qianyi Zhang[1], Shijian Ma[2], Boyi Liu[3], Jingtai Liu[1], Jianhao Jiao[†,4,5], and Dimitrios Kanoulas[†,4]

***Abstract*— Enabling robots to robustly follow leaders supports tasks such as carrying supplies or guiding customers. While existing methods often fail to generalize to arbitrary leaders, and struggle when the leader temporarily leaves the robot's field of view, this work presents a unified framework to address both challenges. First, a segmentation model replaces traditional category-specific detection models, allowing the leader to be of any shape or type. To improve robustness, a distance frame buffer is designed to store high-confidence leader embeddings across distance intervals, accounting for the unique characteristics of leader-following tasks. Second, a goal-aware adaptation mechanism is designed to govern robot planning states based on the leader's visibility and motion, complemented by a graph-based planner that generates candidate trajectories for each state, ensuring efficient following with obstacle avoidance. Simulations and real-world experiments with a legged robot follower and diverse leaders in indoor and outdoor settings demonstrate an improved follow success rate of 75.1%, a reduced visual loss of 13.1%, a fewer collisions of 65.1%, and a shorter average distance of 0.4 m. Visit https://follow-everything.github.io for the video and code.**

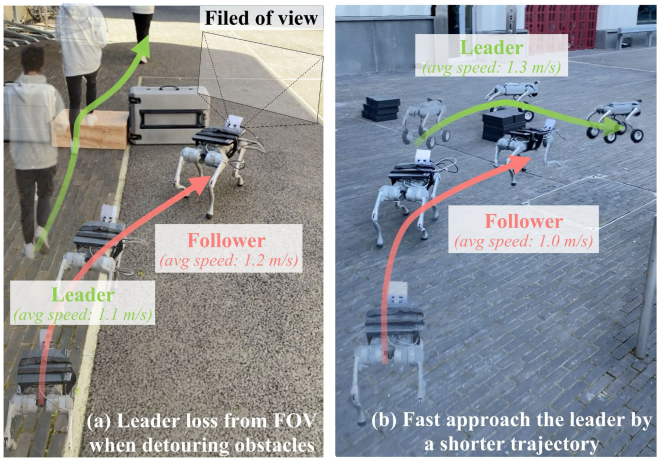



Fig. 1. Illustration of the challenges in leader-following tasks. (a) When the leader steps over a box that the follower must detour around, the leader leaves the follower's field of view. (b) The follower is expected to follow the leader via a shorter trajectory, not by replicating the leader's past path.

## I. Introduction

Recent advances in visual perception, robot navigation, and large models have enabled robots to follow leaders with increased safety and intelligence [1]. Serving as powerful assistants, robots can follow explorers to carry more supplies, guide customers and introduce products in shopping centers, help police patrolling, etc. [2]–[5]. Despite extensive efforts to improve the stability of leader following [6]–[9] and the smoothness of robot motion [10]–[15], existing approaches still either fail to re-identify the leader after it temporarily leaves the robot's field of view (FOV) and later reappears (see Fig.1a), or they cannot approach the leader in the most efficient manner (see Fig.1b). This paper breaks down these challenges from both the perception and planning perspectives and proposes the distance frame buffer, goal-aware adaptation, and graph-based planner to address them.

The challenge on the perception side involves re-identifying the leader after it exits and re-enters the follower's FOV. One specific approach for leader-following tasks is to have the leader carry a beacon, such as infrared antennas [16] or wave emitters [17], [18], but requiring the leader to carry additional equipment is often impractical and not generalizable. Therefore, another widely used solution is to leverage detection models based on cameras [7], LiDAR [8], multimodal sensors [9], sonar [19], and so on. However, since their training labels correspond to object categories rather than object features, when the leader (typically a pedestrian) remains in the follower's FOV, its ID remains stable due to the help of the Kalman Filter [20]. However, when the leader leaves and reappears, a new ID will be assigned to it. This leads to the expected leader having the original ID permanently lost.

An alternative solution is segmentation models [21] that use features as labels: with an initial feature input, the model tracks and bootstrap-updates the leader's features continuously, enabling re-identification upon its reappearance [10]. These models are not category-limited and can track arbitrary-shaped leaders (humans, vehicles, robots, etc.). However, a limitation of existing segmentation datasets is that they often involve a fixed camera position (follower) with the leader's movement, location and size constantly changing in the video, where bootstrapping is beneficial.

However, in the leader-follower task, the follower is also in motion, making it easy for the leader to leave the follower's FOV, as illustrated in Fig. 1(a). Just before leaving, the leader often appears on the edge of the robot's FOV, resulting in partial and low-quality features. Despite their poor quality, these features can still dominate the bootstrapping process,

[1]Institute of Robotics and Automatic Information System, Nankai University, China. [2]Centre for Data Science, University of Macau, China. [3]Electrical and Computer Engineering Department, Hong Kong University of Science and Technology, China. [4]Department of Computer Science, University College London, UK. [5]Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, China.
[†] The corresponding authors. This work was done during Qianyi, Shijian, and Boyi's visit at University College London. This work is supported by UK Research and Innovation Future Leaders Fellowship (RoboHike, Grant No. MR/V025333/1) and in part by the National Natural Science Foundation of China under Grant 62573244.

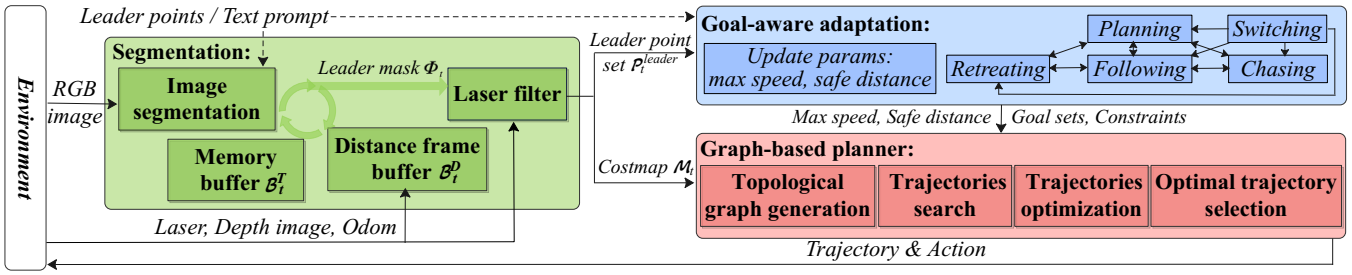



Fig. 2. Illustration of the proposed *follow everything* framework. Given a leader prompt and an RGB image, the leader mask is segmented with the aid of both a memory buffer and a distance frame buffer. The segmented leader is then filtered from raw laser point clouds to produce a leader point set, which is passed to the goal-aware adaptation to dynamically adjust parameters and provide goal sets and constraints according to the follower-leader interaction. Given the costmap, goal sets, and constraints, the graph-based planner searches, optimizes, and selects an optimal trajectory for the follower to execute.

since the older features that capture the full-body shape of the leader differ significantly from the current appearance of the leader [10]. This can mislead the robot to bootstrap in the wrong direction. As a result, when the leader reappears later, the robot may not be able to retrace the leader accurately, even if the full leader is in front of it. To address this issue, this work introduces the distance frame buffer to SAM2 (see Sec. II-B), which divides the distance ranges between the follower and leader into equal intervals and maintains the historical leader feature with the highest confidence in each distance interval, greatly improving the ability to retrace the leader after it reappears.

The challenge on the planning side involves obstacle avoidance and robust motion planning when the relationship between the leader and the follower is dynamic. While most existing approaches simply treat the leader's position as a target and use PID [10], MPC [11], sampled trajectories [12], optimized trajectories [13], or learning-based methods [15] to plan trajectories for the follower (robot), they often lack robustness and fail to consider the various interactive states that arise from the leader-follower relationship. For example, (a) when the leader is within the follower's FOV and close by, the main goal of the follower is to avoid obstacles while following, a scenario that frequently occurs when a legged robot follows a UAV leader, as shown in Fig. 6, where the UAV can fly over short obstacles that the ground robot must detour around. (b) When the leader is within the follower's FOV but at a greater distance, the main goal of the follower is to quickly approach the leader, which requires the robot to maintain high speed and choose optimal time trajectories to avoid increasing the risk of losing the leader due to excessive distance, as shown in Fig. 8. (c) When the leader retreats, the follower should also retreat accordingly while maintaining a safe margin to avoid collisions with the leader or losing sight of it due to too-close proximity, as shown in Fig. 5d. (d) When the leader leaves the follower's FOV, the follower should intentionally approach the last known position and orientation of the leader to recover it as quickly as possible, as illustrated in Fig. 9f-g. (e) A robust model should also be able to replace the leader if necessary, improving the practicality. To realize these robust following capabilities, this work designs a goal-aware adaptation (see Sec.II-D) under the proposed *follow everything* framework, which dynamically adjusts the robot's speed and the desired safe distance between the leader and the follower, and adaptively determines the follower's state to provide unique goals and constraints for the subsequent trajectory planner.

As for the planner, the primary concern is to find a time-optimal trajectory for the follower while adapting to the goals and constraints provided by the goal-aware adaptation. Building on previous work [22], [23], the environment can be modeled as a graph, with obstacles clustered as nodes and their shortest connections as edges. Given that each node has different detour directions, a series of candidate trajectories with different meanings [24], [25] can be generated. Besides kinematic constraints, the goal sets and unique constraints provided by goal-aware adaptation are incorporated into the optimization. The trajectory with the shortest time to reach the goal is selected as the best trajectory to execute. This graph-based planner (see Sec. II-C) enables the follower to approach the leader efficiently, rather than simply replicating their historical paths, as shown in Fig. 1(b).

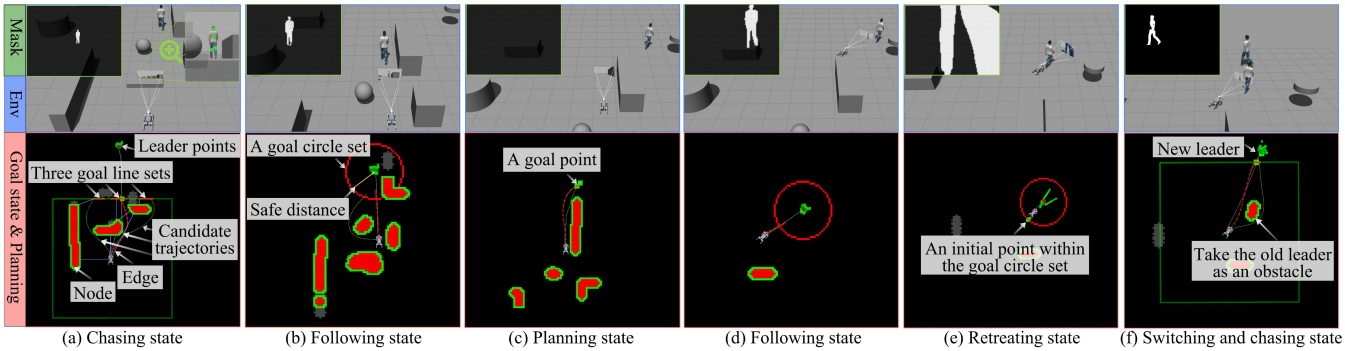
In summary, this work presents a unified framework for robust leader-following and obstacle avoidance. The main contributions are as follows:

- A segmentation module with a distance frame buffer is introduced to enable everything to be a leader.
- A goal-aware adaptation strategy is proposed to enable state switching for robust leader-following.
- A graph-based trajectory planner is developed to produce candidate trajectories for safe following.

## II. Methodology

### A. Problem Formulation

Given the RGB image $I_t$ at timestamp $t$, the leader is segmented into an embedding $\eta_t$ and a mask $\phi_t$ with the assistance of a temporal memory buffer $\mathcal{B}_t^T$ and a distance frame buffer $\mathcal{B}_t^D$. Using the depth image, the leader mask $\phi_t$ is transformed into a point set $\mathcal{P}_t^{leader} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$, where each point $\mathbf{p}_i = (x, y, z)$ represents the 3D coordinates of the leader. The leader's position $\bar{\mathbf{p}}_t^l$ is obtained by averaging the point set $\mathcal{P}_t^{leader}$, and the leader's velocity $\bar{\mathbf{v}}_t^l$ is estimated based on the positional difference over a fixed time interval. Given the robot's position $\mathbf{p}_t^r$ and the leader's states




(a) Chasing state (b) Following state (c) Planning state (d) Following state (e) Retreating state (f) Switching and chasing state

Fig. 3. Illustration of goal-aware adaptation and graph-based planner in a demo. (a) The leader is segmented after selecting points. *Chasing* is activated as the leader is far away. (b) *Following* is triggered when the robot gets closer, maintaining a safe distance. (c) *Planning* is activated when the leader exits the robot's FOV, guiding the robot to the last known leader pose. (d) *Following* resumes once the leader is visible again. (e) *Retreating* is triggered when the leader steps back. (f) A new prompt "follow the person on the left side" activates *Switching*, followed by *Chasing* as the new leader is distant.

$\bar{\mathbf{p}}_t^l$ and $\bar{\mathbf{v}}_t^l$, the goal-aware adaptation mechanism determines the robot's maximum speed $V_t^{max}$ and the safe distance $D_t$ between the robot and the leader. Additionally, the robot's goal is provided as a series of sets $\{\mathcal{G}_t^0, \mathcal{G}_t^1, ..., \mathcal{G}_t^n\}$, each associated with the desired orientations. A topological graph is constructed based on a costmap $\mathcal{M}_t$, where obstacle groups are defined as nodes $\mathcal{N}_i$ and the shortest connections between groups are defined as edges $\mathcal{E}_{i,j}$. This graph generates several generalized trajectories $\mathcal{T}_i$, which are then transformed into normal trajectories $\tau_i$. By optimizing these trajectories with constraints imposed by the robot's kinematics, goal sets $\mathcal{G}_t^i$, maximum speed $V_t^{max}$, and safe distance $D_t$, the optimal trajectory is selected and executed.

### B. Leader Segmentation with Distance Frame Buffer

Given an initial RGB image $I_0$ and a set of leader point prompts (often manually selected; see Fig. 3a), the segmentation model [21] expands the prompts to the entire leader, producing a leader embedding $\eta_0$ and a leader mask $\phi_0$. Then, at each subsequent timestep $t$, the top $n$ historical leader embeddings with the highest confidence scores are maintained and updated in a temporal memory buffer $\mathcal{B}_t^T$:

$$\mathcal{B}_t^T = \{\eta^1, ...\}_{n_1},\ \eta^i = \underset{\forall \eta_t \cap \eta_t \notin [\eta^1, ..., \eta^{i-1}]}{\arg\max} S(\eta_t), \quad (1)$$

where $\forall \eta_t$ contains all historical embeddings and $S(\eta_t)$ is the confidence score. The distance frame buffer $\mathcal{B}_t^D$ considers the spatial dimension. It has a length of $n_2$, with each element $\eta^i$ storing the leader embedding with the highest confidence score from historical leader embeddings within the distance range $[(i-1)\Delta d, i\Delta d]$ between the follower and the leader:

$$\mathcal{B}_t^D = \{\eta^1, ...\}_{n_2}, \eta^i = \underset{\forall \eta_t \cap d(\eta_t) \in [(i-1)\Delta d, i\Delta d]}{\arg\max} S(\eta_t), \quad (2)$$

where $\Delta d$ is a distance interval, and the distance between the follower and the leader is defined as $d(\eta_t) = |\bar{\mathbf{p}}_t^l - \mathbf{p}_t^f|$. Given the current image $I_t$ and the two buffers, the current leader embedding $\eta_t$ and mask $\phi_t$ can be generated:

$$\eta_t, \phi_t = \text{Segmentation}(I_t, \mathcal{B}_t^T, \mathcal{B}_t^D). \quad (3)$$

The introduction of the distance frame buffer enables the segmentation model to isolate embeddings based on distance, reducing the contamination of the entire buffer when bootstrap occurs in the wrong direction, such as when the leader is partially observable or in the process of leaving the follower's FOV. Its effectiveness can be seen in Table. I, where our Follow Everything achieves the highest follow success rate metric compared to Alaa and FE-N-DFB.

### C. Graph-based Planner

The graph-based planner is based on time-optimal optimization [26] and provides a unified framework to incorporate various constraints of goal-aware adaptation, making the decision-making of the robot consistent even when motion states frequently alternate. Filtering the leader points $\mathcal{P}_{leader}$ from the laser scan, the remaining points form a binary costmap $\mathcal{M}_t$. Following prior work [22], [23], adjacent obstacle points in $\mathcal{M}_t$ are clustered into obstacle groups, each outlined by a boundary. A topological graph is established with the obstacle groups as nodes $\mathcal{N}_i$. For any two boundaries with indices $i$ and $j$, their shortest collision-free connection is defined as an edge $\mathcal{E}_{i,j}$ (see Fig. 3a) of the graph. Given the goal point or goal sets $\{\mathcal{G}_t^0, \mathcal{G}_t^1, ...\}$ from the goal-aware adaptation, which will be detailed later, several generalized trajectories can be generated over the topological graph using depth-first search, with each one $\mathcal{T}$ following the form:

$$\mathcal{T} = \mathbf{p}_t^r \rightarrow \mathcal{E}_{r,i} \rightarrow \widehat{\mathcal{N}}_i \rightarrow \mathcal{E}_{i,j} \rightarrow \widehat{\mathcal{N}}_j \rightarrow \ldots \rightarrow \mathcal{G}_k. \quad (4)$$

Since each obstacle node $\mathcal{N}_i$ can be detoured in clockwise or counterclockwise directions, a generalized trajectory $\mathcal{T}$ containing $n$ nodes can produce up to $2^n$ distinct trajectories $\tau$. To eliminate redundant trajectories that represent the same detour behavior, the trajectory homotopy classes [24], [25] are applied. The remaining trajectories are then optimized in parallel. For each trajectory $\tau = \{\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_m\}$, it follows the time-optimal optimization framework:

$$\tau = \underset{\{\mathbf{p}_1, ..., \mathbf{p}_m\}}{\arg\min} \sum_{t=1}^{m} ||\mathbf{p}_i - \mathbf{p}_{i-1}||_2 / V_t^{max}. \quad (5)$$

$$\text{s.t.} \quad \mathcal{O}_{dynamic}(\mathbf{p}_i) = 0, \quad \mathcal{O}_{static}(\mathbf{p}_i) = 0, \quad (6)$$

$$\mathcal{H}(\mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}) = 0, \quad \mathcal{A}(\mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}) \leq 0, \quad (7)$$

$$\mathcal{I}(\mathbf{p}_m, \mathcal{G}_i) \leq 0, \quad \mathcal{V}(\mathbf{p}_i, \mathbf{p}_{i+1}, V_t^{max}) \leq 0, \quad (8)$$

where $\mathcal{O}_{dynamic}(\cdot)$ and $\mathcal{O}_{static}(\cdot)$ represent the constraints for avoiding dynamic [22] and static [23] obstacles, respectively. $\mathcal{H}(\cdot)$ and $\mathcal{A}(\cdot)$ denote the kinematic and acceleration constraints [26], respectively. $\mathcal{I}(\cdot)$ and $\mathcal{V}(\cdot)$ are the goal and velocity constraints, both of which are provided by the goal-aware adaptation and will be detailed in the next subsection.

All the trajectories are evaluated, and the one $\tau_t^*$ with the lowest cost is selected and executed by the robot:

$$\begin{aligned} \tau_t^* &= \arg\min_{\tau} f(\tau_{t-1}^*, \tau) g(\tau), \\ f(\tau_{t-1}^*, \tau) &= (1-\alpha) + \alpha \sum_{i=1}^{l} \frac{val_i(\tau_{t-1}^*) - val_i(\tau)}{l}, \end{aligned} \tag{9}$$

where $f(\tau_{t-1}^*, \tau)$ evaluates the similarity between the previous optimal trajectory $\tau_{t-1}^*$ and the current candidate $\tau$, and $g(\tau)$ evaluates the time cost of the trajectory, as defined in Equ.(5). In a scenario with $l$ obstacle groups, a trajectory is assigned a homology signature $\{val_1, val_2, ..., val_l\}$, where each $val_i \in \{1, -1\}$ indicates whether the trajectory detours the $i$-th obstacle group in a clockwise or counter-clockwise direction, respectively. $\alpha$ is a scaling factor. This design enhances the robustness of trajectory selection, even when the goal sets dynamically change due to goal-aware adaptation.

### D. Goal-aware Adaptation

The goal-aware adaptation governs the goal constraint $\mathcal{I}(\cdot)$ and velocity constraint $\mathcal{V}(\cdot)$ during trajectory optimization, balancing safety in avoiding obstacles and efficiency in following the leader. The transition between goal states is determined by conditions, including whether the leader is within the robot's FOV, whether it lies within the costmap, the robot-leader distance, and whether a new leader is assigned. See Fig. 2 for possible transitions between states.

**Chasing State** (Fig. 3a) is activated when the leader is within the robot's FOV but outside the costmap range, where the robot's primary task is to navigate around nearby obstacles and chase the leader as quickly as possible. Unlike conventional methods that select the local goal as the intersection point between the global path and the costmap edge, the chasing state extends this point along the costmap edge to form a goal line of length $L_t$, which is proportional to the distance between the robot $\mathbf{p}_r$ and the leader $\bar{\mathbf{p}}_l$:

$$L_t = \alpha(|\mathbf{p}_t^r - \bar{\mathbf{p}}_t^l| - \frac{W_{map}}{2}), \tag{10}$$

where $W_{map}$ is the costmap width and $\alpha$ is a scaling factor.

The whole goal line is separated at the locations where it intersects with obstacles, resulting in several sub-goal line sets $\mathcal{G}_i = \{\mathbf{p}_1^g, \mathbf{p}_2^g, ..., \mathbf{p}_{n_i}^g\}$, which are then added as additional nodes to the topological graph along with their corresponding connections. Therefore, the goal constraint $\mathcal{I}(\mathbf{p}_m^\tau, \mathcal{G}_i)$ of Equ.(8) in this state is defined as:

$$\begin{aligned} \overrightarrow{\mathbf{p}_1^g \mathbf{p}_{n_i}^g} \times \overrightarrow{\mathbf{p}_1^g, \mathbf{p}_m^\tau} = 0, \\ 0 \leq \overrightarrow{\mathbf{p}_1^g \mathbf{p}_{n_i}^g} \cdot \overrightarrow{\mathbf{p}_1^g, \mathbf{p}_m^\tau} \leq |\overrightarrow{\mathbf{p}_1^g \mathbf{p}_{n_i}^g}|^2, \end{aligned} \tag{11}$$

where $\mathbf{p}_1^g$ and $\mathbf{p}_{n_i}^g$ denote the endpoints of a goal set $\mathcal{G}_i$, and the $\mathbf{p}_m^\tau$ is the last point on the trajectory $\tau$. The goal constraint in this state allow the point to $\mathbf{p}_m^\tau$ slides along the goal set during optimization so that a faster trajectory can be optimized and obtained.

The velocity constraint $\mathcal{V}(\mathbf{p}_i^\tau, \mathbf{p}_{i+1}^\tau, V^{max})$ of Equ.(8) considers only the robot's physical speed limit $V^{max}$ in this state. For each adjacent points, the constraint is defined as:

$$||\mathbf{p}_i^\tau - \mathbf{p}_{i+1}^\tau||_2 / \Delta T \leq V^{max}, \tag{12}$$

where $\Delta T$ is a fixed time interval. In this case, the robot can approach the goal as quickly as possible.

**Following State** (Fig. 3b and Fig. 3d) is activated when the leader is within the robot's FOV and also within the current costmap. The primary task is to follow the leader while avoiding unnecessary detours, as in the experiment in Fig. 9b. A Kalman filter is used to estimate the leader's state, where the Normalized Innovation Squared (NIS) [20] value serves as an indicator of prediction uncertainty. Based on this, a safe distance between the follower and the leader is dynamically adjusted as:

$$D_t = \text{Clip}\left(\alpha\text{NIS},\ D^{min},\ D^{max}\right), \tag{13}$$

where $D^{min}$ and $D^{max}$ are predefined lower and upper bounds, and $\alpha$ is a scaling factor. Instead of using goal line sets as in the *chasing* state, candidate goal points are sampled along a circle centered on the leader, with a radius equal to the safe distance $D_t$ in this state. These points are further segmented by obstacles into several sets of subgoal circles $\mathcal{G}_i$. In this state, the goal constraint $\mathcal{I}(\mathbf{p}_m^\tau, \theta_m^\tau, \mathcal{G}_i)$ at Equ.(8) allows the last trajectory point $\mathbf{p}_m^\tau$ to slide along the goal circle set $\mathcal{G}_i = \{\mathbf{p}_1^g, \mathbf{p}_2^g, ..., \mathbf{p}_{n_i}^g\}$, while constraining the orientation $\theta_m^\tau$ to face the leader, thereby reducing the risk of losing visual contact:

$$\begin{aligned} &\overrightarrow{\bar{\mathbf{p}}_l\ \mathbf{p}_1^g} \times \overrightarrow{\bar{\mathbf{p}}_l\ \mathbf{p}_m^\tau} \geq 0, \qquad \overrightarrow{\bar{\mathbf{p}}_l\ \mathbf{p}_m^\tau} \times \overrightarrow{\bar{\mathbf{p}}_l\ \mathbf{p}_{n_i}^g} \geq 0, \\ &|\overrightarrow{\bar{\mathbf{p}}_l\ \mathbf{p}_m^\tau}| = D_t, \qquad |\theta_m^\tau - atan(\bar{\mathbf{p}}_l - \mathbf{p}_m^\tau)| \leq \epsilon, \end{aligned} \tag{14}$$

where $\epsilon$ is a small threshold.

The maximum speed of the robot $V_t^{max}$ in this state is determined by the relative distance between the robot $\mathbf{p}_t^r$ and the leader $\bar{\mathbf{p}}_t^l$, as well as the estimated velocity of the leader $\bar{\mathbf{v}}_t^l$. The velocity constraint $\mathcal{V}(\mathbf{p}_i^\tau, \mathbf{p}_{i+1}^\tau, V^{max})$ of Equ.(8) reduces the robot's physical speed limit $V^{max}$ to $V_t^{max}$, avoiding frequent stop-and-go behavior of the robot. For each adjacent points, the constraint is defined as:

$$\begin{aligned} V_t^{max} = \text{Clip}\left(\alpha_1|\bar{\mathbf{v}}_t^l| + \alpha_2|\bar{\mathbf{p}}_t^l - \mathbf{p}_t^r|,\ 0,\ V^{max}\right), \\ ||\mathbf{p}_i^\tau - \mathbf{p}_{i+1}^\tau||_2 / \Delta T \leq V_t^{max}, \end{aligned} \tag{15}$$

where $\alpha_1$ and $\alpha_2$ are scaling factors. The function Clip$(\cdot)$ limits the value within the physical kinematics of the robot.

**Planning state** (Fig. 3c) is activated when the leader is no longer within the robot's field of view. This typically occurs when the robot is detouring around an obstacle or alternates with the *following* state when the leader is partially occluded. In this state, the goal constraint $\mathcal{I}(\mathbf{p}_m^\tau, \theta_m^\tau, \bar{\mathbf{p}}_t^l)$ of Equ.(8)

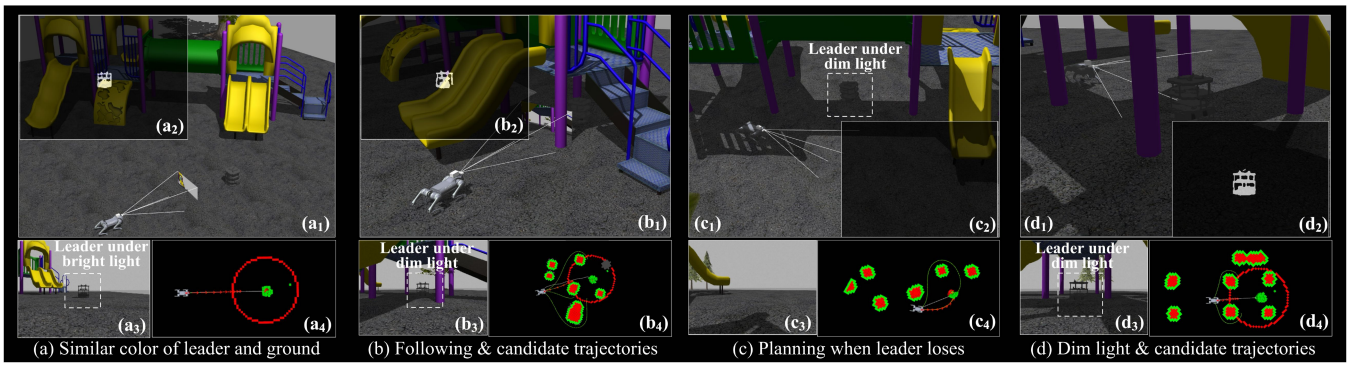



Fig. 4. Simulation for following a mobile robot in a playground. (a) The leader can be robustly segmented even when its color closely resembles the ground. (b) Multiple candidate trajectories are generated for the robot to choose from. (c) When the leader is lost, the planning state enables the follower to actively search for the leader instead of awkwardly stopping in place. (d) Even under dim lighting conditions, the leader can still be robustly segmented.

sets the trajectory endpoint $\mathbf{p}_m^\tau$ and $\theta_m^\tau$ to be the last known position $\bar{\mathbf{p}}_t^l$ and orientation $\bar{\theta}_t^l$ of the leader:

$$|\mathbf{p}_m^\tau - \bar{\mathbf{p}}_t^l| <= \epsilon, \qquad |\theta_m^\tau - \bar{\theta}_t^l| <= \epsilon. \tag{16}$$

The velocity constraint $\mathcal{V}(\mathbf{p}_i^\tau, \mathbf{p}_{i+1}^\tau, V^{max})$ of Equ.(8) obeys the same way as that of *chasing* state, so that the robot can quickly approach the leader to reduce the risk of losing the visual contact.

**Retreating State** (Fig. 3e) is activated when the leader actively approaches the robot. The robot is expected to move backward to avoid potential collisions with the leader. The velocity constraint $\mathcal{V}(\mathbf{p}_i^\tau, \mathbf{p}_{i+1}^\tau, V_t^{max})$ and goal constraints $\mathcal{I}(\mathbf{p}_m^\tau, \theta_m^\tau, \mathcal{G}_i)$ in this state are identical to those used in the *following* state. The one closest to the robot is selected as the initial goal, and the endpoint of the trajectory is allowed to slide along the corresponding goal set $\mathcal{G}_i$.

**Switching state** (Fig. 3f) is activated when a new leader is assigned, most commonly triggered by a large language model [21]. In this state, both the memory buffer and the distance frame buffer are reset based on the current input. The system then directly transitions to the appropriate subsequent state according to the relative relationship between the robot and the newly assigned leader.

## III. Simulation

The simulation evaluates our Follow Everything in a Gazebo with a legged robot follower ($V^{max} = 1.5, \mathrm{m/s}$), using a Realsense D435i for segmentation and a Mid360 for point clouds. Four scenarios are tested: a mobile robot as the leader in a playground, a UAV as the leader in a forest, a pedestrian as the leader in a factory, and a stop sign as the leader in a dynamic scenario. In each test scenario, 10 leader trajectories are recorded using ROS bag and replayed to ensure a fair comparison. Each trajectory is tested four times to minimize the effects of environmental noise, resulting in 40 tests per scenario and 160 tests in total. Four metrics [13] are considered: (1) the *follow success rate* (the percentage of runs where the robot maintains a safe distance and keeps the leader in view after the leader reaches its target), (2) the *average leader loss time ratio* (the ratio of time the leader is out of view), (3) the *collision rate* (the ratio of collisions over 160 tests), and (4) the *average distance* (the average robot–leader distance, where larger values risk visual loss). Our method, Follow Everything, is compared against four baselines: (1) Alaa [10], which segments the leader using SAM2 and plans trajectories with a PID controller; (2) SA-MPC [11], which detects the leader using YOLO11 and utilizes an enhanced MPC-based planner; (3) FE-N-DFB, an ablation of our framework that does not incorporate the distance frame buffer; and (4) FE-N-GP, another ablation of our framework that replaces the graph-based planner with the MPC controller of SA-MPC. Further details regarding setups and parameters are available at the website.

**Mobile robot leader in playground**. In this scenario, the leader is a mobile robot whose color closely resembles that of the ground, as shown in Fig. 4(a). A building is placed at the center of the scenario, where narrow gaps are just wide enough for both the leader and the follower to pass through.

This scenario presents three major challenges. The first challenge is whether the follower can robustly segment and track the leader under varying lighting conditions. As shown in Fig. 4(a), when the leader is outside the slide, the lighting is relatively bright. In contrast, inside the slide (Fig. 4b–d), the lighting becomes dim, which can significantly alter the visual features of the leader. This discrepancy often leads to detection or segmentation failures, even when the leader is directly in front of the follower. Such failures are particularly evident in methods that rely on YOLO11 for leader detection. In contrast, the SAM-based memory buffer in Follow Everything continuously updates the leader embedding over time, adapting to illumination changes and reducing the risk of tracking loss. This advantage is reflected in Table. I, where our method greatly outperforms SA-MPC regarding the average leader loss time ratio (10.7% vs 55.3%).

The second challenge is whether the robot can safely follow the leader. As shown in Fig. 4(b) and (d), the slide and pillars create several impassable regions. The robot is therefore unable to move directly toward the leader and must instead follow while simultaneously avoiding obstacles. Our graph-based planner addresses this by generating multiple candidate trajectories with distinctive meanings, from which the safest and fastest path can be selected.

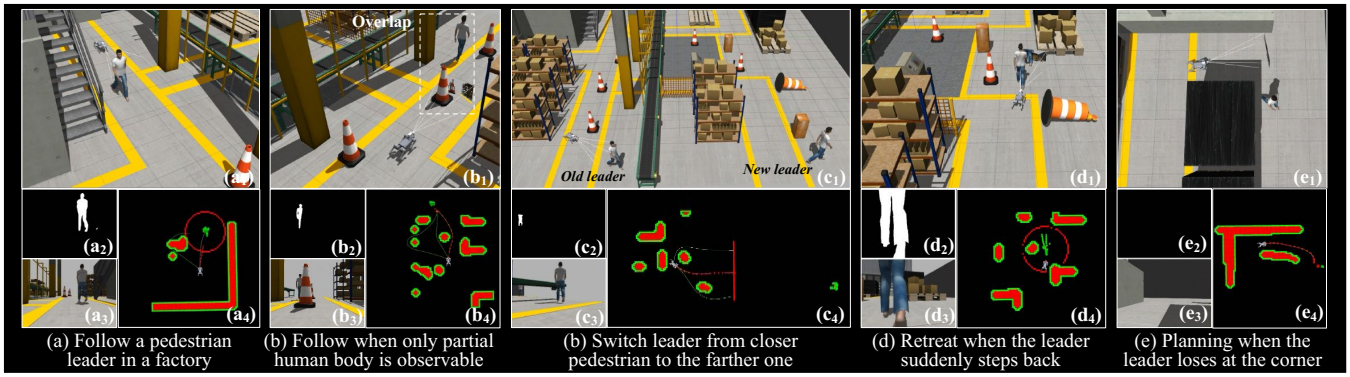



Fig. 5. Simulation for following a pedestrian in a factory. (a–b) The robot follows the leader when partially observed. (c) Upon receiving a prompt specifying a new leader, the chasing state is activated, and the robot passes underneath the workstation to approach the new leader. (d) When the leader steps backward, the robot retreats to maintain a safe distance. (e) When the leader turns a corner and leaves the robot's FOV, the planning state is triggered.

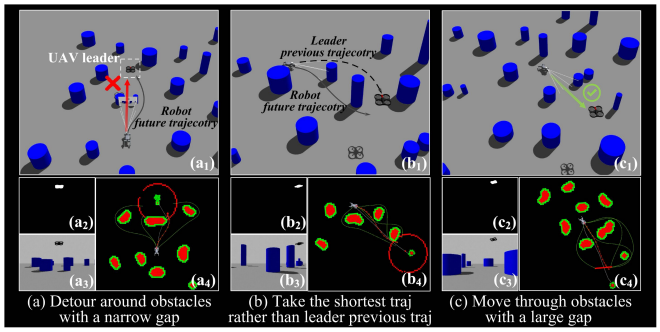



Fig. 6. Simulation for following a UAV in a forest. (a) After the UAV flies through a narrow gap, the robot detours for safety. (b) Rather than replicating the UAV's historical trajectory, the robot identifies a shorter one. (c) When the gap is wide enough, the robot passes through it for efficiency.

The third challenge involves re-identifying the leader after it disappears from the robot's FOV. As illustrated in Fig. 4(c), Our Follow Everything enables the robot to rapidly approach the leader's last known pose at maximum speed, thereby minimizing the duration of disconnection and achieving the highest follow success rate of 96.9%. In comparison, the ablated variant FE-N-GP does not employ a graph-based planner, nor does it consider the leader's final orientation or use a speed-maximizing strategy. As a result, its follow success rate drops to 81.3%. When it reaches the leader's last known pose, the leader has moved far away, causing the leader to be forever lost. In methods such as Alaa and FE-N-DFB, which rely solely on the temporal memory buffer, as the leader often becomes partially visible in the final frames prior to disappearance, these incomplete embeddings may mislead the model and prevent accurate matching with the current scene, resulting in re-identification success rates of only 21.8% and 62.5%, respectively. In contrast, Follow Everything leverages the distance frame buffer to mitigate this issue, enabling robust leader re-identification.

**UAV leader in forest**. In this scenario, multiple randomly generated cylindrical obstacles are placed to simulate trees in a forest, and the leader is a small UAV. The main challenge lies in balancing following the leader and avoiding obstacles.

When the UAV flies over two closely spaced obstacles, as shown in Fig. 6(a), the gap between them is too narrow for the follower to pass through. So our Follow Everything enables the robot to promptly detour around the obstacles. During this process, the UAV may temporarily disappear from the robot's FOV. While the goal-state adaptation transitions from the following state to the planning state, the graph-based planner ensures behavioral consistency by switching the robot's goal from a circle set to a point.

After the UAV detours around two obstacles, as shown in Fig. 6(b), the graph-based planner generates multiple feasible trajectories and selects the fastest one, rather than replicating the UAV's historical path. This strategy helps the robot maintain a close distance to the leader. As shown in Table. I and Fig. 7, our method achieves the lowest average distance of 2.0 meters and robustly maintains it with the smallest variance, which reduces the likelihood of the leader leaving the robot's FOV (too close can cause the UAV to leave the robot's view, too far makes it hard to recognize), thus improving the overall follow success rate.

When the gap between obstacles is sufficiently wide for the robot to pass through, as shown in Fig. 6(c), the robot directly moves through it in order to maximize efficiency.

**Pedestrian leader in factory**. In this scenario, the leader is a pedestrian, and the environment contains various obstacles with diverse shapes, sizes, and heights. As shown in Fig. 5(a), due to the relatively large size of the leader, it is frequently partially observed by the robot, as shown in (b). This partial observation, combined with frequent visibility loss during obstacle avoidance and the visual similarity between parts of the obstacles and parts of the pedestrian's body, pose a great challenge for the leader detection/segmentation.

Fig. 5(c) demonstrates the performance of the switching state. Upon receiving the prompt "follow the pedestrian on the left", the robot switches its target to a different pedestrian far away from it. Given the large distance between the robot and the new leader, the chasing state is activated. The robot moves at its maximum speed and passes underneath a workstation to rapidly approach the new leader.

Fig. 5(d) illustrates the retreating state, where the leader

TABLE I

COMPARISON OF OUR FOLLOW EVERYTHING TO BASELINES AND ABLATIONS. OUR METHOD ACHIEVES THE BEST PERFORMANCE IN ALL METRICS

| | Follow success rate [↑] | Average leader loss time ratio [↓] | Collision rate [↓] | Average distance [↓] |
|---|---|---|---|---|
| Alaa (baseline) [10] | 21.8% | 23.8% | 66.9% | 3.3 m |
| SA-MPC (baseline) [11] | 11.9% | 55.3% | 80.6% | 2.4 m |
| FE-N-DFB (ablation) | 62.5% | 44.0% | 25.0% | 3.1 m |
| FE-N-GP (ablation) | 81.3% | 20.0% | 16.3% | 2.6 m |
| Follow Everything (ours) | **96.9%** | **10.7%** | **1.8%** | **2.0 m** |

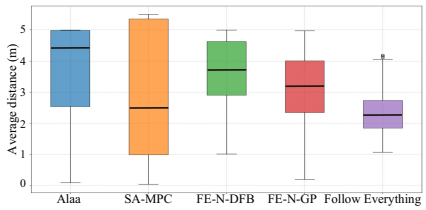



Fig. 7. Average robot-leader distance in 160 tests: Alaa 3.3 m, SA-MPC 2.4 m, FE-N-DFB 3.1m, FE-N-GP 2.6 m, and Follow Everything 2.0 m.

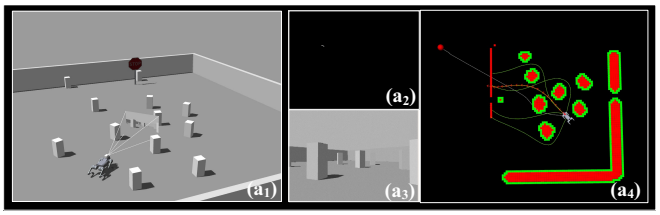



Fig. 8. Simulation for following a stop sign. The graph-based planner provides multiple candidate trajectories so that a safer one can be identified.

actively steps backward. The robot retreats accordingly while avoiding nearby construction cones, ensuring safety. As a result, Follow Everything achieves a collision rate of only 1.8%, which is substantially lower than that of other methods. A key perception challenge here is that when the leader is close to the robot, only a portion of the leader's body is visible. The YOLO-based method struggles to recognize such partial views as the leader, resulting in a high collision rate of 80.6%. Moreover, when the leader starts moving forward again, SAM2 without a distance frame buffer often fails to re-identify the leader, leading to a collision rate of 66.9% in Alaa. and 25.0% in FE-N-DFB. Fig. 5(e) shows how the planning state contributes to maintaining robust following behavior after the leader turns a corner and temporarily disappears from the robot's FOV.

**Stop sign leader in dynamic scenario**. As shown in Fig.8, in this scenario, the leader is a stop sign, and dynamic obstacles move randomly, changing their velocities every 3 seconds. The graph-based planner enables the generation of multiple candidate trajectories, ensuring that a safer trajectory is not only available but also selected. This capability allows Follow Everything to significantly outperform other methods, achieving the lowest collision rate of 1.8%.

## IV. REAL-WORLD EXPERIMENT

In the experiment, the same legged robot is used as the follower, with a person or a wheel-legged robot as the leader. The experiment is conducted in both indoor and outdoor environments. A laptop with Intel i7 and Nvidia RTX 3070 is mounted on the legged robot as the onboard computer.

**Following state in indoor scenario**. As shown in Fig. 9(a), a suitcase is placed in the scene center, and the robot first follows the pedestrian, maintaining a relatively stable safe distance. As shown in Fig. 9(b), during the pedestrian's clockwise detour around the obstacle, the robot continuously stays in the following state, adjusting its orientation to always face the leader while maintaining a safe distance, thus avoiding unnecessary detours. As shown in Fig. 9(c), once the pedestrian moves towards a further location, the robot detours counterclockwise around the obstacle and continues to follow the pedestrian. As shown in Fig. 9(d-e), when the pedestrian leader is replaced by a legged robot with wheels, the robot's following behavior remains stable.

**Planning state in indoor scenario**. As shown in Fig. 9(f-g), the robot initially follows the pedestrian's movement. Once the person exits the room, the planning state is activated, and the robot moves to the last known position of the leader and successfully re-identifies the leader.

**Following state in outdoor scenario**. As shown in Fig. 9(h-i), a wheeled legged robot leader moves through obstacles and then turns right. By maintaining a safe distance from the leader, the robot avoids replicating the leader's path and instead turns right to quickly approach the leader.

**Planning state in outdoor scenario**. As shown in Fig. 9(j-k), the leader is a pedestrian, moving in front of the robot and then stepping over a box. To avoid collision, the following state enables the robot to turn right and detour around the obstacles. During this process, the pedestrian temporarily leaves the robot's FOV, and the planning state ensures the robot continues detouring until the leader reappears in the robot's FOV, triggering the following state again.

## V. CONCLUSION

This paper proposes a unified framework for robust leader-following, which integrates a segmentation model with a distance frame buffer to enable tracking of arbitrary leaders, along with a goal-aware adaptation module and a graph-based planner for dynamic parameter adjustment, goal state decision-making and trajectory generation. Simulations and indoor/outdoor experiments with a legged robot follower and diverse leaders verify that the framework significantly boosts following success rates, shortens leader visual loss duration, lowers collision rates, and reduces the average follower-leader distance.

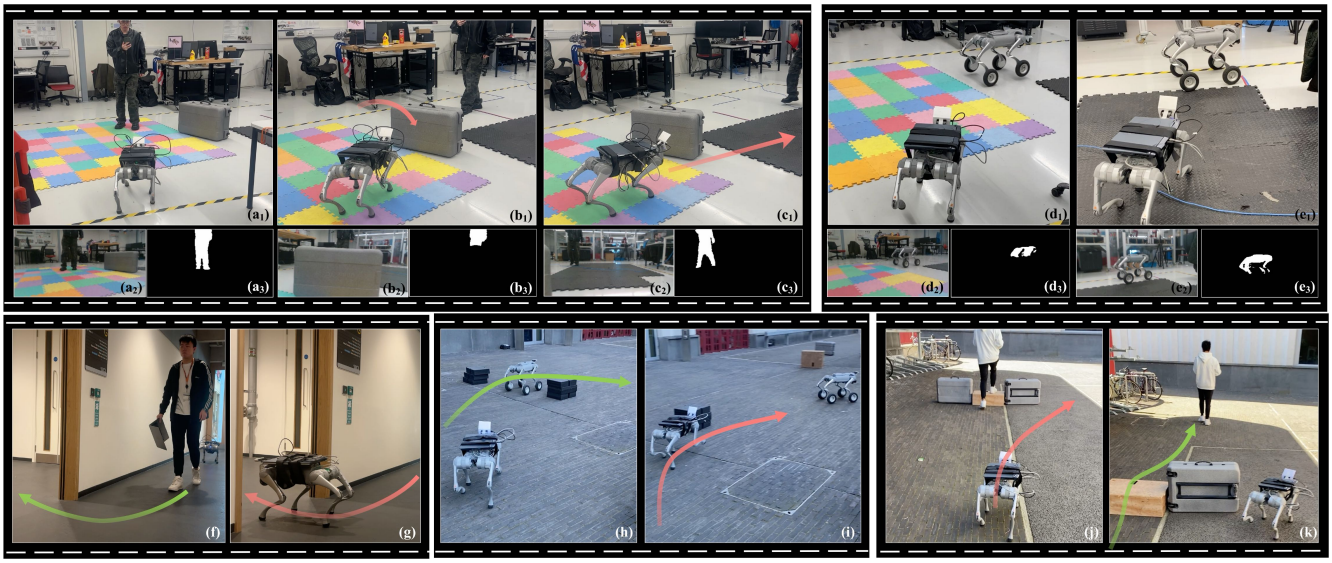



Fig. 9. Illustration of real-world experiments. (a-c) While following a pedestrian, an appropriate safe distance allows the robot to rotate in place and avoid unnecessary detours. (d-e) The robot leader is reliably identified and followed. (f-g) The planning state allows the robot to follow and re-identify the leader after moving out of the room. (h-i) The robot takes a shorter path to quickly approach the leader. (j-k) The alternation between following and planning states enables the robot to follow the leader while avoiding obstacles when the leader steps over a box.

## REFERENCES


[1] L. Roy, E. A. Croft, A. Ramirez, and D. Kulić, "Gpt-driven gestures: Leveraging large language models to generate expressive robot motion for enhanced human-robot interaction," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 4172–4179, 2025.

[2] M. Stamatopoulou, J. Liu, and D. Kanoulas, "Dippest: Diffusion-based path planner for synergistic trajectory generation applied on quadrupedal robots," in *IEEE/RSJ IROS*, 2024.

[3] J. Liu, M. Stamatopoulou, and D. Kanoulas, "Dipper: Diffusion-based 2d path planner applied on legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9264–9270.

[4] L. Yao, V. Modugno, A. M. Delfaki, Y. Liu, D. Stoyanov, and D. Kanoulas, "Local path planning among pushable objects based on reinforcement learning," 2024.

[5] J. Wang, L. Bao, T. Yang, D. M. Plasencia, J. Jiao, and D. Kanoulas, "Sand-planner: Sample-efficient diffusion planner in b-spline space for robust local navigation," 2026.

[6] S. Wang, J. Zhang, M. Li, J. Liu, A. Li, K. Wu, F. Zhong, J. Yu, Z. Zhang, and H. Wang, "Trackvla: Embodied visual tracking in the wild," *arXiv preprint arXiv:2505.23189*, 2025.

[7] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine learning and knowledge extraction*, vol. 5, no. 4, pp. 1680–1716, 2023.

[8] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[9] Z. Meng, X. Xia, R. Xu, W. Liu, and J. Ma, "Hybrid object detection and tracking for cooperative perception using 3d lidar," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4069–4080, 2023.

[10] 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.

[11] Y. Song, Q. Zhang, Z. Hu, and J. Liu, "Safe and robust human following for mobile robots based on self-avoidance mpc in crowded corridor scenarios," in *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2023, pp. 1–6.

[12] Q. Zhang, Z. Hu, Y. Song, J. Pei, and J. Liu, "The human gaze helps robots run bravely and efficiently in crowds," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 7540–7546.

[13] Z. Zhang, J. Yan, X. Kong, G. Zhai, and Y. Liu, "Efficient motion planning based on kinodynamic model for quadruped robots following persons in confined spaces," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 4, pp. 1997–2006, 2021.

[14] J. Liu, M. Stamatopoulou, and D. Kanoulas, "Diffusion-based 2d path planner applied on legged robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 9264–9270.

[15] R. Han, S. Wang, S. Wang, Z. Zhang, J. Chen, S. Lin, C. Li, C. Xu, Y. C. Eldar, Q. Hao *et al.*, "Direct point robot navigation with end-to-end model-based learning," *IEEE Transactions on Robotics*, 2025.

[16] Z. Li, B. Li, Q. Liang, W. Liu, L. Hou, and X. Rong, "A quadruped robot obstacle avoidance and personnel following strategy based on ultra-wideband and three-dimensional laser radar," *International Journal of Advanced Robotic Systems*, vol. 19, no. 4, p. 17298806221114705, 2022.

[17] V. Reijgwart, C. Cadena, R. Siegwart, and L. Ott, "Efficient volumetric mapping of multi-scale environments using wavelet-based compression," *arXiv preprint arXiv:2306.01279*, 2023.

[18] C. Scheidemann, L. Werner, V. Reijgwart, A. Cramariuc, J. Chomarat, J.-R. Chiu, and M. Hutter, "Obstacle-avoidant leader following with a quadruped robot," *arXiv preprint arXiv:2410.00572*, 2024.

[19] A. Tripathi, M. A. Khan, A. Pandey, P. Yadav, and A. K. Sharma, "Human following robot using ultrasonic sensor," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. IEEE, 2021, pp. 764–770.

[20] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, 2015, pp. 74–77.

[21] Y. Zhang, T. Cheng, R. Hu, L. Liu, H. Liu, L. Ran, X. Chen, W. Liu, and X. Wang, "Early vision-language fusion for text-prompted segment anything model," *arXiv preprint arXiv:2406.20076*, 2024.

[22] Z. Zhu, Q. Zhang, Y. Song, Y. Yang, and J. Liu, "Stc-teb: Spatial-temporally complete trajectory generation based on incremental optimization," *IEEE Robotics and Automation Letters*, 2024.

[23] Q. Zhang, W. Luo, Z. Zhang, Y. Wang, and J. Liu, "Ga-teb: Goal-adaptive framework for efficient navigation based on goal lines," 2024.

[24] O. de Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-driven parallel trajectory optimization in dynamic environments," *IEEE Transactions on Robotics*, vol. 41, pp. 110–126, 2025.

[25] S. Bhattacharya, "Search-based path planning with homotopy class constraints," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 24, no. 1, 2010, pp. 1230–1237.

[26] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.