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

An End-to-End Trajectory Planner for Safe and Efficient Navigation in Crowded Dynamic Environments

Shuting Zhang[1], Haowen Wang[1], and Guangchen Li[1]

Abstract— **This paper presents a novel end-to-end trajectory planning framework that integrates LiDAR-based perception with trajectory optimization, enabling safe and efficient navigation in dynamic environments without relying on semantic detection or explicit kinematic modeling. Learning-based dynamic collision avoidance methods often depend on reinforcement learning, which introduces challenges related to training efficiency, model generalization, and deployment safety. To address these limitations, we propose a lightweight map representation for temporally continuous dynamic obstacles, facilitating unsupervised network training with physically simulated data. Additionally, a repulsion-based adjustment method built upon motion primitives allows adaptive trajectory planning in highly crowded scenarios where no feasible trajectory exists, balancing target-reaching objectives with motion safety. Extensive simulations and real-world experiments demonstrate that the proposed framework achieves millisecond-level planning latency while ensuring high safety, trajectory smoothness, and flight efficiency. The demonstration video is available on the project website: https://swift520.github.io/Dynamic-Planner/.**

I. Introduction

The rapid advancements in autonomous systems and robotics have driven the development of aerial vehicle technologies, particularly Unmanned Aerial Vehicles (UAVs), increasing the demand for efficient motion planning algorithms capable of navigating complex and dynamic environments.

Traditional planning methods typically adopt a hierarchical structure, separating perception and trajectory optimization into distinct modules [1]–[4]. While effective in static environments, their performance degrades in real-world scenarios involving unknown obstacle movements. To address this, some researchers have introduced modeling of dynamic obstacles under static environment assumptions [5]–[9]. However, these approaches often rely on simplified motion models, such as constant velocity or acceleration, which struggle to handle cluttered environments with multiple unpredictable dynamic obstacles, thereby limiting their practical utility.

Learning-based planning methods [10]–[15] offer a promising alternative with lower latency and faster responses compared to optimization-based approaches. Networks enable novel processing of dynamic obstacles, as seen in studies such as [16]–[20], which integrate object recognition or semantic segmentation to infer the motion attributes of obstacles within the field of view (FOV). However, these methods are constrained by the detection accuracy of the underlying networks and are typically limited to identifying common movable objects. Moreover, their high computational demands make deployment on resource-constrained platforms challenging. Reinforcement learning (RL)-based methods eliminate the reliance on expert trajectories and are widely used in modern dynamic planning networks [17]–[25]. Despite the flexibility of RL methods, inherent training challenges and safety concerns during deployment hinder their ability to meet practical navigation requirements.

In this paper, we present an end-to-end planning framework that avoids map reconstruction, dynamic obstacle preprocessing, and iterative optimization. It enables exploration of multiple trajectories by partitioning motion primitives and employing a flexible steering strategy, effectively mitigating stagnation when no immediately feasible path exists ahead. A lightweight representation of the time-varying dynamic map evaluates trajectory safety from discretized samples and, together with smoothness and target guidance, defines a unified physics-based loss for network training.

The main contributions of this work are:

- We propose an end-to-end trajectory planner and design a lightweight map representation for the spatial distribution of dynamic obstacles, enabling the network to be trained based on physical loss functions.
- To obtain safe trajectories in highly crowded environments, we propose a repulsion-based direction adjustment method to flexibly rotate the planning region.
- Simulations and experiments demonstrate that our method outperforms existing baselines in terms of success rate, trajectory quality, and flight efficiency across diverse dynamic scenarios.

II. Related Work

A. Learning-Based Trajectory Planning

Learning-based trajectory planning methods have gained popularity by leveraging large datasets to learn complex mappings from inputs to outputs. Imitation learning [10]–[12] uses low-cost deep learning architectures to model intricate relationships among environments, states, and trajectories, replacing computationally expensive expert strategies. Deep-PANTHER [13] presents a multi-modal learning-based framework for imitating multiple locally optimal expert trajectories. LPNet [14] employs an improved Q-network to select optimal primitives, enabling rapid local decision-making. However, their reliance on expert-generated data limits both the upper bound of performance and the generalization capability to new environments.

RL-based methods leverage extensive interactions with simulated environments to acquire richer data. For instance,

[1]School of Electronics, Peking University, Beijing 100871, China. zhangshuting@stu.pku.edu.cn, wanghaowen19@stu.pku.edu.cn, liguangchen@pku.edu.cn

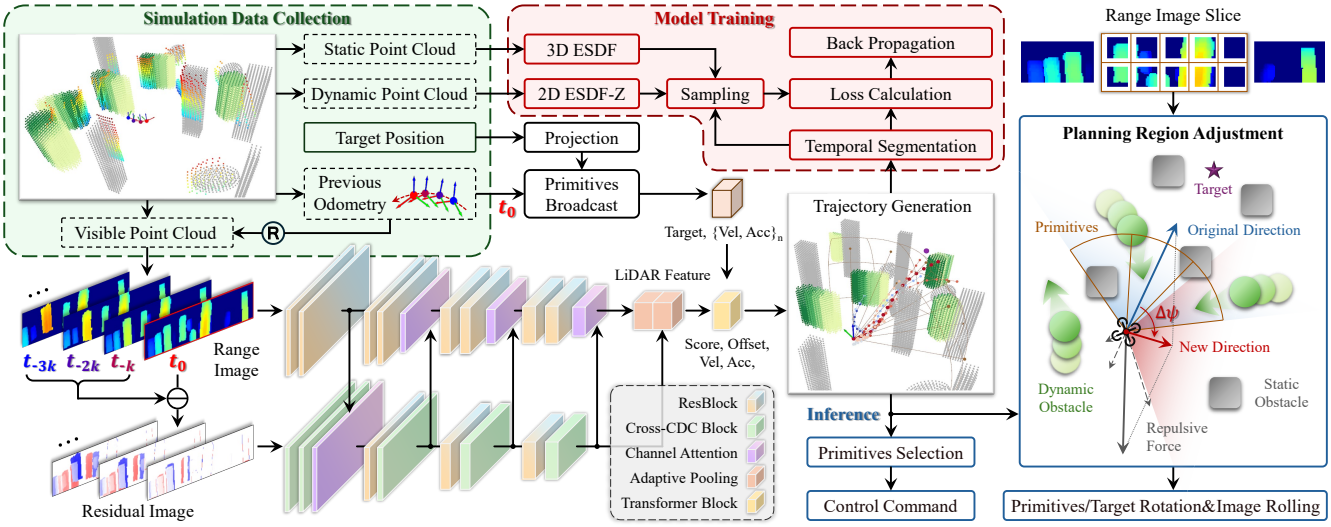


Fig. 1: Overview of the Trajectory Planner Architecture. The planner takes historical point clouds, odometry, and the target position as input, refines the trajectory of each motion primitive, selects one based on predicted scores, and generates motion control commands. During training, the ESDF map constructed from simulated static and dynamic point clouds provides collision losses and gradients; during testing, the predicted trajectories and slices of the current frame's range image are used to adapt the planning region.

DRL-VO [18] integrates LiDAR and pedestrian kinematic data, using a reward function based on velocity obstacles. [21] employs adaptive contrastive learning to enhance image encoding, while NavRL [20] defines a dual-branch network structure consisting of static and dynamic components, improving safety through the concept of velocity obstacles. However, these methods often suffer from prolonged training cycles, unstable convergence, and complex reward design, which limit their robustness and scalability in dynamic environments with dense obstacles and complex motion patterns.

YOPO [15] proposes a guided learning paradigm that frames planning as a trajectory adjustment task and directly evaluates the safety of each trajectory using obstacle distribution information. This approach addresses the limitations of imitation learning, offering improved efficiency and stability compared to RL-based methods. However, its reliance on a limited FOV from depth sensors and its focus on static physical information constrain its ability to comprehensively model dynamic environments. In densely crowded scenarios, this limitation often leads to infeasible paths, increasing the likelihood of trajectory failures and collisions.

B. Dynamic Obstacle Avoidance

Traditional motion planning methods often treat the detection and modeling of dynamic obstacles as separate processes, which are typically time-consuming. Vision-based approaches [5]–[7] detect obstacles in images, while point-cloud-based methods [8], [9] operate on clusters; trajectories are then obtained by solving optimization problems conditioned on simplified motion models, which introduces inherent biases due to simplified kinematic assumptions. Another line of work discretizes the environment into grids or voxels and updates cells affected by dynamic obstacles online to enable fast distance queries [26], [27], but their purely reactive nature leads to delayed avoidance decisions and poor robustness in dense and fast-moving scenarios.

Recent advancements in deep learning have enabled the development of more sophisticated techniques for dynamic obstacle avoidance. [28] employs a variational autoencoder to enhance the network's understanding of scene semantics. [17], DRL-VO [18], [19], and NavRL [20], integrate object detection or point cloud segmentation algorithms to isolate dynamic obstacles. These methods demonstrate significant advantages in handling common movable obstacles, such as humans. However, they are often limited by poor scalability and high computational demands, which impose stringent hardware requirements and necessitate trade-offs between real-time performance and accuracy. To address these limitations, some studies have introduced custom feature designs to assist network learning. [23] proposes a method for grouping and clustering 2D LiDAR points to determine motion attributes, while [24] introduces two quantitative indicators to characterize environmental complexity based on LiDAR scans. [25] encodes 3D point cloud data into a 2D obstacle map, effectively capturing both static and dynamic obstacle contours and motion information. However, most of these methods focus solely on 2D feature extraction, leading to the loss of z-axis information and making them more suitable for ground robots operating on flat surfaces.

III. Method

The training and inference pipeline of the proposed trajectory planner, which consists of a neural network-based trajectory prediction module and a repulsion-based direction adjustment module, is depicted in Fig. 1.

A. Point Cloud Input Representation

The range image representation of LiDAR projects scanning data into a 2D image format, typically using horizontal and vertical angles as the coordinate axes, with pixel values representing distance. This representation effectively compresses 3D point clouds while preserving spatial information,

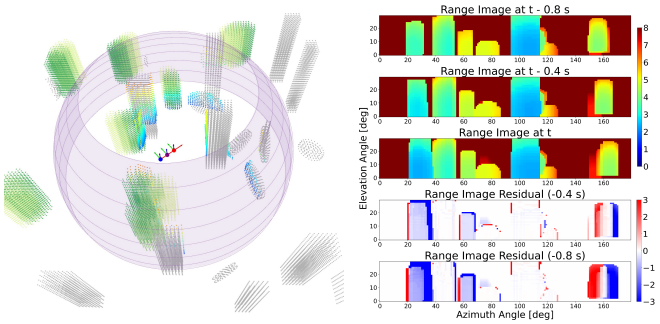


Fig. 2: Range Images and Residuals. The gray and green point clouds represent static and dynamic obstacles, respectively. The colored point cloud, along with the red coordinate system, corresponds to the simulated LiDAR scanning data under the current pose. The purple and blue coordinate systems represent sequential shifts forward by $0.4\,s$. The range images are generated by unfolding the scanning data clockwise on the xy-plane, starting from the backward orientation, with a resolution of 2° per pixel.

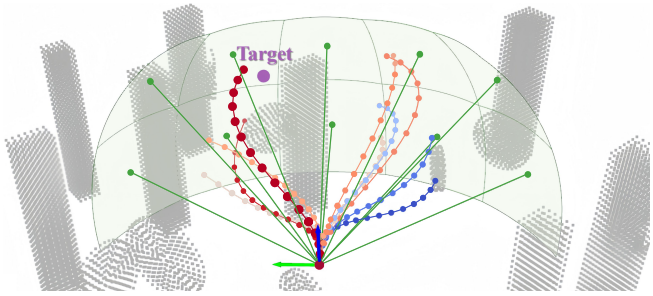


Fig. 3: Partitioning and Refinement of Motion Primitives. The coordinate system represents the current pose, while the green frame delineates the adjustable range of each primitive. The trajectory planner regulates the flight trajectory by modifying the terminal state of each primitive. The color gradient from blue to red indicates increasing predicted scores, with the bright red, bold trajectory representing the highest score, selected for execution.

characterized by fixed resolution and pixel arrangement. Such a lightweight representation facilitates the adaptation of point clouds to well-established and optimized 2D convolutional neural networks, thereby eliminating the computational overhead associated with data preprocessing and meeting the real-time requirements of trajectory planning tasks.

To capture temporal information from point cloud sequences and accurately identify dynamic obstacles, a common approach involves generating residual images from consecutive scans [29]. Specifically, the previous frame's scan is transformed into the current coordinate system using relative poses, reprojected into a range image, and then the absolute pixel-wise difference is computed with the range image of the current frame. An example is illustrated in Fig. 2. While this method effectively identifies the motion states of point clouds, trajectory planning tasks require a deeper understanding of the motion trends of obstacles, including their direction and velocity. Therefore, we use a single range image and stack multiple residuals, generated at fixed time intervals in the past, normalized to the ranges $[0, 1]$ and $[-1, 1]$, respectively, as inputs to the planning network.

B. Primitive Partitioning and Trajectory Representation

Since the primary objective of trajectory planning is to guide the UAV to a specified target position, the forward LiDAR scanning region is defined as the effective flight space. As shown in Fig. 3, this space is uniformly partitioned into quadrangular pyramids, with the UAV's current position serving as the apex. The initialized center point within each pyramid is represented in spherical coordinates as:

$$\mathbf{p}_j^c = (r\cos\theta_j\cos\varphi_j,\ r\cos\theta_j\sin\varphi_j,\ r\sin\theta_j) \qquad (1)$$

where r denotes the fixed planning length, corresponding to the effective detection range of the LiDAR (set to $8\,m$ in simulation). θ_j and φ_j represent the elevation and azimuth angles of the center of the j-th primitive, respectively.

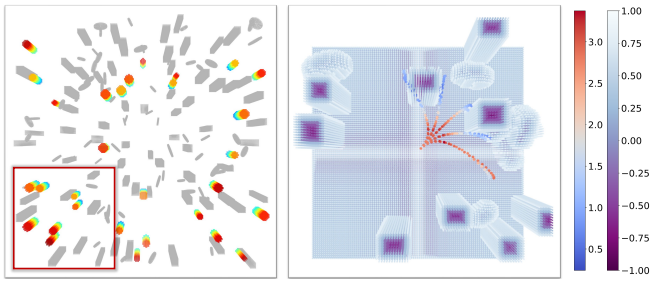
The planner can project the terminal position of a trajectory to any point within the corresponding pyramidal region by adjusting the planning length and introducing offsets to the two angular coordinates. Based on this partitioning, the planner further predicts velocity and acceleration for each motion primitive to fully define the trajectory constraints, which are represented as $[\mathbf{p}^i, \mathbf{v}^i, \mathbf{a}^i, \mathbf{p}^e, \mathbf{v}^e, \mathbf{a}^e]$. These constraints specify the position, velocity, and acceleration at both the initial and terminal states of the trajectory. Following the optimizer-based trajectory planning approach [9], the trajectory is formulated as a polynomial. Specifically, a unique 5th-degree polynomial trajectory in 3D space is determined by the 18 constraint parameters described above.

C. Planning Network Architecture

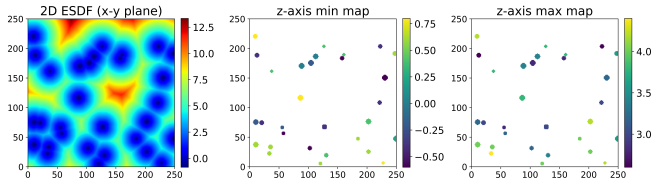
1) LiDAR Feature Encoding: The planning network adopts a dual-branch architecture to effectively process spatial and temporal features, as illustrated in Fig. 1. The range branch uses stacked residual blocks to process the current frame's range image and capture spatial occupancy information. To enhance sensitivity to pixel-level changes, certain convolutional layers in the residual branch are replaced with cross central difference convolutions [30]. This modification improves the network's ability to capture fine-grained motion details while maintaining computational efficiency.

To integrate spatial and temporal embeddings across multiple scales, the planner incorporates squeeze-and-excitation blocks [31] after every pair of residual modules. Given the inherent sparsity of residual images—particularly in scenarios without moving obstacles within the FOV—the initial fusion step in the residual branch is specifically designed to ensure the production of valid data and stable gradients. In the final layer, features from both branches are resized using an adaptive pooling layer to match the resolution of the primitive lattices. This process ensures that the channel-wise expanded descriptors effectively represent the LiDAR features extracted by the backbone, which are subsequently utilized to predict the corresponding primitive trajectory.

2) Trajectory Refinement: The projection of the example primitives in Fig. 3 onto the range image is horizontally centered and spans only a limited lateral interval. This anchoring strategy eliminates translation invariance in LiDAR feature generation, ensuring features consistently correspond



(a) Example Environment and 3D ESDF Map for Static Obstacles.



(b) 2D ESDF-Z Map for Dynamic Obstacles.

Fig. 4: Obstacle Collision Cost Sampling. The example map measures $50\,m \times 50\,m$ with an ESDF resolution of $0.2\,m$. The gray and colored point clouds represent static and dynamic obstacles, respectively. (a) The 3D ESDF shows the distance from each point to the nearest static obstacle. Data within the range $[-1, 1]$ are visualized using the *BuPu* color scheme, while sampled values along primitive trajectories are represented with a *coolwarm* gradient. (b) The 2D ESDF projects dynamic obstacles onto the xy-plane, calculates the distance to the nearest dynamic obstacle, and visualizes the results using the *jet* color scheme. The *viridis*-colored points indicate the z-axis bounds of obstacles at corresponding plane coordinates.

to specific spatial directions relative to the UAV's orientation.

We define the terminal constraints as queries and use a lightweight transformer block as the prediction head. Specifically, assuming m primitives, the query $\mathbf{q} \in \mathbb{R}^{m\times(3+6n)}$ consists of the target position and the motion states from the past n frames, both transformed into the corresponding primitive coordinate systems and normalized. The cross-attention mechanism derives keys and values from LiDAR features, unfolded in the predefined primitive order to maintain spatial consistency. The second linear layer of the feed-forward network predicts the score, position offset, velocity, and acceleration for each primitive. To ensure motion feasibility, the hyperbolic tangent activation function is employed to decode the predicted trajectory constraints.

Finally, the terminal position is calculated based on the predefined primitive pyramids, with the velocity and acceleration scaled by their respective maximum values and transformed into the body coordinate system. By retrieving the trajectory coefficients from the initial and terminal constraints, the adjusted motion trajectory is generated.

3) Dynamic Obstacle Collision Cost and Gradient: YOPO [15] constructs a global Euclidean Signed Distance Field (ESDF) for numerical sampling and gradient querying to guide the network's learning. Inspired by this, we aim to identify a comparable physical cost representation for dynamic obstacles to support trajectory generation in dynamic environments. However, pre-constructing a high-resolution 3D ESDF for dynamically changing environments imposes significant memory demands, and performing synchronous computations can severely impact training efficiency.

To address these challenges, we propose dividing the map into two distinct components: the static map, which retains the original 3D ESDF construction as illustrated in Fig. 4a, and the dynamic obstacles, represented using a lightweight sparse map. Assuming dynamic obstacles are convex and do not overlap vertically, their spatial distributions can be uniquely determined by their projections onto the xy-plane and their z-axis bounds, as depicted in Fig. 4b.

To compute the collision cost of dynamic obstacles at a given point $\mathbf{p} = (x, y, z)$ along the trajectory, if the nearest obstacle is at the same z-height (including cases where $\mathbf{p}$ lies within the obstacle), the 2D ESDF value and gradient are directly retrieved, with the z-gradient set to zero. Otherwise, the cost and gradient are calculated as:

$$c^d(\mathbf{p}) = \sqrt{\mathbf{D}(x,y)^2 + |d_z(\mathbf{p}')|^2} \tag{2}$$

$$\nabla c^d(\mathbf{p}) = \frac{1}{c^d(\mathbf{p})}\left(d_{xy}(\mathbf{p}')\nabla\mathbf{D}(x,y), d_z(\mathbf{p}')\right) \tag{3}$$

where $\mathbf{D}$ denotes the 2D ESDF value matrix. The point $\mathbf{p}'$ represents the closest point in the z-direction among those nearest to $\mathbf{p}$ on the projection plane, defined as:

$$\mathbf{p}' = \text{argmin}_{\mathbf{p}'}\left\{|d_z(\mathbf{p}')|^2 \mid |d_{xy}(\mathbf{p}') - \mathbf{D}(x,y)| < \epsilon\right\} \tag{4}$$

where ϵ is a small positive threshold. This sampling strategy prioritizes azimuth adjustments over altitude changes during evasive maneuvers, aligning with the characteristics of LiDAR data, which feature a wide horizontal FOV and a higher horizontal scanning resolution compared to the vertical direction, thereby enhancing safety. $d_{xy}(\cdot)$ defines the horizontal distance from any point on the map to $\mathbf{p}$, and the vertical distance vector $d_z(\cdot)$ is calculated as:

$$d_z(\mathbf{p}') = \begin{cases} z - \mathbf{Z}_{\text{min}}(\mathbf{p}'), & z < \mathbf{Z}_{\text{min}}(\mathbf{p}') \\ z - \mathbf{Z}_{\text{max}}(\mathbf{p}'), & z > \mathbf{Z}_{\text{max}}(\mathbf{p}') \end{cases} \tag{5}$$

where $\mathbf{Z}_{\text{min}}$ and $\mathbf{Z}_{\text{max}}$ represent the matrices of the lowest and highest obstacle points, respectively.

4) Training Loss: To guide the planning network in generating smooth, target-oriented, and safe motion trajectories, the loss function is defined as:

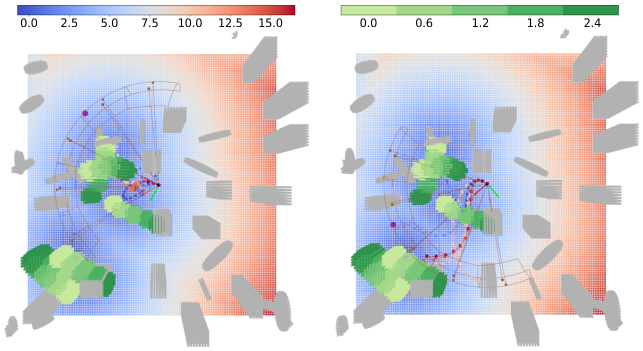
$$\mathcal{L}_{\text{traj}} = \lambda_s\mathcal{L}_s + \lambda_t\mathcal{L}_t + \lambda_c(\mathcal{L}_c^s + \mathcal{L}_c^d) \tag{6}$$

where $\mathcal{L}_s$ is the smoothness loss, $\mathcal{L}_t$ is the target attraction loss, and $\mathcal{L}_c^s$ and $\mathcal{L}_c^d$ represent the static and dynamic obstacle collision losses, respectively. $\lambda_s, \lambda_t, \lambda_c$ control the relative weighting of each term in the overall loss function.

The smoothness loss penalizes high-order derivatives of the generated trajectory to enforce fluid and continuous motion. Over the planning horizon T, we adopt the minimum-snap formulation [32] to constrain the fourth-order derivative of position with respect to time:

$$\mathcal{L}_s = \int_0^T \left\| \frac{d^4\mathbf{p}(t)}{dt^4} \right\|^2 dt \tag{7}$$

To enhance the learnability of the trajectory, the target is randomly assigned to any angle within the visible planning



(a) Planning at the current moment. (b) Planning at the next moment.

Fig. 5: Rotation of Planning Region. The movements of dynamic obstacles within $2.4\,s$ from the current moment are visualized using the *YlGn* color scheme. A pre-constructed 2D ESDF map for dynamic obstacles at the current moment is displayed on the virtual floor, rendered with a *coolwarm* color gradient. The obstacle group on the right exerts a stronger repulsive force, causing the motion primitives and the preset target to rotate counterclockwise.

region, with its distance sampled from the interval $[1, 1.25r]$, following an asymmetric exponential distribution with r as the peak value. The target attraction loss is defined as:

$$\mathcal{L}_t = \alpha \left\| \Delta\mathbf{p} \times \mathbf{g} \right\|^2 + (1-\alpha)\left(\|\Delta\mathbf{p}\| - \|\mathbf{g}\| \right)^2 \tag{8}$$

where $\Delta\mathbf{p} = \mathbf{p}^e - \mathbf{p}^i$, $\mathbf{g} = \mathbf{g}^{\text{global}} - \mathbf{p}^i$ represents the target position in the body coordinate system, and α is a weighting factor balancing the two terms. The first term encourages alignment of the trajectory with the target direction, while the second term ensures the trajectory length approximates the target distance. This formulation ensures that the trajectory remains target-oriented within the planning region.

Safety is enforced through collision constraints for both static and dynamic obstacles. The collision loss is derived from ESDF values using an exponential function:

$$\mathcal{L}_c = \int_0^T \exp\left(-\frac{c(\mathbf{p}(t)) - c_0}{\gamma} \right) dt \tag{9}$$

where c_0 and γ are used to adjust the penalty range and sensitivity. The function $c(\cdot)$ directly returns the 3D ESDF value in the static obstacle collision loss and is calculated according to Eq. 2 for dynamic obstacles. In practice, the integral losses are calculated by summing sampled values at discrete time intervals along the trajectory.

A motion-feasible, target-oriented, and safe trajectory corresponds to a lower loss value. Consequently, the trajectory score is supervised using the smooth L1 loss applied to the trajectory loss, and the optimal motion primitive is selected based on the minimum score during execution.

D. Adjustment of Planning Region

Under normal conditions, the planner prioritizes motion primitives oriented toward the target, typically aligning with the direction of movement. However, when the UAV encounters clusters of highly crowded obstacles, as shown in Fig. 5a, the planner may fail to identify an unoccupied space within its current planning FOV, leading to distorted and unsafe trajectories. By leveraging the omnidirectional horizontal FOV provided by the LiDAR, the planning region can be adjusted through the rotation of primitives and target position, as well as by horizontally rolling the range images of historical frames. This adjustment allows exploration of alternative spaces based on the obstacle distribution within the current FOV and the compressibility of trajectories from the end of each primitive toward the UAV's current position. The repulsive potential of obstacles is defined as:

$$\mathbf{U}_j^o = \frac{1}{2W_j H_j} \left(\frac{1}{\mathbf{R}_j^{\min}} - \frac{1}{r} \right)^2 \cdot \sum_{u=1}^{W_j} \sum_{v=1}^{H_j} \mathbb{I}_{\{\mathbf{R}_j(u,v) < r\}} \cdot \hat{\mathbf{e}}_j \tag{10}$$

where $\mathbf{R}_j \in \mathbb{R}^{W_j \times H_j}$ represents a slice of the current frame's raw range image corresponding to the j-th primitive, and the superscript $(\cdot)^{\min}$ denotes the minimum value. The indicator function $\mathbb{I}_{\{\cdot\}}$ equals 1 if the condition inside the braces is satisfied, and 0 otherwise. $\hat{\mathbf{e}}_j$ is the unit vector pointing from the initialized primitive center $\mathbf{p}_j^c$ defined in Eq. 1 (the green dots in the Fig. 3) to the UAV's current position $\mathbf{p}^i$. The repulsive potential of planning compressibility is defined as:

$$\mathbf{U}_j^p = \begin{cases} \frac{1}{2}\left(\frac{1}{\|\Delta\mathbf{p}_j\|} - \frac{1}{r} \right)^2 S_j \cdot \hat{\mathbf{e}}_j, & \|\Delta\mathbf{p}_j\| < r \\ \mathbf{0}, & \|\Delta\mathbf{p}_j\| \geq r \end{cases} \tag{11}$$

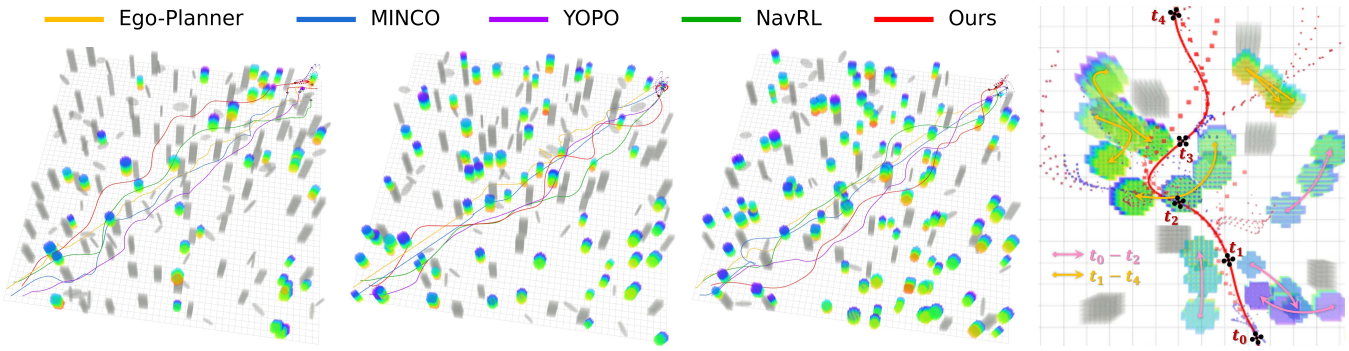
where S_j represents the predicted score of the j-th primitive trajectory. When a primitive generates a severely curved trajectory without the presence of nearby obstacles, it typically reflects an evasive maneuver based on predicted positions of dynamic obstacles. Consequently, this term enables the planner to integrate future dynamic information, allowing the UAV to anticipate and adjust its trajectory in advance.

Finally, the total repulsive force $\mathbf{F}$ can be expressed as $\mathbf{F} = -\sum_j (\nabla\mathbf{U}_j^o + \nabla\mathbf{U}_j^p)$, which is then transformed into the body coordinate system and normalized to $\mathbf{F}^i$. As illustrated in the right diagram of Fig. 1, closer obstacles and distorted primitive trajectories generate stronger repulsive forces, pushing the trajectory outward. The planning region is rotated by an angle $\Delta\psi = \arctan(\mathbf{F}_y^i, \mathbf{F}_x^i + 1)$ around the z-axis, indicating that the new planning direction is determined by the combined effects of the repulsive force and the forward direction of the original orientation. As shown in Fig. 5b, this adjustment allows the planner to explore a broader range of motion trajectories, enhancing its ability to navigate through extremely crowded environments.

IV. Experiments

A. Experimental Setup

In the simulation environment, static obstacles are instantiated as square columns or discs with arbitrary shape parameters and randomly initialized positions. Dynamic obstacles are modeled as cylindrical point clouds with diverse radii and heights, and their motions are governed by continuously varying acceleration profiles. Specifically, each dynamic obstacle updates its motion state at a frequency of $20\,\text{Hz}$. At each update, the acceleration increment $\Delta\mathbf{a}^o$ is sampled from a uniform distribution within the interval $[-0.05, 0.05]$, with the maximum jerk constrained to $1\,m/s^3$. The resulting



(a) 140 Static & 40 Dynamic. (b) 100 Static & 70 Dynamic. (c) 70 Static & 100 Dynamic. (d) Visualization of Trajectory.

Fig. 6: Examples of Different Dynamic Scenario Settings and Visualization of Trajectory Planning Results.

velocity increment and displacement are approximated as $\Delta \mathbf{v}^o = \mathbf{a}^o \Delta t$ and $\Delta \mathbf{p}^o = \mathbf{v}^o \Delta t + 0.5 \mathbf{a}^o \Delta t^2$, respectively.

To ensure stable and efficient trajectory learning, full environmental information is used during training to compute trajectory-level costs and their gradients with respect to the terminal constraints, which are backpropagated through the network. Data collection was performed using Ego-Planner [2], with each randomly initialized map undergoing $600\,s$ of data acquisition at a sensor sampling rate of $10\,\text{Hz}$. At each sampling instance, simulated LiDAR scanning data, odometry information (including pose, velocity, and acceleration), and global dynamic point clouds were recorded.

To bridge the gap between simulation and real-world scenarios, we used a CUDA-accelerated ray-casting algorithm to simulate 32-line LiDAR data with a vertical FOV of -7° to 52° and a resolution of 2°, mimicking a Mid-360 LiDAR with an integration time of $0.1\,s$. The range image is represented as a 40×240 matrix, generated by projecting visible points, and processed with a 3×3 dilation operation. Additionally, six residual frames are recorded over the past $1.4\,s$ at $0.2\,s$ intervals as network input. As illustrated in the green box in Fig. 3, motion primitives are defined as 5×2 lattices with an angular resolution of 30°.

It is worth emphasizing that the Ego-Planner is employed exclusively for generating data with temporally continuous states and observations, improving data utilization. Its trajectories are not used for imitation or regression purposes. To ensure data quality, the collected data were filtered prior to training by excluding samples with visible point cloud distances smaller than the map resolution.

In total, 250,000 valid samples were collected. The training process utilized the Adam optimizer with 150 epochs, a learning rate of 2.5×10^{-4}, and a batch size of 64. During training, the horizontal direction of the center primitive was initially aligned with the x-axis of the body coordinate system, with a 10% probability of random rotation in the xy-plane to introduce scenarios where the planning direction deviates from the flight direction, as described in Sec. III-D.

B. Simulation Experiments

The proposed trajectory planner was evaluated in a $50\,m \times 50\,m \times 5\,m$ simulated environment with varying densities and velocities of dynamic obstacles, and compared against four established open-source algorithms: Ego-Planner [2] (EGO), MINCO [3], YOPO [15], and NavRL [20]. To ensure a fair comparison, obstacle velocity was not directly provided to NavRL. Notably, as NavRL utilizes YOLO for dynamic obstacle detection and cannot differentiate point clouds lacking semantic labels, both dynamic and static point clouds were supplied as segmentation inputs to facilitate its subsequent motion prediction. All simulations were executed on an AMD 3960X CPU and an RTX 3090 GPU.

1) Environmental Dynamism Analysis: Three sets of simulations with varying dynamic obstacle densities were conducted, as illustrated in Figs. 6a–6c. In each simulation, five UAVs, each controlled by a different algorithm, simultaneously took off and traversed the map diagonally toward the opposite corner. The maximum flight velocity was set to $2.5\,m/s$, while the maximum velocity of dynamic obstacles was limited to $1.25\,m/s$. Each experiment was repeated 100 times, and the averages were reported in Table I. Key evaluation metrics included success rate, minimum safe distance to obstacles, trajectory length, total trajectory curvature, and flight time. A trial was deemed successful if the UAV reached the vicinity of the target without collision. For curvature calculation, the trajectory was discretized, and the three-point method based on Heron's formula was employed.

For optimizer-based methods (Ego-Planner and MINCO), replanning is initiated only when a potential conflict is detected. These approaches lack the ability to proactively anticipate dynamic obstacles in the surrounding environment, resulting in frequent abrupt turns and oscillatory trajectories. NavRL utilizes a Kalman filter to model dynamic obstacles under a constant acceleration assumption, which yields high accuracy in low-dynamic scenarios. However, as the density of dynamic obstacles increases, this modeling approach exhibits inherent deviations. The primitive-based method produces trajectories with greater smoothness. In contrast, YOPO's limited FOV and reliance on single-frame depth images restrict its capacity to analyze dynamic information.

Compared to the baselines, the proposed method exhibits clear evasive behaviors in densely crowded regions, steering the UAV toward more open areas. Although the resulting trajectories are longer, the method achieves optimal flight efficiency in highly dynamic tests. Under identical maximum velocity constraints, it either surpasses or closely matches the

TABLE I: Performance Comparison under Varying Environmental Dynamism Levels

Dynamism	Method	Success Rate [%]	Safe Dist. [m]	Traj. Length [m]	Curvature [$1/m$]	Flight Time [s]
140 static 40 dynamic	EGO [2]	34	0.49	74.11	370.78	36.34
	MINCO [3]	41	0.41	**66.93**	161.83	**32.98**
	YOPO [15]	49	0.68	68.58	122.16	37.82
	NavRL [20]	94	**0.79**	70.42	176.58	35.90
	Ours	**96**	0.76	73.15	**80.87**	33.48
100 static 70 dynamic	EGO [2]	11	0.33	76.78	406.46	43.36
	MINCO [3]	15	0.47	68.66	158.27	37.23
	YOPO [15]	34	0.54	**68.44**	140.57	41.29
	NavRL [20]	66	0.67	71.64	185.31	37.80
	Ours	**81**	**0.88**	72.06	**73.53**	**36.04**
70 static 100 dynamic	EGO [2]	0	–	–	–	–
	MINCO [3]	3	0.39	**68.92**	186.51	41.75
	YOPO [15]	18	0.51	70.59	135.59	48.03
	NavRL [20]	41	0.57	71.16	168.28	43.59
	Ours	**62**	**0.81**	72.74	**75.05**	**39.11**

TABLE II: Success Rate [%] and Average Flight Velocity [m/s] under Different Maximum Velocities of Dynamic Obstacles

Method	$1.5\,m/s$		$2.0\,m/s$		$2.5\,m/s$	
	S. R.	Vel.	S. R.	Vel.	S. R.	Vel.
EGO [2]	27	2.167	16	2.069	12	1.981
MINCO [3]	31	2.291	23	2.190	15	2.303
YOPO [15]	44	2.024	41	1.991	26	1.942
NavRL [20]	87	2.223	73	2.267	68	2.185
Ours	**94**	**2.440**	**92**	**2.502**	**87**	**2.489**

fastest method (MINCO). Overall, the proposed approach provides superior safety and a favorable trade-off between trajectory smoothness and efficiency.

2) Dynamic Obstacle Velocity Analysis: We selected the scenario containing 140 static and 40 dynamic obstacles for evaluation. The maximum flight velocity was set to $3\,m/s$, and each parameter configuration was assessed over 100 independent trials. Table II summarizes the resulting statistics for success rate and average flight velocity for each method.

As obstacle velocity increases, the post-conflict avoidance strategies employed by Ego-Planner and MINCO exhibit limited responsiveness. Specifically, when confronted with obstacles exhibiting lateral motion, Ego-Planner tends to evade along the direction of obstacle movement, causing its trajectory to be persistently displaced as obstacles advance and resulting in substantial detours. Due to the restricted FOV of the depth camera, YOPO typically decelerates when obstructed and subsequently steers to avoid obstacles, which markedly diminishes motion efficiency. Although NavRL continues to encounter avoidance failures in highly dynamic scenarios, its overall performance remains superior to methods that do not explicitly address dynamic obstacles.

In contrast, the proposed method consistently achieves the safest and most robust trajectories across all scenarios. Leveraging the extensive perception range afforded by omnidirectional horizontal FOV, the planner demonstrates behaviors such as accelerating through narrow gaps formed by converging obstacles and swiftly evading obstacles approaching rapidly from behind, as shown in Fig. 6d. These acceleration maneuvers improve both collision avoidance success and flight efficiency, enabling the UAV to navigate crowded dynamic environments with greater speed and reliability.

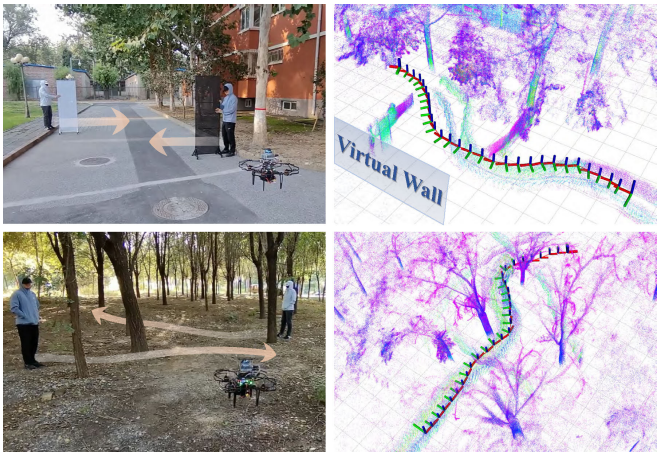


Fig. 7: Examples of Flight Experiments. Sparse point clouds were inserted as virtual walls prior to range image generation to prevent the UAV from circumventing the test obstacles in open areas.

C. Real-World Experiments

We conducted real-world flight experiments in dynamic environments involving trees, movable rectangular boards, and human obstacles, as illustrated in Fig. 7. The UAV is equipped with an NVIDIA Jetson Orin NX onboard computer, which is used to deploy Fast-LIO2 [33] for odometry estimation and to execute the proposed trajectory planner. A Livox Mid-360 LiDAR, mounted atop the UAV, simultaneously provides point clouds and inertial measurements.

Compared to the simulated environments, real-world obstacles are denser, smaller, and induce stronger airflow disturbances among tree branches. To ensure safe operation, we limited the UAV's maximum flight speed to $1.5\,m/s$, reduced the LiDAR's effective detection range to $6\,m$, and increased the dilation kernel for the input image to 5×5.

The computational efficiency was evaluated on the mobile platform, with the average inference time across all test scenarios measured at $3.65\,ms$, representing the duration from receiving LiDAR data to generating planning commands. Experimental results demonstrate the planner's ability to achieve autonomous navigation and safe target reaching with millisecond-level response times, enabling rapid obstacle avoidance in crowded dynamic environments.

V. Conclusion

This paper presents a UAV trajectory planner for crowded dynamic environments. The approach avoids semantic learning for dynamic obstacle detection and explicit kinematic modeling of obstacle motion, which can be redundant and error-prone for fundamental collision avoidance. Instead, the network learns from sequences of range images to infer the evolution of free space along multiple feasible directions. The end-to-end design provides robustness against dynamic misjudgments caused by sensor noise and occlusions. A lightweight map representation supplies accurate and reliable safety metrics during network training, circumventing the suboptimality of imitation learning from expert trajectories and the convergence issues of reinforcement learning. Meanwhile, multi-trajectory prediction over motion primitives, together with a planning-region adjustment strategy, fully exploits the omnidirectional horizontal FOV of LiDAR, further improving adaptability to environmental congestion and dynamics. Remaining challenges include achieving higher flight speeds and more accurate target reaching performance. Future work will investigate speed-insensitive perception inputs, enhanced target localization, and extensions of the framework to more complex swarm-planning scenarios.

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)*, pp. 1934–1940, 2019.

[2] 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.

[3] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[4] 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, 2022.

[5] B. Guo, N. Guo, and Z. Cen, "Obstacle avoidance with dynamic avoidance risk region for mobile robots in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5850–5857, 2022.

[6] M. Lu, H. Chen, and P. Lu, "Perception and avoidance of multiple small fast moving objects for quadrotors with only low-cost rgbd camera," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11657–11664, 2022.

[7] Z. Xu, D. Deng, Y. Dong, and K. Shimada, "Dpmpc-planner: A real-time uav trajectory planning framework for complex static environments with dynamic obstacles," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 250–256, 2022.

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

[9] M. Lu, X. Fan, H. Chen, and P. Lu, "Fapp: Fast and adaptive perception and planning for uavs in dynamic cluttered environments," *IEEE Transactions on Robotics*, vol. 41, pp. 871–886, 2025.

[10] A. Tagliabue, D.-K. Kim, M. Everett, and J. P. How, "Demonstration-efficient guided policy search via imitation of robust tube mpc," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 462–468, 2022.

[11] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[12] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1989–1995, 2023.

[13] J. Tordesillas and J. P. How, "Deep-panther: Learning-based perception-aware trajectory planner in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1399–1406, 2023.

[14] J. Lu, B. Tian, H. Shen, X. Zhang, and Y. Hui, "Lpnet: A reaction-based local planner for autonomous collision avoidance using imitation learning," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7058–7065, 2023.

[15] J. Lu, X. Zhang, H. Shen, L. Xu, and B. Tian, "You only plan once: A learning-based one-stage planner with guidance learning," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6083–6090, 2024.

[16] G. Cheng and J. Y. Zheng, "Sequential semantic segmentation of road profiles for path and speed planning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 23869–23882, 2022.

[17] H.-C. Wang, S.-C. Huang, P.-J. Huang, K.-L. Wang, Y.-C. Teng, Y.-T. Ko, D. Jeon, and I.-C. Wu, "Curriculum reinforcement learning from avoiding collisions to navigating among movable obstacles in diverse environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2740–2747, 2023.

[18] Z. Xie and P. Dames, "Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2700–2719, 2023.

[19] Z. Li, T. Shang, and P. Xu, "Multi-modal attention perception for intelligent vehicle navigation using deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 6, pp. 8657–8669, 2025.

[20] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada, "Navrl: Learning safe flight in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3668–3675, 2025.

[21] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza, "Contrastive learning for enhancing robust scene transfer in vision-based agile flight," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5330–5337, 2024.

[22] W. Yu, J. Peng, Q. Qiu, H. Wang, L. Zhang, and J. Ji, "Pathrl: An end-to-end path generation method for collision avoidance via deep reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9278–9284, 2024.

[23] J. de Heuvel, X. Zeng, W. Shi, T. Sethuraman, and M. Bennewitz, "Spatiotemporal attention enhances lidar-based robot navigation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4202–4209, 2024.

[24] P. Chen, Q. Liu, Y. Li, and S. Ma, "An environmental-complexity-based navigation method based on hierarchical deep reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5119–5125, 2024.

[25] X. Fan, M. Lu, B. Xu, and P. Lu, "Flying in highly dynamic environments with end-to-end learning approach," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3851–3858, 2025.

[26] B. Lau, C. Sprunk, and W. Burgard, "Efficient grid-based spatial representations for robot navigation in dynamic environments," *Robot. Auton. Syst.*, vol. 61, p. 1116–1130, Oct. 2013.

[27] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4423–4430, 2019.

[28] M. Kulkarni, H. Nguyen, and K. Alexis, "Semantically-enhanced deep collision prediction for autonomous navigation using aerial robots," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3056–3063, 2023.

[29] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, "Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6529–6536, 2021.

[30] Z. Yu, Y. Qin, H. Zhao, X. Li, and G. Zhao, "Dual-cross central difference network for face anti-spoofing," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 1281–1287, 2021.

[31] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141, 2018.

[32] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research: Springer Tracts in Advanced Robotics*, pp. 649–666, 2016.

[33] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.