A Safer Vision-based Autonomous Planning System for Quadrotor UAVs with Dynamic Obstacle Trajectory Prediction and Its Application with LLMs

Jiageng Zhong[†,1], Ming Li[*,†,1,2], Yinliang Chen[†1], Zihang Wei[3], Fan Yang[4], Haoran Shen[1]

[1]Wuhan University, [2]ETH Zürich, [3]WFLS, [4]HUST

Abstract

For intelligent quadcopter UAVs, a robust and reliable autonomous planning system is crucial. Most current trajectory planning methods for UAVs are suitable for static environments but struggle to handle dynamic obstacles, which can pose challenges and even dangers to flight. To address this issue, this paper proposes a vision-based planning system that combines tracking and trajectory prediction of dynamic obstacles to achieve efficient and reliable autonomous flight. We use a lightweight object detection algorithm to identify dynamic obstacles and then use Kalman Filtering to track and estimate their motion states. During the planning phase, we not only consider static obstacles but also account for the potential movements of dynamic obstacles. For trajectory generation, we use a B-spline-based trajectory search algorithm, which is further optimized with various constraints to enhance safety and alignment with the UAV's motion characteristics. We conduct experiments in both simulation and real-world environments, and the results indicate that our approach can successfully detect and avoid obstacles in dynamic environments in real-time, offering greater reliability compared to existing approaches. Furthermore, with the advancements in Natural Language Processing (NLP) technology demonstrating exceptional zero-shot generalization capabilities, more user-friendly human-machine interactions have become feasible, and this study also explores the integration of autonomous planning systems with Large Language Models (LLMs).

1. Introduction

In recent years, small-scale quadrotor Unmanned Aerial Vehicles (UAVs) have experienced rapid development due to their compact size, high agility, and strong flexibility. They have been widely employed in various civilian applications [19, 23, 50], such as industrial inspection, cinematography, and mining operations. These application sce-

narios typically feature limited space, unknown, variable structures, and the presence of dynamic obstacles, which pose challenges and risks to UAV flight. Therefore, a reliable real-time onboard planning algorithm is a key prerequisite for enabling UAVs to navigate safely, avoid obstacles, and ultimately achieve autonomous flight capabilities in such environments.

The autonomous navigation of UAVs using onboard sensors has been extensively researched and successfully validated in static environments [6, 14, 15, 39, 50]. However, in dynamic environments, the presence of dynamic obstacles can pose significant difficulties for UAV flight. Since the motion states of these objects are typically unknown, if the UAV treats them as static, it may not have sufficient time to replan its trajectory as it approaches them, potentially resulting in collisions. Thus, unlike obstacle avoidance in static environments, it is imperative to take into account the motion states of dynamic obstacles in environments. However, dealing with dynamic obstacles proves to be a challenging task, especially the accurate perception of such objects during flight, requiring robust and efficient algorithms. Furthermore, considering the limited computational capabilities of onboard computers, which are responsible for environmental perception, navigation, positioning, and interaction functionalities. So, evidently, it is hard to satisfy the efficiency and accuracy requirements at the same time. Existing obstacle avoidance systems [4, 5, 19, 45, 51] in the dynamic environments of quadrotors typically rely on the detection and tracking of moving objects (DATMO) [44] to acquire the motion states of dynamic objects and process them in various ways for subsequent planning. However, these methods still have some shortcomings in terms of dynamic obstacle detection or obstacle avoidance capabilities.

In this paper, we propose a vision-based autonomous planning system for quadrotor UAVs that can predict the future trajectories of dynamic obstacles and generate safer trajectories. First, a very efficient object detection network, NanoDet [34], is first applied to detect dynamic obstacles, followed by Kalman Filtering (KF) to track and estimate their motion states. Then, the static and the dynamic obstacles are both considered in our planning process. We utilize

---

a B-spline-based search algorithm, optimized with several constraints to enhance safety and smoothness. The effectiveness of our system is verified through various experiments in simulation and real-world environments. Unlike planners that rely on traditional methods for detecting dynamic obstacles [5, 19, 45], the detector used in our method is based on deep learning. Consequently, it not only provides information about the sizes and locations of the objects but also offers rich semantic data, enabling broader potential applications in the future.

To make drones more user-friendly, further support from human-machine interaction technology is also essential. Natural Language Processing (NLP) has long been recognized as an important way of human-robot interaction, and the recent rapid advancement has given rise to the development of Large Language Models (LLMs), which have demonstrated remarkable performance in different applications [41]. Several LLMs have already achieved significant accomplishments, such as BERT [11], GPT-3 [3], GPT-4 [28] and Codex [9]. The most influential among them is ChatGPT [27], which is a pre-trained generative text model that underwent fine-tuning through human feedback. Recent studies also demonstrate LLMs can be applied to generalize robotics domains [20, 29, 41]. With the support of LLMs, autonomous UAVs undoubtedly gain enhanced versatility. In this paper, we explore how our proposed autonomous planning system integrates with LLMs to assist users in controlling UAVs better, with the hope of providing valuable insights for subsequent research in this domain.

2. Related work

In environments with dynamic obstacles, a robust and reliable autonomous flight system typically requires two essential components: one is the dynamic obstacle perception, and the other is the trajectory planner designed for obstacle avoidance in dynamic environments.

For obstacle detection and tracking, the most commonly used sensors are RGB cameras [7, 47] or depth cameras [19, 26, 48]. Depth cameras, in particular, are favored for small-sized UAV navigation since they provide both images and point cloud data. Some methods [19, 26, 48]utilize U-depth maps extracted from depth maps for obstacle detection. Among them, [19] has achieved impressive results in obstacle avoidance in dynamic environments. While these methods exhibit high computational efficiency, their performance stability is contingent on parameter settings, and they do not possess the capability to recognize objects. [25] applies a 2D feature-based method to achieve obstacle avoidance. On the other hand, [13, 36] utilizes machine learning techniques to enhance performance and acquire semantic information about objects, albeit at a higher computational cost. Additionally, there are methods based on point cloud data [4, 5, 24, 45], which handle dynamic obstacles

by building a dynamic map for both static and dynamic obstacles. [45] integrates dynamic obstacle detection results into the static map generated from the depth point cloud, while [4] utilizes the voxel map to identify dynamic voxels and estimate their velocities. [5] builds a dual-structure particle-based dynamic occupancy map to concurrently depict the static obstacles and dynamic obstacles. To achieve efficient and robust dynamic obstacle perception, we apply a very efficient object detection neural network, and the semantic information obtained is also very useful.

For obstacle avoidance, various strategies have been explored in static environments. The most common pipeline is to detect obstacles and represent them in the occupancy map [40], and then generate a safe flight trajectory using optimization-based methods [8, 22, 49, 50] or sampling-based methods [6, 12]. Among these, EGO-Planner [50] stands as one of the most representative methods, distinguished by its robustness and efficiency. However, in dynamic environments, obstacle avoidance becomes more demanding, as it entails the prediction of future states for dynamic obstacles [5]. Currently, obstacle avoidance systems for quadrotor UAVs in dynamic environments typically apply DATMO [44] for dynamic obstacle perception, followed by the application of modified planning algorithms to generate suitable flight trajectories. The ASAA system [4] uses YOLOv3 [35] for dynamic obstacle detection, subsequently utilizing SORT [1] and active vision for object tracking. It then performs trajectory planning based on real-time sampling and uncertainty-aware collision checking, ultimately enabling the UAV to avoid collisions with slow-moving objects. Similarly, [45] also applies a detection and tracking method, and further optimizes trajectories by considering the predicted position of the dynamic obstacles, thereby ensuring flight safety. In order to address the uncertainty in trajectory prediction, [19] develops a chance-constrained model predictive controller to ensure that the collision probability between the UAV and each moving object remains acceptably low, thus achieving robust obstacle avoidance. [5] achieves efficient trajectory planning through sampling motion primitives in the state space and forming risk with the cardinality expectation in a risk-checking corridor. Based on robust mobile object detection, our method features trajectory prediction and reliable trajectory optimization under several constraints, possessing good generality and user-friendliness.

3. Method

3.1. System framework

The whole vision-based autonomous planning system can be divided into two components, as shown in Figure 1. The first component, the perception module, is primarily dedicated to the construction of local maps and the identi-
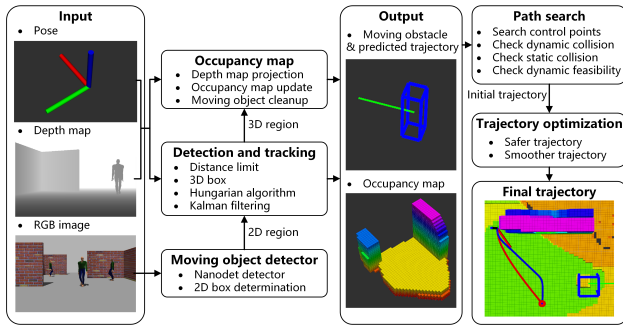
Figure 1. The framework of our autonomous planning system.

fication, tracking, and prediction of dynamic obstacles. On the other hand, the second component, the trajectory generation module, serves the purpose of conducting an initial trajectory search and subsequent trajectory optimization. The input of the system consists of RGB images, depth maps, and the UAV pose. RGB images and depth maps can be directly acquired through an RGB-D camera such as the Realsense D435i, while precise UAV pose information can be obtained using Simultaneous Localization and Mapping (SLAM) techniques, such as VINS-Fusion [31]. These data sources are synchronized using timestamps and then fed into the perceptual module. The output is a collision-free and dynamically feasible trajectory leading to the target point.

In the first stage, the UAV is tasked with the comprehensive perception of its surroundings, encompassing both the static environment and dynamic obstacles. RGB images are input into a lightweight object detection neural network, NanoDet [34], for the purpose of dynamic obstacle detection, yielding 2D regions corresponding to the detected dynamic obstacles. Subsequently, these 2D regions are combined with depth data and UAV poses to derive their respective 3D regions. Further, the Hungarian algorithm is employed for multi-object tracking, enabling the determination of trajectories for the dynamic obstacles over a time interval. Furthermore, the real-time trajectory prediction of dynamic obstacles is achieved using KF algorithm. Additionally, the depth data is employed for the construction of local occupancy maps representing the static components of the scene. In the second phase, inspired by the work of Tang and et al. [37], we design a trajectory generation module. The trajectory planning process can be considered as the task of identifying a set of B-spline control points. This problem is analogous to a graph optimization problem and can be effectively resolved. Based on occupancy maps, the process initiates with the generation of an initial trajectory employing the A* algorithm. Subsequently, an optimization function is formulated, accounting for constraints related to obstacles, UAV velocity, and acceleration. This function undergoes optimization through a solver, ultimately yielding a final trajectory characterized by improved safety and
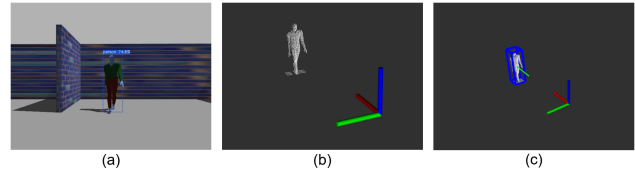

Figure 2. (a) The 2D region of the dynamic obstacle. (b) The corresponding position of the object in the depth map. (c) Residuals of the dynamic obstacle in the local map.

smoothness.

3.2. Dynamic obstacle detection and tracking

To avoid collisions with dynamic obstacles, the first step is to determine the state of these dynamic obstacles, which requires detection and tracking. As depicted in Figure 1, the initial phase involves utilizing captured RGB images to obtain 2D regions of potential moving objects (as shown in Figure 2 (a)). Given the limited computing resources of UAV, a lightweight neural network, NanoDet, is employed for this purpose. In comparison to point cloud-based clustering algorithms [17, 26], this detection method exhibits greater robustness, with reduced sensitivity to algorithm parameters and environmental variations, while simultaneously delivering efficiency and high accuracy. Additionally, the extracted semantic information can also contribute to further human-machine interaction. After acquiring the 2D regions of objects, it becomes feasible to determine their positions in the depth map, as shown in Figure 2 (b). Finally, the depth information of obstacles can be extracted with the elimination of background depth values. This procedure enables the extraction of 3D information about the objects, including 3D regions and fine structural details.

Based on the results of object detection, object tracking can be performed. By calculating the pairwise distances between the sets of detected objects in two different time-step images, the Hungarian algorithm is applied to obtain the optimal assignment. This allows the association of information across multiple frames, facilitating object tracking. Furthermore, with the poses of cameras, it becomes feasible to estimate the object's velocity, trajectory, and other state information over a certain time period. The motion status of an object is determined based on its velocity. If its estimated velocity consistently exceeds the predefined velocity threshold v_d, it is classified as a moving object; otherwise, it can be considered a component of the static map.

Typical autonomous planning systems utilize a local map for planning, however, without special handling, these systems often incorporate the depth point cloud of detected dynamic obstacles into the map, rather than removing it. However, if the depth point cloud of dynamic obstacles is added to the local map, the UAV may perceive an obstruction in its path, as depicted in Figure 2 (c). Therefore, it is essential to exclude dynamic obstacles from the static map

and clear their motion histories. In practice, based on the previously detected 3D regions, we can directly remove occupancy voxels at the current positions of dynamic obstacles, effectively eliminating their interference and obtaining a static map for subsequent planning.

Our planning system for avoiding dynamic obstacles is based on predicting their future paths. Predicting the future trajectories of objects has consistently been an interesting subject, with diverse methods [30, 43, 46]. However, given the need to control UAV flight efficiently, the prediction algorithm must be highly efficient. Following the work of Eppenberger [13], we adopt a conservative motion model to estimate the velocities and short-term future trajectories of detected dynamic obstacles. Assuming dynamic obstacles move on a horizontal plane, their velocity can be estimated using Kalman Filter (KF). For a dynamic obstacle, its state vector $\vec{x} = [x, y, v_x, v_y]$ represents its position and velocity on the horizontal plane. The measurement input $\vec{z}_t$ at time t for KF is the centroid of the object measured in the horizontal plane. The system dynamics and measurement model are defined as:

$$\vec{x}_{t+1} = A \cdot \vec{x}_t + Q, \tag{1}$$

$$\vec{z}_t = H \cdot \vec{x}_t + R, \tag{2}$$

where A is the state-transition model, H is the observation model, and Q and R model the system noise and measurement noise, respectively. This allows for real-time short-term prediction of dynamic obstacle trajectories. It should be noted that if a significant deviation between the measured target position and the position estimated by KF is detected during tracking, it is considered as a tracking error. In such cases, previous results are cleared, and tracking is reinitiated.

3.3. Trajectory generation and optimization

In this section, we present a trajectory planning algorithm suitable for dynamic environments. Many existing algorithms either solely consider static environments [49, 50] or focus exclusively on avoiding dynamic objects [19, 38], making it challenging to meet the condition of collision-free trajectories entirely. Inspired by [37, 38], we introduce a trajectory generation method based on B-spline curves and convex hull collision detection, as B-splines can be used to ensure the required smoothness of the trajectory. This approach can be divided into two phases: initial trajectory search and trajectory optimization. It not only ensures collision avoidance in static environments but also takes into account dynamic obstacles and their future motion states. Specifically, we employ collision detection between convex hulls to rapidly assess the safety of trajectories. As illustrated in Figure 3, for instance, the initial trajectory comprises three B-spline control points, p_0, p_1, p_2. During
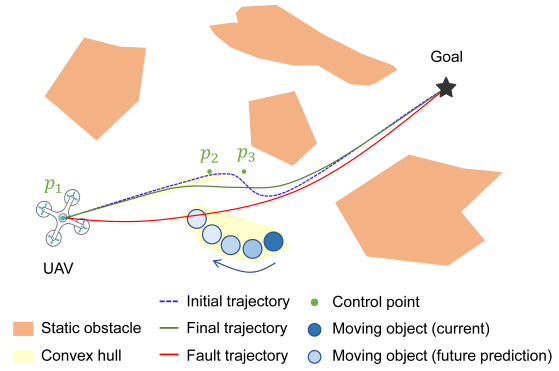

Figure 3. Trajectory generation. This stage takes into account not only the static environment but also dynamic obstacles and their future possible trajectories.

the search process, control points are extended in multiple directions, and the trajectory is enclosed within the convex hull formed by these control points. Additionally, the convex hulls of dynamic obstacles are modeled, requiring collision checking with dynamic and static obstacles for each generated trajectory while preserving dynamic constraints. This process ultimately yields a trajectory to the target point.

In the planning process, the initial trajectory search is essential, as obtaining a reasonable initial trajectory can reduce the optimization time. Inspired by the A* algorithm [16] and B-spline-based Non-uniform Kinodynamic (BNUK) search [37], we apply the A* algorithm to directly search for control points within the free space. The cost function of the algorithm is defined as follows:

$$F = g + \lambda h, \tag{3}$$

where g represents the smoothness of the curve above, h is the heuristic function, and λ is the bias, employed to standardize units and to fine-tune the quality of the curve. Here, the heuristic function is defined as the distance to the target point. A smoother curve can lead to less trajectory optimization time and enhance the feasibility of UAV flight control, and the heuristic function aids in guiding the trajectory closer to the target point. To ensure that the B-spline curve passes through the start and target points, we position the initial three control points at the starting location. Likewise, we apply the same treatment to the control points at the target point. Moreover, to guarantee that the obtained trajectories are both smooth and continuous, the latter three control points of the current curve segment are used as the initial control points. This ensures that each trajectory segment is uniformly smooth and seamlessly connected.

Although the initial trajectory ensures collision avoidance, it may still exhibit issues such as proximity to obstacles or excessive changes in direction, as shown in Figure 4. Therefore, further optimization is required to achieve a trajectory that is not only safer but also better suited for UAV
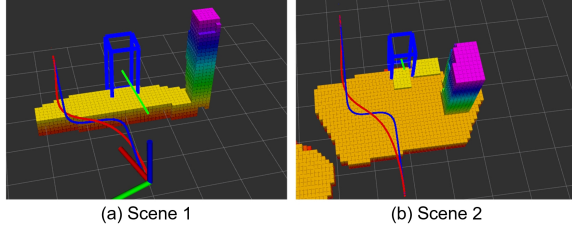
(a) Scene 1   (b) Scene 2

Figure 4. Trajectory Optimization. The blue line represents the trajectory before optimization, the red line represents the trajectory after our optimization, the blue rectangular wireframe represents the bounding box of the dynamic obstacle, and the green line denotes its predicted future trajectory. After optimization, the trajectory becomes smoother.

flight. Following Usenko et al. [40], we represent this problem as an optimization of the following cost function:

$$E_{total} = E_{collision} + E_{smoothness}, \qquad (4)$$

where $E_{collision}$ is a collision cost function and $E_{smoothness}$ is the cost of the integral over the squared derivatives (acceleration, jerk, snap). The former ensures that the UAV will not collide with the static environment or moving targets, while the latter contributes to a smoother trajectory, aiding in flight control. Furthermore, assuming the plane that separates the convex hull of control points and the obstacles is defined as $n^T x + d = 0$, where n and d are the parameters of this plane, the optimization must also satisfy the following conditions:

$$\begin{cases} n^T O + d \geq 0 \\ n^T Q + d \leq 0 \\ abs(v) \leq v_{max} \\ abs(a) \leq a_{max} \end{cases}, \qquad (5)$$

where O represents the vertices of the obstacle's convex hull, and Q represents the control points of the trajectory, $abs(\cdot)$ represents the element-wise absolute value, and v and a represent velocity and acceleration of the UAV, respectively. In this study, this problem is solved using the augmented Lagrangian method [2, 10], and with the low-storage BFGS algorithm [21] for local optimization. The interface used for these algorithms is NLopt [18].

4. Experiments

We test our method in both simulation and real-world environments. All the methods are implemented in C++ and executed using the robot operating system (ROS) [32] in Linux. Sections 4.1 and 4.2 present experiments in the simulation environments. The Gazebo simulation environment with the PX4 firmware is utilized. Section 4.1 covers experiments about dynamic obstacle detection and tracking, and Section 4.2 focuses on assessing the dynamic obstacle

(a) Experiment 1   (b) Experiment 2

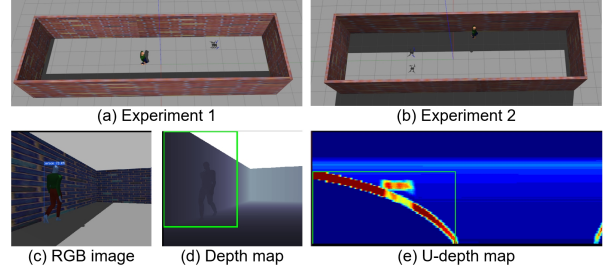(c) RGB image   (d) Depth map   (e) U-depth map

Figure 5. The simulation experiments and object detection examples. (a) and (b) display the overhead views and reconstructed maps of two experiments in this simulation environment. (c) shows the detection results on the RGB image, (d) presents the detection results on the depth map, and (e) is the U-depth map corresponding to the depth map in (d).

Table 1. The RMSE of the estimated positions and velocities of the pedestrian.

Method	Position error (m)	Velocity error (m/s)
Method I [45]	0.11	0.19
Method II [19]	0.14	0.36
Method III [48]	0.11	0.08
Our method	0.14	0.16

avoidance capability. In Section 4.3, we conduct relevant experiments by implementing our method on a real UAV.

4.1. Dynamic obstacle detection and tracking

For quantitative evaluation, we design a simulation environment, as depicted in Figure 5 (a). This setting represents a corridor measuring 15 meters in length and 4 meters in width, where a pedestrian moves back and forth. Initially, the UAV and the pedestrian approach each other. In this experiment, our method is compared with three current detectors, denoted as Method I [45], Method II [19] and Method III [48]. During the flight, the UAV estimates the positions and velocities of the pedestrian, and these measurements are compared with the ground truth values provided by the simulation environment. The comparison is performed by calculating the Root Mean Square Error (RMSE) for both position and velocity. The results are shown in Table 1. In terms of position estimation, there is little difference in accuracy among the four methods. However, concerning velocity estimation, our method exhibits higher accuracy, surpassing Method I and Method II, albeit slightly inferior to Method III.

Method II and Method III are based on U-depth map, and the U-depth map is derived from the depth map. It's worth noting that although Method III achieves higher accuracy, it may fail in certain scenarios. For example, in the scenario depicted in Figure 5 (b), we position the pedestrian not in the center of the corridor but rather 0.5 meters away from the wall. Since our method is based on RGB images,
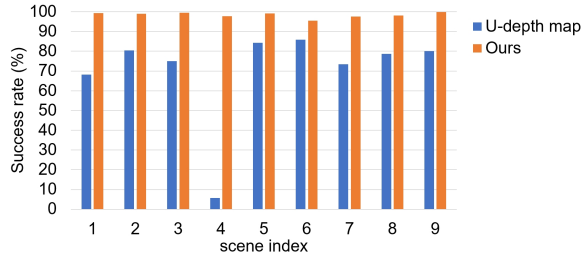
Figure 6. The success rate of moving target detection.

it can accurately detect the pedestrian (Figure 5 (c)). In this scenario, the depth values for the person and the adjacent wall are very close (as indicated by the green box in Figure 5 (d)). As shown in Figure 5 (e), the green box in the U-depth map corresponds to the wall and the pedestrian, and it is hard to distinguish them, potentially leading to a failure in detection. For further evaluation, we place the pedestrian in various positions, and for each frame, we perform detection using the U-depth map and our method. Subsequently, we calculate the detection success rate, and the results are illustrated in Figure 6.

4.2. Autonomous obstacle avoidance

This section evaluates the effectiveness of our planning algorithm in dynamic obstacle avoidance in the corridor simulation environment described in Section 4.1. Similarly, we position a pedestrian in the center of the corridor, who moves back and forth at a constant velocity. The depth sensing range of the UAV is 6 m. In the initial state, the UAV has no depth values for the pedestrian, and consequently, the local occupancy map used for trajectory planning does not contain this information. When the UAV detects the pedestrian, it rapidly replans the trajectory to avoid a head-on collision with the pedestrian. At the start of the experiments, the pedestrian's velocity is set to 0 and increased by 0.1 m/s until a collision occurs. The maximum speed of the UAV is set to 1.5 m/s, and the maximum acceleration is set to 3 m/s2. EGO-Planner [50] is also employed for comparison, and the results are presented in Table 2. When the pedestrian's velocity exceeds 0.8 m/s, the trajectory planned by EGO-Planner results in a collision with the person. This is primarily because it does not consider that the pedestrian is moving towards it. In contrast, our method considers the short-term future movements of the UAV, resulting in a trajectory that causes the UAV to turn in advance to avoid the pedestrian. Typically, a pedestrian's walking velocity is around 1 m/s, indicating that our method is applicable to everyday environments.

Following [5], we conduct tests in additional scenes. We set up two different scenes, where Scene I includes 1 pedestrian, and Scene II includes 4 pedestrians, and then let the UAV fly multiple times in different scenes. Each method is tested 15 times in each scene, and the number of times dif-

Table 2. The maximum velocity of pedestrians without collisions.

Method	Maximum velocity of the pedestrian
EGO-Planner [50]	0.8 m/s
Our method	1.7 m/s

Table 3. The maximum velocity of pedestrians without collisions.

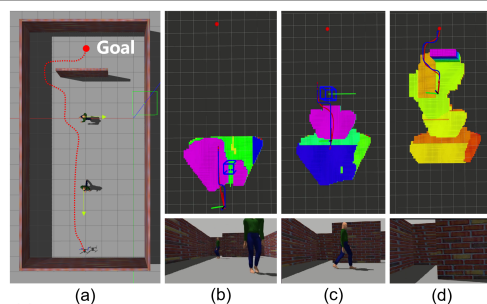| World	Event	EGO-Planner	Ours
Scene I (1 pedestrian)	Collision	13	0
	Freezing	0	0
	Success	2	15
Scene II (4 pedestrians)	Collision	15	3
	Freezing	0	0
	Success	0	12

Figure 7. The planning process in the simulation environment. (a) is an overview of the scene, with the red line representing the approximate flight path of the UAV, and yellow arrows indicating the pedestrians' movement direction as the UAV approaches. (b)-(d) represent different stages of flight, with the blue line indicating the initial trajectory and the red line showing the final trajectory.

ferent events occur is counted, as shown in Table 3. "Collision" represents a collision occurrence, "Freezing" indicates situations where the UAV cannot plan a correct trajectory and remains stationary, and "Success" denotes normal and safe flight. It is evident that our method exhibits a significantly higher success rate. In the scene with only 1 pedestrian, no collisions occur, and in cases with multiple pedestrians, the probability of collision is low. This is due to the consideration of the future motion state of dynamic obstacles in the planning process.

A UAV planning process in a simulated environment with multiple moving objects is present in Figure 7. As shown in Figure 7 (a), the goal is located behind a wall in front of the UAV, and there are two pedestrians with different movement directions performing back-and-forth motion along the path. When encountering the first pedestrian (Figure 7 (b)), the UAV can proactively adjust its course from a distance to facilitate avoidance. When facing the second pedestrian (Figure 7 (c)), the UAV chooses to maneuver around from behind. Overall, the UAV can effectively plan safe and flight-friendly trajectories in the presence of both dynamic and static obstacles. The velocity profile of the
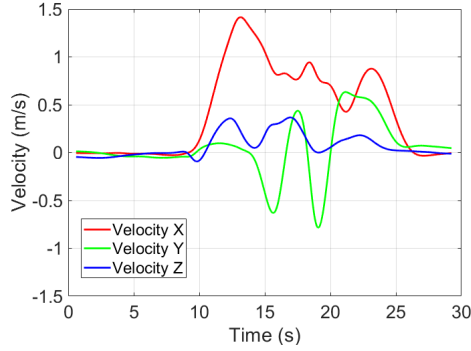
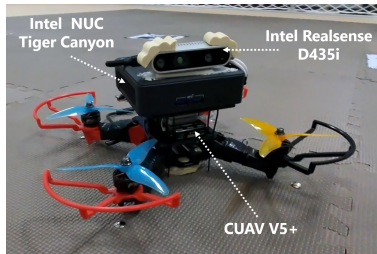Figure 8. The velocity profile of the UAV in flight.

Figure 9. The hardware structure of our quadrotor.

UAV during this process is illustrated in Figure 8, which aligns with the experimental requirements.

4.3. Real-world experiments

In real-world experiments, we employ a compact quadrotor with the hardware configuration shown in Figure 9. The sensor employed for depth map capture is the Realsense D435i, and the flight control system used is the CUAV V5+. The onboard computer used is a NUC equipped with an i5-1145G7 processor. This UAV utilizes the stereo infrared images captured by D435i for running VINS-Fusion [31], thereby achieving visual localization, pose estimation, and 3D reconstruction.

We perform tests in an indoor environment, as shown in Figure 10. In this scenario, the pedestrian is considered as a dynamic obstacle. Initially, the drone approaches the pedestrian, and ultimately, it needs to fly to a location near the pedestrian's starting point. When planning with the EGO-Planner, there is a higher risk of colliding with the oncoming pedestrian, and mapping errors could lead to incorrect judgments. In contrast, when using our system, the UAV will go around in advance, remove the pedestrian from the local map, and finally reach the target point successfully. We also evaluated the efficiency of the system, finding that it takes approximately 200 milliseconds for the system to complete one trajectory planning cycle, which is sufficient to handle common indoor dynamic obstacles.
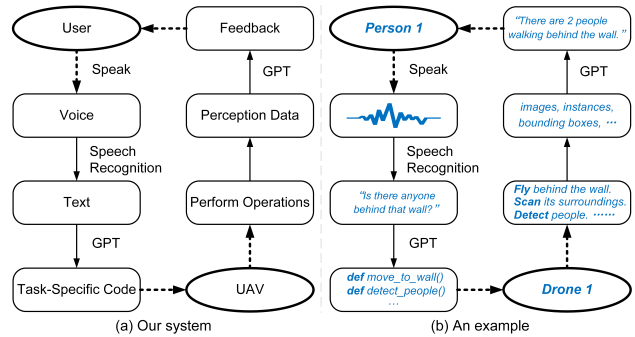

Figure 10. The testing scenario in the real world.

Figure 11. (a) A system combining autonomous UAVs and LLMs. (b) An example of a task executed by this system. "GPT" stands for Generative Pre-trained Transformer, representing models such as ChatGPT.

5. Autonomous UAVs with LLMs

In this section, we examine the integration of LLMs with autonomous UAVs, primarily aimed at enhancing user-drone interaction. This integration allows users to assign tasks to UAVs directly using natural language, without requiring in-depth knowledge of programming languages, while enabling UAVs to provide feedback to users in natural language or other modalities. Previous research [41] has demonstrated that ChatGPT [27] can take high-level textual feedback about generated code or its performance and map it to the required low-level code changes, making it accessible to users who may not possess technical expertise. For instance, ChatGPT is capable of generating complex code structures for drone navigation by utilizing only the fundamental Application Programming Interfaces (APIs) provided in the prompt, and there are many explorations about the formulation of initial prompts for various problem-solving tasks. Furthermore, GPT-4 supports multi-modal capabilities [28], which enhance the diversity of human-machine interactions. Therefore, in this study, we use ChatGPT as an example to explore the integration of autonomous UAVs with LLMs.

Building upon ChatGPT and existing relevant research, we exemplify a system integrating autonomous UAVs with LLMs, as demonstrated in Figure 11 (a). This system is based on UAVs with autonomous flight capability, and utilizes LLMs for facilitating communication between users and UAVs. Specifically, the user provides a task descrip-

```
USER: Fly a rectangular trajectory
GPT:
To fly a rectangular trajectory, you can use the
`publish_goal()` function to send the drone to each
corner of the rectangle. Here's an example code to
fly a rectangle with the corners at (-5, -5, 1), (5,
-5, 1), (5, 5, 1), and (-5, 5, 1):
```python
# Fly to the first corner
publish_goal(-5, -5, 1)
# Fly to the second corner
publish_goal(5, -5, 1)
# Fly to the third corner
publish_goal(5, 5, 1)
# Fly to the fourth corner
publish_goal(-5, 5, 1)
# Fly back to the starting position
publish_goal(-5, -5, 1)
```
This code will send the drone to each corner of the
rectangle in sequence, and then send it back to the
starting position. You can adjust the coordinates
and the height of the rectangle to fit your
specific needs.
```
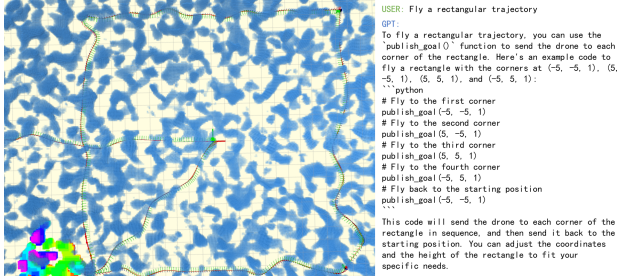
Figure 12. An example experiment of the integration of autonomous UAV with LLM in a simulation environment. The UAV, guided by an GPT-enhanced Ego-Planner, can successfully navigate a user-specified rectangular trajectory while adeptly avoiding obstacles. This experiment underscores the advanced spatial awareness and command understanding capabilities obtained by the integrating LLMs with UAV control algorithms.

tion to the UAV through natural language. Initially, the UAV uses voice recognition technology (such as Whisper [33]) to convert the task description into textual information. Subsequently, it utilizes a Generative Pre-trained Transformer (GPT) (such as ChatGPT) to comprehend and analyze the task, generating the necessary code for task execution, which is then transmitted to the UAV. The UAV finally executes the programs to complete the task, all the while returning acquired data to provide feedback to the user. An exemplary task executed using this system is shown in Figure 11 (b). A user inquires if there are persons behind a wall. Subsequently, the UAV comprehends the task, generates the requisite code, and proceeds to fly to the wall. It then scans the surroundings, detects persons, and finally expresses the collected information to the user using GPT. During this process, it can also provide real-time updates to the user, such as reporting the number of people observed during flight or its need to navigate from the right side due to obstruction by a box on the left. The simulation experiment in Figure 13 showcases an autonomous UAV integrated with LLM.

The two central components of this system are GPT and autonomous UAVs. A primary role of GPT is to generate task-specific code. The basic drone control operations (including takeoff, landing, steering, detection, and recognition) use universal code that does not need to be regenerated. Therefore, for finer drone control, it is advantageous to establish a library of APIs for robot control and perception, which can be readily adapted for use with GPT. Based on previously defined APIs, GPT can build a new function for more complex tasks [41]. And given the functions, GPT can generate controls for long-step tasks [42]. As for autonomous UAVs, they must possess the capability for safe execution of diverse operations and fundamental decision-making abilities. For the instance in Figure 11(b), when the UAV is required to fly to the other side of a wall, it should at least be able to perform trajectory planning in a

static environment. Given the potential presence of moving objects, such as people, it should also be capable of avoiding dynamic obstacles. And given the task specified by the user that involves more multi-steps, it should be able to break down the task and generate a schedule, then execute sequentially. In our proposed approach, significant interaction occurs between the UAV and real-time data, enabling dynamic obstacle avoidance—an important aspect of drone control. It is worth noting that while LLMs inherently face challenges in directly leveraging real-time sensor data for immediate responses, their integration with our system plays a different but critical role. In this case, the LLM acts as a bridge, providing a more user-friendly interface that effectively exploits the capabilities of real-time drone methods. By combining the advanced technical capabilities of drones in dynamic obstacle navigation with the intuitive, user-centered interface provided by LLM, our approach makes a step towards realizing intelligent UAVs. This synergy not only improves operational efficiency, but also significantly improves ease of use, making advanced UAV technology more accessible to a wider user base.

## 6. Conclusion and future work

This paper presents an autonomous planning system for quadcopter UAVs suitable for dynamic environments. We employ a lightweight neural network to detect dynamic obstacles, then use KF for object tracking and prediction, and take into account both static environments and dynamic obstacles in trajectory planning. For trajectory generation, we utilize a B-spline-based trajectory search algorithm and optimize the trajectory with multiple constraints, resulting in trajectories that are not only safer but also better compliant with the UAV's kinematics. Experimental results in both simulation and real-world environments demonstrate that our method can successfully detect and avoid collisions with obstacles in dynamic environments in real-time. Building upon this, we explore the integration of UAVs equipped with autonomous flight systems with LLMs to serve upper-level applications. This represents an important avenue for future development and an area of focus for our forthcoming research efforts.

## Acknowledgments

# References

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016. 2

[2] Ernesto G Birgin and Jose Mario Martinez. Improving ultimate convergence of an augmented lagrangian method. *Optimization Methods and Software*, 23(2):177–195, 2008. 5

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

[4] Gang Chen, Wei Dong, Xinjun Sheng, Xiangyang Zhu, and Han Ding. An active sense and avoid system for flying robots in dynamic environments. *IEEE/ASME Transactions on Mechatronics*, 26(2):668–678, 2021. 1, 2

[5] Gang Chen, Peng Peng, Peihan Zhang, and Wei Dong. Risk-aware trajectory sampling for quadrotor obstacle avoidance in dynamic environments. *IEEE Transactions on Industrial Electronics*, 2023. 1, 2, 6

[6] Gang Chen, Dongxiao Sun, Wei Dong, Xinjun Sheng, Xiangyang Zhu, and Han Ding. Computationally efficient trajectory planning for high speed obstacle avoidance of a quadrotor with active sensing. *IEEE Robotics and Automation Letters*, 6(2):3365–3372, 2021. 1, 2

[7] Hsiang-Chieh Chen. Monocular vision-based obstacle detection and avoidance for a multicopter. *IEEE Access*, 7:167869–167883, 2019. 2

[8] Jing Chen, Tianbo Liu, and Shaojie Shen. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1476–1483. IEEE, 2016. 2

[9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 2

[10] Andrew R Conn, Nicholas IM Gould, and Philippe Toint. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991. 5

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[12] Mihir Dharmadhikari, Tung Dang, Lukas Solanka, Johannes Loje, Huan Nguyen, Nikhil Khedekar, and Kostas Alexis. Motion primitives-based path planning for fast and agile exploration using aerial robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 179–185. IEEE, 2020. 2

[13] Thomas Eppenberger, Gianluca Cesari, Marcin Dymczyk, Roland Siegwart, and Renaud Dubé. Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10528–10535. IEEE, 2020. 2, 4

[14] Davide Falanga, Suseong Kim, and Davide Scaramuzza. How fast is too fast? the role of perception latency in high-speed sense and avoid. *IEEE Robotics and Automation Letters*, 4(2):1884–1891, 2019. 1

[15] Fei Gao, William Wu, Wenliang Gao, and Shaojie Shen. Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *Journal of Field Robotics*, 36(4):710–733, 2019. 1

[16] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 4

[17] Omid Hosseini Jafari, Dennis Mitzel, and Bastian Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 5636–5643. IEEE, 2014. 3

[18] Steven G. Johnson. The nlopt nonlinear-optimization package. http://github.com/stevengj/nlopt, 2010. 5

[19] Jiahao Lin, Hai Zhu, and Javier Alonso-Mora. Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2682–2688. IEEE, 2020. 1, 2, 4, 5

[20] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023. 2

[21] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 5

[22] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J Taylor, and Vijay Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters*, 2(3):1688–1695, 2017. 2

[23] Liang Lu, Giancarmine Fasano, Adrian Carrio, Maolin Lei, Hriday Bavle, and Pascual Campoy. A comprehensive survey on non-cooperative collision avoidance for micro aerial vehicles: Sensing and obstacle detection. *Journal of Field Robotics*, 2023. 1

[24] Youngjae Min, Do-Un Kim, and Han-Lim Choi. Kernel-based 3-d dynamic occupancy mapping with particle tracking. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5268–5274. IEEE, 2021. 2

[25] Tomoyuki Mori and Sebastian Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *2013 IEEE International Conference on Robotics and Automation*, pages 1750–1757. IEEE, 2013. 2

[26] Helen Oleynikova, Dominik Honegger, and Marc Pollefeys. Reactive avoidance using embedded stereo vision for mav flight. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 50–56. IEEE, 2015. 2, 3

[27] OpenAI. Chatgpt. https://openai.com/blog/chatgpt/, 2023. 2, 7

[28] OpenAI. Gpt-4 technical report. https://cdn.openai.com/papers/gpt-4.pdf, 2023. 2, 7

[29] Aadi Palnitkar, Rashmi Kapu, Xiaomin Lin, Cheng Liu, Nare Karapetyan, and Yiannis Aloimonos. Chatsim: Underwater simulation with natural language prompting. *arXiv preprint arXiv:2308.04029*, 2023. 2

[30] Mark Pfeiffer, Giuseppe Paolo, Hannes Sommer, Juan Nieto, Rol Siegwart, and Cesar Cadena. A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5921–5928. IEEE, 2018. 4

[31] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors. *arXiv preprint arXiv:1901.03638*, 2019. 3, 7

[32] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009. 5

[33] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023. 8

[34] RangiLyu. Nanodet-plus: Super fast and high accuracy lightweight anchor-free object detection model. https://github.com/RangiLyu/nanodet, 2021. 1, 3

[35] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[36] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2759–2764. IEEE, 2016. 2

[37] Lvbang Tang, Hesheng Wang, Peng Li, and Yong Wang. Real-time trajectory generation for quadrotors using b-spline based non-uniform kinodynamic search. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1133–1138. IEEE, 2019. 3, 4

[38] Jesus Tordesillas and Jonathan P How. Mader: Trajectory planner in multiagent and dynamic environments. *IEEE Transactions on Robotics*, 38(1):463–476, 2021. 4

[39] Jesus Tordesillas, Brett T Lopez, and Jonathan 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)*, pages 1934–1940. IEEE, 2019. 1

[40] Vladyslav Usenko, Lukas Von Stumberg, Andrej Pangercic, and Daniel Cremers. Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 215–222. IEEE, 2017. 2, 5

[41] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities. *Microsoft Auton. Syst. Robot. Res*, 2:20, 2023. 2, 7, 8

[42] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered longstep robot control in various environments: A case application. *arXiv preprint arXiv:2304.03893*, 2023. 8

[43] Chuhua Wang, Yuchen Wang, Mingze Xu, and David J Crandall. Stepwise goal-driven networks for trajectory prediction. *IEEE Robotics and Automation Letters*, 7(2):2716–2723, 2022. 4

[44] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007. 1, 2

[45] Yingjian Wang, Jialin Ji, Qianhao Wang, Chao Xu, and Fei Gao. Autonomous flights in dynamic environments with onboard vision. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1966–1973. IEEE, 2021. 1, 2, 5

[46] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017. 4

[47] Linhai Xie, Sen Wang, Andrew Markham, and Niki Trigoni. Towards monocular vision based obstacle avoidance through deep reinforcement learning. *arXiv preprint arXiv:1706.09829*, 2017. 2

[48] Zhefan Xu, Xiaoyang Zhan, Baihan Chen, Yumeng Xiu, Chenhao Yang, and Kenji Shimada. A real-time dynamic obstacle tracking and mapping system for uav navigation and collision avoidance with an rgb-d camera. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10645–10651. IEEE, 2023. 2, 5

[49] Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536, 2019. 2, 4

[50] Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485, 2020. 1, 2, 4, 6

[51] Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019. 1