Integrated Planning and Control for Quadrotor Navigation in Presence of suddenly appearing Objects and Disturbances

Wenyi Liu*, Yunfan Ren* and Fu Zhang

***Abstract*—Autonomous flight for quadrotors in environments with suddenly appearing objects and disturbances still faces significant challenges. In this work, we propose an integrated planning and control framework called IPC. Specifically, we design a framework consisting of a lightweight frontend and an MPC backend. On the frontend, we employ the A* algorithm to generate the reference path on a local map. On the backend, we model the trajectory planning and control problem as a linear model predictive control (MPC) problem. In the MPC formulation, the quadrotor is modeled as a high-order integral system (a linear system) to follow the reference path from the frontend. We use a series of convex polyhedrons (*i.e.*, Safe Flight Corridor, SFC) to represent the free space in the environment and employ the multiple hyperplanes of the polyhedrons as a linear inequality constraint of the MPC problem to ensure flight safety. In this way, the linear MPC generates control actions that strictly meet the safety constraints in a short time ($2\,\mathrm{ms}$-$3.5\,\mathrm{ms}$). Then, the control actions of the linear MPC (*i.e.*, jerk) are transformed to the actual control commands (*i.e.*, angular velocity and throttle) through the differential flatness of the quadrotor. Since the MPC computes the control actions directly according to the obstacles and quadrotor's state at a rather high frequency (*i.e.*, $100\,\mathrm{Hz}$), it improves the quadrotor's response speed to dynamic obstacles and disturbance rejection ability to external disturbances. In simulation experiments involving avoiding a suddenly appearing object, our method outperforms state-of-the-art baselines in terms of success rate. Furthermore, we validate our method in real-world environments with dynamic objects and disturbances using a fully autonomous LiDAR-based quadrotor system, achieving autonomous navigation at velocities up to $5.86\,\mathrm{m/s}$ in dense forests. Our IPC is released as a ROS package on GitHub[1] as open source software.**

***Index Terms*—Aerial Systems: Applications, Integrated Planning and Control, Collision Avoidance;**

I. Introduction

In recent years, quadrotors have rapidly developed and have found widespread applications in urban logistics, inspections, and multi-drone formation flights. These scenarios often involve suddenly appearing objects and disturbances, such as tossed objects, flying birds, and wind gusts, which significantly challenge the reliability and effectiveness of quadrotors in completing flight tasks. To ensure safe flight in these scenarios, quadrotors must respond rapidly to these environmental changes and disturbances, and plan high-quality trajectories that satisfy quadrotors' dynamics.

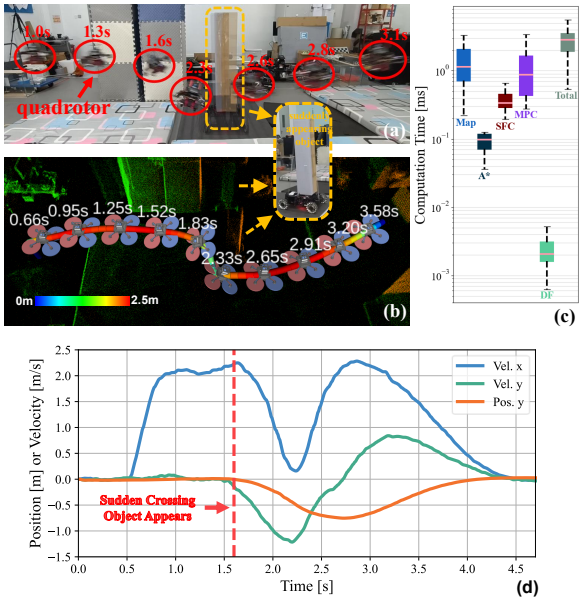

Fig. 1. Indoor flight with suddenly appearing object (Sec. IV-B1). (a) Composite images of the flight. (b) The generated point cloud map during the flight. (c) Average computation time for each step and total latency of IPC in this flight. (d) The position and velocity of the quadrotor during the flight. The dotted line represents the time when the suddenly appearing object appears in front of the quadrotor. More details can be found in the attached video at https://youtu.be/EZFxTkqqat4

Existing quadrotor navigation approaches [1–7] typically employ a planning and control separation framework. In this framework, the planner generates a trajectory that adheres to the dynamical constraints within a safe space, while the controller produces control signals to converge system errors based on the errors between state feedback and trajectory reference. However, this framework could lead to two issues: firstly, the multi-stage pipeline results in increased system latency. Some soft-constraint-based methods [1, 2] can reduce the latency by planning trajectories in very short time, but when faced with suddenly appearing objects, the soft constraints may cause the trajectory optimization results to be trapped in local minima, leading to the trajectory colliding with the obstacle. Secondly, the disturbance rejection issue arises in this planning and control separation framework. The planner does not consider disturbances, resulting in an inability to respond promptly to disturbances (*e.g.*, wind gusts), subsequently affecting the safe flight of the quadrotor.

To address these problems, we propose an **I**ntegrated **P**lanning and **C**ontrol framework called **IPC**. Overall, the contributions of our method are summarized as follows:

Manuscript received: April, 7, 2023; Revised June, 30, 2023; Accepted July, 30, 2023. This paper was recommended for publication by Editor Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers' comments.

*These two authors contributed equally to this work.

Corresponding author: Fu Zhang.

W. Liu, Y. Ren, and F. Zhang are with the Department of Mechanical Engineering, University of Hong Kong {liuwenyi, renyf}@connect.hku.hk, fuzhang@hku.hk

Digital Object Identifier (DOI): see top of this page.

[1]https://github.com/hku-mars/IPC

1) We propose a novel integrated planning and control framework for quadrotors to achieve autonomous navigation in unknown and dynamic environments. The framework consists of a frontend that searches a reference path and a backend that solves the trajectory optimization and control in one step.
2) We propose an efficient, low-latency MPC for integrated quadrotor planning and control. Notably, the MPC decouples the nonlinear quadrotor dynamics into a linear one through differential flatness. Along with the linear corridor constraints and control constraints, we formulate a linear MPC that can be solved in a few milliseconds. The computed control actions are then transformed using differential flatness into angular velocity commands (i.e., the actual control commands) for execution.
3) We compared our method with state-of-the-art baselines in a simulation experiment to avoid suddenly appearing object and demonstrated the superiority of our approach in terms of the success rate.
4) Employing a LiDAR-based fully autonomous quadrotor system, we validated our method in suddenly appearing object avoidance experiments, showcasing its low latency and robust performance. Moreover, our approach demonstrated strong disturbance rejection capabilities during autonomous flight tests in environments with external forces and wind disturbances. Furthermore, the proposed method was validated through autonomous navigation in cluttered environments, achieving flight speeds up to $5.86\,\mathrm{m/s}$ in dense woods.

II. Related Works

A. Corridor-based Trajectory Planning

In recent years, corridor-based trajectory planning methods [3, 5–11] have gained popularity for ensuring safe quadrotor flights. Chen *et al*. [8] divide the space into the OctoMap structure and use free grids directly as corridor constraints. The sphere-shaped corridor has also become popular because each sphere only introduces one constraint in trajectory generation and it can be rapidly obtained through Nearest Neighbor Search (NN-Search) using a KD-tree structure. Gao *et al.*[9] propose generating sphere-shaped corridors within the RRT* framework. Ren *et al.*[3] utilize a sampling-based method to produce spheres while simultaneously maximizing the volume of the corridor and the volume of the overlapping area of adjacent spheres. However, both cube-shaped and sphere-shaped representations may not efficiently represent the free space in cluttered environments, leading to the following trajectory generation problems over-constrained. In contrast, polyhedrons are commonly used as they can represent more complex free space. Liu *et al.*[6] generate free convex polyhedrons through a region inflation method, also known as convex decomposition. The hyperplanes of the polyhedrons are then added to subsequent trajectory optimization problems as hard constraints to ensure safe quadrotor flight. Similar to Liu *et al.* [6], we also employ convex polyhedrons to represent the free space. We formulate the MPC-based planning and control problem as a quadratic programming (QP) problem and enforce the hyperplanes of the polyhedral corridor as linear inequality constraints to achieve collision avoidance robustly and efficiently.

B. Model Predictive Control-based Collision Avoidance

Model Predictive Control (MPC) is a popular strategy for implementing feedback control loops in various systems, as it fully considers input, state, and output constraints. [12] proposes an adaptive trajectory tracking algorithm to compensate the uncertainties in quadrotor models. [13] introduces a data-driven MPC using neural networks, leading to a significant enhancement in quadrotor control effectiveness. [14] presents a Perception-Aware MPC framework that enables simultaneous optimization of perception and control. Although some methods, such as Local Model Predictive Contouring Control [15] and Dynamic Control Barrier Function-based MPC [16], have achieved pedestrian avoidance for autonomous vehicles in unstructured scenarios, the computation complexity of these methods increases with the number of obstacles, resulting in high computation time in complex environments. Furthermore, these methods are mainly designed for ground vehicles moving on 2D, hence not suitable for quadrotor systems moving in full 3D space. In structured indoor scenarios, MPC-based collision avoidance methods have been applied to quadrotor platforms. Neunert *et al.*[17] achieved flight through a tilted window in a motion capture system using Sequential Linear Quadratic (SLQ) to solve unconstrained nonlinear MPC problems. However, their method can only avoid static obstacles in known environments. Small *et al.*[18] implemented autonomous navigation of a quadrotor in a motion capture system with only one obstacle. They incorporated collision avoidance constraints into the cost function and used a warm-start method to improve computation efficiency. However, their method may fail in unstructured scenarios as the warm-start method may require more iterations to converge when the initial trajectory is infeasible, leading to a significant increase in computation time. Lindqvist *et al.* [19] wirelessly controlled a quadrotor to avoid dynamic objects, such as balls and pedestrians, using an external computing device under a motion capture system. They formulated the collision avoidance problem into a nonlinear MPC problem with hard constraints, which is computationally heavy and not suitable for real-world navigation in unknown and unstructured environments with onboard computation devices. Our framework is similar to [20], which uses a similar MPC framework with linear system and polyhedral corridor constraints to plan the trajectories of UAV swarms. However, [20] only considered verification of UAV swarms in simulation, while our framework aims to fullfill autonomous navigation of quadrotor in real-world environments, such as cluttered indoor and outdoor environments and environments with dynamic objects and disturbances.

Compared to [1–12] for autonomous quadrotor navigation, which first optimize a smooth trajectory and subsequently track the trajectory, our IPC achieves trajectory optimization and tracking control in one step through an efficient MPC. The design of our MPC also differs substantially from existing MPC methods in literature. Our MPC both plans the UAV trajectory and controls the UAV movements in unknown and dynamic environments with external disturbances. In contrast,

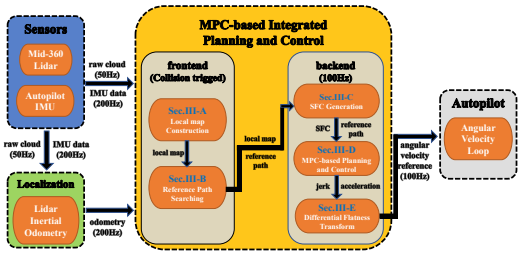

Fig. 2. The overall quadrotor navigation structure. IPC is an integrated planning and control framework consisting of the frontend and backend. By inputting the raw sensor data and localization results, the IPC outputs the angular velocity reference and throttle commands that are further tracked by the UAV onboard autopilot.

[14] proposes perception-aware MPC but does not consider obstacle avoidance. [17–19] utilize NMPC but only achieve obstacle avoidance flight in known indoor environments. [7] applies NMPC with external force modeling, but it is used as a trajectory planner and requires a subsequent tracking controller. [13] uses Neural MPC as a controller to reduce trajectory tracking error.

III. MPC-BASED INTEGRATED PLANNING AND CONTROL

The overall quadrotor navigation structure is illustrated in Fig. 2. LiDAR and IMU sensors provide data for LiDAR-inertial Odometry, which estimates the quadrotor's full state. Based on the odometry and raw cloud, our Integrated Planning and Control (IPC) framework directly generates angular velocity and throttle commands. Specifically, the IPC framework comprises two main components: the frontend and the backend. The frontend consists of Local Map Construction (Sec. III-A) and Reference Path Searching (Sec. III-B). When receiving the current point clouds ($50\,\mathrm{Hz}$), the local map will be constructed based on localization results ($200\,\mathrm{Hz}$, the same as the IMU data). If the obstacle intersects with the reference path or a new user-defined target is received, the Reference Path Searching will be triggered to regenerate the new collision-free reference path.

The backend runs in a loop at a constant frequency ($100\,\mathrm{Hz}$) and consists of three steps: Safe Flight Corridor (SFC) Generation (Sec. III-C), MPC-based Planning and Control (Sec. III-D) and Differential Flatness Transform (Sec. III-E). On the reference path in the local map, a series of overlapping polyhedrons (*i.e.*, SFC) will be generated and used as hard linear constraints for the next step in the MPC problem. At the same time, we generate the reference state of the MPC inside the SFC. By solving the MPC problem and using differential flatness to transform the MPC optimization variables into actual angular velocity references, the quadrotor can directly control the rotor speed through a lower-level angular velocity controller. This way enables the complete motion of the quadrotor in free space.

The symbols used in this paper are defined in Table I. In the system model presented in this paper, the system state of the quadrotor is denoted by $\mathbf{x} = [\mathbf{p}, \mathbf{v}, \mathbf{a}]^T$ and the system input is represented by $\mathbf{u} = \mathbf{j}$.

A. Local Map Construction

Following our previous work ROG-Map [21], we build a robot-centric local map using uniform grids. Considering the high-precision and low-noise characteristics of LiDAR measurements, we propose a temporal forgetting mechanism to update the occupancy status of the grid map and handle dynamic obstacles, instead of using raycasting, thus reducing mapping latency. In the local map, we store the timestamp of the last LiDAR hit for each grid cell, initializing the timestamp for all grid cells to negative infinity. To query the state of a grid cell, we compare its hit time t_{hit} with the current time t_{cur}. If the difference between t_{cur} and t_{hit} is greater than a predetermined forgetting threshold t_{thr}, the cell is considered free; otherwise, it is considered occupied. For occupied cells, we utilize the incremental inflation strategy in [21] to inflate the obstacle by the quadrotor's radius to ensure safety.

TABLE I
NOMENCLATURE

$\mathbf{p}$	position vector p_x, p_y, p_z in the world frame
$\mathbf{v}$	velocity vector v_x, v_y, v_z in the world frame
$\mathbf{a}$	acceleration vector a_x, a_y, a_z in the world frame
$\mathbf{j}$	jerk vector j_x, j_y, j_z in the world frame
N	horizon length in the MPC
$\triangle t$	time step of the MPC

B. Reference Path Searching

To guide the quadrotor to a given target, we search for a collision-free reference path on the local map (Sec. III-A). We opted for A* [22] to search collision-free path although other popular path planning methods such as RRT* could also work. A* has a more deterministic computation time and optimal path length, which would render a more predictable performance. As shown in Fig. 3, once the A* path on the grid map has been determined (yellow path), we start from the first node in the path and search for the farthest node on the path that is visible from the first node (i.e., without being occluded by occupied cells), then we connect the two nodes and repeat the process from the new node until the target node is reached. In this way, we obtain a piecewise shortest reference path (green path).

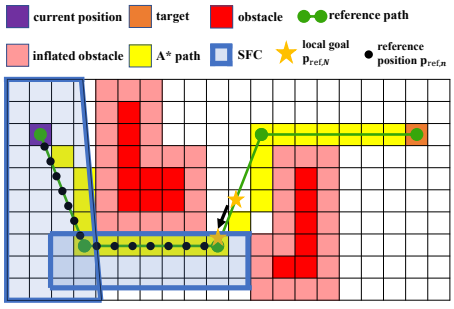

Fig. 3. Reference path searching and SFC generation in a simplified 2D case.

C. SFC Generation

With the local map (Sec. III-A) and a collision-free reference path (Sec. III-B), the backend runs at a constant $100\,\mathrm{Hz}$ to track the reference path. The backend starts with a safe flight corridor (SFC) generation. We directly adopt the method proposed in [6] to partition the free space along the reference path, starting from the position on the reference path that is closest to the UAV current position, into overlapping convex

polyhedrons (see Fig. 3). A polyhedron is represented by a set of hyperplanes $(\mathbf{C}, \mathbf{d})$, which constrains a position $\mathbf{p}$ as:

$$\mathbf{C} \cdot \mathbf{p} - \mathbf{d} \leq 0 \tag{1}$$

where $\mathbf{C}$ is a $K \times 3$ matrix representing the normal vectors of the hyperplanes of the polyhedron, $\mathbf{d}$ is a $K \times 1$ vector representing the constants of the hyperplanes, and K is the number of hyperplanes of a convex polyhedron.

In our current implementation, we limit the maximum number of polyhedrons to two, which is sufficient considering only local trajectory is planned. Moreover, the polyhedron representing free space is generated based on maps that contain occupied cells due to both static and dynamic obstacles in the last t_{thr} seconds (see Sec. III-A). This uniform treatment does not distinguish dynamic objects, nor track or predict their movements. The lack of dynamic objects tracking and prediction are compensated by the high planning and control rate of our overall framework, which can avoid dynamic objects in a purely reactive manner.

D. MPC-based Planning and Control

The second step of the backend is a model predictive controller (MPC), which integrates planning and control into one framework. The goal of the MPC is to guide the quadrotor along the reference path (Sec. III-B) at preset reference speed v_r, while keeping the UAV in the free space represented by SFC (Sec. III-C) and satisfying necessary constraints. Ultimately, MPC yields optimal control actions and a smooth local trajectory.

In order for the MPC to follow the reference path (Sec. III-B), we sample N, the horizon length of the MPC, reference positions $\mathbf{p}_{ref,n}, n = 1, 2, ..., N$, on the reference path. The first reference position $\mathbf{p}_{ref,1}$ is the position on the reference path that is closest to the current UAV position $\mathbf{p}_{odom}$, which is estimated by an odometry, and the last position $\mathbf{p}_{ref,N}$ is the position on the reference path that is $v_r * N * \Delta t$ (with Δt being the model discretization time in MPC). If the calculated $\mathbf{p}_{ref,N}$ falls outside the SFC, we modify it to be the farthest waypoint within the SFC (the movement of yellow star in Fig. 3). Similarly, if $\mathbf{p}_{ref,N}$ falls out of the reference path, we fix it to the end of the reference path. With the determined $\mathbf{p}_{ref,N}$, the rest reference positions $\mathbf{p}_{ref,n}$ are sampled uniformly on the reference path between $\mathbf{p}_{ref,1}$ and $\mathbf{p}_{ref,N}$.

With the reference positions, our MPC is formulated as

$$\begin{aligned}
\min_{\mathbf{u}_k} \quad & \sum_{n=1}^{N} (\|(\mathbf{p}_{ref,n} - \mathbf{p}_n)\|^2_{\mathbf{R}_p} + \|\mathbf{u}_{n-1}\|^2_{\mathbf{R}_u}) && \text{(2a)}\\
& + \|\mathbf{v}_N\|^2_{\mathbf{R}_{v,N}} + \|\mathbf{a}_N\|^2_{\mathbf{R}_{a,N}} + \sum_{n=0}^{N-2} \|\mathbf{u}_{n+1} - \mathbf{u}_n\|^2_{\mathbf{R}_c} \\
\mathbf{s.t.} \quad & \mathbf{x}_n = \mathbf{f}_d(\mathbf{x}_{n-1}, \mathbf{u}_{n-1}), \quad n = 1, 2, \cdots, N && \text{(2b)}\\
& \mathbf{x}_0 = [\mathbf{p}_{odom}, \mathbf{v}_{odom}, \mathbf{a}_{odom}]^T && \text{(2c)}\\
& |v_{i,n}| \leq |v_{i,max}|, |j_{i,n}| \leq |j_{i,max}|, i = x, y, z && \text{(2d)}\\
& |a_{j,n}| \leq |a_{j,max}|, j = x, y && \text{(2e)}\\
& a_{z,min} \leq a_{z,n} \leq a_{z,max} && \text{(2f)}\\
& \mathbf{C}_n \cdot \mathbf{p}_n - \mathbf{d}_n \leq 0 && \text{(2g)}
\end{aligned}$$

where the cost function (2a) consists of $\|\mathbf{p}_{ref,n} - \mathbf{p}_n\|^2_{\mathbf{R}_p}$, the reference path following error, $\|\mathbf{u}_{n-1}\|^2_{\mathbf{R}_u}$, the control efforts, $\|\mathbf{u}_{n+1} - \mathbf{u}_n\|^2_{\mathbf{R}_c}$, the control variation, $\|\mathbf{v}_N\|^2_{\mathbf{R}_{v,N}}$, the terminal velocity, and $\|\mathbf{a}_N\|^2_{\mathbf{R}_{a,N}}$, the terminal acceleration.

The constraints in the formulated MPC problem (2) consist of three. The first one is the model constraints (2b) subject to initial state (2c) estimated by an odometry. To reduce the MPC complexity, we adopt a third-order integrator for the quadrotor:

$$\begin{aligned}
\mathbf{p}_n &= \mathbf{p}_{n-1} + \triangle t \cdot \mathbf{v}_{n-1} + \frac{1}{2}\triangle t^2 \cdot \mathbf{a}_{n-1} + \frac{1}{6}\triangle t^3 \cdot \mathbf{j}_{n-1} \\
\mathbf{v}_n &= \mathbf{v}_{n-1} + \triangle t \cdot \mathbf{a}_{n-1} + \frac{1}{2}\triangle t^2 \cdot \mathbf{j}_{n-1} \\
\mathbf{a}_n &= \mathbf{a}_{n-1} + \triangle t \cdot \mathbf{j}_{n-1} \\
\mathbf{x}_n &= [\mathbf{p}_n, \mathbf{v}_n, \mathbf{a}_n]^T, \quad \mathbf{u}_n = \mathbf{j}_n
\end{aligned} \tag{3}$$

The second constraints are the kinodynamic constraints (2d-2f), which ensure the quadrotor's dynamics are within feasible limits. Specifically, we impose upper and lower limit constraints on the velocity, acceleration, and jerk. We set an independent lower limit on $a_{z,n}$ to constrain the acceleration in the negative direction of the quadrotor's z-axis, ensuring that it does not exceed the gravitational acceleration g.

The third constraints are the corridor constraints (2g), which ensure the quadrotor to remain within the safe flight corridor hence avoiding collision with both dynamic and static obstacles in the environments. In (2g), the predicted UAV position $\mathbf{p}_n$ is constrained to lie in the polyhedrons $(\mathbf{C}_n, \mathbf{d}_n)$ that contain $\mathbf{p}_{ref,n}$. If $\mathbf{p}_{ref,n}$ lies in the overlapped area between two polyhedrons, $(\mathbf{C}_n, \mathbf{d}_n)$ are the union of the two polyhedrons parameters.

The optimization problem (2) involves a quadratic cost and linear constraints in terms of the optimization variables $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{N-1}]^T$, which presents a standard quadratic programming (QP) problem. This QP problem is solved by OSQP-Eigen, a C++ library that depends on OSQP and Eigen3. The resulting solution generates the optimal control actions and local trajectory according to the cost function.

When the MPC (2) is solved successfully at the step k, a series of optimal control actions $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{N-1}]^T$ is recorded, and the $\mathbf{u}_0$ from $\mathbf{U}$ is used as the control action for this step. If MPC cannot be solved at the step k, the time interval ΔT between the last successful step and the failed step k is calculated. Then the $\mathbf{u}_{\lfloor \Delta T / \Delta t \rfloor}$ from the previously successful control actions $\mathbf{U}$ is used as the control action for this step. This control strategy allows for adaptability to failures in MPC and ensures that a reasonable control action is still taken even if MPC cannot be solved.

E. Differential Flatness Transform

After solving the MPC problem (Sec. III-D), the optimal control actions $\mathbf{j}$, defined in the world frame, cannot be directly applied to the quadrotor in the real world because it is not the commands to the quadrotor actuators (*i.e.*, motors). Therefore, we utilize the differential flatness property [23] of the quadrotor to transform the jerk $\mathbf{j}$ along with other states such as acceleration $\mathbf{t}$ into angular velocity reference. The

angular velocity reference is finally tracked by lower-level controllers implemented onboard the autopilot to produce the motor commands.

$$\mathbf{z}_B = \frac{\mathbf{t}}{\|\mathbf{t}\|}, \quad \mathbf{t} = [a_x, a_y, a_z + g]^T \tag{4a}$$

$$\mathbf{x}_C = [\cos\phi, \sin\phi, 0]^T \tag{4b}$$

$$\mathbf{y}_B = \frac{\mathbf{z}_B \times \mathbf{x}_C}{\|\mathbf{z}_B \times \mathbf{x}_C\|}, \quad \mathbf{x}_B = \mathbf{y}_B \times \mathbf{z}_B \tag{4c}$$

$$\mathbf{h}_w = \frac{(\mathbf{j} - (\mathbf{z}_B \cdot \mathbf{j})\mathbf{z}_B)}{\|\mathbf{a}\|} \tag{4d}$$

$$p_r = -\mathbf{h}_w \cdot \mathbf{y}_B, \quad q_r = \mathbf{h}_w \cdot \mathbf{x}_B \tag{4e}$$

$$r_r = (\phi_r - \phi) \cdot \mathbf{z}_B \cdot (0, 0, 1)^T \tag{4f}$$

where g represents the gravitational acceleration, ϕ_r and ϕ are the reference and feedback of the quadrotor's yaw angle in the world frame, (p_r, q_r, r_r) denote the pitch, roll and yaw angular velocity reference in the body frame.

In addition, we also need to calculate the throttle T_r of the quadrotor to control its motion along the Z-axis:

$$T_r = C_T \cdot \|\mathbf{t}\| \tag{5}$$

where C_T is the throttle thrust coefficient that is calibrated beforehand.

IV. Experiments

A. Benchmark Comparison

In this section, we compare **IPC** with two state-of-the-art (SOTA) planning algorithms: **1) Fast-Planner**, which is based on the Euclidean Signed Distance Field (ESDF), and **2) EGO-Planner**, which is a computationally efficient gradient-based optimization method. Simulation experiments were conducted using an open-source LiDAR-based UAV simulator [24], as shown in Fig. 4. It is worth noting that both methods **Fast-Planner** and **EGO-Planner** generate b-spline trajectories parameterized by time t, and the output of those planners are $\mathbf{p}, \mathbf{v}, \mathbf{a}$ evaluated at time t, while the output of IPC is the angular velocity and thrust. In order to eliminate the influence of the controller, the robot model in the simulation environment was configured as an ideal model, where the robot state (*i.e.*, position, velocity, acceleration) are directly set to the b-spline trajectories planned by **Fast-Planner** and **EGO-Planner**, or the robot angular velocity and thrust are directly set to the values as planned by **IPC**.

The quadrotor was given a target point located 10 meters ahead. When it reached a triggering distance d_t during forward flight at a speed of v_f, a object would suddenly appear between the quadrotor and the target point. An experiment was deemed successful if: **1)** the quadrotor effectively avoided the obstacle (*i.e.*, the minimum distance d_{min} to the obstacle greater than the radius r =0.25 m of the quadrotor), and **2)** the quadrotor satisfied all dynamical constraints (*e.g.*, maximum velocity $10\,\mathrm{m/s}$ and maximum acceleration $20\,\mathrm{m/s^2}$). We set up different combinations of d_t and v_f, with the triggering distance set at intervals of $0.5\,\mathrm{m}$ from $0.5\,\mathrm{m}$ to $3.0\,\mathrm{m}$, and the forward speed set at intervals of $0.5\,\mathrm{m/s}$ from $0.5\,\mathrm{m/s}$ to $10\,\mathrm{m/s}$. For each combination, the three methods were tested 10 times to calculate the success rate.

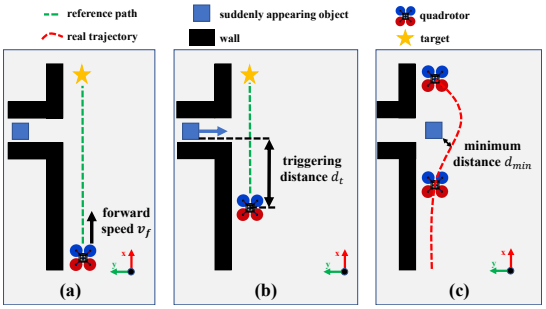

Fig. 4. Description of the simulation experiment. (a) The initial simulation environment is where the suddenly appearing object with dimension of length: $0.2\,\mathrm{m}$, width: $0.2\,\mathrm{m}$, and height: $3\,\mathrm{m}$ is located behind the wall, and the quadrotor flies forward in a straight line. (b) The suddenly appearing object appears on the quadrotor's flight path at a triggering distance d_t. (c) The suddenly appearing object moves to the front of the quadrotor and stops, while the quadrotor will avoid the suddenly appearing object after sensing it.

The simulation results are shown in Fig. 5. Along the simulation results, we also show the boundary (the red line in Fig. 5) below which an ideal second-order integral model can fully avoid the suddenly appearing object. In calculating the boundary, the quadrotor is assumed to decelerate at the maximum deceleration $-20\,\mathrm{m/s^2}$ in the x-direction and accelerates at the maximum acceleration $20\,\mathrm{m/s^2}$ in the y-direction to avoid the obstacle, the boundary is where the minimum distance between the quadrotor and the object is just zero. As can be seen, as forward speed v_f increases and triggering distance d_t decreases, the success rates of all methods decrease, which is expected due to the reduced reaction time. Moreover, our proposed method has a significantly higher success rate under all forward speeds and triggering distances, reaching almost the theoretical bound with a small gap, which is due to the additional jerk constraints in MPC.

For **Fast-Planner**, the acceleration on the trajectory is prone to violate the soft constraints (*e.g.*, the maximum acceleration of the trajectory is $28\,\mathrm{m/s^2}$ at d_t=3 m and v_f=7 m/s, which violates the maximum acceleration $20\,\mathrm{m/s^2}$), rendering the trajectory untrackable by a real quadrotor and leading to the failure of obstacle avoidance. For **EGO-Planner**, it proposes time reallocation to ensure dynamic feasibility, but its warm-starting method (*i.e.*, using the previous trajectory as the initial value) results in a low success rate of trajectory optimization when a suddenly appearing object appears, then the quadrotor follows the previous trajectory and collides with the obstacle. Compared to the computation time of each method in the simulation environment (with a map contains 3030 points in the point cloud and a resolution of $0.1\,\mathrm{m}$), the planning time of **IPC** ($\sim 2.1\,\mathrm{ms}$) falls between that of **EGO-Planner** ($\sim 0.9\,\mathrm{ms}$) and **Fast-Planner** ($\sim 5\,\mathrm{ms}$), and **Fast-Planner** requires the construction of an ESDF map in addition ($\sim 6\,\mathrm{ms}$). The high computation efficiency of **EGO-Planner** is achieved by the warm start of the planning and local modification of the trajectory, which, however, limits the success rate in case of suddenly appearing objects.

B. Real-world Experiments

To evaluate the real-world performance of our IPC in real-world environments, we developed a LiDAR-based quadrotor platform weighing $1.5\,\mathrm{kg}$ with a thrust-to-weight ratio of

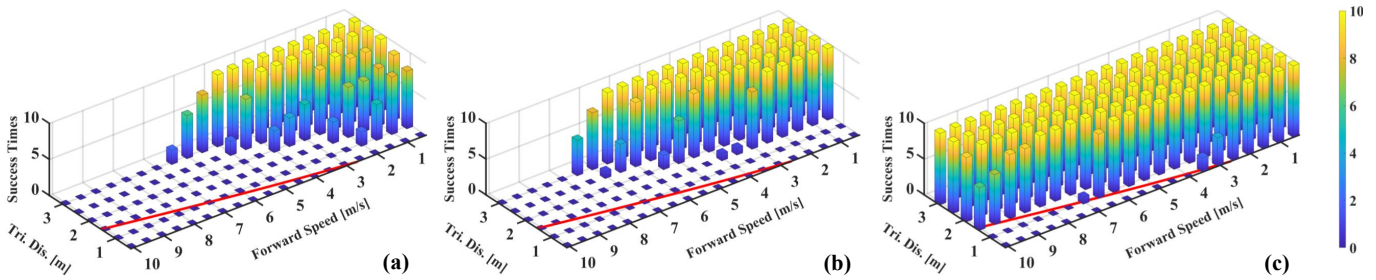

Fig. 5. The success times of the three methods in avoiding suddenly appearing object. The red line represents the theoretical boundary for successful avoidance considering an ideal second-order integral model. When the suddenly appearing object moves, the forward speed v_f represents the quadrotor's speed at this moment, and the triggering distance d_t represents the distance between the quadrotor and the suddenly appearing object edge in the x-direction (as shown in Fig. 4(b)). We conducted 10 experiments for each case and recorded the success times (i.e., the number of times the quadrotor successfully avoided the suddenly appearing object). (a) Fast-Planner. (b) EGO-Planner. (c) Our IPC.

over 4.0. All perception and planning algorithms run in real-time on an onboard Intel NUC with CPU i7-10710U. For sensors, we employ the Livox Mid360 LiDAR and the built-in IMU of Pixhawk. The FAST-LIO2 [25] algorithm serves as the localization module, providing 100 Hz high-quality state estimation and 50 Hz point cloud. Finally, the angular velocity control of the quadrotor is achieved by Pixhawk autopilot.

TABLE II
PARAMETERS OF THE MPC

Parameter	Value	Description
$\triangle t$	0.1	time step of the MPC
N	15	horizon length in the MPC
d_{rad}	0.3	quadrotor radius
g	9.81	gravitational acceleration
$v_{i,max}$	8	maximum velocity in the x,y,z direction
$j_{i,max}$	50	maximum jerk in the x,y,z direction
$a_{j,max}$	$2g$	maximum acceleration in the x,y direction
$a_{z,min}$	$-g$	minimum acceleration in the z direction
$a_{z,max}$	$2g$	maximum acceleration in the z direction

For the MPC problem in the backend of our method, the parameters are shown in Table II. Our weight matrices are diagonal: $\mathbf{R}_p = \mathrm{diag}(2000, 2000, 2000)$, $\mathbf{R}_{v,N} = \mathbf{R}_{a,N} = \mathrm{diag}(200, 200, 200)$, $\mathbf{R}_u = \mathrm{diag}(0, 0, 0)$ and $\mathbf{R}_c = \mathrm{diag}(0.2, 0.2, 0.2)$. Additionally, we adjust the preset reference speed v_r in Sec. III-C to control the nominal flight speed of the quadrotor flight. Our IPC solved the MPC problem at a frequency of 100 Hz onboard and controlled the angular velocity loop of the quadrotor at the same frequency through the MAVLink protocol (*i.e.*, 100 Hz).

To obtain a more comprehensive understanding of our experiments, we invite readers to watch our video[2].

1) Indoor flight with suddenly appearing object: We conducted experiments in an indoor environment with dimension of length: 2.5 m, width: 8 m, and height: 4 m, that is similar to our simulation environment (Sec. IV-A) to verify the performance of our method in avoiding the suddenly appearing object with dimension of length: 0.2 m, width: 0.2 m, and height: 1.0 m, as shown in Fig. 1. We performed ten experiments with forward speeds v_f ranging from 0.5 m/s to 2.3 m/s and triggering distances d_t ranging from 0.5 m to 1.0 m. In these experiments, our quadrotor successfully evaded the suddenly appearing object. A successful agile flight was performed with forward speed of 2.3 m/s and triggering distance of 1.0 m, as shown in Fig. 1(a). In this flight, the average total computation time of the IPC was 2.8 ms, as shown in Fig. 1(c). Although the computation time may exceed 2.8 ms in a few cases due to a small solution space in narrow areas, the total computation time remained below 10 ms, which did not affect our IPC's ability to operate at 100 Hz.

2) Indoor flight with dynamic objects: To demonstrate the safety and low latency of our method, we conducted an indoor experiment where two dynamic objects acted as obstacles to interfere with the flight while the quadrotor performed a five-point star flight on the x-y plane, as shown in Fig. 6. One object moved automatically along a rectangular path at a constant velocity of 0.5 m/s, while the other moved randomly via human operation with a maximum velocity of 0.8 m/s. In this experiment, our quadrotor was able to navigate among the five waypoints safely without colliding with any of the manual and automatic moving objects.

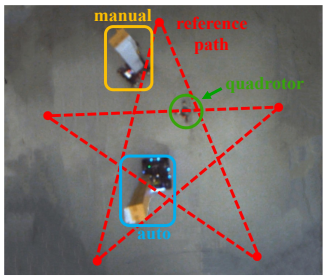

Fig. 6. Top view of the five-star flight: The constant velocity of the automatic object (blue) is 0.5 m/s, and the max velocity of the manual object (yellow) is 0.8 m/s. The two objects have similar shape, with dimension of length: 0.2 m, width: 0.2 m, and height: 1.0 m. The dimension of indoor environment is length: 4.5 m, width: 8.0 m, and height: 4.0 m.

3) Indoor flight with wind and external force disturbances: Besides planning obstacle-free trajectory, our IPC, as an integrated planning and control framework, is also able to cope with external disturbances exerted on the UAV. In this section, we subject the hovering quadrotor to a sudden and violent external force using a stick to test our IPC's disturbance rejection ability. To prevent the quadrotor from unintentionally avoiding the stick and causing the hover position to shift, we disabled the collision avoidance module of the IPC in this experiment. When subjected to rapid and severe external force disturbances, the quadrotor was able to quickly respond and return to the hovering position with a slight overshoot. An example of a disturbance is shown in Fig. 7. We set the hover position at $(0, 0, 0.8)$ m. Due to the simplified linear model used in the MPC, IPC has a small steady state hovering error

[2]https://www.youtube.com/@marslabhku1418

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2023.3311358

(*i.e.*, $0.07\,\mathrm{m}$), as evidenced in Fig. 7(a). When the quadrotor was pushed to $0.46\,\mathrm{m}$ away from the target position by an external force, it took $2.0\,\mathrm{s}$ to recover to the steady state, with an overshoot of $0.05\,\mathrm{m}$.

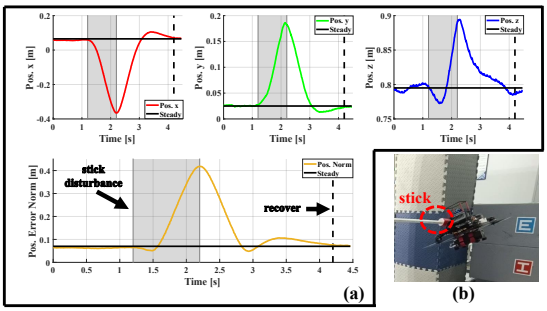

Fig. 7. (a) Time evolution of the quadrotor's position and position norm error in the presence of stick disturbance: The grey area represents the period during which the quadrotor is subject to external forces, while the solid black line indicates the stable state. The black vertical dashed line represents the quadrotor recovery to the hovering state. (b) Third-person view of the flight with external force disturbance by stick.

Furthermore, we re-enabled the collision avoidance module and tested the disturbance rejection ability of our IPC against wind disturbances from two fans and external force disturbances from a stick, as shown in Fig. 8. Despite being disturbed by wind and external forces while flying in a narrow space, the quadrotor could still navigate to its preset target position safely without colliding with any obstacles.

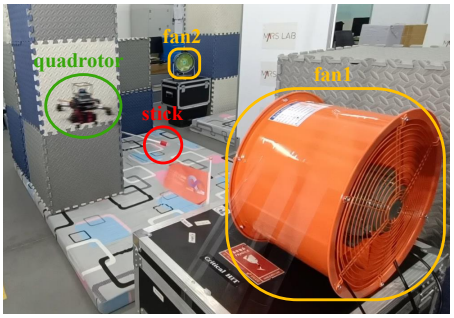

Fig. 8. Third-person view of indoor flight with wind and external force disturbances: The wind speed of fan1 is about $6.6\,\mathrm{m/s}$, while fan2 is about $3.8\,\mathrm{m/s}$. The quadrotor can fly autonomously in narrow spaces with wind disturbances (orange) and quickly adjust and safely reach the target position even under external force disturbances from a stick (red).

4) Outdoor flight in the forest: To further verify the robustness and effectiveness of our method in the outdoor environment, we conducted quadrotor's autonomous navigation in a dense forest with low bushes, as depicted in Fig. 9(a). We performed over ten successful experiments with the maximum velocity ranging from $1.0\,\mathrm{m/s}$ to $6.0\,\mathrm{m/s}$ in the presence of wind speeds ranging from $0.5\,\mathrm{m/s}$ to $3.2\,\mathrm{m/s}$. Fig. 9 shows a typical field flight. Our IPC achieved a maximum velocity of $5.86\,\mathrm{m/s}$. During this experiment, the quadrotor flew towards the target position along the reference path, achieving autonomous navigation with an average speed of $4.51\,\mathrm{m/s}$ along a trajectory of $58.64\,\mathrm{m}$.

C. Evaluation of MPC with Varying Horizon Length N and Running Frequency f

To further demonstrate that the high frequency ($100\,\mathrm{Hz}$) of MPC has improved the quadrotor's ability to avoid suddenly

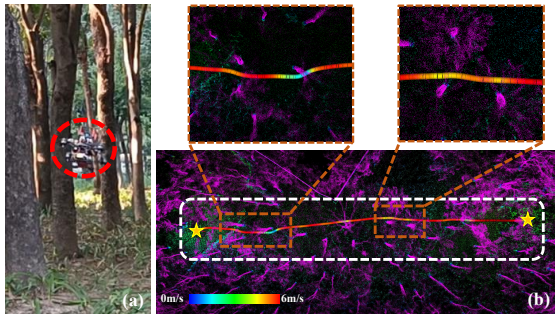

Fig. 9. (a) Third-person view of the flight in the forest. (b) Top view with two partially enlarged views of this outdoor flight: The quadrotor flew through a white rectangular area with dimensions $60\,\mathrm{m} \times 10\,\mathrm{m}$. The white-executed trajectory covered a distance of approximately $58.64\,\mathrm{m}$ with a maximum velocity of $5.86\,\mathrm{m/s}$ and an average velocity of $4.51\,\mathrm{m/s}$. The yellow stars represent the start and end positions of the quadrotor.

appearing objects and attenuate external disturbances, we conducted experiments in two scenarios with different values of N and f in MPC: **1)** Avoiding suddenly appearing object in simulation (v_f is $2.5\,\mathrm{m/s}$, d_t is $1\,\mathrm{m}$). The simulation environment remains similar to Sec. IV-A except that a realistic quadrotor dynamic model is also included in the simulation. We performed five experiments and recorded the success rate R and computation time t_c for each setting. **2)** Real-world hovering experiment with a constant wind disturbance (speed about $6.6\,\mathrm{m/s}$ produced by a fan). We measured the hovering error δ_p and its standard deviation σ_p.

TABLE III
COMPARISON OF VARYING HORIZON LENGTH N AND RUNNING FREQUENCY f IN MPC

N	$f\ (Hz)$	R	$t_c\ (ms)$	$\delta_p\ (m)$	$\sigma_p\ (m)$
5	100	20%	0.2315	0.0967	0.0084
10	100	100%	0.6351	0.0948	0.0099
15	**10**	0%	1.6841	crash	crash
15	**20**	60%	1.5998	crash	crash
15	**50**	80%	1.4724	0.1024	0.0083
15	**100**	100%	1.1488	0.0988	0.0105
25	100	100%	3.3654	0.0988	0.0117
50	<100	0%	18.7223	crash	crash

The results are shown in Table III. As can be seen, at a constant running frequency $100\,\mathrm{Hz}$, the success rate R increases with the horizon N as long as the computation time t_c, which also increases with N, does not prevent the real-time running. This is because when N is small (*e.g.*, 5), the quadrotor fails to avoid obstacles on time due to the shorter prediction horizon of the MPC, while a longer horizon N enables the quadrotor to avoid the dynamic objects further ahead. However, when the horizon N is too large (*e.g.*, 50), the increased t_c affects the real-time generation of control commands (the MPC cannot be solved at $100\,\mathrm{Hz}$), leading to collisions with obstacles. For the control accuracy, it remains more or less similar as long as the MPC runs at the rated frequency $100\,\mathrm{Hz}$.

In terms of running frequency f, it can be seen that both the success rate R and hovering accuracy (evaluated by the error δ_p) monotonically increases with f when N is fixed at 15, proving that the running frequency is a vital factor for both dynamic obstacle avoidance and disturbance attenuation. This is because, at lower frequencies (*e.g.*, $10\,\mathrm{Hz}$,

20 Hz), the quadrotor cannot respond promptly to dynamic obstacles, nor against external disturbances, leading to crashes. As the running frequency increases, the quadrotor can monitor the environment and control errors at a shorter interval and intervene in a more timely manner.

V. Conclusion and Future Work

In this paper, we presented an integrated planning and control (IPC) framework to enable the safe flight of a quadrotor in dynamic and unknown environments. To enable safe flight in the environments with suddenly appearing objects and disturbances, we propose two novel designs. Firstly, we use an MPC-based optimization method with hard linear constraints to significantly reduce calculation time and improve the quadrotor's response to changes in the environment (*e.g.*, suddenly appearing object). Secondly, we integrate planning and control in the same framework by MPC, enhancing the quadrotor's response and disturbance rejection ability in the environments with disturbances (*e.g.*, wind gusts).

The IPC avoids dynamic objects in a reactive manner without tracking and predicting the object movement. Such reactive avoidance is fundamentally limited by the affordable maneuverability of the quadrotor. Detecting, tracking and predicting the dynamic objects movement could break such limit and enable avoidance of higher-speed moving objects. Furthermore, in the current MPC design, the terminal velocity and acceleration are penalized to be close to zero, limiting the quadrotor's ability for high-speed flight (e.g., $\geq 10\,\mathrm{m/s}$). Addressing these issues could be an interesting future work.

Acknowledgment

The authors gratefully acknowledge DJI for fund support and Livox Technology for equipment support during the project. The authors would like to thank Kuntian Dai, Jiafan Xu, Zehuan Yu, and Sheng Xie for their support and the robot team, Critical HIT, for supporting the experiment site. This work was supported by Information Science Academy of China Electronics Technology Group Corporation (ISA CETC) under Project 200010756 and University Grants Committee of Hong Kong General Research Fund (UGC GRF) under Project 17204523.

References

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

[2] 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, 2021.

[3] Yunfan Ren, Fangcheng Zhu, Wenyi Liu, Zhepei Wang, Yi Lin, Fei Gao, and Fu Zhang. Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6332–6339. IEEE, 2022.

[4] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE international conference on robotics and automation*, pages 1478–1483. IEEE, 2011.

[5] Yunfan Ren, Siqi Liang, Fangcheng Zhu, Guozheng Lu, and Fu Zhang. Online whole-body motion planning for quadrotor using multi-resolution search. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1594–1600. IEEE, 2023.

[6] 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.

[7] Yuwei Wu, Ziming Ding, Chao Xu, and Fei Gao. External forces resilient safe motion planning for quadrotor. *IEEE Robotics and Automation Letters*, 6(4):8506–8513, 2021.

[8] Jing Chen, Kunyue Su, and Shaojie Shen. Real-time safe trajectory generation for quadrotor flight in cluttered environments. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1678–1685. IEEE, 2015.

[9] 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.

[10] Youngsang Suh, Jiseock Kang, and Dongjun Lee. A fast and safe motion planning algorithm in cluttered environment using maximally occupying convex space. In *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, pages 173–178. IEEE, 2020.

[11] Weidong Sun, Gao Tang, and Kris Hauser. Fast uav trajectory optimization using bilevel optimization with analytical gradients. *IEEE Transactions on Robotics*, 37(6):2010–2024, 2021.

[12] Ban Wang, Youmin Zhang, and Wei Zhang. Integrated path planning and trajectory tracking control for quadrotor uavs with obstacle avoidance in the presence of environmental and systematic uncertainties: Theory and experiment. *Aerospace Science and Technology*, 120:107277, 2022.

[13] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023.

[14] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[15] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters*, 4(4):4459–4466, 2019.

[16] Zhuozhu Jian, Zihong Yan, Xuanang Lei, Zihong Lu, Bin Lan, Xueqian Wang, and Bin Liang. Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3679–3685. IEEE, 2023.

[17] Michael Neunert, Cédric De Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. Fast nonlinear model predictive control for unified trajectory optimization and tracking. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1398–1404. IEEE, 2016.

[18] Elias Small, Pantelis Sopasakis, Emil Fresk, Panagiotis Patrinos, and George Nikolakopoulos. Aerial navigation in obstructed environments with embedded nonlinear model predictive control. In *2019 18th European Control Conference (ECC)*, pages 3556–3563. IEEE, 2019.

[19] Björn Lindqvist, Sina Sharif Mansouri, Ali-akbar Agha-mohammadi, and George Nikolakopoulos. Nonlinear mpc for collision avoidance and control of uavs with dynamic obstacles. *IEEE robotics and automation letters*, 5(4):6001–6008, 2020.

[20] Charbel Toumieh and Alain Lambert. Decentralized multi-agent planning using model predictive control and time-aware safe corridors. *IEEE Robotics and Automation Letters*, 7(4):11110–11117, 2022.

[21] Yunfan Ren, Yixi Cai, Fangcheng Zhu, Siqi Liang, and Fu Zhang. Rog-map: An efficient robocentric occupancy grid map for large-scene and high-resolution lidar-based motion planning. *arXiv preprint arXiv:2302.14819*, 2023.

[22] Peter Hart, Nils 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.

[23] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

[24] Fanze Kong, Xiyuan Liu, Benxu Tang, Jiarong Lin, Yunfan Ren, Yixi Cai, Fangcheng Zhu, Nan Chen, and Fu Zhang. Marsim: A light-weight point-realistic simulator for lidar-based uavs. *IEEE Robotics and Automation Letters*, 8(5):2954–2961, 2023.

[25] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 2022.