Proceedings of CONF-CDS 2025 Symposium: Application of Machine Learning in Engineering
DOI: 10.54254/2755-2721/2025.AST34483

The Specific Application Method of EGO-Planner in Indoor Scenario

Haoran Sun

Low-altitude Economy, The Hong Kong Polytechnic University, Hong Kong, China
1102554130@qq.com

Abstract. This paper investigates EGO - Planner's application in an indoor UAV simulation environment built with XTDrone and Gazebo. It doesn't redesign the original algorithm but focuses on a practical question: whether an ESDF - free local planner can support autonomous flight in a factory - like indoor scene. In many conventional UAV trajectory planning methods, ESDFs are used to represent obstacle information and assist local trajectory optimization. This approach is useful, yet it may introduce additional computational pressure, especially when the UAV needs real - time local map updates. In this context, EGO - Planner offers a lightweight planning option. It skips explicit ESDF map construction and uses B - spline trajectory optimization to generate feasible local trajectories. To test its performance, this study constructs an indoor factory scenario in Gazebo and runs the experiment via the XTDrone simulation platform. During simulation, the quadcopter UAV first takes off manually and then switches to altitude - hold mode at about 1 meter. After it stabilizes, EGO - Planner is activated to plan local trajectories based on predefined target points and perceived obstacle information. RViz is used to observe the planned trajectory, point cloud data, and environmental changes. The simulation results show that the UAV can complete autonomous flight in the tested indoor scene and generate feasible trajectories in most cases. This result, though not surprising, is useful as it shows EGO - Planner's performance when perception, local mapping, and trajectory planning are integrated in one simulation workflow. Meanwhile, the experiment reveals several limitations. The UAV may get trapped in local minimums, and in some tests, delayed visual feedback and limited computing resources affected trajectory execution, causing unstable flight or crashes. So, the current result is a preliminary simulation validation, not a complete performance evaluation. Future work may focus on real UAV experiments, more stable perception feedback, and further optimization under limited onboard computing resources.

Keywords: EGO-Planner, UAV, motion planning, indoor navigation, B-spline optimization, XTDrone, ROS, Gazebo

1. Introduction

Typically, UAV motion planning consists of global planning and local planning. Global planners (e.g., A* and RRT*) can be used to obtain an initial route between the start position and the target position [1-4]. This path is handy, but many times it is not sufficient for the interior flight. The UAV

DOI: 10.54254/2755-2721/2025.AST34483

may be required to avoid walls, pillars or narrow passages upon running in a factory-like environment. Therefore, it is still necessary to have a local planner to modify the trajectory based on the perception information in real time.

Euclidean Signed Distance Fields (ESDFs) are commonly used by many local planners to describe the distance and gradient information of obstacles [5]. ESDF based approaches can be used to achieve good obstacle avoidance, however the creation and maintenance of ESDF maps may impose additional computational burden on the system. This issue is more likely to manifest itself in small UAVs, which have restricted on-board computing power and/or sensor update rates. In indoor environments with clutter, the planner needs to react fast and maintain smooth enough trajectory for a stable flight.

EGO-Planner offers a trajectory planning method for local planning without the need of an ESDF. It does not explicitly construct ESDF and applies the B-spline trajectory optimization to obtain feasible trajectories [6]. This is ideal for indoor UAV simulations, in which real-time replanning and smoothness of the trajectory are both required. But, still, its performance will be dependent on the quality of perception feedback and available computing resources.

For this study EGO-Planner is evaluated in an indoor environment in a factory-like setting with the XTDrone simulation platform [8, 7]. The virtual environment is created in Gazebo and visualization of the planned trajectory, the point cloud data in its vicinity and mapping results is done in RViz. The UAV first takes off manually, and stabilizes at an altitude of approximately 1 m during the experiment. Then, it is operated in autonomous trajectory planning mode where it moves through predefined target points, in the simulated scene.

This work doesn't introduce any changes to the basic EGO-Planner algorithm. Rather, it concentrates on its feasibility to be used in the tested simulation environment. The results indicate that the planner is able to generally create feasible local trajectories in the factory like scene. Meanwhile, a number of issues are identified. In some cases, the UAV may get stuck in local minimum area, while the lack of visual feedback may also affect the trajectory tracking and cause the drone to become unstable or crash. These results indicate that not only the optimization algorithm used by the planner, but also perception latency and computing resources are of importance.

2. Related works

To generate collision-free paths, the UAV trajectory is represented as a B-spline curve, with the control points $\mathbf{Q}$ used as optimization variables. An initial feasible curve $\mathbf{\Phi}$ is first constructed under basic motion constraints. If collision segments are found during optimization, a global planner such as A* or RRT* is used to provide an auxiliary obstacle-avoiding path $\mathbf{\Gamma}$. For each colliding control point $\mathbf{Q}_i$, anchor points $\boldsymbol{P}_{ij}$ and repulsive direction vectors $\boldsymbol{v}_{ij}$ are defined near obstacle surfaces, guiding the B-spline trajectory away from unsafe regions.

3. Algorithm of EGO-Planner

3.1. Modelling

We employ a uniform B-spline curve Φ to parameterize the trajectory, The property of convex hull in B-spline curves ensures that each segment of the curve is influenced only by p_{b+1} consecutive control points and lies within their convex hull.

DOI: 10.54254/2755-2721/2025.AST34483

Another property of B-spline curves is its reciprocity property. Through these tow characteristics, we can obtain the first, second, and third derivative (velocity V_i, acceleration A_i, jerk J_i) of the trajectory Φ.

$$\mathrm{V}_i = \frac{Q_{i+1}-Q_i}{\triangle t}\,,\ \ \mathrm{A}_i = \frac{V_{i+1}-V_i}{\triangle t}\,,\ J_i = \frac{A_{i+1}-A_i}{\triangle t} \tag{1}$$

Thus, based on the differential flatness features of the drone, it is enough to manipulate the x, y coordinates, as well as yaw angle Ψ of the UAV. All other states can be represented through the algebraic mix of these three variables and their finite - order derivatives, thus cutting down the control quantity we have to design. On the basis of the above, the optimization problem is redefined as.

$$\min_{Q} J = \ \lambda_s J_s + \lambda_c J_c \ + \lambda_d J_d \tag{2}$$

Here, J_s is the smoothness penalty, J_c is the collision penalty, J_d is the dynamic feasibility penalty, and λ represents the weights of the penalty terms.

3.1.1. Smoothness penalty

In B-spline path planning, the smooth penalty is expressed as the time integral of the squared derivatives (e.g., acceleration, jerk) of the trajectory parameters. To achieve smoother motion trajectories, the optimization objective can be formulated to minimize the integral of squared second and third derivatives of the path function Φ, thereby effectively constraining both acceleration and jerk magnitudes throughout the planned trajectory. The formula is presented below.

$$J_s = \sum_{i=1}^{N_c-2} \|A_i\|_2^2 + \sum_{i=1}^{N_c-3} \|J_i\|_2^2, \tag{3}$$

This ensures minimal higher-order derivatives, resulting in a smooth trajectory.

3.1.2. Collision penalty

The collision penalty enforces a safety margin $d_{ij} < s_f$, pushing control points away from obstacles. A twice continuously differentiable penalty function j_c is constructed, with its slope suppressed as d_{ij} decreases. The piecewise function is as follows.

$$\mathrm{j_c}\left(i,j\right) = \begin{cases} 0 & \left(c_{ij} \le 0\right) \\ c_{ij}^3 & \left(0 \le c_{ij} \le s_f\right), \\ 3s_f c_{ij}^2 - 3s_f^2 c_{ij} + s_f^3 & \left(c_{ij} > s_f\right) \end{cases} \tag{4}$$

$$c_{ij} = s_f - d_{ij}$$

DOI: 10.54254/2755-2721/2025.AST34483

The total penalty for the collision term is acquired by adding up the penalties of all control points.

$$J_c = \sum_{i=1}^{N_c} j_c\left(Q_i\right) \tag{5}$$

Unlike conventional ESDF-based approaches that calculate gradients through trilinear interpolation, the derivative of the continuously differentiable quadratic penalty function to Q_i.

$$\frac{\partial j_c}{\partial Q_i} = \sum_{i=1}^{N_c}\sum_{j=1}^{N_p}\begin{cases} 0, c_{ij} \le 0 \\ -3c_{ij}^2, 0 \le c_{ij} \le s_f \\ -6s_f c_{ij} + 3s_f^2, c_{ij} > s_f \end{cases} \tag{6}$$

3.1.3. Feasibility penalty

The enforcement of dynamic feasibility constraints can be achieved through bounding the magnitude of higher-order temporal derivatives across all spatial dimensions of the trajectory. Using $\left|\Phi_r^{(k)}\left(t\right)\right| < \Phi_{r,max}^{(k)}$. Due to the feature of convex hull, the sufficient derivative of the constrained control point is capable of limiting the whole B-spline curve. The penalty function is built with F $(\cdot)$ as the higher-order derivative in each dimension (velocity, acceleration, and acceleration). The formula is presented below.

$$J_d = \sum_{i=1}^{N_c-1}\omega_v F\left(V_i\right) + \sum_{i=1}^{N_c-2}\omega_a F\left(A_i\right) + \sum_{i=1}^{N_c-3}\omega_j F\left(J_i\right) \tag{7}$$

In the formula, $\omega_v, \omega_a, \omega_j$ are weights, and F ($\cdot$) is a twice continuously differentiable function:

$$F\left(C\right) = \sum_{r=x,y,z} f\left(c_r\right), \tag{8}$$

$$f\left(c_r\right) = \begin{cases} a_1 c_r^2 + b_1 c_r + c_1, & c_r < c_1 \\ \left(-\lambda c_m - c_r\right)^3, -c_j < c_r < -\lambda c_m \\ 0, \quad . -\lambda c_m < c_r < \lambda c_m \\ \left(c_r - \lambda c_m\right)^3, \lambda c_m < c_r < c_j \\ a_2 c_r^2 + b_2 c_r + c_2, & c_r > c_j \end{cases} \tag{9}$$

DOI: 10.54254/2755-2721/2025.AST34483

Here, $c_r \in \mathrm{C} \in \{\mathrm{V}_i, \mathrm{A}_i, \mathrm{J}_i\}$, c_m is the derivative limit, c_j splits the quadratic and cubic regions, and $\lambda < 1 - \epsilon$ ensures the constraints are satisfied

4. Numerical optimization

For the objective function:

$$\min_{Q} J = \lambda_s J_s + \lambda_c J_c + \lambda_d J_d \,, \tag{10}$$

With new obstacles being continuously added, the solution environment will constantly change. This demands that the solver rapidly restart and keep solving. Furthermore, the objective function mostly comprises quadratic terms. Employing the Hessian data matrix can accelerate the convergence speed. Nevertheless, gaining a precise Hessian matrix devours a substantial quantity of computational resources. Consequently, we adopt the quasi - Newton method for approximating the Hessian matrix by means of gradient information.

Furthermore, what follows is the deeper refinement of time distribution and trajectory. Relying on the safe trajectory Φ_s acquired in the prior step, a B - spline trajectory Φ_f with uniform time allocation is produced. Afterward, an anisotropic curve fitting approach is utilized to flexibly optimize the control points of Φ_f, guaranteeing adherence to higher - order derivative constraints and keeping derivative characteristics almost the same as those of Φ_s.

Firstly, calculate the exceedance ratio (the proportion of instances exceeding specified limits).

$$r_e = \max\left\{ \left|\mathrm{V}_{i,r}/v_m\right|, \sqrt{\left|\mathrm{A}_{i,r}/a_m\right|}, \sqrt[3]{\left|\mathrm{J}_{i,r}/j_m\right|}, 1 \right\} \tag{11}$$

Consequently, the new time interval for Φ_f is derived.

$$\Delta t\prime = re\Delta t \tag{12}$$

Through solving a closed-form least squares issue, the trajectory Φ_f with an initial time span $\Delta t'$ is produced under given constraints, while maintaining the same geometric shape and number of control points as Φ_s. The optimization process subsequently recomputes both trajectory smoothness metrics and dynamic feasibility constraints, thereby generating an updated objective function formulation.

$$\min_{Q} J' = \lambda_s J_s + \lambda_d J_d + \lambda_f J_f \tag{13}$$

Here, λ_f represents the weight of the fitness term, and J_f is defined as the integral of anisotropic displacement from $\Phi_f(\alpha T')$ to $\Phi_s(\alpha T)$ over ($\alpha \in [\,0,\ 1\,]$), where T and T' denote the durations of trajectories Φ_s and Φ_f respectively. *Given the collision-free nature of the fitted trajectory Φ_s, the optimization framework applies differential weighting: minimal constraints on tangential adjustments (d_a) to enhance smoothness, while enforcing strict limits on normal deviations (d_r) to guarantee obstacle avoidance.*As shown in Figure 1, we employ a spherical metric to ensure displacements on the same spherical surface incur identical penalties.

DOI: 10.54254/2755-2721/2025.AST34483

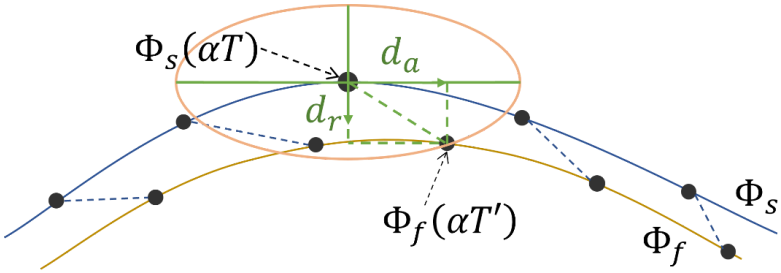

Figure 1. Displacement diagram of diameter

The ellipsoid used for $\Phi_f(\alpha T')$ is an ellipse centered at $\Phi_s(\alpha T')$, obtained by rotating it around its major axis or tangent by a certain angle. Then, the axial displacement d_a and radial displacement d_r are followed by.

$$d_a\left(\alpha T'\right) = \left(\Phi_f\left(\alpha T'\right) - \Phi_s\left(\alpha T'\right)\right) \cdot \frac{\dot{\Phi}_s(\alpha T')}{\|\dot{\Phi}_s(\alpha T')\|}, \tag{14}$$

$$d_r\left(\alpha T'\right) = \| \left(\Phi_f\left(\alpha T'\right) - \Phi_s\left(\alpha T'\right)\right) \times \frac{\dot{\Phi}_s(\alpha T')}{\|\dot{\Phi}_s(\alpha T')\|} \|, \tag{15}$$

Thus, the fitting term is defined as.

$$J_f = \int_0^1 \left[\frac{d_a\left(\alpha T'\right)^2}{a^2} + \frac{d_r\left(\alpha T'\right)^2}{b^2} \right] \mathrm{d}\alpha, \tag{16}$$

Where a and b are the semi-major axis and semi-minor axis of the ellipse, respectively.

5. Simulation

To verify the correctness of the trajectory planning algorithm and reduce costs, this paper adopted the XTDrone simulation platform, added a simulated factory map environment, and used the ROS Visualization (Rviz) and Gazebo simulation interfaces to conduct simulation experiments on the trajectory planning of quadcopter drones, thereby verifying the correctness of the algorithm [9, 10].

5.1. Construction of a simulation platform based on the ROS system

The Robot Operating System (ROS) is a flexible software framework for developing robotics applications [10]. It's a set of middleware built on top of existing operating systems and is particularly widely used in multi-robot systems, autonomous mobile robots, unmanned vehicles, and service robots. This article is developed based on ROS Melodic under Ubuntu 18.04 and uses the XTDrone simulation platform for algorithm testing.

DOI: 10.54254/2755-2721/2025.AST34483

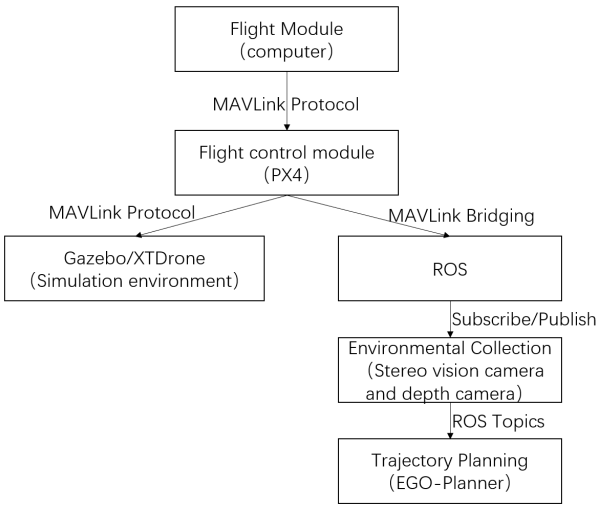

Figure 2. Simulation operation flow chart

5.2. Simulation experiment

To examine whether the EGO-Planner algorithm can work effectively in a practical simulation setting, this study builds a factory-like indoor environment in Gazebo and conducts flight experiments through the XTDrone simulation platform. The purpose of this experiment is relatively direct: to observe whether the planner can generate feasible trajectories for a quadcopter UAV in a constrained indoor scenario. In this hardware-in-the-loop simulation, both the quadcopter model and the surrounding environmental model are first constructed in Gazebo. After the planning algorithm is launched, the real-time trajectory planning process is visualized in RViz, which makes the UAV's motion and local planning behavior easier to observe.

The experimental procedure is carried out step by step. First, the Gazebo indoor simulation environment is started. Then, the RViz visualization interface is opened to monitor the real-time changes around the UAV. The UAV is manually controlled during takeoff. After reaching a stable height, it switches to altitude-hold mode and maintains a flight height of approximately 1 meter. This setting is simple, but it provides a relatively stable basis for observing the performance of the planner. After that, the autonomous trajectory planning algorithm is activated, and the planned flight path can be viewed continuously in RViz.

Once the UAV enters autonomous flight mode, it begins to plan its trajectory according to predefined target points. From the front-facing perception view, the UAV constructs a local map based on the information collected by its onboard perception system. The planning algorithm then uses this local map to generate feasible trajectories and guide the UAV through the indoor environment. Interestingly, this process shows how the perception module and the planning module are connected in the simulation workflow. The UAV is not simply following a fixed route; instead, it adjusts its movement according to the locally perceived environment. The indoor simulation scene and the corresponding real-time point cloud visualization are shown in Fig. 3 and Fig. 4.

DOI: 10.54254/2755-2721/2025.AST34483

Figure 3. Indoor simulation flight of unmanned aircraft

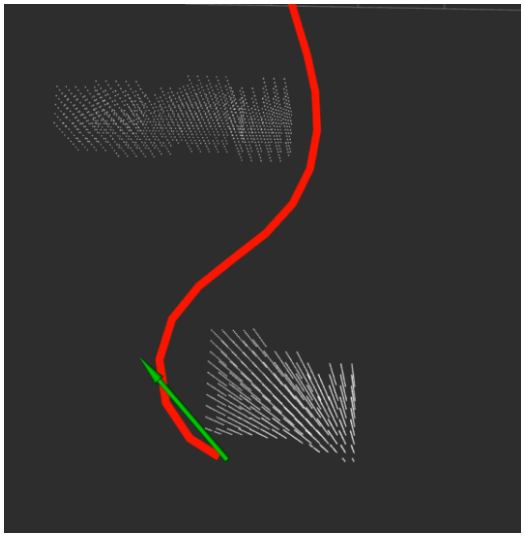

Figure 4. Real-time point cloud image

6. Conclusion

In this paper, EGO-Planner was tested in an indoor UAV simulation environment built with Gazebo and XTDrone. The test environment was a factory-like scene, and the flight experiment was carried out with a quadcopter UAV. The UAV performed manual takeoff first. After it reached a height of approximately 1 m, it was switched to autonomous trajectory planning mode. The whole process was monitored using RViz to visualize the planned trajectory, local point cloud information, and mapping results.

In the tested scenario, EGO-Planner was able to generate feasible local trajectories for the UAV. The vehicle could follow predefined target points and complete autonomous flight in the virtual indoor environment. This result suggests that the planner can be used for indoor UAV simulation, especially when frequent ESDF construction may increase the computational load of the system. Still, this result should be understood within the current simulation setting.

Several problems were also observed during the experiment. The UAV sometimes became trapped in local minimum regions, and delayed visual feedback affected trajectory execution in some cases. When the local perception update was not fast enough, the UAV could become unstable or even collide with obstacles. These results show that the performance of EGO-Planner depends not

DOI: 10.54254/2755-2721/2025.AST34483

only on the planning algorithm itself, but also on perception feedback and available computing resources.

More research is required before more definitive conclusions can be drawn. Real UAV flight tests should be carried out to examine whether the same planning behavior can be maintained outside the simulation environment. On top of this, there should be quantitative measures like planning time, collision rate, length of trajectory, and success rate of flight in the future experiments. The measurements would give better indications to support the assessment of the stability of EGO-Planner in more complex indoor environments.

References

[1] Kiran B R, Sobh I, Talpaert V, et al. Deep reinforcement learning for autonomous driving: A survey [J]. IEEE transactions on intelligent transportation systems, 2021, 23(6): 4909-4926.

[2] Yao P F, Geng B, Yang M, et al. Research on technology of autonomous inspection system for UAV based on improved YOLOv4 [C]//2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE). IEEE, 2020: 664-668.

[3] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths [J]. IEEE transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[4] Noreen I, Khan A, Habib Z. A comparison of RRT, RRT* and RRT*-smart path planning algorithms [J]. International Journal of Computer Science and Network Security (IJCSNS), 2016, 16(10): 20.

[5] Jones M W, Baerentzen J A, Sramek M. 3D distance fields: A survey of techniques and applications [J]. IEEE Transactions on visualization and Computer Graphics, 2006, 12(4): 581-599.

[6] Zhou X, Wang Z, Ye H, et al. Ego-planner: An esdf-free gradient-based local planner for quadrotors [J]. IEEE Robotics and Automation Letters, 2020, 6(2): 478-485.

[7] Xiao K, Tan S, Wang G, et al. XTDrone: A customizable multi-rotor UAVs simulation platform [C]//2020 4th International Conference on Robotics and Automation Sciences (ICRAS). IEEE, 2020: 55-61.

[8] Gordon W J, Riesenfeld R F. B-spline curves and surfaces [M]//Computer aided geometric design. Academic Press, 1974: 95-126.

[9] Kam H R, Lee S H, Park T, et al. Rviz: a toolkit for real domain data visualization [J]. Telecommunication Systems, 2015, 60(2): 337-345.

[10] Koenig N, Howard A. Design and use paradigms for gazebo, an open-source multi-robot simulator [C]//2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566). Ieee, 2004, 3: 2149-2154.