FELP:Fast and Effective Autonomous Flight on Large-Scale and Cluttered Environments Based on Unified Linear Parametric Map

Hongyu Nie, Xingyu Li, Xu Liu, Decai Li, and Yuqing He

***Abstract*—Current AAV autonomous flights exhibit efficient performance in both indoor and field environments. However, they often face significant challenges in large-scale and cluttered environments, where the vast amount of captured data can lead to computation and storage bottlenecks. Additionally, the existing gradient-based planning methods depend on appropriate resolutions to adapt to different scenarios. In this letter, we present FELP, a fast and effective autonomous flight system for large-scale and cluttered environments based on the unified linear parametric map. It can enhance the adaptability of planners to diverse environments. First, by the random mapping method (RMM), the original irregular points in low-dimensional space are mapped into the high-dimensional space, where the points are approximately linearly separable or distributed. Leveraging the features of this mapping space, we can quickly obtain the occupancy state and Euclidean distance (the distance to the nearest obstacle) rather than relying on a large number of queries and repeated iterations. Then we learn a unified linear parametric model about grid maps and ESDF maps. Based on the linear parametric model, path searching is quickly executed in the front-end. Unlike traditional methods that compute the ESDF through interpolation, the closed-form ESDF can be solved efficiently, enabling real-time online trajectory optimization in the back-end. Compared to EGO-Planner, FELP reduces the mapping time by 68% and the planning time by 29%. Simulation and real-world experiments are conducted to verify their comprehensive performance compared to typical methods and state-of-the-art methods.**

***Index Terms*—Autonomous navigation, occupancy grid map, ESDF, trajectory optimization.**

Received 9 January 2025; accepted 22 May 2025. Date of publication 1 August 2025; date of current version 12 August 2025. This article was recommended for publication by Associate Editor M. Greeff and Editor G. Loianno upon evaluation of the reviewers' comments. This work was not supported by any funding. *(Hongyu Nie and Xingyu Li contributed equally to this work.) (Corresponding author: Hongyu Nie.)*

Hongyu Nie is with the School of Information Science and Engineering, Shenyang University of Technology, Shenyang 110870, China (e-mail: niehongyu@sia.cn).

Xingyu Li is with the College of Information Science and Engineering and the State Key Laboratory of Synthetic Automation for Process Industries, Northeastern University, Shenyang 110819, China, and also with the Woozoom Technology Company Ltd., Shenyang 110179, China (e-mail: 2300902@stu.neu.edu.cn).

Xu Liu, Decai Li, and Yuqing He are with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: liuxu1@sia.cn; lidecai@sia.cn; heyuqing@sia.cn).

Digital Object Identifier 10.1109/LRA.2025.3595028

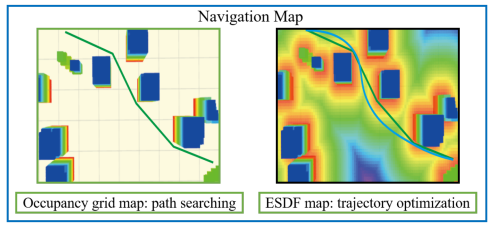


Fig. 1. The whole framework of autonomous flight includes two parts: the front-end, which performs path searching on the grid map, and the back-end, which handles trajectory optimization on the ESDF map.

I. Introduction

AUTONOMOUS aerial vehicles (AAVs), especially quadrotors, have become widely utilized in various applications such as search-and-rescue [1], target tracking [2] and map reconstruction [3]. This widespread use is primarily attributed to their compact size and high-speed flight capabilities. However, building suitable maps to enable quadrotors' autonomous navigation on large-scale and cluttered scenarios presents significant challenges due to limited onboard computational resources. As is seen in Fig. 1, grid maps and ESDF maps are commonly used for path searching and trajectory optimization, respectively. A desirable map for autonomous navigation must accommodate a high-quality environmental model and ensure good performance in motion planning.

Occupancy grid map can distinguish between occupied, free, and unknown regions of the environment, which is crucial for motion planning. Despite its good performance in A^* path search [4], and autonomous exploration [5], it poses challenges due to high computation and storage consumption in large-scale scenes. Furthermore, to detect small obstacles in the cluttered scenes, an occupancy grid map with sufficiently high resolution significantly increases computational complexity. The main drawback of these maps is that they will face challenges in balancing memory consumption and mapping accuracy, especially for large-scale and unstructured environments.

ESDF map: are primarily used for gradient-based trajectory optimization in the back-end [6] [7]. Based on the global occupancy map, the **ESDF map** (local map) is constructed based on Euclidean distance [8] [9], representing the distance between the trajectory and its nearest obstacle. Euclidean distance and gradient values are derived by interpolation [10] [11]. Interpolation necessarily depends on suitable resolution. A finer resolution enhances the accuracy of distance and gradient calculations. High-precision ESDF maps facilitate trajectory optimization to

safely avoid obstacles, and it performs better in complex scenes. However, for lightweight multi-rotor systems with limited resources, it is necessary to balance computing efficiency and accuracy, especially for large-scale and complex environments.

To tackle large-scale challenges, we propose a novel linear parametric map for AAV navigation, **motivated** by the need to enhance planning efficiency in resource-constrained AAVs. This map reduces computational demands by avoiding iterative calculations and storing only essential model parameters. Furthermore, our adaptive ESDF map requires no resolution adjustments for various environments, streamlining the planning process and reducing computational overhead.

The contributions of this paper are summarized as follows:

- In occupancy grid mapping, we use RMM to transform it into a classification task to learn a linear parametric model. The novel linear parametric model is continuous and memory-saving, which can accelerate the kinodynamic path searching efficiency in the front-end.
- Based on the above occupied grid map model, the gradient of the occupancy degree can be obtained to reduce a large number of queries and improve the efficiency of calculating Euclidean distance.
- In ESDF mapping, we use RMM to transform it into a regression task to learn a linear parametric model. Unlike traditional ESDF maps that relies on interpolation by appropriate resolution, a closed-form ESDF map is obtained by the linear parameter model. The continuous and arbitrary resolution ESDF map is effectively used for trajectory optimization in the back-end to adapt to different scenarios.

II. Related Work

A. Occupancy Grid Map

Occupancy grid maps are a common and classic map type for motion planning in [4], [7], [13]. Representation form and update approaches of occupancy grid map are important factors for robotics systems in motion planning. Based on the assumed representations of the map model, occupancy grid maps are classified into two types: explicit representation and implicit representation.

Explicit representation assumes priori discretization of the space into voxel grids, and then estimates the occupancy state (free or occupied) by the Bayesian-based [14] [15] [16]. The occupancy grid map can be classified into three categories based on its data structure: grid-based map [5], tree-based map [17], and hash table-based [18]. Although these maps are simple and efficient, the main drawback of these maps is that they will face challenges in balancing memory consumption and mapping accuracy, especially for large-scale and unstructured environments. To address these challenges, recent research has focused on developing more efficient mapping techniques for AAVs. [19] proposes a method for converting 3D voxel maps into 2D occupancy maps, which is particularly beneficial for efficient path planning for both AAVs and ground robots.

Implicit representation is a continuous map, which can capture the relationship between spatial locality and model environments through continuous occupancy distribution. Gaussian process regression is proposed to acquire occupancy state (free or occupied), which has been extended to 3-D scenes [20]. While the Gaussian process is capable of capturing the correlations among grid cells through a covariance matrix, they either suffer from $\mathcal{O}(n^3)$ computational complexity with respect to the number of measurement data or require large amounts of memory, making them impractical for online path planning, especially in large-scale environments. Besides, some novel learning-based methods use occupancy networks [21], neural radiance fields [22], and DeepSDF [22] to occupancy mapping. Typically, these methods needs a large amount of training data and time to obtain a high-quality map.

B. ESDF Map

ESDF maps have attracted widespread interest in robotics motion planning, and have been applied since [23]. [8] proposed an envelope algorithm to pre-compute an ESDF map so that the query process can be executed in O(n) time. Unfortunately, the computation cost is extremely high when maintaining a global ESDF map in large-scale environment. To reduce the time on computing ESDF, [24] use GPU programming techniques to quickly construct ESDF map for motion planning. Recently, [25] used Neural Radiance Fields to construct an ESDF map. Additionally, [26] evaluated the use of Neural Euclidean Signed Distance Fields for distance-aware local path planning, showing promising results in 3D local path planning applications. The 'Voxblox' method and FIESTA are proposed in [10] [11] to build the ESDF map incrementally. The Euclidean distance and the gradient values are obtained by interpolation. These gradient-based planning methods can use the ESDF map for obstacle avoidance [4] [27] [7]. Furthermore, [28] proposed Robo-centric ESDF, a fast and accurate whole-body collision evaluation tool that can be applied to any-shape robotic planning, enhancing the capabilities of ESDF-based methods in complex environments.

In summary, in this letter, we utilize a unified paradigm of linear parameter models to construct occupancy grid maps and ESDF maps, thereby enhancing planning efficiency for resource-constrained AAVs.

III. Unified Linear Parametric Map

Occupancy grid maps and ESDF maps are commonly used for path searching and trajectory optimization, respectively. In occupancy grid mapping, the key is to estimate the occupancy state. Traditional methods, such as Bayesian methods, need to store a large amount of predicted data for the evaluation of occupancy states. To obtain such a discrete map and then access the occupancy state for path searching brings challenges to storage and computing, especially for large-scale environments. In ESDF mapping, it is necessary to calculate and save the Euclidean distance of every center voxel from the global grid map. For the distance and gradient values of random point this local map, which are calculated by interpolation through appropriate resolution. However, it still needs to find a balance between efficiency and accuracy. To solve the above problems, we propose a lightweight and continuous linear parametric model for occupancy mapping and ESDF mapping. Firstly, in online autonomous navigation in the field environment, the AAV obtains a series of irregular point cloud datasets $\boldsymbol{D} = \{x_i, t_i\}_{i=1}^{L}$, the data in the low-dimensional space of the origin are irregularly separable. We use RMM to project the irregular

points into a high-dimensional space where they become approximately linearly separable. This transformation is crucial because it allows us to represent the environment with a simple linear model, significantly reducing the computational cost of subsequent occupancy and distance estimations. The random mapping matrix S is obtained as follows:

$$\boldsymbol{S} = \boldsymbol{G}(\boldsymbol{W}\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{W}$ is defined as a $M \times N$ matrix. In particular, all elements of $\boldsymbol{W}$ are generated from a uniform distribution between [-1, 1]. M is the random mapping dimension, which is typically determined by trials. Here, S is defined as the $M \times L$ random mapping matrix. $G(\cdot)$ is the sine function, $G(\cdot) = \sin(\cdot)$. Assuming M is large enough, the mapping matrix $G(\boldsymbol{W}x)$ can map the original irregular points into a high-dimensional space set, where the points are approximately linearly separable or distributed. To uphold this assumption, the RMM needs sufficient capacity, including a well-chosen activation function and dimension M, ample and representative training data, effective feature extraction for the task, and acceptable sensor noise levels to ensure reliability.

In an occupancy grid map, the discrete target vector *t* represents the occupancy state (0 for free, 1 for occupied). We use RMM to transform this into a classification task to learn a linear parametric model. In an ESDF map, the continuous target vector *t* is the Euclidean distance. By treating the ESDF modeling as a regression problem, we attempt to build a linear parametric model between position and Euclidean distance.

In this letter, the linear classifier (grid map) or regression model (ESDF map) $f(x) = \boldsymbol{\beta^T} g(\boldsymbol{W}x)$ is learned online for fast autonomous flight. From the practical perspective, M can be much less than L to reduce the computation complexity. It should be noted that the weight matrix is generated at random from a uniform distribution, thus avoiding the process of training and constructing the kernel. So, it will have better real-time performance.

A. Learning Occupancy Grid Map

In the occupancy grid map, we use RMM to transform it into a classification task to learn a linear parametric model as follows:

$$f = f(x, \beta) = \boldsymbol{\beta}^T \boldsymbol{g}(\boldsymbol{W}\boldsymbol{x}) + b = \boldsymbol{\beta}^T s + b = \boldsymbol{\beta}^T s \tag{2}$$

To simplify the notations, we expand $\boldsymbol{\beta}$ with b and s with 1. When autonomous flight in large-scale environments, we learn a linear classifier model $f(x, \beta) = \boldsymbol{\beta^T} \boldsymbol{g}(\boldsymbol{W}x)$ to update the occupancy state. To solve $\boldsymbol{\beta}$, we minimize the objective function with L2-norm regularizer, as following:

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{L}(t_i - y_i)^2 + \frac{1}{2}\alpha||\beta||_2^2 \tag{3}$$

where α is the regularizer, y_i is the predicted occupancy state value (0 or 1). To fast real-time update, the stochastic gradient descent (SGD) method can be used to update β as follows :

$$\beta_{t+1} = \beta_t - \frac{\eta_t}{2}\left(\frac{\partial((t_i - \beta^T s_i)^2 + \alpha||\beta||_2^2)}{\partial\beta}\right) \tag{4}$$

The learning rate is denoted by η_t. The time complexity of Stochastic Gradient Descent (SGD) is $\mathcal{O}(kL\bar{p})$, where k represents the iteration number, L represents the number of cloud points, and $\bar{p}$ represents the mean number of occupancy points.

Algorithm 1: Kinodynamic A* With Linear Parametric Map.

Input: D, $g(v)$, M, α, β, s, g, heuristic
Output: $\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^{n+1}$

```
x, T ← getLocalPointCloudData(D)
W, b ← random(M, size(x)[1]), random(M, 1)
S ← g(Wx + b)
β ← SolveLinearModel(S, T, α)
occuState ← β^T g(Wx + b)
gridMap ← GenerateGridMap(occuState)
Initialize
while not PathFound(s) do
    current ← SelectCurrentNode()
    if ReachGoal(current, g) ∨ AnalyticExpand(current) then
        Prune(current)
        return RetrievePath()
    end if
    for neighbor ∈ GetNeighbors(current, gridMap) do
        if occuState(neighbor) == 1 then
            if checkKinoDynamicFeasible(neighbor) then
                continue
            end if
        end if
        ProcessNeighbor(neighbor, heuristic)
    end for
end while
return P
```

Consequently, the SGD algorithm efficiently updates the map in linear time. Based on the parametric map, kinodynamic A* is performed quickly in Algorithm 1.

The algorithm combines A* path planning with linear parametric mapping to enhance the path search process by efficiently computing and updating occupancy states. The process starts with preprocessing the dataset D, where the **getLocalPointCloudData** function extracts spatial and occupancy information from locally perceived point cloud data. As a local perception database, D supports the training of the feature set x and target set T. The algorithm initializes a random mapping matrix W and bias vector b, applies a linear transformation $Wx + b$ to the feature set, and derives state representations (S) using a nonlinear activation function g. A linear model is then solved to calculate the model parameter (β), which are used to compute the occupancy states ($occuState$) of nodes. Finally, a linear parametric map is constructed to model the environment for path searching.

After initialization, the algorithm proceeds to path search. Using the **Initialize** function, it assigns the computed $occuState$ to configure the initial node's state, ensuring accurate initialization. During planning, the algorithm checks if the target is reached or if the path is expanded. If so, it calls **Prune** to remove redundant paths and returns the final planned path. Otherwise, it identifies neighboring nodes using **GetNeighbors** and evaluates their safety by verifying $occuState(\text{neighbor}) = 1$. Dynamic feasibility checks are conducted through **checkKinoDynamicFeasible** to ensure compliance with physical and dynamic constraints. For valid neighbors, the path costs are calculated via a heuristic function [4], and the path data is updated using

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

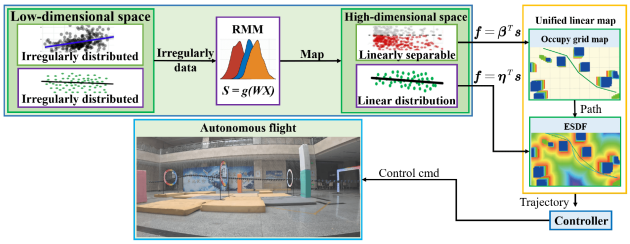


Fig. 2. The data in the original space is irregularly distributed. By following our previous work in [12], it can be mapped to a high-dimensional space, which is either linearly distributed or separated. We applied RMM to the occupancy grid map and ESDF map in a unified paradigm, respectively. Then, path searching and trajectory optimization are executed on the novel linear parametric maps.

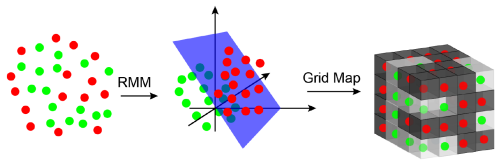


Fig. 3. By the RMM method, the original point clouds are mapped to a high-dimensional space, and the red and green points are linearly separated by the blue hyperplane. Red points falling into a voxel are marked as occupied, while green points falling into a voxel are marked as free.

ProcessNeighbor. If a path is successfully identified, the algorithm returns the path set P. By integrating global search with efficient environmental modeling, this method enables fast and effective path planning in large-scale environments.

Algorithm 2: Euclidean Distance.

Input: $\boldsymbol{OccState}, \beta, \boldsymbol{ESDF_{max}}, \boldsymbol{ESDF_{min}}, \boldsymbol{x}$
Output: Distance field: $\boldsymbol{D}$
Function updatedist($\boldsymbol{OccState}, \boldsymbol{x}, \boldsymbol{\beta}$)
  for $x \in [\boldsymbol{ESDF_{min}(x)}, \boldsymbol{ESDF_{max}(x)}]$ **do**
    for $y \in [\boldsymbol{ESDF_{min}(y)}, \boldsymbol{ESDF_{max}(y)}]$ **do**
      Initialize case1, case2, case3 as false
      if $\frac{\partial(\boldsymbol{\beta G(Wx)})}{\partial(\boldsymbol{z})} == 0$ **and** $\boldsymbol{OccState}(0) == 1$ **then**
        $D(z) = 0, case2 = true$
      else if $\frac{\partial(\boldsymbol{\beta G(Wx)})}{\partial(\boldsymbol{z})} == 0$ **and** $\boldsymbol{OccState}(0) == 0$ **then**
        $D(z) = \infty, case1 = true$
      else if $\frac{\partial(\boldsymbol{\beta G(Wx)})}{\partial(\boldsymbol{z})} \neq 0$ **then**
        $D(z) = \text{FILLESDF}(z, 0), case3 = true$
      end if
    end for
  end for
  for $x \in [\boldsymbol{ESDF_{min}(x)}, \boldsymbol{ESDF_{max}(x)}]$ **do**
    for $z \in [\boldsymbol{ESDF_{min}(z)}, \boldsymbol{ESDF_{max}(z)}]$ **do**
      $D(y, z) = \text{FILLESDF}(y, D(z))$
    end for
  end for
  for $y \in [\boldsymbol{ESDF_{min}(y)}, \boldsymbol{ESDF_{max}(y)}]$ **do**
    for $z \in [\boldsymbol{ESDF_{min}(z)}, \boldsymbol{ESDF_{max}(z)}]$ **do**
      $D(x, y, z) = \text{FILLESDF}(x, D(y, z))$
    end for
  end for
End Function

B. Learning ESDF Map

In ESDF map, each voxel stores the occupancy state and the distance to the nearest obstacle, as illustrated in Fig. 3. Based the linear parametric model (grid map), our goal is to find the nearest obstacle with respect to each voxel(x, y, z). Taking 1-D as an example, the initial ESDF values of occupied voxels are set to 0, while those of free voxels are set to inf. Each point within the same column must be visited and accessed to calculate the distance to the nearest obstacle within that column. The ESDF map requires sweeping each column to obtain ESDF values. The process begins from each occupied voxel with a distance of 0 and increments monotonically in free voxels. This procedure can be described as follows:

$$\boldsymbol{G}(x, y, z) = \min_{k \in rows} \{|k - z| \boldsymbol{D}(x, y, k)\} \tag{5}$$

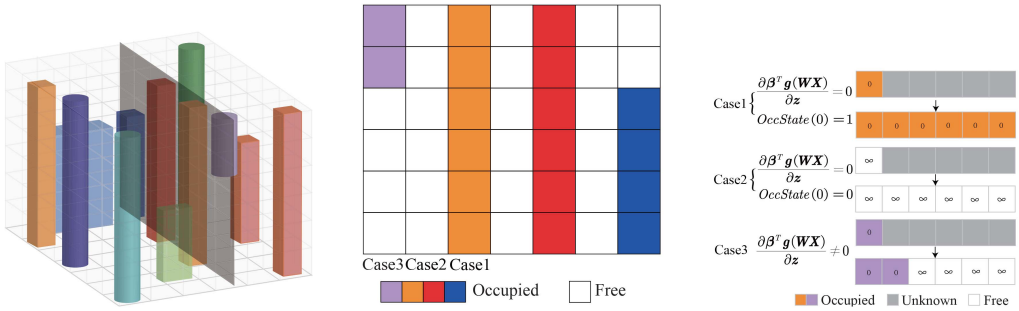
where $\boldsymbol{G}(x, y, z)$ is the distance values of each voxel when scanning the first dimension. Scan begins on the Z-X-plane. Perform the scanning process along the X-axis while keeping the values on the Y-axis and Z-axis unchanged. Based on different obstacle situations on the grid map, we divided them into three cases to calculate the Euclidean distance. As shown in Fig. 4, in case 1, all the points on this column are occupied. Unfortunately, the occupancy status of each voxel on this column cannot be obtained if we don't visit the voxel in advance. Traditionally, we need to access the occupancy status of each voxel and calculate the distance between each voxel to the nearest obstacle. To reduce a large amount of access and repetitive iterative calculations, we use a linear parametric model to obtain the occupancy status of batch voxel in advance. Firstly, based on the model in (2), batch voxel $\{\boldsymbol{z}\} = \{z_1, z_2, z_3, \ldots, z_l\}$ of occupancy status can be directly obtained using the following formula:

$$\nabla f_{\boldsymbol{z}} = \frac{\partial \boldsymbol{\beta}^T \boldsymbol{g}(\boldsymbol{Wx})}{\partial \boldsymbol{z}} \tag{6}$$

$\nabla f_{\boldsymbol{z}}$ means the gradient of the occupancy degree. In case 1, the gradient of the occupancy degree $\nabla f_{\boldsymbol{z}} = 0$, which means the occupancy state of all voxels on this column is the same. If the first visited voxel is free, all voxels on this column are free, then the Euclidean distance values are set to ∞. Similarly, in case 2, if the first visited voxel is occupied, all voxels on this column are occupied, then the Euclidean distance values are set to 0. In case 3, the gradient of the occupancy degree $\nabla f_{\boldsymbol{z}} \neq 0$, which means both free and occupied voxels exist on this column. We need to visit all voxels in this column to calculate the nearest distance to each voxel. As seen in Fig. 5, the distance function of each voxel can be represented as a parabola. Essentially, it is to find the lower envelope of the union of all parabolas, which can be formulated as

$$\boldsymbol{D}(x, y, z) = \min_{i \in cols} \left\{(z - i)^2\right\} + G(x, y, i)^2 \tag{7}$$

where $\boldsymbol{D}(x, y, z)$ is the 3-D Euclidean distance values, $\boldsymbol{G}(x, y, i)$ is a constant value and it is already calculated by FILLESDF() in [29]. The detailed process of calculating the 3-D Euclidean distance is described in Algorithm 2.



(a) The entire global grid map and the distribution of obstacles in the environment.

(b) The cut-plane perspective helps to better understand the occupied and free voxels on the map.

(c) Three different cases for computing Euclidean distance.

Fig. 4. Example of occupancy state determination and Euclidean distance calculation for three cases in the map.

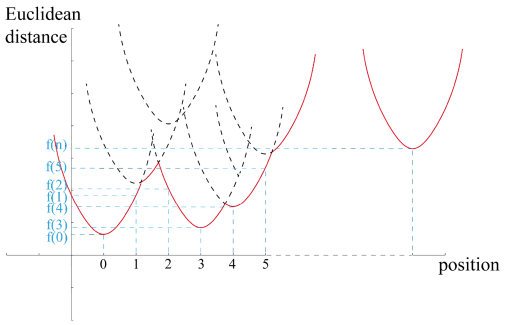


Fig. 5. An illustration of distance functions in FILLESDF(). Each voxel needs to find the lower envelope of the parabola to calculate its distance.

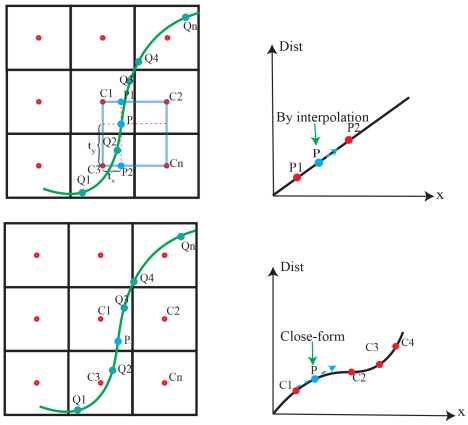


Fig. 6. The green B-spline trajectory is the trajectory to be optimized, and each red dot stores the Euclidean distance of the grid center point. The Euclidean distance and gradient values of p are calculated by interpolation and the closed-form method.

IV. Closed-Form ESDF

To obtain the Euclidean distance and gradient for point p on the ESDF map, we use a 2D example where each voxel stores the distance at its center. As shown in Fig. 6, $C1 - C_n$ are voxel centers, and the ESDF values for $p1$ and $p2$ are computed via linear interpolation function $f()$ using Zhou et al.'s 3D function [4].

$$d_{p_1}=f(c_1,c_2,t_x,\text{resolution}),\quad d_{p_2}=f(c_3,c_4,t_y,\text{resolution}) \tag{8}$$

$$d_p = f(p_1,p_2,t_y,\text{resolution}) \tag{9}$$

$$\nabla f_{d_{p_1}} = f(c_1,c_2,t_y,\text{resolution}) \tag{10}$$

$$\nabla f_{d_{p_2}} = f(c_3,c_4,t_x,\text{resolution}) \tag{11}$$

$$\nabla f_{d_p} = f(\nabla f_{p_1},\nabla f_{p_2},t_y,\text{resolution}) \tag{12}$$

d_{p_1} is the distance value of p_1, d_{p_2} is the distance value of p_2, $\nabla f_{d_{p_1}}$ is the gradient value of p_1, $\nabla f_{d_{p_2}}$ is the gradient value of p_2. It can be seen from (9) and (12) that the distance value d_p of point p is calculated via interpolation between p_1 and p_2. Gradient value ∇f_{d_p} of point p is also calculated via interpolation between $f_{d_{p_1}}$ and $f_{d_{p_2}}$. Both the distance and gradient values of point p rely on resolution. Undoubtedly, the resolution size will affect the accuracy of the ESDF map. To solve this problem, we attempt to build a linear parametric model between position $\boldsymbol{x}$ and Euclidean distance $\boldsymbol{D}$ as follows:

$$D = \boldsymbol{\eta}^T \boldsymbol{g}(\boldsymbol{W}\boldsymbol{x}) \tag{13}$$

The $\boldsymbol{\eta}$ solution form can refer to (4). The gradient values can be directly derived by the closed-form function as follows:

$$\nabla f_{\boldsymbol{x}} = \frac{\partial D}{\partial \boldsymbol{x}} = \frac{\partial\left(\boldsymbol{\eta}^T \boldsymbol{G}\left(\boldsymbol{W}\boldsymbol{x}\right)\right)}{\partial \boldsymbol{x}} \tag{14}$$

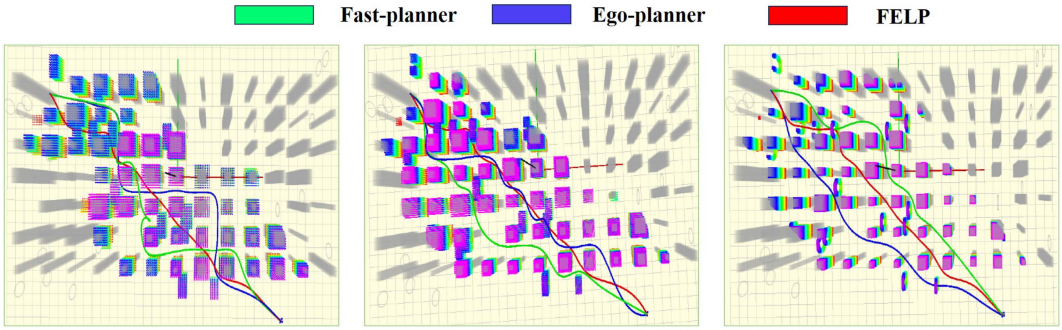
Unlike traditional methods that require extensive interpolation and repeated iterations, we can obtain the distance and gradient values via the linear parametric model. The parametric model can be regarded as an implicit representation of the ESDF map. For an arbitrary point $\boldsymbol{p} = \{Q_1, Q_2, Q_3, \ldots, Q_n\}$ on the map in Fig. 6, the distance and gradient values can be obtained by substituting the optimal path point p into the parametric model (13), and (14) respectively. This eliminates the need for computationally expensive interpolation, making it particularly beneficial in large-scale environments where traditional ESDF methods suffer from resolution-dependent accuracy and increased computational complexity.

V. Trajectory Optimization

In this paper, we adopt B-spline [4] to conduct gradient-based trajectory optimization. For a p_b-degree B-spline trajectory defined by $N+1$ control points $\{Q_0, Q_1, \ldots, Q_N\}$, the overall cost function is defined as:

$$f_{total} = \lambda_s f_s + \lambda_c f_c + \lambda_d (f_v + f_a)$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) Resolution = 0.2 (b) Resolution = 0.15 (c) Resolution = 0.1

Fig. 7. Comparison of fully autonomous flight experiments at different resolutions. The red trajectory is generated by FELP, the green trajectory is generated by Fast-Planner, and the blue trajectory is generated by EGO-Planner.

$$= \lambda_s \sum_{i=p_b-1}^{N-p_b+1} \|\mathbf{Q}_{i+1} - 2\mathbf{Q}_i + \mathbf{Q}_{i-1}\|^2 \qquad (15)$$

$$+ \lambda_c \sum_{i=p_b}^{N-p_b} \mathbf{F}(d(\mathbf{q}_i), d_{\min})$$

$$+ \lambda_d \sum_{\{x,y,z\}} \left\{ \sum_{i=p_b-1}^{N-p_b} \mathbf{F}(v_{\max}^2, \tilde{q}_{i,\mu}^2) + \sum_{i=p_b-2}^{N-p_b} \mathbf{F}(a_{\max}^2, \tilde{q}_{i,\mu}^2) \right\}$$

where f_s and f_c are smoothness and collision cost. f_v and f_A are dynamic limits on velocity and acceleration. λ_s, λ_c and λ_d trade off the smoothness, safety and dynamic feasibility.

Here $F()$ is a penalty function for general variables as follows:

$$F(x,y) = \begin{cases} (x-y)^2, & x \le y \\ 0, & x > y \end{cases} \qquad (16)$$

Collision Penalty: The trajectory is optimized on the local ESDF map to avoid obstacles safely.

$$F_c(\mathbf{D}(\mathbf{Q_i})) = \begin{cases} (\mathbf{D}(\mathbf{Q_i}) - S_f)^2 & \mathbf{D}(\mathbf{Q_i}) \le S_f \\ 0 & \mathbf{D}(\mathbf{Q_i}) > S_f \end{cases} \qquad (17)$$

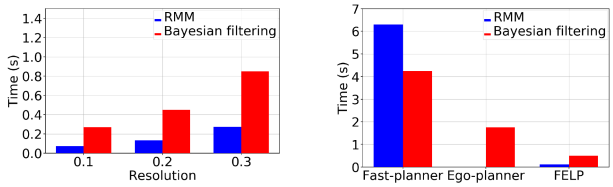
where $D(Q_i)$ is the Euclidean distance obtained from (13), S_f is the safe distance between an obstacle and the trajectory. Unlike traditional ESDF maps that rely on interpolation, we obtain distance and gradient values from a closed-form ESDF map. We obtain gradient values by directly computing the derivative of F_c with respect to Q_i.

$$f_c = \frac{dF_c(\mathbf{D}(\mathbf{Q_i}))}{dQ_i} = \begin{cases} 2(\mathbf{D}(\mathbf{Q_i}) - S_f)\dot{\mathbf{D}}(\mathbf{Q_i}) & \mathbf{D}(\mathbf{Q_i}) \le S_f \\ 0 & \mathbf{D}(\mathbf{Q_i}) > S_f \end{cases} \qquad (18)$$

VI. Experiment and Results

A. Experiment Settings

The planning framework is summarized in Fig. 2. We follow Fast-Planner on our novel unified linear parametric map. We verify the performance of our method in different cluttered environments. We set the regularization coefficient α to 0.01 to reduce model complexity and avoid overfitting. A key design challenge is balancing real-time performance with mapping accuracy. We found that a mapping dimension of $M = 200$ provides an optimal trade-off, enhancing model performance with manageable computational costs while maintaining accuracy. This is noted in [12], and it served as the basis for our selection through multiple experiments.



(a) Comparison of updating time of the global grid map with different methods.

(b) Comparison of updating time of the ESDF map and planning time with different methods.

Fig. 8. Comparison of the proposed method against two SOTA planners with the same control parameters.

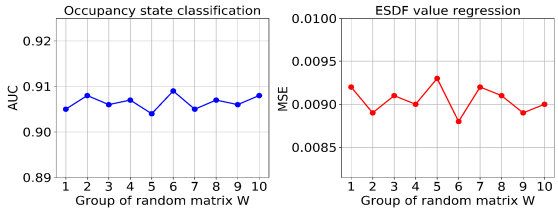


Fig. 9. Sensitivity analysis of RMM performance to random matrix W (ten trials): Occupancy state classification (left), ESDF value regression (right).

B. Grid Map: Kinodynamic Searching

We use the same kinodynamic path search method [4] to evaluate the search efficiency in occupancy grid maps with different occupancy mapping methods. We perform a ROS simulation in a $50 \times 50 \times 4$ m map with 60 randomly deployed obstacles. For comparison purposes, the popular uniform grid map [4] is compared using the same kinodynamic path searching methods. Mapping time, access time, and memory consumption of the grid map are crucial for path searching in the front-end. Thus, mapping time, access time, and memory consumption are presented in Table I. The mapping time of RMM, referring to

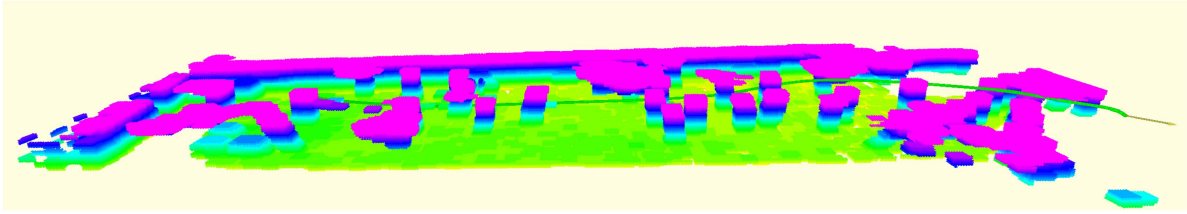


Fig. 10. Fast autonomous flight in a large-scale forest environment. The map is colored by height, and the blue trajectory is executed by the quadrotor.

TABLE I
THE PROPERTIES AND PERFORMANCE OF DIFFERENT OCCUPANCY MAPPING METHODS

Resolution	Method	map form	update	Access time	Storage space	mean AUC	Search time
Resolution = 0.2	RMM	linear classifier	SVD	**0.073**	**0.128**	**0.931**	**0.0053**
	Bayesian filtering	uniform grid	probabilistic fusion	0.271	23.764	0.962	0.0072
Resolution = 0.15	RMM	linear classifier	SVD	**0.132**	**0.613**	**0.921**	**0.0067**
	Bayesian filtering	uniform grid	probabilistic fusion	0.451	39.764	0.873	0.0132
Resolution = 0.1	RMM	linear classifier	SVD	**0.273**	**3.123**	**0.901**	**0.0083**
	Bayesian filtering	uniform grid	probabilistic fusion	0.851	53.764	0.8512	0.0329

TABLE II
PLANNER COMPARISON

Planner method	t(s)	Length	t_{ESDF}	t_{plan}	Success rate (%)
Fast-Planner	26.31	63.6	4.2	2.71	80
EGO-Planner	24.26	52.5	**0**	1.3	90
FELP-Planner	**23.02**	**51.01**	0.03	0.92	100

its training time, is more efficient than that of the uniform grid map, given the iterative nature of Bayesian filtering. Besides that, Bayesian filtering must keep the probability values to determine the occupation status, which brings challenges due to high computation and storage consumption in large-scale environments. Unlike Bayesian methods, our linear parameter map only needs to update and store some limited model parameters, making its storage more efficient, especially for large-scale environments. The novel parametric maps can be easily accessed by substituting $\boldsymbol{g}(\boldsymbol{W}x + b)$ into (2), giving rise to a shorter access time. Thus, as seen in Table I, compared to other methods at different resolutions, our path search time is the shortest. We tested the robustness of our occupancy grid map model by varying only the random matrix W. As shown in Fig. 9, the impact on AUC values was minimal.

C. ESDF Map: Trajectory Optimization

We compare the proposed planner with two state-of-the-art methods, Fast-Planner [4] and EGO-Planner [30], using their default recommended parameter settings. Undoubtedly, resolution is a crucial factor in ESDF map construction and trajectory optimization. As seen in Fig. 7, different resolutions are used for trajectory optimization in the back-end. The average performance statistics, ESDF updating time, and planning time are shown in Table II and Fig. 8. The results show that our closed-form ESDF map has better performance in ESDF updating and planning time. A similar sensitivity analysis for the ESDF map model, shown in Fig. 9, revealed a negligible impact from varying the random matrix W on its performance.

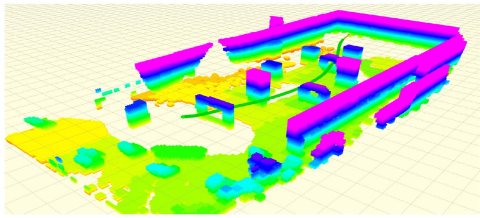


Fig. 11. Fast autonomous flight in a cluttered indoor environment. The map is colored by height, and the blue trajectory is followed by the quadrotor.

(a) Fully autonomous flight in an indoor environment. (b) Fully autonomous flight in a field forest environment.

Fig. 12. Fully autonomous flight in two different scenarios.

D. Real-World Experiments

We perform autonomous flight tests using a 360° field-of-view Lidar Mid360 on the NVIDIA Jetson Xavier NX, which features a 6-core NVIDIA Carmel ARM v8.2 64-bit CPU, a 384-core NVIDIA Volta GPU with 48 Tensor Cores, and 8 GB of 128-bit LPDDR4x RAM. One experiment involves navigating through pre-determined waypoints in a cluttered indoor environment (approximately $15 \times 25 \times 4$ meters), where obstacles are spaced less than one meter apart, with the smallest having a radius of less than a meter, as shown in Figs. 11 and 12. The drone follows the designated waypoints while dynamically avoiding obstacles. The maximum flight speed can reach 8 m/s. Therefore, this experiment demonstrates that the proposed method is capable of performing aggressive flight in cluttered environments.

A field forest scene is conducted for fast aggressive flight experiments to further validate our method. This field environment is a real-world outdoor scene, approximately 30×100 meters in area, characterized by dense trees, randomly distributed obstacles (trunks, branches), and both large-scale and unstructured features. The large amount of irregular point cloud data from sensors poses challenges to building high-quality maps and executing efficient real-time planning on limited resources. The drone should perform agile flight to avoid obstacles such as trunks and leaves in the forest, as shown in Fig. 10 and 12. The experiment shows that the proposed method enables fully autonomous flight in a large-scale and complex field environment.

VII. Conclusion and Future Work

We use RMM to build grid maps and ESDF maps into a unified linear model framework. The proposed linear map model has significant advantages in autonomous navigation, especially in large-scale unstructured environments. First, based on the parameter representation form of the grid map, we can map the grid map and access the occupancy status in a faster way, allowing us to quickly complete kinodynamic path searching in the front-end. Secondly, we can quickly solve the Euclidean distance instead of iterating extensively. Unlike traditional methods, where the ESDF map is obtained by trilinear interpolation, its accuracy and computing efficiency depend on resolution. The learned linear parametric model can obtain the closed-form ESDF map, which is then used for high-accuracy and effective trajectory optimization in the back-end. Finally, we validate our proposed map in various complex environments. The results show that our method possesses better comprehensive performance in terms of mapping time, memory consumption, access time, accuracy, and planning time.

In the future, we will perform large-scale multi-AAV exploration based on this particular environmental model. This linear parameter model representation has the following advantages. Firstly, updating restricted model parameters to store and map the occupancy grid map efficiently. Secondly, this lightweight environment model is very suitable for information exchange under limited bandwidth.

References

[1] J. P. Queralta et al., "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *IEEE Access*, vol. 8, pp. 191617–191643, 2020.

[2] D. Huo, L. Dai, R. Chai, R. Xue, and Y. Xia, "Collision-free model predictive trajectory tracking control for UAVs in obstacle environment," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 59, no. 3, pp. 2920–2932, Jun. 2023.

[3] C. Jiang et al., "H_2-Mapping: Real-time dense mapping using hierarchical hybrid representation," *IEEE Robot. Automat. Lett.*, vol. 8, no. 10, pp. 6787–6794, Oct. 2023.

[4] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3529–3536, Oct. 2019.

[5] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 779–786, Apr. 2021.

[6] B. Zhou, J. Pan, F. Gao, and S. Shen, "RAPTOR: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1992–2009, Dec. 2021.

[7] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "FASTER: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 922–938, Apr. 2022.

[8] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory Comput.*, vol. 8, no. 1, pp. 415–428, 2012.

[9] B. Lau, C. Sprunk, and W. Burgard, "Improved updating of Euclidean distance maps and Voronoi diagrams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 281–286.

[10] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.

[11] L. Han, F. Gao, B. Zhou, and S. Shen, "FIESTA: Fast incremental Euclidean distance fields for online motion planning of aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4423–4430.

[12] X. Liu, D. Li, and Y. He, "A unified framework for large-scale occupancy mapping and terrain modeling using RMM," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5143–5150, Apr. 2022.

[13] H. Nie, J. Chen, G. Zhang, D. Li, and Y. He, "Efficient and hierarchical quadrotor planner for fast autonomous flight," in *Proc. Int. Conf. Intell. Robot. Appl.*, 2023, pp. 166–175.

[14] K. Doherty, T. Shan, J. Wang, and B. Englot, "Learning-aided 3-D occupancy mapping with Bayesian generalized kernel inference," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 953–966, Aug. 2019.

[15] S. Srivastava and N. Michael, "Efficient, multifidelity perceptual representations via hierarchical Gaussian mixture models," *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 248–260, Feb. 2019.

[16] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *Int. J. Robot. Res.*, vol. 31, no. 1, pp. 42–62, 2012.

[17] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, pp. 189–206, 2013.

[18] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, 2013.

[19] S. Fredriksson, A. Saradagi, and G. Nikolakopoulos, "Voxel map to occupancy map conversion using free space projection for efficient map representation for aerial and ground robots," *IEEE Robot. Automat. Lett.*, vol. 9, no. 12, pp. 11625–11632, Dec. 2024.

[20] J. Wang and B. Englot, "Fast, accurate Gaussian process occupancy maps via test-data octrees and nested Bayesian fusion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1003–1010.

[21] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4460–4470.

[22] B. Mildenhall, P.P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[23] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 489–494.

[24] Y. Chen, S. Lai, J. Cui, B. Wang, and B. M. Chen, "GPU-accelerated incremental Euclidean distance transform for online motion planning of mobile robots," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 6894–6901, Jul. 2022.

[25] M. Pantic, C. Cadena, R. Siegwart, and L. Ott, "Sampling-free obstacle gradients and reactive planning in neural radiance fields," in *Proc. Workshop Motion Plan. Implicit Neural Representations Geometry IEEE Int. Conf. Robot. Automat.*, 2022, pp. 1–3.

[26] G. Gil, J. A. Cobano, F. Caballero, and L. Merino, "Evaluation of neural Euclidean signed distance fields for distance-aware local path planning," in *Proc. 7th Iberian Robot. Conf.*, 2024, pp. 1–6.

[27] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 5332–5339.

[28] S. Geng, Q. Wang, L. Xie, C. Xu, Y. Cao, and F. Gao, "Robo-centric ESF: A fast and accurate whole-body collision evaluation tool for any-shape robotic planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 290–297.

[29] A. Meijster, J. B. Roerdink, and W. H. Hesselink, "A general algorithm for computing distance transforms in linear time," in *Mathematical Morphology and Its Applications to Image and Signal Processing*. Boston, MA, USA: Springer, 2000, pp. 331–340.

[30] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An EDF-free gradient-based local planner for quadrotors," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 478–485, Apr. 2021.