2025 International Conference on
Unmanned Aircraft Systems (ICUAS)
May 14-17, 2025. North Carolina, USA

Optimization-Based Motion Planning for Vector Field Following in Dynamic Environments

David Akhihiero, Uthman Olawoye, and Guilherme A. S. Pereira

***Abstract*— This paper proposes a method for integrating trajectory optimization with vector field-based motion planning methods. This approach aims to address motion planning challenges, particularly in scenarios like UAV navigation, where vector fields are efficient but struggle with dynamic obstacles and motion constraints. Such challenges also include scenarios where there is no defined goal configuration such as border following, loitering, and curve circulation. Several vector field methods have been proposed to solve these problems but they are prone to failure when encountering previously unmodeled or dynamic obstacles as well as no-fly zones. The method proposed in this paper uses a vector field for high-level planning. The vector field is used to create paths for the vehicle, which are optimized for smoothness, obstacle avoidance, and vector field adherence before they are followed. The result is a smooth path that is fast to plan and easy to follow for a motion-constrained vehicle. A series of simulations was used to validate this methodology, which is compared with a previous method that uses RRT*.**

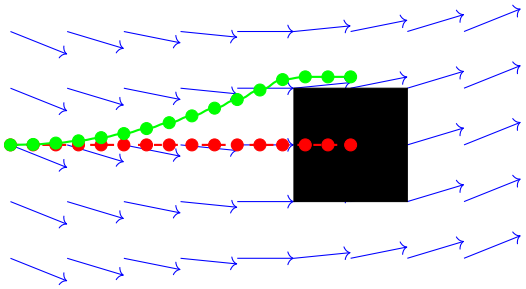

Fig. 1. Illustration of the proposed optimization-based method for UAV motion planning. The vector field (blue arrows) guides the UAV but does not account for dynamic or previously unknown obstacles. Therefore, the initial waypoint parametrized trajectory (red), obtained as the integration of the field, collides with an obstacle (black square). The proposed method optimizes the trajectory so it avoids the obstacle, is smooth, and adheres to the vector field (green).

I. Introduction

Vector field techniques are popular in robot motion planning because they are intuitive, simple, and have low computational cost. As closed-loop methods, they are robust to small localization and actuation errors, making them popular for tasks like UAV/UGV patrolling, inspections, and monitoring. In these methods, a velocity or acceleration vector drives the robot through collision-free trajectories from any start configuration towards some goal or along some path. Several vector field methodologies have been proposed. In the first method proposed by Khatib [1], the vector field was the negative gradient of an artificial potential field where the potential field has its minimum value at the goal configuration. This method had limitations, like sometimes converging to a local minima, and motion oscillation when the robot moves at high speeds among close obstacles. To address some of these issues, other potential field variants like the generalized potential fields method [2] and harmonic functions [3] were proposed. Some vector field methods that are not based on gradients of functions have been developed to solve planning tasks like tracking [4] and curve circulation [5], [6].

However, vector field techniques face some challenges: (i) although originally designed for dynamic environments, vector fields often lose global convergence properties when encountering dynamic or previously unknown obstacles; and (ii) they do not consider the vehicle's motion constraints, limiting their applicability in real-world scenarios. In a recent paper, the authors of [7] address the obstacle avoidance problem by modifying a curve-circulation vector field as soon as a new obstacle is detected. The methodology is very elegant but can only be applied to very specific vector fields. The framework proposed by Pereira et al. [8] is more generic and works by integrating vector fields with the optimal version of the Rapidly-Exploring Random Tree (RRT) algorithm, RRT* [9]. While this approach is adaptive for dynamic environments, this framework has some limitations: sample rejection during path generation leads to wasted computational effort, and the straight-line paths produced by RRT* do not consider the vehicle motion constraints (although this could be addressed by using Dubins paths as local planners, as proposed by the authors in [10]). These drawbacks show the need for a more efficient, adaptive, and motion-constrained planning framework that retains the advantages of vector field techniques while addressing their shortcomings. The methodology proposed in this paper can address the shortcomings of the framework proposed in [8].

Therefore, the contribution of this paper is the extension of the framework proposed by Pereira et al. [8] by replacing RRT* with an optimization-based planner inspired by the Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [11]. CHOMP has been modified and applied before to multiple motion planning problems like manipulation [12] and UAV trajectory planning [13]. A work by Bonatti et. al [14] used a CHOMP-based planning algorithm to optimize the trajectories of a cinematography drone for smoothness, obstacle avoidance, shot quality, and occlusion

David Akhihiero, Uthman Olawoye, and Guilherme A. S. Pereira are with the Department of Mechanical, Materials and Aerospace Engineering, Benjamin M. Statler College of Engineering and Mineral Resources, West Virginia University. Morgantown, WV, 26501, USA. Emails: daa00017@mix.wvu.edu, uoo00001@mix.wvu.edu, guilherme.pereira@mail.wvu.edu

avoidance.

Unlike RRT*, our CHOMP-inspired method uses gradient-based optimization to generate smooth trajectories while considering constraints such as obstacle avoidance and the vector field that guides the path. It avoids the inefficiencies of sampling-based methods and provides a computationally efficient solution for real-time applications. This method is illustrated in Fig. 1, where a UAV navigates an area guided by a vector field that pulls it towards a super-ellipse curve at a fixed height. We assume that the vector field ignores the obstacles when planning an initial trajectory. This trajectory is then optimized for smoothness and obstacle avoidance resulting in a trajectory that is close to the original trajectory but avoids obstacles. By optimizing a functional that balances smoothness and obstacle avoidance, the optimizer iteratively enhances the quality of an initial trajectory using functional gradient techniques. This algorithm is based on two main ideas: (i) gradients are available and can be cheaply computed and (ii) trajectory optimization is invariant to parametrization [11]. The first idea is formalized with three objective functionals (a) a smoothness functional, $\mathcal{F}_{smooth}[\xi]$, reflecting the trajectory dynamics, (b) an obstacle functional, $\mathcal{F}_{obs}[\xi, \mathcal{D}]$, which represents the obstacle avoidance requirement, and (c) a vector field functional, $\mathcal{F}_{vf}[\xi, u]$ which aligns the trajectory with the guidance of a predefined vector field. The smoothness functional is defined in terms of a metric in the trajectory space enabling it to include higher-order derivatives. The obstacle cost is framed in terms of distance fields and using Euclidean distance transforms, the obstacle functional gradients can be efficiently computed. The smoothness functional controls the timing along the path while the obstacle functional controls the shape of the path. Additionally, the vector field functional guides the trajectory alignment with the desired path specified by the vector field, ensuring that the robot adheres to high-level motion directives while maintaining feasibility. For vector field design, we acknowledge the importance of singularity-free constructions as discussed in [15], which is especially relevant for more complex patrol shapes like lemniscates.

The next section formally defines the problem solved in this paper. Our methodology is explained in Section III. MATLAB simulations are presented in Section IV, and conclusions and future work are stated in Section V.

II. Problem Definition

Let $Q \subset \mathbb{R}^n$ represent the configuration space of a robot, where each point in Q corresponds to a unique configuration of the robot. The configuration space is divided into valid and invalid subsets: the valid set of configurations $Q_{\text{free}} \subset Q$, where the robot can safely navigate, and the invalid set $Q_{\text{obs}} \subset Q$, which represents obstacles or no-fly zones. The invalid set Q_{obs} can be further partitioned as $Q_{\text{obs}} = Q^s_{\text{obs}} \cup Q^d_{\text{obs}}$, where Q^s_{obs} represents previously known obstacles or no-fly zones, and Q^d_{obs} denotes dynamic or previously unknown obstacles or no-fly zones that may be encountered during the robot's motion.

Let $u : Q \setminus Q^s_{\text{obs}} \to \mathbb{R}^n$ be a continuous vector field that specifies the desired motion of the robot. This vector field is generated by a global planner without knowledge of Q^d_{obs}. Let $\xi : [0, 1] \to Q$ be a continuous path representing the robot's initial trajectory pre-computed as the integral curve of the vector field, with $\xi(0) = q_0 \in Q_{\text{free}}$ as the initial configuration and $\xi(1) = q_1 \in Q_{\text{free}}$ as the final configuration.

Because the vector field is constructed without the knowledge of Q^d_{obs}, trajectory ξ may be unsafe for the vehicle. So, our problem is to optimize ξ for smoothness and obstacle avoidance, including both known obstacles in Q^s_{obs} and previously unknown obstacles in Q^d_{obs} that are detected or communicated to the vehicle during the flight. To guarantee that the task of the robot is achieved, ξ must also follow the vector field as closely as possible. They then propose a motion planning problem that locally changes the original trajectory as:

Problem 1: Given an initial trajectory ξ computed as the integral of vector field u with $\xi(0) = \mathbf{q}_0 \in Q_{\text{free}}$ and $\xi(1) = \mathbf{q}_1 \in Q_{\text{free}}$, where $\mathbf{q}_0$ is at the center of a ball of radius R and $\mathbf{q}_1$ is at the border of the ball, optimize ξ keeping $\mathbf{q}_0$ fixed such that ξ is smooth, collision-free and followed the vector field u as close as possible.

This problem can be mathematically expressed as:

$$\begin{aligned} \text{minimize} \quad & \mathcal{U}[\xi] = \lambda_1 \mathcal{F}_{smooth}[\xi] + \lambda_2 \mathcal{F}_{obs}[\xi, \mathcal{D}] + \\ & \qquad + \lambda_3 \mathcal{F}_{vf}[\xi, u] \\ \text{subject to :} \quad & \xi(0) = \mathbf{q}_0, \\ & \xi(s) \in Q_{\text{free}}, \quad \forall s \in [0, 1] \end{aligned} \tag{1}$$

where:

- $\mathcal{F}_{smooth}[\xi]$ represents the smoothness functional,
- $\mathcal{F}_{obs}[\xi, \mathcal{D}]$ represents the obstacle functional,
- $\mathcal{F}_{vf}[\xi, u]$ represents the vector field functional, and
- λ_1, λ_2, and λ_3 are scalars that represent the weight of each term in the functional.

Constraint $\xi(0) = \mathbf{q}_0$ ensures that the trajectory starts at the initial configuration and $\xi(s) \in Q_{\text{free}}$ ensures that the trajectory remains within the valid configuration space Q_{free} at all times. Notice that there is no constraints for the final configuration, expect that it must be in the free space.

The previous local motion planning problem is solved during the motion of the vehicle in intervals that depend mainly on its sensors' field of view and its speed. The basic idea is that the vehicle (i) optimizes the path inside the planning horizon (ball of radius R in our case) by solving Problem 1, (ii) follows the path for some time, (iii) creates a new planning cube with new obstacles information obtained by the sensors, (iv) repeat the process starting in (i).

III. Methodology

In this work, an optimizer based on CHOMP is used to solve Problem 1. CHOMP is a trajectory optimization method that can solve simple planning problems and is based on Elastic Bands proposed by Quinlan and Khatib [16]. It is not a complete planner (in the sense it may fail to

find a trajectory free of collisions, even if one exists) but it is excellent for local planning [17] which means that it is well suited for repairing a locally generated vector field trajectory that intersects obstacles, which is the scope of this paper. The application of optimization to address the problem stated in the previous section will be shown in the ensuing subsections.

A. Optimizer

The algorithm optimizes an objective functional that takes into account the proximity to obstacles, the smoothness of the trajectory, and adherence to a vector field. It works using steepest descent optimization directly on the trajectories and uses reparameterization to maintain consistency independent of trajectory parameterization. It uses a discretized waypoint representation for computational efficiency while maintaining theoretical behavior. An important aspect is the use of workspace gradient information, which allows the robot to be placed in collision-free configurations even if the original trajectory is not feasible. This eliminates the need for a separate planner to first find a feasible trajectory, as is common with traditional approaches [18].

In this work, there is no predefined global goal configuration. The robot starts from an initial configuration, uses the vector field to plan a preliminary trajectory from the initial to an intermediate goal configuration some distance R from the start, and optimizes that trajectory for obstacle avoidance, smoothness, and vector field adherence. It then generates a new preliminary trajectory starting from the last intermediate goal configuration to a new goal and repeats the trajectory optimization. This continues until some stopping condition is met. The algorithm for this methodology is shown in Algorithm 1 which is explained below.

Algorithm 1 Iterative Trajectory Optimization

1: Initialize q_0
2: **while** not stopping condition **do**
3: $traj_{opt\_old} \leftarrow$ GETVECTORFIELDPATH(q_0, R)
4: $[D, \nabla D] \leftarrow$ SIGNEDDISTANCEFIELD(M)
5: **for** $j = 1$ **to** m **do**
6: $traj_{opt} \leftarrow$ OPTIMIZETRAJ$(traj_{opt_old}, D, \nabla D)$
7: $traj_{opt} \leftarrow$ RESAMPLETRAJECTORY$(traj_{opt})$
8: $traj_{opt\_old} \leftarrow traj_{opt}$
9: **if** $\|\Delta traj\| < thresh$ **then**
10: break
11: **end if**
12: **end for**
13: $q_0 \leftarrow$ final configuration of the optimized trajectory
14: **if** stopping condition **then**
15: break
16: **end if**
17: **end while**

In line 1, the initial configuration, q_0 is set. Then, as long as the stopping condition is not met, the robot plans trajectories to a goal configuration distance R from the initial. In line 3, a discretized preliminary trajectory is computed through the integration of the vector field. In line 4, using a discrete map M of the environment, centered at q_0 and of width and height $2R$, a signed distance field $\mathcal{D}$ and its gradient $\nabla\mathcal{D}$ are computed. In lines 5 to 12, the preliminary trajectory is iteratively optimized. After each optimization iteration (line 6), the optimized trajectory is resampled (line 7) to ensure that the spacing between waypoints in the parametrized trajectory remains relatively constant. The algorithm for this resampling is detailed in Algorithm 2. Lines 9-11 terminate the iteration when the optimization converges. After optimization, the initial configuration is reset to the final configuration of the previous optimized trajectory (line 13).

Algorithm 2 Resample Trajectory

Input: Optimized trajectory $traj$, $avg\_spacing$
Output: Resampled trajectory $traj$
$distances \leftarrow \sqrt{\sum(\text{diff}(traj, 1, 2)^2)}$
$arc\_len \leftarrow [0, \text{cumsum}(distances)]$
$total\_len \leftarrow arc\_length(\text{end})$
$des\_num\_pts \leftarrow \lceil \frac{total\_len}{avg\_spacing} \rceil$
$new\_len \leftarrow \text{linspace}(0, total\_len, des\_num\_pts)$
for $j = 1$ **to** $\text{size}(traj_{\text{opt}})$ **do**
$traj(j) \leftarrow \text{interp1}(arc\_len, traj(j), new\_len)$
end for

The OPTIMIZETRAJ() function optimizes an existing trajectory for smoothness, obstacle avoidance, and vector field adherence. The optimization involves the minimization of the functional $\mathcal{U}[\xi]$ defined in Equation 1 which is composed of three functionals; a smoothness functional, $\mathcal{F}_{smooth}[\xi]$, an obstacle avoidance functional, $\mathcal{F}_{obs}[\xi, \mathcal{D}]$ and a vector field functional, $\mathcal{F}_{vf}[\xi, u]$ which are described in the subsequent subsections. Once functional $\mathcal{F}_{smooth}[\xi]$ is defined, its gradient $\nabla\mathcal{U}[\xi_i]$ is computed as:

$$\nabla\mathcal{U}[\xi] = \lambda_1\nabla\mathcal{F}_{smooth}[\xi] + \lambda_2\nabla\mathcal{F}_{obs}[\xi, \mathcal{D}] + \lambda_3\nabla\mathcal{F}_{vf}[\xi, u]\,, \tag{2}$$

and the trajectory is updated as:

$$\xi_{i+1} = \xi_i - \eta\nabla\mathcal{U}[\xi_i]\,, \tag{3}$$

where η is the step size. Each functional and their gradients are explained in the following subsections.

B. Smoothness Functional

The smoothness functional $\mathcal{F}_{smooth}$ measures the integral over squared velocity norms as proposed in [11]. It is of the form:

$$\mathcal{F}_{\text{smooth}}[\xi] = \frac{1}{2}\int_0^1 \left\| \frac{d}{dt}\xi(t) \right\|^2 dt\,. \tag{4}$$

$$\frac{d}{dt}\xi_i \approx \frac{q_{i+1} - q_i}{\Delta t} \tag{5}$$

For a waypoint parametrized trajectory it can be approximated as:

$$\mathcal{F}_{\text{smooth}}[\xi] = \frac{1}{2}\sum_{i=1}^{n+1} \left\| \frac{q_{i+1} - q_i}{\Delta t} \right\|^2 . \tag{6}$$

In this way, for waypoint i, the gradient of this functional can be computed as:

$$\nabla \mathcal{F}_{\text{smooth}}[q_i] = \frac{1}{\Delta t^2}\left[(q_{i+1} - q_i) + (q_i - q_{i-1})\right]. \quad (7)$$

C. Obstacle Avoidance Funtional

The obstacle avoidance functional $\mathcal{F}_{obs}$ enforces collision-free trajectories [11] and is of the form:

$$\mathcal{F}_{\text{obs}}[\xi, \mathcal{D}] = \int_0^1 c(\xi(t)) \left\| \frac{d}{dt}\xi(t) \right\| dt\,, \quad (8)$$

For a discretized trajectory, it is

$$\mathcal{F}_{\text{obs}}[\xi, \mathcal{D}] = \sum_{i=1}^{n+1} c(q_i) \left\| \frac{d}{dt}\xi_i \right\|, \quad (9)$$

where c is a cost function that penalizes closeness to obstacles given by:

$$c(q_i) = \begin{cases} -\mathcal{D}(q_i) + \frac{1}{2}\epsilon & \text{if } \mathcal{D}(q_i) < 0 \\ \frac{1}{2\epsilon}\left(\mathcal{D}(q_i) - \epsilon\right)^2 & \text{if } 0 < \mathcal{D}(q_i) \leq \epsilon \\ 0 & \text{otherwise}\,, \end{cases} \quad (10)$$

where

- $\mathcal{D}$ is the signed distance field function, and
- ϵ is the minimum distance from obstacles.

The gradient of the cost function is given by:

$$\nabla c(q_i) = \begin{cases} -\nabla \mathcal{D}(q_i) & \text{if } \mathcal{D}(q_i) < 0 \\ \frac{1}{\epsilon}\left(\mathcal{D}(q_i) - \epsilon\right)\nabla \mathcal{D}(q_i) & \text{if } 0 < \mathcal{D}(q_i) \leq \epsilon \\ 0 & \text{otherwise}\,. \end{cases} \quad (11)$$

For waypoint i, the gradient of this functional can be approximated as:

$$\nabla \mathcal{F}_{\text{obs}}[\xi] = \|q_i'\|((I - \hat{q_i}'\hat{q_i}'^T)\nabla c - c\kappa)\,, \quad (12)$$

where:

- q_i' is the first derivative of q_i computed numerically,
- $\|q_i'\|$ is the norm of the vector q_i' computed numerically,
- $\hat{q_i}'$ is the unit vector in the direction of q_i', defined as $\hat{q_i}' = \frac{q_i'}{\|q_i'\|}$, and
- κ is the trajectory curvature given by:

$$\kappa = \frac{1}{\|q_i'\|^2}\left(I - \hat{q_i}'\hat{q_i}'^T\right) q_i''\,. \quad (13)$$

where q_i'' is the second derivative of q_i computed numerically.

D. Signed Distance Field (SDF)

As presented in [11], assuming that every obstacle is a closed object with a finite volume, the value of the signed distance field $\mathcal{D}(q)$ is negative if the point q is inside an obstacle, positive if q is outside of all obstacles, and zero if the point is on the edge of an obstacle. The signed distance field can be computed using a discrete Euclidean distance transform (EDT) algorithm starting with a boolean discrete obstacle grid where 1 represents obstacles and 0 is free space. The EDT can be calculated for both the obstacle grid and its complement, and the Signed Distance Field (SDF) is derived as the difference between these two distance fields. An example of SDF construction for a 2D environment is shown in Fig. 2

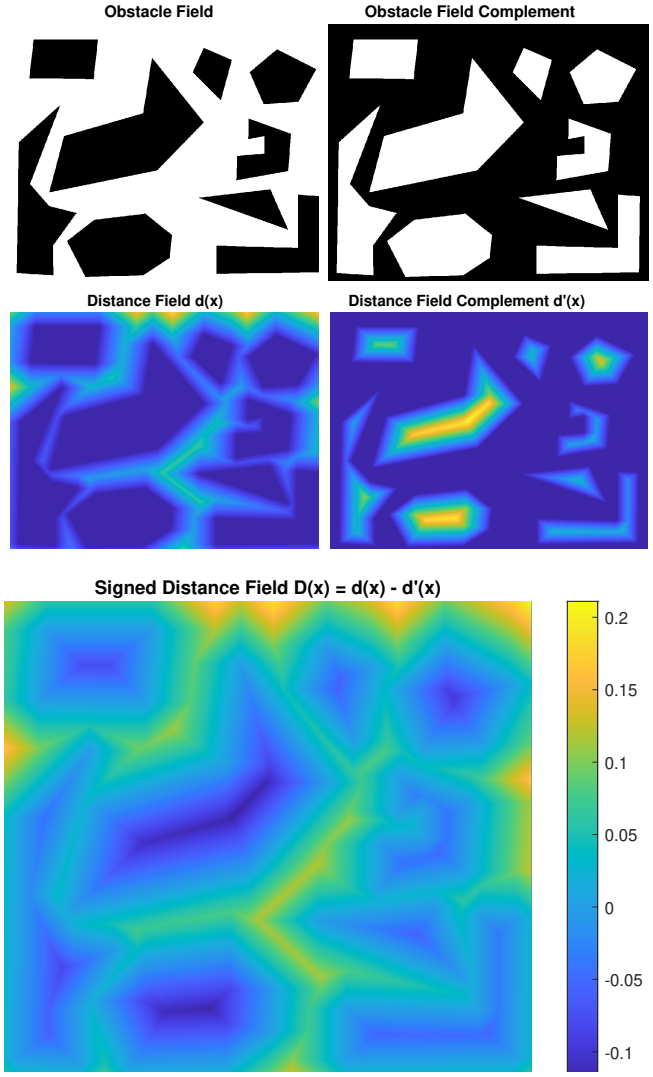

Fig. 2. Signed Distance Field (SDF) construction. The SDF is created as the difference between two distance transforms: one over a binary grid where obstacles are set to 1, and the other over its complement where obstacles are set to 0, so that $D(x) = d(x) - d'(x)$. Brighter colors represent higher values.

E. Vector Field Functional

The vector field functional $\mathcal{F}_{\text{vf}}$ is one of the main contributions of this paper. It represents the alignment of the trajectory with a given vector field so that, by minimizing the functional, the deviation of the trajectory from the desired vector field is reduced, ensuring that the motion respects predefined directional influences throughout its path. In this paper, the functional is defined as:

$$\mathcal{F}_{\text{vf}}[\xi, u] == \int_0^1 \left(1 - \frac{\frac{d}{dt}\xi(t) \cdot \mathbf{u}(\xi(t))}{\left\|\frac{d}{dt}\xi(t)\right\| \|\mathbf{u}(\xi(t))\|}\right) dt\,, \quad (14)$$

where:

- $\xi(t)$ is the trajectory,
- $\mathbf{u}(\xi(t))$ is the vector field at the current position $\xi(t)$,
- $\frac{d}{dt}\xi(t)$ is the first derivative (velocity) of the trajectory at time t.

Notice that the functional is minimum (0) when the internal product between the field and the derivative of the trajectory is maximum (they are aligned) and maximum (2) when they are colinear but point in different directions.

Our objective is to minimize the difference between the trajectory's velocity and the vector field at each point along the path. So, for a waypoint parametrized trajectory we have:

$$\mathcal{F}_{\text{vf}}[\xi, u] = \sum_{i=1}^{n+1} \left(1 - \frac{\frac{d}{dt}\xi_i \cdot \mathbf{u}(\xi_i)}{\left\| \frac{d}{dt}\xi_i \right\| \|\mathbf{u}(\xi_i)\|} \right) . \tag{15}$$

After dropping higher derivative terms, a good approximation of the gradient of this functional is computed as the direction of the normalized vector field at each waypoint and can be approximated as:

$$\nabla \mathcal{F}_{\text{vf}}[\xi, \mathbf{u}] \approx -\mathbf{u}_i = -\frac{\mathbf{u}(\xi_i)}{\|\mathbf{u}(\xi_i)\|} , \tag{16}$$

where:

- $\mathbf{u}_i$ is the normalized vector at waypoint i,
- $\mathbf{u}(\xi_i)$ is the vector field at waypoint i,
- $\|\mathbf{u}(\xi_i)\|$ is the norm of the vector field at waypoint i.

This gradient deforms the trajectory in a way it follows the direction specified by the vector field at each point.

IV. Analysis

Our proposed methodology's theoretical and practical limitations are defined by several underlying assumptions. Stable gradient computation and consistent global guidance depend on the fundamental assumption that the vector field $u(q)$ is continuous over $Q \setminus Q^s_{obs}$. This continuity guarantees that the robot's motion has a smooth and consistent path due to the vector field. Furthermore, we assume that the sensing modality is error-free within a local planning radius R, implying that obstacles are detected instantly and that the conversion to a binary occupancy grid is exact. Finally, the robot's stable velocity must be higher than the velocities of the dynamic obstacles. These assumptions allow the approach to preserve computational efficiency and give reliable motion planning capabilities, even though they may be restrictive in real-world situations.

Our method's computational complexity is principally determined by the signed distance field (SDF) computation, which scales as $O(n^3)$ for a 3-dimensional workspace discretized at resolution n. The workspace discretization introduces additional constraints, with a finer discretization providing more accurate obstacle representation but increasing computational overhead, whilst a coarser discretization may overlook narrow passages or define artificial boundaries in the configuration space. The scaling behavior of the computational complexity sets realistic restrictions on the maximum planning radius and minimum discretization resolution.

The performance of the optimization framework is theoretically limited by several factors. The convergence of the optimization is critically dependent on the selection of weights $\lambda_1, \lambda_2, \lambda_3$ and step size η, resulting in a trade-off between smoothness, obstacle avoidance, and vector field adherence. Although the approach ensures that the trajectory is locally optimal in terms of the defined functionals, it cannot guarantee global optimality. This limitation arises because the optimizer refines the trajectory locally, starting from an initial path generated by the vector field, and may not explore alternative paths that could be globally optimal. The method's performance is further bound by time constraints, which require that the maximum number of optimization iterations be limited to enable real-time operation and that the computation time not exceed the planning horizon. Moreover, it is well known that local optimization methods are not complete, which means that the resultant trajectory may no be free of collisions, depending on the shape of the obstacles and the number of optimization iterations.

V. Simulations

To evaluate our method, we implemented the methodology of Pereira et al. [8] and our proposed methodology in MATLAB on a computer with the following specifications: Ubuntu 22.04.5 LTS operating system, Intel Core i7-10700 processor (16 threads @ 4.8 GHz), and 32 GB of RAM. We ran two simulations, in different environments, with both methods and compared their performances. The first simulation environment was a $40\,\text{m}$ wide corridor with a continuous vector field (similar to the one in used [8]). The robot was tasked to navigate along a longitudinal line at a distance d_0 from the corridor center. This vector field can be calculated as:

$$\text{u(q)} = \begin{bmatrix} 1 \\ k(d_0 - y) \end{bmatrix} , \tag{17}$$

which assumes that the corridor is parallel to the world coordinate frame's x-axis. The speed at which the robot approaches the line is determined by a positive gain, k.

In our simulation, $k = 0.1$, $d_0 = 5$ and the planning radius, $R = 70\,\text{m}$. The other parameters for Pereira et al. [8] methodology are $\delta = 0.5\,\text{m}$, $a = 10$, $b = 9$, $\eta = 1\,\text{m}$, $p_r = 90\%$ and $\theta = 60°$ and for our method, the parameters are $\eta = 0.01$, $\epsilon = 2\,\text{m}$, $\lambda_1 = 10$, $\lambda_2 = 300$, $\lambda_3 = 0.2$. The discretization resolution for the environment and the signed distance field was $0.1\,\text{m}$. Figure 3 shows a simulation of the robot, with both methods, starting at an initial configuration $q_0 = [-25, -15]^T$ tasked with navigating in a corridor with unmodeled obstacles. In the simulation, the robot computes a trajectory from q_0 to the border of the planning region, follows the path, and before completing the trajectory, plans a new trajectory starting from the last configuration of the current trajectory.

In the second simulation, a UAV is tasked to patrol a neighborhood by circulating a planar curve of the form $x^4 + y^4 = c^4$, at a fixed height where $c = 20\,\text{m}$, while detecting obstacles and avoiding no-fly zones with a planning radius of $12\,\text{m}$. The parameters for Pereira et al. [8] methodology are

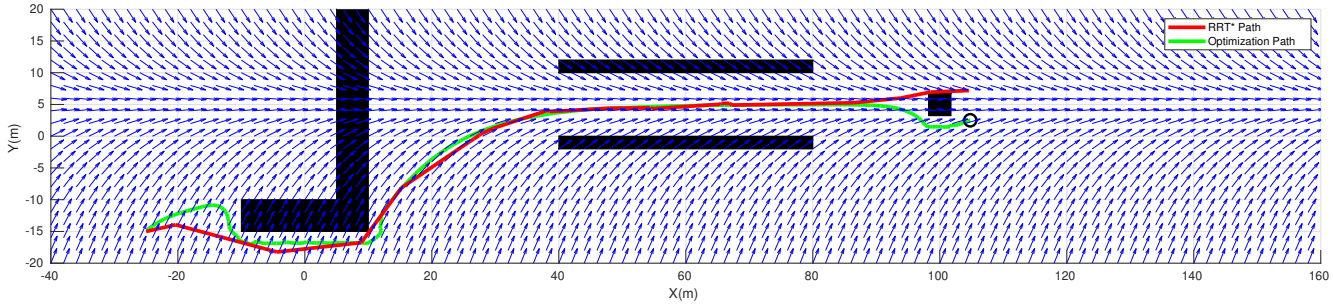

Fig. 3. A simulation of a robot navigating in a corridor with obstacles. The vector field is shown with normalized blue arrows, the path calculated using Pereira et al. [8] methodology is shown in red and our path is shown in green. The black circle shows the final configuration.

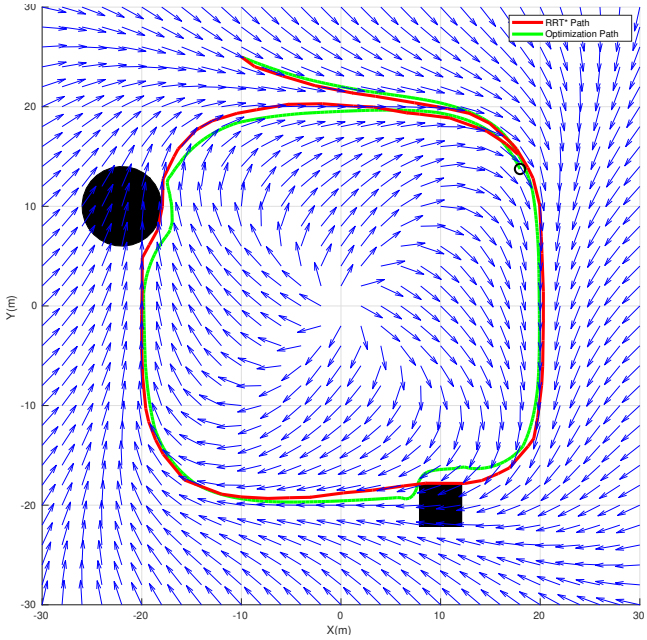

Fig. 4. A simulation of a UAV patrolling a neighborhood by circulating a planar curve with unmodeled obstacles. The vector field is shown with normalized blue arrows, the path calculated using Pereira et al. [8] methodology is shown in red and our path is shown in green. The black circle shows the final configuration.

$\delta = 0.5\,\text{m}$, $a = 10$, $b = 9$, $\eta = 1\,\text{m}$, $p_r = 90\%$ and $\theta = 60°$. For our method, the parameters are $\eta = 0.001$, $\epsilon = 2\,\text{m}$, $\lambda_1 = 10$, $\lambda_2 = 300$, $\lambda_3 = 1$. The discretization resolution for the environment and the signed distance field was $0.1\,\text{m}$. Figure 4 shows a simulation of the UAV, using both methods, starting from an initial configuration $q_0 = [-10, 25]^T$ tasked with patrolling a neighborhood along a planar curve at a fixed height. The UAV computes a trajectory from q_0 to the border of the planning region, follows the path for one second then computes a new trajectory from the current configuration to the border of the new planning region centered around the current configuration. During the flight, the UAV also takes into account no-fly zones and adjusts its path accordingly.

Table I shows the comparison between the two methods for both simulations for path length and computation time. Note that the computation time reported in Table I for Simulation 2 corresponds to the cumulative time required to generate all trajectories throughout the simulation. In practice, the UAV performs local replanning within a 12 m radius at one-second intervals, with each individual planning step requiring significantly less time than the total trajectory time reported in the table. The pre-processing time for our method includes the time for generating the discretized binary grid and the signed distance field. The method by Pereira et al. [8] sometimes computes a slightly shorter path but it takes a much longer time than our method. The path computed by our method is smooth, avoids obstacles, and conforms closely to the vector field guidance for most of the trajectory.

To further illustrate the behavior of our method in a 3D setting, we present a third simulation, as depicted in Fig. 5. In this scenario, the UAV starts on the ground and ascends while following a 3D vector field around the super-ellipse curve until it reaches the desired height. The optimized trajectory refines the UAV's path, ensuring smooth motion, obstacle avoidance, and adherence to the vector field. For simplification, the obstacles in this environment maintain the same 2D cross-section irrespective of height, but the method can be extended to handle more complex 3D obstacles. This visualization highlights the adaptability of our approach in handling challenging environments, reinforcing its effectiveness for real-world applications.

VI. Conclusion

In this work, we have proposed an integrated approach that combines vector field-based motion planning with trajectory optimization. Inspired by the work of Pereira et al. [8], which explored integrating optimal sample-based motion planning with vector fields, we extended the concept to optimize trajectories for smoothness, obstacle avoidance, and vector field adherence using a CHOMP-based approach. Our contribution compared to CHOMP [11] has been to enable vector fields to guide the path by including them in an optimization formulation while ensuring that the trajectories are smooth, feasible, and adaptive to dynamic environments. The results indicated that the optimization-based method produced smooth and obstacle-free paths that closely followed the original vector field guidance, with higher computational efficiency

TABLE I

COMPARISON OF METHODOLOGIES FOR PATH LENGTH AND COMPUTATION TIME IN BOTH SIMULATIONS

Methodology	Pre-processing Time (s)	Path-finding Time (s)	Total Time (s)	Path Length (m)
Simulation 1: Robot in a Corridor				
Pereira et al. [8]	0	281.4208	281.4208	138.4182
Our Method	80.5409	16.4052	96.9461	144.6308
Simulation 2: UAV Patrolling a Neighborhood				
Pereira et al. [8]	0	442.5522	442.5522	167.7933
Our Method	22.6284	13.0947	35.7231	166.5466

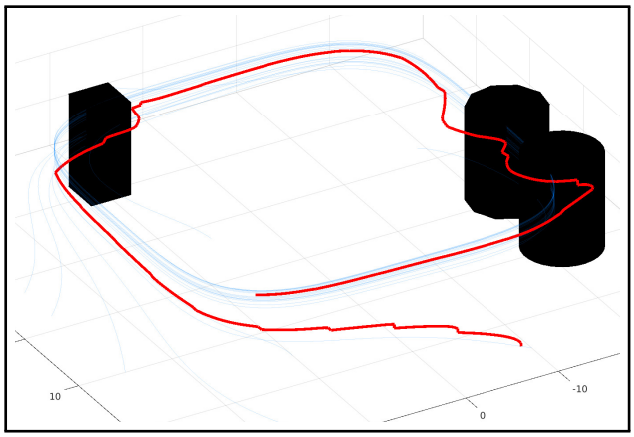

Fig. 5. 3D simulation of the proposed method for UAV navigation. The UAV follows a vector field that guides it towards a super-ellipse curve at a fixed height. The vector field, shown in blue, dictates the direction of movement but does not account for dynamic or unknown obstacles. The optimized trajectory, shown in red, is generated through optimization to ensure smoothness, obstacle avoidance, and adherence to the vector field.

compared to the framework to [8]. While the optimization-based method generated slightly longer paths, it significantly reduced computation times. Hybrid approaches that combine the strengths of both sample-based and gradient-based methods may enhance the versatility of the approach in handling more complex scenarios. The scalability of this framework to multi-robot systems could also be explored, where coordination between robots might be optimized using vector field guidance. A clear drawback of the approach is the fact that CHOMP-based solutions are not complete and may not return a feasible path. Future work could address this issue by utilizing a global planner, such as RRT*, just when such situations are detected.

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[2] B. Krogh, "A generalized potential field approach to obstacle avoidance control," in *Proc. SME Conf. on Robotics Research: The Next Five Years and Beyond, Bethlehem, PA, 1984*, 1984, pp. 11–22.

[3] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using laplace's equation," in *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, 1990, pp. 2102–2106.

[4] A. Shivam and A. Ratnoo, "Arcsine vector field for path following guidance," *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 12, pp. 2409–2420, 2023.

[5] V. M. Goncalves, L. C. Pimenta, C. A. Maia, B. C. Dutra, and G. A. Pereira, "Vector fields for robot navigation along time-varying curves in n-dimensions," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 647–659, 2010.

[6] A. M. Rezende, V. M. Goncalves, and L. C. Pimenta, "Constructive time-varying vector fields for robot navigation," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 852–867, 2021.

[7] A. H. Nunes, A. M. Rezende, G. P. Cruz, G. M. Freitas, V. M. Gonçalves, and L. C. Pimenta, "Vector field for curve tracking with obstacle avoidance," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2031–2038.

[8] G. A. Pereira, S. Choudhury, and S. Scherer, "A framework for optimal repairing of vector field-based motion plans," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 261–266.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[10] G. A. Pereira, S. Choudhury, and S. Scherer, "Nonholonomic motion planning in partially unknown environments using vector fields and optimal planners," in *XXI Congresso Brasileiro de Automática*, 2016.

[11] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International journal of robotics research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[12] S. Liu and P. Liu, "Robot motion planning benchmarking and optimization through motion planning pipeline," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 633–638.

[13] J. Men and J. R. Carrión, "A generalization of the chomp algorithm for uav collision-free trajectory generation in unknown dynamic environments," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2020, pp. 96–101.

[14] R. Bonatti, C. Ho, W. Wang, S. Choudhury, and S. Scherer, "Towards a robust aerial cinematography platform: Localizing and tracking moving targets in unstructured environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 229–236.

[15] W. Yao, H. G. de Marina, B. Lin, and M. Cao, "Singularity-free guiding vector field for robot navigation," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1206–1221, 2021.

[16] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 802–807.

[17] M. Faroni, N. Pedrocchi, M. Beschi *et al.*, "Adaptive hybrid local–global sampling for fast informed sampling-based optimal path planning," *Autonomous Robots*, vol. 48, no. 2–3, 2024.

[18] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.