SEMANTIC-AWARE TRAJECTORY PLANNING FOR UAV IN DYNAMIC ENVIRONMENTS

Submitted: 07th July 2025; accepted: 1st October 2025

Van Hung Nguyen, The Tien Nguyen, Tran Thang Le, Viet Hong Le

DOI: 10.14313/jamris-2026-011

Abstract:
Generating trajectories that leverage semantic information to guide a UAV safely and accurately to its destination in a dynamic environment remains an open problem. In the existing literature, semantics have been used to prioritize certain areas – either to guide the UAV through or to avoid them – for specific objectives, such as reducing errors in visual-inertial SLAM (VI-SLAM). However, prior work typically assumes a static environment when performing collision checking, even in cluttered and dynamic settings.

We propose a two-stage workflow: The first stage performs semantic-aware pathfinding. The second stage optimizes the resulting path, incorporating kinematic constraints and performing collision checking that accounts for obstacle motion, while still ensuring real-time performance.

To the best of our knowledge, this is the first approach that generates UAV trajectories by simultaneously leveraging semantic information and accounting for cluttered, dynamic environments. A summary video is available at https://youtu.be/I5w6AP7HThU.

Keywords: *Unmanned Aerial Vehicles, Trajectory Planning, Dynamic Obstacle Avoidance, Semantic-Aware, Dynamic environment*

1. Introduction

A higher level of autonomy in unmanned aerial vehicles (UAVs) expands their potential for deployment across various real-world applications. This autonomy relies heavily on simultaneous localization and mapping (SLAM) and the capability to generate safe and precise trajectories toward a target.

VI-SLAM systems are widely used in UAVs due to their high accuracy, real-time performance, and autonomy, especially in GPS-denied environments such as indoor spaces or obstructed areas [1–5]. Additionally, for quadrotors with limited payload capacity and battery life, cameras serve as ideal onboard sensors for navigation. However, a key drawback of VI-SLAM is its rapid decline in accuracy when encountering texture-less regions. A common approach to improving accuracy involves keeping specific features or landmarks within the field of view (FOV) [6–8].

Nowadays, the advancements in artificial intelligence (AI), particularly deep learning applied to semantic segmentation [9] and object detection [10], have achieved high accuracy and performance. These techniques enable to label the regions with different characteristics semantically. Semantic information is often incorporated as a term or constraint in optimization frameworks to help avoid textureless or problematic regions, such as lakes and oceans, which can cause significant drift or failures in pose estimation [11, 12]. Additionally, it helps prioritize high-textured regions, thereby improving the quality of pose estimation [13–19]. Moreover, semantics have also been employed in the multi-robot planning problem [20].

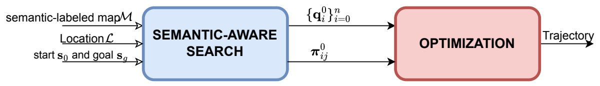
semantic-labeled map $\mathcal{M}$
Location $\mathcal{L}$
start $\mathbf{s}_0$ and goal $\mathbf{s}_g$
SEMANTIC-AWARE SEARCH
$\{\mathbf{q}_i^0\}_{i=0}^n$
$\boldsymbol{\pi}_{ij}^0$
OPTIMIZATION
Trajectory

Figure 1. Workflow of semantic-aware trajectory planning including two consecutive steps. They are semantic-aware search and optimization, described in section 4 and 5, respectively

Ensuring safe arrival at the destination also requires effective obstacle avoidance. However, many existing studies assume the environment is static during collision checking. This limits their deployment in real-world scenarios. Because the real world is exactly the cluttered and dynamic environment.

Common collision-checking methods involve decomposing free space into convex regions such as sequences of axis-aligned cubes [21], convex regions from seeding [22, 23], or creating safe flight corridors (SFC) by inflating pre-existing trajectories (often the global trajectory) [24–28]. Collision-checking can be performed using either discretizing the trajectory into points or outer polyhedral representations. While discretized points are computationally intensive and do not guarantee collision-free paths between sampled points, increasing the number of samples to improve accuracy further adds to the computational burden [29–32].

To reduce the burden of computation, outer representation techniques enclose the trajectory within a polyhedron. If this polyhedron remains inside the free space, the entire trajectory is considered collision-free. For example, in polynomial trajectory optimization [33, 34], it is verified whether the outer polyhedral representation of each trajectory segment is contained within the free space.

A common approach to obtaining this polyhedral representation is by using the convex hull of the control points from the Bernstein or B-Spline basis [35–37]. However, in cluttered and dynamic environments, the free space is significantly reduced. A more compact outer representation improves the likelihood of successful trajectory generation while reducing computational time.

Getting the idea from the prior publications [38–40], this work uses the MINVO basis [41] instead of the Bernstein or B-Spline basis. Depending on the polynomial degree n, MINVO [41] can yield a significantly smaller volume.

Decomposition is especially challenging in cluttered and dynamic environments. In dense environments, it is difficult to construct a tight representation of free space. In dynamic settings, an additional dimension of time makes the decomposition much more complicated, and sometimes it is infeasible. To eliminate the need for decomposition, this work imposes a constraint that verifies the existence of a separating plane between the UAV's trajectory and obstacle trajectories. This plane constraint is incorporated into the optimization process [38, 40].

This study presents a novel workflow that guides UAVs to prioritize high-texture areas while avoiding texture-less and hazardous regions in dynamic environments. The proposed approach consists of two main stages: (i) Semantic-Aware Trajectory Initialization, called *semantic-aware A* search*, prioritizes safe and high-texture areas while avoiding texture-less and hazardous regions. The output of this step serves as the initial guess for the second stage. (ii) Dynamic-Aware Optimization accounts for environmental dynamics by combining: (a) Eliminating free space decomposition and replacing it with a separating plane constraint. (b) Utilizing the MINVO basis. At the same time, the trajectory is also energy-optimal and satisfies dynamic constraints.

2. Problem declaration and solving approach

The UAV is modeled by the geometric shape and state at t. Its shape is a set of vertices in 3D space $\mathcal{V}^{\mathcal{U}} = [\mathbf{V}_0, \mathbf{V}_1, \ldots] \subset \mathbb{R}^3$. And the state vector $\mathbf{s}^T(t) = [\mathbf{x}^T, \dot{\mathbf{x}}^T, \ddot{\mathbf{x}}^T] = [\mathbf{x}^T, \mathbf{v}^T, \mathbf{a}^T]$, where $\mathbf{x}, \mathbf{v}$ and $\mathbf{a}$ are the position, velocity and acceleration, respectively.

The environment in which the UAV operates is a cluttered and dynamic environment. It is modeled by a metric-semantic map $\mathcal{M}$. It consists of unknown regions $\mathcal{M}_{unknown}$ and known regions $\mathcal{M}_{known}$. These known regions contain static obstacles $\mathcal{O}_{static}$, dynamic obstacles $\mathcal{O}_{dyn}$ and the regions which semantically-labelled as texture-high $\mathcal{M}_{att}$ or hazardous/texture-less $\mathcal{M}_{rep}$. So $\mathcal{M} = \mathcal{M}_{unknown} \cup \mathcal{O}_{dyn} \cup \mathcal{O}_{static} \cup \mathcal{M}_{att} \cup \mathcal{M}_{rep}$.

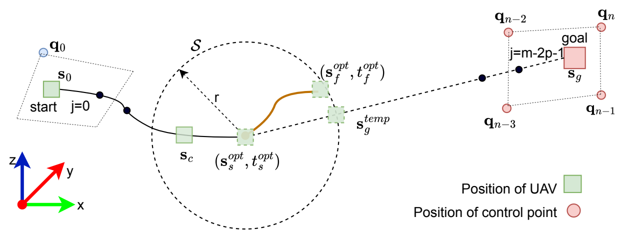

Figure 2. The problem of UAV trajectory planning in a cluttered and dynamic environment. The trajectory is generated portion by portion to ensure real-time performance and feasibility

At this time, the problem is how to generate a feasible trajectory that guides the UAV from the initial state $\mathbf{s}_0$ to the goal state $\mathbf{s}_g$ safely and accurately within the environment $\mathcal{M}$, while ensuring the minimization of control energy by leveraging the semantic information available in that semantic map.

To solve this problem, several key subproblems need to be addressed as follows:

- subproblem 1: Defining the trajectory.
- subproblem 2: HOW TO check collision in cluttered and dynamic environment even during trajectory generation. Details are presented in section 3 below.
- subproblem 3: HOW TO leverage semantic information to improve the process of trajectory generation for a certain purpose, more detailed in section 4.
- subproblem 4: Formulating and solving the programming optimization. It is described in more detail in section 5.

We use the method of polynomial trajectory planning [28, 33] with *clamped uniform* B-Splines. So, the UAV's trajectory $\mathbf{x}(t) := [\mathrm{x}(t), \mathrm{y}(t), z(t)]^T = \sum_{k=0}^{n} B_{k,p}(t)\mathbf{q}_k$ is defined by $n+1$ control points $\{\mathbf{q}_0, \ldots, \mathbf{q}_n\}$ and $m+1$ knots $\{t_0, t_1, ..., t_m\}$. Its each segment is a p-degree B-spline function and indexed by j ($j \in J$, J is the total number of intervals) starting from 0 (As described in Fig. 2, $j = 0, \ldots, m-2p-1$). In total, it has $m-2p-1$ intervals. It is *clamped* to ensure that it passes through $\mathbf{s}_0$ (the first $p+1$ knots are identical) and $\mathbf{s}_g$ (the last $p+1$ knots are identical). The knots between the first $p+1$ and the last $p+1$ knots are called internal knots. The *uniform* means that the internal knots are equally spaced.

In this paper, we use the cubic splines (i.e., $p = 3$). This balances the dynamic feasibility of a UAV and computational efficiency [40]. So, the input control $\mathbf{u}(t)$ is jerk $\mathbf{j}(t)$ and it is constant at the same interval $\mathbf{j}(j) = const$.

To ensure the real-time performance and feasibility, the consuming time of trajectory generation needs to be limited within a time interval of $\delta(t)$. This is achieved by generating only the portion of the trajectory (illustrated by Fig. 2, it is the golden-brown segment) that lies within a sphere $\mathcal{S}$ with the radius r. During re-planning, r remains fixed. The trajectory generation starts at a time when UAV is staying at $\mathbf{s}_c$ and at the moment before the time of completing the execution of the previous portion of the trajectory by $\delta(t)$ (illustrated in Fig. 2).

Thus, at this point, the starting point is the moment when the execution of the previous portion of the trajectory is completed t_s^{opt}, corresponding to state $\mathbf{s}_s^{opt}$. The goal of the trajectory is no longer $\mathbf{s}_g$ but instead a temporary target $\mathbf{s}_g^{temp}$. This temporary target is obtained by the intersection between sphere $\mathcal{S}$ and a piece-wise linear path that goes from $(\mathbf{s}_s^{opt}, t_s^{opt})$ that avoids the static obstacles. The final target of the resulting portion is $\mathbf{s}_f^{opt}$ at time t_f^{opt}, which does not necessarily coincide with $\mathbf{s}_g^{temp}$. Next, we delve into the details of solving the remaining subproblems.

3. Collision-checking in dynamic environment

As described in section 1, to alleviate the burden of computing, All of the trajectories are represented by outer polyhedrals. And then checking whether or not the intersection of them for collision.

3.1. UAV representation and the bounding polyhedron of its trajectory

The bounding polyhedron (or outer representation) of UAV's trajectory, which is defined in section 2, can be generated from its control points. They are indexed using the symbol l. The number of control points for each segment is one more than the degree of the B-spline, $p+1$. However, we use MINVO because, as demonstrated in [41], it provides a much tighter representation. In detail, with degree $n = 3$ shows that, the volume reduces 2.36 and 254.9 times smaller than the ones obtained by the Bernstein and B-Spline bases, respectively. When n = 7, these ratios increase to 902.7 and $2.997.10^{21}$, respectively.

Thus, from each interval of the B-spline, the control B-spline points Q_j^{BS} are computed, forming the set $\mathbf{Q}_j^{BS}$. From there, the MINVO control points are determined according to [42], resulting in the MINVO control points $Q_j^{MV} = f_{BS}^{MV}(Q_j^{BS})$ and their corresponding sets $\mathbf{Q}_j^{MV}$. As stated in section 2, we use the B-spline basis to be cubic (degree $p = 3$). Thus, each interval j is guaranteed to lie within the convex hull of its 4 control points $\{\mathbf{q}_j, \mathbf{q}_{j+1}, \mathbf{q}_{j+2}, \mathbf{q}_{j+3}\} \in Q_j^{MV}$.

3.2. Representation of obstacle and the bounding polyhedron of its trajectory

In the environment, there are I obstacles including static and dynamic ones. The *i-th* obstacle is symbolled by i ($i \in I$). An obstacle is characterized by its trajectory $\xi_i(t) := [\xi_x(t), \xi_y(t), \xi_z(t)]^T$ and dimension $\mathcal{V}_i^{\mathcal{O}} = \{\mathbf{v}_1^{\mathcal{O}}, \mathbf{v}_2^{\mathcal{O}}, \dots, \mathbf{v}_m^{\mathcal{O}}\} \subset \mathbb{R}^3$. Obviously, for static obstacles, $\xi_i(t) = const$. It is inflated by the size of UAV (that is Minkowski sum, the mathematical notation is $\oplus$) and then inferring the convex hull (mathematical notion is $conv(.)$) of inflated obstacle, $conv(\mathcal{V}_i^{\mathcal{O}} \oplus \mathcal{V}^{\mathcal{U}})$, as described in Fig. 4a.

For dynamic obstacles, its trajectory is predicted segment by segment with the prediction error α_{ij} (illustrated by Fig. 4b). Each segment of i-th obstacle's trajectory, $\xi_{ij}(t)$, is corresponding to a j-th interval time Δt_j of the UAV's trajectory. In this case, the convex hull is calculated as follows: First, it is inflated by the UAV's size, similar to a static obstacle. Next, it is expanded with the prediction error α_{ij}. Then, it is slid along its predicted trajectory with a sampling time of β_{ij}. The entire occupied space of the obstacle, $\mathcal{O}_{ij} = \mathcal{V}_i^{\mathcal{O}} \oplus \mathcal{V}^{\mathcal{U}} \oplus 2\alpha_{ij} \oplus 2\beta_{ij}$, is now the union of all occupied regions at each time step β_{ij}. Finally, the convex hull is generated for this occupied space. The set of all vertices of this convex hull, $\mathcal{C}_{ij} = conv(\mathcal{O}_{ij})$. All these steps are illustrated in Fig. 4b.

The problem now is how to predict the obstacle's motion trajectory. In the scope of this paper, the trajectory prediction function is assumed to be precomputed. Some particular obstacle's trajectories are used for simulation, with details provided in section 6.

3.3. Collision Checking

In subsection 3.2, the convex hull of the obstacles has already accounted for the UAV's size, so the UAV is now treated as a point of mass. This means that the UAV and the obstacle do not collide if the MINVO convex hull $\boldsymbol{\mathcal{Q}}_j^{\text{MV}}$ (computed in subsection 3.1) of the UAV's trajectory does not intersect with the convex hull of the obstacle $\boldsymbol{\mathcal{C}}_{ij}$ (computed in subsection 3.2).

$$\begin{cases} \mathbf{n}_{ij}^T \mathbf{c} + d_{ij} > 0, & \forall \mathbf{c} \in \boldsymbol{\mathcal{C}}_{ij}, \forall i \in I, j \in J \\ \mathbf{n}_{ij}^T \mathbf{q} + d_{ij} < 0, & \forall \mathbf{q} \in \boldsymbol{\mathcal{Q}}_j^{\text{MV}}, \forall j \in J \end{cases} \tag{1}$$

where:

- $\mathbf{n}_{ij}$ is the normal vector defining the separating hyperplane.
- d_{ij} is a bias term shifting the hyperplane.
- $\boldsymbol{\mathcal{C}}_{ij}$ is the convex region representing the obstacle.
- $\boldsymbol{\mathcal{Q}}_j^{\text{MV}}$ is the convex region representing the UAV trajectory (using the MINVO basis).
- I is the total number of obstacles.
- J is the set of UAV trajectory intervals.

In other words, they do not collide if there exists a separating hyperplane π_{ij} (characterized by the normal vector $\mathbf{n}_{ij}$ and bias d_{ij}) between their convex hulls, described by Eq.1. The first inequality of Eq.1 ensures that all points $\mathbf{c}$ in the obstacle set $\boldsymbol{\mathcal{C}}_{ij}$ lie on one side of the plane. The second one ensures that all points $\mathbf{q}$ in the UAV's trajectory set $\mathcal{Q}_j^{\text{MV}}$ lie on the other side.

An illustration of an outer polyhedral representation and collision-checking that includes static and dynamic obstacles, as well as UAV is shown in Fig. 3. Fig. 3a illustrates all of the convex hulls and collision checking at the first and second intervals. Fig. 3b and 3c illustrate the convex hull and collision checking at the third and fourth intervals, respectively. This problem is solved using GLPK [43] or Gurobi [44].

4. Semantic-aware A* search

This algorithm is inspired by the original A* [45] for path-finding based on not only traditional costs but also the semantic information about the environment. In particular, its resulting path is to prioritize the rich-informative regions while avoiding hazardous or low-informative areas. So it is called as semantic-aware A*. This is achieved by introducing additional cost values into the total cost function f. Each MINVO control point serves as a node in the search. All open nodes are maintained in a priority queue $openSet$, where elements are ordered in ascending order of f.

With semantic-labeled map $\mathcal{M}$, the regions with high information tend to attract the UAV, modeled by C_{att}. Whereas the hazardous or low-informative regions tend to repel the UAV, modeled by C_{rep} (described in Fig. 5).

$$f(.) = \lambda_{\text{g}} g(.) + \lambda_{\text{h}} h(.) + \lambda_{\text{att}} C_{\text{att}}(.) - \lambda_{\text{rep}} C_{\text{rep}}(.) \tag{2}$$

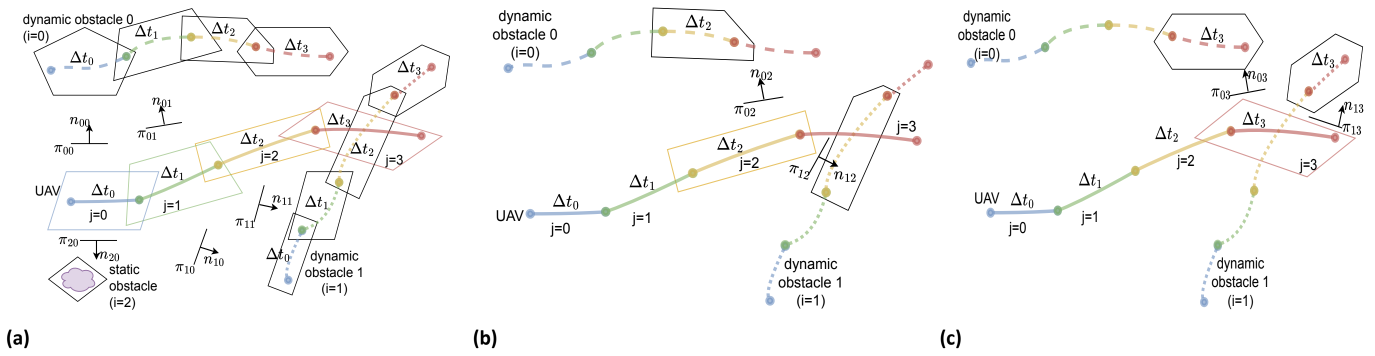

Figure 3. Outer representation of trajectories of UAV, dynamic obstacles (for instance, obstacle 0 and 1), static obstacle (for example, obstacle $i = 2$) and Collision-checking by separating planes π_{ij}: (a) Collision-checking at the 1^{st} and 2^{nd} interval of UAV trajectory. (b) Collision-checking at the 3^{rd} interval of UAV trajectory.(c) Collision-checking at the 4^{th} interval of UAV trajectory

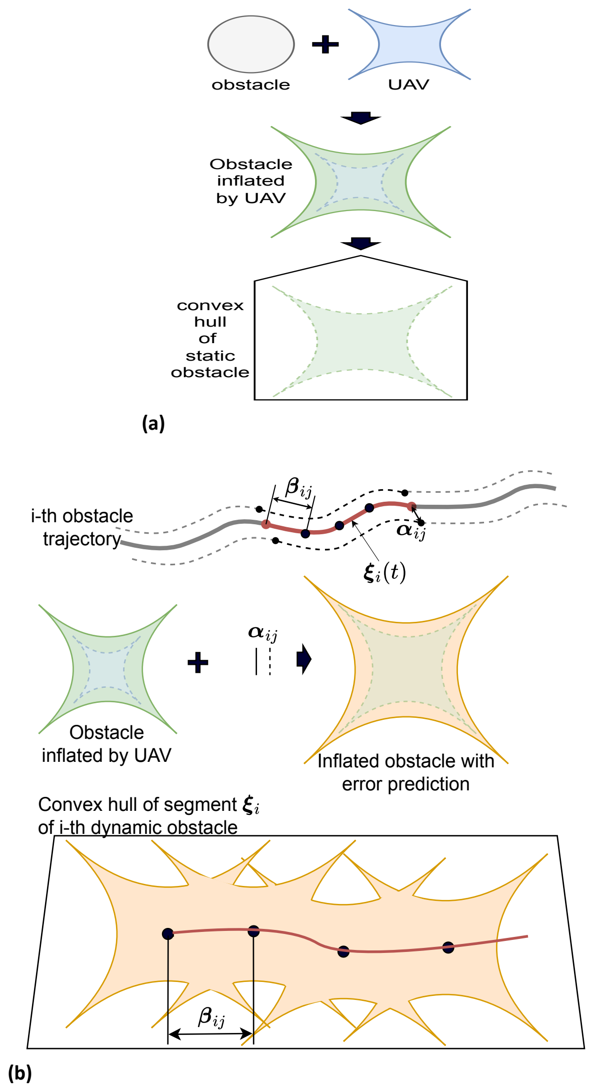
obstacle
UAV
Obstacle inflated by UAV
convex hull of static obstacle
(a)
i-th obstacle trajectory
β_{ij}
α_{ij}
$\xi_i(t)$
Obstacle inflated by UAV
Inflated obstacle with error prediction
Convex hull of segment ξ_i of i-th dynamic obstacle
(b)

Figure 4. Outer representation of the obstacles: (a) For static obstacle. (b) For dynamic obstacle

So, at this time, the total cost function $f(.)$ includes four terms, where $g(.)$ is the sum of the distances (between successive control points) from $\mathbf{q}_0$ to the current node $\mathbf{q}_l$ (cost-to-come), $h(.)$ is the distance from the current node $\mathbf{q}_l$ to the goal $\mathbf{s}_g$ (heuristics of the cost-to-go), $C_{\text{rep}}(.)$ is for repelling UAV away from the low-texture regions, while conversely, $C_{\text{att}}(.)$ is for attracting towards informative areas, as modeled in Eq. 2.

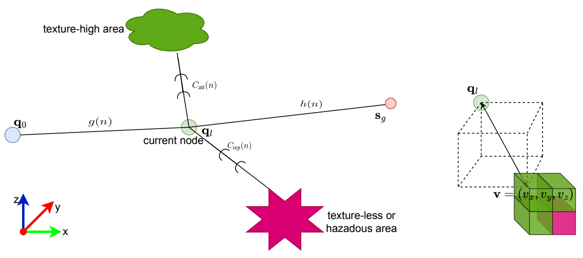

Figure 5. The total cost of semantic-aware A* includes traditional costs, along with repulsive and attractive costs arising from texture-less/hazardous regions (vivid pinkish-magenta color) and high-texture areas (fresh green color), respectively. Along with HOW TO calculate the Euclidean distance in 3D space used as primitive for calculating the costs

The position of each voxel in the map $\mathcal{M}$ is $\mathbf{v}^{vx} = (v_x^{vx}, v_y^{vx}, v_z^{vx})$, so $\mathbf{v}^{vx} \in \mathcal{M}$. Let $\mathcal{M}_{att}, \mathcal{M}_{rep} \subseteq \mathcal{M}$ be the set of all voxels in the attractive and repulsive regions, respectively. The cost value of these two regions ($\mathcal{M}_{att}$ and $\mathcal{M}_{rep}$) is calculated as the sum of the distances from the current control point $\mathbf{q}_l$ to all the voxels in that region.

However, their influence on the total cost (according to Eq. 2) is opposite. By increasing the total cost, C_{att} prioritizes regions with smaller distances. In contrast, C_{rep} decreases the total cost, thereby prioritizing regions with larger distances, meaning it tends to push the trajectory away from those regions. They are calculated by Eq. 3, where the sign "_" indicates att or rep.

$$C_{\_}(.) := \frac{1}{|\mathcal{M}_{\_}|} \sum_{\mathbf{v}^{vx} \in \mathcal{M}_{\_}} d(\mathbf{q}_l, \mathbf{v}^{vx}) \qquad (3)$$

The weights $\lambda_g, \lambda_h, \lambda_{\text{att}}, \lambda_{\text{rep}} \in \mathbb{R}^+$ in Eq. 2 determine the influence of each component on the total cost $f(\cdot)$. If a weight is greater than 1, the corresponding component has a stronger influence (for example, if $\lambda_{\text{att}} > 1$, the trajectory is more strongly guided toward the attractive region). On the other hand, when the weight is less than 1 ($0 < \lambda < 1$), the corresponding component has a weaker influence compared to the others. If the weight equals 1, it has a balanced influence according to its original value.

The algorithm is represented as a pseudo-code in Alg. 1. Firstly, control points $\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2$ is determined from the initial state $\mathbf{s}_s^{opt}$ at the moment t_s^{opt} (line 1). At the starting moment t_0, we have $t_s^{opt} = t_0$ and $\mathbf{s}_s^{opt} = \mathbf{s}_0$. And then it initializes the $openSet$ queue, $gCost$ and $fCost$ (line 2 to 5).

Algorithm 1 Semantic-Aware A*

```
1:  $(\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2) \leftarrow \text{CALCONTROLPOINT}(\mathbf{s}_s^{opt})$
2:  $openSet \leftarrow \mathbf{q}_2$
3:  $gCost, fCost \leftarrow \infty$
4:  $gCost[\mathbf{q}_0] \leftarrow 0$
5:  Calculating $fCost[\mathbf{q}_0]$ by Eq. 2 and 3
6:  while ($openSet$ is not empty) or timeout do
7:      $\mathbf{q}_l \leftarrow$ First item of $openSet$
8:      if $\|\mathbf{q}_l - \mathbf{s}_g^{temp}\|_2 < \epsilon$ and $l = n-2$ then
9:          $\{\mathbf{q}_i\}_{i=0}^{n-2} \leftarrow \text{GETSCPS}(\mathbf{q}_l)$
10:         $\mathbf{q}_{n-1} \leftarrow \mathbf{q}_{n-2}$
11:         $\mathbf{q}_n \leftarrow \mathbf{q}_{n-1}$
12:         return $[\{\mathbf{q}_i\}_{i=0}^{n}, \boldsymbol{\pi}_{ij}]$
13:     end if
14:     $openSet.\text{REMOVE}(\mathbf{q}_l)$
15:     $[isCols, \boldsymbol{\pi}_{ij}] \leftarrow \text{CHECKCOLLISION}(\mathbf{q}_l)$
16:     if $\|\mathbf{q}_l - \mathbf{s}_s^{opt}\|_2 > r$ or $\|\mathbf{q}_l - \mathbf{q}_k\|_\infty \leq \delta$ or
        $isCols$ then
17:         continue
18:     end if
19:     for $\Delta v \in \text{UNIFORMSAMPLING}(\mathbf{v}_{max}, \mathbf{a}_{max})$ do
20:         $\mathbf{q}_{l+1} \leftarrow \mathbf{q}_l + \frac{\Delta t . \Delta \mathbf{v}}{p}$
21:         $gCost_{temp} \leftarrow gCost[\mathbf{q}_l] + \text{DIST}(\mathbf{q}_l, \mathbf{q}_{l+1})$
22:         if $gCost_{temp} < gCost[\mathbf{q}_{l+1}]$ then
23:             $\mathbf{S}_{eq}[\mathbf{q}_{l+1}] \leftarrow \mathbf{q}_c$
24:             $gCost[\mathbf{q}_{l+1}] \leftarrow gCost_{temp}$
25:             Calculating $fCost[\mathbf{q}_{l+1}]$ by Eq. 2 and 3
26:             if $\|\mathbf{q}_{l+1} - \mathbf{q}_k\|_\infty > \delta$ then
27:                 $openSet \leftarrow \mathbf{q}_{l+1}$
28:             end if
29:         end if
30:     end for
31: end while
32: return $[\text{GETBESTSCPS}(\mathbf{S}_{eq}), \boldsymbol{\pi}_{ij}]$
```

The loop of semantic-aware A* search is run until $openSet$ queue is empty or out of time: The first item, which is with f value is lowest, is popped (line 7) and remove it (line 14) if it simultaneously does not touch the target (line 8) and satisfies some conditions (line 16). The search process is considered complete if $\mathbf{q}_l$ is within a predefined distance ϵ from the intermediate goal $\mathbf{s}_g^{temp}$ and it's index $l = n-2$ (line 8). Since the velocity $\mathbf{v}$ and acceleration $\mathbf{a}$ of the UAV are zero when it reaches the final target $\mathbf{s}_g$, it follows that $\mathbf{q}_{n-2} = \mathbf{q}_{n-1} = \mathbf{q}_n$. The result will be a sequence of control points $\{\mathbf{q}_i\}_{i=0}^{n}$, which come from inferring the sequence from $\mathbf{q}_0$ to $\mathbf{q}_{n-2}$ (line 9) backward and appending two ending points (line 10 and 11), and hyperplanes $\boldsymbol{\pi}_{ij}$ that separate them from obstacles.

For $\mathbf{q}_l$ node to be accepted, it must satisfy several conditions (line 16): it must be inside the sphere $\mathcal{S}$ ($\|\mathbf{q}_l - \mathbf{s}_s^{opt}\|_2 \leq r$) as stated in section 2, not be too close to another $\mathbf{q}_k$ already in the $openSet$ ($\|\mathbf{q}_l - \mathbf{q}_k\|_\infty > \delta$) to alleviate the burden of computing, and obviously not collide ($isCols$ is false). The collision is checked by determining which $\boldsymbol{\mathcal{Q}}_j^{\text{MV}}$ contains $\mathbf{q}_l$, then solving Eq. 1 (details in subsection 3.3).

The result is a hyperplane added to $\boldsymbol{\pi}_{ij}$ and a binary variable $isCols$ indicating whether or not a collision occurs (line 15).

Next, the best neighbor of $\mathbf{q}_l$ is expanded and added to $openSet$ queue: For each value of Δv, which is uniform-sampled ensuring the limits $\mathbf{v}_{\max}$ and $\mathbf{a}_{\max}$. The neighbor is computed (line 20) using a time step of $\Delta t/p$, where $\Delta t = \frac{\|\mathbf{s}_s^{opt} - \mathbf{s}_g^{temp}\|_2}{\mathbf{v}_{\max}}$. The best neighbor is selected, and if it is not already in the $openSet$, it is added (lines 22 to 27), while its $fCost$ value is evaluated (line 25). Moreover, if the search time exceeds the predefined limit (that is $timeout$), it will return the sequence of control points, which is with the last point being the closest to the intermediate goal $\mathbf{s}_g^{temp}$, and it's corresponding hyperplanes $\boldsymbol{\pi}_{ij}$ (line 32). This algorithm is tested in subsection 6.1 below.

5. Optimization

Based on the initial trajectory (converted from $[\{\mathbf{q}_i^0\}_{i=0}^{n}, \boldsymbol{\pi}_{ij}^0]$) getting from semantic-aware A*. We need to smooth and make it feasible by programming optimization with the goal of minimizing energy consumption and reaching the target as closely as possible. This problem is parameterized by control points $\mathbf{Q}_j^{\text{BS}}$ and planes variables $\boldsymbol{\pi}_{ij}(\mathbf{n}_{ij}, d_{ij})$. It minimizes the energy consumption through control input $\int_{t_s^{opt}}^{t_f^{opt}} \|\mathbf{u}(t)\|^2 dt = \int_{t_s^{opt}}^{t_f^{opt}} \|\mathbf{j}(t)\|^2 dt = \sum_{j \in J} \|\mathbf{j}(j)\|^2$. And the target is reached as closely as possible in terms of distance $\|\mathbf{q}_n - \mathbf{s}_g^{temp}\|_2$. Because of $\mathbf{q}_{n-2} = \mathbf{q}_{n-1} = \mathbf{q}_n$, it should be $\|\mathbf{q}_{n-2} - \mathbf{s}_g^{temp}\|_2$. This is modelled by Eq. 4.

$$\min_{\mathbf{Q}_j^{\text{BS}}, \mathbf{n}_{ij}, d_{ij}} \omega_u \sum_{j \in J} \|\mathbf{j}(j)\|^2 + \omega_g \left\| \mathbf{q}_{n-2} - \mathbf{s}_g^{temp} \right\|_2^2$$

subjects to:

$$\begin{aligned}
&\text{(i)} \quad \mathbf{s}_s(t_s^{opt}) = \mathbf{s}_s^{opt}, \\
&\text{(ii)} \quad \mathbf{v}(t_g^{opt}) = \mathbf{0}, \quad \mathbf{a}(t_g^{opt}) = \mathbf{0}, \\
&\text{(iii)} \quad \begin{cases} \mathbf{n}_{ij}^T \mathbf{c} + d_{ij} > 0, & \forall \mathbf{c} \in \boldsymbol{\mathcal{C}}_{ij}, \forall i, j, \\ \mathbf{n}_{ij}^T \mathbf{q} + d_{ij} < 0, & \forall \mathbf{q} \in \boldsymbol{\mathcal{Q}}_j^{\text{MV}}, \forall i, j, \end{cases} \\
&\text{(iv)} \quad \|\mathbf{q} - \mathbf{s}_s^{opt}\|_2^2 \leq r^2, \quad \forall \mathbf{q} \in \boldsymbol{\mathcal{Q}}_j^{\text{MV}}, \forall j, \\
&\text{(v)} \quad \begin{cases} |\mathbf{v}| \leq \mathbf{v}_{\max}, & \forall \mathbf{v} \in \mathcal{V}_j^{\text{MV}}, \forall j, \\ |\mathbf{a}_l| \leq \mathbf{a}_{\max}, & \forall l \in L \setminus \{n-1, n\} \end{cases}
\end{aligned} \tag{4}$$

This problem subjects to some constraints:

(i) The starting point of this inverval is the end-point of previous one. That is, $\mathbf{s}(t_s^{opt}) = \mathbf{s}_s^{opt}$ as described in detail in section 2. Obviously $\mathbf{s}_s(0) = \mathbf{s}_0$.

(ii) When $\mathbf{s}_g$ is inside the sphere $\mathcal{S}$, it means that the last segment of trajectory is optimized (that is, $j = m - 2p - 1$ or $\mathbf{s}(t_g^{opt}) = \mathbf{s}_g$). The UAV's velocity $\mathbf{v}(t_g^{opt}) = \mathbf{0}$ and acceleration $\mathbf{a}(t_g^{opt}) = \mathbf{0}$.

(iii) Ensuring the safety (collision-avoiding) in a dynamic environment, detailed in subsection 3.3.

(iv) Guaranteeing that the generated segment of trajectory remains inside sphere $\mathcal{S}$, detailed in section 2.

(v) The trajectory must comply with kinematic constraints. Specifically, velocity and acceleration must not exceed UAV's physical limits $\mathbf{v}_{max}$ and $\mathbf{a}_{max}$, respectively. Because the trajectory is represented by B-Splines, which is a continuous function of time. So directly imposing these constraints $(\mathbf{v}_{max}, \mathbf{a}_{max})$ at every single point in time along this continuous trajectory would lead to an infinite number of constraints, making the optimization problem computationally intractable.

Moreover, the control points $\mathbf{q}$ parameterize the entire trajectory segment. Therefore, by placing constraints on these control points (velocity $\mathbf{v}$ and acceleration $\mathbf{a}$), we can indirectly influence and bound the physical velocity and acceleration throughout the interval. The bound of the velocity $\mathbf{v}_{\max}$ and acceleration $\mathbf{a}_{\max}$ of control points are inferred the physical ones, respectively. This problem (Eq. 4) is solved by Gurobi [44].

6. Experiments

In the experiments, we use the configuration of the system as following:

- Hardware: 16 cores Intel Core i7-10875H @ 2.30GHz; GPU Nvidia TU117GLM [Quadro T1000 Mobile]; RAM memory of 32 GB.
- Software: Ubuntu 20.04 LTS; ROS noetic [46] serves as the middleware framework for communication between the planner, simulator, and visualization/logging tools. It provides standardized message passing and modular integration of different software components.

For all planning and collision checking, the UAV is modeled as a sphere of radius $r_{UAV} = 0.1$ (m) which upper-bounds the vehicle body and rotor sweep. While rotor disks are omitted in the figures for visual clarity, this inflated model guarantees safe clearance in all experiments. To guarantee real-time performance and feasibility, the planning horizon is limited to a sphere of radius $r = 4.0$ m, as detailed in Section 2.

Moreover, to focus on the trajectory planning algorithms, it is assumed that the UAV can perfectly track the trajectories generated by the planner.

6.1. Testing semantic-aware A* solely

This section evaluates the semantic-aware A* algorithm (presented in section 4) in two aspects: the influence of semantic information and the avoidance of dynamic obstacles with the following configuration: $\lambda_{\text{g}} = 1.0$, $\lambda_{\text{h}} = 1.0$, runtime = 0.1 second, degree of B-spline = 3 (cubic b-spline), number of segments = 6 and one trefoil-knot-based dynamic obstacle.

The trajectory of the dynamic obstacle, $\xi(t)$, is modeled as a trefoil knot [47,48]. Although the trefoil knot does not reflect realistic obstacle motion, it provides several advantages:

(i) Challenging yet structured – its 3D, non-trivial trajectory is more demanding than linear or circular paths, making it a strong test for collision avoidance.

(ii) Repeatable – its mathematical definition ensures identical, reproducible runs.

(iii) Controlled complexity – the trajectory is well understood, enabling systematic evaluation of algorithm performance.

(iv) Visually distinct – its clear shape facilitates observation and validation in simulations.

The first case considers no influence of semantic information with $\lambda_{\text{att}} = 0, \lambda_{\text{rep}} = 0$. In this case, the search process tends toward the goal while avoiding obstacles (Fig. 6a). In the second case, testing the algorithm in an environment containing a texture-high region (the green area in Fig. 6c and 6d) with $\lambda_{\text{att}} = 2.0, \lambda_{\text{rep}} = 0$. The experimental results in Fig. 6c show that the search process tends to shift toward the texture-high region compared to the first case. The third case examines the algorithm in an environment with a texture-less or unsafe region (the red area in Fig. 6e and 6f) with $\lambda_{\text{att}} = 0, \lambda_{\text{rep}} = 0.5$. The results indicate that the search process tends to avoid this region.

In all of these cases, the results in (Fig. 6b, 6d and 6f) show that dynamic obstacle avoidance is fully ensured at each segment (from 1^{st} to 6^{th} one). Because the obstacles are still considered to be in motion (no temporarily-static) during the collision-checking process. The experimental results (Figure 6) demonstrate that the proposed semantic-aware A* algorithm behaves as expected, showing a tendency to move toward regions rich in information while avoiding information-poor areas.

6.2. Experiment with cluttered and dynamic environment

In this section, we evaluate the system's capability to simultaneously leverage the semantic information – both to avoid and/or to prioritize traversing certain regions – while operating in a dynamic environment, by comparing it with the MADER system [38], which does not utilize semantic information. The evaluation is conducted in a dynamic environment measuring 70 x 4.0 x 4.0 meters (Fig. 7a and 7d), where 65% of the obstacles are dynamic – represented by red cubes, each sized 0.8 x 0.8 x 0.8 (m) – while the remaining 35% are static obstacles, depicted as blue rectangular boxes, each measuring 0.4 x 0.4 x 8.0 (m).

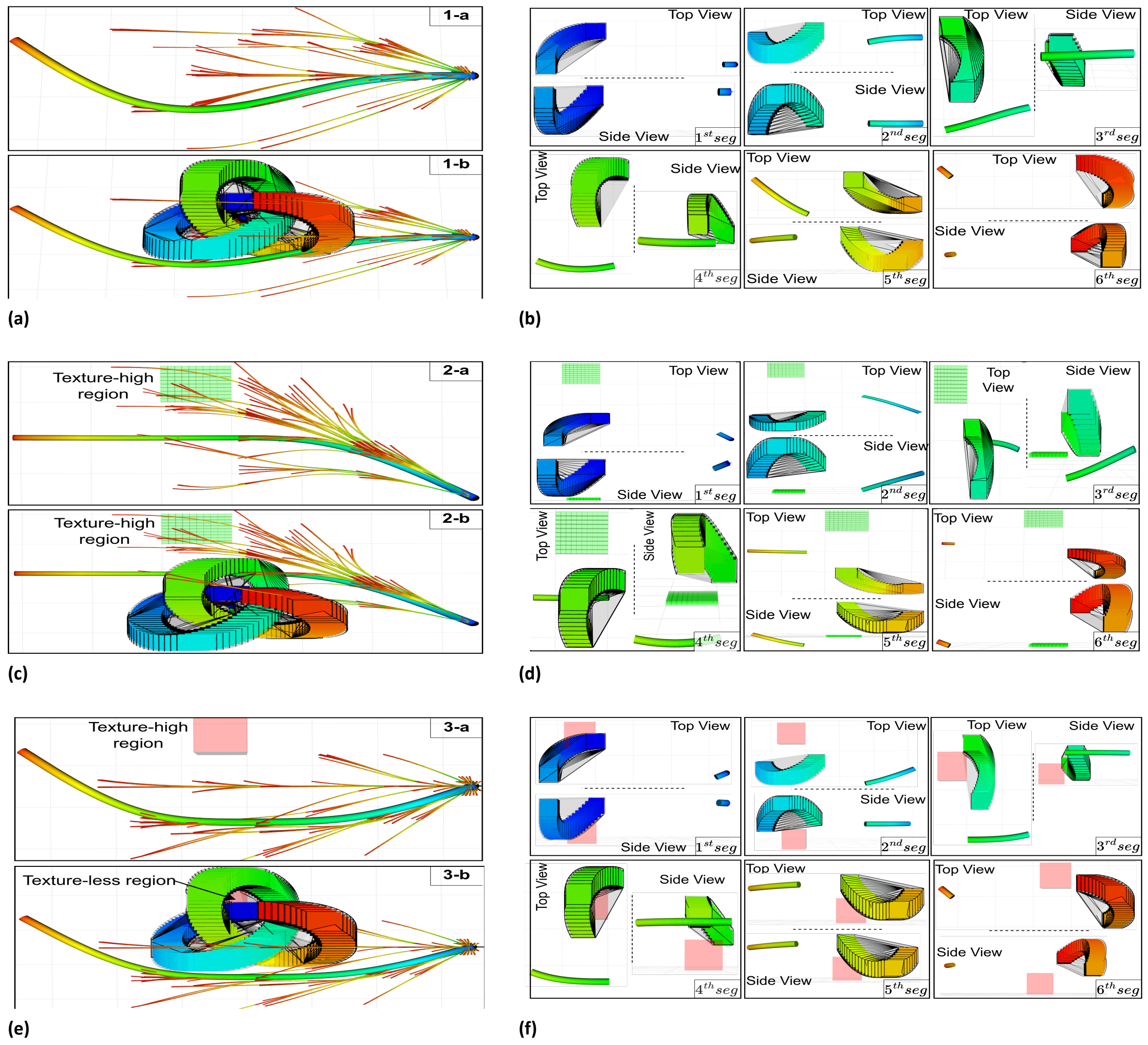

Figure 6. The experimental results of semantic-aware A* with one dynamic obstacle in three cases (Without semantics , Considering the influence of the attractive region and Considering the influence of the repulsive region): (a), (c), and (e) show the complete trajectories, while (b), (d), and (f) display the corresponding individual trajectory segments (1–6) for each case

Two simulation scenarios are conducted. The first features a dynamic environment containing a high-texture region, visually indicated by a green rectangle (Fig. 7a, 7b and 7c). The second includes a poor-texture or hazardous region, represented by a red rectangle. Both scenarios share the same UAV dynamic constraints, with a maximum velocity of $\mathbf{v}_{\max} = [6.0 \quad 6.0 \quad 6.0] \quad m/s$ and a maximum acceleration of $\mathbf{a}_{\max} = [20 \quad 20 \quad 10] \quad m/s^2$, corresponding to the velocity and acceleration limits, respectively. Additionally, in the first scenario, the goal position is set to $(75.0, \quad -10.0, \quad 1.0) \quad m$, whereas in the second scenario, it is set to $(75.0, \ -1.0, \ 1.0)\ m$. The runtime of the MILP phase is bounded between 0.05 and 0.35 second.

In the first scenario, simulations are conducted by MADER [38] and our suggestion ($\lambda_{att} = 2.0$), followed by a comparison of the resulting trajectories. The simulation results show that the UAV is capable of navigating safely in a dynamic environment (Fig. 7b), while also exhibiting a tendency to pass through the high-texture region (the dark blue trajectory in Fig. 7c).

Similarly, in the second scenario, considering the low-texture or hazardous area corresponds to $\lambda_{rep} = 2.0$. The resulting trajectory shows that the UAV tends to avoid these regions (the dark blue trajectory in Fig. 7f), while still successfully reaching the goal within the dynamic environment (Fig. 7e).

7. Conclusion

This work introduces a complete workflow for semantic-aware trajectory planning that enables UAVs to navigate autonomously in cluttered, dynamic environments while simultaneously exploiting semantic information to prioritize or avoid specific regions

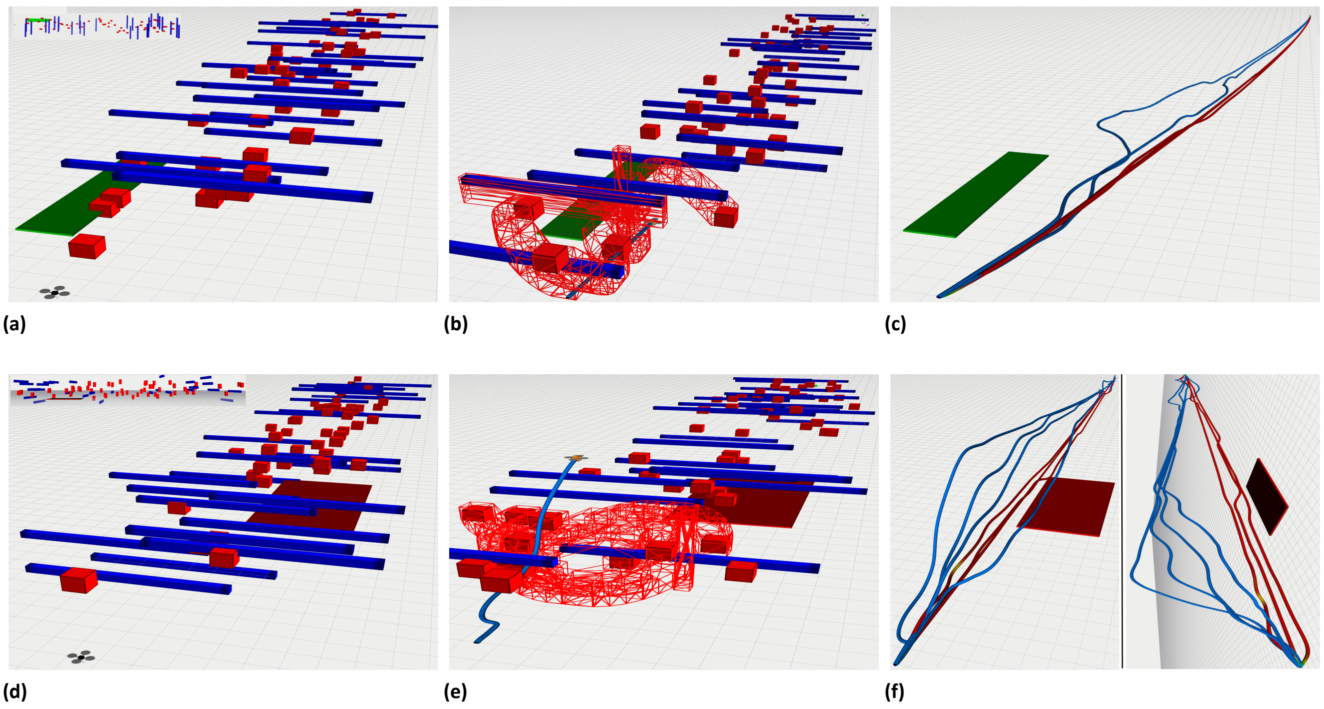

Figure 7. Experiments in dynamic corridor environment: (a) The environment includes dynamic and static obstacles (the red cube and the blue rectangular box, respectively), as well as a rich-information region (the green rectangular box); (b) The quadrotor flies in a dynamic environment, with outer polyhedra (red boxes) surrounding obstacles; (c) The dark blue and dark red trajectories come from our suggestion and MADER [38], respectively; (d-f) The corresponding setup, flight, and trajectory results for a scenario including a low-texture or hazardous region (red rectangular box)

depending on task objectives. Compared to non-semantic planners (such as MADER), which do not leverage semantics, our approach achieves more efficient and context-aware navigation, with trajectories that tend to avoid hazardous zones and approach information-rich regions.

The proposed approach combines a semantic-aware A* initialization, which biases trajectories toward safe and informative regions, with a dynamic-aware optimization that refines the path using separating plane constraints and the MINVO basis while ensuring energy efficiency and dynamic feasibility.

Nevertheless, several limitations remain. The optimization currently considers only the UAV's position and not its orientation, which may constrain applicability in tasks requiring viewpoint control. Our simulations also do not yet report quantitative state-estimation errors under semantic versus non-semantic settings, and dynamic obstacles are assumed to follow known trajectories rather than stochastic, uncertain motions.

Concrete directions for future work include extending the formulation to explicitly handle UAV orientation and perception-aware objectives, incorporating online estimation of dynamic obstacle motion with uncertainty, and validating the approach in hardware experiments with real UAVs. Another promising direction is to adapt or learn the semantic weights $(\lambda_{\text{att}}, \lambda_{\text{rep}})$ online using reinforcement or imitation learning, thereby tailoring behavior to specific missions while maintaining robustness.

Overall, this work represents a first step toward bridging high-level semantic understanding with low-level dynamic feasibility, paving the way for safer and more intelligent UAV autonomy in complex real-world environments.

AUTHORS

Van Hung Nguyen* – Control, automation in production and improvement of technology institute (CAPITI), Academy of Military Science and Technology 89 Ly Nam De Street, Hanoi city, Vietnam, e-mail: nvhung.v2k@gmail.com.

The Tien Nguyen – Le Quy Don Technical University 236 Hoang Quoc Viet, Hanoi city, Vietnam, e-mail: tiennt.isi@lqdtu.edu.vn.

Tran Thang Le – Control, automation in production and improvement of technology institute (CAPITI), Academy of Military Science and Technology 89 Ly Nam De Street, Hanoi city, Vietnam, e-mail: ltranthang@gmail.com.

Viet Hong Le – Control, automation in production and improvement of technology institute (CAPITI), Academy of Military Science and Technology 89 Ly Nam De Street, Hanoi city, Vietnam, e-mail: lehong0174@gmail.com.

*Corresponding author

References

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system", *IEEE transactions on robotics*, vol. 31, no. 5, 2015, 1147–1163, Publisher: IEEE.

[2] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras”, *IEEE transactions on robotics*, vol. 33, no. 5, 2017, 1255–1262, Publisher: IEEE.

[3] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM”, *IEEE Transactions on Robotics*, vol. 37, no. 6, 2021, 1874–1890, 10.1109/tro.2021.3075644, Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[4] D. Scaramuzza and Z. Zhang, “Visual-inertial odometry of aerial robots”, *arXiv preprint arXiv:1906.03289*, 2019, https://doi.org/10.48550/arXiv.1906.03289.

[5] C. Debeunne and D. Vivet, “A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping”, *Sensors*, vol. 20, no. 7, 2020, 10.3390/s20072068.

[6] I. Spasojevic, V. Murali, and S. Karaman, “Perception-aware time optimal path parameterization for quadrotors”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, 3213–3219.

[7] V. Murali, I. Spasojevic, W. Guerra, and S. Karaman, “Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness”, 2019, 3936–3943, 10.23919/ACC.2019.8814697.

[8] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, “PAMPC: Perception-aware model predictive control for quadrotors”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, 1–8.

[9] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, “A review on deep learning techniques applied to semantic segmentation”, *arXiv preprint arXiv:1704.06857*, 2017, https://doi.org/10.48550/arXiv.1704.06857.

[10] Joseph Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 779–788, 10.1109/CVPR.2016.91.

[11] L. Bartolomei, L. Teixeira, and M. Chli, “Perception-aware Path Planning for UAVs using Semantic Segmentation”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, 5808–5815, 10.1109/IROS45743.2020.9341347.

[12] S. Achat, J. Marzat, and J. Moras, “Path Planning Incorporating Semantic Information for Autonomous Robot Navigation”. In: *19th International Conference on Informatics in Control, Automation and Robotics (ICINCO) 2022*, Lisbonne, Portugal, 2022, 285–295, 10.5220/0011134300003271.

[13] I. Spasojevic, V. Murali, and S. Karaman, “Joint feature selection and time optimal path parametrization for high speed vision-aided navigation”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, 5931–5938.

[14] K. Lee, J. Gibson, and E. A. Theodorou, “Aggressive perception-aware navigation using deep optical flow dynamics and pixelmpc”, *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020, 1207–1214, Publisher: IEEE.

[15] Z. Zhang and D. Scaramuzza, “Perception-aware Receding Horizon Navigation for MAVs”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, 2534–2541, 10.1109/ICRA.2018.8461133.

[16] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, “Perception-aware path planning”, *arXiv preprint arXiv:1605.04151*, 2016, https://doi.org/10.48550/arXiv.1605.04151.

[17] K. M. Frey, T. J. Steiner, and J. P. How, “Towards online observability-aware trajectory optimization for landmark-based estimators”, *arXiv preprint arXiv:1908.03790*, 2019, https://doi.org/10.48550/arXiv.1908.03790.

[18] P. Salaris, M. Cognetti, R. Spica, and P. R. Giordano, “Online optimal perception-aware trajectory generation”, *IEEE Transactions on Robotics*, vol. 35, no. 6, 2019, 1307–1322, Publisher: IEEE.

[19] B. Zhou, J. Pan, F. Gao, and S. Shen, “Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight”, *IEEE Transactions on Robotics*, vol. 37, no. 6, 2021, 1992–2009, Publisher: IEEE.

[20] S. Kalluraya, G. J. Pappas, and Y. Kantaros, “Multi-Robot Mission Planning in Dynamic Semantic Environments”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, 1630–1637, 10.1109/ICRA48891.2023.10160344.

[21] J. Chen, T. Liu, and S. Shen, “Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, 1476–1483, 10.1109/ICRA.2016.7487283.

[22] R. Deits and R. Tedrake, “Computing large convex regions of obstacle-free space through semidefinite programming”. In: *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, 2015, 109–124.

[23] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, “Aggressive quadrotor flight through cluttered environments using mixed integer programming”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, 1469–1475, 10.1109/ICRA.2016.7487282.

[24] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments", *IEEE Robotics and Automation Letters*, vol. 2, no. 3, 2017, 1688–1695, 10.1109/LRA.2017.2663526.

[25] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning", *SIAM Journal on Control and optimization*, vol. 56, no. 4, 2018, 2712–2733, Publisher: SIAM.

[26] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments", *Journal of Field Robotics*, vol. 36, 2018, 10.1002/rob.21842.

[27] S.-p. Lai, M.-l. Lan, Y.-x. Li, and B. M. Chen, "Safe navigation of quadrotors with jerk limited trajectory", *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 1, 2019, 107–119, 10.1631/FITEE.1800719.

[28] G. Rousseau, C. S. Maniu, S. Tebbani, M. Babel, and N. Martin, "Minimum-time B-spline trajectories with corridor constraints. Application to cinematographic quadrotor flight plans", *Control Engineering Practice*, vol. 89, 2019, 190–203, https://doi.org/10.1016/j.conengprac.2019.05.022.

[29] M. Szmuk, C. A. Pascucci, and B. AÇikmeşe, "Real-time quad-rotor path planning for mobile obstacle avoidance using convex optimization". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, 1–9.

[30] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments", *IEEE Robotics and Automation Letters*, vol. 4, no. 2, 2019, 776–783, Publisher: IEEE.

[31] N. D. Potdar, G. C. de Croon, and J. Alonso-Mora, "Online trajectory planning and control of a MAV payload system in dynamic environments", *Autonomous Robots*, vol. 44, no. 6, 2020, 1065–1089, Publisher: Springer.

[32] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams". In: *2012 IEEE international conference on robotics and automation*, 2012, 477–483.

[33] C. Richter, A. Bry, and N. Roy. "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments". 649–666. April 2016.

[34] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, 434–440.

[35] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, 2019, 3529–3536.

[36] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "Faster: Fast and safe trajectory planner for navigation in unknown environments", *IEEE Transactions on Robotics*, vol. 38, no. 2, 2021, 922–938, Publisher: IEEE.

[37] J. A. Preiss, K. Hausman, G. S. Sukhatme, and S. Weiss, "Trajectory Optimization for Self-Calibration and Navigation.". In: *Robotics: Science and Systems*, vol. 13, 2017.

[38] J. Tordesillas and J. P. How, "MADER: Trajectory planner in multiagent and dynamic environments", *IEEE Transactions on Robotics*, vol. 38, no. 1, 2021, 463–476, Publisher: IEEE.

[39] J. Tordesillas and J. P. How, "PANTHER: Perception-Aware Trajectory Planner in Dynamic Environments", *IEEE Access*, vol. 10, 2022, 22662–22677, 10.1109/access.2022.3154037, Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[40] K. Kondo, R. Figueroa, J. Rached, J. Tordesillas, P. C. Lusk, and J. P. How, "Robust mader: Decentralized multiagent trajectory planner robust to communication delay in dynamic environments", *IEEE Robotics and Automation Letters*, vol. 9, no. 2, 2023, 1476–1483, Publisher: IEEE.

[41] J. Tordesillas and J. P. How, "MINVO basis: Finding simplexes with minimum volume enclosing polynomial curves", *Computer-Aided Design*, vol. 151, 2022, 103341, Publisher: Elsevier.

[42] K. Qin, "General matrix representations for B-splines". In: *Proceedings Pacific Graphics' 98. Sixth Pacific Conference on Computer Graphics and Applications (Cat. No. 98EX208)*, 1998, 37–43.

[43] GNU Project, Free Software Foundation, *GLPK – GNU Linear Programming Kit*, Free Software Foundation, 2025.

[44] Gurobi Optimization LLC. "Gurobi Optimizer Reference Manual", 2020.

[45] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, 1968, 100–107, 10.1109/TSSC.1968.300136.

[46] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*, vol. 3, 2009.

[47] M. C. Escher, F. Bool, J. Locher, and B. Ernst, "MC Escher: his life and complete graphic work: with a fully illustrated catalogue", *Harry N. Abrams*, 1982.

[48] E. W. Weisstein, *Trefoil Knot*, MathWorld, 2025.