ISSN 2007-9737

Vision–Language Navigation for Rapid UAV Deployment in Emergency Response

Carlos Osorio, Jose Martinez-Carranza*

Instituto Nacional de Astrofísica, Óptica y Electrónica
Computer Science Department, Puebla,
Mexico

{caoq, carranza}@inaoep.mx

Abstract. GPS-denied search-and-rescue missions require UAVs to navigate quickly through unfamiliar, cluttered, and dynamic environments while following high-level human instructions. This work presents a vision-and-language navigation framework that allows operators to issue free-form commands, such as "go to the far corridor, then approach the victim," without predefined waypoint maps. Unlike VLM methods that directly generate text actions, the proposed approach formulates navigation as a 2D spatial-grounding task. From a first-person RGB image and a natural-language instruction, the model iteratively selects image-grounded waypoint annotations as short-horizon navigation goals. These 2D waypoints are fused with the estimated travel distance to generate 3D displacement commands, while adaptive step-length control balances efficiency and safety in constrained spaces. The closed-loop system continuously replans under moving targets, changing visibility, and emerging obstacles. The framework is validated through simulation in high-fidelity GPS-denied environments representative of indoor SAR scenarios, achieving an online XY RMSE of 0.098 m. The results show improved performance over strong baselines across simulated rescue scenarios and ablation studies.

Keywords. Vision–Language Navigation (VLN), GPS-denied UAV autonomy, 2D/3D spatial grounding, Waypoint-based closed-loop control, Search-and-Rescue (SAR).

1 Introduction

Rapidly locating and assisting survivors in the aftermath of natural disasters is time-critical; yet, sending human responders into unstable, partially collapsed, or hazardous zones can be extremely risky. Unmanned aerial vehicles (UAVs) provide a practical alternative: they can be deployed quickly, operate at low cost, and reduce the exposure of first responders while supporting tasks such as reconnaissance [24], emergency response [11], victim search [32], and the delivery of essential supplies [15]. Modern UAVs can also leverage commercial off-the-shelf autopilots to execute waypoint missions with minimal operator workload [27], which is especially valuable when communications and situational awareness are limited.

A key limitation, however, is that most autonomous waypoint navigation pipelines rely heavily on Global Positioning System (GPS) measurements. In disaster environments, GPS can become unreliable or unavailable due to urban canyons [37], collapsed infrastructure [30], multipath and occlusions [9], indoor operation [14], or deliberate interference such as jamming and spoofing [4]. These conditions motivate robust GPS-denied navigation methods [1] that can maintain safe motion, interpret mission intent, and adapt online as the scene evolves.

Classical approaches address GPS denial through Simultaneous Localization and Mapping (SLAM) [8, 34] and related visual–inertial odometry (VIO) techniques [22], using onboard sensors (e.g., cameras, IMUs, LiDAR/sonar) to estimate pose while constructing a map for planning [38]. SLAM has enabled reliable navigation in many settings. Still, rescue deployments remain challenging for several reasons: (i) sensor noise

and perceptual degradation (smoke, dust, low light, textureless corridors) can corrupt maps and localization; (ii) real-time 3D mapping and planning can be computationally demanding on embedded hardware; and (iii) geometry-centric navigation often struggles to incorporate high-level semantic intent communicated by operators (e.g., "go to the far corridor, then approach the victim") without extensive task-specific training and hand-engineered logic.

Recent progress in large vision–language models (VLMs) [39] offers a complementary capability: these models combine a vision encoder (e.g., CNN/ViT features from the onboard RGB stream) with a language model to jointly reason over images and natural language [31]. This enables a UAV to interpret free-form instructions, recognize salient landmarks, and link language (corridor, doorway, person, exit) to what is visible in the scene [33, 20].

Building on this idea, the navigation strategy highlighted in the abstract reframes action selection not as text generation but as 2D spatial grounding: the VLM iteratively converts an instruction into image-space waypoint annotations [21].

Each grounded waypoint, together with an estimated travel distance, is transformed into a 3D displacement command for flight control. At the same time, an online step-length adaptation improves efficiency and safety in tight spaces. During execution, the system continuously updates its waypoint grounding using the latest RGB view, enabling iterative replanning for dynamic targets and changing environmental conditions. This feedback-driven operation supports adaptability and robustness, which are essential capabilities for search-and-rescue missions [32]. The main contributions of this work are summarized as follows:

— **2D Spatial Grounding for VLM-based UAV Navigation:** The proposed method formulates navigation as an image-grounded waypoint selection problem instead of direct text-action generation, enabling the UAV to connect natural-language instructions with visible regions in the camera view.

— **Closed-Loop Waypoint-to-Motion Control:** The framework converts VLM-selected 2D waypoints into 3D displacement commands using estimated travel distance and adaptive step-length control, supporting continuous replanning in GPS-denied rescue environments.

— **Integration of Adapted Navigation Modules:** Existing perception, waypoint parsing, smoothing, safety validation, and low-level control modules are adapted and integrated to support the reliable execution of VLM-grounded navigation commands.

— **Improvement Over Text-Action VLM Navigation:** By replacing direct textual action generation with spatially grounded waypoint reasoning, the framework improves interpretability, reduces command ambiguity, and provides a practical interface for free-form human instructions.

2 Related Work

Recent work has explored the use of large language models (LLMs) and vision–language models (VLMs) for robotic and UAV navigation, shifting from purely geometric planning toward semantic, instruction-aware autonomy. Existing studies can be grouped into three main directions: LLM-assisted planning, UAV vision–language navigation, and classical or learning-based motion planning. LLM-assisted planning methods use language models to interpret goals, decompose tasks, or select navigation actions [12, 18, 5, 6]. These approaches demonstrate that foundation models can encode high-level reasoning and user intent; however, many remain dependent on simplified environments, cloud-based inference, or predefined action libraries. As a result, they often treat the LLM as a supervisory module rather than evaluating its ability to generate executable waypoint sequences under consistent motion constraints.

UAV vision–language navigation has advanced through frameworks such as AerialVLN, UAV-VLN, and GRaD-Nav++, which connect visual perception, language grounding, and

navigation policy generation [20, 21, 33, 7]. These methods improve semantic understanding of aerial scenes and natural-language commands, but their evaluations are often benchmark-oriented or simulator-specific. In addition, many studies focus on route interpretation or instruction following, while fewer provide systematic comparisons against classical planners under identical environmental information, controller settings, smoothing procedures, and motion limits.

Classical and learning-based planners, including A*, RRT, Q-learning, and reinforcement-learning-based navigation, remain important baselines for UAV path planning [41, 13, 35, 17]. These methods are generally more interpretable and computationally predictable than VLM-based approaches, but they typically require explicit cost maps, hand-designed reward functions, or extensive training. Their adaptability to semantic mission instructions and changing task priorities is therefore limited.

Compared with prior work, the proposed framework evaluates VLM-guided waypoint generation under a unified experimental protocol. All methods are tested using the same obstacle information, controller, waypoint-smoothing strategy, and motion constraints, allowing differences in performance to be attributed primarily to the generated navigation strategy rather than unequal execution conditions. This work therefore complements existing UAV-VLN and LLM-planning studies by providing a controlled benchmark focused on obstacle-aware trajectory generation, path efficiency, and success-rate comparison in constrained 3D navigation scenarios.

3 VLM-Guided Path Planner

The VLM–guided path-planning approach investigates drone navigation using pretrained vision-language models (VLMs), with particular emphasis on their ability to infer collision-free flight paths from visual representations of the environment. VLMs—ChatGPT-4o [40], Gemini [10], Microsoft Copilot [25], Qwen [26], Claude 3.5 Sonnet [28], and Grok 3 [36] —were selected to assess the generative capabilities of multimodal models when presented with drone navigation scenarios in the form of annotated Gazebo simulation maps, including the start position P_s and goal position P_g.

Prompt engineering was used to extract waypoint-based trajectories from each model. In the few-shot prompting setting, each VLM was first provided with three example cases, where each example consisted of a visual map and a corresponding obstacle-free sequence of (x,y,z) waypoints. These examples helped the models capture spatial relationships and generalize safe navigation patterns. In the instruction-based prompting setting, the models received direct commands, such as: “Analyze the visual map and generate a safe path from start to goal while avoiding obstacles.” This approach leverages the multimodal reasoning capability of VLMs to transform visual scene information into coherent navigation plans.

Each VLM generated a sequence of 3D waypoints representing a hypothetical collision-free trajectory from the specified start location to the target goal. The raw waypoint output was then processed by a path interpolation module, which applies Euclidean interpolation to smooth the transitions between consecutive waypoints. A small step size of 0.1 m was adopted to provide fine spatial resolution and support physically feasible drone motion. This interpolation produces a denser set of intermediate waypoints while preserving the geometric intent of the original path. The same interpolation strategy was also applied around obstacles to obtain smoother and safer trajectories.

4 Framework and Methodology for VLM Navigation

This work proposes a method that follows a hybrid navigation pipeline for autonomous indoor drone operation in partially structured environments containing thin walls, corridor-like passages, and hazardous regions modeled as vertical cylinders. As shown in the architecture Fig. 1, the pipeline begins with global and local camera acquisition, where follow-view and top-down cameras provide RGB, depth, and mask information of the

Table 1. UAV VLN/VLA literature comparison: environment, core approach, and evaluation

Work	Environment / Platform	Core Approach and Models	Evaluation and Metrics
AerialVLN [21]	Simulated 3D city-scale environments (Unreal Engine + AirSim)	UAV VLN benchmark with cross-modal alignment navigation baselines; 4-DOF action space; large-scale instruction dataset.	Success Rate (SR), Navigation Error (NE), and human vs. baseline performance gap.
GRaD-Nav++ [7]	Photorealistic simulator (3DGS) and real UAV hardware.	Lightweight onboard Vision–Language–Action model (CLIP-based); differentiable RL; Mixture-of-Experts action head.	Task success, generalization to unseen tasks/environments, and sim-to-real transfer.
UAV-VLN [16]	Diverse simulation (Gazebo + ROS 2), indoor and outdoor.	End-to-end LLM+vision navigation; TinyLlama with YOLO, CLIP-Seg, Grounding-DINO.	SR, Success weighted by Path Length (SPL), and generalization to unseen settings.
OpenUAV [2]	OpenUAV simulation; continuous 6-DOF trajectories; 22 scenarios and 89 objects.	LLM-based assistant-guided object search; hierarchical multimodal LLM trajectory generation; graded assistant intervention levels.	SR, Object Search Rate (OSR), SPL, Navigation Error (NE), and human baseline trajectories.
Ours	Real UAV, laboratory setting.	VLM-based multimodal system with MiDaS depth; interactive UI; collaborative user-in-the-loop decision cycle.	Task success rate; explanation clarity/informativeness; user trust; confidence alignment; instruction approval rates.

Table 2. UAV VLN/VLA literature comparison: human-in-the-loop and distinctive features/gaps

Work	Human-in-the-Loop / Explainability	Distinctive Features / Gaps
AerialVLN	No explicit human-in-the-loop; emphasis on agent-only performance.	First large-scale UAV VLN benchmark; exposes difficulty of 3D long-horizon language tasks; limited to simulation and no analysis of user trust or explainability.
GRaD-Nav++	No explicit explainable outputs or human-facing interface.	Real-time fully onboard VLA and strong sim-to-real transfer; prioritizes efficiency and low-level control; lacks human-in-the-loop studies and language-level rationales.
UAV-VLN	No user study; instruction-following focus without transparency evaluation.	LLM-driven UAV control and cross-modal grounding; includes datasets and ablations over LLM/vision modules; limited human interaction and explainability evaluation.
OpenUAV	Provides graded “assistant” guidance; realism comparisons to human trajectories; does not quantify user trust or explanation quality	High-fidelity UAV VLN platform with flight dynamics and hierarchical planning; highlights performance gap to humans; limited user-facing trust/explainability analysis.
Ours	Yes: explicit approval/rejection; quantitative analysis of trust, transparency, and confidence calibration.	Real UAV deployment with systematic human-centric evaluation of explainability and trust; closes gap on user-facing validation and rationale-based interaction.

environment. These observations are processed in a perception layer, which extracts the visual and geometric cues required for navigation and scene understanding.

Next, the sensed information is organized into a scene model that represents structural walls, hazard locations, and an online 3D Gaussian Splatting (3DGS) map for local geometric refinement. Based on the global top-down observation, the system performs route selection, where a high-level decision is made between alternative corridors or motion strategies, such as upper, lower, or automatic route choice. This stage provides the coarse navigation intent rather than direct low-level control.

The selected route is then passed to the waypoint generation module, which converts the macro-route into a feasible waypoint sequence. This module also applies path repair and gating operations to improve safety and remove invalid or risky transitions between segments.

Once the waypoint set is constructed, the navigation process enters the local guidance stage, where the drone tracks the path using a combination of pure-pursuit style tracking, artificial potential field (APF) avoidance, and clearance-aware speed reduction. In this layer, attractive motion guides the drone toward the next local target, while repulsive and tangential components push the vehicle away from walls and hazards and help it move smoothly around corners and narrow passages.

At the final stage, the guidance commands are converted into yaw and velocity kinematic inputs and then applied to the PyBullet drone model [29]. Throughout execution, the drone continuously refreshes its perception and scene representation, thereby maintaining a closed loop between sensing, planning, and control. Within this methodology, the VLM serves as a high-level supervisory module for route reasoning and optional local steering assistance.

In contrast, the real-time flight dynamics remain governed by geometry-based waypoint tracking and obstacle avoidance. This hybrid formulation enhances interpretability, stability, and operational safety by clearly separating strategic scene understanding from low-level reactive control.

4.1 State-Space Elements and Environment Representation

Two primary obstacle classes represent the environment. First, fire or hazardous objects are modeled as vertical cylinders with the state vector [x, y, radius, height]. Second, walls are modeled as oriented bounding boxes (OBBs) with the row definition [c_x, c_y, h_x, h_y, H, yaw_{deg}, $base_z$], where c_x and c_y are the wall center coordinates, h_x and h_y are half-dimensions in the local frame, H is the height, yaw_{deg} is the in-plane rotation, and $base_z$ is the base elevation. The drone itself is modeled as a rigid body with mass m = 0.80 kg, gravity g = 9.81 m/s^2, principal inertias I_x = 0.0056 $kg \cdot m^2$, I_y = 0.0071 $kg \cdot m^2$, I_z = 0.0121 $kg \cdot m^2$, arm lengths l_1 = 0.14 m and l_2 = 0.12 m, drag coefficient k_d = 0.008, and a safety radius r_d = 0.26 m. In the navigation code, the drone radius serves as the core inflation term for all collision and clearance calculations; see Table 3.

The scenario is instantiated in PyBullet. The plane is loaded, the hazards are spawned as cylinders, the wall segments are spawned as fixed boxes, and the quadrotor URDF is inserted at the starting point. Start and goal markers are rendered as small colored spheres so that the global top-down VLM can infer a route directly from the scene image. This is a practical form of semantic prompting: the VLM sees the mission endpoints as image tokens instead of receiving only coordinates.

4.2 Geometric Scene Recovery for Vision-Language Navigation

The simulation model contains a complete projection and inverse-projection toolchain. PyBullet produces an RGBA image, a depth buffer, and a segmentation mask. We defined a converter that samples pixels from the OpenGL depth buffer and transforms them into 3D world points by applying the inverse of the product of the projection and view matrices. After homogeneous normalization, each valid pixel becomes a 3D point in world coordinates.

If a pixel at image location (u, v) has a depth value d in the normalized depth buffer, the corresponding normalized device coordinates are

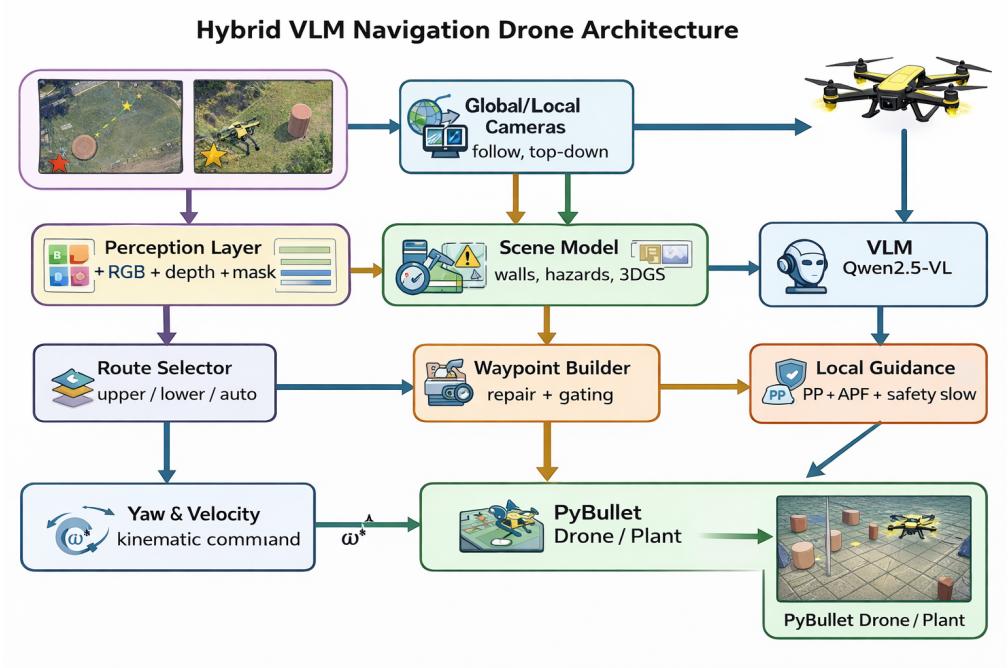

Fig. 1. High-level architecture of the VLM-assisted drone navigation pipeline

Table 3. Main state elements and structural variables

Element	State / Parameters	Role
Drone	$m, g, I_x, I_y, I_z, l_1, l_2, k_d, r_d$	Vehicle physics and safety radius
Cylinder hazard	$[x, y, \text{radius}, \text{height}]$	Fire/hazard obstacle in corridor
Wall OBB	$[c_x, c_y, h_x, h_y, H, \text{yaw}, z_0]$	Thin wall or structural segment
Waypoint	$[x, y, z]$	Reference path control point
Gaussian splat	$\mu, \sigma, \text{color}, \text{count}$	Online depth-derived soft obstacle

computed as $x_{ndc} = 2u/(W-1) - 1$, $y_{ndc} = 1 - 2v/(H-1)$, and $z_{ndc} = 2d - 1$.

The homogeneous clip-space point $[x_{ndc}, y_{ndc}, z_{ndc}, 1]^T$ is then multiplied by $(PV)^{(-1)}$, where P is the projection matrix, and V is the view matrix. The world point is obtained through homogeneous division.

This operation has two functions in the model. First, it enables online geometric reconstruction of the visible scene. Second, it provides a bridge between image-space sensing and metric-space avoidance.

Since the map is rebuilt from depth rather than from hand-coded occupancy alone, the architecture can be extended to unknown environments where walls and new obstacles are discovered online.

5 VLM-Based Route Planning Strategy

The VLM strategy is deliberately conservative, see algorithm 1. A global top-down image of the complete floor plan is captured before the mission. The prompt asks the model to output only a compact JSON object containing either: `{"route": "upper"}` or `{"route": "lower"}`. The `"upper"` route corresponds to a safer, corridor-like detour, while the “lower” route attempts a more direct path near the central hazard

region. The route selection is embedded inside RoutePolicyWrapper. The decision hierarchy is: if the user hard-codes a preference, use it; if recent clearance indicates danger, bias to the upper route; otherwise, accept the VLM suggestion; if no VLM is available, fall back to the automatic geometric route score.

This creates a hybrid neuro-symbolic decision policy where semantic image reasoning is filtered by deterministic safety logic. From a learning perspective, the VLM is not trained within this code. It is a pre-trained Qwen2.5-VL model [3] used in zero-shot or instruction-following mode. The "learning" therefore occurs in the sense of in-context policy shaping: the prompt, the symbols in the image, and the constrained JSON output produce a task-specific inference behavior without gradient updates. This is a useful strategy when data for end-to-end drone training is scarce, but a powerful multimodal prior is already available.

5.1 Waypoint Generation, Gating, and Repair

Once a route is chosen, the system generates control points. The upper template rises toward the top corridor before approaching the goal, whereas the lower template threads through a more central passage.

The resulting polyline is not accepted blindly. Each segment is tested by sampling points along it and checking geometric clearance against both cylinders and OBB walls. Let P be a sampled point on a candidate segment. The cylinder clearance is $d_{\text{cyl}}(P) = \|P - o\| - r_{\text{obs}} - r_d$, where o is the cylinder center in the XY plane, r_{obs} is the obstacle radius, and r_d is the drone radius. For a wall box, the signed distance is computed in the local OBB frame. If $q = \left[q_x, q_y\right]^T$ is the point expressed in the local frame, then the 2D rectangle signed distance is based on $\Delta_x = |q_x| - h_x$ and $\Delta_y = |q_y| - h_y$, with outside distance $\sqrt{\max(\Delta_x, 0)^2 + \max(\Delta_y, 0)^2}$ and inside $\min(\max(\Delta_x, \Delta_y), 0)$. The final wall clearance is the signed distance minus the drone radius.

If a segment is blocked, insert a lateral detour point. The midpoint of the blocked segment is shifted in both perpendicular directions with increasing offsets until a safe detour is found.

This is a lightweight local repair algorithm rather than a full graph search. Its advantage is simplicity and low cost; its limitation is that it cannot guarantee completeness in very cluttered scenes. For corridor navigation, however, it is highly practical.

5.2 Local Guidance Law and Dynamic Strategy

The active local control law is a hybrid of pure-pursuit path following and artificial potential field (APF) avoidance [19]. The pure-pursuit portion is implemented as an algorithm 2, which projects the current position onto the active path segment, then walks forward along the polyline by a lookahead distance. This target point reduces the "shock" that appears when the controller switches abruptly from one waypoint to the next.

The attraction term is generated as a proportional-derivative law: $v_att = k_p e(t) - k_d v(t)$, where $e(t) = p_{goal} - p$ is the position error and $v(t)$ is the current body velocity. The repulsion term is the sum of contributions from cylinders, walls, and optionally the 3DGS map. The final command is $v_{cmd} = v_{att} + s_{goalvrep}$, where $s_goal = 1 - exp(-(d_{goal}/r_{relax})^2)$ smoothly reduces obstacle dominance near the final goal region. For a nearby obstacle, the code uses an influence zone and a safety zone. If the obstacle distance is d, the safety distance is d_{safe} and the influence distance is d_{inf}, then the normalized urgency is $tau = clip((d_{inf} - d)/(d_{inf} - d_{safe}), 0, 1)$.

The repulsive velocity is proportional to tau^2 times the outward direction. A tangential velocity is also added to produce wall-following behavior instead of deadlock. The tangent sign is selected according to the cross-product sign between obstacle direction and goal direction, which encourages bypassing on the side that better preserves progress toward the goal. The local controller in Algorithm 1 executes after the global waypoint generation stage of the VLM navigation framework described in Algorithm 2. In this way, the VLM module provides the reference path, while the hybrid pure-pursuit and APF controller performs smooth tracking and local obstacle avoidance.

Algorithm 1: Hybrid VLM Navigation with Geometric Safety

Input: scenario S, start state $\mathbf{x}_s$, goal state $\mathbf{x}_g$, hazards $\mathcal{O}$, walls $\mathcal{W}$, control parameters Θ
Output: executed path $\mathcal{P}$, clearance metrics, RMSE metrics

Initialize the PyBullet scenario and load the drone at $\mathbf{x}_s$;
Spawn start/goal markers and capture a global top-down image;
Query the VLM for route $r \in \{\texttt{upper}, \texttt{lower}\}$; if unavailable, use automatic route scoring;
Build route control points and repair blocked segments;
Set the active polyline and initialize RMSE memory and 3D Gaussian Splatting memory;
while *goal not reached* ***and*** *time budget not exceeded* **do**
  Read the drone pose $\mathbf{p}$, velocity $\mathbf{v}$, and camera data;
  if *scheduled update is active* **then**
    Unproject depth observations and update the Gaussian splat map;
  end
  Compute the current clearance from cylindrical hazards, wall segments, and nearby Gaussian splats;
  Find a look-ahead target on the active polyline;
  Compute the attraction velocity $\mathbf{v}_{att}$ from position and velocity errors;
  Compute the repulsion/tangential velocity $\mathbf{v}_{rep}$ from all active obstacles;
  Form the nominal command

$$\mathbf{v}_{cmd} = \mathbf{v}_{att} + s_{goal}\,\mathbf{v}_{rep}$$

  if *clearance is below the safety threshold* **then**
    Query the local VLM for steering correction and blend the commands;
  end
  Apply clearance-based speed reduction, temporal smoothing, and slew-rate limiting;
  Compute the yaw command so that the drone faces the motion direction;
  Step the PyBullet simulation and append the new state to the path log;
  Update RMSE, render displays, and record output video if enabled;
end
Return the final metrics, executed path, and diagnostic information;

5.3 Vision-Language Steering Assistance and Command Fusion

During local flights, the VLM can also provide a steering bias from an ego-centered top-down image captured around the drone. The prompt asks the model to return JSON of the form $\{$`"steer_deg": 45, "speed_scale": 1.0`$\}$.

The steering angle rotates the normalized goal direction in the XY plane, and the speed scale compresses the commanded speed near complicated areas.

If g_{xy} is the normalized goal direction and $_{vlm}$ is the steering correction, then the VLM-guided direction is $g^{'}_{xy} = R(_{vlm})g_{xy}$, where $R(_{vlm})$ is the 2D rotation matrix. A VLM command velocity is then built as $v_{vlm} = [g^{'}_{xy} \cdot v_{max} \cdot s_{vlm}, z_{term}]$. The fused command is $v_m ix = (1 - w_{vlm}) \cdot v_{cmd} + w_{vlm} \cdot v_{vlm}$, where w_{vlm} is the VLM blending weight.

In the final code, the VLM is invoked only every few control steps and only when the measured clearance is below a threshold.

This is a sensible strategy because VLM calls are expensive and should be reserved for ambiguous local geometry.

The system, therefore, behaves as a triggered advisory module rather than as a continuous end-to-end controller.

This design improves computational efficiency and reduces the risk of excessive oscillation caused by frequent language-model steering updates.

Algorithm 2: Hybrid Pure-Pursuit + APF Local Control

Input: $\mathbf{p}, \mathbf{v}, \mathcal{P}, \mathcal{O}, k_p, k_d, d_{\text{safe}}, d_{\text{inf}}, r_{\text{relax}}$
Output: $\mathbf{v}_{cmd}$

$\mathbf{p}_{goal} \leftarrow \texttt{progress_target_xy}(\mathbf{p}, \mathcal{P})$;
$\mathbf{v}_{att} \leftarrow k_p(\mathbf{p}_{goal} - \mathbf{p}) - k_d\mathbf{v}$;
$\mathbf{v}_{rep} \leftarrow 0$;
foreach $o \in \mathcal{O}$ **do**
  $d \leftarrow \texttt{distance}(\mathbf{p}, o)$;
  if $d < d_{inf}$ **then**
    $\tau \leftarrow \text{clip}\left(\frac{d_{\text{inf}} - d}{d_{\text{inf}} - d_{\text{safe}}}, 0, 1\right)$;
    $\hat{\mathbf{n}}_{out} \leftarrow \texttt{unit_vector_away_from}(o, \mathbf{p})$;
    $\hat{\mathbf{t}} \leftarrow \texttt{tangential_direction}(o, \mathbf{p})$;
    if $cross(\mathbf{d}_{obs}, \mathbf{d}_{goal}) < 0$ **then**
      $\hat{\mathbf{t}} \leftarrow -\hat{\mathbf{t}}$;
    end
    $\mathbf{v}_{rep} \leftarrow \mathbf{v}_{rep} + \tau^2(\hat{\mathbf{n}}_{out} + \hat{\mathbf{t}})$;
  end
end
$s_{goal} \leftarrow 1 - \exp\left[-\left(\frac{\|\mathbf{p}_{goal} - \mathbf{p}\|}{r_{\text{relax}}}\right)^2\right]$;
$\mathbf{v}_{cmd} \leftarrow \mathbf{v}_{att} + s_{goal}\mathbf{v}_{rep}$;
return $\mathbf{v}_{cmd}$;

5.4 3D Gaussian Splatting Map

The Gaussian Splat Map improves navigation by providing an online depth-derived local scene representation that complements the predefined wall and hazard model.

Instead of relying solely on raw depth points, the method aggregates observations into voxel-based Gaussian primitives, each storing a mean position, spread, color, and observation count.

This representation smooths noisy depth data, preserves short-term geometric memory, and enables efficient clearance estimation and local obstacle avoidance.

During navigation, nearby splats are treated as soft obstacles that contribute repulsive and tangential guidance terms, thereby improving safety in narrow corridors, corners, and partially observed regions.

As a result, the drone benefits from more robust local mapping, better clearance-aware speed adaptation, and more stable navigation behavior.

The Gaussian-based implementation can improve navigation in the simulated scenario in several ways:

— The wall boxes and hazard cylinders are predefined. That is useful, but it assumes the environment is already known. The Gaussian Splat Map gives the drone a way to build a local perception-based map from what the camera currently sees. In simulation, this means the drone can react not only to hand-defined obstacles, but also to geometry reconstructed from depth.

— Raw depth points are dense and noisy. If the controller used all points directly, avoidance would be unstable. By averaging points inside each voxel, the splat map creates a smoother representation Eq. 1:

$$\mu_k = \frac{n_k \cdot \mu_k + p}{n_k + 1}, \tag{1}$$

where p is a new observed point and n_k is the current observation count of voxel k. This makes the local map more stable over time and reduces sudden jumps in obstacle locations.

— Each splat behaves like a small, soft obstacle with a radius related to its Gaussian spread $r_{splat} \approx 3\sigma$; then, the drone can compute local clearance as $d_{clear} = \|p - \mu\| - r_{splat} - r_d$, where p is the drone position and r_d is the drone safety radius.

5.5 Error Metrics

For each drone position, the minimum squared distance to the full waypoint polyline is computed. If d_i is the minimum distance for the i-th sample, then the online root mean square error is $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2}$. At the end of the mission, we also report MSE, RMSE, a 95th-percentile error, and maximum error. These metrics are useful because waypoint tracking error alone does not reveal the safety margin, while clearance alone does not reveal smoothness. Algorithmically, the full flight loop can be summarized as follows. First, initialize the drone, map, route, and waypoints.

Second, at each control step, estimate the state, update 3DGS if needed, compute the current clearance, select a lookahead target along the polyline, generate attraction and avoidance components, optionally mix in a VLM steering suggestion, apply speed and yaw smoothing, and step the simulator. Third, render, log, and update the online RMSE. The resulting controller operates at the configured $control_{hz}$ while rendering at a lower $render_{hz}$.

To evaluate the effectiveness of each pretrained language model for drone path planning, we use three primary metrics: Success Rate (SR), Average Completion Time (ACT), and Path Length (PL), together with the derived Efficiency–Success Score (ESS). SR represents the percentage of trials in which the drone reaches the goal without collision, thereby reflecting reliability and obstacle-avoidance capability. ACT measures the average time required to complete a successful navigation task, indicating operational efficiency and trajectory smoothness. However, considering these metrics independently can be misleading; for instance, a model may complete the task quickly but exhibit a low success rate, or it may achieve high success at the cost of excessive completion time. To better capture this trade-off, we define ESS as a composite metric Eq. 2:

$$ESS = \frac{1}{2} SRx \left(\frac{T_{ref}}{ACT} + \frac{L_{ref}}{PL} \right). \tag{2}$$

6 Experiments and Results

6.1 Simulation Setup

For these experiments, the simulation tool shown in Fig. 2 was developed to evaluate pretrained foundation models for prompt-based autonomous drone trajectory generation. Tests were conducted in a constrained indoor 3D environment with static obstacles, as summarized in Table 3. Twenty independent scenarios were generated, each repeated 100 times, resulting in 2000 trials. Obstacles occupied 20–30% of the scenario area and were randomly distributed while preserving clearance from the start position P_s and goal position P_g. The drone used the default Iris model with a PX4-based controller. For each trial, the VLM received P_s, P_g, and obstacle positions $\{P_{o,i}\}_{i=1}^{N_o}$ as input. Generated waypoints were validated, refined, and interpolated at 0.5 m resolution. The controller operated at 50 Hz with a 5–10 m/s velocity limit. Success required reaching P_g without collision, timeout, boundary violation, or invalid VLM output.

6.2 Evaluation Metrics

Table 4 shows that the proposed method achieves the best overall performance for VLM-based navigation in scenarios with ≥ 3 obstacles. To ensure a fair comparison, all methods were evaluated using the same environmental information, controller, smoothing strategy, and motion constraints. Therefore, the observed improvement can be mainly attributed to the quality of the generated waypoint sequence and the navigation strategy, rather than to differences in controller tuning, obstacle representation, or post-processing. The proposed method obtains the shortest path length, 11.47 m, and the highest ESS, 1.260, while maintaining a 100% success rate. In addition, its average completion time of 80 s is lower than that of all other compared methods, indicating faster goal completion, more efficient navigation, and better trajectory quality.

Among the baseline VLMs, Llama-3.1-8B-Instruct performs best, with 90 s ACT, 16.1 m PL, 100% SR, and an ESS of 1.111. The Qwen-based models show lower efficiency, especially in path length and ESS, and Qwen-2.5-coder-32B also drops to an 80% success rate. Compared with classical planners, the proposed method also outperforms A*, RRT, and Q-learning, demonstrating that it provides a better balance between speed, safety, and navigation efficiency in cluttered environments.

Table 5 shows the contribution of each component in the proposed framework. The APF-only configuration provides basic obstacle avoidance but achieves the weakest performance, with longer paths, higher completion times, and lower success rates due to local-minimum and clutter-related limitations.

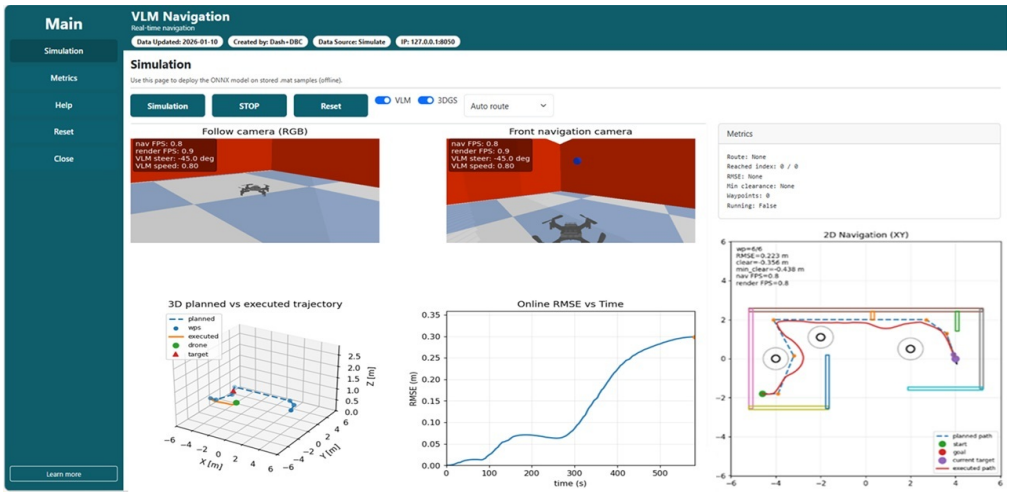

Fig. 2. VLM Navigation simulator interface for evaluating autonomous drone trajectory generation, navigation performance, and obstacle-aware path execution in a simulated environment

Table 4. Comparative evaluation of VLM models and baseline planners over obstacle-avoidance scenarios with ≥ 3 obstacles. Values are reported as estimated mean $\pm$ standard deviation over $N = 2000$ trials

Model	ACT (s) ↓	PL (m) ↓	SR (%) ↑	ESS ↑
Llama-3.1-8B-Instruct [23]	90.0 ± 7.8	16.10 ± 1.80	100.0 ± 0.0	1.111 ± 0.110
Qwen-2.5-coder-7B	102.0 ± 9.5	18.20 ± 2.10	100.0 ± 0.0	0.980 ± 0.130
Qwen-2.5-coder-32B	87.0 ± 11.6	15.50 ± 2.40	80.0 ± 40.0	0.920 ± 0.160
A* [41]	87.0 ± 4.5	15.60 ± 1.10	100.0 ± 0.0	1.115 ± 0.070
RRT [13]	84.0 ± 7.2	15.00 ± 1.80	90.0 ± 30.0	1.071 ± 0.120
Q-learning [35]	87.0 ± 6.0	15.60 ± 1.30	100.0 ± 0.0	1.115 ± 0.090
Proposed method	80.0 ± 5.2	11.47 ± 0.92	100.0 ± 0.0	1.260 ± 0.075

Adding the VLM improves global waypoint selection, reducing both ACT and PL while increasing SR and ESS. The complete APF + VLM + 3DGS configuration achieves the best overall results, with the shortest path length, lowest completion time, 100% success rate, and highest ESS. This indicates that 3DGS improves spatial representation and obstacle awareness, while the VLM enhances high-level route reasoning and the APF module supports local safety refinement.

6.3 Reproducibility and Implementation Details

The APF–VLM–3DGS framework is documented with explicit implementation and experimental settings to improve reproducibility. Experiments were performed in obstacle-avoidance scenarios with at least three obstacles and averaged over $N = 2000$ episodes using ACT, PL, SR, and ESS as evaluation metrics. The ablation study compares APF only, APF + VLM, and the full APF + VLM + 3DGS configuration. Results show that adding VLM guidance improves semantic route selection, reducing ACT from 88.5 s to 84.2 s and increasing SR from 92.0% to 98.0%. The complete framework achieves the best performance, reducing PL from 15.85 m to 11.47 m and reaching 100.0% SR.

For reproducibility, the manuscript reports the VLM prompt (see Fig. 3), controller frequency, planner update rate, APF parameters, obstacle-clearance thresholds, environment configuration, obstacle distribution policy,

Table 5. Proposed planning framework over obstacle-avoidance scenarios with ≥ 3 obstacles. Values are reported as estimated mean $\pm$ standard deviation over $N = 2000$ trials

Configuration	ACT (s) ↓	PL (m) ↓	SR (%) ↑	ESS ↑
APF only	88.5 ± 6.8	15.85 ± 1.45	92.0 ± 27.1	1.045 ± 0.105
APF + VLM	84.2 ± 5.9	13.65 ± 1.18	98.0 ± 14.0	1.165 ± 0.088
APF + VLM + 3DGS	80.0 ± 5.2	11.47 ± 0.92	100.0 ± 0.0	1.260 ± 0.075

PX4 settings, VLM query scheduling, and inference latency.

The VLM is queried every K control steps and additionally triggered by waypoint arrival, low obstacle clearance, progress stagnation, or new obstacles detected by the 3DGS map. These details support independent replication and fair comparison with future UAV navigation frameworks. The VLM prompt is provided to improve reproducibility. It guides the model to analyze a top-down navigation map, choose the safer route between two alternatives, and output only a valid JSON object for direct parsing by the navigation controller.

6.4 VLM Navigation for Search-and-Rescue Drones

The obtained results indicate that the proposed VLM-navigation framework is well-suited for drone-based search-and-rescue missions. The system successfully performs route generation and trajectory adaptation in structured obstacle fields and in the presence of close obstacles, showing effective scene understanding and obstacle-aware decision-making. The low and bounded online RMSE further confirms stable navigation performance. Overall, these findings highlight the framework's capacity to provide autonomous, safe, and adaptable navigation for SAR drones operating in complex and dynamic environments.

Moreover, the framework offers key advantages for real search-and-rescue missions, where drones must navigate under uncertainty, with limited map information and strict time constraints. By integrating semantic scene understanding with route-level decision-making, the VLM-based method improves efficiency in tasks such as survivor search, hazardous-area inspection, and target reaching in obstructed environments. Its ability to maintain stable navigation while adapting to environmental changes can reduce operator workload and increase mission autonomy.

7 Limitations and Future Challenges

Although the proposed method achieved the best overall performance in Table 4, several limitations remain. The approach obtained the lowest average completion time, 80.0 ± 5.2 s, the shortest path length, 11.47 ± 0.92 m, and a 100% success rate. However, these results were obtained under controlled simulation conditions using the same environmental information, controller, smoothing strategy, and motion constraints for all methods. Therefore, broader validation in more complex and realistic environments is still required.

A key limitation is visual grounding. Since the VLM generates waypoints from image and prompt information, incorrect associations between obstacles, free space, and target regions may produce unsafe or inefficient paths. This risk increases in cluttered scenes with occlusions, overlapping objects, or low visual contrast. Similarly, VLM hallucinations may generate missing waypoints, unreachable goals, unsafe routes, or incorrectly formatted outputs. Although waypoint parsing, validation, interpolation, and refinement reduce these risks, stronger safety filters, feasibility checks, and fallback planners are needed.

The framework is also sensitive to environmental conditions such as illumination changes, motion blur, shadows, smoke, dust, and low-texture regions. These factors may reduce perception reliability and limit direct transfer to real UAV search-and-rescue missions.

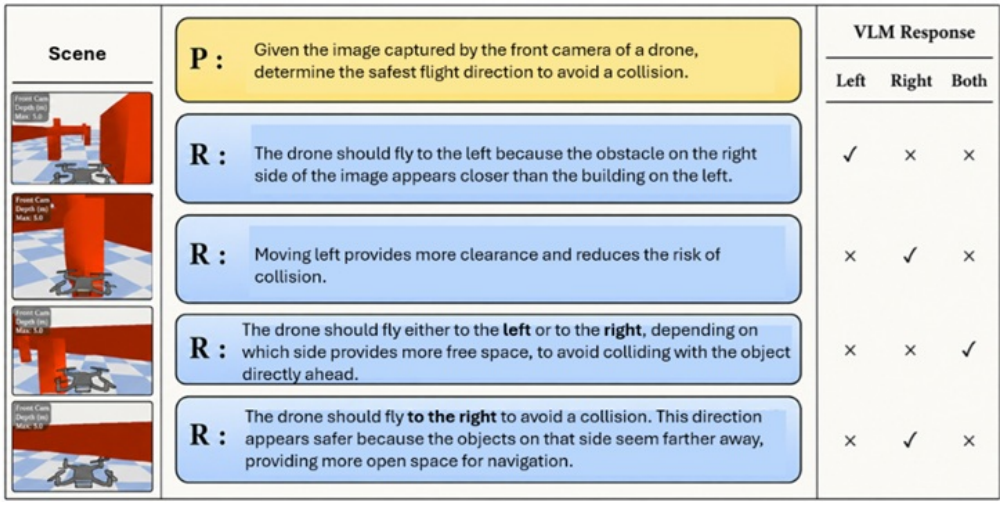

Fig. 3. VLM-based UAV collision-avoidance evaluation using front-camera images, prompt–response reasoning, and directional decision labels: left, right, or both

Integrating depth cameras, LiDAR, thermal imaging, or multi-view perception could improve robustness.

The APF-based local refinement strategy may suffer from local minima, oscillations, and narrow-passage failures in highly cluttered spaces. Computational latency from VLM inference, waypoint validation, smoothing, and control execution may also affect real-time replanning, especially for onboard deployment. Future work should explore hybrid planning, lightweight VLMs, edge acceleration, asynchronous replanning, and real-time safety monitoring.

8 Conclusion

This work presented a foundation-model-guided framework for drone path planning and a systematic benchmark of multiple VLM-based strategies. The results showed that pretrained foundation models, especially VLMs with visual reasoning capabilities, can generate feasible waypoint sequences and support obstacle-aware global planning in autonomous drone navigation. Under controlled simulation conditions, the proposed method achieved shorter paths, lower completion time, and reliable success rates compared with baseline methods.

These findings suggest that VLM–guided planning is a promising direction for practical UAV autonomy; however, physical flight experiments were not conducted in this study. Therefore, real-world feasibility, robustness, and deployment reliability remain topics for future validation. Future work will investigate physical UAV experiments, dynamic and unstructured environments, and computational optimization for real-time onboard implementation.

Acknowledgments

The first author is thankful to the National Council for Technological and Scientific Research (SECIHTI) No. CVU: 661331.

References

1. **Alghamdi, S., Alahmari, S., Yonbawi, S., Alsaleem, K., Ateeq, F., Almushir, F. (2025).** Autonomous navigation systems in GPS-denied environments: A review of techniques and applications. 2025 11th International Conference on Automation, Robotics, and Applications (ICARA), pp. 290–299. DOI: 10.1109/ICARA64554.2025.10977619.

2. **Anand, H., Rees, S. A., Chen, Z., Poruthukaran, A. J., Bearman, S., Prasad Antervedi, L. G., Das, J. (2021).** OpenUAV cloud testbed: A collaborative design studio for field robotics. 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), pp. 724–731. DOI: 10.1109/CASE49439.2021.9551638.

3. **Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., Lin, J. (2025).** Qwen2.5-VL technical report. arXiv preprint arXiv:2502.13923.

4. **Burbank, J., Caleb, T., Andam, E., Kaabouch, N. (2026).** Detection and mitigation of cyber attacks on UAV networks. Electronics, Vol. 15, No. 2. DOI: 10.3390/electronics15020317.

5. **Chen, G., Yu, X., Ling, N., Zhong, L. (2025).** Typefly: Low-latency drone planning with large language models. IEEE Transactions on Mobile Computing, Vol. 24, No. 9, pp. 9068–9079. DOI: 10.1109/TMC.2025.3561282.

6. **Chen, H., Tang, Y., Tsourdos, A., Guo, W. (2025).** Contextualized autonomous drone navigation using LLMs deployed in edge-cloud computing. 2025 International Conference on Machine Learning and Autonomous Systems (ICMLAS), pp. 1373–1378. DOI: 10.1109/ICMLAS64557.2025.10967934.

7. **Chen, Q., Gao, N., Huang, S., Low, J., Chen, T., Sun, J., Schwager, M. (2025).** GRaD-Nav++: Vision-language model enabled visual drone navigation with gaussian radiance fields and differentiable dynamics.

8. **Dai, Z. (2024).** A review of the development of visual SLAM. 2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC), pp. 915–919. DOI: 10.1109/ICAIRC64177.2024.10899961.

9. **Delamer, J.-A., Watanabe, Y., Chanel, C. P. (2021).** Safe path planning for UAV urban operation under GNSS signal occlusion risk. Robotics and Autonomous Systems, Vol. 142, pp. 103800. DOI: https://doi.org/10.1016/j.robot.2021.103800.

10. **Dong, A., Choot, X., Seow, B., Trinh, R., Wang, Z.**, **.** Large Language Model-Enabled Multimodal UAV for Rescue. pp. . DOI: 10.2514/6.2025-106062.

11. **Farhan, M., Eesaar, H., Ahmed, A., Chong, K. T., Tayara, H. (2025).** Transforming highway safety with autonomous drones and AI: A framework for incident detection and emergency response. IEEE Open Journal of Vehicular Technology, Vol. 6, pp. 829–845. DOI: 10.1109/OJVT.2025.3549387.

12. **Gopee, M. A., Prieto, S. A., García de Soto, B. (2026).** Self-hosted multimodal large language models for speech-driven perception and navigation in construction robotics. Automation in Construction, Vol. 183, pp. 106805. DOI: https://doi.org/10.1016/j.autcon.2026.106805.

13. **Hu, B., Cao, Z., Zhou, M. (2021).** An efficient RRT-based framework for planning short and smooth wheeled robot motion under kinodynamic constraints. IEEE Transactions on Industrial Electronics, Vol. 68, No. 4, pp. 3292–3302. DOI: 10.1109/TIE.2020.2978701.

14. **Katkuri, A. V. R., Madan, H., Khatri, N., Abdul-Qawy, A. S. H., Patnaik, K. S. (2024).** Autonomous UAV navigation using deep learning-based computer vision frameworks: A systematic literature review. Array, Vol. 23, pp. 100361. DOI: https://doi.org/10.1016/j.array.2024.100361.

15. **Khelfa, C., Drias, H., Khennak, I. (2024).** Multi-objective sma for medical supply delivery using drones and generative ai in post-disasters. 2024 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), pp. 1–7. DOI: 10.1109/ICT-DM62768.2024.10798949.

16. **Kim, S., Park, J., Yun, J.-K., Seo, J. (2020).** Motion planning by reinforcement learning for an unmanned aerial vehicle in virtual open space with static obstacles. 2020 20th International Conference on Control, Automation and Systems (ICCAS), pp. 784–787. DOI: 10.23919/ICCAS50221.2020.9268253.

17. **Kulhánek, J., Derner, E., de Bruin, T., Babuška, R. (2019).** Vision-based navigation using deep reinforcement learning. 2019 European Conference on Mobile Robots (ECMR), pp. 1–8. DOI: 10.1109/ECMR.2019.8870964.

18. **Latif, E. (2024).** 3p-LLM: Probabilistic path planning using large language model for autonomous robot navigation.

19. **Lee, M. C., Park, M. G. (2003).** Artificial potential field based path planning for mobile robots using a virtual obstacle concept. Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), Vol. 2, pp. 735–740 vol.2. DOI: 10.1109/AIM.2003.1225434.

20. **Liu, S., Zhang, H., Qi, Y., Wang, P., Zhang, Y., Wu, Q. (2023).** AerialVLN: Vision-and-language navigation for UAVs. 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 15338–15348. DOI: 10.1109/ICCV51070.2023.01411.

21. **Liu, S., Zhang, H., Qi, Y., Wang, P., Zhang, Y., Wu, Q. (2023).** Aerialvln: Vision-and-language navigation for uavs. International Conference on Computer Vision (ICCV).

22. **Luo, H., Li, G., Zou, D., Li, K., Li, X., Yang, Z. (2023).** UAV navigation with monocular visual inertial odometry under GNSS-denied environment. IEEE Transactions on Geoscience and Remote Sensing, Vol. 61, pp. 1–15. DOI: 10.1109/TGRS.2023.3323519.

23. **Luthardt, S., Willert, V., Adamy, J. (2018).** LLama-SLAM: Learning high-quality visual landmarks for long-term mapping and localization. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2645–2652. DOI: 10.1109/ITSC.2018.8569323.

24. **Moon, S., Jeon, J., Kim, D., Kim, Y. (2023).** Swarm reconnaissance drone system for real-time object detection over a large area. IEEE Access, Vol. 11, pp. 23505–23516. DOI: 10.1109/ACCESS.2022.3233841.

25. **Moradi Dakhel, A., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., Jiang, Z. M. J. (2023).** Github copilot ai pair programmer: Asset or liability?. Journal of Systems and Software, Vol. 203, pp. 111734. DOI: https://doi.org/10.1016/j.jss.2023.111734.

26. **Nunes, D., Amorim, R., Ribeiro, P., Coelho, A., Campos, R. (2025).** A framework leveraging large language models for autonomous UAV control in flying networks. 2025 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), pp. 12–17. DOI: 10.1109/MeditCom64437.2025.11104434.

27. **Ortner, R., Kurmi, I., Bimber, O. (2021).** Acceleration-aware path planning with waypoints. Drones, Vol. 5, No. 4. DOI: 10.3390/drones5040143.

28. **Palla, D., Slaby, A. (2025).** Evaluation of generative ai models in python code generation: A comparative study. IEEE Access, Vol. 13, pp. 65334–65347. DOI: 10.1109/ACCESS.2025.3560244.

29. **Panerati, J., Zheng, H., Zhou, S., Xu, J., Prorok, A., Schoellig, A. P. (2021).** Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. .

30. **Panigati, T., Zini, M., Striccoli, D., Giordano, P. F., Tonelli, D., Limongelli, M. P., Zonta, D. (2025).** Drone-based bridge inspections: Current practices and future directions. Automation in Construction,

Vol. 173, pp. 106101. DOI: https://doi.org/10.1016/j.autcon.2025.106101.

31. **Patten, T., Jacobs, P. (1994).** Natural-language processing. IEEE Expert, Vol. 9, No. 1, pp. 35–. DOI: 10.1109/64.295134.
32. **Quero, C. O., Martinez-Carranza, J. (2025).** Unmanned aerial systems in search and rescue: A global perspective on current challenges and future applications. International Journal of Disaster Risk Reduction, Vol. 118, pp. 105199. DOI: https://doi.org/10.1016/j.ijdrr.2025.105199.
33. **Saxena, P., Raghuvanshi, N., Goveas, N. (2025).** UAV-VLN: End-to-end vision language guided navigation for UAVs. 2025 European Conference on Mobile Robots (ECMR), pp. 1–6. DOI: 10.1109/ECMR65884.2025.11163198.
34. **Sun, C.-Z., Zhang, B., Wang, J.-K., Zhang, C.-S. (2021).** A review of visual SLAM based on unmanned systems. 2021 2nd International Conference on Artificial Intelligence and Education (ICAIE), pp. 226–234. DOI: 10.1109/ICAIE53562.2021.00055.
35. **Wu, J., Sun, Y., Li, D., Shi, J., Li, X., Gao, L., Yu, L., Han, G., Wu, J. (2023).** An adaptive conversion speed Q-learning algorithm for search and rescue UAV path planning in unknown environments. IEEE Transactions on Vehicular Technology, Vol. 72, No. 12, pp. 15391–15404. DOI: 10.1109/TVT.2023.3297837.
36. **xAI (2025).** Grok 3 Beta: The age of reasoning agents. Accessed: 2026-06-07.
37. **Xiong, S., Liu, H. H.-T. (2024).** Dop-based drift correction control for uavs operating in urban canyon. IEEE/ASME Transactions on Mechatronics, Vol. 29, No. 4, pp. 3109–3116. DOI: 10.1109/TMECH.2024.3402628.
38. **Xu, W., Lin, Z., Wang, W. (2022).** A localization and trajectory planning method for UAVs with visual-inertial odometry. 2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 710–715. DOI: 10.1109/AIM52237.2022.9863253.
39. **Zhang, J., Huang, J., Jin, S., Lu, S. (2024).** Vision-language models for vision tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 46, No. 8, pp. 5625–5644. DOI: 10.1109/TPAMI.2024.3369699.
40. **Zhang, Z., Hu, C., Lye, S., Lv, C. (2025).** A VLM-drone system for indoor navigation assistance with semantic reasoning for the visually impaired. 2025 IEEE/SICE International Symposium on System Integration (SII), pp. 1260–1265. DOI: 10.1109/SII59315.2025.10871009.
41. **Zhou, Q., Liu, G. (2022).** UAV path planning based on the combination of A-star algorithm and RRT-star algorithm. 2022 IEEE International Conference on Unmanned Systems (ICUS), pp. 146–151. DOI: 10.1109/ICUS55513.2022.9986703.

Article received on 11/04/2026; accepted on 18/07/2026.
** Corresponding author is Jose Martinez-Carranza.*