RESEARCH ARTICLE

VLM-Nav: Mapless UAV navigation using monocular vision driven by vision-language models

Gobinda Chandra Sarker[1,2], **AKM Azad**[3], **Sejuti Rahman**[2,4], **Md Mehedi Hasan**[2*]

1 Department of Electrical and Computer Engineering, University at Albany, Albany, New York, United States of America, **2** Department of Robotics and Mechatronics Engineering, University of Dhaka, Dhaka, Bangladesh, **3** Department of Mathematics and Statistics, Faculty of Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia, **4** Department of Computer Science, New Uzbekistan University, Tashkent, Uzbekistan

* mmhasan@du.ac.bd

Check for updates

OPEN ACCESS

Citation: Sarker GC, Azad A, Rahman S, Hasan MM (2026) VLM-Nav: Mapless UAV navigation using monocular vision driven by vision-language models. PLoS One 21(4): e0345778. https://doi.org/10.1371/journal.pone.0345778

Editor: Hassan Elahi, National University of Sciences and Technology, PAKISTAN

Received: September 11, 2025

Accepted: March 10, 2026

Published: April 3, 2026

Copyright: © 2026 Sarker et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data availability statement: All relevant data and materials are fully available without restriction. The source code for the VLM navigation framework, AirSim simulation configuration files, the trained depth estimation model, and the simulation results are publicly accessible at https://github.com/gcsarker/vlm_nav. The

Abstract

Autonomous vehicles, such as Unmanned Aerial Vehicles (UAVs), have the potential to completely reshape various industries such as parcel delivery, agriculture, surveillance, monitoring, and search-and-rescue missions. Consequently, the demand for safe, cost-effective, and intelligent navigation systems is crucial to ensure reliable performance in complex and dynamic environments. In this study, we propose a novel vision-based UAV navigation method that integrates depthmap estimation with a Vision-Language Model (VLM) for efficient obstacle avoidance and path planning. The system processes RGB images captured by the UAV, transforming them into depth maps using DepthAnything-V2, a powerful zero-shot depth estimator. These depth maps are then analyzed by the VLM, which detects nearby obstacles and plans avoidance maneuvers. We have explored the Gemini-flash and GPT-4o models as VLM in our study. A fully connected network integrates the VLM output with the UAV's relative heading angle to predict the optimal course of action, enabling the UAV to dynamically navigate complex environments toward its target. The system's effectiveness is validated through simulations in AirSim using Blocks and the Downtown West environment. The UAV consistently reaches its destination, avoiding obstacles and achieving a near-perfect task completion rate of 0.98. By eliminating the need for costly sensors such as LiDAR and operating without pre-existing maps, our solution provides a cost-efficient, generalizable approach to real-time UAV navigation, especially in unfamiliar or dynamic settings, and highlights emerging trends in autonomous systems research that utilize VLMs.

1 Introduction

Autonomous UAV systems are poised to redefine the future by seamlessly integrating intelligence, adaptability, and efficiency across a wide range of industries. By

materials are also available via the authors' academic project page at https://gcsarker.github.io/vlmnav/.

Funding: This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) (grant number IMSIU-DDRSP2602). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

navigating complex terrains, autonomous UAVs will unlock new possibilities in urban air mobility, disaster relief, and sustainable practices [1–3]. The task of autonomous UAV navigation can be viewed as a process in which the UAV develops a strategy to reach a designated destination safely and efficiently without human intervention. The Key features of such systems include real-time obstacle avoidance, map independence, and adaptability to unknown terrains. These capabilities are achieved by integrating advanced AI and sensor technologies, which enable the seamless processing of complex visual and spatial data, ensuring safe, efficient, and responsive flight paths. To successfully complete its designated mission, a UAV requires comprehensive awareness of its surroundings, including its location, velocity, heading direction, and target destination [4,5]. The UAV first acquires sensor data regarding its current state through the perception module, which is then sent to the planner module for tasks such as path planning, collision avoidance, localization, and the generation of continuous control signals. These control signals guide the UAV toward its desired destination.

The perception module utilizes a wide range of sensors. In comparison to traditional sensors, such as LiDAR and ultrasonic sensors, visual sensors offer distinct advantages, as they can capture detailed information about the surroundings, including color, texture, and other visual details. Additionally, visual sensors are more cost-effective and easier to deploy, making them a focal point of research in the field of navigation [6]. The types of visual sensors typically include monocular, stereo, RGB-D, and fisheye cameras. Monocular cameras, specifically, are well-suited for applications where compactness and minimal weight are crucial factors, and thus they are the primary focus of this research. Table 1 highlights the merits and demerits of monocular sensors compared to LiDAR. Vision-guided navigation systems are categorized into three main groups: map-independent, map-based, and map-building techniques. Map-independent (mapless) systems function without a predefined map; instead, they rely on observing and extracting distinctive environmental features to navigate, as described by [4]. In this study, we focused on developing such a system, as it offers several advantages over map-based alternatives. Notably, it eliminates the need for a global map, which is a requirement in map-based methods [7,8]. Map-building techniques encounter difficulties in highly dynamic environments and face challenges related to the time-consuming creation and updating of obstacle maps [9,10]. Compared to map-based UAV navigation, the mapless approach eliminates the need for complex path planning or path tracking while enhancing the system's ability to handle dynamic obstacles.

Over the past few years reinforcement learning (RL) methods became popular for the navigation task [11]. It allows for an end-to-end learning mechanism, where UAV learns to map directly sensory input to control action. Unlike the optical flow or feature tracking methods, it learns from mistakes, which gradually improves navigation performance over time. Furthermore, obtaining labelled training data can be difficult using supervised and motion based approaches. However, many prior researchers utilizing RL for UAV navigation typically train their models in static, predefined environments [12,13]. In such setups, the RL agent learns specific visual and spatial features within the training

Table 1. Comparison between different sensors for perception.

Sensor	Price Range (USD)	Range (Meters)	Weight (kg)	Accuracy (cm)	Power (W)
LiDAR	$1,000 – $75,000	100–300+	0.3–1.5	2–5	5–15
Monocular Camera	$50 – $500	10–50	< 0.1	Varies	1–2
Stereo Camera	$150 – $2,000	10–50	0.1–0.5	5–10	3–5
RGB-D Camera	$150 – $5,000	5–10 (active)	0.2–0.7	1–3	3–10

https://doi.org/10.1371/journal.pone.0345778.t001

environment, which limits its generalizability to unseen or more complex settings with different lighting, textures, or obstacle configurations, leading to poor real-world performance [14,15]. Moreover, training RL policy can be computationally expensive and time-consuming. The UAV must interact with the environment repeatedly, collecting experience through trial and error. This requires extensive simulations or real-world flight data, both of which come with high computational overhead. RL algorithms also often involve complex reward structures and tuning, which further increases training time [16].

VLMs and Large Language Models (LLMs) have shown significant success in various tasks, including image-to-text generation, object detection, semantic segmentation, and content creation [17]. These advancements are creating new opportunities for decision-making in navigation, especially within vision-and-language navigation (VLN) [18]. Recently, researchers have been investigating the use of LLMs and VLMs for robotic navigation. These models are particularly advantageous due to their zero-shot learning capabilities and their ability to explain reasoning, enabling effective navigation in unfamiliar settings. Typically, a VLN agent navigates by following a series of human-given instructions and utilizing visual cues to reach a designated target, with success measured by proximity to the goal location [18]. While the integration of VLMs in autonomous vehicles is still being developed, the increasing sophistication of cutting-edge models indicates their potential for broader applications in this area.

In this research, we have developed VLM-Nav, a cost-efficient method for controlling a multirotor UAV to travel from its starting point to a designated destination while avoiding collisions with obstacles, utilizing visual data obtained from an onboard monocular camera. Since depth perception cannot be achieved with RGB images alone, the RGB images are transformed into a depth map using a pretrained depth-estimation model, which provides crucial spatial information about the environment. The depth maps, along with specific prompts, are then fed into a VLM, which processes the visual data and generates context-aware decisions related to obstacle avoidance. The outputs from the VLM, along with the heading angle, are input into a fully connected network (FCN). This model predicts the optimal action at each timestep, ensuring that the UAV navigates safely while avoiding obstacles in real-time [19]. Our objective is to replicate human-like decision-making for UAV control. Fig 1 illustrates the advantages of the proposed system compared to previous methods in the literature. Our approach enhances generalizability across varied environments by decoupling perception and decision-making. This contrasts with many prior research approaches that tend to overfit to specific training environments [14,20,21], where UAVs learn environmental features such as color, lighting, and textures that may not be relevant in new, unseen environments. We achieved the following contributions in our work.

- This study develops an end-to-end autonomous UAV navigation framework guided by monocular vision.
- The proposed method relies on RGB scene images, which are converted into depth maps using a zero-shot pretrained depth estimator and heading angle, making it fully map-independent.
- By separating perception from decision-making, the approach enhances generalizability and explainability, allowing the UAV to navigate diverse and complex environments.
- Unlike prior research in visual language navigation in the literature, the proposed method does not rely on any external instructions from humans or utilize any extensive training and dataset.

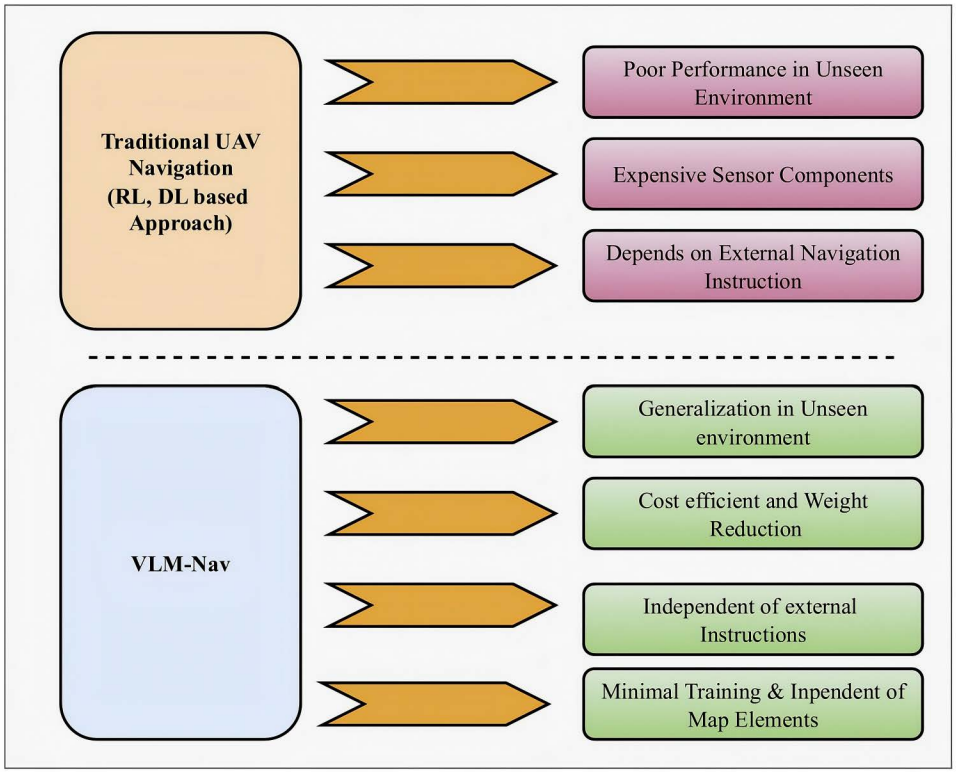


Fig 1. Compared to traditional RL/DL-based UAV navigation systems which struggle with unseen environments, require expensive sensors, and rely on external instructions, VLM-Nav offers better generalization, cost efficiency, independence from external inputs, and minimal training needs, making it a more adaptable and efficient navigation approach.

https://doi.org/10.1371/journal.pone.0345778.g001

- This research offers insights into the use of VLMs for UAV navigation and explores current challenges and future research directions.

Section 2 provides a comprehensive overview of previous research on vision-guided navigation systems and discusses the remaining challenges. In Section 3, we outline the specific navigation task this study focuses on. The methodology for our proposed VLM-Nav approach is detailed in Section 4, followed by the simulation results in Section 5. The contributions of each component in our system are provided in section 5.5. Finally, Section 6 summarizes the study's key findings.

2 Literature review

Although UAVs have been utilized in various applications over the years, achieving safe flight during autonomous navigation remains a significant challenge [6]. Map-based UAV navigation relies on pre-defined 2D or 3D environmental representations. These maps vary in complexity, incorporating terrain details, obstacle locations, and strategic waypoints. Cui et al. [7] introduced a multi-layer strategy based on RL algorithm leveraging both global and local informations for efficient path planning. However the study was carried out in a grid-based environment, where UAV movements are defined by eight possible actions within a two-dimensional state space. Kim et al. [22] utilized DL-based object detection to identify and label ground objects, converting them into labeled points. The spatial configuration of these objects is then matched with a pre-existing map database to determine the UAV's precise location. Other approaches [23,24] leverage GPS signals, georeferenced maps, and optical flow techniques to refine UAV positioning. Additionally, Simultaneous Localization and Mapping (SLAM) has emerged as a key method for real-time map generation, allowing UAVs to navigate without relying on

pre-existing maps. SLAM frameworks, such as LIO-SAM [25], integrate LiDAR and inertial data to enhance accuracy, particularly in complex outdoor environments. While map-based strategies offer reliable navigation, their effectiveness depends on the availability and accuracy of pre-existing maps. Consequently, mapless navigation, which leverages onboard sensors and AI-driven adaptability, has attracted attention for its robustness in dynamic or unfamiliar terrain, making it the focus of this research. The various approaches discussed in the literature are systematically compared in Table 2.

Early methods for monocular vision-based navigation include appearance-based approaches that treat obstacles as foreground objects against a consistent background, like the ground or sky [32]. These methods are typically limited to environments where obstacles are easily distinguishable from their surroundings. Motion-based techniques, such as optical flow, rely on the assumption that nearby objects show distinct movements detectable through motion vectors in the image. Since objects closer to the camera have greater displacements, any point with a displacement exceeding a specific threshold is classified as an obstacle pixel [33]. However, the effectiveness of optical flow estimation is often poor because it assumes brightness constancy and smooth transitions between frames, conditions that are rarely fulfilled in real-world situations [34]. Another method is feature tracking. For example, the study in [35] uses a monocular camera to estimate the depth of obstacles in both indoor and outdoor environments. The algorithm employs SURF (scale-invariant feature transform) to achieve this. At each step, the depth of an obstacle is estimated using a Kalman filter. When the estimated distance to an obstacle exceeds a certain threshold, a hover command is triggered, prompting the UAV to rotate before continuing forward.

Table 2. Recent studies in vision based UAV navigation.

Ref	Objective	Method	Action Space	Simulation Setup	Limitations
[21]	Indoor navigation & obstacle avoidance	VAE & SAC (delayed learning)	Continuous Control	Airsim	• Performance decrease in new environment
[26]	Indoor navigation	DDPG	velicity (x,y)	Airsim	• Simple environment. • No target location considered
[12]	Indoor & outdoor navigation	Memory Enhanced DQN	yaw left, yaw right, straight	Airsim	Simulation only, Fixed height
[20]	Obstacle avoidance using monocular vision	FCRN (depth estimation) & D3QN	15 discrete actions (linear & angular velocity)	ROS and Gazebo	• Cannot generalize well in unseen environment.
[27]	Indoor navigation	H-DrQ & temporal attention	Continuous Yaw Angle [−60,60]	Airsim	• Cannot generalize well in unseen environment. • High computational cost.
[28]	3D UAV navigation	DDPG & TD3 with human in the loop	Acceleration (a_x, a_y, a_z)	Airsim	• Not considered collision in the reward function. • Not considered how relative target location will be received
[29]	Visual language navigation	CNN + Bi-LSTM+GRU with cross modal attention	series of action (Move forward, left right, up and down)	Airsim	• Requires external instruction
[30]	Visual language navigation	LLM (GPT-3, GPT-4)	series of action	Airsim	• Requires external instruction. • Need to specify landmarks. • Requires high computational resources.
[31]	MultiObject Navigation	Scene Object Graph + GPT-3.5-turbo/GPT-4 + Imitation Learning	Forward, left, right	–	• Requires high computational resources
Proposed	UAV Navigation and obstacle avoidance	DepthAnything + VLM	Forward, Yaw left and Yaw right, Up and Down	Airsim	• Only static obstacles considered.

https://doi.org/10.1371/journal.pone.0345778.t002

RL-based methods are widely used for robotic navigation tasks because they effectively overcome the limitations of earlier techniques. Chen et al. [13] introduced a method for collision-free UAV navigation in small indoor environments using monocular camera images. The proposed technique utilizes object detection to assist in training a Deep Q-Network (DQN). Doukhi et al. [36] developed an end-to-end mapless navigation system using 2D Lidar and depth images, applicable in both unknown indoor and outdoor environments. The architecture comprises a collision-awareness module (CAM) and a collision-free control policy module (CFCPM). The CAM processes and fuses sensory data to generate an observation, which is then passed to the CFCPM. In the CFCPM, a DQN algorithm with a CNN policy is employed to determine the optimal collision-free policy by selecting the best action from right, left, and forward movements. The authors in [26] proposed an indoor navigation system utilizing the DDPG algorithm. Their approach takes inputs such as a depth map, the drone's current position, and the Euclidean distance to its destination. The reward function incorporates two margins. In the soft-margin area, a linear function gradually penalizes the drone as it approaches an obstacle. On the other hand, the hard-margin area uses a reciprocal function that rapidly increases the penalty to push the drone away from it. The research conducted by Li et al. [28] devised a UAV navigation system for a 3D environment using DDPG and TD3 algorithms. The observation space includes the UAV's 3D orientation (position, velocity), environmental data from depth images, and information from 12 distance sensors. To ensure safe navigation across environments with varying levels of obstacles, the authors trained another deep learning model using human labeling to select appropriate reward schemes.

To capture temporal dependencies, researchers have applied various techniques such as recurrent neural networks (RNNs) and attention-based methods. In a study by Fu et al. [12], a memory-enhanced deep Q-network (DQN) algorithm was introduced for navigation, using visual inputs and kinetic data such as orientation, current position, and target location. Their proposed memory structure incorporates historical observations and actions, which are processed through an attention mechanism. Liu et al. [27] developed a UAV navigation system for indoor and narrow corridor environments, integrating hierarchical learning, recurrent neural networks, and the data-regularized Q (DrQ) algorithm. This system demonstrated improved handling of long input sequences and better exploration of temporal dependencies through temporal attention. While reinforcement learning (RL) has been successfully applied to navigation tasks in recent years, manually tuning hyperparameters, including reward functions, often leads to suboptimal solutions [37]. In multi-objective scenarios, determining the coefficients for reward functions involves significant trial and error, heavily relying on the experience of the researchers [38].

Due to the inability of monocular sensors to perceive depth information, researchers use various depth estimation techniques. Singla et al. [39] proposed a deep recurrent Q network with Long Short-Term Memory (LSTM) and temporal attention for obstacle avoidance in cluttered and unfamiliar indoor environments. The model takes monocular RGB images as input, which are transformed into depth maps using a conditional generative adversarial network (cGAN). The proposed approach by Xue et al. [21] involves using depth camera images as input, which are preprocessed using a variational autoencoder (VAE). The processed images are then utilized by the delayed soft actor-critic algorithm to generate continuous control commands. Kim et al. [20] employed a Fully Connected Residual Network (FCRN) to estimate depth from an RGB image, generating a corresponding depth image. These depth images, along with the RGB inputs, serve as inputs for the D3QN algorithm, which generates control commands. Although these approaches for obstacle avoidance have demonstrated satisfactory performance in indoor environments with limited obstacle settings, their effectiveness in outdoor environments with varied and complex obstacles remains untested. This is because the trained depth estimator learns only the features of known obstacles. To overcome this challenge, it would be beneficial to explore pretrained depth estimator models that are trained on various scenarios for depth estimation, such as DepthAnything [40], MiDaS [41] etc.

In recent years, large language models (LLMs) and vision-language models (VLMs) have proven to be highly effective for navigation tasks. Liu et al. [29] developed a system in which a UAV generates a series of actions based on natural language instructions, integrating both the instructions and various perception inputs like depth maps, RGB images, and pose data. This system is trained using flight trajectories collected with human support, with guidance from experienced

annotators. Schumann et al. [30] created VELMA, which processes a sequence of prompts—including a task description, navigation instructions, and the current trajectory—to predict the next action through next-word prediction using an LLM. This action is executed in the environment, while landmarks identified from the instructions are verified for visibility in the panorama view at each step. The verbalizer combines this landmark data with potential intersection information to generate text observations, which are then added to the prompt sequence for ongoing action prediction until the agent reaches its target. In a similar approach, LM-Nav introduced in [42] handles raw observations and free-form textual instructions to create actionable plans using three pre-trained models: an LLM for landmark extraction, a VLM for grounding, and a visual navigation model (VNM) for execution. This enables LM-Nav to follow textual instructions in complex environments based solely on visual data, without needing fine-tuning. Rajvanshi et al. introduced sayNav for multi-object navigation tasks, where the UAV first constructs a 3D Scene Graph of various objects [31]. An LLM-based dynamic planner extracts a subgraph, converts it into text prompts, and inputs it into an LLM. Each planned step by the LLM is executed by Dagger, an imitation learning algorithm that produces control commands. In contrast to these approaches, our system operates independently of external instructions or additional environmental information.

3 Navigation task

The goal of the proposed VLM-Nav system is to autonomously guide a multirotor UAV from an initial position (x, y) to a target location (x_{dest}, y_{dest}). The UAV operates at a constant velocity of v and maintains starting altitude of h, adjusting its altitude only to avoid obstacles by moving vertically as needed.

3.1 Observation space

To navigate the UAV safely, in other words, to determine which action should be taken at each time step, the UAV uses state information from the environment. In this paper, the observation space, S can be expressed by equations 1 and 2. The drone captures RGB images M through a monocular camera. Then, a corresponding depth map for each image is generated, denoted by D. L_{dist} and R_{dist} represents the distances measured by the left and right distance sensors, respectively. ϕ is the angle between the drone's forward direction and the target location. Fig 2 depicts the UAV setup for this study, highlighting different sensor placements.

$$D = DepthEstimation(M) \tag{1}$$

$$S = \{D, L_{dist}, R_{dist}, \phi\} \tag{2}$$

3.2 Action space

The UAV observes its surroundings at every time step and generates low-level control commands. It is equipped with advanced autopilot systems that allow for straightforward control by specifying any particular position or velocity, with the control signals automatically generated by the autopilot system. We use Simpleflight, offered by Airsim [43], which aids our algorithm by facilitating seamless control between control commands and generated output. The action space encompasses five discrete actions, outlined as follows.

Forward = Go straight for 1s
Yaw Left = Rotate counter clockwise by 25°
Yaw Right = Rotate clockwise by 25°
Up = Move upward for 1s
Down = Move downward for 1s

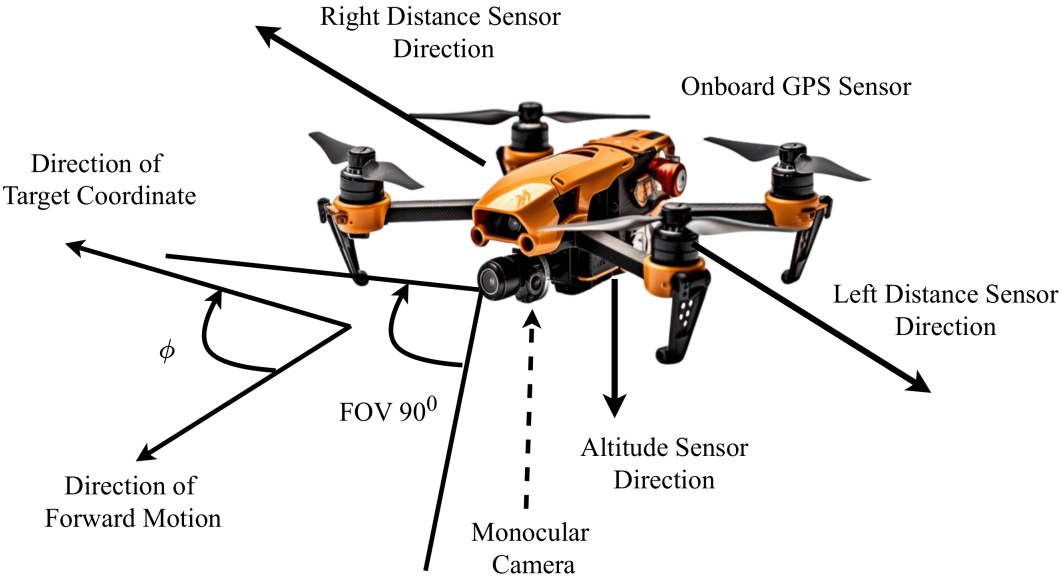


Fig 2. Navigation setup used in this study.

https://doi.org/10.1371/journal.pone.0345778.g002

3.3 Simulation environment

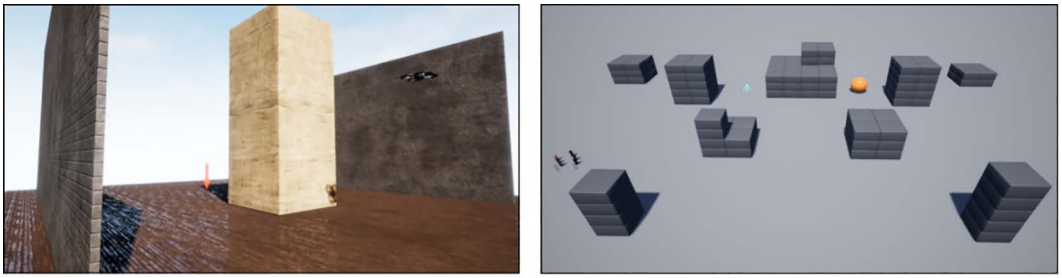
Simulation environments provide a secure, controlled space for testing and refining navigation algorithms, mitigating the risk of damaging expensive equipment or harming individuals or the drone. By simulating real-world scenarios and challenges, developers can evaluate the performance of navigation algorithms across various conditions, such as adverse weather, complex terrain, or unexpected obstacles. AirSim [43], a plugin for Unreal Engine (UE) developed by Microsoft, enables developers to implement their system for controlling multirotors in realistic 3-D environments. It is one of the most widely used platforms for autonomous vehicle research [21,26,44].

We have utilized three environments in UE for our research, as shown in Fig 3. Environment A, depicted in Fig 3, is a simple single-obstacle environment with two side walls, and its dimensions are $30\,\text{m} \times 30\,\text{m}$. The width and height of the obstacle are randomly scaled by a factor of 0.5 to 5 during each navigation trajectory. Environment B, shown in Fig 3, is the Blocks environment provided by the AirSim package, featuring several obstacles of different shapes spaced throughout. It is comparatively large, with dimensions of approximately $220\,\text{m} \times 100\,\text{m}$. Environment C, as shown in Fig 3, is made from the Downtown West pack, available in the UE marketplace. The pack contains assets of various objects in a city environment. Such as building, food cart, bench, rocks, posters, etc. We have made a customized environment using the pack to validate the UAV performance in realistic scenarios.

The objective for the UAV is to navigate from a given coordinate location to another while avoiding obstacles. At each episode, the spawn and target coordinates are generated randomly. Our approach enables the UAV to navigate from any starting point to the target location. Since the UAV will be flying at much higher altitudes in a practical scenario, we have ignored objects such as trees and poles. street lamp, etc.

4 Methodology

This study develops a system for autonomous UAV navigation from a starting point to a target destination, using visual inputs to navigate around obstacles. The system functions in three phases, as shown in Fig 4. First, the UAV's vision sensor captures monocular images, which are then converted into depth maps. These depth maps are analyzed by a Vision Language Model (VLM), which suggests actions for obstacle avoidance. In the final phase, a fully connected network



(a) Environment A (b) Environment B



(c) Environment C

Fig 3. Our system is validated in the following three environments- (a) Simple environment with a single obstacle between two walls. This environment is used to train the navigator model. (b) The block environment is provided by AirSim. (c) Created using the Downtown West pack from the UE marketplace.

https://doi.org/10.1371/journal.pone.0345778.g003

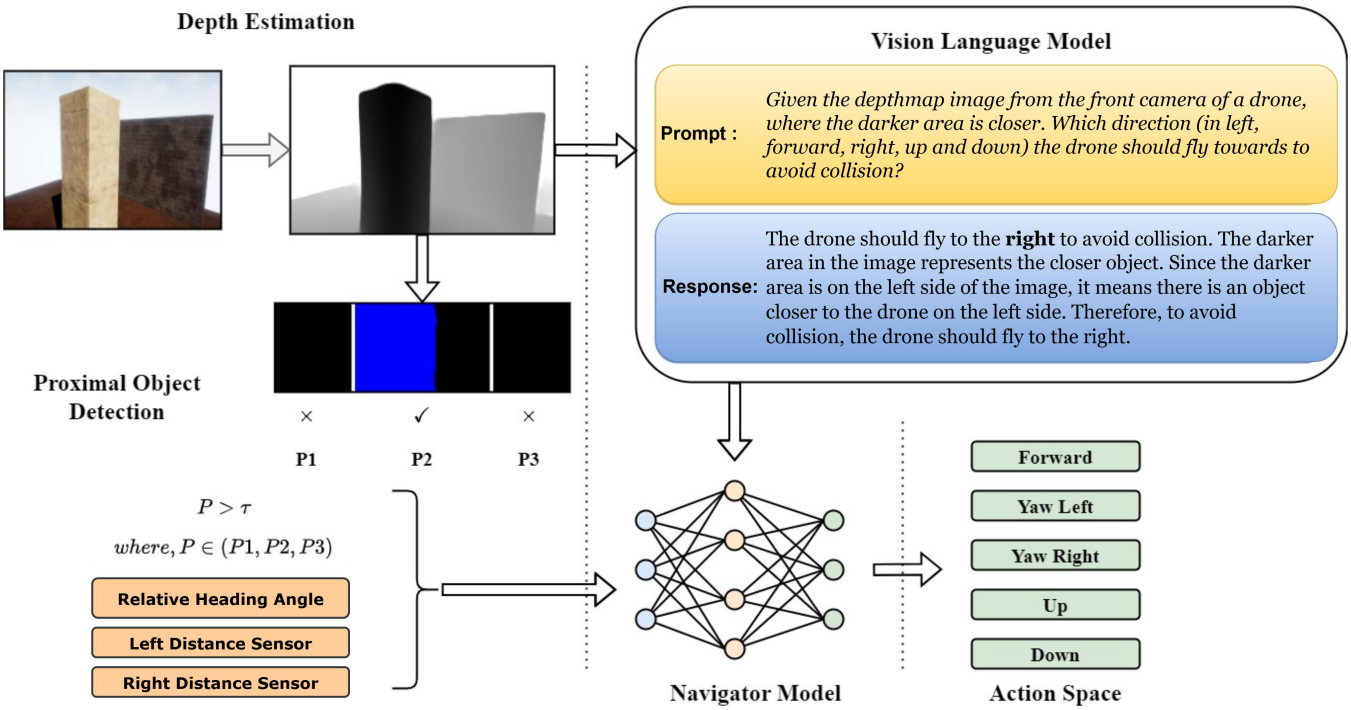


Fig 4. Overview of the proposed VLM-Nav method. (Left) First, RGB scene images are captured and converted into a depth map. (Top right) This depth map is then analyzed by the vision-language model (VLM), which provides the corresponding action response. (Bottom right) Lastly, the VLM's feedback, along with the relative heading angle, left and right distance sensor measurements, and proximal object detection output, are sent to the navigation model.

https://doi.org/10.1371/journal.pone.0345778.g004

processes data from depth map regions, distance sensor readings, heading angle, and VLM feedback to determine the UAV's final action. Algorithm 1 presents the pseudocode for our proposed VLM-Nav approach.

Algorithm 1: VLM-Nav Algorithm

$M \leftarrow$ RGB scene
$L_{dist} \leftarrow$ Left distance sensor
$R_{dist} \leftarrow$ Right distance sensor
$Prompt \leftarrow$ VLM Prompt to avoid obstacle
$\tau \leftarrow$ Threshold for proximal object detection
$\tau_d \leftarrow$ Threshold for distance sensors
$P_{current} \leftarrow$ Current Position
$P_{Target} \leftarrow$ Target Location
$d \leftarrow Distance(P_{current}, P_{target})$
while d<3m **do**
  $\phi \leftarrow$ Relative heading angle
  $D \leftarrow DepthAnything(M)$
  $D' \leftarrow D > \tau$ // Cropped D of shape $(m \times n)$
  $D' \leftarrow Spaghetti(D')$ // Connected Component
  $P1, P2, P3 \leftarrow D'$ //Three regions
  for $P \in [P1, P2, P3]$ **do**
    n←Number of connected groups∈P
    $P' \leftarrow Bool(if\, n > 1)$
  end for
  $l \leftarrow Bool(L_{dist} > \tau_d)$
  $r \leftarrow Bool(R_{dist} > \tau_d)$
  $F \leftarrow VLM(D, Prompt)$
  $States \leftarrow [P', l, r, \phi, F]$
  $a \leftarrow FCN(States)$ // navigator model
  step(a) //take action a
end while

4.1 Depth estimation

Effective navigation and obstacle avoidance require precise depth information. Since monocular images lacks spatial features, a real-time depth estimator is used in this study. While previous methods have employed various depth estimation techniques, such as FCRN and cGAN, they are generally trained on simulation engines. For this reason, a zero-shot depth estimation method is vital, allowing it to work with any images. DepthAnything [40] is a pre-trained, zero-shot depth estimation model designed to improve generalization using a dataset of 62 million automatically annotated, unlabeled images. A teacher model, initially trained on 1.5 million labeled images, generates pseudo-labels for the unlabeled data. The model follows an encoder-decoder architecture, with ViT-L (Vision Transformer large) as the encoder and DPT as the decoder. The combination of labeled and pseudo-labeled images is used to train a student model. To enhance the teacher model's depth prediction, various perturbations, such as color distortions and Gaussian blurring, are applied, along with the integration of high-level semantic features. DepthAnything achieves superior zero-shot accuracy compared to models like MiDaS [41]. In DepthAnything V2 [45], manual labeling is replaced by more precise synthetic images from LiDAR and stereo sensors. The teacher is trained on these synthetic images, and to address distribution shifts and scene limitations between synthetic and real data, the model is jointly trained on pseudo-labeled real images, thereby enhancing its robustness and generalization to unseen data. In our work, real-scene images of size 144 × 256 captured by the UAV are processed with DepthAnything V2, with the outputs scaled to (0–255) and converted to single-channel images of the same shape. DepthAnything provides depth maps where the closer the distance, the larger the pixel value. To prepare the VLM for feeding, we first invert the depth maps. An example of a depth map generated by this module is shown in Fig 5.

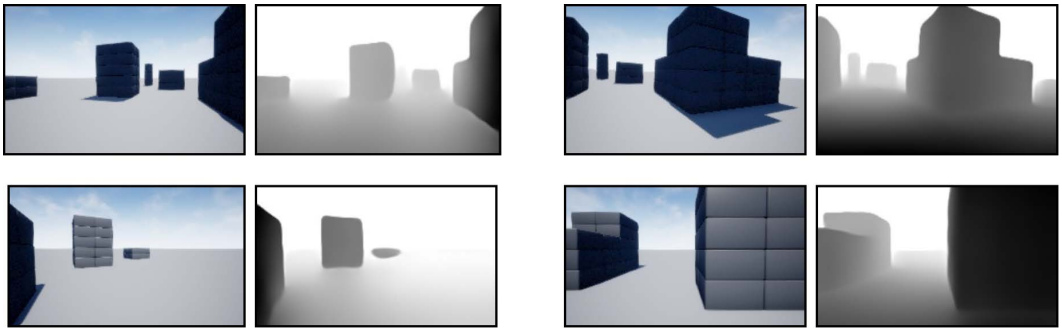


Fig 5. Examples of depth image estimation from Depth-Anything v2.

https://doi.org/10.1371/journal.pone.0345778.g005

4.2 Vision language model

With successful applications of vision language such as image captioning, segmentation, object detection, we are particularly interested in the visual question answering (VQA) task. VLMs such as Florence 2, PaliGemma, CogVLM are some of the popular options. However this algorithms mainly focus on vision task, and often offer limited language capability. In this work, we have utilized two language models with vision capabilities. One is GPT-4o (omni) model, a variant of generative pretrained transformer (GPT) architecture and is developed by OpenAI. GPT-4o stands out with improvements like larger context windows, faster processing, and efficient tokenization, excelling in text, audio, video, and image tasks [46]. Comparisons with top language models show GPT-4o's superiority in performance metrics like throughput and response time.

Another model is Gemini-1.5-flash model from the Gemini family developed by Google DeepMind [47]. Similar to the GPT-4o, this model can process multimodal data such as images, text, and videos, integrating advanced reasoning, planning, and memory capabilities making it highly versatile for complex real-world tasks [5]. This model is developed using Transformer Decoder architecture and is designed for more sophisticated problems across domains, offering superior performance in multimodal and complex reasoning tasks. Both models provides robust API support, making them suitable for integration into diverse applications without relying on large memory spacce, which is ideal for small UAV. Fig 6 illustrates how VLM is employed in VLM-Nav. The estimated depth map from the scene images is first normalized, rescaled, inverted, and then sent to the Gemini-1.5-flash model via an API key. The input to the VLM consists of a depth map and a predefined prompt that asks the UAV to take a direction to avoid colliding with any obstacles. The model provides the desired direction along with a detailed explanation for its response. Subsequently, directions such as left, right, or either direction are extracted using a keyword search, as shown on the left side of the figure. The incorporation of VLM significantly enhances generalization without requiring obstacle detection or semantic segmentation, which were used in previous navigation systems.

4.3 Proximal object detection (POD)

We analyze the generated depth map as illustrated in Fig 7 to evaluate the distance to any obstacles that may exist in the front view. First, the depth map is cropped to a shape of $(m \times n)$. This cropping is especially necessary during low-altitude flight to exclude the ground and focus solely on the obstacles ahead. Next, binary thresholding is applied with a threshold value of τ, which helps ignore any obstacles beyond a certain distance. Afterward, connected component analysis is performed using the spaghetti technique [48], an efficient algorithm for identifying and labeling distinct connected components (i.e., clusters) in an image or grid. The algorithm is named after its approach of traversing a pixel grid in a manner akin to strands of spaghetti. First, the pixel grid is scanned row by row, from left to right and top to bottom. Labeling is then done

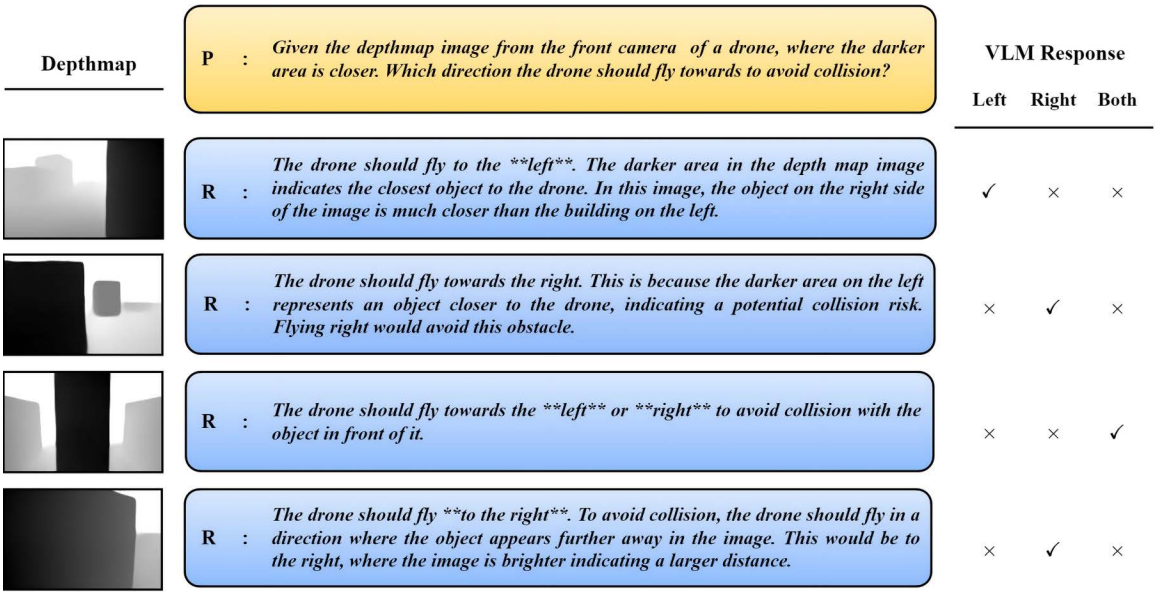


Fig 6. (left) The depthmap is estimated from the RGB scene, which is then normalized and rescaled into (0–255). (middle) The depthmap is sent to VLM along with the preset Prompt (P). (right) Based on VLM feedback (R), the suggested direction to avoid the obstacle is extracted using keyword search.

https://doi.org/10.1371/journal.pone.0345778.g006

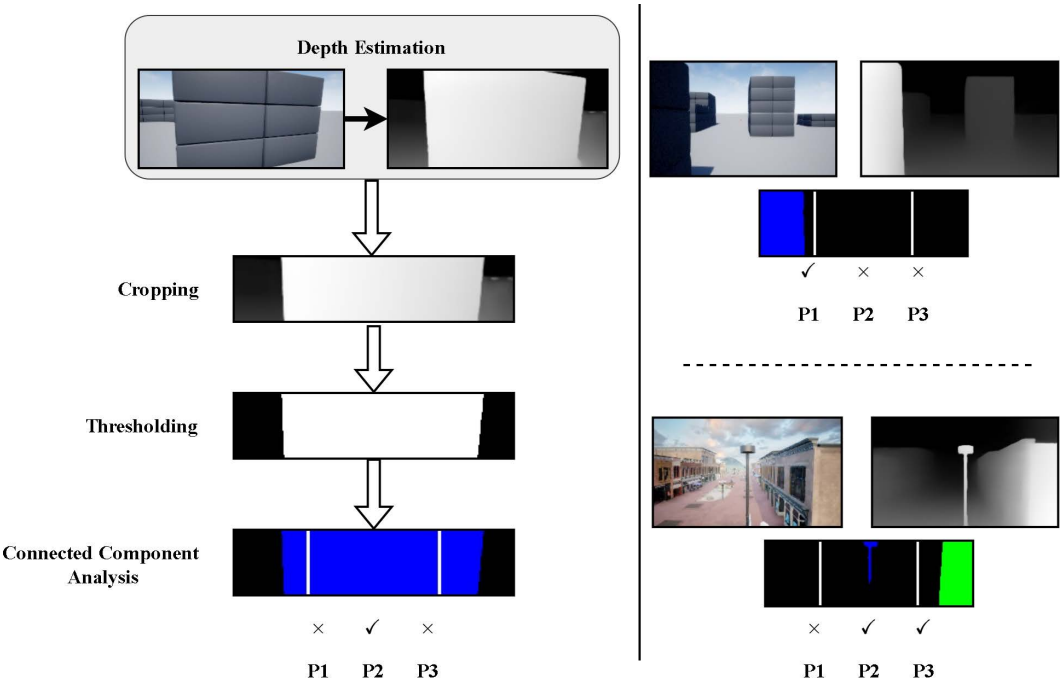


Fig 7. (Left) The process of proximal object detection in VLM-Nav: First, the depthmap is cropped, and the pixel values are binarized using a threshold τ, followed by connected component analysis. Finally, the output indicates whether any connected groups exist within the three defined regions. (Right) An example of the process with two scenarios.

https://doi.org/10.1371/journal.pone.0345778.g007

by following the labels of neighboring pixels. In 8-connectivity, diagonal pixels are also considered. Only connected groups of more than 100 pixels are retained. We then divide the image into three sections ($P1$, $P2$, $P3$). The VLM (Gemini) cannot determine the distance to an obstacle based solely on the depth map. In fact, the depth map produced by DepthAnything does not provide true distances. These three sections serve two main purposes: first, they provide the UAV with a sense of obstacle depth. Secondly, they help reduce the number of API calls. We only send a request to the Gemini model for feedback if we detect any connected component in the P2 region, which is directly in front of the UAV. This significantly speeds up our system by avoiding request delays. Regions 'P1' and 'P3' assist the UAV in determining whether it is safe to turn, providing crucial information in such scenarios. The dimensions of the sections are carefully configured.

4.4 Navigator

The navigator model in Fig 4 is a fully connected network (FCN) designed to mimic human-controlled drone flight. The RGB stream is converted to a depth map using DepthAnything V2, and then POD module detects the presence of any frontal obstacle using threshold value τ. Distance measurements from the left and right distance sensors are also collected. These distance values are compared against another threshold τ_d. The purpose of this threshold is to maintain a minimum distance from an obstacle on the left or right while the UAV is flying beside it. It also aids the navigation task when rotating in a corner. The angle ϕ, which is the angle between the direction of the drone's forward motion and the target location, is also calculated. We also incorporate feedback from the VLM module. This module provides a boolean array indicating the advised directions (Left, Right, Forward, Up, or Down). The array is generated through keyword searches in the VLM's output text. In cases where an obstacle is directly ahead and the VLM suggests both left and right as optimal maneuvers as shown in Fig 6, both directions are set to True. Ultimately, the all these parameters are sent to an FCN, Which converts the multi-dimensional input into five discrete navigational actions (left, right, forward, up and down). The input parameters of the navigator model are provided in Table 3 in details. Fig 8 illustrates the model architecture.

To train the navigator model, we use the simplified environment A in Fig 3. A human annotator fly the UAV at each timestep while getting feedback from the input paramters. 10,000 steps of these parameters and the corresponding action taken by the human pilot is recorded as flight trajectories. These are then used to train the navigator model. The human annotators are instructed to maintain the following rules while selecting an action.

- Maintain a relative heading angle ϕ close to 0 during flight. Only initiate obstacle avoidance maneuvers when the POD module detects an obstacle ahead.
- To avoid obstacles, follow the direction provided by the VLM, unless it issues repetitive commands (e.g., alternating between left, right, and left) or generates incorrect directions.

Since the model does not rely on any images or location elements, it is completely map-independent and can be easily applied to other environments. Table 4 shows the hyperparameters of the navigator model.

Table 3. Details of input parameters for the navigator model.

Parameter	Type	Shape
Output from POD, ($P_1 < \tau?, P_2 < \tau?, P_3 < \tau?$)	(Bool, Bool, Bool)	(3,)
Relative heading angle, ϕ	Float	(1,)
Left distance sensor, ($L_{\text{dist}} < \tau_d?$)	Bool	(1,)
Right distance sensor, ($L_{\text{dist}} < \tau_d?$)	Bool	(1,)
VLM feedback (action to avoid collision)	(Bool, Bool, …, Bool)	(5,)
Total Input Shape	–	(11,)

https://doi.org/10.1371/journal.pone.0345778.t003

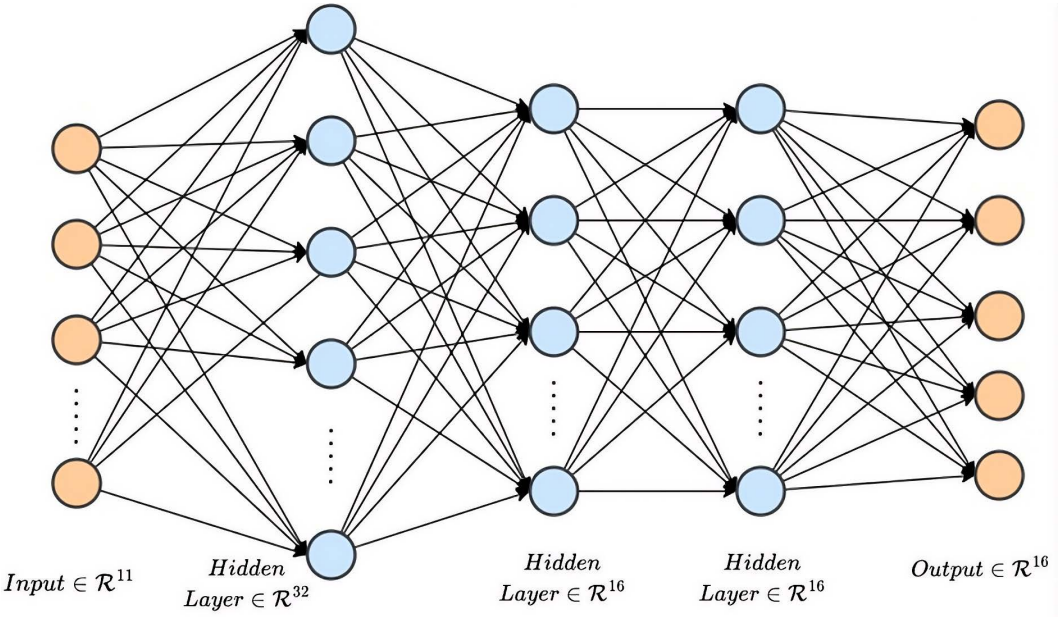


Fig 8. Navigator model architecture.

https://doi.org/10.1371/journal.pone.0345778.g008

Table 4. Hyperparameters of the Navigator FCN model.

Hyperparameter	Value
No. of hidden layers	3
No. of hidden Units / layer	32, 16, 16
Activation Function	ReLU
Loss Function	Cross-entropy
Optimizer	Adam
Batch Size	32

https://doi.org/10.1371/journal.pone.0345778.t004

5 Result and discussion

The objective of this study is to develop an autonomous navigation system that enables a UAV to navigate from a randomly chosen spawn coordinate to a target coordinate while maintaining a fixed velocity and flight altitude. Different configuration settings for our approach are provided in Table 5.

5.1 Experimental setup

All experiments were carried out using the AirSim simulation plugin integrated with Unreal Engine version 4.27. The hardware platform for running the simulations was equipped with an Nvidia GeForce MX550 GPU and 16 GB of memory. This configuration ensured smooth rendering of the 3D environments and real-time data processing. The GPU's high computational power was critical for handling the intensive tasks related to model training, inference and simulation in real-time.

Table 5. VLM-Nav Configurations.

Experiment Settings	Value
Scene dimension	(144 × 256)
Depthmap dimension	(144 × 256)
Velocity	3 meter/second
Initial altitude	10 meter
Yaw rate	25deg
τ	8
τ_d	4
Connected component algorithm	Spaghetti
Minimum pixels in connected group	100
Command duration	1 second
Input token	137
Maximum Output token	300

https://doi.org/10.1371/journal.pone.0345778.t005

5.2 Quantitative analysis

The initial stage of VLM-Nav entails creating a depth map from the UAV's front-facing camera. We evaluated four algorithms for this task: MiDaS (based on DPT and Swin V2 models) and DepthAnything (versions V1 and V2). MiDaS offers various models depending on the underlying backbone architecture. We selected the DPT and Swin V2 large based models for experiments, as they offer an optimal balance between speed and accuracy. Table 6 shows a comparison of these methods. We gathered 10,000 scene images and corresponding ground truth depth maps from Unreal Engine in environments B and C. The ground-truth pixel values were generated by projecting rays from the camera to obstacles in Unreal Engine. These ground-truth values were then compared with the estimated depth maps, averaging the results across the 10,000 images using the following metrics.

1. Delta (δ): It computes the ratio of depth predictions $\hat{d}$ that fall within a certain threshold δ_t of the true depth d as shown in Eq. 3. Typically, multiple thresholds (e.g., $\delta_1 < 1.25$, $\delta_2 < 1.25^2$, $\delta_3 < 1.25^3$) are used to assess different levels of accuracy. Higher delta values indicate better performance.

$$\delta = \max\left(\frac{\hat{d}}{d}, \frac{d}{\hat{d}}\right) < \delta_t \tag{3}$$

2. Absolute Relative Error (AbsRel): This metric calculates the absolute difference between predicted $\hat{d}_i$ and true depth values d_i, normalized by the true depth as shown in Eq. 4. It shows how far the predictions deviate from the ground truth on average, scaled by the actual depth values. Lower AbsRel scores indicate better performance.

$$\text{AbsRel} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{d}_i - d_i|}{d_i} \tag{4}$$

3. Structural Similarity Index (SSIM): SSIM measures the similarity between the predicted and true depth maps in terms of structure, luminance, and contrast, taking into account spatial relationships. It evaluates the perceived quality of

Table 6. Performance comparison of depth estimation algorithms.

Method	Env.	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	AbsRel ↓	SSim Index ↑	Inference Time(ms) ↓	Par. (M) ↓
MiDaS (DPT)	B	0.5813	0.7807	0.9029	1.4887	0.5956	62.07	344
	C	0.6677	0.8615	0.6617	1.2940	0.5283	66.64	
MiDaS (Swin V2)	B	0.4232	0.6452	0.7586	0.8043	0.3034	1044.9	213
	C	0.6160	0.8629	0.7632	0.6246	0.5490	1027.2	
DepthAnything-V1	B	0.4196	0.6578	0.6783	0.4722	0.8507	23.77	24.8
	C	0.5801	0.7846	0.9414	0.1498	0.8899	21.07	
DepthAnything-V2	B	0.4590	0.7110	0.7930	0.3904	0.8901	22.93	24.8
	C	0.5125	0.7311	0.8963	0.4723	0.7439	24.94	

https://doi.org/10.1371/journal.pone.0345778.t006

the depth estimation and measured using Eq. 5. SSIM ranges from −1–1, where values closer to 1 indicate higher similarity.

$$\mathrm{SSIM}(\hat{d}, d) = \frac{(2\mu_{\hat{d}}\mu_d + C_1)(2\sigma_{\hat{d}d} + C_2)}{(\mu_{\hat{d}}^2 + \mu_d^2 + C_1)(\sigma_{\hat{d}}^2 + \sigma_d^2 + C_2)} \tag{5}$$

Here, $\mu_{\hat{d}}$ and μ_d are the means of the predicted and ground truth maps, $\sigma_{\hat{d}}^2$ and σ_d^2 are the variances, $\sigma_{\hat{d}d}$ is the covariance between the two images, C_1 and C_2 are small constants to avoid division by zero.

4. Inference time: It measures the time required for each method to estimate depthmap from a single scene image and indicates how fast the algorithm is.
5. Parameters (Par.): The number of trained weights in the model and presented in milions. Higher number requires more computational resources.

Both DepthAnything V1 and V2 significantly outperform the other techniques, demonstrating superior performance. These models are particularly well-suited for deployment in UAV systems due to their low parameter count and fast inference speed. Additionally, they offer higher structural similarity compared to the other models. While V1 and V2 are quite similar, V2 shows a slight edge in performance in both environments B and C.

The POD module in our method estimates whether any obstacles within a certain distance, defined by τ, exist as connected groups in three regions of the depth map. These three regions capture only a small portion of the depth map and provide limited depth information from the UAV's front perspective. In contrast, the VLM analyzes the entire depth map to identify obstacle locations and determine the appropriate direction to avoid them. A higher threshold value allows early detection, while a lower value requires the UAV to get closer. However, the POD module largely depends on the accuracy of the estimated depth map. DepthAnything v2 struggles to generate accurate depth values for distant small obstacles. As the obstacle approaches during navigation, the depth map improves, allowing the POD module to detect it. Fig 9 shows how our system detects cubes of different sizes, from 25 cm to 10 m, at varying distances depending on τ. Smaller obstacles require the UAV to be closer for detection, while larger obstacles are detected from farther away.

The navigator module in our method is an FCN that uses feedback from a VLM, distance sensor measurements, object detection from the POD module, and the heading angle to select the optimal action from five discrete options. The model is trained in a multiclass classification framework using a dataset gathered from human-operated flights in environment A. To evaluate the model's performance, we collected human-annotated trajectories

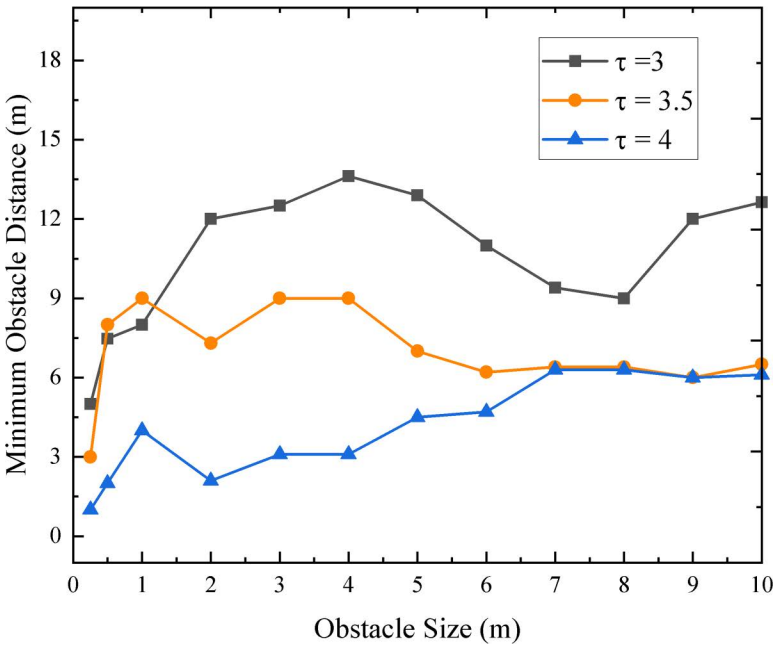


Fig 9. Minimum distance to obstacle detection based on different value of thresholds.

https://doi.org/10.1371/journal.pone.0345778.g009

in two different environments, B and C. For each discrete action class, we measured true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) using a one-vs-all approach (treating each class as positive and all others as negative), and then averaged these values across classes. TP and TN represent how accurately the model predicted the correct actions relative to human annotations. FP and FN capture the instances of incorrect predictions, where the model either predicted a wrong action (FP) or missed the correct one (FN). These metrics were then used to compute precision, recall, and F1 score, which we briefly discuss with respect to the navigation task as follows:

- Precision: It refers to the number of correct actions out of predictions and calculated using Eq. 6. Higher value means the model often predicts correct actions.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

- Recall: It indicates out of all the times the model should have predicted a specific action, how often did it actually predict it. High recall ensures that the model catches most of the correct actions.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

- F1 Score: As shown in Eq. 8, it is the harmonic mean of precision and recall. It provides a single measure that balances both precision and recall.

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

Table 7 presents the performance of the navigator module across all environments. The results indicate that both the Gemini and GPT-4o models achieve high accuracy, with F1 scores around 90% in previously unseen environments (B & C). This demonstrates the strong generalization capability of our approach. While both models perform well, the Gemini Flash model shows a slight edge in performance compared to the GPT-4o model, indicating its enhanced adaptability in navigating new environments. The overall navigation performance is then assessed by averaging the results of 100 flight trajectories and is presented in Table 8, using three key metrics:

1. Task Completion Rate (TCR): The navigation task is deemed successful if the UAV reaches within 3 meters of the target location; otherwise, it is considered a failure. TCR represents the percentage of successful flight trajectories.
2. Collision Rate (CR): This measures the percentage of times the UAV collides with an obstacle. It is calculated by summing collisions across the entire trajectory and converting to percentage with respect to distance travelled during flight as shown in Eq. 9.

$$CR = \frac{Number\ of\ Collisions}{Total\ Distance\ (m)} \tag{9}$$

Table 7. Performance of navigator module.

VLM	Environment	Precision ↑	Recall ↑	F1-Score ↑
Gemini	A	**0.9978**	**0.9889**	**0.9933**
	B	**0.9149**	0.9034	**0.9070**
	C	**0.9129**	**0.8961**	**0.9026**
GPT 4	A	0.9921	0.9875	0.9899
	B	0.8932	**0.9046**	0.8988
	C	0.9089	0.8742	0.8912

https://doi.org/10.1371/journal.pone.0345778.t007

Table 8. Overall navigation performance of VLM-Nav.

VLM	Depth Estimator	Environment	TCR ↑	CR ↓	TLR ↑
GPT-4	DepthAnything-V1	A	1.00	0.00	0.98
		B	1.00	0.00	0.98
		C	0.98	0.06	0.93
	DepthAnything-V2	A	1.00	0.00	0.99
		B	1.00	0.00	0.99
		C	1.00	0.02	0.93
Gemini	DepthAnything-V1	A	1.00	0.00	0.99
		B	1.00	0.01	0.98
		C	1.00	0.05	0.92
	DepthAnything-V2	A	1.00	0.00	0.99
		B	1.00	0.00	0.99
		C	0.98	0.03	0.94

https://doi.org/10.1371/journal.pone.0345778.t008

3. Trajectory Length Ratio (TLR): This metric measures the ratio of trajectory length of the human-operated UAV (L_H) and the proposed system ($L_{VLM-Nav}$) as shown in Eq. 10. It indicates whether the UAV follows a path close to the shortest route to the target and mimics human decision making. Values close to 1 indicates better formance.

$$TLR = \frac{\min(L_H, L_{VLM-Nav})}{\max(L_H, L_{VLM-Nav})} \tag{10}$$

From the table, it is evident that despite being trained in the simple environment, the navigator model achieves an almost perfect TCR in both the blocks and downtown west environments. Additionally, the system maintains a very low collision rate for static obstacle, though collisions tend to increase when encountering smaller obstacles or when the depth map estimation is inaccurate. For this reason we have also observed higher collision rate with DepthAnything V1 model compared to V2. The low TLR values suggest that the system closely follows the human-guided navigation path. The inference time of VLM-Nav is shown in Table 9. While components like depth estimation and POD have a minimal impact on the total time, the VLM module consumes the most time, approximately 1 second, primarily due to the delay from the API call. Despite this, the VLM significantly improves system performance, making it domain-invariant and reliable in unseen environments. To address this, we use POD, which skips the API call when no frontal obstacle is detected, reducing the total runtime to just 108 milliseconds.

5.3 Qualitative analysis

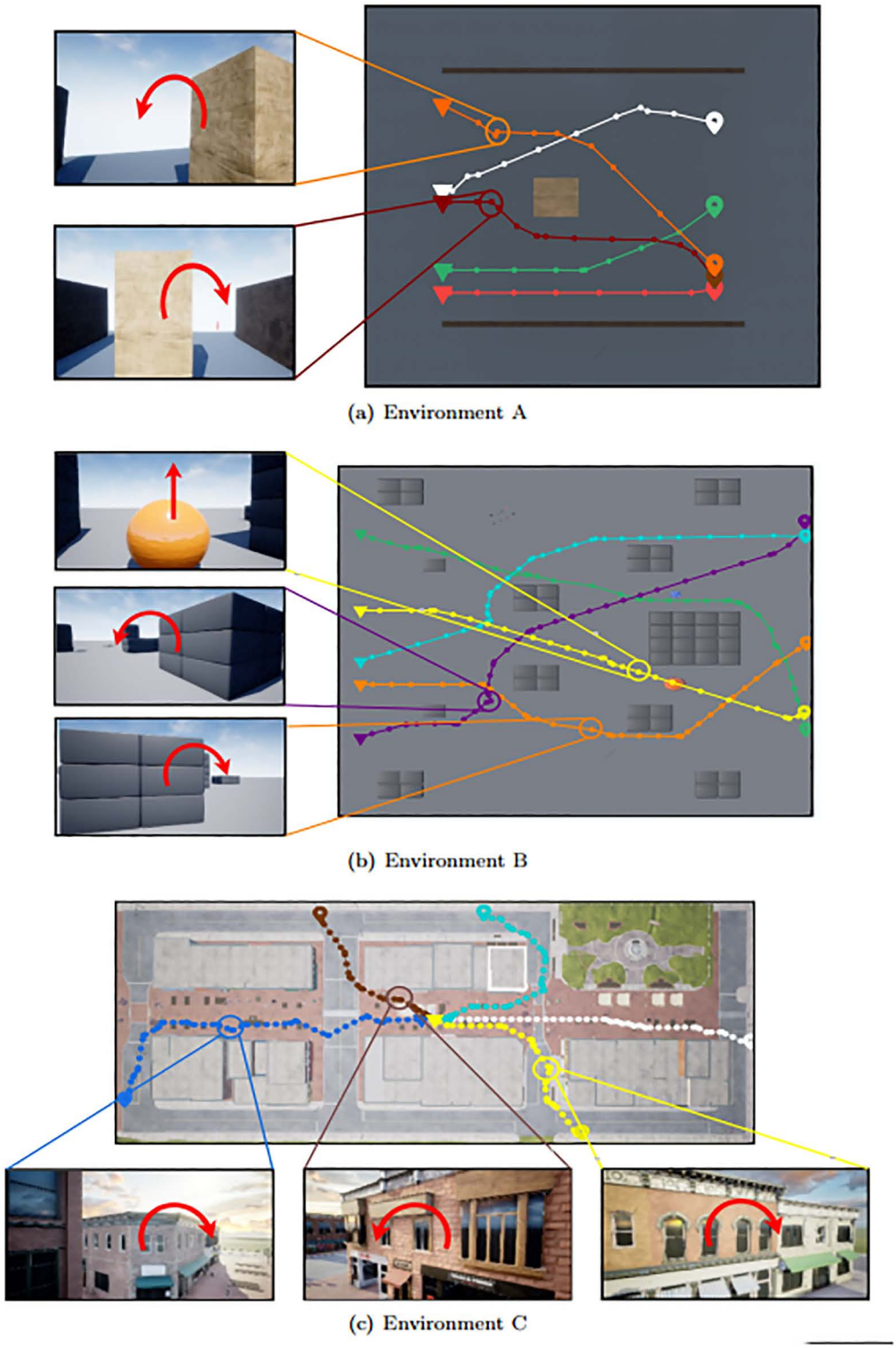
After training the navigator model, the approach is validated in environments B and C within Unreal Engine. Fig 10 shows a top view of the three environments (simple, blocks, and downtown west). The flight paths of ten consecutive flights following the proposed model are illustrated with a colored plot, where each color represents a different episode. In the simple and block environments, the spawn and target coordinates are randomly selected from fixed start and end zones, which can be seen at the top and bottom of Fig 10A and Fig 10B, respectively. However, in the downtown west environment, the starting zone is in the middle of the map, and the target coordinate is selected around the perimeter edges of the map. The proposed method successfully navigates to the target location without colliding with any obstacles while keeping the flight path as short as possible. The navigator model attempts to replicate human flight patterns based on inputs from other components of our system. Fig 11 illustrates three flight paths generated by VLM-Nav after training the navigator model, which is then compared to a human-guided flight from the same starting point to the target location. It is evident that the flight paths generated by our system closely resemble the human-controlled flights in both the blocks and downtown west environments.

An additional observation we made was that when complex instructions are given to VLM, it produces inconsistent output. We have implemented faster, simpler prompts for each API call to reduce the impact of these problems and ensure more reliable results. In addition, it is essential to keep in mind that not all VLMs are capable of managing depth maps.

Table 9. Inference time of VLM-Nav.

Module	Time (mS)
State collection	25
Depth Estimation	24.94
Proximal Object Detection	1
VLM Feedback	1000
Navigator Model	1
Total	51.94 / 1052

https://doi.org/10.1371/journal.pone.0345778.t009



(a) Environment A

(b) Environment B

(c) Environment C

Fig 10. Five example flight paths generated by VLM-Nav in three different environments are presented. The starting and target coordinates are marked by (▼) and (📍) symbols, respectively. Selected points (indicated by circles) along the flight paths are shown from the UAV's front camera perspective. At these points, the UAV's movement direction, taken to avoid obstacles, is depicted with red arrow symbols (e.g., ↷ for yaw right, ↶ for yaw left, and ↑ to go upward).

https://doi.org/10.1371/journal.pone.0345778.g010

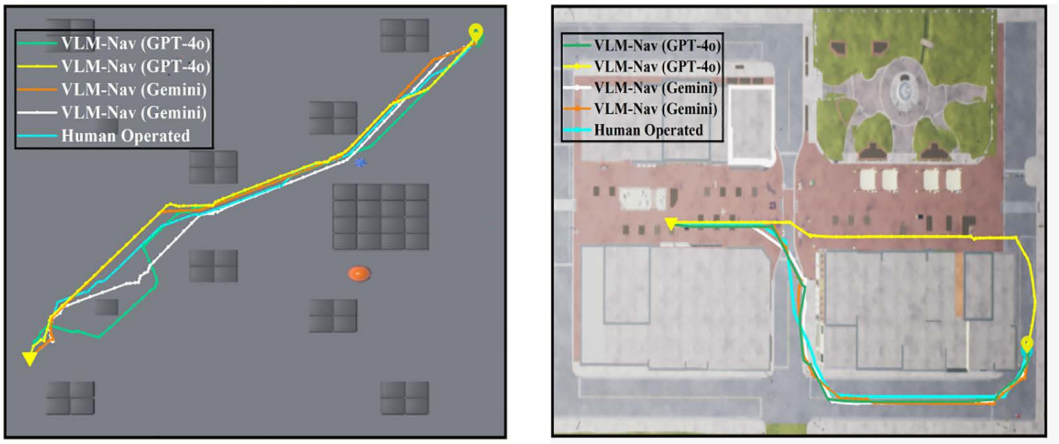


Fig 11. The flight path comparison between VLM-Nav and human-controlled flight in (a): Blocks and (b) Downtown West environment. The starting and target location is shown using color (▼) and (📍) markers.

https://doi.org/10.1371/journal.pone.0345778.g011

Therefore, selecting a VLM that supports depth maps is essential for applications that rely on depth information for accurate decision-making and action execution.

5.4 Comparative analysis

In recent studies, many researchers focus on qualitative analysis without directly reporting key metrics such as test time and computational cost. The study in [49] addresses the challenge of UAVs struggling to generalize in unseen environments. Traditional DRL models often rely on non-causal features, which are not directly related to the task, leading to performance degradation in new settings. The paper introduces Causal Feature Selection (CFS), which prioritizes causal features that directly impact navigation and obstacle avoidance. By filtering out non-causal features, the UAV's model becomes more adaptable, improving the Success Rate (SR) from 60% to 85% in unseen scenarios. However, the method only works with simple geometric obstacles and has low computational cost. In [50], Wei et al. proposed an approach using Dual-Transformer Encoders within PPO to improve multi-UAV collaboration.

This method increased transferability by 30% and obstacle avoidance success by 20% (from 70% to 90%) in dynamic, complex environments. In contrast, Javaid et al. [51] presented a monocular vision-based obstacle-avoidance system for UAVs that uses depth-estimation models to detect obstacles. This method achieves high SR across varying light conditions (83.33% in high light, 100% in moderate light), though it struggles in low-light and dynamic environments. It is cost-effective but has limitations in extreme lighting. LMNav, developed by Shah et al. [42], enables UAVs to follow natural language instructions using pre-trained models such as GPT-3, CLIP, and ViNG. Unlike traditional models, it doesn't require fine-tuning and is more generalizable, but it is computationally expensive and struggles with visual grounding and landmark recognition in complex settings [52]. In comparison, VLM-Nav excels at generalizing to unseen environments, with a high task completion rate. It is a lightweight, fast method that requires no extensive pretraining, making it more efficient than models like LMNav, which rely on large, computationally intensive pre-trained models. Table 10 highlights how VLM-Nav outperforms these methods, offering a more efficient and adaptable solution for UAV navigation.

5.5 Ablation study

This section explains how each component contributes to the navigation system as a whole. Three cases, illustrated in Fig 12A-Fig 12C are presented to demonstrate this. While the VLM module has strong reasoning abilities for detecting

Table 10. Comparison of our approach with recent studies in UAV navigation.

Author	Computing Cost	Inference Time	Result	Pros	Cons
Zhuang *et al.* [49]	Low–moderate	Low–moderate	85% SR in unseen env.	• Generalizes better by filtering out irrelevant features (background texture). • Improved accuracy on unseen environments.	• May require significant preprocessing. • Implemented with simple geometric obstacles.
Wei *et al.* [50]	Moderate–high	High	90% SR	• Improves generalization of multi-UAV systems in unseen environments. • Increased transferability and obstacle avoidance success rate.	• Computationally intensive due to the use of dual transformer encoders.
Javaid *et al.* [51]	High	Approx. 3s	80–100% SR (lighting-dependent)	• Cost-effective and lightweight. • Capable of operating in low light conditions up to a certain threshold.	• May experience false positives/accidents in challenging conditions (e.g., reflective surfaces). • Requires many processing steps which adds delay.
Shah *et al.* [42]	High	High	85% SR	• Capable of generalizing to new environments without requiring fine-tuning. • Suitable for practical robotic navigation with long-horizon tasks (over 100 meters).	• Depends heavily on landmark-based navigation. • Requires human instructions throughout the navigation step.
Proposed VLM-Nav	Low	Low (52ms without & 1s with obstacles)	98% Success Rate	• Strong generalization in unseen maps. • Real-time navigation and obstacle avoidance. • Doesn't require extensive training.	• Works only with static obstacles.

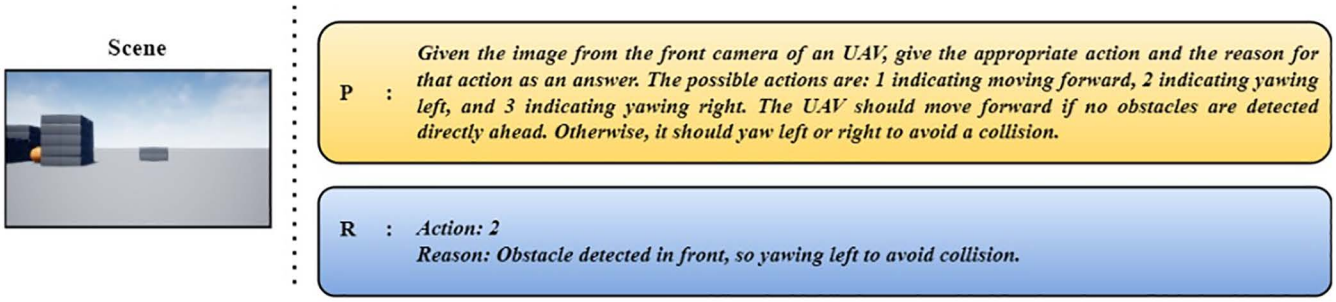
https://doi.org/10.1371/journal.pone.0345778.t010

and avoiding obstacles, it lacks spatial awareness, making depth map estimation necessary. Case 1 in Fig 12A, for example, the VLM is given an RGB scene image and asked to navigate by choosing one of three actions. As shown, despite the obstacle being far away, the UAV still chooses to rotate because it cannot determine the object's distance. Without depth information, detecting nearby obstacles in POD module is also not feasible.

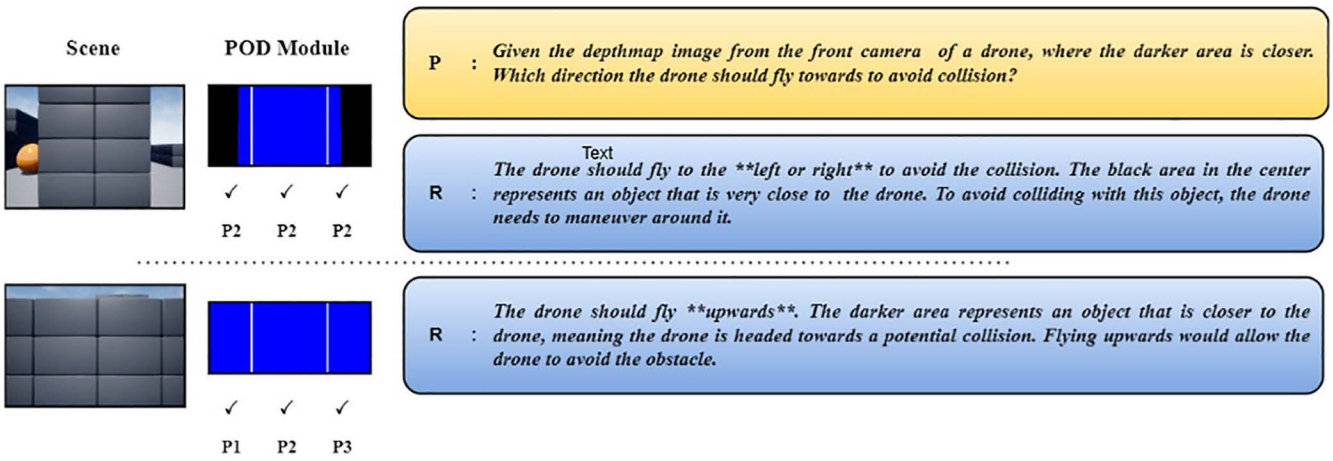
In our system, the goal of POD module is to determine whether there is any obstacle closeby in front of the UAV. The obstacle is represented by connected components and the depthmap is divided into three regions. However, relying solely on the proximal object detection module for navigation isn't sufficient. Often times, in situations where the drone is directly facing the obstacle at a close distance as shown in Case 2 in Fig 12B, decision-making becomes difficult. In such cases, the VLM analyzes the entire depth map and determines which direction is best for avoiding collisions. Although VLMs show great potential in various applications, their performance can be inconsistent, especially with complex prompts. This complexity often leads to variability in the VLMs' results, affecting the reliability of their outputs. The inconsistency is more evident when VLMs must follow detailed instructions. Case 3 in Fig 12C, for example, the VLM is given an image and angle ϕ to generate a navigation action. Despite the target being 1.5 radians to the right and the right side being obstacle-free, the VLM produces two different responses. Therefore, simpler prompts, as used in our system, yield more consistent and meaningful results. Then they are supported by the POD module and navigator model.

6 Conclusion

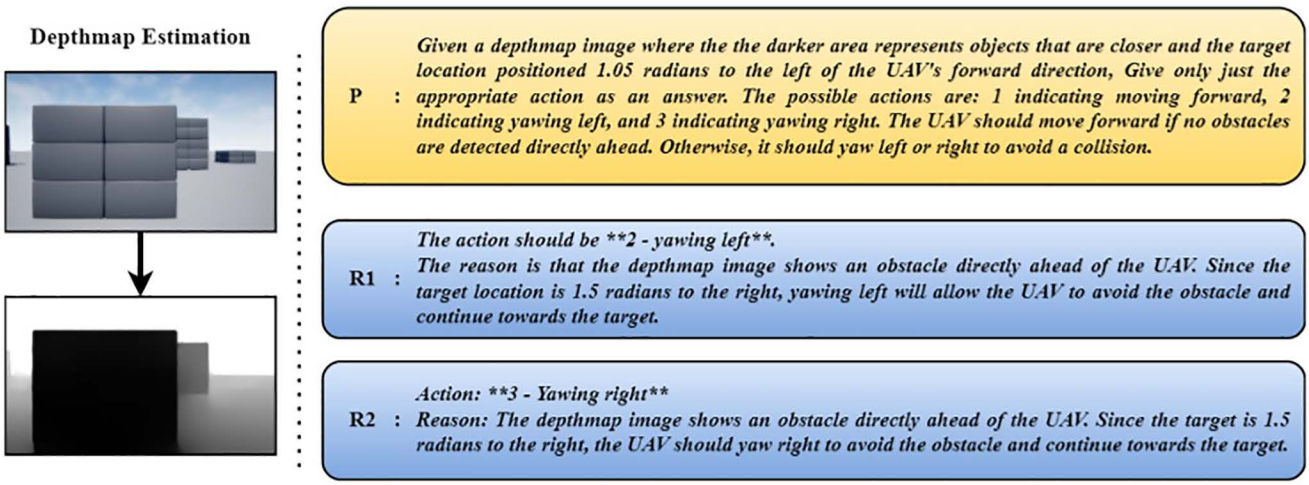
This research presents VLM-Nav, a cost-effective UAV navigation system that uses monocular vision and advanced vision-language models (VLM) for robust autonomous navigation. Using a single RGB camera with depth estimation eliminates the need for expensive sensors while maintaining high accuracy. Integrating state-of-the-art VLMs such as Gemini-flash-1.5 and GPT-4o improves decision making. The proposed approach is trained to mimic human controlled flight, which is then verified using the Airsim simulator in Unreal Engine. Using a zero-shot depth estimator and a VLM to avoid obstacles, this approach overcomes the limitation of generalization in unfamiliar environments highlighted in previous research. Future work will explore the deployment of lightweight, on-board vision–language models on edge computing



(a) (CASE 1) Without depth features in the scene images, incorrect actions may occur. Depth estimation helps reduce API call and aids VLM to select the optimal action.



(b) (CASE 2) Although POD module can detect frontal obstacle direction, it alone is not sufficient in many scenarios. Two such examples are shown in the figure.



(c) (CASE 3) This illustrates that the VLM struggles to produce consistent responses when given complex instructions. The Navigator model takes inputs from VLM feedback, distance sensors, and the POD module to select consistent and optimal action.

Fig 12. Ablation case studies.

https://doi.org/10.1371/journal.pone.0345778.g012

platforms such as NVIDIA Jetson Orin to reduce latency and enable fully offline operation. Extending the navigation module to continuous control is expected to generate smoother and more energy-efficient trajectories, improving flight stability and performance. Further research could also investigate creating a specialized VLM for UAV navigation that functions without depth estimation and can interpret complex instructions for consistent results. Dynamic obstacle avoidance can also be explored with the help of VLM. For example, it might be possible to use the generated depthmap as input for models such as spatio-temporal graph neural network and considering multiple timesteps to predict the movement pattern of objects. The proposed method operates independently of the instructions of the human pilot, providing valuable information and directions for integrating VLMs into autonomous UAV systems for practical applications such as parcel delivery, surveillance, and disaster management.

Author contributions

Conceptualization: Gobinda Chandra Sarker, Md Mehedi Hasan.

Data curation: Gobinda Chandra Sarker, Md Mehedi Hasan.

Formal analysis: Gobinda Chandra Sarker, Sejuti Rahman.

Investigation: Gobinda Chandra Sarker, AKM Azad.

Methodology: Gobinda Chandra Sarker, Md Mehedi Hasan, Sejuti Rahman.

Project administration: AKM Azad, Md Mehedi Hasan.

Resources: AKM Azad, Md Mehedi Hasan.

Supervision: Sejuti Rahman, Md Mehedi Hasan.

Validation: AKM Azad, Md Mehedi Hasan.

Writing – original draft: Gobinda Chandra Sarker, Md Mehedi Hasan.

Writing – review & editing: AKM Azad, Sejuti Rahman.

References

1. Arafat MY, Alam MM, Moh S. Vision-based navigation techniques for unmanned aerial vehicles: Review and challenges. Drones. 2023;7(2):89.
2. Mohsan SAH, Othman NQH, Li Y, Alsharif MH, Khan MA. Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends. Intelligent Service Robotics. 2023;16(1):109–37.
3. Talha M, Hussein A, Hossny M. Autonomous UAV Navigation in Wilderness Search-and-Rescue Operations Using Deep Reinforcement Learning. In: Australasian Joint Conference on Artificial Intelligence. Springer; 2022. p. 733–46.
4. Lu Y, Xue Z, Xia GS, Zhang L. A survey on vision-based UAV navigation. Geo-spatial information science. 2018;21(1):21–32.
5. Chowdhury AM, Imran A, Hasan MM, Ahmed R, Azad A, Alyami SA. U-ActionNet: Dual-Pathway Fourier Networks With Region-of-Interest Module for Efficient Action Recognition in UAV Surveillance. IEEE Access. 2024;12:189547–63.
6. Lin HY, Peng XZ. Autonomous quadrotor navigation with vision based obstacle avoidance and path planning. IEEE Access. 2021;9:102450–9.
7. Cui Z, Wang Y. UAV path planning based on multi-layer reinforcement learning technique. IEEE Access. 2021;9:59486–97.
8. Chowdhury AM, Imran A, Hasan MM. FFT-UAVNet: FFT Based Human Action Recognition for Drone Surveillance System. In: 2023 5th International Conference on Sustainable Technologies for Industry 5.0 (STI); 2023. p. 1–6.
9. Tsai CY, Nisar H, Hu YC. Mapless LiDAR navigation control of wheeled mobile robots based on deep imitation learning. IEEE Access. 2021;9:117527–41.
10. Zhu K, Zhang T. Deep reinforcement learning based mobile robot navigation: A review. Tsinghua Science and Technology. 2021;26(5):674–91.
11. Aburaya A, Selamat H, Muslim MT. Review of vision-based reinforcement learning for drone navigation. International Journal of Intelligent Robotics and Applications. 2024:1-19.
12. Fu C, Xu X, Zhang Y, Lyu Y, Xia Y, Zhou Z, et al. Memory-enhanced deep reinforcement learning for UAV navigation in 3D environment. Neural Computing and Applications. 2022;34(17):14599–607.
13. Chen Y, González-Prelcic N, Heath RW. Collision-free UAV navigation with a monocular camera using deep reinforcement learning. In: 2020 IEEE 30th international workshop on machine learning for signal processing (MLSP). IEEE; 2020. p. 1–6.

14. Wang J, Wang T, He Z, Cai W, Sun C. Towards better generalization in quadrotor landing using deep reinforcement learning. Applied Intelligence. 2023;53(6):6195–213.
15. Korkmaz E. Deep reinforcement learning policies learn shared adversarial features across mdps. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36; 2022. p. 7229–38.
16. Ladosz P, Weng L, Kim M, Oh H. Exploration in deep reinforcement learning: A survey. Information Fusion. 2022;85:1–22.
17. Zhang J, Huang J, Jin S, Lu S. Vision-language models for vision tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2024.
18. Zhang Y, Ma Z, Li J, Qiao Y, Wang Z, Chai J, et al. Vision-and-Language Navigation Today and Tomorrow: A Survey in the Era of Foundation Models. Transactions on Machine Learning Research. 2024. Survey Certification. Available from: https://openreview.net/forum?id=yiqeh2ZYUh
19. Imran A, Ahmed R, Hasan MM, Ahmed MHU, Azad A, Alyami SA. FaceEngine: A Tracking-Based Framework for Real-Time Face Recognition in Video Surveillance System. SN Computer Science. 2024;5(5):609.
20. Kim M, Kim J, Jung M, Oh H. Towards monocular vision-based autonomous flight through deep reinforcement learning. Expert Systems with Applications. 2022;198:116742.
21. Xue Z, Gonsalves T. Vision based drone obstacle avoidance by deep reinforcement learning. AI. 2021;2(3):366–80.
22. Kim Y, Back S, Song D, Lee BY. Aerial Map-Based Navigation by Ground Object Pattern Matching. Drones. 2024;8(8):375.
23. Jun M, Lilian Z, Xiaofeng H, Hao Q, Xiaoping H. A 2d georeferenced map aided visual-inertial system for precise uav localization. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE; 2022. p. 4455–62.
24. Wei J, Karakay D, Yilmaz A. A Gis Aided Approach for Geolocalizing an Unmanned Aerial System Using Deep Learning. In: 2022 IEEE Sensors. IEEE; 2022. p. 1–4.
25. Shan T, Englot B, Meyers D, Wang W, Ratti C, Rus D. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE; 2020. p. 5135–42.
26. Song S, Zhang Y, Qin X, Saunders K, Liu J. Vision-guided collision avoidance through deep reinforcement learning. In: NAECON 2021-IEEE National Aerospace and Electronics Conference. IEEE; 2021. p. 191–4.
27. Liu Z, Cao Y, Chen J, Li J. A Hierarchical Reinforcement Learning Algorithm Based on Attention Mechanism for UAV Autonomous Navigation. IEEE Transactions on Intelligent Transportation Systems. 2022.
28. Li X, Fang J, Du K, Mei K, Xue J. UAV Obstacle Avoidance by Human-in-the-Loop Reinforcement in Arbitrary 3D Environment. arXiv preprint arXiv:230405959. 2023.
29. Liu S, Zhang H, Qi Y, Wang P, Zhang Y, Wu Q. AerialVln: Vision-and-language navigation for uavs. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2023. p. 15384–94.
30. Schumann R, Zhu W, Feng W, Fu TJ, Riezler S, Wang WY. Velma: Verbalization embodiment of llm agents for vision and language navigation in street view. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38; 2024. p. 18924–33.
31. Rajvanshi A, Sikka K, Lin X, Lee B, Chiu HP, Velasquez A. Saynav: Grounding large language models for dynamic planning to navigation in new environments. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 34; 2024. p. 464–74.
32. Lee TJ, Yi DH, Cho DID. A monocular vision sensor-based obstacle detection algorithm for autonomous robots. Sensors. 2016;16(3):311.
33. Badrloo S, Varshosaz M, Pirasteh S, Li J. Image-based obstacle detection methods for the safe navigation of unmanned vehicles: A review. Remote Sensing. 2022;14(15):3824.
34. Rostum HM, Vásárhelyi J. A review of using visual odometery methods in autonomous UAV Navigation in GPS-Denied Environment. Acta Universitatis Sapientiae, Electrical and Mechanical Engineering. 2023;15(1):14–32.
35. Padhy RP, Sa PK, Narducci F, Bisogni C, Bakshi S. Monocular vision aided depth measurement from RGB images for autonomous UAV navigation. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM). 2022.
36. Doukhi O, Lee DJ. Deep reinforcement learning for autonomous map-less navigation of a flying robot. IEEE Access. 2022;10:82964–76.
37. Wei Y, Zhang H, Wang Y, Huang C. Maneuver Decision-Making through Automatic Curriculum Reinforcement Learning without Handcrafted Reward Functions. Applied Sciences. 2023;13(16):9421.
38. An G, Wu Z, Shen Z, Shang K, Ishibuchi H. Evolutionary Multi-Objective Deep Reinforcement Learning for Autonomous UAV Navigation in Large-Scale Complex Environments. In: Proceedings of the Genetic and Evolutionary Computation Conference; 2023. p. 633–41.
39. Singla A, Padakandla S, Bhatnagar S. Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge. IEEE transactions on intelligent transportation systems. 2019;22(1):107–18.
40. Yang L, Kang B, Huang Z, Xu X, Feng J, Zhao H. Depth anything: Unleashing the power of large-scale unlabeled data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2024. p. 10371–81.
41. Birkl R, Wofk D, Müller M. Midas v3. 1–a model zoo for robust monocular relative depth estimation. arXiv preprint arXiv:230714460. 2023.
42. Shah D, Osiński B, Levine S, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In: Conference on robot learning. PMLR; 2023. p. 492–504.

43. Shah S, Dey D, Lovett C, Kapoor A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: Field and Service Robotics; 2017. http://arxiv.org/abs/arXiv:1705.05065
44. Zhou B, Wang W, Liu Z, Wang J. Vision-based navigation of uav with continuous action space using deep reinforcement learning. In: 2019 Chinese Control And Decision Conference (CCDC). IEEE; 2019. p. 5030–5.
45. Yang L, Kang B, Huang Z, Zhao Z, Xu X, Feng J, et al. Depth Anything V2. arXiv preprint arXiv:240609414. 2024.
46. Islam R, Moushi OM. Gpt-4o: The cutting-edge advancement in multimodal llm. Authorea Preprints. 2024.
47. Team G, Anil R, Borgeaud S, Wu Y, Alayrac JB, Yu J, et al. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:231211805. 2023.
48. Bolelli F, Allegretti S, Baraldi L, Grana C. Spaghetti labeling: Directed acyclic graphs for block-based connected components labeling. IEEE Transactions on Image Processing. 2019;29:1999–2012.
49. Zhuang J, Han G, Xia Z, Wang B, Li W, Wang D, et al. Robust policy learning for multi-uav collision avoidance with causal feature selection. arXiv preprint arXiv:240704056. 2024.
50. Wei A, Liang J, Lin K, Li Z, Zhao R. DTPPO: Dual-Transformer Encoder-Based Proximal Policy Optimization for Multi-UAV Navigation in Unseen Complex Environments. Drones. 2024;8(12).
51. Javaid A, Alnaser M, Baroudi U, Alfaraj A. Monocular vision navigation system for UAV autonomous mission: a real-time window-based obstacle avoidance approach. Neural Computing and Applications. 2025:1–18.
52. Hasan MM, Arnold JF, Frater MR. Subjective evaluation and statistical analysis for improved frame-loss error concealment of 3D videos. In: 2014 IEEE Visual Communications and Image Processing Conference; 2014. p. 181–4.