CO-SIMULATION-BASED EVALUATION OF UAV OBSTACLE AVOIDANCE USING 3DVFH* AND STEREO CAMERAS

Le Duong Khang, Nguyen Ngoc Trung, Ngo Viet Huy Hoang, Tran Thi Minh Hanh, Huynh Thanh Tung, Duy-Tuan Dao*

The University of Danang - University of Science and Technology, Vietnam

*Corresponding author: ddtuan@dut.udn.vn

(Received: May 01, 2025; Revised: June 10, 2025; Accepted: June 18, 2025)

DOI: 10.31130/ud-jst.2025.23(9B).498

Abstract - The paper aims to simulate UAVs in complex network environments to evaluate their performance in obstacle avoidance using stereo sensors. Since there are strict security regulations limiting real-world UAV flight testing, this research addresses the critical need for comprehensive simulation-based approaches.. The system integrates real-time obstacle detection and avoidance algorithms, enabling UAVs to navigate safely in environments with obstacles. Developed on the ROS framework with PX4 SITL and Gazebo, the system supports comprehensive end-to-end testing, from stereo image acquisition to UAV navigation using the 3DVFH* algorithm. It utilizes the MAVLink protocol for control and QGroundControl for monitoring. Simulation results confirm the system's effectiveness, establishing a solid foundation for advanced autonomous UAV navigation in real-world applications.

Key words - Drone simulation; PX4; Gazebo; ROS; QGroundControl.

1. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly utilized in various applications, including security, rescue, delivery, agriculture, and defense. To operate effectively in real-world environments, UAVs must autonomously avoid obstacles and maintain stable communication despite changing conditions. Wireless communication is crucial for maintaining reliable connections between UAVs and ground control stations, particularly in urban environments where signal degradation is prevalent [1, 2]. Therefore, simulating dynamic network conditions is essential for evaluating UAV performance in adverse scenarios [3, 4]. Recent studies have focused on evaluating UAV performance under degraded communication environments, such as latency, packet loss, and signal degradation [3, 4]. Platforms like XTDrone and PX4-Gazebo provide flexible and modular setups to simulate different UAV configurations in both communication and perception tasks [4, 8].

Obstacle avoidance is another key capability for autonomous flight. Stereo vision enables real-time depth estimation and object detection, allowing UAVs to navigate safely and avoid collisions [5 - 7]. For obstacle avoidance and perception, stereo and RGB-D vision systems have become standard for depth estimation and object detection [5, 6, 9]. Some methods enhance stereo vision systems for lightweight UAVs, including micro air vehicles (MAVs), enabling real-time onboard processing [9, 10].

Multi-sensor fusion approaches, such as combining stereo vision with radar, are also explored to improve obstacle detection reliability [11]. Systems like 3DVFH+ have been proposed to implement real-time 3D obstacle avoidance using OctoMap-based representations [12]. Meanwhile, UAV swarm simulation attracts coordinated applications in search and rescue, inspection, and environmental monitoring [13]. The proposed method aims to contribute to this evolving field by enabling the evaluation of stereo vision-based obstacle avoidance in the presence of network instability.

This paper proposes a UAV simulation framework that integrates two core components: dynamic wireless network modeling and real-time stereo vision-based obstacle avoidance using 3DVFH*. Built on PX4, ROS, and Gazebo, the framework supports testing under diverse conditions, contributing to developing more robust and adaptive UAV systems.

2. UAV Software Framework

This study conducts simulations in Software-in-the-Loop (SITL) mode, where the PX4 firmware runs entirely in software, receiving synthetic sensor inputs from Gazebo such as GPS, IMU, and stereo camera data. Communication occurs over UDP port 14560, forming a closed loop where Gazebo sends sensor data to PX4, and PX4 returns actuator commands to update the UAV's simulated state. Figure 1 illustrates the connection among these components. The Iris quadrotor, developed by 3D Robotics, is selected for its compatibility with PX4 SITL and robust support for onboard sensors, making it a reliable platform for evaluating autonomous navigation and control strategies.

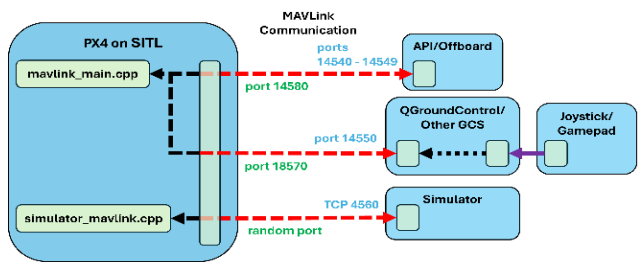

Figure 1. *Connection diagram of PX4, QgroundControl, and Gazebo simulator*

2.1. Ground Control System (QGroundControl)

The Ground Control Station (GCS) acts as the key interface between the operator and the UAV simulation. For this work, QGroundControl was chosen for its

seamless compatibility with PX4 and Gazebo, intuitive interface, and active support community [9]. It allows mission planning over MAVLink, permitting users to set UAV parameters.

QGroundControl includes tools like GeoFence for defining flight boundaries, Rally Points for emergency landings, and pattern generators for creating survey paths. Features such as flexible waypoint control and Goto Location support add further versatility. By providing robust planning and real-time control, QGroundControl offers a dependable environment for testing UAV navigation and control strategies in simulation.

2.2. Inter-System Communication

Within the PX4 architecture, network communication is structured two core mechanisms: uORB for internal data exchange and MAVLink for external communication. uORB is a publish-subscribe messaging system that facilitates asynchronous data flow between PX4 modules running in the same environment, such as flight controllers, sensor interfaces, and localization components [14]. Although uORB supports custom message definitions, this study utilizes only predefined PX4 message types [15]. For example, flight status and sensor data are subscribed to by internal modules, while outputs like obstacle avoidance commands are published on relevant topics.

On the external side, MAVLink serves as a lightweight communication protocol between PX4 and ground stations like QGroundControl. It supports the real-time exchange of telemetry, control commands, and system parameters. A key feature is the Mission Protocol, which enables the UAV to receive mission plans via a request-response cycle, where the UAV sends MISSION_REQUEST messages and the ground station replies with mission items containing position, altitude, and speed.

Additionally, MAVLink's Parameter Protocol allows dynamic updates of flight control parameters, such as velocity, GPS coordinates, and heading, during operation. Together, uORB and MAVLink provide a reliable and modular communication foundation that supports autonomous UAV functionality in both simulation and real-world deployments.

2.3. Simulation Tools in PX4 SITL

The PX4 flight control architecture allows seamless The PX4 flight control architecture enables seamless integration with simulation tools, allowing for the testing of UAV software in virtual environments without the need for physical hardware. This method is advantageous during early development, as it supports testing and debugging of autonomous behaviors under diverse environmental conditions. Gazebo is normally used for this purpose due to its realistic physics engine, modular design, and extensive library of UAV and sensor models. It communicates with QGroundControl via the MAVLink protocol for real-time mission planning and monitoring, providing a user-friendly GUI for defining waypoints and viewing system states.

2.4. Gazebo Virtual Environment

Gazebo [16] is a powerful 3D simulation tool widely used for developing and testing autonomous robotic systems. Its realistic physics and accurate sensor simulations make it ideal for evaluating UAV navigation and obstacle avoidance algorithms in a virtual environment. In this study, the Iris quadrotor was chosen for its support for PX4 SITL and built-in sensors such as GPS, IMU, and magnetometer, allowing for autonomous flight missions within the simulation environment.

Simulation scenarios were configured using the Simulation Description Format (SDF), which uses XML to describe simulation elements. The main.world file defines the environment layout and model references, while individual.sdf files specify each model's structure and properties. In this study, the Iris model with a ready-to-use iris.sdf file was selected due to its seamless PX4 SITL integration. Additionally, the stereo camera was used to simulate obstacle detection, enabling accurate environment interaction and perception modeling.

3. Obstacle Detection and Avoidance Simulation System

3.1. System Architecture Overview

The simulation system replicates the complete process of obstacle detection and reactive navigation during UAV missions using a modular architecture. It integrates Gazebo for environment simulation, PX4 SITL for flight control, ROS for processing and coordination, MAVLink for communication, and QGroundControl for mission monitoring. The overall architecture and data flow among components are illustrated in Figure 2.

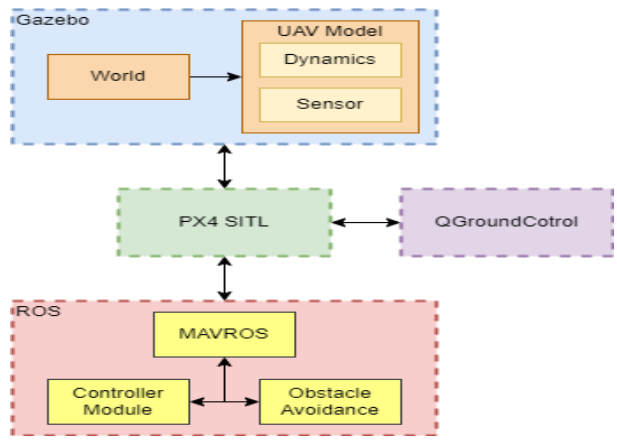

Figure 2. *Architecture of the simulation system with QGroundControl, PX4, ROS, and Gazebo*

3.2. Communication Between Components

In the system, PX4 functions as the primary flight controller, responsible for executing waypoints, maintaining altitude, and stabilizing the UAV. However, high-level decision-making functions, such as obstacle avoidance, are not managed by PX4; instead, they are entirely delegated to the ROS framework. ROS acts as the central processing platform, where independent nodes are responsible for tasks including image processing, local navigation, and control communication. Communication between ROS and PX4 is achieved through MAVROS, which serves as a bridge using the MAVLink protocol. This enables ROS to send control commands to PX4, such as updating waypoints, changing

flight modes, or initiating emergency landing without interrupting the ongoing mission.

Gazebo serves as a 3D physics simulation environment, where the UAV is deployed alongside simulated objects (such as walls, boxes, trees, etc.) to create real-world obstacle scenarios. The UAV uses the Iris model, which is equipped with a stereo camera to capture stereoscopic image data. The UAV's sensors and movements are synchronized with the simulation environment to ensure high realism in interactions.

QGroundControl is used as a real-time monitoring interface during simulation. It provides a visual overview of flight status, navigation behavior, and mission logs. Thanks to its integration with MAVLink, all control behaviors driven by the navigation algorithm are clearly reflected in the user interface.

3.3. Object detection using a stereo camera

In this study, we utilize the Gazebo simulation platform to construct a UAV system equipped with a stereo camera setup, aiming to evaluate obstacle perception through stereo image processing. The stereo camera is simulated by attaching two virtual cameras to the front of the UAV, capturing left and right images simultaneously. These images are processed using the ROS node stereo_image_proc [17] to generate a disparity map and point clouds. The disparity image is computed from incoming stereo pairs using OpenCV's block matching algorithm [18, 19] that uses the sum of absolute differences to measure the similarity between image blocks. Figure 3 illustrates a simulated scenario where the UAV is positioned in front of an obstacle, showing the corresponding left image, right image, and disparity result.

Figure 3. *Stereo camera simulation in Gazebo with an obstacle*

The disparity map clearly illustrates depth variations, with brighter regions indicating closer objects and darker areas representing farther distant ones. This contrast allows the system to estimate the relative position of obstacles within the camera's field of view. The Point Cloud Library (PCL) [20] is employed on disparity maps to generate 3D point clouds representing the relative distance to objects in the environment as presented in Equations 1, 2, and 3.

$$Z = \frac{f \cdot B}{d} \tag{1}$$

$$X = \frac{(u - c_x) \cdot Z}{f} \tag{2}$$

$$Y = \frac{(v - c_y) \cdot Z}{f} \tag{3}$$

In these equations, Z represents the distance from the camera to the point in space, while X and Y denote the spatial coordinates of the point. The parameter f refers to the camera's focal length, and B is the baseline, which is the distance between the two cameras. d is the disparity value. The coordinates (u, v) indicate the position of the point in the image plane, and $(c_x,\ c_y)$ represent the principal point, which is the origin of the image coordinate system.

Figure 4 shows point clouds in 3D coordinates. Each point corresponds to a depth-estimated feature in the stereo image pair. These 3D points are expressed relative to the left camera's coordinate system, with depth increases along the Z-axis, lateral position along the X-axis, and vertical position along the Y-axis.

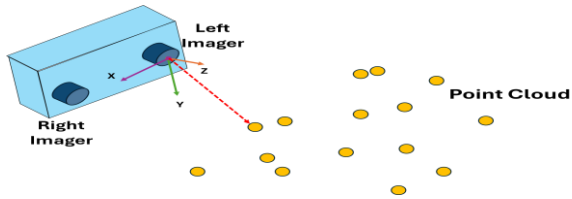

Figure 4. *3D point clouds [17]*

*3.4. Obstacle Avoidance Algorithm Using 3DVFH**

The system employs the 3D Vector Field Histogram Star (3DVFH*) algorithm [20] to navigate the UAV in an unknown environment. This is a path planning algorithm operating in three-dimensional space, which combines a two-dimensional polar histogram with the A* search algorithm [21] to determine the optimal flight direction. Input data is obtained from a stereo camera.

An overview of this process is shown in Figure 5. The operational procedure of the 3DVFH* algorithm involves the following steps:

Transforming the point cloud into a polar histogram:

All point clouds within the scanning range of the stereo camera are converted into polar coordinates, characterized by two angles: yaw angle and pitch angle. These angles are illustrated in Equations 4 and 5.

Yaw angle (azimuth angle):

$$\beta_z = \left\lfloor \frac{1}{\alpha} arctan \arctan\left(\frac{x_i - x_0}{y_i - y_0}\right) \right\rfloor \tag{4}$$

Pitch angle (elevation angle):

$$\beta_e = \left\lfloor \frac{1}{\alpha} arctan\ arctan\ \left(\frac{z_i - z_0}{\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}}\right) \right\rfloor \tag{5}$$

where (x_0, y_0, z_0) denotes the current UAV position, (x_i, y_i, z_i) is the coordinates of the i^{th} obstacle point, and α is the angular resolution between two adjacent cells in the histogram.

Constructing the 2D Polar Histogram:

Each obstacle point is assigned to a cell in the direction grid (e, z), corresponding to the discrete pitch and yaw angles. For each cell, the algorithm computes the average distance $hist(e, z)$ to the obstacles in the (e, z) direction of the cell as in Equation 6.

$$hist(e, z) = \frac{1}{N} \sum_{i=1}^{N} r_i \tag{6}$$

where N is the number of points that fall within the cell and r_i is the distance from UAV to the i^{th} obstacle point in that cell.

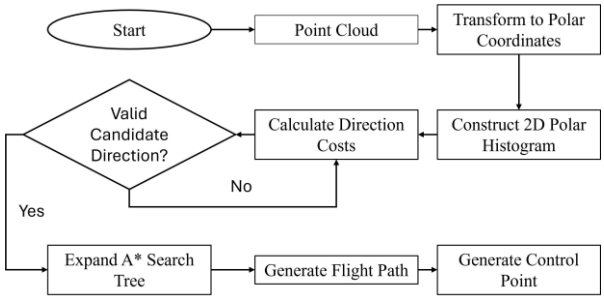

Figure 5. *Flowchart of the UAV obstacle avoidance process using stereo camera data*

Calculating the Cost Function for Each Flight Direction

For each cell (e, z) that is not blocked by obstacles, the algorithm calculates the total flight direction cost using Equation 7.

$$cost_{total} = cost_{yaw} + cost_{pitch} + cost_{vel} + cost_{obst} \quad (7)$$

Components are defined as in Equations 8, 9, 10, and 11. While yaw cost ($cost_{yaw}$) reflects the deviation of the candidate direction from the target heading, pitch cost ($cost_{pitch}$) reflects the deviation in the elevation angle from the target. In addition, velocity cost ($cost_{vel}$) evaluates the deviation of the candidate direction from the current velocity direction. Finally, obstacle cost ($cost_{obst}$) penalizes directions close to obstacles:

$$cost_{yaw} = k_{yaw} \cdot \left(\beta_z - \beta_{z_{goal}}\right)^2 \quad (8)$$

$$cost_{pitch} = k_{pitch} \cdot \left(\beta_e - \beta_{e_{goal}}\right)^2 \quad (9)$$

$$cost_{vel} = k_{vel} \cdot \left[\|\vec{v}\| - \hat{d}_{candidate} \cdot \vec{v}\right] \quad (10)$$

$$cost_{obst} = 5000 \times \left(1 + \frac{(k_{obst} - d_{obs})}{\sqrt{1 + (k_{obst} - d_{obs})^2}}\right) \quad (11)$$

these equations, k_{yaw}, k_{pitch}, k_{vel}, k_{obst} are weighting factors that regulate the influence of each cost component. The terms β_z, β_e represent the yaw and pitch angles of the candidate direction respectively, while $\beta_{z_{goal}}$, $\beta_{e_{goal}}$ denote the yaw and pitch angles from the UAV to the goal. $\vec{v}$ is the current velocity vector of the UAV. $\hat{d}_{candidate}$ is the unit vector of the candidate direction and d_{obs} is the distance from the UAV to the nearest obstacle in the candidate direction.

Build Look-Ahead Search Tree

The algorithm expands the search tree from the current UAV position along candidate directions with low cost. Each node in the tree contains: position p_n, accumulated cost $g(n)$, heuristic cost $h(n)$ to the goal. The cost at each node is calculated based on the A* principle:

$$f(n) = g(n) + h(n) \quad (12)$$

where $g(n)$ is the total cost from the root to node n which is derived from $cost_{total}$, $h(n)$ is calculated as the Euclidean distance to the goal as illustrated in Equation 13.

$$h(n) = \|p_n - p_{goal}\| \quad (13)$$

where p_goal is a local target point positioned ahead of the UAV along its intended flight direction. It is computed from the UAV's current position and heading, typically at a fixed look-ahead distance. This point serves as a reference for evaluating candidate directions in the 3DVFH algorithm, including the calculation of desired yaw and pitch angles and the heuristic function $h(n)$, which estimates the remaining cost to the target. The expansion of the search tree stops when a node reaches the maximum depth or is located near the target.

Generate Flight Path

From the node with the maximum depth or lowest cost in the tree, the algorithm traces back through the parent nodes to form a feasible flight path:

$$path = \{p_0, p_1, \ldots, p_n\} \quad (14)$$

Generate Control Point (Setpoint)

Based on the generated path and the time elapsed since path generation, the algorithm computes the current control point using interpolation:

$$s(t) = p_i + \left(v \cdot \frac{\Delta t}{\|p_{i+1} - p_i\|}\right) \cdot (p_{i+1} - p_i) \quad (15)$$

where $s(t)$ is the point that the UAV should move toward, Δt is time elapsed since the path was generated, v is UAV velocity and p_i, p_{i+1} are two consecutive points on the path.

The 3DVFH* algorithm allows the UAV to navigate in unknown environments, quickly respond to new obstacles, and ensure smooth and safe flight in complex spaces such as dense forests, urban areas, or enclosed buildings.

4. Obstacle Avoidance Evaluation

To comprehensively assess obstacle avoidance performance, three key metrics are utilized: collision rate, minimum obstacle distance, and computational efficiency. The results show that in various flight scenarios, the collision rate is always zero, demonstrating the system's reliable obstacle avoidance ability. The system maintains a stable safety buffer by setting the sensor's detection range between 0.2 m and 15.0 m, ensuring a balance between real-time feedback and calculation ability. Computational efficiency analysis confirmed approximately 87,360 operations per planning cycle, suitable for embedded platforms with latency typically under 20 ms per cycle.

4.1. Test Scenarios

Based on the evaluation metrics described above, two simulation scenarios were designed to validate the UAV's obstacle avoidance performance under different environmental conditions: one with no obstacles and another with dense obstacles. These test cases allow us to examine the system's stability, trajectory control, and reactive behavior when facing potential collisions.

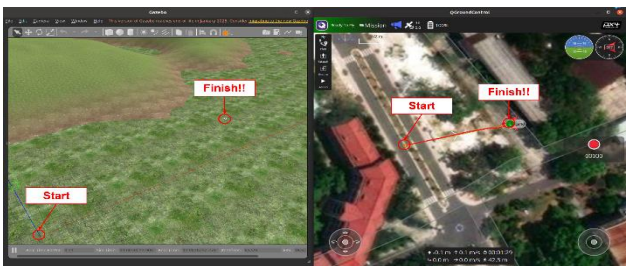

Figure 6. *The flight path of UAV from the Start point to the Finish point in an environment without obstacles*

Figure 7. *The flight path of UAV from Start point to Finish point in an environment with obstacles*

Figure 6 illustrates the UAV's path in an obstacle-free environment. The UAV follows a simple mission: take off at Start, fly directly to the Finish point, and land without encountering obstacles. This scenario verifies the controller's stability, waypoint accuracy, and smooth trajectory in ideal conditions.

In contrast, Figure 7 shows the UAV's path in an obstacle-rich environment, where the same mission is performed but with added obstacles requiring real-time avoidance via stereo vision. The curved red path from Start to Finish demonstrates adaptive behavior. In this setup, each simulated tree obstacle is approximately 4 meters tall, while the UAV is configured to fly at a fixed altitude of 2 meters. This configuration creates a realistic challenge where the UAV must continuously detect and avoid obstacles within its flight path. The scenario thus evaluates the system's ability to perform detection, reaction, and safe path planning. The UAV's rerouting in Figure 7 is further corroborated by variations in velocity, pitch, and yaw shown in Figures 9, 11, and 13, respectively, confirming effective avoidance behavior under dynamic conditions. These behaviors are consistent with the evaluation metrics defined, demonstrating that the UAV successfully avoided collisions, preserved safe separation from obstacles, and operated within acceptable computational latency throughout the obstacle-rich scenarios.

4.2. Flight Velocity Analysis

Figure 8 illustrates the velocity graph for the obstacle-free flight throughout the entire mission. As can be seen in this figure, the velocity remains nearly stable, with no abnormal fluctuations. Significant changes mainly occur during take-off and landing phases, while the UAV maintains a steady speed along all three axes during the mission. There are no signs of sudden deceleration or direction reversal, indicating that the UAV does not need to brake or change course abruptly, consistent with a straight, unobstructed flight mission.

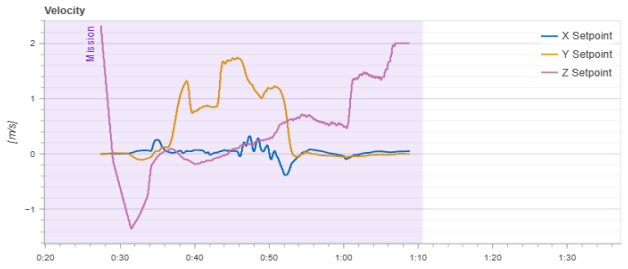

Figure 8. *Velocity graph during obstacle-free flight*

Figure 9 illustrates the velocity graph for the flight with obstacles throughout the entire mission. The UAV slows down, stops, adjusts its flight direction, and then accelerates again to avoid collisions. Greater velocity fluctuations confirm continuous adjustments made to ensure the UAV completes its mission safely. The sudden deceleration and directional changes are key indicators of obstacle presence along the flight path.

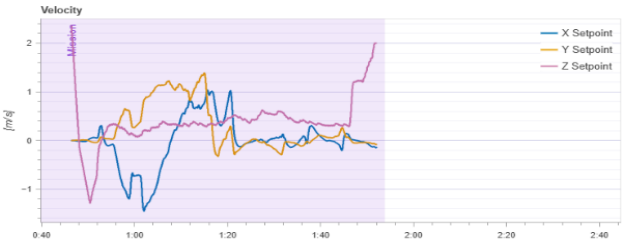

Figure 9. *Velocity graph during obstacle-filled flight*

4.3. Pitch Angle Analysis

Figures 10 and 11 illustrate pitch angles in two scenarios. The UAV exhibits minor oscillations after achieving stable flight, without any significant fluctuations in pitch angle in an obstacle-free environment. Changes occur only during the initiation and completion phases of the mission, but no large or abnormal deviations are observed. However, in an environment with obstacles, the pitch angle shows clear variations, indicating that the UAV had to adjust its direction continuously to avoid obstacles. Compared to obstacle-free flight, the changes in pitch angle are more pronounced, with distinct peaks and troughs. This is consistent with a flight path that requires frequent and adjustments due to obstacle avoidance.

Figure 10. *Pitch angle graph during obstacle-free flight*

Figure 11. *Pitch angle graph during obstacle-filled flight*

4.4. Yaw Angle Analysis

Figures 12 and 13 show the yaw angle of the UAV for the obstacle-free and obstacle-filled flight throughout the entire mission, respectively. As shown in Figure 12, the UAV's yaw angle exhibits slight fluctuations but remains smooth, with no abrupt or abnormal changes. This indicates that the system maintains a stable flight heading for most of the mission, adequately meeting basic trajectory control requirements. However, at certain moments, the UAV performs minor yawing motions to scan and inspect the surrounding environment using the stereo camera. This is normal behavior that supports navigation and does not significantly affect overall flight

stability. In contrast, the UAV undergoes more frequent and sudden yaw changes, with continuous directional adjustments as shown in Figure 13. The pronounced yaw angle oscillations reflect that the UAV had to change its heading multiple times to avoid obstacles, demonstrating that the navigation system is effectively enabling the UAV to detect, react to, and safely correct its path.

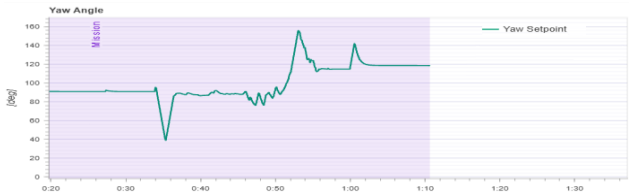

Figure 12. *Yaw angle graph during obstacle-free flight*

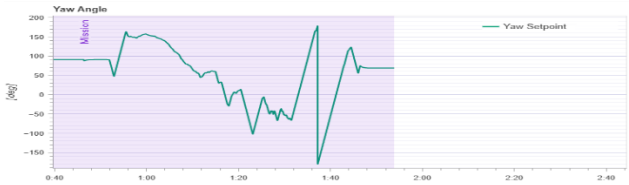

Figure 13. *Yaw angle graph during obstacle-filled flight*

5. Conclusion

This study developed a complete UAV simulation system using stereo camera sensors for real-time obstacle detection and avoidance. Built on the ROS framework with PX4 SITL and Gazebo, the system enables end-to-end testing from stereo image acquisition to UAV navigation using the 3DVFH* algorithm, with control via the MAVLink protocol and monitoring through QGroundControl. The results demonstrated the system's effectiveness, providing a strong foundation for advanced autonomous UAV navigation in real-world scenarios.

Future research will extend the current system by benchmarking the 3DVFH* algorithm against classical planners such as A* and RRT* to better understand trade-offs in efficiency and safety in constrained 3D environments. In parallel, efforts will focus on optimizing stereo image processing to improve computational performance and output quality, through techniques such as image denoising, disparity stabilization, and enhanced point cloud generation. The system will also integrate deep learning for improved spatial awareness and be tested in increasingly complex environments like urban areas, forests, and indoor spaces.

Acknowledgment: This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.04-2023.40.

REFERENCES

[1] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges", **IEEE Commun. Mag.**, vol. 54, no. 5, pp. 36–42, May 2016. DOI: 10.1109/MCOM.2016.7470933.

[2] I. Bekmezci, O. K. Sahingoz, and S. Temel, "Flying ad-hoc networks (FANETs): A survey*", *Ad Hoc Netw.**, vol. 11, no. 3, pp. 1254–1270, May 2013. DOI: 10.1016/j.adhoc.2012.12.004.

[3] D. Fabio, C. Santoro, and F. F. Santoro, "An integrated framework for the realistic simulation of multi-UAV applications", *Computers & Electrical Engineering,* vol. 74, pp. 196-209, 2019. https://doi.org/10.1016/j.compeleceng.2019.01.016.

[4] K. Xiao, S. Tan, G. Wang, X. An, X. Wang, and X. Wang, "XTDrone: A customizable multi-rotor UAVs simulation platform", *in Proc. Int. Conf. Robot. Autom. Sci. (ICRAS), Chengdu, China*, Jun. 2020, pp. 55–61. DOI: 10.1109/ICRAS49812.2020.9134922.

[5] S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs*", in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), San Francisco, CA, USA,* Sep. 2011, pp. 807–814.

[6] B. Ruf, S. Monka, M. Kollmann, and M. Grinberg, "Real-time on-board obstacle avoidance for UAVs based on embedded stereo vision", *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.,* vol. XLII-1, pp. 363–370, 2018.

[7] M. Ferrick, C. Fang, and B. Boots, "Deep stereo obstacle avoidance with monocular depth estimation for UAVs", **IEEE Robot. Autom. Lett.**, vol. 6, no. 2, pp. 2922–2929, Apr. 2021. DOI: 10.1109/LRA.2021.3062833.

[8] J. Vautherin, "PX4-Gazebo Simulator (Headless)", *GitHub Repository,* 2023. [Online]. Available: https://github.com/JonasVautherin/px4-gazebo-headless [Accessed: Dec. 31, 2024].

[9] S. Tijmons, G. de Croon, B. Remes, C. De Wagter, and M. Mulder, "Obstacle avoidance strategy using onboard stereo vision on a flapping wing MAV", *arXiv preprint arXiv:1604.00833*, 2016. [Online]. Available: https://arxiv.org/abs/1604.00833. [Accessed: Jan. 19, 2025].

[10] S. Tijmons, G. de Croon, B. Remes, C. De Wagter, and M. Mulder, "Stereo vision based obstacle avoidance on flapping wing MAVs", *in Advances in Aerospace Guidance, Navigation and Control, Springer*, 2013, pp. 463–482. DOI: 10.1007/978-3-642-38253-6_25.

[11] M. Ferrara, G. Gennarelli, and M. Montanari, "Fast obstacle detection system for UAS based on complementary use of radar and stereoscopic camera", *Drones,* vol. 6, no. 11, p. 361, 2022. DOI: 10.3390/drones6110361.

[12] S. Vanneste, B. Bellekens, and M. Weyn, "3DVFH+: Real-time three-dimensional obstacle avoidance using an Octomap", *in Proc. 9th Int. Conf. Intell. Environ.*, 2013, pp. 91–102.

[13] S. Waharte, N. Trigoni, and S. Julier, "Coordinated search with a swarm of UAVs", *in Proc. IEEE Int. Conf. Mobile Adhoc Sens. Syst. (MASS), Macau, China*, Oct. 2009, pp. 1–6. DOI: 10.1109/MOBHOC.2009.5337011.

[14] QGroundControl Dev Team, "QGroundControl guide (daily build 5.0)", *QGroundControl Documentation*. [Online]. Available: https://docs.qgroundcontrol.com/master/en/qgc-user-guide/index.html [Accessed: Jan. 19, 2025].

[15] PX4 Development Team, "Middleware", *PX4 Documentation*. [Online]. Available: https://docs.px4.io/main/en/middleware/ [Accessed: Jan. 19, 2025].

[16] Open Source Robotics Foundation (OSRF), "Gazebo: Tutorials", GazeboSim.org. [Online]. Available: https://classic.gazebosim.org/tutorials [Accessed: Jan. 19, 2025].

[17] ROS Community, "stereo_image_proc - ROS Wiki", *Ros.org.* [Online]. Available: https://wiki.ros.org/stereo_image_proc. [Accessed: April 08, 2025].

[18] OpenCV Dev Team, "Camera Calibration and 3D Reconstruction — OpenCV 2.4.13.7 documentation", *Opencv.org*, 2019. https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#stereobm (Accessed April 12, 2025).

[19] OpenCV Dev Team, "OpenCV: Depth Map from Stereo Images", *docs.opencv.org*. https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html. [Accessed: April 12, 2025].

[20] ROS Community, "pcl - ROS Wiki", *Ros.org*, 2024. https://wiki.ros.org/pcl (Accessed April 13, 2025).

[21] T. Baumann, "*Obstacle Avoidance for Drones Using a 3DVFH* Algorithm*", M.S. thesis, Dept. of Computer Science, ETH Zurich, Zurich, Switzerland, 2018.