Journal of Robotics, Networking and Artificial Life
Vol. 11(3); March (2026), pp. 204-209
ISSN (Online): 2352-6386; ISSN (Print): 2405-9021
https://alife-robotics.org/jrnal.html

Research Article

Development of a Drone Navigation System Using Depth Estimation for Obstacle Avoidance

Sora Takahashi[1], Eiji Hayashi[1]
[1]*Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi, Fukuoka, 820-8502, Japan*
Email: takahashi.sora614@mail.kyutech.jp, haya@mse.kyutech.ac.jp

ARTICLE INFO

Article History

Received 29 November 2024
Accepted 19 December 2025

Keywords

Deep Learning,
Autonomous robot,
Estimation of metric Depth

ABSTRACT

This research introduces a drone obstacle avoidance system built upon depth estimation derived from monocular RGB images, with the objective of minimizing reliance on high-cost sensors such as LiDAR and RGB-D cameras. The proposed system incorporates ZoeDepth, a deep learning model for monocular depth prediction, and is implemented within a simulated environment using ROS and Gazebo. Two autonomous configurations were tested: one equipped with an RGB-D camera, and the other utilizing depth information inferred from RGB inputs. Evaluation results revealed that, although the RGB-D-equipped setup achieved higher accuracy, the RGB-based system was able to avoid obstacles and reach the target destination, despite exhibiting some localization errors. Future efforts will focus on increasing the system's resilience in densely populated environments and conducting validation through real-world trials.

1. Introduction

Over the last several years, the deployment of drones for surveying purposes has expanded in regions affected by natural disasters and in remote insular areas. However, in these contexts, direct human operation is frequently constrained by safety hazards, workforce limitations, and financial burdens. Although existing technological frameworks support remote execution of predefined aerial routes by drones, they are prone to collisions when unexpected obstacles appear along these routes. Furthermore, most drones are not equipped with sensors that can detect depth across varying distances, such as RGB-D cameras or LiDAR, which are important for reliable obstacle avoidance.

In this study, we implemented an autonomous navigation system with obstacle avoidance functionality that uses only an RGB camera, typically installed on general-purpose drones. We integrated computer vision with deep learning to allow our drone to identify and avoid obstacles based on images captured via an RGB camera. We tested the system in a simulation to evaluate its performance, and the results showed that it was able to move autonomously without colliding with obstacles.

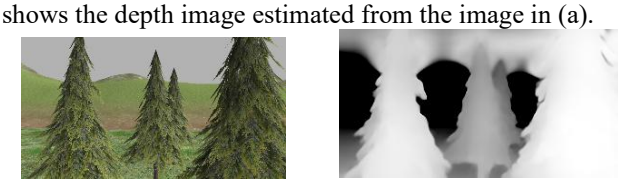
2. System Overview

In this study, two navigation systems were developed, each employing a different type of visual sensor. One system uses an RGB-D camera, while the other relies on an RGB camera. The latter serves to validate the proposed approach, where depth estimation from RGB input is used to extract distance information necessary for obstacle detection.

2.1. Depth Estimation Model

Depth information was estimated by ZoeDepth, a deep learning model that approximates depth from a single image [1]. This model specializes in monocular depth estimation and offers greater versatility and accuracy compared to conventional depth estimation models. ZoeDepth is a highly accurate depth estimation model that fine-tunes a network initially trained for relative depth estimation using metric depth datasets. In this process, RGB images are first processed by the MiDaS [2] depth estimation framework for relative depth estimation, and then combined with pixel-wise depth bin centers computed by a metric bin module to produce the final metric depth output. Through this sequence of operations, the ZoeDepth model integrates information across different depth levels and achieves accurate depth estimation even in complex scenes and under various lighting conditions. Fig1 shows

an example of a depth image estimated from an RGB image using ZoeDepth. (a) shows the RGB image and (b) shows the depth image estimated from the image in (a).



(a)RGB image (left) (b) Depth image (right)

Fig 1 Example of depth estimation

2.2. Compare with other estimation models

The depth estimation model for this system was selected based on two criteria: "estimated depth accuracy" and "generalization performance across diverse environments." Relative depth estimation models like MiDaS were deemed unsuitable because they cannot estimate the "absolute physical scale" essential for robot navigation. Furthermore, conventional metric depth estimation models such as AdaBins and BTS faced challenges in estimation accuracy when encountering unknown environments. ZoeDepth bridges this gap by combining a relative depth prior distribution with metric fine-tuning, achieving the robust zero-shot metric estimation required for this system. Although newer models emerged later, at the time of development, ZoeDepth demonstrated state-of-the-art performance in zero-shot generalization on the NYU Depth V2 dataset and the KITTI dataset (table1 and table2).[3] [4] Furthermore, the model used in this system is interchangeable with subsequent superior models.

table 1 Comparison of Depth Estimation Accuracy in the NYU Depth V2 Benchmark

Model	Type	Abs Rel	δ1
BTS	Metric	0.110	0.885
AdaBins	Metric	0.103	0.903
Zoedepth	Metric	0.075	0.955

table 2 Comparison of Depth Estimation Accuracy in the KITTI

Model	Type	Abs Rel	δ1
BTS	Metric	0.059	0.956
AdaBins	Metric	0.058	0.964
Zoedepth	Metric	0.057	0.970

2.3. Simulation Environment

The simulation environment was designed based on the assumption that the DJI Air 2S would be used as the physical platform. To construct this environment, Gazebo, ROS, and ArduPilot were employed. Gazebo offers a real-time 3D simulation platform that incorporates a physics engine to replicate environmental dynamics and drone behavior. Meanwhile, ROS facilitates communication and data exchange among system components, contributing to modularity and scalability. The simulation system setup is shown in Fig2. Fig3 provides a view of the drone and its environment as visualized in Gazebo.

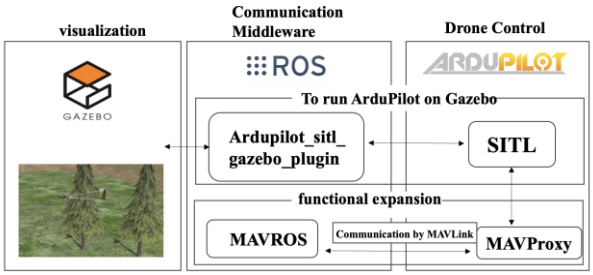
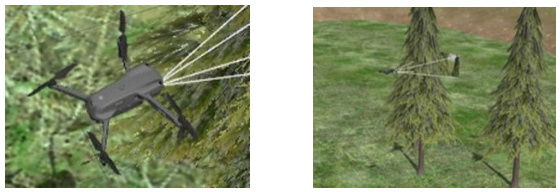


Fig 2 System configuration for the simulation



(a) Drone (left) (b) Environment (right)

Fig 3 Simulation Environment

2.4. Autonomous Navigation Systems

For autonomous navigation and obstacle avoidance, we adopted the ROS Navigation Stack, which offers the essential components for robot movement, such as path planning and localization. With this setup, the robot was able to move through its environment without manual intervention. In our system, RGB-D images were fed into RTAB-Map for mapping and localization. [5]. We implemented autonomous navigation and obstacle avoidance using the ROS Navigation Stack, which provides key software modules for robot control without human intervention. In this system, the robot navigates its surroundings independently. RGB-D image data are processed by RTAB-Map for mapping and localization. An overview of the system's operation is shown in Fig4.

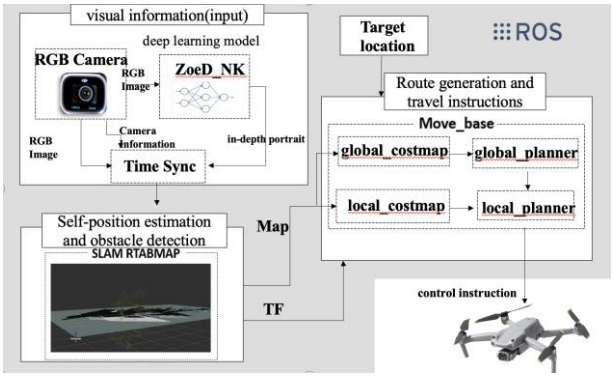


Fig 4 System overview of drone system

2.5. *Algorithm used for path planning*

In our system, we adopted a hybrid approach to path planning that incorporates both global and local strategies. For the global planner, Dijkstra's algorithm is applied to compute a path from the starting point to the destination based on a predefined static map. Since it expands nodes in the order of increasing total cost, it is well suited for generating a reference trajectory in environments where the map does not change. For local path planning, the Dynamic Window Approach (DWA) is applied to generate feasible and dynamically consistent velocity commands in real time. The DWA evaluates candidate trajectories within the robot's velocity space, considering dynamic constraints, obstacle avoidance, and the goal direction. This allows the robot to reactively navigate through dynamic and partially unknown environments while following the global path generated by Dijkstra's algorithm.

By combining Dijkstra's algorithm for global planning with the DWA for local planning, the system achieves both optimal path generation and robust real-time obstacle avoidance in actual navigation.

2.6. *Processing Pipeline*

This system consists of four main modules: perception using a monocular camera, SLAM, path planning, and flight control. The data flow between each process is as follows. It is implemented by looping through the following steps 1 through 6.

1. Acquire RGB images from a virtual camera in Gazebo.
2. The depth estimation model subscribes to images and infers metric depth using ZoeDepth. The results are published as a depth map.
3. Synchronize the timestamps of asynchronously arriving RGB images and depth maps to generate RGB-D pairs.
4. RTAB-Map updates the environmental map and performs self-localization based on synchronized data.
5. The Navigation Stack performs global planning (Dijkstra) and local planning (DWA) to generate the velocity command.
6. MAVROS converts the speed command to MAVLink, and ArduPilot controls the drone's thrust.

2.7. *Hardware Configuration and Real time performance*

To verify the system's real-time performance and reproducibility, processing time and update frequency were measured under the following hardware environment (table3).

table 3 Hardware Configuration

CPU	AMD Ryzen 7 5700X
GPU	NVIDIA GeForce RTX 3090
Memory	16GB
OS	Ubuntu 20.04 LTS (64-bit)

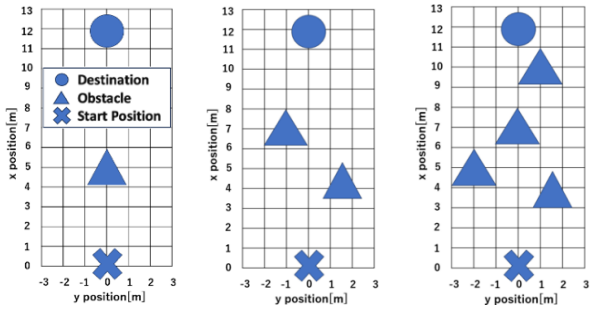
Measurements for each module were taken using `rostopic hz`, while ZoeDepth's inference time was measured precisely in milliseconds using PyTorch's CUDA event timer. Table4 shows the result.

table 4 Real time performance

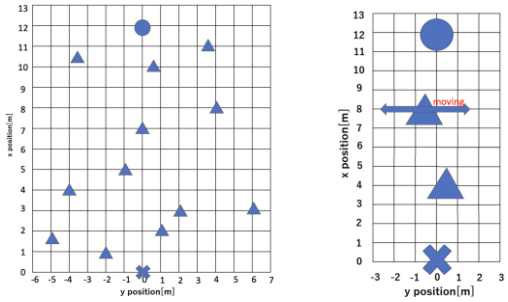
Module	Metric	Measured Value
ZoeDepth	Inference Time	~800 ms
	Throughput	~1.25 Hz
RTAB-Map	Update Rate	~1.25 Hz
Nav Stack	Command Rate	~3.1 Hz

3. Experiment

To assess their ability to navigate around obstacles and arrive at the intended target, the two autonomous obstacle avoidance systems were tested within a simulated environment. In the setup illustrated in Fig5 and Fig6, we defined the starting point of the robot as (0, 0, 3) m, treating it as the origin, with the goal position specified as (12, 0, 3). Regarding Scenario 5, one of the two trees is moving back and forth horizontally, acting as a dynamic obstacle. In each of the three obstacle scenarios, the drone moved autonomously three times toward its destination in two patterns. In these experiments, the allowable positional error was set to 1 m. The experiments evaluated whether the robot can avoid the obstacles in both patterns and measured arrival error. Also, we adopted the root mean squared error (RMSE) as the metric for error evaluation.

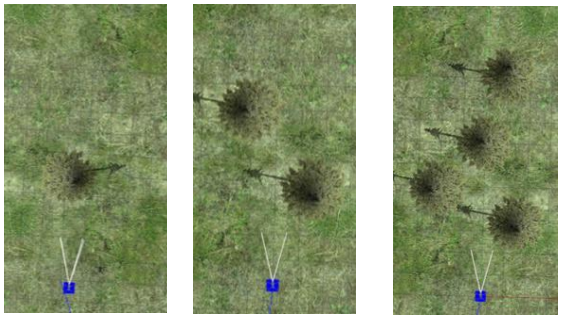


(a) Scenarios 1 (b) Scenarios 2 (c) Scenarios 3

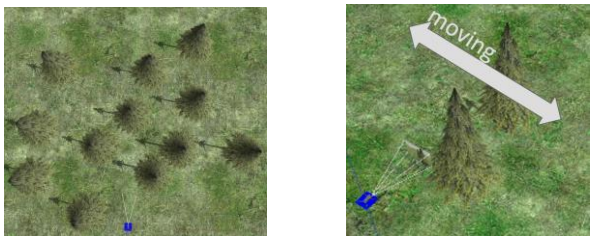


(d) Scenarios4 (e) Scenarios5

Fig 5 Obstacle scenarios



(a) Scenarios 1 (b) Scenarios 2 (c) Scenarios 3



(d) Scenarios4 (e) Scenarios5

Fig 6 Obstacle scenarios in simulation

4. Experimental Results

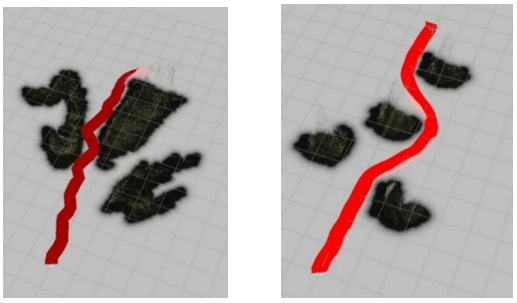
The results of the experiments for both navigation systems-one employing depth estimation from RGB input and the other using an RGB-D camera-are presented in table5 and table6, respectively. As shown in the tables, both systems avoided obstacles under all tested conditions. Across all tested obstacle configurations, the system equipped with the RGB-D camera consistently exhibited lower deviation from the target location. As detailed in table6, its maximum positional error was 0.5881 m, which remained within an acceptable range for every scenario. In contrast, the autonomous navigation system using depth estimation exhibited a minimum positional error of 2.3988 m, resulting in an error more than twice the allowable margin of 1 m. The autonomous system using depth estimation exhibited smaller errors with an increasing number of obstacles. In Scenario 4, which adds more obstacles than Scenario 3, the depth estimation-based system collided with an obstacle once. In Scenario 5, which featured dynamic obstacles, both systems collided with the obstacles and failed to reach the goal, making it impossible to measure other metrics such as arrival error or time to goal. Focusing on travel time, the results of both methods were generally comparable, falling within the range of approximately 60 to 80 seconds for all successful scenarios. Furthermore, as obstacle placement became more complex, a trend of increasing travel time was observed for both monocular depth estimation and RGB-D methods. These results demonstrate that even a system based on monocular depth estimation can reach the destination in a time comparable to that of an RGB-D system. Next, comparing path length, both methods showed a tendency for path length to increase as scenario difficulty rose. While systems using RGB-D cameras tended to generate slightly shorter paths compared to those using monocular depth estimation, the difference was less than 1 m, indicating no significant distinction. Fig7 illustrates the flight trajectory of the drone and its obstacle recognition results in the third scenario for both autonomous systems. It can be observed that the system based on depth estimation tended to interpret a broader region as a single obstacle, in contrast to the RGB-D camera system, which produced more localized recognition.

table 5 Navigation System Using Depth Estimation

Scenarios	Avoiding Obstacle	Destination Error(Average)[m]	Time [s]	Length of path [m]
Obstacle Scenarios 1	○	6.2334m	63.57	12.04
Obstacle Scenarios 2	○	3.9669 m	59.66	11.79
Obstacle Scenarios 3	○	2.2863 m	71.85	13.97
Obstacle Scenarios 4	2/3○	2.3988 m	77.91	14.01
Obstacle Scenarios 5	X	-	-	-

table 6 Navigation System Using RGB-D Camera

Scenarios	Avoiding Obstacle	Destination Error(Average)[m]	Time [s]	Length of path [m]
Obstacle Scenarios 1	○	0.4951 m	62.45	11.86
Obstacle Scenarios 2	○	0.5881 m	63.45	11.68
Obstacle Scenarios 3	○	0.3669 m	70.45	13.23
Obstacle Scenarios 4	○	0.5201 m	70.38	13.27
Obstacle Scenarios 5	X	-	-	-



(a) Drone trajectory of estimation system (left)

(b) Drone trajectory of RGB-D camera system(right)

Fig 7 Drone's trajectory and detected obstacle

5. Consideration

Compared to the RGB-D camera, the depth estimated by the system was less accurate. This likely caused inconsistencies in perceived obstacle distances, leading to broader regions being identified as obstacles, as illustrated in Fig7 (a). Furthermore, the drone estimated its position by relying on both visual data from the camera and depth measurements. During flight, small errors in depth readings appeared to accumulate, causing a noticeable offset between the intended goal and the actual point where the drone stopped. According to table5, the positional error of the depth-estimation-based system tended to decrease as the number of obstacles increased. The exploration path was generated based on features obtained from obstacles. In Obstacle Scenario 3, it is believed that the accuracy of self-localization improved because feature points contributing to self-localization could be acquired near the goal. Furthermore, as the number of obstacles increased, the number of edges, corners, and other feature points in the image also increased, which likely contributed to higher self-localization accuracy. Therefore, it can be inferred that environments with a greater number of obstacles that generate more feature points enable more accurate self-localization. To further improve position estimation accuracy, we believe that integrating additional sensors, such as GPS and IMU, rather than relying solely on visual and depth information, would be effective. Although the monocular depth estimation–based system generally produced slightly longer paths than the RGB-D–based system, the resulting travel times remained comparable across all successful scenarios. This indicates that the conservative obstacle interpretation caused by depth estimation uncertainty did not significantly degrade overall navigation efficiency. Instead, the system compensated by selecting safer, albeit marginally longer, trajectories without incurring substantial time penalties. The introduction of more complex obstacle configurations in Scenario 4 further highlights this behavior. As obstacle density increased, both path length and travel time tended to rise, reflecting the increased difficulty of the environment. While the depth estimation–based system failed to avoid a collision in this specific instance, it otherwise maintained robustness in moderately complex static environments. Moreover, the dynamic obstacle scenario revealed clear limitations of the proposed navigation framework. In Scenario 5, both the monocular depth estimation–based system and the RGB-D–based system failed to reach the goal due to collisions, highlighting the inherent difficulty of navigating dynamic environments. Since the RGB-D–based system also exhibited failures under the same conditions, these results indicate that the observed performance degradation cannot be attributed solely to the use of monocular depth estimation.

6. Conclusion

This study involved the construction of a simulation environment and the implementation of a drone obstacle avoidance system based on depth estimation. Experimental evaluation demonstrated that the system could successfully detect and avoid obstacles, although its localization accuracy was inferior to that of a system utilizing RGB-D cameras. Moving forward, we plan to further develop the autonomous navigation framework to enhance the self-localization performance of the depth-estimation-based obstacle avoidance system, as evidenced in our experimental results. Specifically, we aim to develop a system that incorporates self-localization using sensors such as an IMU and GPS, rather than relying solely on RGB-D odometry. In addition, we plan to evaluate the developed obstacle avoidance system in real-world environments with the goal of automating drone-based surveys.

1. References

1. Bhat, S. F., Birkl, R., Wofk, D., Wonka, P., & M ¨ uller, M., ZoeDepth: Zero-shot Transfer by Com-bining Relative and Metric Depth, arXiv preprint arXiv:2302.12288, 2023.
2. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., & Koltun, V., Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer , arXiv preprint arXiv:1907.01341v3, 2019.
3. Lee, J. H., Han, M.-K., Ko, D. W., & Suh, I. H., From big to small: Multi-scale local planar guidance for monocular depth estimation, arXiv preprint arXiv:1907.10326, 2019.
4. Bhat, S. F., Alhashim, I., & Wonka, P., AdaBins: Depth estimation using adaptive bins , Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4009–4018, 2021.
5. b´e, M., & Michaud, F. RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation. Journal ofField Robotics, 36(2), 416-446. 2019.

Authors Introduction

Mr. Sora Takahashi

He received his bachelor's degree in engineering in 2024 from mechanical system engineering, Kyushu Institute of Technology in Japan. He is currently a Master student at Kyushu Institute of Technology and conducts research at Hayashi Laboratory.

Prof. Eiji Hayashi

Prof. Eiji Hayashi is a professor in the Department of Intelligent and Control Systems at Kyushu Institute of Technology. He received the Ph.D. (Dr. Eng.) degree from Waseda University in 1996. His research interests include Intelligent mechanics, Mechanical systems and Perceptual information processing. He is a member of The Institute of Electrical and Electronics Engineers (IEEE) and The Japan Society of Mechanical Engineers (JSME).