Autonomous Drones in Artificial Pollination

Eric Lin *(el38),* Shaun Lin *(hl116),* and Karl Sheng *(ks156)*
Advisor: Dr. Joseph Young
Electrical and Computer Engineering, Rice University

Abstract

In this report, you'll see the result produced in the spring 2023 semester. As our team, AutoDrone, has equipped drones with computational capability by running a YOLOv5 detection model combined with a MiDaS depth estimation model on the embedded system Jetson Nano 2GB. Using DroneKit and pymavlink libraries, we are able to navigate the drone with Jetson Nano 2GB and/or Raspberry Pi, emulating the behavior of bees. Our tests have shown good precision and agility in visual detection, and we have conducted a successful test flight using a combination of Jetson Nano 2GB and/or Raspberry Pi with a drone. Our results demonstrate the feasibility of using autonomous drones for artificial pollination and resolving the shortage of pollinators.

Keywords: Autonomous Drone, Drone Navigation, Artificial Pollination, YOLOv5, Object Detection, MiDaS, Depth Estimation, Flower Detection, Jetson Series, Raspberry Pi.

I. Introduction

Bees are essential for pollinating crops and flowers, but their populations are declining rapidly. According to Bee Informed Partnership, an estimated 32.2% of managed colonies in the United State lost during the winter of 2020-2021 [1], and Food and Agriculture Organization of United Nations report that global losses of 10-15% each year, with some countries reporting losses of up to 30% [2]. To address this problem, there are researchers investigating small insect-like flying robots with remote control [3], which shows the possibility to make it autonomous.

Additionally, another previous robotic pollination researches [4], developed an precision robotic pollination system, they focused on localization and mapping, visual perception, path planning and motion control, the system is based on a ground-based robotic arm for pollination tasks, and the system is unable to handle high sensitivity and tackle complex terrain areas for pollination tasks as well as a real bee.

Therefore, this project aims to explore the use of a flying drone for pollination. Robotic pollinators, such as autonomous drones, have the potential to revolutionize the future of pollination and agriculture, using low-cost camera-based sensors and deep learning technology to develop an effective and sustainable solution with drones to address the problem of declining honeybee populations and their impact on food production.

II. Methods

A. Hardware

This section contains all hardware used in this capstone project.

1) Drone Kit:

- QWinOut Q705 6-Aixs RC Drone
- QWinOut J630 4-Aixs RC Drone

Fig. 1: (left) 6-Axis RC Drone / (right) 4-Axis RC Drone.

There are two types of drones used in the project, as shown in Fig. 1, the larger 6-axis RC drone and the new smaller 4-axis drone purchased this 2023 spring semester. The larger 6-axis drone is a continuation of the team's work from last semester, and the smaller 4-axis drone is a new drone that we assembled this semester.

2) Transistor & Receiver:

- SIK Telemetry Radio
- Flysky FS-i6 6CH 2.4GHz RC Transmitter
- Flysky FS-i6X 10CH 2.4GHz RC Transmitter
- iA6B Receiver
- iA10B Receiver

Fig. 2: (left) SIK Telemetry Radio / (middle) FS-i6 6CH Transmitter & iA6B Receiver / (right) FS-i6X 10CH Transmitter & iA10B Receiver.

Since we use two types of drones, there are different Transistors & Receivers for each, as shown in Fig. 2. FS-i6 6CH and iA6B Receiver pairing is for 6-axis drone. FS-i6X 10CH and iA10B Receiver pairing is for 4-axis drone. Then SIK radio just used to communicate between flight controller and Laptop/PC.

3) Flight Controller:

- Cube Black Flight Controller
- APM2.8 Flight Controller

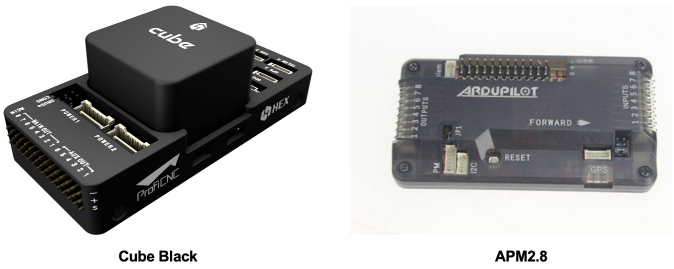


Fig. 3: (left) Cube Black / (right) APM2.8.

Flight controller plays one of the crucial roles in this project, as it is responsible for controlling the drone's flight and stability. It receives sensor data from various sensors and makes necessary adjustments to maintain the desired altitude, orientation, and movement of the drone. It's including an accelerometer, gyroscope, magnetometer, and barometer.

We used two flight controller modules that came with the Drone Kit in this project, as shown in Fig. 3, so that we could expedite our development and spend more time on Deep Learning algorithms.

4) Embedded System / Controller:

- Jetson Nano 2GB
- Raspberry Pi 3B
- Raspberry Pi 4B

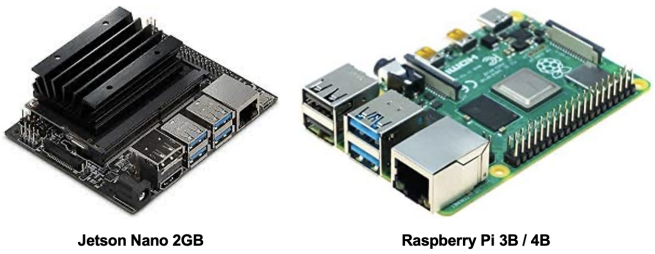


Fig. 4: (left) Jetson Nano 2GB / (right) Raspberry Pi 3B / 4B.

The embedded system is the main controller in the system, which plays a critical role in performing all the tasks. It collects images from the camera sensor and uses deep learning algorithms to correctly identify flower types, track flower locations and calculate depth information, and then sends commands to the flight controller to precisely control the drone to successfully complete the artificial pollination task.

Here we use two types of embedded devices, as shown in Fig. 4, the Jetson Nano 2GB with built-in GPU and the smaller Raspberry Pi 3B / 4B. We will test these two types of embedded devices and compare which one is more compatible for the project.

B. System Design

The system consists of essential components, including a Jetson Nano 2GB and/or Raspberry Pi to process deep learning models. The Drone-Kit API and MAVLink protocol facilitate communication with the Cube Black flight controller, which ultimately controls the drone's motor and navigation. The system flow chart is depicted in Fig. 5.

Two embedded devices, the Jetson Nano and Raspberry Pi, are incorporated into the drone to compare their performance. Additionally, a SIK radio block is utilized to send the drone's status back to a Laptop/PC for real-time monitoring. For added safety, a Flysky FS-i6 6CH 2.4GHz RC Transmitter and iA6B Receiver are included to enable manual drone control.

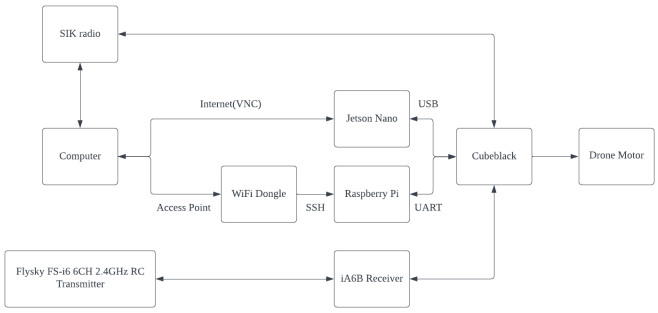


Fig. 5: System Flow Chart.

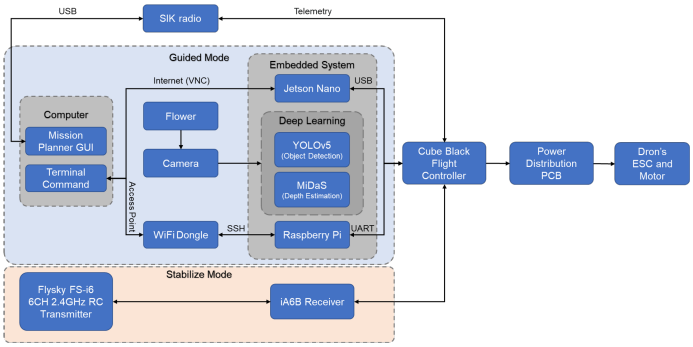


Fig. 6: Overall System Design.

Fig. 6 provides a detailed overview of the system design. The process starts with capturing an image of a flower using a camera sensor. The image is then analyzed using Deep Learning's YOLOv5 object detection model and MiDaS depth estimation model, both of which run on the Embedded System Jetson Nano 2GB and/or Raspberry Pi. YOLOv5 is used to detect flowers in the image (Please note sections II-C below for more information on YOLOv5), while MiDaS calculates the distance between the flowers and the drone. (Please note sections II-D below for more information on MiDaS)

For manual control, we use the Flysky FS-i6 6CH 2.4GHz RC Transmitter in Stabilize Mode, which disables pre-arming checking and puts the drone in an unstable mode. However, to operate the drone using scripts, it must be set to Guided Mode, which allows pre-arming checks to ensure the drone is in a safe condition to complete the pollination task.

C. YOLOv5

Object detection is a crucial aspect of autonomous drone operation. While there are many deep learning models that can be used for flower detection, Convolutional Neural Networks (CNNs) are the most researched method [5], [6]. However, these models mostly use a two-stage pipeline which may not be efficient for autonomous drone purposes. In contrast, one-stage models such as YOLO (You Only Look Once) [7] model is a state-of-the-art real-time object detection algorithm, known for its high speed and accuracy. YOLO processes the input image by dividing it into a grid and predicting a set of bounding boxes, objectness scores, and class probabilities for each grid cell. These predictions are combined and subjected to non-maximum suppression to remove overlapping boxes, generating the final object detections.

In our autonomous drone pollination project, the primary objective is to detect flowers using a webcam mounted on the drone. The YOLO model plays a crucial role in identifying flowers within the camera frame. We utilized the YOLOv5 architecture [8] for this project, which is an enhanced version of the original YOLO model. YOLOv5 is implemented in the PyTorch framework and offers a range of pre-trained models, each with different trade-offs between speed and accuracy. Given the limited computational resources available on drones, we chose the smaller YOLOv5 model, YOLOv5s, which has only 7.2 million parameters.

In addition to the model trained by the Ultralytics team, we also investigated the methods employed by the previous Autodrone team. They incorporated knowledge distillation [9] into the original YOLOv5 training framework, a technique that transfers knowledge from a larger model to a smaller model, thereby enhancing the performance of the smaller model. A larger model served as a teacher model, with its output functioning as a soft label to instruct the student model.

By leveraging the efficient YOLOv5s architecture and incorporating knowledge distillation, our flower detection model achieves high accuracy while maintaining real-time performance, making it suitable for deployment on resource-constrained platforms such as drones. This combination of techniques enables the autonomous drone pollination system to effectively detect flowers or other objects in real-world environments, thus constructing a key foundation for the autonomous pollination system.

D. MiDaS

To accurately estimate the distance of flowers from the drone, we employed the MiDaS (Monocular Depth Estimation of High Resolution Images with Pyramid Stereo Fusion) model [10]. The MiDaS model is a state-of-the-art depth estimation algorithm known for its speed and accuracy. It can estimate depth from a single image without the need for stereo cameras. The model utilizes a deep neural network to learn a mapping between the input image and its corresponding depth map, which is subsequently used to calculate the distance of the flower from the drone.

While there are alternative methods to determine the distance between objects and the camera with higher accuracy, such as stereo vision, these approaches tend to consume more computing resources and power. Given the constraints of the embedded system on the drone, we opted to use the MiDaS depth estimation model based on a monocular camera. Theoretically speaking, this model allows us to estimate the depth of the flower using a single image captured by the drone, reducing computational requirements and power consumption while still providing reliable depth estimation.

The MiDaS depth map estimates the depth of each pixel in an input image, representing the distance of that pixel from the camera. However, several factors can affect the accuracy of the depth map, including the quality of the input image, the distance of the object from the camera, and the lighting conditions. To evaluate the performance of the MiDaS model and assess its suitability for depth estimation on a drone, we conducted two experiments aimed at determining the numerical relationship between the depth map intensity and the real-world distance in meters.

The first experiment was set up inside the ECE lab, as illustrated in Fig. 7. Multiple tapes were placed on the floor at 1-meter intervals. The camera was positioned steadily at the 0-meter line. In this controlled environment, chairs were placed on the marked lines and served as detected objects.

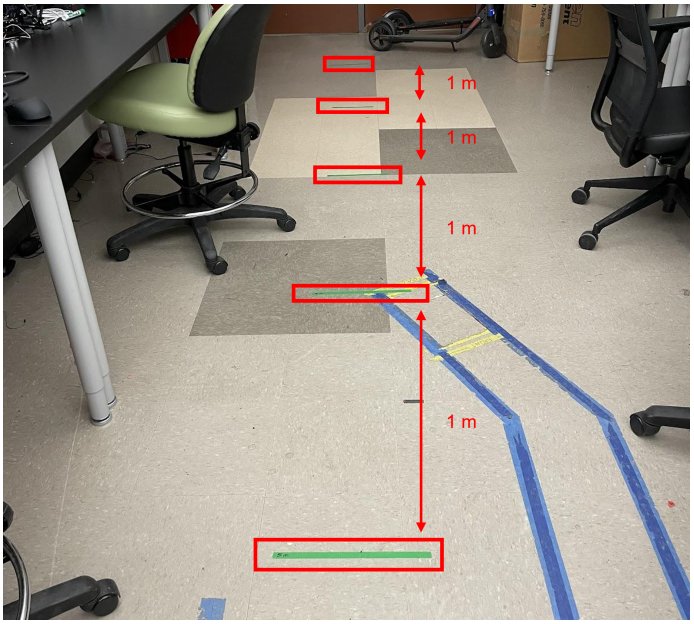


Fig. 7: Indoor Environment of the First Experiment

The second experiment was conducted outdoors in an open environment, as shown in Fig. **??**. The objective was to simulate object detection and depth estimation in a real autonomous pollination task with noise and unexpected anomalies occurring in the camera frame within a dynamic environment. The experiment involved a webcam mounted onto the drone, a team member carrying the drone and holding one end of a tape measure, and another team member holding the other end of the tape measure, acting as the detected object, and recording the actual distances from the camera. This experiment was performed twice from two different orientations to test whether the orientation of sunlight was another contributing factor to the depth map intensity.

Fig. 8: Outdoor Environments of the Second Experiment

The results and interpretation of these experiments will be elaborated in detail in the following Results subsection.

E. UI

To monitor the drone's status in real-time, we utilize the SIK Telemetry Radio and Mission Planner GUI. This eliminates the need to develop a complex GUI interface and provides detailed drone status information.

III. Results

A. Raspberry Pi Integration

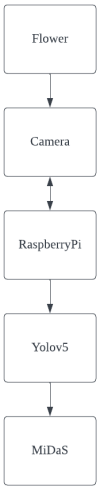


Fig. 9: Object Detection Flow Chart.

Fig. 9 shows the flowchart of our object detection with the Raspberry Pi embedded system. As you can see the sequence in the flowchart is first using YOLOv5 to detect Flower and then using MiDaS to measure the distance.

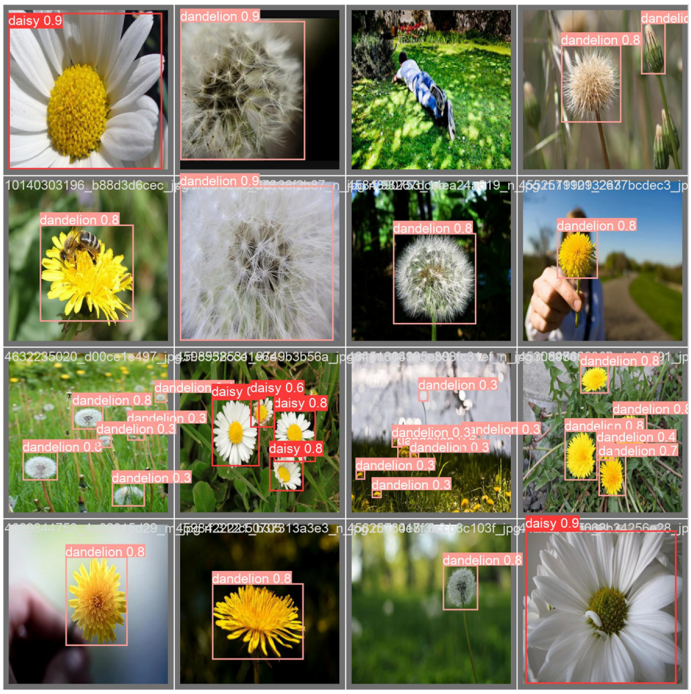


Fig. 10: YOLOv5 model test on real-world dataset with Raspberry Pi 3B.

Fig. 10 shows the results of real-world dataset testing on the Raspberry Pi 3B using the YOLOv5 model. It can be found that the accuracy is considerable, and one of the test images with no flowers on purpose is not misidentified.

As we run Yolov5 on Raspberry Pi 3B, Raspberry Pi 4B, we can clearly see the performance differences among them, as shown in TABLE I.

TABLE I: Performance Comparison on running Yolov5

Rubric	Controller	
	RPi3	*RPi4*
CPU (GHz)	1.2	1.5
RAM (GB)	1	8
RAM Usage (MB)	340	332
yolov5s.pt (fps)	0.21	0.95
yolov5n.pt (fps)	0.43	1.67
Load Speed (from cmd to start detect)	Very slow	Very fast

We are also able to control the drone to take off, hover, move in direction, rotation, and landing through Raspberry Pi.

B. Jetson Nano 2GB Integration

During the integration of the Jetson Nano 2GB and the drone system, we encountered unexpected issues and device malfunctions, which prevented us from incorporating the YOLOv5 detection model and MiDaS depth estimation model into the system. Instead, we utilized the OpenCV library and developed a QR code tracking script that enables the drone to detect QR codes and rotate to track them.

Fig. 11: Hover drone in the air operated by Jetson Nano 2GB.

Before implementing this experiment, we tested several other simple movement scripts. Fig. 11 shows the drone hovering successfully in the air for the first time, while Fig. 12 shows the final experiment in which the drone hovers in the air and tracks the QR code. This experiment was conducted in a real-world environment.

Fig. 12: QR code tracking results on Jetson Nano 2GB.

C. YOLOv5 + MiDaS

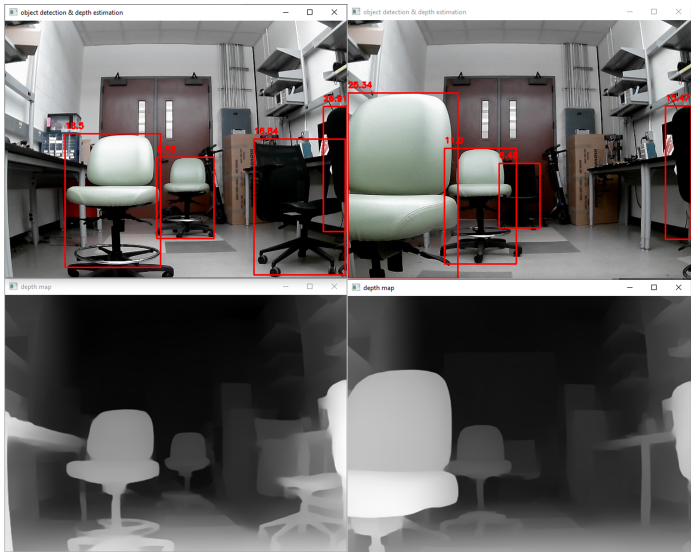


Fig. 13: Results of the First Experiment. (top left) YOLOv5 Objects Detected in 2m & 4m / (top right) YOLOv5 Objects Detected in 1m & 3m & 5m / (bottom) Corresponding MiDaS Depth Maps for Visualization

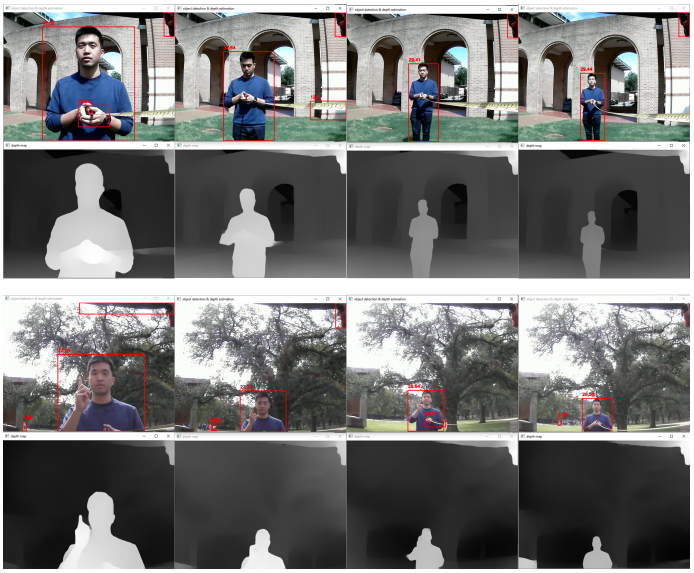


Fig. 14: Results of the Second Experiment. / (top rows) YOLOv5 Objects Detected from the First Orientation in 1m, 2m, 3m, and 3.5 m from Left to Right and Corresponding MiDaS Depth Maps for Visualization / (bottom rows) YOLOv5 Objects Detected from the Second Orientation in 1m, 2m, 3m, and 3.5 m from Left to Right and Corresponding MiDaS Depth Maps for Visualization

Fig. 13 and 14 display the results of the experiments discussed above. The bounding boxes indicate the objects detected by the YOLOv5 model. The MiDaS model is executed simultaneously, with its depth map displayed at the bottom of each corresponding camera frame. The median intensity of the detected object in the depth map is listed in the top left corner of each bounding box. The depth map intensities and real-world distances of the detected objects in each experiment are listed in Tables II and III, and their relationships are illustrated in Scatter Plots 1 and 2 below.

TABLE II: Data from Experiment 1

Real-World Distance (m)	Depth Map Intensity
1	25.34
2	18.50
3	11.00
4	8.68
5	6.48

Plot 1: Experiment 1
Real-World Distance vs. Depth Map Intensity

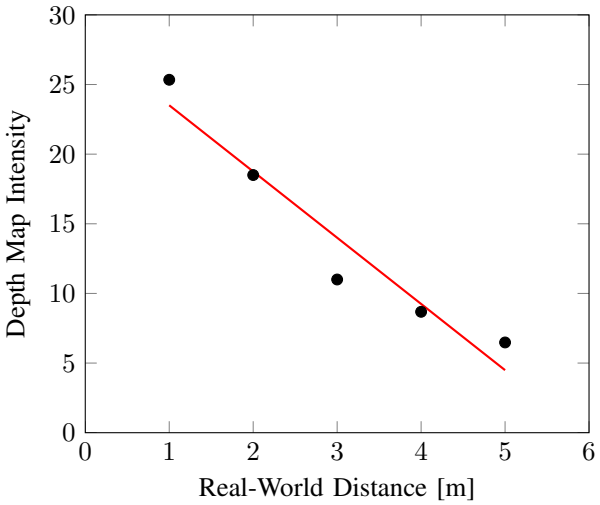
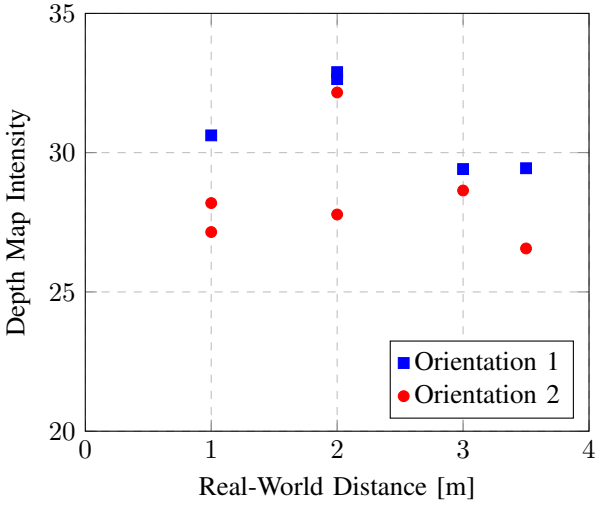


TABLE III: Data from Experiment 2

	Real-World Distance (m)	Depth Map Intensity	
		Trial 1	Trial 2
Orien 1	1	30.62	/
Orien 2		28.19	27.15
Orien 1	2	32.89	32.64
Orien 2		32.16	27.78
Orien 1	3	29.41	/
Orien 2		28.64	/
Orien 1	3.5	29.44	/
Orien 2		26.56	/

Plot 2: Experiment 2
Real-World Distance vs. Depth Map Intensity



In the indoor environment, due to the stable webcam setup and consistent lighting conditions, the scatter plot reveals an approximately linear relationship between the depth map intensity and real-world distance. By determining the slope

and intercept of this approximate straight line, we successfully convert a depth map intensity into an approximate real-world distance in the program. The error of this approximation is estimated to be around 10%.

In the outdoor environment, two observations can be made from the data. Firstly, the object with higher brightness in the first orientation exhibits a higher depth map intensity than the one in the second orientation at the same real-world distance. This discrepancy occurs due to the webcam's auto-exposure adjustment. The orientation of sunlight does affect the depth map intensity, even when objects are positioned at an equal distance from the webcam. Secondly, the depth map intensity varies significantly due to the vibration of the webcam. As the webcam has a fixed focal length and image distance, along with constant vibration on the drone, objects may inevitably appear blurry when they are further away from the camera. From Scatter Plot 2, it is difficult to discern the trend of depth map intensity versus real-world distance. This issue may significantly impact the depth map intensity produced by the MiDaS model, leading to substantial inaccuracies in measurement.

D. UI Interface

From the system design diagram in Fig. 6, it can be seen that we connect the Flight Controller to the SIK radio and then to the Mission Planner GUI of the computer, we follow this design to setup on the real drone system, the SIK Telemetry Radio devices on the left side of Fig. 2, connected one devide to the Cube Black Flight Controller's Telem Port 1, and the other one is connected to the Laptop's USB port, after initializing the settings in Mission Planner, you can successfully see the monitor screen as shown in Fig. 13.

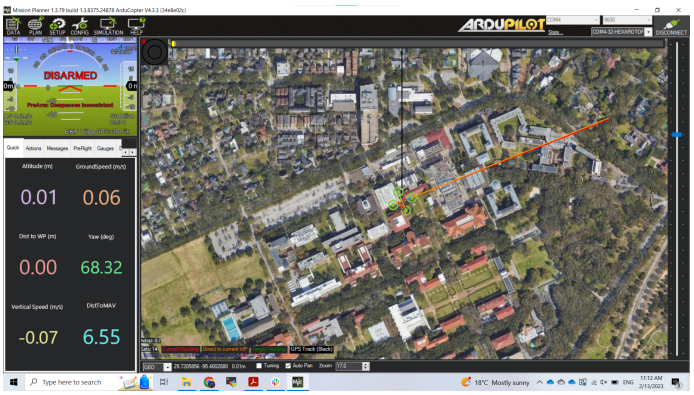


Fig. 15: Mission Planner GUI Interface.

Fig. 14 shows the real-time status of the drone from SIK Telemetry Radio in Mission Planner GUI. The left side of the image shows the "DISARMED" status, which means the drone is not armed yet, and the right picture shows the "ARMED" status, which means the drone is armed and starts to spin the propellers at a fixed speed.

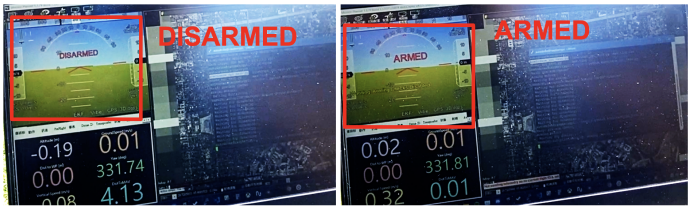


Fig. 16: Drone's real-time status on Mission Planner GUI.

IV. Discussion

One of the significant accomplishments in this semester is the successful integration of the QR code tracking script and drone movement script on the Jetson Nano 2GB. The QR code tracking script allowed the drone to detect QR codes and rotate to track them. The drone could also turn right until the QR code was in the center, showing the potential for autonomous navigation using computer vision techniques.

The results from the first MiDaS evaluation experiment reveal that it is reasonably accurate to perform object detection and depth estimation using YOLOv5 and MiDaS models in an indoor environment. However, the outdoor experiment simulating a practical object depth estimation task yields unfavorable results. The depth map intensity is not as accurate as expected and is significantly affected by various factors, such as sunlight orientation, webcam exposure, and drone vibration. Although the MiDaS model is reliable in a stable environment with minimal interference, its application in autonomous drone pollination tasks remains challenging.

These results demonstrate the progress made in the project and pave the way for future developments. In the next semester, we will further research solutions for autonomous drone navigation from the current position to the target position. We will consider either developing a new algorithm utilizing a stereo camera with higher accuracy or exploring alternative methods, such as target and self-localization to determine the relative position first, followed by motion planning. This approach will hopefully enable us to improve the reliability of the autonomous drone navigation system in real-world scenarios.

Acknowledgment

Repository for this project are available at: https://github.com/Rice-MECE-Capstone-Projects/Autodrone

Documents and tutorial for this project are available at: https://riceautodrone.github.io/

References

[1] "United States Honey Bee Colony Losses 2020-2021: Preliminary Results," Bee Informed, Jun. 21, 2021, https://beeinformed.org/2021/06/21/united-states-honey-bee-colony-losses-2020-2021-preliminary-results.

[2] "FAO - News Article: Declining bee populations pose threat to global food security and nutrition," Food and Agriculture Organization of the United Nations, May 20, 2019, https://www.fao.org/news/story/en/item/1194910/icode.

[3] K. Ma, P. Chirarattananon, S. Fuller, R. Wood, “Controlled Flight of a Biologically Inspired, Insect-Scale Robot,” Science Magazine, vol.340, no. 6132, pp. 603-607, May 2013.

[4] N. Ohi et al., “Design of an Autonomous Precision Pollination Robot,” 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 7711-7718, doi: 10.1109/IROS.2018.8594444.

[5] M. T. Pratama et al., “Deep Learning-based Object Detection for Crop Monitoring in Soybean Fields,” 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-7, doi: 10.1109/IJCNN48605.2020.9207400.

[6] Z. Sun, X. Guo, Y. Xu, S. Zhang, X. Cheng, Q. Hu, W. Wang, and X. Xue, “Image recognition of male oilseed rape (brassica napus) plants based on convolutional neural network for UAAS navigation applications on supplementary pollination and aerial spraying,” Agriculture, vol. 12, no. 1, p. 62, 2022, doi: 10.3390/agriculture12010062.

[7] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), May 2016, doi: 10.1109/cvpr.2016.91.

[8] G. Jocher et al., ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference (v6.1). Zenodo, 2022, doi: 10.5281/zenodo.6222936.

[9] ”Knowledge Distillation.” Wikipedia, Wikimedia Foundation, 23 Apr. 2023, en.wikipedia.org/wiki/Knowledge_distillation. Accessed 24 Apr. 2023.

[10] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, vol. 44, no. 3, pp. 1623–1637, doi: https://doi.org/10.48550/arXiv.1907.01341