Autonomous Drones for Person Re-Identification

Eric Lin *(el38),* Karl Sheng *(ks156),* Ze-Ning Li *(zl156),* and Yi-Han Hong *(yh110)*
Advisor: Dr. Joseph Young
Electrical and Computer Engineering, Rice University

Abstract

In this report, you'll see the result produced in the Fall 2023 semester. As our team, Autonomous Drone (AutoDrone), has equipped drones with computational capability by running a YOLOv8 detection model on the embedded system Jetson Orin Nano. Using Robot Operating System (ROS), Video4Linux (V4L), and MAVLink libraries, we are able to navigate the drone with Jetson Orin Nano and/or Raspberry Pi. Our tests have shown good precision and agility in visual detection (LiDAR and camera), Ardupilot Software-in-the-Loop (SITL), motion control and we have conducted a successful test flight using a combination of Jetson Orin Nano and/or Raspberry Pi with a drone. Our results demonstrate the feasibility of using autonomous drones for identification purposes.

Keywords: Autonomous Drone, Drone Navigation, Identification, YOLOv8, Object Detection, Depth Estimation, Jetson Series, Raspberry Pi, ROS, MAVLink, V4L.

I. Introduction

Recent advancements in person Re-ID are classified into four main categories based on metric and representation learning: deep metric learning, local feature learning, generative adversarial learning, and sequence feature learning. Each of these categories is further subdivided based on methodologies and motivations, which helps in understanding their unique advantages and limitations. These classifications signify the depth and breadth of current research in person Re-ID, highlighting how each approach contributes to overcoming the challenges posed by varied camera views, poses, illumination, and resolution [1].

The integration of autonomous drones in person Re-ID presents a novel approach to addressing the limitations of current methods, especially in complex environments like crowded spaces with non-uniform lighting. The project's future focus involves refining drone capabilities and AI algorithms to enhance accuracy and adaptability. This direction aligns with the ongoing research trends in person Re-ID, where the emphasis is on developing more robust, efficient, and adaptable systems for real-world applications.

Central to this project is the integration of the digital twin concept, a virtual replica of the physical drone. The use of Software in the Loop (SITL) simulation is instrumental in this project, which ensures robust autonomous decision-making. ROS, a comprehensive middleware framework for robot software development, facilitates efficient integration of computational resources and algorithms. The combination of these advanced technologies, SITL simulation and ROS, enhances the drone's performance and flexibility in dynamic environments.

II. Methods

A. Hardware

This section contains all hardware used in this capstone project.

1) Drone Kit:

- QWinOut Q705 6-Aixs RC Drone (Archive)
- 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 in the 2023 spring semester. The larger 6-axis drone is a continuation of the team's work from the previous semester, and the smaller 4-axis drone is a new drone that we assembled in the 2023 spring 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 (Archive)

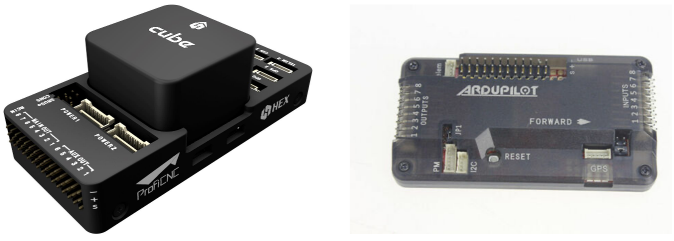

Fig. 3: (left) Cube Black / (right) APM2.8.

In the Spring 2023 semester of our project, we employed two flight controller modules, vital for governing the drone's flight and stability. These controllers, processing data from various sensors like accelerometers, gyroscopes, magnetometers, and barometers, ensure optimal altitude, orientation, and movement. Initially, as illustrated in Fig. 3, both the APM2.8 and Cube Black modules were used. However, challenges with calibration and parameter input on the APM2.8 led to its exclusion from the project. Consequently, our focus shifted entirely to utilizing the Cube Black for its superior performance and reliability in flight control operations.

4) Embedded System / Controller:

- Jetson Orin Nano
- Raspberry Pi 3B
- Raspberry Pi 4B

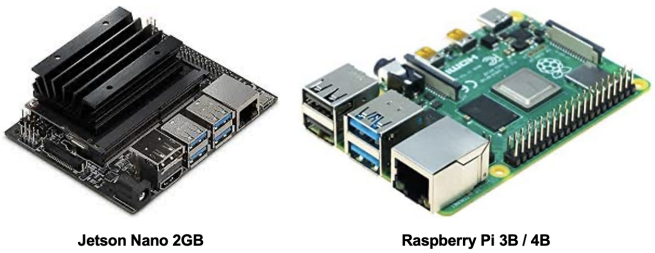

Fig. 4: (left) Jetson Orin Nano / (right) Raspberry Pi 3B / 4B.

In the AutoDrone project, our main controller is the embedded system, essential for executing tasks. It captures images from the camera and LiDAR, applying deep learning algorithms for accurate person re-identification and tracking. These algorithms process depth information, guiding the drone's flight controller for precise maneuvering in complex environments.

Our project utilizes two embedded devices: the GPU-equipped Jetson Orin Nano and the smaller Raspberry Pi 3B / 4B, as shown in Fig. 4. We are testing both to assess their compatibility and effectiveness for the project, ensuring optimal performance in real-world scenarios.

Fig. 5: LiDAR.

5) LiDAR: This LiDAR module eas used to detected 2D range around the drone Fig. 5.

B. System Design

The system consists of essential components, including a Jetson Orin Nano 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. 6.

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.

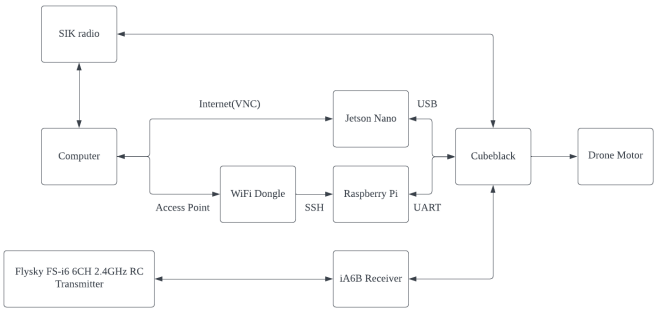

Fig. 6: System Flow Chart.

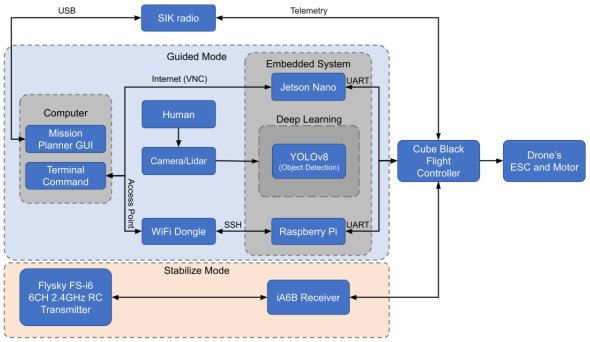

Fig. 7: Overall System Design.

Fig. 7 provides a detailed overview of the system design. The process starts with capturing an image of people using a camera. The image is then analyzed using Deep Learning's YOLOv8 object detection model, which runs on the Embedded System Jetson Orin Nano and/or Raspberry Pi. YOLOv8 is used to detect human in the image (Please note sections II-C below for more information on YOLOv8), while LiDAR scans for the distance between the human and the drone. (Please note sections II-D below for more information on LiDAR)

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. YOLOv8

Object detection is a crucial aspect of autonomous drone operation. While there are many deep learning models that can be used for object detection, Convolutional Neural Networks (CNNs) are the most researched method. 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) [2] 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 detection. During this semester, our team underwent a significant transition from YOLOv5 [3] to the latest iteration, YOLOv8 [4], in our pursuit of advancing object detection and segmentation capabilities. The model is built on a foundation of extensive pre-training on millions of images and places a particular emphasis on its zero-shot learning capabilities.

In our AutoDrone Re-ID project, our primary focus is on the detection of people using a webcam mounted on the drone. The YOLOv8 model assumes a crucial role in identifying individuals within the camera frame, thereby enhancing the overall efficiency and safety of the drone's operation. The YOLOv8 model is implemented using the PyTorch framework, offering a diverse selection of pre-trained models, each striking a unique balance between speed and accuracy. Our team opted for the YOLOv8s model, a more compact version with a parameter count of 11.2 millions. This strategic decision ensures optimal performance within the limitations of the drone's hardware.

In the context of our AutoDrone project, the YOLOv8 model has successfully demonstrated several key functionalities. Utilizing the YOLOv8 model, our implementation enables precise people counting in a given scene over a specified period. This capability facilitates the comprehensive analysis of crowd dynamics, providing valuable insights for operational decision-making. Furthermore, in alignment with the global emphasis on safety measures, our YOLOv8 model has been enhanced to incorporate mask detection capabilities by trained with a random sample of MaskedFace-Net dataset [5]. It can accurately identify individuals, distinguishing between those wearing masks and those without, thereby adding an extra layer of safety in public spaces.

D. Camera, LiDAR, and FCU

First, the implementation of the Robot Operating System (ROS) on the Jetson Orin Nano involved a detailed process of hardware and software setup. This setup was fundamental for installing ROS libraries and configuring the operating system to support the demands of real-time robotic control. A key aspect of this implementation was the development of custom ROS nodes. These nodes were designed to perform specific functions such as data acquisition, processing, and transmission of commands. They were capable of publishing and subscribing to topics within the ROS environment, facilitating a dynamic data exchange system.

Next, various sensors were integrated, including cameras, GPS, IMUs, and LiDAR into the ROS framework, this required the development of custom drivers and interfaces to ensure a seamless flow of data from the sensors into the ROS system. In parallel, a robust connection between the Jetson Orin Nano and the drone's flight control unit was established using MAVROS and MAVProxy. MAVROS acted as a bridge, enabling real-time communication and command exchange.

A link with this protocol was set up for transmitting control commands from the ROS nodes to the drone and for receiving flight data from various sensors in the flight control unit.

Then, to enhance the drone's sensory capabilities, the drone involved mounting additional cameras and LiDAR [6], as shown in Fig. 8. The placement of these sensors was strategically chosen to optimize the field of view and depth perception. An important step in this process was the calibration of the sensors to ensure accuracy in data capture. Algorithms were then developed to merge data streams from these cameras and Lidar. This fusion created a comprehensive view combining both image and depth information, essential for advanced perception. To ensure efficiency, the optimized image capture package V4L was employed instead of openCV to minimize latency in data transfer and processing. RVIZ, a visualization tool within ROS, was customized for real-time monitoring and debugging of the fused data. This setup facilitated a clear visualization of how image and depth data were combined, aiding in system refinement and development.

Fig. 8: Physical Setup of the Camera and Lidar

The project focused on developing algorithms for obstacle avoidance, tracking, person detection, and counting. These algorithms were designed to utilize the data provided by the integrated sensors, enabling the drone to dynamically interact with its environment. Gazebo, a robotic simulator was heavily used in testing these algorithms, as shown in Fig. 9. It provided a controlled environment to simulate various scenarios, allowing for the fine-tuning of algorithms before real-world application. By setting up the obstacles in the simulated world, the drone's obstacle avoidance function can be tested by letting the drone found its own way out in a maze. This process involved the use of Lidar and the motion control of the drone.

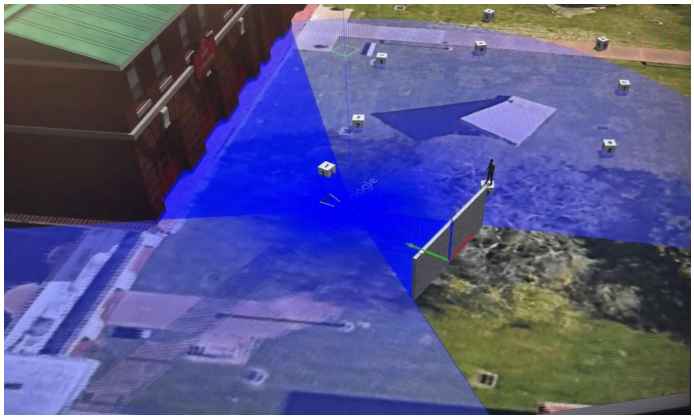

Fig. 9: Scenario Setup in Gazebo

Finally, the data collection and analysis during these simulations were vital in assessing and optimizing the performance of the algorithms. This iterative process involved making adjustments based on performance metrics to enhance efficiency and accuracy. Once the algorithms were satisfactorily developed and tested in the simulator, they were integrated into the ROS environment. This integration entailed coding the algorithms into ROS nodes and ensuring they interacted seamlessly with other system components. Preliminary field tests were then conducted to evaluate the performance of these algorithms in real-world conditions, with observations from these tests used to make further adjustments and improvements.

E. Software in the Loop (SITL)

In executing the SITL simulation, a detailed virtual environment was established in the Gazebo simulator to closely mimic real-world conditions. This environment included models like buildings and trees, sourced directly from the Gazebo library. Additionally, to recreate an accurate geographical setting, a plane was generated using satellite imagery from the Google Maps API [7]. This incorporation of real-world geographic data and physical objects greatly improved the authenticity of the simulated environment.

Additionally, human models were crafted using MakeHuman, a sophisticated tool designed to create realistic human avatars, and were subsequently rendered with Blender software. The files exported from MakeHuman included a mesh model, texture images, and a UV map — which is a flat representation of the 3D model used to map textures. All these components were imported into Blender to produce the final 3D rendered model. A demonstration of the human model creation process using MakeHuman and Blender is presented in Figure 10. This method provided lifelike representations of humans, which is essential for the drone's person re-identification capabilities.

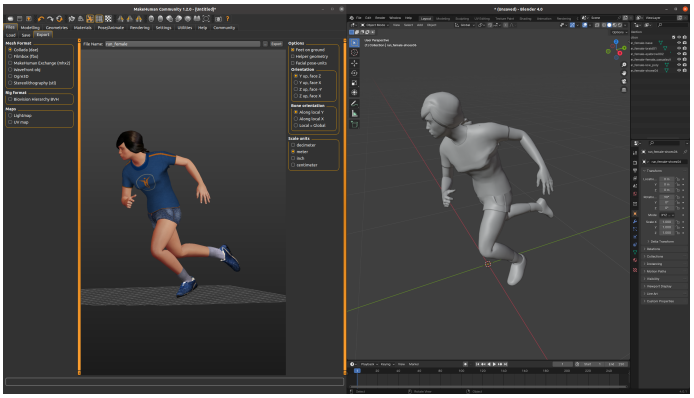

Fig. 10: Human Model Creation Process: (left) MakeHuman / (right) Blender

The centerpiece of the simulation was the Iris quadcopter model, equipped with a camera and a LiDAR sensor. The integration of these sensors into the Iris quadcopter model was crucial for the autonomous drone to navigate and recognize humans within the simulated environment accurately. A sample of the fully integrated world is illustrated in Figure 11.

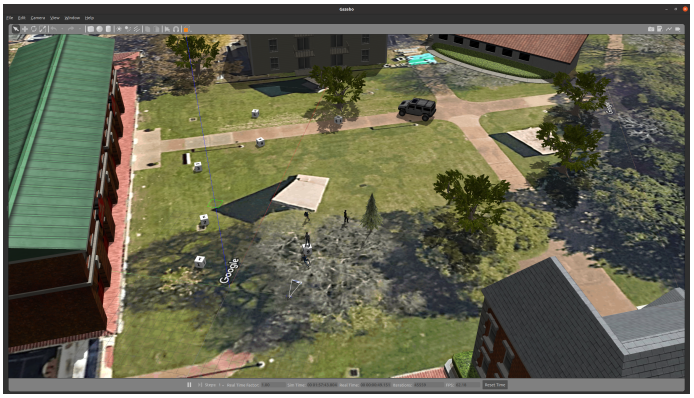

Fig. 11: Gazebo World Environment

Ardupilot's SITL is essential in the setup of the simulation for the project. It simulates the flight control hardware for the quadcopter in Gazebo and establishes a bridge to the ROS environment. The block diagram shown in Figure 12 illustrates the communication pathway between the drone's Flight Control Unit (FCU) within Ardupilot's SITL and the Gazebo simulation environment, facilitated by the Gazebo Driver and MAVProxy. Gazebo handles the simulation of the drone's physical responses and sensor inputs, which are then interfaced with the ROS environment through ROS topics. ROS, in turn, sends control signals to the FCU using the MAVLink protocol via UDP ports. This process coordinates the drone's simulated activities and completes the loop of the integrated simulation system.

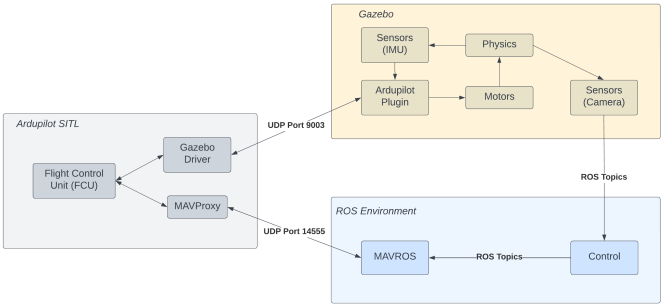

Fig. 12: SITL Block Diagram

In developing the autonomous drone's control and object detection functionalities, specific libraries were selected for optimal performance and system compatibility. The "gnc_functions" library from Intelligent Quads was chosen for drone control [8]. It provides a comprehensive set of guidance, navigation, and control (GNC) functions made for autonomous drones, which enables precise maneuvering and stable flight. For object detection, the OpenCV deep neural network (DNN) module was used. The shift from Python to C++ for enhanced performance led to the selection of OpenCV's DNN module, which is well-suited for C++ environments. This module, particularly effective in real-time object detection, was utilized to load the ONNX YOLO model.

The integration of various simulation tools and models played a key role in developing a robust and realistic testing environment for the autonomous drone. This foundation is crucial for the drone's successful application in real-world scenarios.

F. 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

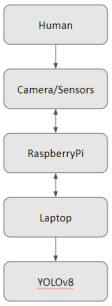

Fig. 13: Object Detection Flow Chart.

Fig. 13 shows the flowchart of our object detection with the Raspberry Pi embedded system. As you can see the sequence

in the flowchart is by streaming video to Laptop via Raspberry Pi and use YOLOv8 to detect Human.

Fig. 14: YOLOv8 model test on real-world dataset with Raspberry Pi 3B.

Fig. 14 shows the results of real-world dataset testing on the Raspberry Pi 4B using the YOLOv8 model. It can be found that the accuracy is considerable.

As we we can clearly see the performance differences among them, As shown in TABLE I was the performance difference we had with running Yolov5 on Raspberry Pi 3B, Raspberry Pi 4B. We kept this table here as a referenece and record.

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 Orin Nano Integration

This section is from the spring 2023 semester. During the integration of the Jetson Orin Nano 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. 15: Hover drone in the air operated by Jetson Orin Nano.

Before implementing this experiment, we tested several other simple movement scripts. Fig. 15 shows the drone hovering successfully in the air for the first time, while Fig. 16 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. 16: QR code tracking results on Jetson Orin Nano.

This section below is from the fall 2023 semester.

For this semester, the Robot Operating System (ROS) was successfully integrated in the Jetson Orin Nano, establishing a robust connection with the Ardupilot Flight Control. This integration enabled the comprehensive control and monitoring of the drone system, laying a foundational framework for advanced drone operations. This integration enabled the accessibility and functionality of both external (camera, LiDAR) and internal (GPS, IMU, barometer) sensors within the flight control unit. These sensors now effectively contribute fundamental flight and awareness data to the system. Additionally, the project made strides in enabling motion control of the drone through ROS nodes, facilitated by the effective communication between the Jetson Orin Nano and the flight control unit via MAVROS/MAVPROXY. However, the current motion control system is somewhat limited by the constraints of existing wrapped functions, highlighting an area for future improvement. The project team anticipates enhancing this aspect by developing custom functions using underlying FCU code for more direct interaction with the flight control system.

The project also achieved considerable success in the area of multiple sensor fusion. The integration of camera and lidar sensors was not only successfully accomplished but also optimized to ensure efficient data streaming to ROS topics. This development provided the drone with basic yet powerful

perception capabilities. This fusion effectively combined environmental imaging with depth estimation, offering a more comprehensive understanding of the drone's surroundings, as shown in Fig. 17 and Fig. 18. Additionally, the team managed to establish and visualize the spatial orientation of the sensors relative to the drone body using RVIZ, thereby enhancing the drone's spatial awareness by rearranging the transformation between the frames. A significant reduction in the latency of data capture was achieved by transitioning to more efficient capture packages. The project aims to further refine the range and angle mapping between the camera and lidar sensors and develop additional applications to enhance the drone's perception capabilities.

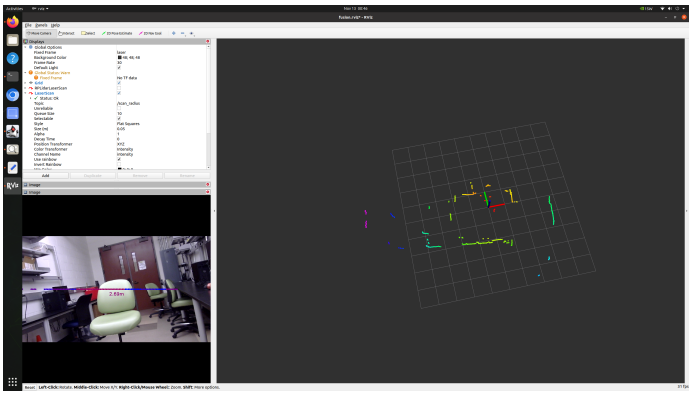

Fig. 17: Camera-LiDAR Fusion with Distance Estimation(Close).

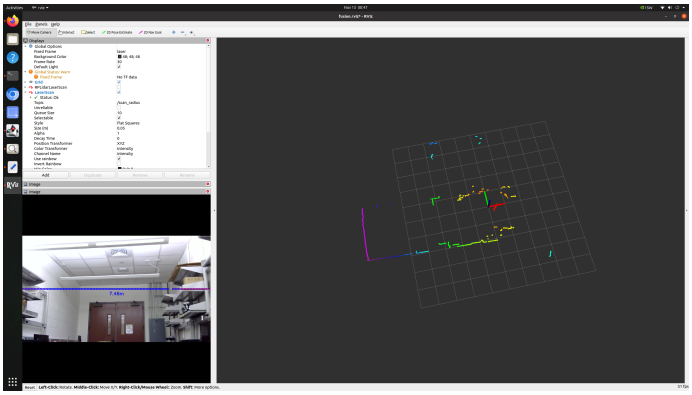

Fig. 18: Camera-LiDAR Fusion with Distance Estimation(Far).

In the realm of on-board algorithm development, the project made substantial progress, aligning closely with initial objectives, although some areas still under development. The obstacle avoidance and tracking algorithms, critical for the drone's navigation and environmental interaction, were developed and tested in a simulator. These preliminary versions demonstrated basic environmental awareness, but full functionality is pending due to a limited understanding of the internal workings of the Flight Control Unit (FCU). However, the mission planning algorithm, a vital component for autonomous drone operations, is still in the development phase. The team's focus is currently on enabling the drone's autonomous sensing and movement in controlled scenarios, which is expected to significantly advance the drone's operational capabilities.

C. Yolo Function

The YOLOv8 model has proven its efficacy in detecting individuals within the camera frame, significantly contributing to the overall safety and situational awareness of the autonomous drone.

Fig. 19: People Detection and Counting.

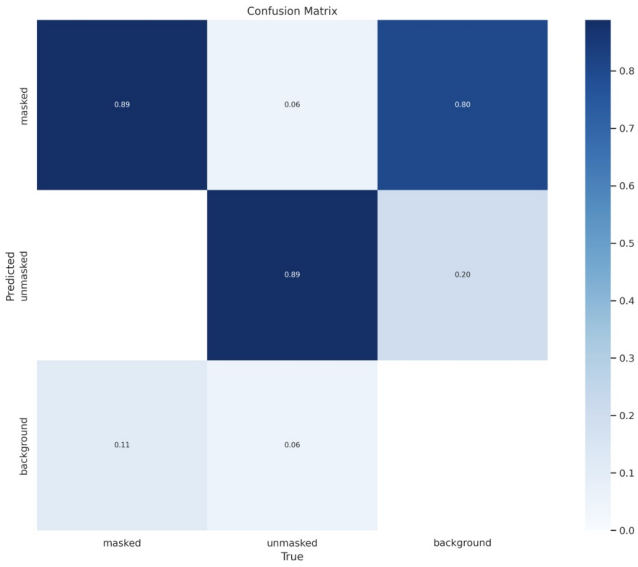

Fig. 20: Confusion Matrix of Mask Detection Model.

Fig. 19 shows the outcomes of the people counting function. Initially, individuals in each frame are identified by a red bounding box. YOLOv8 tracks these individuals and tallies them when the center of the bounding box intersects with a predefined line. For this specific experiment, the line is positioned at the center of the image.

Fig. 20 illustrates the performance metrics of the mask detection model. The achieved accuracy of this model is 89%, indicating that individuals wearing masks can be correctly identified with an accuracy rate of 89%.

D. Software in the Loop (SITL)

The SITL simulation conducted in this semester was showcased through a series of three experiments, each designed to test a specific aspect of the autonomous drone's capabilities. The first experiment focused on waypoint control, where waypoints were set with precise coordinates including position, altitude, and orientation utilizing set_destination() function in the GNC library. The waypoints were designated as (5, 5), (15, 5), (15, -5), and (5, -5), with an altitude of 2 meters. At each waypoint, the orientations were set to 90, 180, 270, and 360 degrees, respectively. The drone successfully executed the task, initiating takeoff and then navigating to each waypoint in the prescribed sequence. A demonstration of a square flight trajectory with 4 waypoints is shown in Fig. 21. This outcome highlights the precision of the drone's flight path control and the efficacy of the GNC library.

Fig. 21: Navigation through 4 Waypoints in Sequence (camera positioned vertically downward): 1 (top left) → 2 (top right) → 3 (bottom left) → 4 (bottom right)

The second experiment evaluated the drone's human detection system. Four human models, featuring various genders and poses, were introduced into the simulation environment. To further challenge the system, interference objects were also added to the environment, simulating potential real-life distractions and occlusions that the drone might encounter. The OpenCV DNN module, equipped with the YOLO algorithm, was utilized to detect and count the number of humans in the frame. The experiment was successful as shown in Fig. 22. The system accurately identified and counted the human models, demonstrating the reliability of the object detection implementation.

Fig. 22: Counting the Number of Humans in the Frame (camera positioned horizontally)

The final experiment was designed to test the drone's capacity for dynamic human detection within its field of view. In this scenario, the drone was programmed to continuously rotate about its axis until it successfully detected a human model within its camera frame. The drone's software was programmed with algorithms to determine the human's relative position in the frame in real-time and adjust its orientation accordingly. As the drone began rotating, it utilized the integrated camera to scan the environment and processed the data with the YOLO model. The rotation speed was carefully calibrated to ensure sufficient time for image processing and recognition and balance between the agility of response and the accuracy of detection. A demonstration of the successful human tracking task is shown in Fig. 23. This result showcases the drone's advanced ability to autonomously identify and focus on human subjects in its environment.

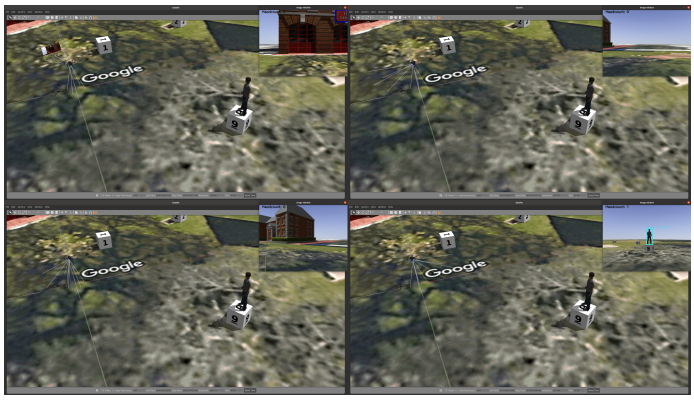

Fig. 23: Human Tracking Task (camera positioned horizontally): Rotate (top left) → Rotate (top right) → Rotate (bottom left) → Stop (bottom right)

E. UI Interface

From the system design diagram in Fig. 7, 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 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. 24.

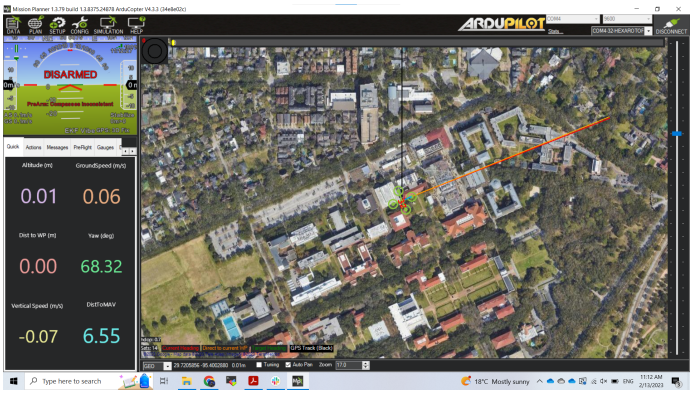

Fig. 24: Mission Planner GUI Interface.

Fig. 25 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.

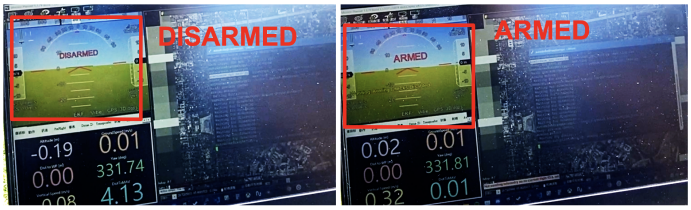

Fig. 25: Drone's real-time status on Mission Planner GUI.

IV. Discussion

This semester's achievements in drone technology demonstrate remarkable progress, especially in hardware and software integration. The successful launch of the 4-axis plane with custom PCB is a testament to our ability in advanced UAV design and integration. Coupled with this, the implementation of Software In The Loop (SITL) for simultaneous object detection and drone control, along with the camera-LiDAR fusion, has significantly enhanced the drone's environmental awareness and operational efficiency. The application of the YOLOv8 model for mask detection further emphasizes our commitment to integrating cutting-edge AI into practical solutions.

However, challenges still remain, particularly in achieving seamless integration and operational consistency across diverse environmental conditions. The complexity of real-world scenarios often presents unforeseen obstacles, such as variable lighting, unpredictable weather, and dynamic obstacles, which can affect the drone's performance. Moreover, ensuring consistent and reliable data processing from the fusion of camera and LiDAR inputs remains a technical challenge, requiring further refinement of our algorithms and sensor calibration methods. Another challenge is that the power consumption for the whole system is insufficient, causing the drone to shut down a few seconds after take off. Addressing these challenges will be crucial for advancing the drone's capabilities and ensuring its applicability in a wider range of scenarios.

Looking forward, the focus will be on enhancing the drone's autonomous functions and decision-making algorithms to tackle more complex tasks and adapt to changing environments. We aim to move from using the SITL simulation to testing with a real drone. A key goal is to make the drone use less power and fly longer distances, which might mean looking into new kinds of energy or better batteries. Adding more sensors and ways to communicate could also help the drone be used in more areas, like checking the environment or helping in emergencies. By continually developing and refining these systems, our team's objective is always to keep the drone technology advanced and reliable for a variety of uses.

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] Zhangqiang Ming, Min Zh, Xiangkun Wang, Jiamin Zhu, Junlong Cheng, Chengrui Gao, Yong Yang, Xiaoyong Wei, "Deep learning-based person re-identification methods: A survey and outlook of recent works", DOI:https://ar5iv.labs.arxiv.org/html/2110.04764

[2] 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.

[3] 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.

[4] Jocher, G., Chaurasia, A., Qiu, J. (2023). YOLO by Ultralytics (Version 8.0.0) [Computer software], DOI: https://github.com/ultralytics/ultralytics

[5] Cabani, A., Hammoudi, K., Benhabiles, H., Melkemi, M. (2021), MaskedFace-Net – A dataset of correctly/incorrectly masked face images in the context of COVID-19. Smart Health, 19, 100144, DOI: https://doi.org/10.1016/j.smhl.2020.100144

[6] RoboPeak Team. (2014). rplidar_ros (Version master) [Computer software]. https://github.com/Slamtec/rplidar_ros

[7] Gazebo Tutorials. (2023). Example World [Online tutorial]. https://classic.gazebosim.org/tutorials?tut=static_map_plugin&cat=build_world#ExampleWorld

[8] Intelligent-Quads. (2023). iq_gnc (Version master) [Computer software]. https://github.com/Intelligent-Quads/iq_gnc/tree/master