Drone Mounted Object Detection and Recognition using Machine Learning and Infrared Sensors

Derek Murdza, Cannen Carpenter, Jazmine Roman, and Kevin Nilsen

University of Central Florida, Department of Computer and Electrical Engineering, Orlando, Florida, 32816, U.S.A.

Abstract - This paper presents the methodology of developing a manual drone utilizing Python to classify objects in conjunction with infrared distance sensing for distance detection. There are multiple engineering specifications that are to be met following the completion of this product including: (1) Manual takeoff and landing procedures; (2) Object detection using a real-life training model; (3) Distance detection of specific objects in correlation to drone position. This paper also demonstrates research of formulas and background information that is crucial to the success of this product

Index Terms — Aircraft navigation, artificial intelligence, LIDAR, object recognition

INTRODUCTION

In recent years, a lot of work has been done in the area of artificial intelligence and quadcopter drone development. Neural networks and AI training has piqued a lot of interest from researchers due to its near limitless applications in human lives. A primary goal is to make the machines people deal with every day more accessible by facilitating a natural method of communication between humans and computers. Additionally, commercial and consumer grade drones and Light Detection and Ranging (LiDAR) technology have developed greatly over the past several decades. Today the aerial robots find many uses in areas like environmental scanning, recreation, and military operations. The group wishes to combine these two impressive technologies together; using machine learning and LiDAR technology to create a quadcopter that can detect a set of given targets. The drone has the capabilities to fly and detect several different objects with its sensor. Our drone has the standard design of a quadcopter and the focal point of this project is the machine learning software for its visual and sensory analysis. A second major focus is the optics of the project and how the infrared sensors communicate with the machine learning software to determine the placement of targets. Machine learning is a concept that has gained a lot of enthusiastic attention in the last decade and has a wide variety of applications. Machine learning is the basis of artificial intelligence, which is developed with a neural network mapping within the machine learning code. This neural network allows for the machine's code to learn according to categories that we will feed the machine via its optical design.

ENGINEERING SPECIFICATIONS

The following specifications are essential to the success of this drone and also required as part of the senior design curriculum:

1. The drone must be able to fly in all three axes
2. The system should be able to be controlled by an external computer if necessary.
3. The system should be able to hover and maintain a single position in air for one minute or more.
4. The system should be able to pass pre-arm checks for safety and physical purposes
5. The system should have sensors that are able to detect and measure the distance of objects at least 0.75 m away
6. The system should be able to accurately and consistently identify objects 4 m away from its cameras.

The above specifications have been met using multiple types of techniques involving hardware and software, more specifically in the aspects of machine learning since the drone model needed to be trained to detect objects prior to achieving flight.

TESTING STANDARDS

The group needed to be able to test the efficiency of the object detection system working with the infrared sensors. The primary workflow of the system was to have the distance scanner look for an object, and whenever it detected something within a certain range, only then would the camera activate and the AI would make a prediction on whether one of the targets was in view. The group hypothesized that this would reduce the needed computational power and speed up the total runtime of the object detection by reducing the number of computations needed.

These characteristics would be tested using the following process: two separate sets of the object recognition software would be done. The first would be done without the distance scanner, and the AI would

continually test whether or not it could see an object for every frame of the continuously running camera. The second set would use the LiDAR scanner, and it would only make a prediction when the sensor detected that an object was nearby. In each of these sets, each of the three unique targets that the AI was trained with would be placed in front of the camera and the processing time and confidence of the model was tested. As was predicted, the test that used the LiDAR scanner reduced the overall computational load on the processor, and reduced the number of unnecessary computations were the AI would try to look for an object when nothing was in front of it.

DESIGN SPECIFICATIONS

The software component of the project consists of three parts: training, simulation, and real-world application. Training is done through the collective effort of three softwares: PyTorch, Gazebo, and PX4. The PyTorch framework is used in conjunction with the open source model we have chosen. Research and testing models with low level abstraction gave us insight into what qualities we should really be looking for. The YOLO object detection system is the model that caught our eye. This system is based in C, as it utilizes the Darknet neural network, but there was a work-around. We were able to find the Computer Vision tool platform Roboflow. The platform's site allows management of public custom datasets, as well as deployment of a variety of models. The packages they supply include a YOLOv8 model that makes use of PyTorch instead of Darknet. This meant we wouldn't stray away from our initial testing and research. Moving forward on this platform, our first step was to create an imageset to train on. We made use of the annotation tool MakeSense. This tool allowed us to upload images of the objects we are detecting and manually annotate each image with a bounding box. Then once every image is annotated appropriately, the site allows you to export in a zip file formatted specifically for the YOLO model. Finally, making use of the Roboflow platform, we uploaded our imageset and ran it through our open source YOLO model. Using this model we were looking to reach a validation accuracy of 90% or higher. We were successful in this aspect and are confident that the model we trained can produce accurate predictions on our personal computer.

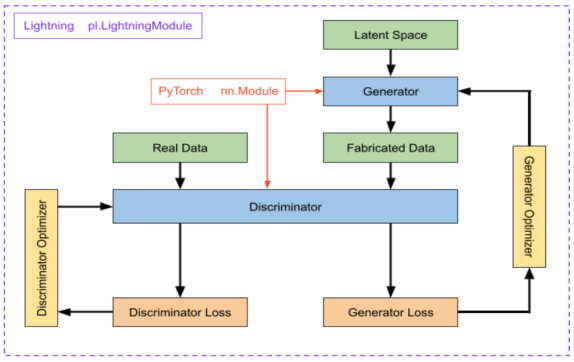

Figure 1: PyTorch Integration Table

Configuring the drone based on how well the model performs. The model's performance is heavily affected by the computing power of the Raspberry PI, so adjustments will need to be made during testing.

On the navigation end, we experimented with the control system of PX4 within the simulation environment for potential/future autonomous flight implementation. Movement is basic, as altitude will not be a variable within our project, but gaining an understanding of the physics environment and how to track velocity/position components of the drone is extremely important. Given that Gazebo and PX4 support each other, the resources they provide within their Guides/APIs is essential to furthering our understanding of how the drone reacts to its environment.

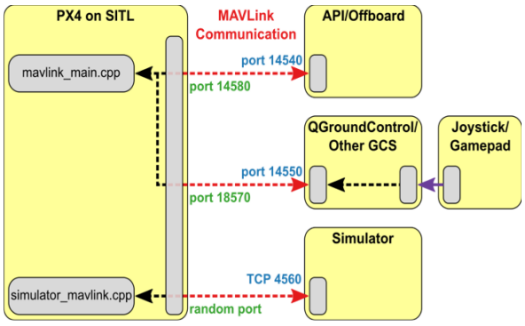

Figure 2: PX4 MAVLink Correlation

INFRARED SENSING

For optical design, the team wanted to ensure that the drone would not collide with large objects in the environment. So, a series of infrared optical sensors are placed around the drone in addition to a distance sensor that rotates continuously, acting similar to a light detection and ranging (LiDAR) scanner

The spinning distance scanner is placed on the bottom of the drone. However, this will lead to the four drone legs blocking any distance information. To supplement this, four static proximity sensors are placed onto the legs of the drone. For these sensors, the power will decrease exponentially as a function of distance, and the farther the object away is. The rate of the power loss depends on the divergence of the LED from the optical axis. This means that their range is quite poor, and the group decided to ensure their range is about 0.3 m.

For the distance sensor, the group wanted a continuously rotating module that would measure the distances of objects in the environment. Although this system is currently only used for nearby object detection, a more advanced system could have a much higher range with more accurate data, and could be used in conjunction with the object recognition AI to create better flight paths.

These are all assuming an ideal scenario, and the surface of the object is perfectly flat with the surface normal vector facing the sensor. Despite this, these assumptions are still useful for getting a close approximation to the distance.

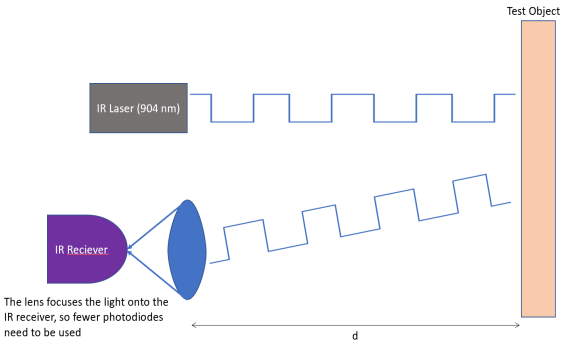

Figure 3: Pulsed time of flight Imaging

The basic diagram of the time of flight circuit, as shown in Figure 4, demonstrates how Since time of flight requires very fast signal processing, the group found it best to analyze the input using an analog signal rather than a digital one to reduce processing power requirements. This drastically reduces the required clock speed from signals in the range of gigahertz to those in the megahertz. [6]

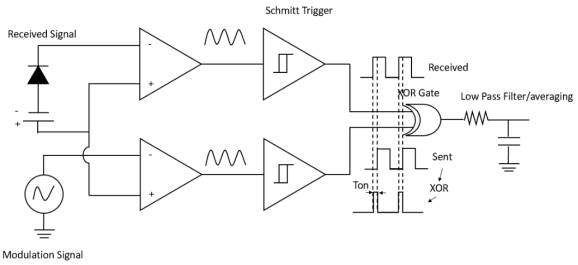

Figure 4: Time of flight circuit diagram (Note: this circuit is simplified and missing several components for the sake of visualization)

The laser scanning system uses the input of the modulator, and then compares that to the signal being received by the sensor. This generates a lower frequency output, and by analyzing the average of the signals, the time of flight, and therefore the object distance, can be calculated. Alternative configurations split the light into two beams, comparing the sent and received signal directly. This is a more accurate way to directly measure the difference in signal between the light exiting the laser and the light being reflected back to the sensor. This ultimately gives a more accurate distance reading. However, this would dramatically reduce the power of the light exiting the system and reflecting back. Partial deflectors exist, but higher end optics like this can often drive up cost.

In general, direct time of flight imaging tends to be quite low resolution compared to other methods of distance sensing. Typically, a LIDAR scanner may have a significantly more powerful laser diode and larger optics, and more sophisticated electronics to increase this resolution. Though, given the time, budget, and size constraints of this project, the group chose their current model.

CAMERA MODULE

The next focus is on the eyes of the drone, its visual system, which starts with the input signals received from a camera that is attached to the drone. There are two cameras attached to the drone, one facing the z axis of the drone and one facing the x axis of the drone. This will allow the drone to see forwards and below it. For our vision navigation drone test, we are requiring the drone to simply navigate in multiple directions as this is a prototype.. This is the reason why we choose to use two drones. One to detect obstacles, such as balloons in its x axis view and another to find its proper landing target labeled on the target below it. This vision navigation machine learning technique is expanded

more within the optics and software sections respectively.

When choosing a camera, it is important to consider the constraints on sensor type, pixel resolution and latency, the built in camera features, the lens and the video transmitter that is attached.For vision navigation drones, the cameras usually have a CMOS or a CCD image sensor inside. CMOS cameras are less expensive than CCD cameras but lack the ability to react quickly to changes in low light. This is important to take into account when choosing the camera for our drone because any sudden change in light that causes a lack of visibility can cause confusion within the drone flight controller system and cause a crash. A CCD camera will give us the best result for a vision navigation drone. On the topic of resolution, the higher the resolution, the slower the latency so this is also a large topic to dive deeper in when looking at the type of resolution we would like our vision navigation drone to have.

The RaspberryPi 4 Model B is used to handle our trained YOLOv8 model. We chose this model for several purposes. There are many resources available for its implementation, and is widely used within the subject area. This makes the build process much smoother for us because it alleviates a lot of the hassle of working with plugins or packages to get a fully trained model onto our drone. With out of box support, and the Raspberry Pi camera module we selected, it also gives us the ability for “real-time inferences.”After training is completed, the model is deployed onto the Raspberry PI and allows for onboard image processing and classification, getting full use of our YOLO model. The setup for this is to first install the 64-bit Ubuntu OS onto the board computer;the model will only support 64-bit ARM so this is required. Once the OS is set up, Roboflow installation is done via pip, a package installer for Python. After that, the camera module had to be enabled via the Raspberry Pi Config. Next we installed the package manager MiniConda. The most important packages we installed were OpenCV, for video manipulation, PyDrive, for file management with Google Drive, and RPi.GPIO, used for utilizing the GPIO pins on the Raspberry Pi. After this we were set to implement our YOLO model and make predictions. The PixHawk Flight Controller is crucial for sending data to the camera, which is connected to the Raspberry Pi motherboard.This camera provides high efficiency and is currently known as the best flight controller to be implemented for a civilian used vision navigation drone.

The PixHawk will allow user interfacing in which the region of interest can be chosen via the camera mapping provided by the PiCam. As described by the figure below, there are in fact multiple different function calls which are used to set values for altitude, takeoff markers, radian to degree conversion and vice versa, as well as simple device connection.

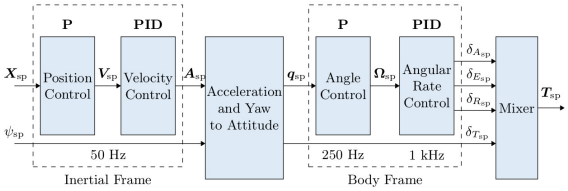

Figure 5: PixHawk Communication Flow Chart

The above figure describes the flow in which the PixHawk module determines specific variables depending on the data being inputted via the camera module. This module is crucial to ensuring that all communications are recorded and that all functions are integrated correctly.

OBJECT DETECTION

The process of testing the object detection software was done using multiple model creations and then each model was trained based on two sets of objects. Initially, basic red, green, and blue cubes were used for training and proceeding that was the implementation of more complex items. These included an orange water bottle, a brown stuffed animal, and a blue stuffed animal. The change in colors allowed a wider scale of object detection as the process became more complex with colors away from the typical RGB field.

The implementation of the object detection model consists of three stages: data collection, training, and inferences. The data collection portion revolves around creating an annotated imageset that the model can validate its training runs on. Our first step was recording dynamic footage of the objects that will be used in our obstacle course. We recorded about a minute worth of footage total for our three objects. This footage was split into its individual frames, ending up with 600 images. We packaged them within a folder to be uploaded to the MakeSense tool. Once in the MakeSense tool, we manually went through each image and labeled them accordingly. The dataset was then expanded using data augmentations to produce “new” instances. Finally, the imageset was exported in YOLO format, concluding the data collection stage.

The training stage starts with uploading our newly imageset to the Roboflow website. The platform

provides an open-source notebook that we were able to augment to take our custom imageset. Training was done with a batch size of 16 and 25 epochs. This was more than enough for the complexity of our imageset. Results proved this to be the suitable configuration as we ended with over 95% mAP for all of our predictions. After the training is finished, the Roboflow package allows us to upload our trained model to their platform to be accessible at any time. Shown below are results from our training stage. We can see how loss and mAP progressed over the 25 epochs.

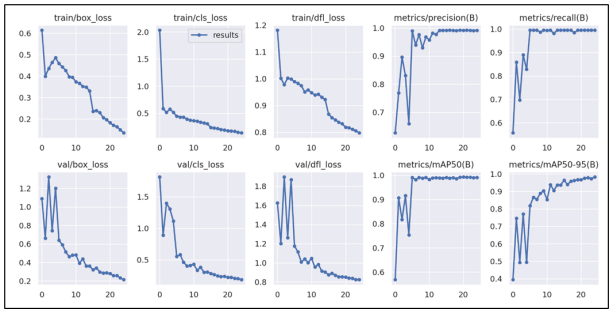

Figure 6: Training Batch Result

The figure below are some of the resulting predictions taken from a random batch of images. Each image within the batch has the classification along with the confidence percentage for each.

Figure 7: Training Batch Result

Inference stage is essentially the testing phase for our object detection. The Roboflow platform again comes in handy since we saved our trained model to their servers. We're able to use pip to install their packages locally and download our model. Actively making predictions on images of objects is done by referencing the Roboflow api and passing images into the prediction function under our model.

The predictions can be saved, displayed, or analyzed by calling other functions within the Roboflow package. This was all done locally in a WSL environment, but deployment of the model onto the Raspberry Pi is possible through their inference server Docker container. This is the most important step in translating our work from a local machine to the Pi device. We did run into performance issues regarding the model. The model was very intense on the Raspberry Pi. This was obvious in hindsight, as for the most part, machine learning is accelerated by GPU utilization. We combatted this by working in unison with our LiDAR sensors, and limited inferences to only when objects are physically in front of the camera. The results of our trial runs are displayed below in Fig. 8. With nearly 500 predictions being made, only 25 were either mislabeled, too low for our confidence threshold, or not detected at all.

Error Rate Over V2 Runs

Predictions	Errors	Error Rate
499	25	5.01%

Figure 8: Total Error Rate % from our Trial Runs

PYTHON PROGRAMMING

Python is one of the most popular programming languages, and reasonably so. Python values accessibility above all else with its simplified syntax and many abundance of useful packages and libraries. Python isn't limited to a single OS, as it can be run on Linux, macOS, Windows, Solaris, etc.. Pair this with the fact that Python scripts can be made to run at a much faster rate with a GPU, this makes it a great money-saver for companies that don't want to invest in cloud services from companies like Microsoft (Azure), Amazon (AWS), Google (Google Cloud). GPUs can cover some of the ground lost by not investing in these more expensive "Machine Learning as a Service" options without losing out on too much productivity. The possibility of GPU utilization is a big contributor to a project as well. We are using a system with a NVIDIA RTX 3060ti, which has 4864 CUDA Cores, and 152 Tensor Cores. This was valuable in our initial training, but we were able to train our model in a Jupyter Notebook, saving our own resources.

PyTorch and TensorFlow are the two machine learning frameworks that are popular in today's world. Those were also the two options given to use by the original sponsor as well. Our sponsor highly advised us to utilize PyTorch though. This was because the resources they had already dedicated to their own project were done using PyTorch. That meant that we would have references for moving forward with our project. Doing our own research though, we found that there were more reasons to select PyTorch over TensorFlow. One was because of the ease of implementation. PyTorch is described as being more in-line with the original Python libraries. We believe this makes Python 'easier' to learn when you already have knowledge of Python. Python is also known as an 'easy' language to learn compared to others. The community backing PyTorch is also massive and out-matches TensorFlow. For our own research, this would be much more advantageous as this is our first time utilizing a framework at all. We can really make use of the open-source packages that are out there to improve our convolutional neural network. This was evident when we came across the YOLO model. We were able to find an implementation of it using PyTorch.

We are using Anaconda to manage our packages and train our model in our simulation system. This was recommended mostly for its access to a large number of data science packages, which should aid us in training our model. PIP is necessary for PyTorch installation onto the Raspberry Pi but will be used in sync with Anaconda to handle package installation. Using PIP within the Anaconda environment will just install the packages as normal with no complications tied to it. Anaconda also grants the ability to update all packages with a single command. The package management of Anaconda will make handling the model safer and easier.

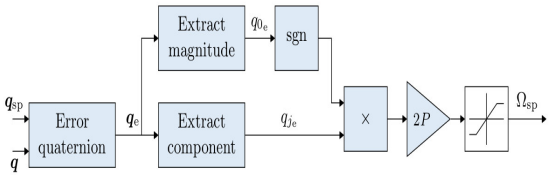

Figure 9: QFlightControl Circuit Diagram

The above diagram is used to help determine full attitude control when it comes to integration of Python scripts into the flight control module. The variable q represents the full attitude control necessary for communication between the script and the actions it is requesting.

SAFETY CONSTRAINTS

The team also needed to consider the safety constraints of the system, Given that a drone can be quite a sensitive system, there are many mass and weight distribution requirements that the group must face to construct a functioning drone. Considering the construction of the drone, the robot is constructed with four legs (or arms) with the PCB board in the middle connecting the four legs. The four legs will each have brushless motors attached to the top plane (or y-plane in space) of each leg.

On the top of each motor, there is a set of two propellers. The propellers are the mechanical aspect of the robot that catches the wind and allows movement in space and time. The quadcopter has a set of two propellers, two that connect clockwise and two that connect counterclockwise. These set of propellers are a pair by legs, therefore one pair of legs is connected via one node and set as the forward propellers and the other set of legs is connected via one node and set as the backwards propellers. The propellers will move at a high speed of anywhere between 8000rpm to 9000 rpm. The propellers are to be of the highest constraint because at this high speed and the novelcy of the team with robot building, it must be held to the utmost safety.

The way our team will solve the propeller safety factor that will ensure safety is have propeller protector rings. Propellers are the aerological part of the robot that allows the drone to fly. The propellers are attached to the brushless motors. There are four motors on the end of each leg of the drone with two propellers attached to the end of the motor, this allows flight upwards, downwards, forward and backwards. In order for the drone to fly forward, the flight controller speeds up the designated corresponding brushless motors which are attached to two out of the four legs of the drone that share the same node. The two legs that are used for movement must be connected in series. The two brushless motors that are attached to these corresponding legs move at a faster rate of about ten times while the two back motors stay at a slower speed or at the speed it was prior to forward / backwards / side command. This causes the two front propellers to go faster, therefore propelling the drone in the direction in space that was determined by the controller.

PHYSICAL CONSTRAINTS

The creation of the drone has multiple constraints that the group must work around in order for the project to be successful. This section contains lists of the different types of challenges that a completed drone will face in order to have a proper flight. These include the size of the system, the weight, and safety concerns among others. Safety is a key factor when creating a drone, and particularly so with an autonomous one. The drone should avoid making any collisions with humans or other private property. This is one of the functions of the object detection system.

Most of the other restrictions are due to the physical and technological limitations of the system. These factors include things like the required size, weight and thrust of the system. With any flying object, and especially a hovering object, all of these must be balanced in perfect harmony. Otherwise, the entire system risks catastrophic failure. An imbalance can easily lead to too much or too little lift, resulting in the whole system crashing. Any accidents at significant speeds and heights may completely destroy the flight systems as well as any number of other component systems. It is for these reasons that these constraints are considered by the group to be some of the most important in the entire project. Above all else, these constraints will make all of the difference between the success or the total failure during the final project and presentations. In order to achieve the expected flight height while also achieving battery life, the battery is another key component needed for extensive research for reliability, durability and safety. Our team is looking to start with a battery that is around 2300-2300 KV with a 3s battery voltage applied to it. This will cause the motors attached to each leg of the drone to spin at about 28,980rpm. More calculations and research within the area will need to be done and should be the team's main focus moving forward.

The battery is the key component for reliability because each motor, the vision system using the camera, the control system and the microcontroller of a raspberry pi will all rely on its efficiency to provide the correct voltage to each component for proper flight control and safety. The battery will also be the component that is causing the most heat. These thermo waves do need to be controlled in order to ensure that the robot does not set on fire neither does it melt or burn any of the wires or components. This aspect of the robot does require a project constraint of heat omission and will require great attention to prevent fires. For safety, the team will purchase a fire casing to enclose the battery chosen as well as a fireproof bag to place the drone in to prevent any spread of the fire. The team should also be aware of the closest fire extinguisher near the testing lab and have their devices in hand to call on emergency if needed. To address some of these concerns, there should also be at least two people in the testing lab at a time when performing testing.

NAVIGATION CONSTRAINTS

For the stretch goal of this project, autonomous navigation became a potential additive to the overall functionality of the prototype. Beyond the camera features, there are three other features that are crucial to building a successful drone that has visual navigation abilities. These include awareness, basic navigation, and expanded navigation. With awareness, the drone needs to have a reference to the edges of the testing environment, as without any insight on that, crashes are much more likely to occur. Adding to the simple object detection for the shapes hung from the top of the mesh cage for testing, the drone shall be able to detect the walls using the same mechanisms.Basic and expanded navigation are simple to integrate, and for this type of project, drastic altitude changes aren’t necessary. For expanded navigation such as “flips”, that kind of movement would be rather detrimental for the drone especially in terms of detecting objects.

In all types of flight, there are many natural occurrences that can easily cause the vehicle to tilt in multiple directions. Steady wind speeds, random wind gusts, or any number of other forms of environmental turbulence can influence the state of the drone’s flight at all times. The infographic describes how the blade direction can affect speed and altitude based on the tilt angle at any given time. If the blade of the propeller is traveling in a climbing direction, such as where the blades are exerting upward force, the drone shall increase in altitude within the same direction as the tilt. The same theory also applies to the retracting blade direction. Understanding these concepts is crucial to fine tuning software to offset any unwanted tilt, as well as to correctly fly in the case where tilt is needed to reach a given object.

CONCLUSION

The plans for the construction of the visual navigation drone show incredible promise. Following extensive research, the group has discovered several methods of developing an artificial intelligence with the capabilities to recognize specific objects in an area. They have also developed plans for ensuring the safety of the drone. The group also sees a great potential if it were to be used in a more commercial setting, where the scope, scale, and budget would be far greater. With

significantly more time and processing power available to the team, the image recognition AI could be trained to recognize more objects, with a greater efficiency. In such a setting, the drone could recognize more everyday objects rather than specific targets. It would be able to recognize trees, buildings, or even people. This could be combined with more mature instructions, where the drone can fly by also changing its height, and not just flying in a rigid two dimensional plane.

Eventually, this drone could be integrated with voice commands, and the drone would be able to carry out specific tasks spoken to it by a human.

Also, with higher quality parts, the image detection system could be made to be far more mature. With a larger budget, the LiDAR system could become able to create a three dimensional map of the environment. This would require much more expensive parts, and circuit components of a higher quality. The scanning system would have to rotate in three dimensions, and there would have to be enough processing power to accommodate this exponential increase in data. If the technology would be able to mature to this stage, it could be combined with the image acquisition algorithm. The system would then be able to tell exactly where an object is in its environment, its distance to it. From here, it could make calculations for the expected flight time, analyzing possible flight patterns and speeds to know exactly when it will arrive at the target. This fully matured three dimensional map could be used for very precise flight paths to specific objects, avoiding all objects and people efficiently. A more efficient flight path can have shorter completion times, and more satisfied customers for a potential business.

ACKNOWLEDGEMENT

The group would like to thank the UCF ECE department for their generous financial assistance with the project.

BIOGRAPHY

Derek Murdza will graduate as a Computer Engineer in May of 2023. He will work with Lockheed Martin as an Integration and Test Engineer Associate. He plans to focus on a Master's in Engineering Management shortly after.

Cannen Carpenter will graduate as a Computer Engineer in May of 2023. He will work at Ernst & Young as a Technology Consultant after interning there in the previous summer. He plans to use the many avenues available to explore possible career paths.

Jazmine Roman will graduate as an Electrical Engineer in May of 2023. She has worked with Siemens Energy as a student contractor since May 2019 within the service groups of sales, technology and operations. After graduation, she will work with Leidos Power and Delivery as a Distribution Planning and Analysis Engineer, upgrading the power grid. She plans to work towards her Professional Engineering License while pursuing her interest in Cyber Security for the Smart Grid.

Kevin Nilsen will graduate as a Photonic Science and Engineer in May of 2023. He plans to continue his studies by pursuing a PhD in Optics starting in the Fall semester of 2023.

REFERENCES

1. Qi, Yuankai, et al. "Object-and-action aware model for visual language navigation." *European Conference on Computer Vision*. Springer, Cham, 2020 https://link.springer.com/chapter/10.1007/978-3-030-58607-2_18#citeas.

2. Anderson, Peter, et al. "Sim-to-real transfer for vision-and-language navigation." *Conference on Robot Learning*. PMLR, 2021 https://arxiv.org/pdf/1904.04195.pdf.

3. Tan, Hao, Licheng Yu, and Mohit Bansal. "Learning to navigate unseen environments: Back translation with environmental dropout." *arXiv preprint arXiv:1904.04195* (2019) https://arxiv.org/pdf/1904.04195.pdf.

4. Zhu, Wanrong, et al. "Diagnosing vision-and-language navigation: What really matters." *arXiv preprint arXiv:2103.16561* (2021) https://arxiv.org/abs/2103.16561.

5. Upadhyay J, Rawat A, Deb D. Multiple Drone Navigation and Formation Using Selective Target Tracking-Based Computer Vision. Electronics. 2021; 10(17):2125. https://doi.org/10.3390/electronics10172125

6. Lineykin, Simon. "Basic Perception." Manual Mobile Robots and Multi-Robot Systems - Motion-Planning, Communication, and Swarming. Chichester, UK: John Wiley & Sons, 2020. 1–1. Web.