ROB498: Capstone Final Report

By

4 Big Fans

Kelvin Cui (1005337324)

Mingshi Chi (1004881096)

Yannis (Yiming) He (1004769707)

Leo Li (1004906943)

github.com/yAya-yns/4bf_drone

Design Philosophy

The following 4 leadership principles form the bedrock of our decision-making process:

1. Solid teamwork and trust: The team has a strong foundation of trust, having worked together for over 3 years on various projects.
2. Learning from experiences: The team's core value is continuously learning from their past experiences, improving, and growing from setbacks.
3. Adaptability and resilience: The team members are prepared to bounce back from challenges and view setbacks as opportunities to innovate and enhance their projects.
4. Perseverance and technical expertise: The team's unique combination of technical ability, teamwork, and determination sets them apart, driving them toward success.

We are excited to present our cutting-edge robotics team, composed of four highly skilled individuals with a passion for innovation and a proven track record of success. As enthusiasts in the field, we have been working together for over three years, developing a strong foundation of trust and collaboration that has allowed us to bring many ideas to life.

Our team members bring a combined total of over eight years of experience in both industry and research, boasting a diverse set of skills in robotics that we have honed throughout our academic and professional journeys. We pride ourselves on our ability to transform our past experiences into valuable learning opportunities that fuel our growth and drive us forward.

Our resilience and adaptability set us apart from the competition. We have faced numerous setbacks in our projects, such as major crashes and motor malfunctions. Yet, we have emerged stronger from each challenge by implementing innovative improvements to our platform and testing processes, like adding foam padding to our chassis and optimizing our wiring systems.

At the heart of our team's success is our in-house software-in-the-loop simulator, developed over weeks of dedicated effort. This cutting-edge simulation tool, built-in Gazebo and integrated with Q Ground Control, allows us to test our algorithms in a safe environment before deployment, reducing the need for physical testing while accelerating our progress towards project milestones.

We believe that our unique blend of technical expertise, teamwork, adaptability, and perseverance makes us an exceptional candidate for your investment. As we continue to push the boundaries of robotics, we are confident that our unwavering commitment to excellence will deliver outstanding results and drive us toward success in our endeavors.

Background

Autonomous drones, a remarkable subset of unmanned aerial vehicles (UAVs), have captured the imagination of enthusiasts and professionals alike with their ability to operate without any human intervention. These cutting-edge flying machines are revolutionizing various industries, including aerial photography, agriculture, construction, and even search and rescue operations, by performing tasks with greater precision and efficiency than their human-controlled counterparts.

At the heart of an autonomous drone is a sophisticated system of sensors, cameras, and advanced algorithms that enable it to navigate and perform tasks independently. These drones are designed to analyze their environment, make decisions, and adapt to changing conditions in real time, all without the need for a human pilot.

One of the key features that set autonomous drones apart is their ability to hover, maneuver, and navigate with exceptional stability and accuracy, even in the absence of human intervention. This is achieved through a combination of cutting-edge hardware, such as motors, propellers, and electronic speed controllers (ESCs), and intelligent software algorithms that ensure smooth and precise flight. Moreover, the integration of GPS and advanced positioning systems allow these drones to maintain their position and follow predetermined flight paths with exceptional accuracy, while also adapting to any unforeseen obstacles or changes in their environment.

In summary, autonomous drones represent a groundbreaking innovation that has the potential to transform the way we approach various tasks and industries. By offering unparalleled versatility, precision, and efficiency, these self-piloted aerial marvels are opening up a world of possibilities, and their potential for future advancements continues to soar.

Autonomous drones offer a game-changing solution for nuclear power plant inspection, significantly enhancing safety and efficiency while minimizing the risks associated with human intervention in hazardous environments. Equipped with high-resolution cameras, LiDAR sensors, and radiation detectors, these advanced UAVs can safely navigate the complex structures of power plants, capturing detailed images and generating accurate 3D maps of the facilities. By autonomously detecting anomalies, signs of wear, and potential safety hazards in real-time, autonomous drones enable operators to make informed decisions on maintenance and repair efforts, ensuring the continued reliable and safe operation of nuclear power plants.

Here are some technical features that specifically allow our drone to fit customers' needs for nuclear power plant inspection:

- Our drones are equipped with high-resolution cameras that can capture detailed images and video footage of the power plant components, structures, and equipment.
- Autonomous Navigation allows our drones to navigate through the power plant and avoid obstacles.
- In addition, drones can collect and transmit real-time data to a central location for analysis. Which can help identify potential issues early on and optimize maintenance.
- Finally, our Drones can be operated remotely from a control center if encounter special

circumstances, allowing inspectors to access hard-to-reach areas of the power plant without exposing themselves to radiation or other hazards.

Overall, these technological features make drone-based inspections of nuclear power plants more efficient, accurate, and safe.

Firmware Setup

The drone's foundation is a lightweight carbon fiber frame, specifically the Minion Mini H-Quad, which is ideally suited for racing drones, providing a perfect balance between strength and maneuverability. At the core of the drone's propulsion system, we have four Spedix ES30 HV 30A ESCs paired with four RotorGeeks 7075 Series 2206 2350 KV motors. The motors are fitted with 5-inch three-blade propellers, ensuring smooth and stable flight characteristics.

The drone's flight control is managed by the Pixhawk 4 Mini system, while the Nvidia Jetson Nano handles high-level planning and decision-making tasks. A Sony IMX219 RGB monocular camera captures high-quality visual data for environmental perception, while an Intel RealSense T265 provides visual odometry, allowing the drone to accurately estimate its position within its surroundings. Additionally, the drone utilizes a Teraranger Evo for altitude sensing and an IMU within the Pixhawk system for position odometry.

Our drone operates on a robust software platform, which includes the QGroundControl and PX4-Autopilot packages for simulation purposes. These tools enable thorough testing of algorithms and flight dynamics in a controlled, virtual environment. The drone's primary power source is a 4-cell LiPo battery, ensuring sufficient energy to complete its tasks efficiently.

The software architecture is built upon ROS 1 Melodic and is based on Python 2.7, providing a flexible and powerful framework for our drone's various functions. Additionally, the drone interfaces with QGroundControl, allowing seamless communication with ground-based systems and providing a comprehensive solution for monitoring and controlling the UAV's performance in real time.

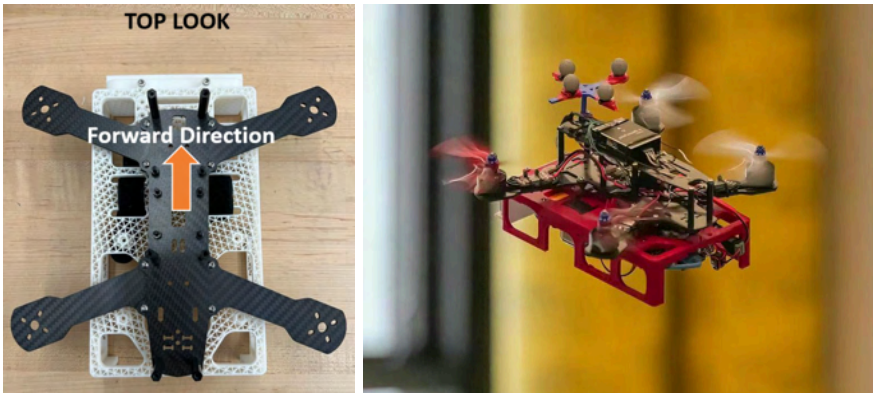

Figure 1: Image from our drone (named Daniel).

Design Objectives and Results

Challenge Task 2: Station-Keeping

Inspection drones are often required to station-keep or maintain a stable hover at a fixed altitude over an area of interest in order to carry out prolonged observation. During an inspection of a nuclear facility, for example, stationkeeping may be necessary to carefully measure radiation levels in a specific location.

Thus, the following precision objectives were set:

Parameter	Description	Value
θ_p	Primary heading goal (desired $\pm$ maximum deviation).	$0^\circ \pm 5^\circ$
r_p	Primary horizontal position goal (maximum deviation).	15 cm
d_p	Primary altitude goal (desired $\pm$ maximum deviation).	150 cm $\pm$ 10 cm
τ	Flight test duration (i.e., total hover time).	30 s

Table 1: Test objectives, specified by the maximum allowable deviation from the initial pose at the start of the test

The Pixhawk system, responsible for processing sensor data to compute the drone's location, initially encountered an issue due to conflicting inputs from the RealSense visual odometry and the Pixhawk IMU. The RealSense odometry's default configuration had its z-axis pointing downwards, which was mistakenly inverted by our team, leading to inconsistent data fusion within the Pixhawk system. As a result, when the drone attempted to reduce its altitude, the flight controller caused it to fly higher instead, forming a positive feedback loop and leading to a severe crash.

Following the reassembly and calibration of the sensors, our drone successfully reached a waypoint set at 1.5 meters. However, during stabilization, the drone's altitude exhibited fluctuations, which nearly jeopardized the desired precision, as shown in Figure 2. To address this issue, we integrated an additional LiDAR sensor pointing toward the ground for accurate altitude sensing. This enhancement provided an extra data source for the sensor fusion process, ensuring robust accuracy and improved stability during flight.

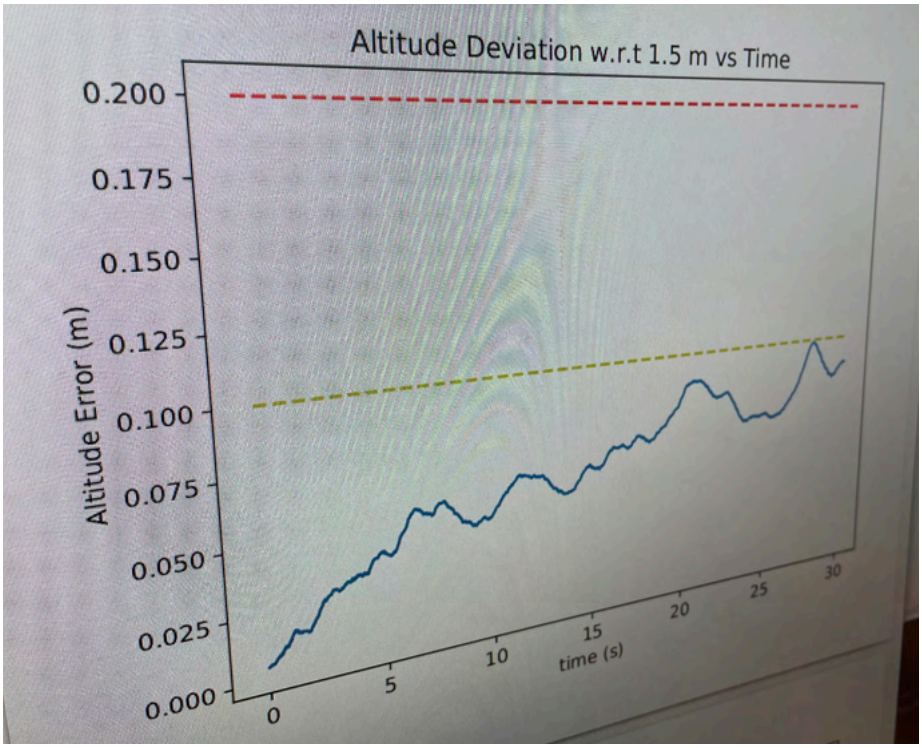

Figure 2: Almost went out of bounds due to the fluctuation

Challenge Task 3: Waypoint Navigation

A waypoint is an intermediate 'stop' along a route or line of travel. Inspection drones must typically visit a series of waypoints to collect all of their observations, following a predefined path. These waypoints can be considered artificial "highways in the sky."

Figure 3 provides an overview of the waypoint navigation task. As shown in the figure, there will be seven waypoints in total. Each waypoint is defined by a tuple of 3D coordinates (in the Vicon reference frame) and a radius value; for this task, the radius is 35 cm. Hence, each waypoint is defined by a spherical volume. The assessment of this challenge is time sensitive and was scored using:

$$S = q\left(1 - \frac{1}{7}\max\left(0, \frac{\tau - 60}{60}\right)\right) \times 1\%$$

Where τ is the total flight time, measured from the moment the TEST message is sent to the drone until the last waypoint in S is visited

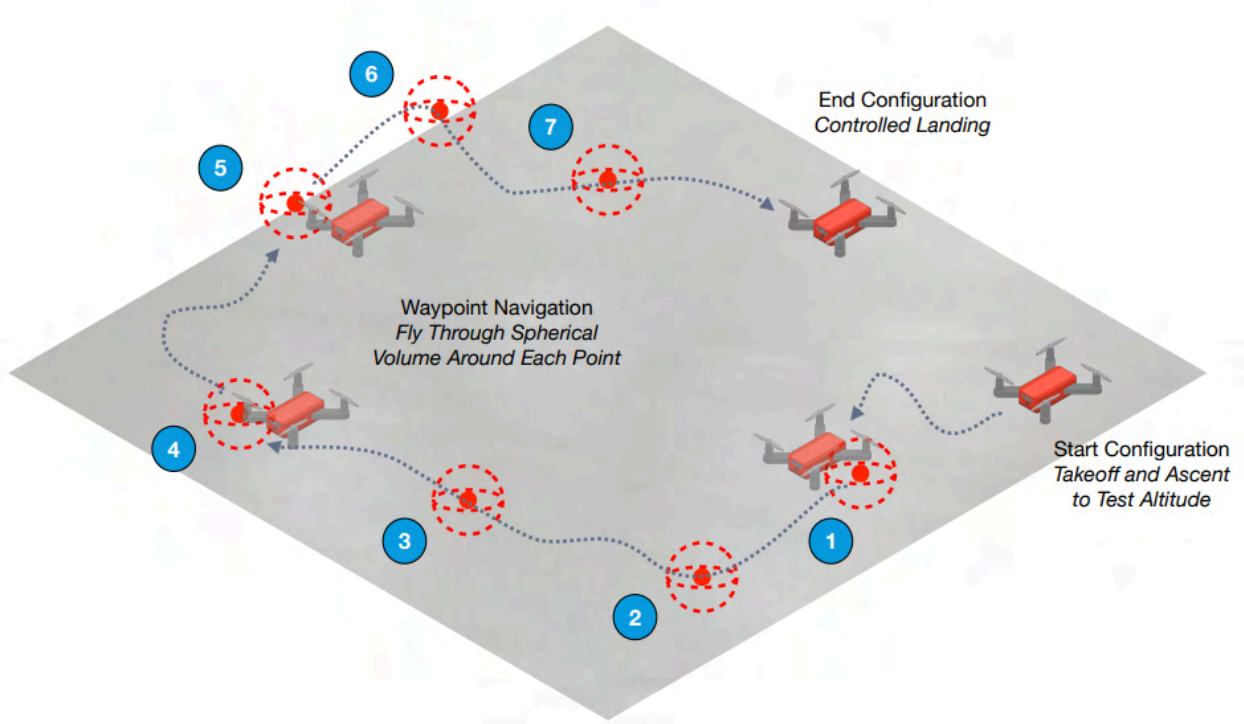

Figure 3. During the flight test, the drone will take off under manual control, stabilize at altitude, and then fly to a series of waypoints in a specified order (sequence).

The process of testing is as follows:

1. **LAUNCH:** Receive a launch command and ascend to desired altitude and hover. When the launch command is received, the drone should take off and ascend to the desired altitude, and then hover in place until the next command is received.

2. **TEST:** When the test command is received, the drone should subscribe to the waypoints topic to receive the list of waypoints that need to be visited. The current position of the drone in Vicon Frame is also published. The drone should fly to each waypoint in sequence.

3. **LAND:** After the task is complete or the maximum allowable time is reached, a land command will be sent, and the drone should land. Once the drone has visited all the waypoints or the maximum allowable time is reached, it should receive a land command to land safely. The drone should descend to the ground and land in a safe location.

Approach:

After crashing and burning a set of motors and observing that our drone was not stable during position hold mode, we re-soldered a power distribution board, ESCs, and motors and reassembled the drone. Then, we recalibrated and reset Pixhawk parameters from a generic quadcopter to a 5” rotor quadcopter and calibrated the ESCs. We also modified maximum

velocity, acceleration, and jerk variables which immensely improved our stability and flight trajectory.

The first thing needed was to be able to receive waypoints from the testing team from a rostopic with a series of goal waypoints in the Vicon Frame. Once we receive this series of points, we begin the navigation procedure, only changing the goal waypoint from Vicon Frame to Local Frame before sending the set_position call if the Vicon boolean variable is True. The Vicon to Local frame transform was calculated as follows:

1. Get the Vicon to Drone Transform from the Vicon Publisher
2. Create a Local frame to Drone transform using the local pose from MAVROS
3. Premultiply the Vicon to drone transform with the inverted local to drone transform to obtain the Vicon to local transform.

The transform is updated whenever a new set of Vicon transforms and local poses are received. This constant update allows us to correct for any drift we see in the local frame, rather than setting it once at launch.

For each waypoint when Vicon is enabled, we transform the waypoint from the Vicon frame to local poses, allowing us to continually correct for drift and command the drone to fly to the correct waypoints in Vicon frame.

Once the waypoints are received, the navigation begins. We start with an empty waypoints queue when hovering at 1.5m. Once the queue is no longer empty, we begin navigating to each waypoint. Once the current drone position is close enough to the current waypoint and the queue is not empty, pop the next waypoint from the front of the queue and set that as the new goal. Closeness to a point is the Euclidean distance between the current position and the current waypoint it is trying to reach. We compare this against a distance threshold to conclude if it is close enough.

We tested this method with dummy waypoints and forced the height to be only 0.5m so that in the event of unexpected behavior, the drop from 0.5m will not damage the drone. However, we did notice that the motors' sound was different than other teams and that our motors would get extremely hot even when flying for a short amount of time. We were extremely keen on the smell of the motors after burning the first set and changing the motors again. This time, flight sounds were normal and the motors stopped getting extremely hot.

We successfully ran both tests without Vicon in the end and obtained an average score of 92% in Challenge Task 3. Before we tested for grading, we tested with dummy waypoints to perfect parameters such as wiggle radius and desired velocity of the drone. Because of this testing, we were able to successfully test ct3 without any major setbacks.

Challenge Task 4: Obstacle Avoidance

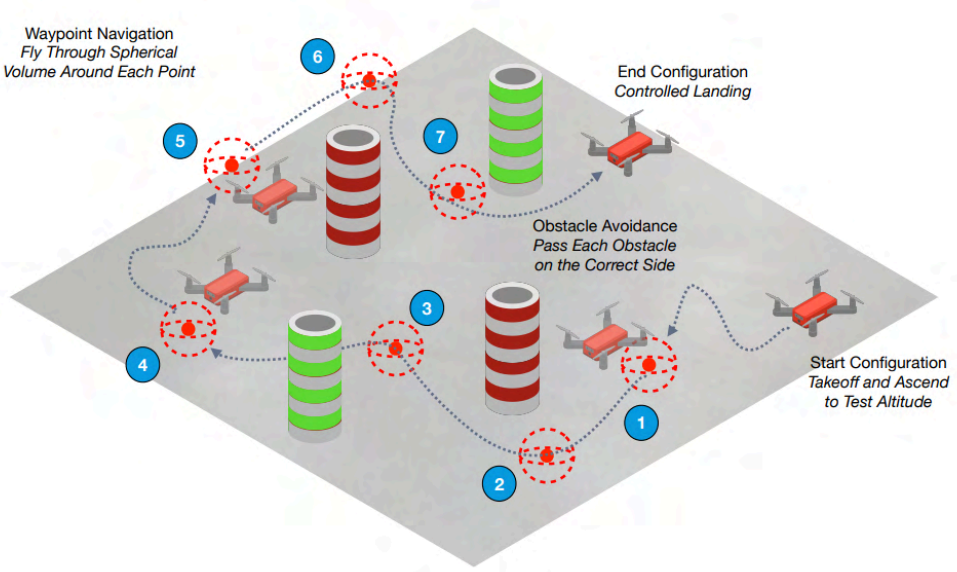

Figure 4: During the flight test, you will take off under manual control (or autonomously), stabilize at altitude, and then fly to a series of waypoints in a specified order (sequence) while avoiding obstacles en route. You must fly past each obstacle on the correct side (red – pass on the left, green – pass on the right)

Challenge Task 4 involved waypoint navigation while avoiding obstacles. To navigate between waypoints while avoiding obstacles, a combination of techniques such as obstacle detection, path planning, and obstacle avoidance can be used. The task involved visiting seven waypoints in total, with four obstacles that were not predetermined and had to be avoided on the fly in a particular direction depending on the color of the obstacle as shown in Figure 4. Red obstacles are avoided on the left and green obstacles are avoided on the right only if the drone flies within a 1m radius of the obstacle. If obstacles are avoided in the wrong direction, a 15-second time penalty is added. The time limit for completing the task was set at 75 seconds with an upper limit of 135 seconds with a time penalty. The drone had to follow a three-part process of launching, navigation, and landing. The scoring is:

$$S_{\text{base}} = \frac{q}{7}\left(1 - \frac{1}{7}\max\left(0, \frac{\tau^* - 75}{60}\right)\right)$$

Where q is the number of waypoints reached in sequential order, and tau is the total flight time including penalty time from the moment the TEST service call is sent.

The process of testing is as follows:

4. **LAUNCH:** Receive a launch command and ascend to desired altitude and hover. When the launch command is received, the drone should take off and ascend to the desired altitude, and then hover in place until the next command is received.

5. **TEST:** When the test command is received, the drone should subscribe to the waypoints topic to receive the list of waypoints that need to be visited. The drone should also activate its obstacle detection system to detect any obstacles in its path.

6. **LAND:** After the task is complete or the maximum allowable time is reached, a land command will be sent, and the drone should land. Once the drone has visited all the waypoints or the maximum allowable time is reached, it should receive a land command to land safely. The drone should descend to the ground and land in a safe location.

Approach:

The general goal of our team was to reach as many waypoints as we could while avoiding obstacles with real-time detection. Although a popular approach was to map out the obstacles before completing the task, we believed the odometry measurements from RealSense were too unreliable for this method. This belief was based on our previous crashes that were caused by RealSense VIO issues. With possible odometry drift, our originally mapped obstacles would also drift and would risk crashes based on inaccurate estimates of obstacle location.

Our first approach was to complete CT4 as if it was CT3 and only perform the obstacle avoidance behavior once an obstacle was detected at a close enough range. One necessary change to waypoint navigation was to turn to face the next point

The obstacle avoidance behavior is to fly directly in front of the obstacle at a set distance to face it head-on, fly sideways, forward, and back sideways to make a half box. First, we must detect obstacles. To do this, we leveraged the stereo camera on Realsense to generate a depth image. First, we rectified and cropped the two stereo cameras using OpenCV functions. The stereo cameras were cropped to a narrow vertical FOV, since the pillars we were detecting were tall and unlikely to be below our drone. This cropped FOV saved on performance and allowed us to calculate a disparity map at 10hz.

An interesting issue we had to engineer around was the placement of the cameras. Typically, stereo matching occurs from the left frame to the right frame - however, the left camera was heavily offset from the drone center, resulting in a skewed perspective. Instead, we decided to flip the images upside down, compare the right frame to the left frame, and flip them back right-side up. This allowed us to get the disparity in the right camera frame, which was closer to the center of the drone.

Once the disparity map was generated, we handcrafted heuristics in order to detect if a pillar was within collision distance, and where it was in the frame as shown in Figure 5. We sorted the indexes of the pixels with the largest disparities, and if the median of all the n largest pixels were above a certain threshold, we activated an avoidance maneuver. We then found the average x pixel coordinate of those n largest pixels, which told us where horizontally the pixels were in the camera frame. In order to account for noisy detections, we placed a sliding detection window, requiring 5 consecutive detections before a detection flag was sent to the rest of the stack. This largely reduced the number of false detections. Using this information, we would either perform a left or right avoidance in order to maximize our chances of avoiding the pillar.

The n (number of pixels required) and threshold were hand-tuned in order to achieve the best performance.

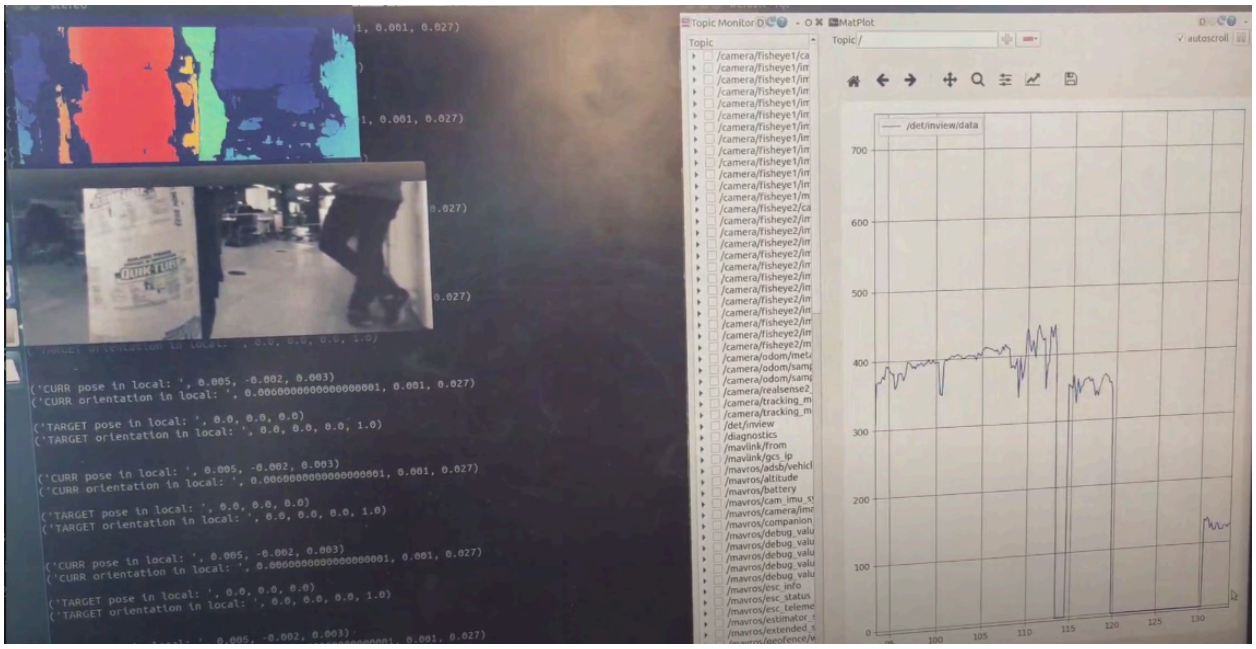

Figure 5. The Realsense rectified and disparity image visualization. The det/inview/data topic publishes the y-pixel location of the middle of the pillar.

However, after many tests and small crashes, it was found that the RealSense depth estimate was unreliable. In parallel, our attempt at using a purely vision-based obstacle detection approach is showing promising results, and thus we decided to pivot towards using only the camera to identify the location of the obstacle. During later testing, we truncated the top half of the image so the recognition isn't disrupted by Myhal's yellowish walls.

We first undistort the image using the camera calibration performed offline. The image is then passed through an HSV filter to obtain sections that closely mimic the yellow color of the pillars. After converting to greyscale and thresholding, a close and then an open morphology is performed to remove noisy sections that are within and outside of the thresholded sections. A border is padded followed by a Gaussian blur, both of which are to make the following canny edge detection more robust. Lastly, contours were drawn and unslanted bounding rectangles were identified on the largest contour. The width of the bounding rectangle is taken to be the width of the pillar in the image, which is then used to calculate the distance between the pillar and our drone. The number of pixels between the center of the bounding rectangle and the left side of the image is used to estimate the location of the pillar in the drone frame. An illustration of the image processing pipeline is shown in Figure 6.

Figure 6: 1. (top left) raw image. 2. (top right) post-undistortion. 3. (bottom left) after HSV filter. 4. (bottom right) the identified rectangle bounding box.

In order to obtain the relationship between the identified pixel width and the actual distance between the drone and the pillar, we decided to adopt a data-oriented approach where we obtain several measurements and perform a regression. Intuitively, the larger the width the smaller the distance, so the relationship should roughly follow a power law. After gathering several data points where we measure the width and the distance with a measuring tape, we entered the data into GoogleSheets and performed a power regression. The resulting formula for converting pixel width to distance in meters was:

$$d = 83.3w^{-0.939}$$

This result was reasonable since we'd expect the relationship to be similar to reciprocal. The raw data to plot Figure 7 is in Appendix.

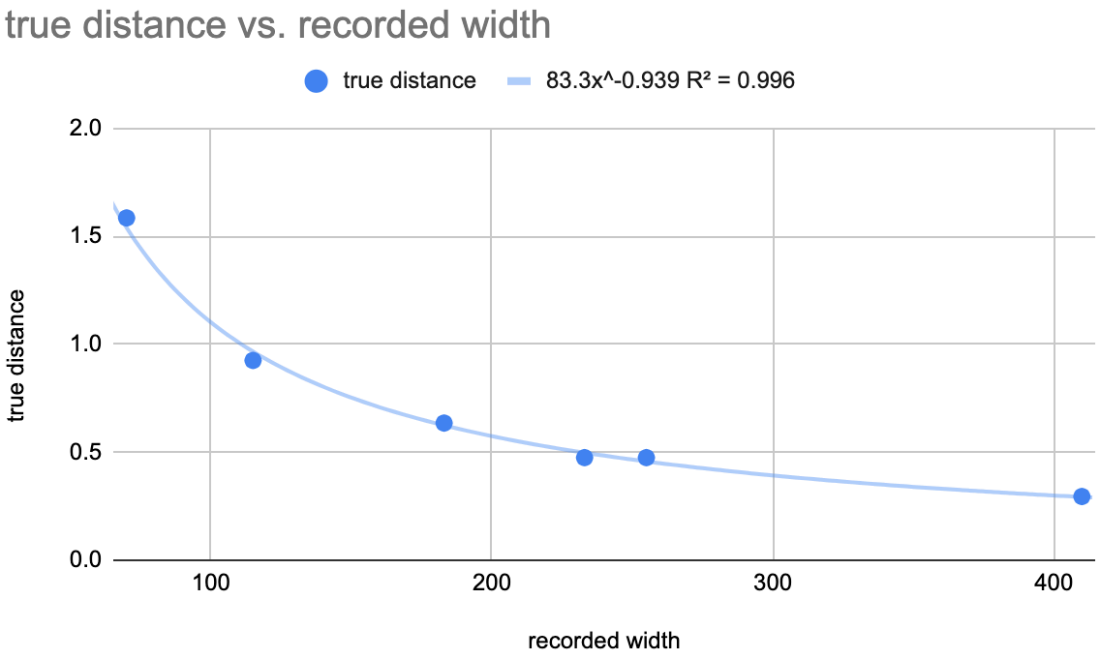

Figure 7: Pillar distance versus pixel width plot.

After confidently detecting and predicting the distance of an obstacle, we send the (x, y) location of the obstacle to the main navigation node. If the distance obstacle is within a danger threshold of the drone, obstacle avoidance is performed in a box-like pattern. The direction of the box is a variable we check against depending on the color type of the obstacle.

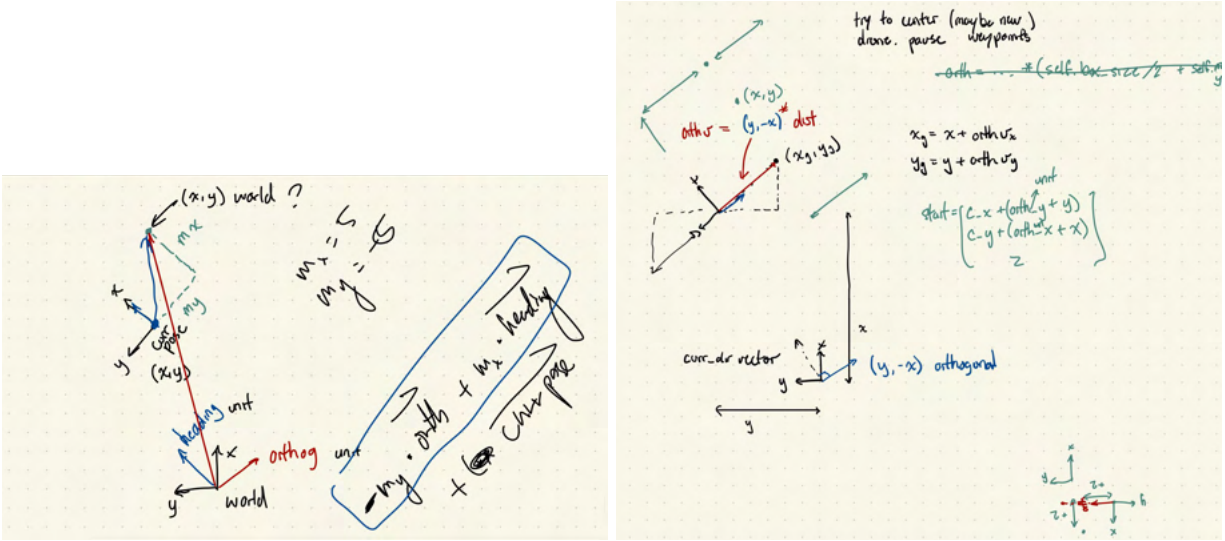

Figure 8. Rough math calculations for traversing a box around an obstacle.

The box traversal calculations are shown in Figure 8. The equations used to calculate each needed point in the local frame are shown below.

$$start = -O_y * \vec{orth} + O_x * \vec{heading} + \vec{curr\_pose}$$

$$P_1 = \begin{pmatrix} Start_x + \vec{Orth}_x \\ Start_y + \alpha * \vec{Orth}_y \\ Curr_z \end{pmatrix} \quad P_2 = \begin{pmatrix} P_{1,x} + \vec{Dir}_x \\ P_{1,xy} - \alpha * \vec{Dir}_y \\ Curr_z \end{pmatrix} \quad P_3 = \begin{pmatrix} P_{2,x} + \vec{Orth}_x \\ P_{2,xy} - \alpha * \vec{Orth}_y \\ Curr_z \end{pmatrix}$$

Where start is the pose and orientation of the drone facing the obstacle at a desired distance, α is the direction factor. If α = -1, the drone will avoid the left of the pillar, and if $\alpha = 1$ then the drone will avoid the right of the pillar. Dir is the direction vector the drone is currently facing, Curr is the current pose of the drone, and Orth is the orthogonal unit vector to the Dir vector. Both Dir and Curr are scaled in relation to the desired box size we want to avoid in. The location of the obstacle is given by the detection module in the drone frame where the (x,y) coordinates of the obstacle are x: forward metric distance away from the drone, and y: horizontal metric distance away from the drone.

To indicate that an avoidance procedure must be done, obstacle detection must be performed and be robust. The detection module must give a Geometry Message Point object where a -1 z value indicates no detection and a positive 1 indicates detection. We check against the Euclidean distance from the current location of the drone to the detected obstacle and if the distance is within a threshold, we perform obstacle avoidance and carry on trying to hit waypoints after the obstacle avoidance procedure is completed.

For this challenge, the architecture of the comm_node was refactored. Instead of pushing every waypoint and wiggle point into the waypoint list as soon as the TEST service call was called, we push the next waypoint to the front of the waypoint list when the current goal waypoint has been reached. In this way, we can push to the front of the waypoint queue and avoid disrupting the sequence of goal waypoint traversal when obstacle avoidance is necessary. The closeness function was also modified to check against goal orientation with the current orientation.

The state machine is as follows in Figure 9 below.

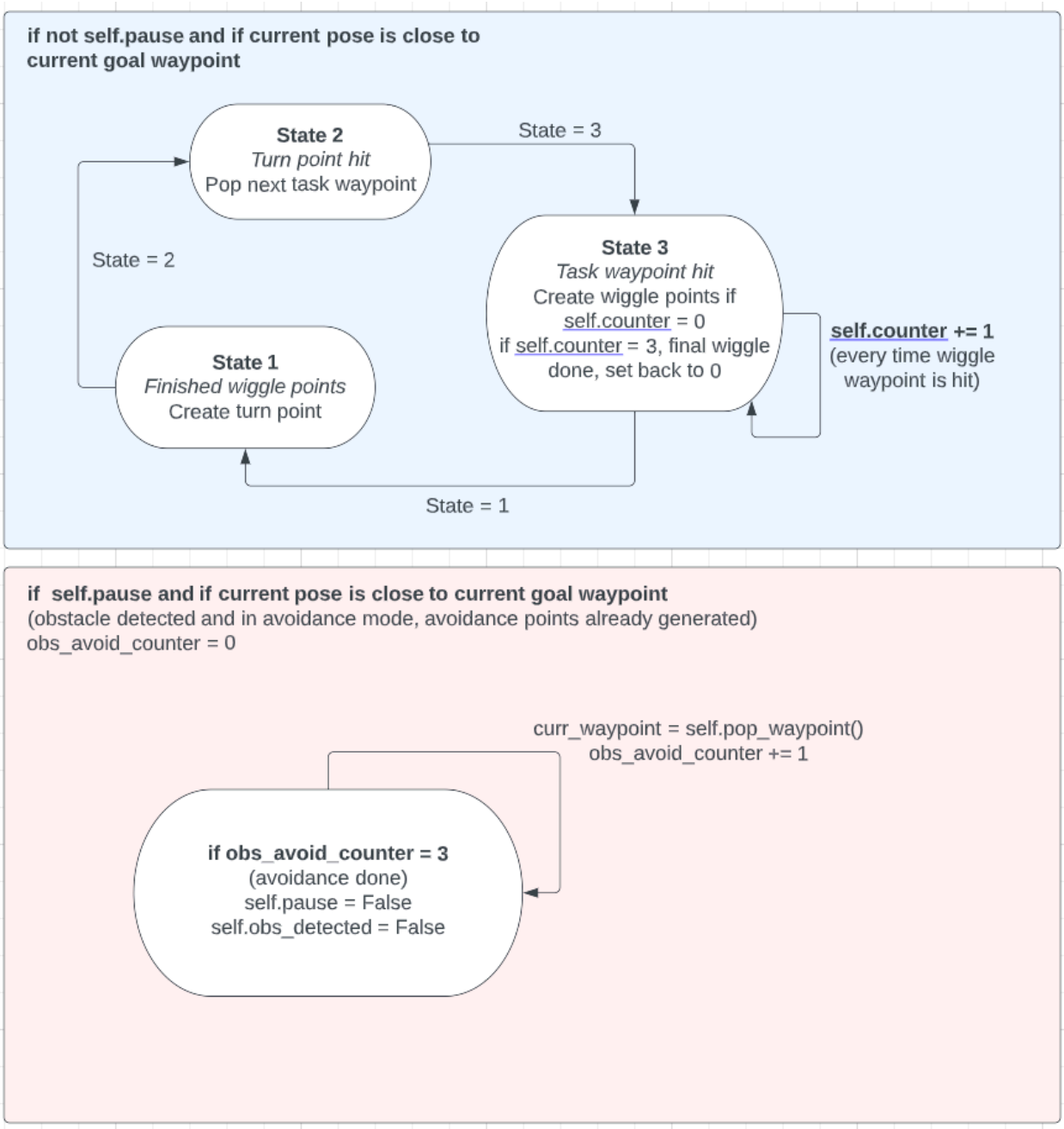

Figure 9. Finite state machine for flight logic

This state machine allows for two main modes: standard waypoint following, and obstacle avoidance.

In standard waypoint following mode, there are three different types of poses sent to the drone - target waypoints, turn points, and wigglepoints. Each set of poses starts with a turnpoint. The turnpoint keeps the drone in the same position and yaws the drone to point toward the next waypoint. This allows the camera to point in the direction of travel, allowing us to detect obstacles. The target waypoint is the commanded target from the published waypoint path, in the

Vicon frame. Finally, the wigglepoints are a set of 2 poses that vary the z position around the waypoint, in order to maximize the chances of hitting the waypoint in the Vicon frame, accounting for local frame drift, particularly in z.

However, if an obstacle is detected, our drone goes into obstacle avoidance mode. This mode can only be activated when the drone is flying towards a target waypoint and is disabled during turning and wiggling. The motivation for this is that yawing the drone often causes false detections due to the rapid camera motion, and wiggling only moves the drone in the z direction, which should not collide with an obstacle. If an obstacle is detected on the way to a target waypoint, that target waypoint is pushed, and a new obstacle avoidance path is generated to fly around the obstacle. Once the avoidance path is complete, the target waypoint is popped, and the drone continues its original state machine.

We relentlessly tested each component of our avoidance and navigation system prior to testing with one obstacle and known goal waypoints to test safely at a fixed height. Once we were confident in our system, we tested for evaluation.

During testing, we managed to reach 5/7 waypoints with one time penalty for avoiding the pillar in the wrong direction. We accounted for this as we decided to cut losses and only avoid obstacles in the shortest path and ignore the intended direction of avoidance. We calculated the desired score we wanted in the task and it was the most logical decision not to implement obstacle direction based on obstacle type as we had very limited time and were not confident that the detection system for the type of obstacle can be created and implemented within the timeframe. This approach however, proved satisfactory and resulted in a final score of 71% in CT4. We were not able to reach the final 2 waypoints because the odometry measurements got corrupted when turning too fast on the spot after the 5th waypoint and an emergency landing had to be done. Overall, we were happy with the results of this challenge task.

Challenge Task 5: Environment Sensing

After modification, the objective of challenge task 5 is to recognize the Yautja numbers from a picture taken. Since the task doesn't involve flying at all, it is relatively isolated from the rest of the code and can be developed separately. There will be 4 Yautja numbers printed on A4 papers stuck to a board horizontally. The only hardware components needed for the task are the jetson to run the code and the camera to capture the image. This constraint is later further relaxed since we didn't have a functioning camera. Our final implementation was to take a picture with our phone, send it through messenger and save it on our local laptop, and run a separate Python script to decode the number sequence.

Our algorithm consists of 3 main steps: unwrap, segment, and classify. The raw image (as shown in Figure 10.1) is first converted to greyscale, thresholded to leave only white sections, and then applied Gaussian blur. After applying a canny edge detector, contours are found and the one with the largest area is assumed to be the region where the numbers occur. The contour is approximated with a polygonal curve with 4 endpoints, and then the polygon is perspective transformed in Figure 10.2. After unwrapping, the remaining image is further reduced in size by finding the minimum unslanted bounding rectangle of the numbers. A simple division in width by 4 separates the individual numbers, as shown in Figure 10.3. Each image is further trimmed to remove edge artifacts. To fully classify the 6 Yautja numbers, only two pieces of information are

required: the total number of strips, and the number of horizontal strips. Numbers 2, 3, and 5 are returned if there are a total of 6, 2, and 5 strips respectively. If the number of strips is 4, a further step in classification is done where 6, 4, and 1 will be returned if there are 0, 1, and 2 horizontal strips respectively.

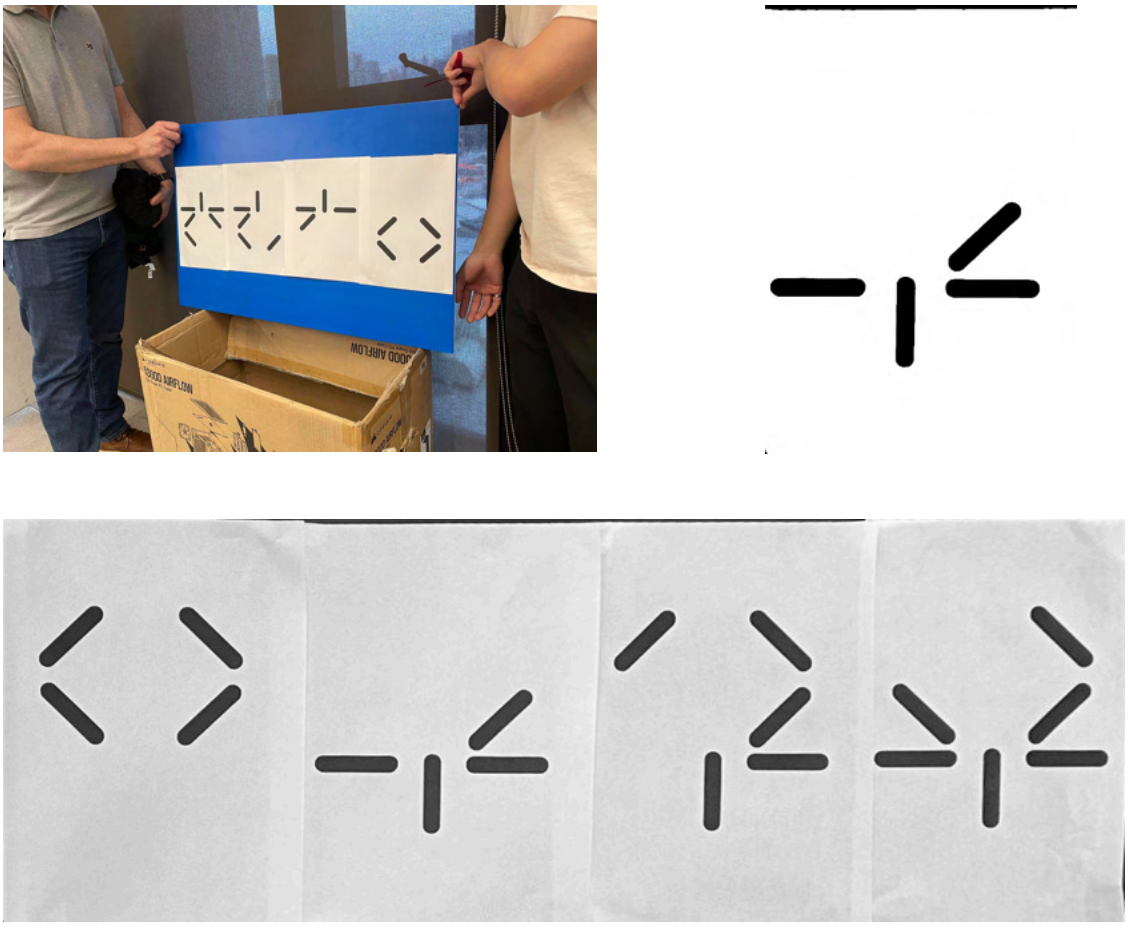

Figure 10: 1. (top left) raw image. 2. (bottom) unwrapped image. 3. (top right) individual numbers.

One advantage of the above classification scheme is that it is invariant for rotations of 180 degrees, which turned out to be critical when we are actually performing our algorithm. Since the number classification is based on the number of strips and the number of horizontal strips, both of which are invariant under 180 degrees rotations, so only the order in which the numbers are printed out needs to be reversed. The main challenge we encountered during the evaluation was that the perspective transforms sometimes transform the image in the wrong orientation, resulting in a failure in subsequent segmentation and classification. However, sometimes when the raw image is rotated 180 degrees this problem disappears. After failing our first evaluation, we added a try–except clause such that both the normal image and the rotated 180 degrees image will be run through, and whichever one succeeded will get its result printed out. Taking the test image at the exact same angle, the try–except clause ran successfully and we

obtained the correct answer. Such a swift response shows the robustness of our team's product and the flexibility with which our team was able to pivot, both of which ultimately contributed to our success.

Lessons Learned

To work effectively as a team under stressful conditions, it is important to maintain a level of professionalism and respect toward one another, while acknowledging the challenges presented by the situation. This includes the need to interact with each other on a daily basis without causing conflict, and recognizing the separation between work-related fatigue and personal friendships.

To ensure that the team's efforts are productive, it is also essential to create functions that can be tested separately, allowing for the identification of any errors or potential conflicts that could arise during testing. When making changes, it is important to test each change independently, to avoid confusion and potential errors.

Ensuring safety is an integral component of any drone operation, and prioritizing safety is especially critical when working under stressful conditions. Flying at the maximum goal height provides a significant buffer zone between the drone and any potential hazards, such as buildings or trees, and enables the pilot to maintain control of the drone in case of any unexpected circumstances. By prioritizing safety and working together as a team, it is possible to operate drones effectively and efficiently under even the most challenging and stressful conditions, ensuring that the objectives of the operation are met while minimizing risk to the team and the equipment.

In the event of a crash, it is essential to remain calm and composed, rather than panicking or becoming discouraged. By staying focused on the task at hand and working collaboratively as a team, it is possible to quickly regroup and address any issues that may arise. This includes conducting a thorough analysis of the cause of the crash, identifying any areas where the team can improve, and implementing appropriate measures to prevent similar incidents from happening in the future.

It is also essential to approach hardware checks with a critical eye, recognizing that even minor issues can quickly escalate into more significant problems if left unaddressed. Regularly checking hardware components and conducting maintenance as necessary helps ensure that the drone remains in optimal condition and reduces the risk of any unexpected malfunctions or breakdowns during operation.

Finally, it is crucial to check the hardware frequently to identify any potential issues before they escalate into more significant problems, minimizing the risk of errors and delays. By following these principles, teams can work together efficiently and effectively under stressful conditions.

Friendships were strained and friendships grew stronger throughout the process. The resilience and perseverance skills needed and forced to complete these challenges are extremely valuable skills to have outside of coursework. Whether it be working on a team in an industry or

doing collaborative research, there will constantly be situations where team members must work together under stressful conditions and maintain high morale and keep optimistic for the sake of the entire team. These skills were the most important lessons learned from this experience.

Team Information:

Mingshi Chi

Kelvin Cui

Yannis He

Leo Li

Acknowledgement:

We had the privilege of learning from a dedicated teaching team, for this graduate-level unmanned aerial vehicles (UAVs) course. The team was led by a professor and supported by four teaching assistants, each with unique expertise.

This was one of the most challenging courses we've taken at the University of Toronto, but it was also the most rewarding. The teaching team put in over 600 hours of work from preparation to completion. It was impressive to see how much effort they put into making sure we had a comprehensive understanding of the subject matter.

The course consisted of 13 groups, with four students in each group. Our teaching team was so attentive that they knew almost every student by name, which made me feel valued as a student. Each group had its own set of talents and strengths, and it was evident that everyone had put in a lot of hard work.
It's safe to say that this course helped me grow both academically and personally. We are grateful for the opportunity to have learned from such a dedicated teaching team, and I'm excited to see where my newfound knowledge will take me in the future.

No one loses their fingers. No fire. No explosion. We made it!

Figure 11: Engineering Science - Robotics 2T2 and their amazing teaching team for ROB498

Appendix

recorded width (pixel)	true distance (m)	recorded distance (cm)
233	0.475	31
410	0.295	13
183	0.635	47
255	0.475	31
115	0.925	76
70	1.585	142

Where:

$$true\ distance\ = \frac{recorded\ distance)+33/2}{100}$$

33 cm is the diameter of the pillar.