1. **Title:** Utilizing Object Detection and Haptic Feedback to Control the Position of a Pan-Tilt Camera

[redacted] - San Jose State University - 12.10.21

2. Background and Significance

2.1. Background

Among the many research initiatives at the NASA Ames Research Center, there has been recent interest in the development of electric flight vehicles (EFV) for civilian transport, which collectively is referred to as Urban Air Mobility (UAM). A number of companies have already designed such vehicles which, to name a few design parameters, range in scale, number of rotors, and rotor diameter. Amid the many milestones that must be achieved before commercial use, one includes the analysis of the rotor disk wake generated by the vehicle when in hover, which is measured at various rotor radii and azimuth angles around the vehicle [1]. Such an analysis is used to determine the minimum distance a civilian can stand in proximity to a hovering EFV without having to combat excessive wind conditions. Further, in addition to velocity analysis, vehicle acoustics must also be analyzed and used to develop quieter vehicles, especially if operated in residential areas. To aid in this development, NASA Ames is looking to test and collect measurements for a variety of possible EFV configurations, which to mention a few parameters, include varied rotor count, spacing, and scale. Once a database has been developed, this data can be used to help EFV companies improve upon current designs.

Figure 1. Shows a concept EFV used for everyday civilian transport, operating in a similar fashion to major car transport systems such as *Uber* or *Lyft* [2].

In order to characterize EFV rotor performance, a method for collecting velocity and noise profile measurements at various EFV rotor radii and azimuth angles is required. Further, the sensors used to collect these measurements need to be oriented perpendicular to the EFV at all times. NASA Armstrong Flight Research Center performed a similar set of tests involving velocity and noise measurements taken from a hovering Boeing CH-47D [1]. Data was collected at various rotor radii from the CH-47D by remotely operating a cart mounted with sensors to traverse forward or backwards along a linear track aligned radically with the CH-47D. In addition to testing at varied radial distances, the track ensured onboard sensors would maintain perpendicularity with the flight vehicle at all times. To adjust azimuth angle, because the CH-47D was not mounted to the ground, the pilot was required to manually controlled vehicle yaw relative to the carts linear track.

Future EFV's will have to undergo similar testing, and require the collection of generated rotor thrust and moment data, which requires the vehicle be fixed to the ground. As such, because a pilot will no longer be in control of vehicle yaw, a modification to the previous approach is required to collect vehicle azimuth angle data.

2.2. Significance

Given that future EFV tests will require the vehicle be fixed to the ground, a newly designed cart will have to incorporate azimuth angle adjustment in addition to traversing radically away from the EFV. Further, as stated previously, onboard sensors need to always be positioned perpendicular to the EFV, regardless of cart location. Cart positioning can be accomplished through remote teleoperation, navigating the carts location by visual inspection, or autonomous navigation. However, to reduce the amount of tasks the operator has to concentrate on, maintaining sensor perpendicularity with the EFV will be automated by use of camera based object detection tracking (Figure 3 below). This will allow the operator to focus solely on cart azimuth angle radial station positioning of the vehicle without having to worry about onboard sensor orientation. However, it is important that the operator be alerted if the autonomous tracking system loses sight of the EFV; this alert can be sent from cart to operator in a number of ways, such as haptic vibration, audio alert, or blinking light. Once alerted, the operator can stop cart motion until EFV tracking has stabilized.

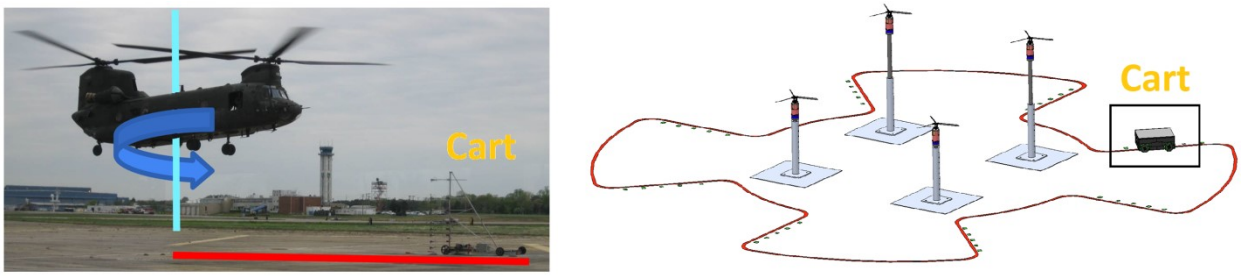

Figure 2. Left shows an image of a helicopter hovering near a cart instrumented with data collection sensors; note that in this situation, the cart was designed to only move along a linear track, requiring the flight vehicle to yaw position for collection at different angles around the vehicle [1]. Right shows a rendered design of a EFV (4 rotor configuration fixed to the ground), a

cart, and its ability to traverse anywhere around the EFV. Being autonomous, the cart could travel along unique paths such as the depicted red line shown above. Further, the vehicle could stop along this path (sample green markers) for data collection.

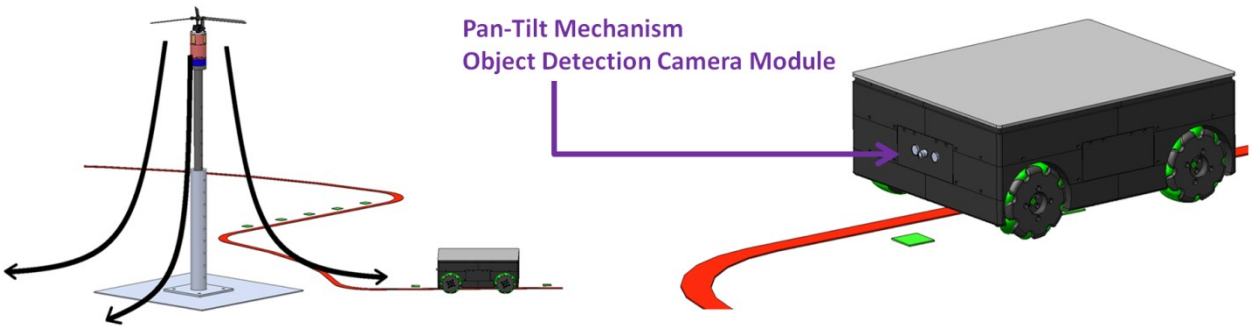

Figure 3. Left provides a visual of the downwash wind velocity generated by a rotor, which would be collected by instrumentation installed on the cart. Right shows a concept cart, along with an installed camera module required for both cart navigation and EFV object detection and tracking during cart movement.

To successfully implement a new cart design for EFV testing, it is necessary to understand system building blocks, starting with the core elements first (Pan-Tilt Mechanism Object Detection Camera Module). The following literature review is to deepen understanding of project components, including intuitive controller interface designs, typical sensors used for teleoperation control, real time object detection strategies, and techniques to optimize performance for reduced power mobile machines.

2.3. Literature Review

2.3.1. Handheld Controller Interface and Functionality for Vehicle Teleoperation

The ability of an operator to control and be situationally aware of a vehicle's motion from a distance is known as teleoperation [3], [4]. This definition can be broken into subcategories and defined as the ability to provide direct control of the vehicle and, if autonomous, the ability for an operator to supersede decisions made by vehicle intelligence [3]. The operator control interface must be intuitive, offering a suite of controls specifically tailored for ease in vehicle control and also useful feedback on the vehicles current conditions [5]. The most common type of teleoperated control input is by gimbaled joystick, allowing for planer vehicle motion [3], [5]. Other control types include touch screen displays, but in accordance with Luz et al., have been confirmed to be a disadvantage as they require the user to visually confirm the correct touch screen button has been pressed [5]. Motion and force input gloves offer a very different type of control, where changes in hand position and or gestures are used to command vehicle motion; however, Doisy et al. found that that based on trail testing, although initially intuitive, gesture control generally leads to frustration [4]. Further, it was reported that operator-head-tracking to control vehicle camera orientation provides an intuitive feeling of vehicle orientation, but often requires the coupled use of a joystick as the head tracking is primarily intended for identifying the space around a vehicle, and not for actual motion [4]. Overall, to the inexperienced user, control by hand or head tracking initially feels most intuitive; however, results demonstrate that to the trained operator, the gimbaled joystick is the preferred approach [4].

When line-of-sight visual between an operator and controlled vehicle is too far away or obstructed from view, live information from sensors mounted directly on the vehicle can be relayed back to the operator for additional control [5]. A common sensor is the use of a camera which has a live feed of what the vehicle *sees*, providing cockpit-like vehicle control; however, Rodriguez-Sedano et al. points out that live camera feeds are sometimes not sufficient as they can be obstructed by obstacles or demonstrate delayed latency [6]. In addition to numeric data that can be displayed visually, haptic feedback through motor vibration or by piezoelectric actuation can provide the user with direct sensory input, alerting the user directly that the vehicle has met a certain condition; Luz et al. successfully implemented haptic feedback to communicate vehicle traction state to the operator [5]. Although having either visual data from a Graphical User Interface (GUI) or haptic feedback is useful when used independently, Rodriguez-Sedano et al. suggest that the combined use of both inputs provides the operator with an increased level of surrounding environment vehicle awareness and control [6].

2.3.2. Vehicle Mounted Sensors

Robotic vehicles tend to have a variety of sensors on them to detect objects in their surrounding environment, which can in turn be used for either human teleoperated or autonomous closed loop feedback control. A commonly used sensor is *lidar*, short for *light-detection-and-ranging*, which is used to detect an objects position by measuring the time it takes for a pulse to be sent and again received. If the lidar sensor is spinning and at a constant velocity, lidar can be used to map a 360 degree surrounding space with high accuracy; this was successfully implemented by Malavazi et al. for autonomous mapping of an agricultural field [7].

Another position mapping device is through a Global-Positioning-System (GPS); although GPS is used for many navigation applications, it is generally only accurate to within one meter of a target and not accurate enough for precise teleoperated location mapping [7]. Stereo cameras, such as the *intel REALSENSE D435i*, can be used to calculate depth, mapping either the 2-D or 3-D location of an object in reference to the camera; this technique is robust when compared with non-camera based approaches [8], [9]. Alternative uses for the stereo camera is in mapping the location of the camera itself in reference to the cameras starting position. This is done by combining depth information with Inertial-Measurement-Unit (IMU) information, a technique successfully demonstrated by Hausamann et al. with the *intel REALSENSE T265* sensor. This sensor demonstrates robust behavior for slower speeds and smaller sized areas [10]. Through the combined use of the *intel* D435i and T265 sensors, the vehicle can map environmental terrain, save its starting location to memory for later recall, and mark the location of new objects within this mapped environment for future reference.

2.3.3.a. Object Tracking: Pre-trained Networks and Transfer Learning

To detect an object, *pre-trained networks* or *network transfer learning* can be used to identify patterns (features) across large labeled datasets (thousands if not millions of images) [11]. *Pre-trained networks* are those that have already be trained on many objects, making them usable out-of-box, assuming the object of interest is already within its trained library. In an effort to predict human age

based off many images of faces, Dagher and Barbara successfully produced high accuracy when using pre-trained networks (such as Google-nets) [11]. *Transfer learning* on the other hand allows for retraining of an existing network to identify new objects [11]. Although pre-trained networks as easier to use, if a desired object is not in the trained library, transfer learning can be used to alter existing architecture to detect new objects [12].

2.3.3.b. Object Tracking: Object Detection Types

Object detection is based off a deep learning technique called *Convolutional Neural Networks* (CNN), which is the backbone architecture of how image classification works [13]. In addition to image identification, position of the detected object within frame is performed through Regional-CNN's (R-CNN). That said, R-CNN's can be prone to low accuracy for smaller objects; however, Chen et al. validated that when existing R-CNN framework are expanded with additional layers, mean average precision can be increased by 29.8% when compared with traditional R-CNN architectures [14].

By comparison, the Mask R-CNN (M-RCNN) builds upon R-CNN positional framework, segmenting off the actual object within a given ROI, effectively *masking off* the objects specific geometry. Mask R-CNN has two types, *semantic segmentation* and *instance segmentation*. *Semantic segmentation* separates all detected segmented objects from the background, grouping all detected objects together and comparing against image backdrop; *instance segmentation* distinguishes each individual detected object from each other, including the background, and as such, is one of the more powerful techniques in use [15]. Under both types, Ruiz-Santaquitera et al. experienced relatively high accuracy when applying these techniques to segment off microscopic algae within images, but points out that the segmentation process is only as good as the initial ROI detection [15]. Further, although additional network layers of an M-RCNN will provide more detailed information on a detected model then R-CNN, the increased computation time for increased network layers will slow inference between detections, which can be an issue if real time detection is required by embedded devices which have slower processors and GPU's.

2.3.3.c Object Tracking: Object Detection on Embedded Devices (Real-Time Detection)

Although the use of R-CNN methods are effective at accurately classifying images and detecting objects, high amounts of input parameters can take long periods to compute, and consequently bottleneck real time detection performance [16], [17]. Further, in addition to algorithm efficiency, an additional obstacle to achieving real time detection is when trying to run these models on lower powered computers. R-CNN model architecture must be further reduced for faster processing time to achieve real time detection on embedded systems.

To improve real time performance on embedded systems, object detection algorithm architectures, such as *MobileNetV1* (MNV1) and *MobileNetV2* (MNV2), were developed to require less features for image classification; further as Barba-Guaman et al. points out, faster ROI position information has been made possible through object detection networks such as *Single-Shot-Detector-MNV1 (SSD-MNV1), Single-Shot-Detector-MNV2 (SSD-MNV2),* and *Pednet* to name a few, with minimal sacrifice in accuracy

[13], [16]. Improvement in these categories decreases computation time and thus inference time between detections [13]. In addition to algorithm optimization, NVIDIA has developed a line of small yet powerful mobile computers specifically designed for running multiple deep learning models in parallel while maintaining the required high frame-rates for real time inference; a few of their models include the NVIDIA TX2 and the NVIDIA Jetson model series [13], [17].

3. Objectives

The objective of this project is to design, construct, and test a pan-tilt camera mechanism used to track the location of a fixed or moving object. Pan-tilt object detection and tracking will be set manually via teleoperated controller or performed automatically. In both cases, successful object tracking will be relayed from pan-tilt mechanism to operator controller, vibrating in unique patterns to indicate tracking success. To evaluate manual pan-tilt tracking, the operator will use object detection feedback to manually adjust camera position until the fixed object is centered in the camera view window. To evaluate autonomous tracking, the pan-tilt camera mechanism will be analyzed for its ability to continuously track a fixed or moving object.

4. Methodologies

4.1. Approach

This project will apply academic knowledge and equations stemming from control theory, computer vision, and mechatronic circuit analysis. Control strategies such as PID-controllers and Linear-Quadratic-Regulars (LQR) will be considered to control pan-tilt motion. Once motor control has been established, a camera will be installed on the pan-tilt mechanism, and be used to track the location of an object. Tracking will be done through a Region-of-Interest Convolutional-Neural-Network (ROI-CNN) algorithm, which recognizes patterns within camera images. In addition to software development, physical prototypes of the handheld-controller and pan-tilt mechanism will require knowledge learned in mechatronic courses for selecting and calculating motor torque and power curves. Further, prototype designs will require circuit analysis techniques, including voltage division, op amp derivation, diode sizing, RC-filter electrical noise reduction, and power requirements. Aside from calculation and theory, a handful of skills in software, hardware, and electrical design will be required. In order to set pan-tilt position, skills in C++ and python will be leveraged to program microcontrollers and computers. As electrical problems arise, expertise in oscilloscopes, function-generators, and digital-multimeters will be utilized to troubleshoot issues. Lastly, to train the object detection algorithm, skills in Tensorflow, reading live image data, and deploying models on the NVIDIA Jetson Platform will be utilized.

4.2. Realistic Implementation Plan

Project objectives are broken into four overall sections. The first is in the development of the handheld-controller, which is used to commands new pan-tilt mechanism positions. In addition, this

handheld-controller must also receive object detection feedback in the form of haptic vibration. The controller will be developed by completing the following task list.

Step 1.1 Select controller interface buttons, including joystick for pan-tilt motor control and haptic vibration motors for object detection feedback.

Step 1.2 Determine power requirements to size the controller battery.

Step 1.3 Bench test handheld controller electrical components and document circuit schematics. Electrical components include computer, power source, power regulation, solid-state relays, BJT's, H-Bridges, etc.

Step 1.4 Manufacture the final circuit boards, either by hand soldering or printed-circuit-board (PCB).

Step 1.5 Develop multiple iterations of the handheld-controller interface using SOLIDWORKS.

Step 1.6 3D-print the final version of the controller housing and install all controller electrical components.

Step 1.7 Write a program used to verify that controller inputs and haptic motors are behaving as intended.

The next set of implementation steps is associated with pan-tilt mechanism development. The pan-tilt mechanism must be able to send and receive commands from the handheld controller and also detect when and where an object is located via live camera feed. Having this information, the mechanism must also actuator motors to point the pan-tilt mechanism camera at the object. This mechanism will be developed by completing the below task list.

Step 2.1 Determine desired camera speed, and use this information to design gears and identify the required pan-tilt motor torque.

Step 2.2 Determine motor power consumption to identify the appropriate power supply.

Step 2.3 Bench test pan-tilt mechanism electrical components and document circuit schematics. Electrical components include, but are not limited to, a computer, power regulation, motors, and H-Bridges.

Step 2.4 Manufacture the final circuit boards, either by hand soldering or PCB.

Step 2.5 Develop multiple iterations of the pan-tilt mechanism using SOLIDWORKS.

Step 2.6 3D-print the final version of the pan-tilt mechanism housing and install all controller electrical components.

Step 2.7 Develop motor control equations (experimentation with position, velocity, and general PID control) and adjust until desired pan-tilt motion is achieved.

Step 2.8 Write a program which uses these equations to receive wireless handheld-controller commands and subsequently use these commands to set pan-tilt position.

Step 2.9 Write a second program for automated pan-tilt object tracking.

The next set of steps is to train the object detection algorithm. This system must detect if an object is in an image taken from a live camera feed, and further, extract the location of where that object is within frame. This will be accomplished by completing the below task list.

Step 3.1 Collect object image data, which includes taking live video of the object, and separating the video into many images. This includes video taken of the object under different lighting conditions, time of day, etc.

Step 3.2 Prepare object image data by labeling images. This includes manually sifting through each image, and labeling the object location within each image.

Step 3.3 Use the prepared data to perform transfer learning on existing model architecture to train a R-CNN to detect the desired object within an image.

Step 3.4 Export the trained model for use on the NVIDIA Jetson computer, and write a program to read in camera images in real time. The trained model will then be used to detect object position within each image.

Finally, the last set of steps is to verify that developed subsystems are working together as intended. More specifically, this includes live object detection feedback relayed to the handheld-controller, automatic and manual pan-tilt motion, and real time object tracking.

Step 4.1 Record a video demonstrating manual control of the pan-tilt mechanism. When controlled manually, the operator will use object detection feedback to orient the camera such that the detected image is centered in the camera view window.

Step 4.2 Record a video demonstrating automatic control of the pan-tilt mechanism. When controlled automatically, the operator will monitor the pan-tilt mechanisms motion as it continuously tracks an object. Further, positioning will be recorded and plotted with the expectation of showing minimal peak overshoot and minimal rise time when tracking objects.

4.3. Cost Estimates

The list below outlines the hardware, tools, computing power, and software required for this project. The total estimated cost is between $500 and $1500, depending on component choices. Note that all required purchasing will be done during ME295A to avoid delays during ME295B.

Table 1. Lists the hardware, tools, computing power, and software required for this project, including cost breakdown.

Category	Needed by:	Resource	Cost/Access	Description
Hardware	3/1/2022	Mechatronic Parts	$400 to $1000	This will include all electrical components required to construct the handheld-controller and pan-tilt mechanism. For the handheld-controller, this includes at minimum a joystick, haptic motors, Raspberry Pi computer, wireless transmitter, battery, and power regulators. For the pan-tilt mechanism, this includes two motors for pan-tilt motion, wireless transmitter, camera for object detection, power source, and NVIDIA Jetson computer for mobile real time object detection.
	3/1/2022	Raw Materials	$100 to 500	Most of the items used in this project involve 3D printing (and require the cost for machine maintenance and refilling plastic filament). Some items will use CNC or laser cutter (and require aluminum, wood, or plastic materials).
Tools	3/1/2022	Machine Shop	Have Access	Work machine shop will provides access to 3D printers, CNC, and laser cutting machines. These will be used heavily throughout this project.
	Immediate	Electrical Equipment	Have Access	To troubleshoot electrical problems, an oscilloscope and digital-multimeter will be used throughout the project.
Computing	Immediate	Computer with GPU	Have Access	When training a R-CNN to detect objects from images, a computer with dedicated GPU is required to minimize training time. A laptop will initially be used, but if the models becomes sufficiently large, a powerful desktop with dedicated GPU will be used to reduce training time.
Software	Immediate	SOLIDWORKS	Have Access	SOLIDWORKS will be used for all handheld-controller and pan-tilt mechanism modeling.
	8/1/2022	Tensorflow	Have Access	Tensorflow GPU will be used to train the object detection model.

4.4. Risks and Contingencies

This section outlines the risks and subsequent contingency plans that might be required when implementing the handheld-controller and pan-tilt mechanism prototypes, as well as when training the object detector. The first risk is in developing a robust transfer learning ROI-CNN model for object detection. Object detection requires image data collection and image labeling of the desired object; if training is not sufficient, the model will infer false detections, and cause the camera to track incorrect objects. Although in many cases this issue can be resolved by collecting more training data under more conditions, test results might be insufficient for robust object detection (or simply too time consuming to manually label an excessive amount of images). As an alternative, pretrained networks are easier to use as they have already been *pretrained* on a variety of objects. In the event that transfer learning produces less than satisfactory results, pretrained networks can be used as alternative detection, allowing for the project to proceed.

The second risk is not being able to achieve real time object detection. Even if model inference is very accurate, the time between image detections might be longer than desired on slower running mobile computers (such as the NVIDIA Jetson computer). As a correction for this risk, a more powerful computer could be used to increase inference time, keeping the project on track.

5. Deliverables

This project has four deliverables which will be completed over the ME295A and ME295B courses. The first and second deliverables involve the development of handheld-controller and pan-tilt mechanism prototypes. The third and fourth deliverables involve object detection training and video demonstration of the working system. The first two deliverables will be accomplished during ME295A and second two deliverables during ME295B. Details on each are described below.

Deliverable 1: The first deliverable will be a 3-D printed prototype including documented files of the handheld-controller used to command and monitor pan-tilt mechanism position. Deliverable specifics are broken into three categories: mechanical, electrical, and programming. 2-D drawings of all mechanical components will be provided. Further, electrical schematics will be generated for handheld-controller computer, joystick inputs, haptic motor vibration, wireless transmission, and power. Lastly, programming source code to read controller inputs used to command pan-tilt positions and receive object detection feedback will be provided.

Deliverable 2: The second deliverable will be a 3-D printed prototype and documented files of the pan-tilt mechanism used to pivot and subsequently track a fixed or moving object. Like the first deliverable, this is also broken into mechanical, electrical, and programming components. 2-D drawings of all mechanical components will be provided. Further, electrical schematics will be generated for pan-tilt mechanism microcontroller, camera, wireless transmission, and power connections. Lastly, programming source code to read object detection information, relay information to handheld-controller, and to set pan-tilt camera position will be provided.

Deliverable 3: The third deliverable will be a written protocol on the steps to train an object detection algorithm capable of real time detection. For project testing, the object itself is yet to be determined, but will most likely be a common object (such as a baseball).

Deliverable 4: The fourth deliverable will be a video demonstration of developed prototypes used to manually or automatically track objects seen from a live camera feed.

6. Timeline

The below timeline reflects all the steps required to completed each deliverable. Although the project could be completed in two semesters, some work will occur over the 2021-2022 Winterim and Summer time periods.

Table 2. Shows a gantt chart of how the overall project will be achieved during courses ME295A and ME295B. The spring 2022 semester will focus on hardware design and development. The fall 2022 semester will focus on object detection training and testing, as well as overall project demonstration. In addition to academic semesters, some work will be performed during the summer 2022.

	ME295A					Summer		ME295B				
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
D1: Prototype of Handheld-Controller (HC)	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
Task 1.1: Identify HC button selection.												
Task 1.2: Determine power requirements.												
Task 1.3: Conduct PT circuit bench tests and build wire diagrams.												
Task 1.4: Manufacture electrical chips.												
Task 1.5: Create CAD model of HC.												
Task 1.6: 3D print HC housing and assemble.												
Task 1.7: Program controller inputs/feedback.												
D2: Prototype of Pan-Tilt Mechanism (PTM)												
Task 2.1: Camera speed, gear ratio, size motor.												
Task 2.2: Select appropriate power supply.												
Task 2.3: Conduct PTM circuit bench tests and build wire diagrams.												
Task 2.4: Manufacture electrical chips.												
Task 2.5: Create CAD model of PTM.												
Task 2.6: 3D print PTM housing and assemble.												
Task 2.7: Control equation development.												
Task 2.8: Program manual wireless HC PTM tracking.												
Task 2.9: Program for automated PTM tracking.												
D3: Object Detection Training and Protocol												
Task 3.1: Collect image data.												
Task 3.2: Label image data.												
Task 3.3: Train object detector and develop model.												
Task 3.4: Test real time detection on Jetson.												
D4: Video of Prototype Demonstration												
Task 4.1: Demonstrate manual control.												
Task 4.2: Demonstrate autonomous control.												

A few items not listed in the above chart include the writing of the report at the end of each semester, academic meetings with advisors, and presentation of results at the end of ME295B. Although the above tasks will be executed during the stated time, additional tasks beyond the scope of this project will be attempted if time allows. Note that two months were left blank at the end the ME295B semester to account of unexpected delays.

7. References

[1] M. J. Silva and R. Riser, "CH-47D tandem rotor outwash survey," *Annu. Forum Proc. - AHS Int.*, vol. 4, pp. 2917–2937, 2011.

[2] "Eve Urban Air Mobility finds Launch Partner in Halo - Aviation Today." https://www.aviationtoday.com/2021/06/01/eve-urban-air-mobility-finds-launch-partner-halo/ (accessed Nov. 28, 2021).

[3] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Auton. Robots*, vol. 11, no. 1, pp. 9–18, 2001.

[4] G. Doisy, A. Ronen, and Y. Edan, "Comparison of three different techniques for camera and motion control of a teleoperated robot," *Appl. Ergon.*, vol. 58, pp. 527–534, 2017.

[5] R. Luz, J. Corujeira, L. Grisoni, F. Giraud, J. L. Silva, and R. Ventura, "On the Use of Haptic Tablets for UGV Teleoperation in Unstructured Environments: System Design and Evaluation," *IEEE Access*, vol. 7, pp. 95443–95454, 2019.

[6] F. J. Rodríguez-Sedano, M. A. Conde, P. Ponsa, L. M. Muñoz, and C. Fernández-Llamas, "Design and evaluation of a graphical user interface for facilitating expert knowledge transfer: a teleoperation case study," *Univers. Access Inf. Soc.*, vol. 18, no. 3, pp. 431–442, 2019.

[7] F. B. P. Malavazi, R. Guyonneau, J. B. Fasquel, S. Lagrange, and F. Mercier, "LiDAR-only based navigation algorithm for an autonomous agricultural robot," *Comput. Electron. Agric.*, vol. 154, no. February, pp. 71–79, 2018.

[8] Z. Shan, R. Li, and S. Schwertfeger, "RGBD-Inertial Trajectory Estimation and Mapping for Ground Robots," *Sensors*, vol. 19, pp. 1–29, 2019.

[9] B. Nenchoo and S. Tantrairatn, "Real-Time 3D UAV Pose Estimation by Visualization," *Proceedings*, vol. 39, pp. 1–5, 2020.

[10] P. Hausamann, C. B. Sinnott, M. Daumer, and P. R. MacNeilage, "Evaluation of the Intel RealSense T265 for tracking natural human head motion," *Sci. Rep.*, vol. 11, no. 1, pp. 1–12, 2021.

[11] I. Dagher and D. Barbara, "Facial age estimation using pre-trained CNN and transfer learning," *Multimed. Tools Appl.*, vol. 80, no. 13, pp. 20369–20380, 2021.

[12] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Using ImageNet Pretrained Networks," *IEEE Trans. Geosci. Remote Sens. Lett.*, vol. 13, no. 1, pp. 105–109, 2016.

[13] L. Barba-Guaman, J. E. Naranjo, and A. Ortiz, "Deep learning framework for vehicle and pedestrian detection in rural roads on an embedded GPU," *Electron.*, vol. 9, no. 4, pp. 1–17, 2020.

[14] C. C. B, M. Liu, O. Tuzel, and J. Xiao, "R-CNN for Small Object Detection," *Springer Link*, vol. 1, pp. 214–230, 2017.

[15] J. Ruiz-Santaquiteria, G. Bueno, O. Deniz, N. Vallez, and G. Cristobal, "Semantic versus instance segmentation in microscopic algae detection," *Eng. Appl. Artif. Intell.*, vol. 87, no. April 2019, p.

103271, 2020.

[16] W. Liu *et al.*, “SSD: Single shot multibox detector,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[17] Z. Zhao, Z. Zhang, X. Xu, Y. Xu, H. Yan, and L. Zhang, “A lightweight object detection network for real-time detection of driver handheld call on embedded devices,” *Comput. Intell. Neurosci.*, vol. 2020, 2020.