# Olivia Ma

## Engineering Portfolio

My name is Olivia Ma, and I am a Robotics Engineering Master’s student at the University of Michigan - Ann Arbor. I will be graduating in May 2026, and I am looking for full-time positions in robot/motor control. I have a background in hardware design and autonomous aircraft, and am passionate about solving unique, highly-dynamic controls problems and optimizing for energy efficiency.

Previously, I have been a GNC Intern at Reliable Robotics (2025), Hardware Engineering Co-op at Amazon Robotics (2024), and Motor Controls Intern at Stellantis (2023). I was also an Undergraduate Research Assistant at the A2Sys Lab working on cheap, precision hexacopter control. I was also Structures team lead for Michigan Autonomous Aerial Vehicles (MAAV) student project team, leading hardware design for various VTOL vehicles. This portfolio will take you through my recent projects.

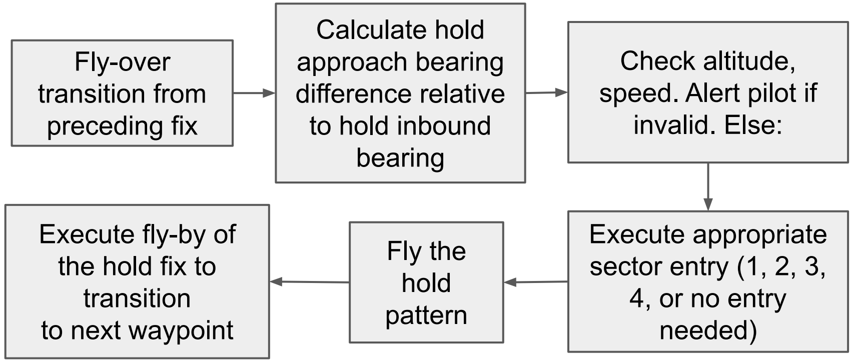
Feel free to contact me with questions or comments at oliviiam1@gmail.com.

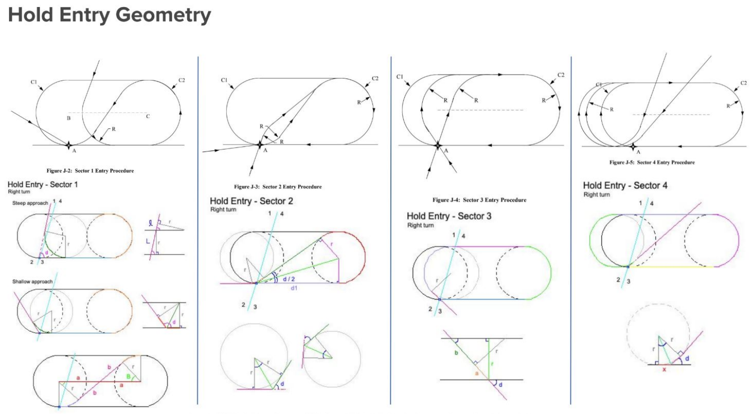
# Reliable Robotics GNC Intern Guidance Project

I was tasked with generating trajectories for autonomous holding patterns. A flight hold is a racetrack-shaped flight path planes might take while waiting (e.g. if a runway is backed up for landing).

I did background research to understand how the FAA defines a hold and what circumstances a pilot may enter a hold under. I worked within the guidance stack to generate and store hold information and set up the pipeline to insert hold legs into the flight plan. I added a state machine to appropriately handle all entry and exit conditions and edge cases to ensure the aircraft would only initiate viable holds at safe altitudes and speeds.

I delivered the hold leg trajectory generation capability, along with a suite of unit tests and integration tests that could be run nightly to check stability as the code base continued to evolve.


Fly-over transition from preceding fix
Calculate hold approach bearing difference relative to hold inbound bearing
Check altitude, speed. Alert pilot if invalid. Else:
Execute appropriate sector entry (1, 2, 3, 4, or no entry needed)
Fly the hold pattern
Execute fly-by of the hold fix to transition to next waypoint
Rough flow chart for hold trajectory generation and execution



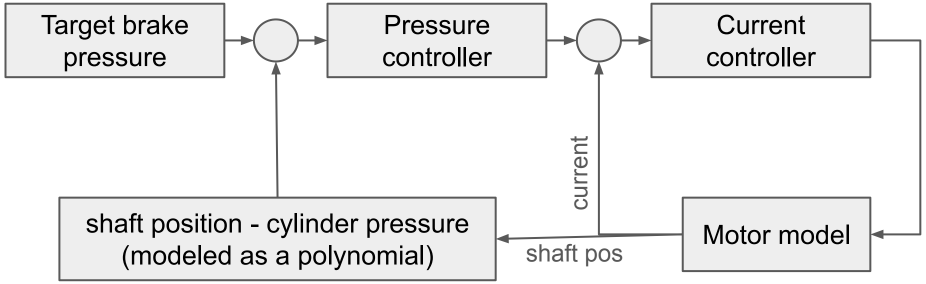
Calculations I made to break hold entries and racetracks into supported geometries

# Reliable Robotics GNC Intern Modeling Project

For this project, my goal was to model the braking system and control surface systems in C++, to be integrated into the simulation.

I started from first principles, with motor electrical and mechanical dynamics. I used existing data to model inertial, viscous friction, and gear compliance effects. I modeled gear compliance as a mechanical damping. I then modeled mechanical dynamics of the aircraft systems driven by the motor. The aircraft system model requires a certain input torque to achieve its desired state, which is passed as an input to the motor model as an external torque on the output shaft. I applied PID controllers to the systems (assuming perfect sensor data).

I unit tested my models by passing step inputs to the motor model and system models inputs.



Block diagram of brake pressure model + control loop

Electrical dynamics for motor model

$$\dot{I} = \frac{V_a - K_e\omega_{motor} - IR}{L}$$

| | |
|---|---|
| $V_a$ = applied voltage | I = current |
| $K_e$ = back emf const | R = resistance |
| $\omega$ = motor speed | L = inductance |

Mechanical dynamics for motor model

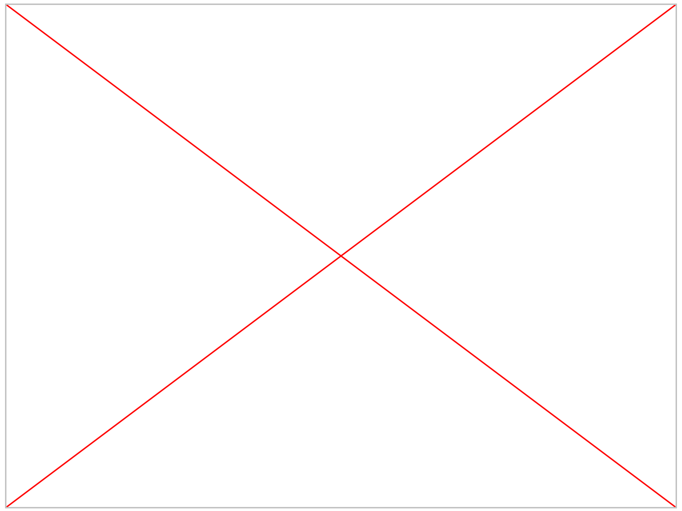
$$J_{total}\dot{\omega}_{load} = \tau_{app} - \tau_{load} - \tau_{motor} - \tau_{ext}$$

J = moment of inertia
$\omega$ = motor speed
$\tau$ = torque (applied by motor, load friction, motor friction, external)

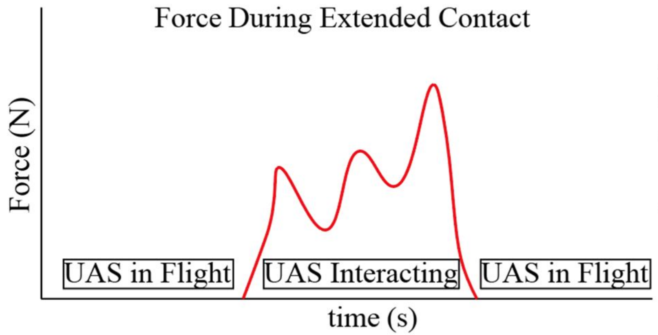
# A2Sys Pogo Drone Switching Controller + Tuning

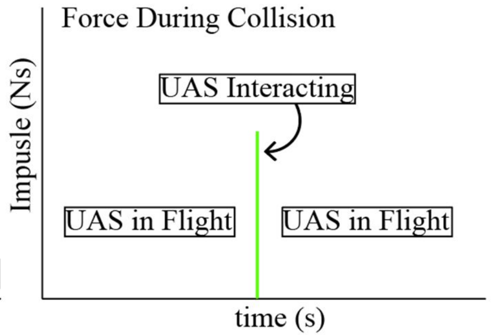
The goal of this project is to demonstrate a controlled, targeted bouncing technique for the application of collecting treetop samples in the Amazon rainforest. To this end, we developed a controller to safely recover an underactuated system under collision.

UAS typically struggle with meaningful physical interactions with environments due to the coupling between the UAS and the environmental dynamics, as external forces will disrupt the UAS's attempts to keep itself stable. This effect is doubly observed when attempting to interact with flexible/unpredictable targets, such as tree branches.

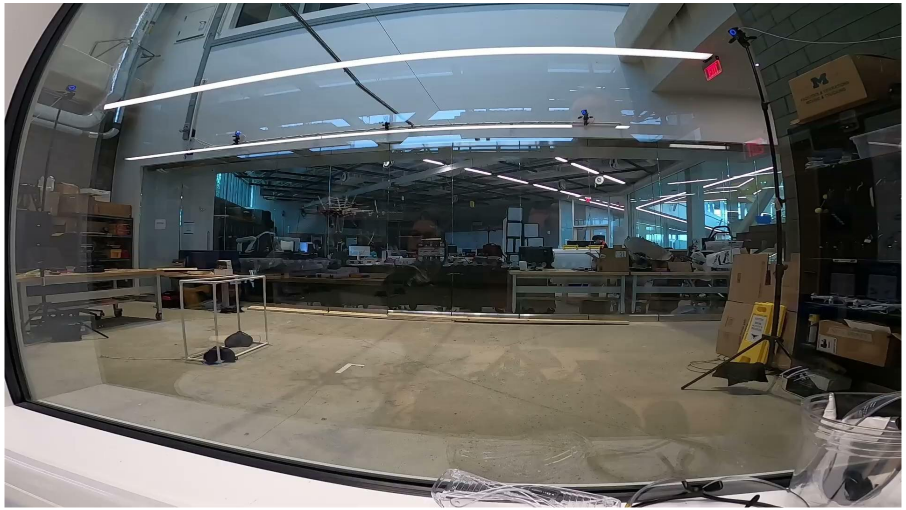

Switching control with bounce on "virtual ground" for safety

The idea is that planned collisions will enable UAS to maintain stability through environmental interactions via short, controllable impacts (a la Marc Raibert's hopping robots).



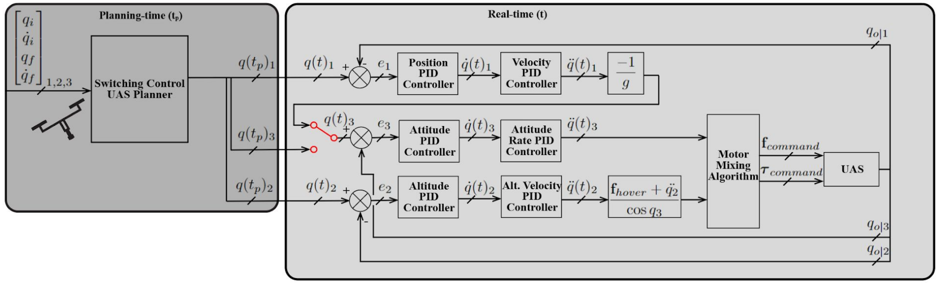


We first proceeded with a velocity-matching approach, which enabled collision recovery by controlling the attitude to align the pogo stick with the UAS's CoM velocity vector while matching a desired horizontal velocity. We identified a contour of recoverable flights (varying initial altitude, horizontal velocity) in simulation and validated experimentally. I conducted flight tests for PID tuning and data collection (paper here).



Velocity matching control with bounce

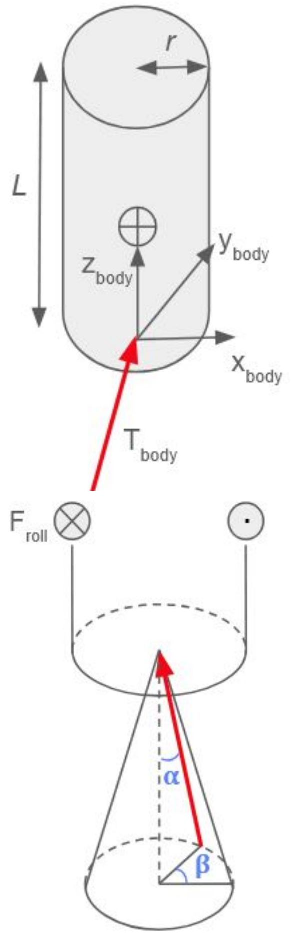
We then moved to a switching controller, for which I contributed to the development of the state machine architecture in C, and led flight testing to tune the cascaded PID controller. I also isolated mismatch between our simulation and flight test time delays created by the command → messaging/mocap → motor spin up pipeline, leading to poor performance of optimally generated trajectories.



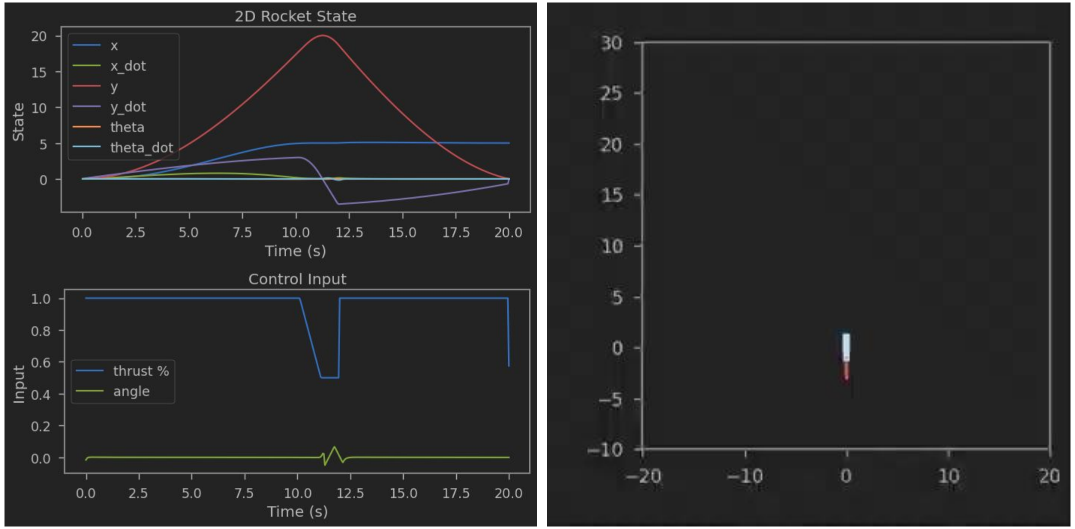
Cascaded PID Switching Controller Architecture

# MERO Self-Landing TVC Rocket Sim + Control (WIP)

My friends and I started this project team to challenge ourselves to design and test a self-landing rocket over the span of our 1-year Masters. My role is to simulate and design a control system for the rocket to be able to perform a hop – fly up to a designated waypoint, and perform a controlled landing. The rocket will be controlled using one gimbaled, throttlable engine, and a series of ducted fans for roll control.

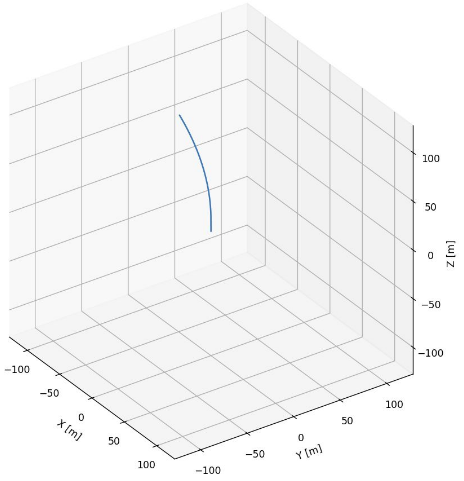


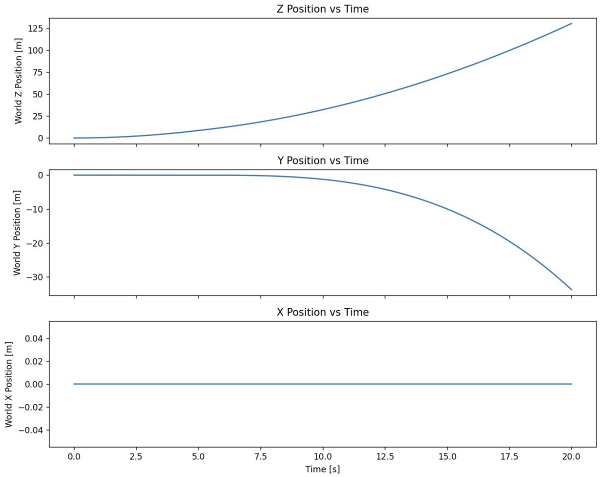
Given an engine's thrust curve, I used CasADi to generate optimal trajectories and evaluate feasibility. My optimization included the rocket dynamics, max gimbal angle and rate, and throttle limits of 50-100%. Through this method, we adjusted our apogee height and landing location requirements to match our engine's capability.



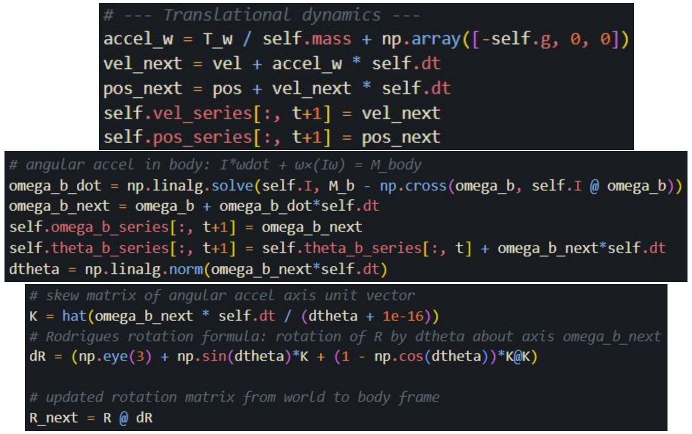
2D Rocket State and Control Input Plotted Over 20 sec. Burn Time (+Video)

I also developed a simple 3D simulation in Python, which models the rocket as a cylinder with uniform mass. I derived the dynamics from first principles, then integrated the rocket state forward through time using forward Euler. I validated my simulation by giving it open-loop control inputs and matching the results to my expectations based on hand calculations.





3D Simulation with Open Loop Control (max thrust for 0.5s at t=0, then a pitch-over command for 0.5s at t=5)



```
# --- Translational dynamics ---
accel_w = T_w / self.mass + np.array([-self.g, 0, 0])
vel_next = vel + accel_w * self.dt
pos_next = pos + vel_next * self.dt
self.vel_series[:, t+1] = vel_next
self.pos_series[:, t+1] = pos_next
```

```
# angular accel in body: I*ωdot + ω×(Iω) = M_body
omega_b_dot = np.linalg.solve(self.I, M_b - np.cross(omega_b, self.I @ omega_b))
omega_b_next = omega_b + omega_b_dot*self.dt
self.omega_b_series[:, t+1] = omega_b_next
self.theta_b_series[:, t+1] = self.theta_b_series[:, t] + omega_b_next*self.dt
dtheta = np.linalg.norm(omega_b_next*self.dt)
```

```
# skew matrix of angular accel axis unit vector
K = hat(omega_b_next * self.dt / (dtheta + 1e-16))
# Rodrigues rotation formula: rotation of R by dtheta about axis omega_b_next
dR = (np.eye(3) + np.sin(dtheta)*K + (1 - np.cos(dtheta))*K@K)

# updated rotation matrix from world to body frame
R_next = R @ dR
```

Forward Euler Step for Translational and Rotational Dynamics

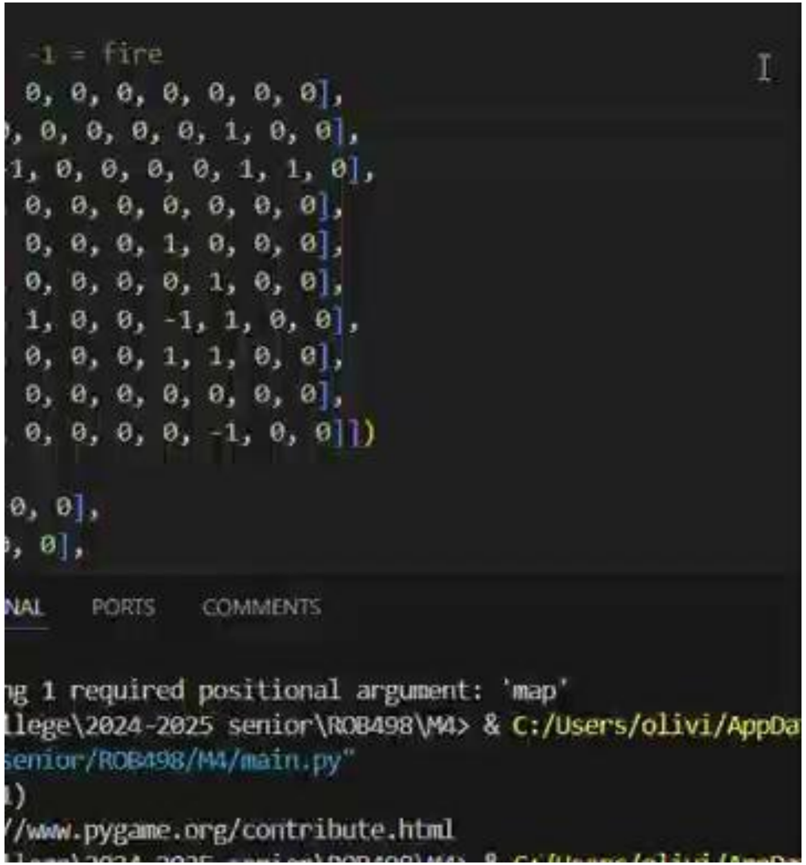
My next steps are to improve the 3DOF simulation (making it 6DOF, adding throttle rate, CoM changes as fuel is consumed, air resistance, more realistic geometry/mass distribution) and improve on the 6DOF cascaded controller.

# ROB 498 (Multi-Robot Control) Final Project

My goal was to explore coordination across multi-robot teams by extending the homogeneous team coordination covered in class to heterogeneous teams – that is, teams comprised of distinct types of specialized robots. I constructed a forest-fire problem, in which teams of Surveyor robots (fixed-wing UAS) and Firefighter robots (quadrotors) were deployed to identify and extinguish fires.

I implemented a forest fire simulation in Python, representing the workspace as a 10x10 grid, with random cells of fire (increasing odds for cells adjacent to fire). I implemented the Surveyor robot using an information-gain exploration method with A* path-planning to efficiently map the evolving environment, then used a distributed optimization algorithm to assign Firefighter robots to the nearest, largest fires.

I implemented the drone swarm as a distributed system, meaning there is no central coordinator. Firefighters could exchange info with Surveyors and other Firefighters within their communication radius to maintain updated maps and solve the Multiple Traveling Salesman problem among themselves to find the optimal fire-assignment.



```
 -1 = fire
 0, 0, 0, 0, 0, 0, 0],
, 0, 0, 0, 0, 1, 0, 0],
1, 0, 0, 0, 0, 1, 1, 0],
 0, 0, 0, 0, 0, 0, 0],
 0, 0, 0, 1, 0, 0, 0],
 0, 0, 0, 0, 1, 0, 0],
 1, 0, 0, -1, 1, 0, 0],
 0, 0, 0, 1, 1, 0, 0],
 0, 0, 0, 0, 0, 0, 0],
 0, 0, 0, 0, -1, 0, 0]])

0, 0],
, 0],
```

NAL    PORTS    COMMENTS

```
ng 1 required positional argument: 'map'
llege\2024-2025 senior\ROB498\M4> & C:/Users/olivi/AppDa
senior/ROB498/M4/main.py"
)
//www.pygame.org/contribute.html
```

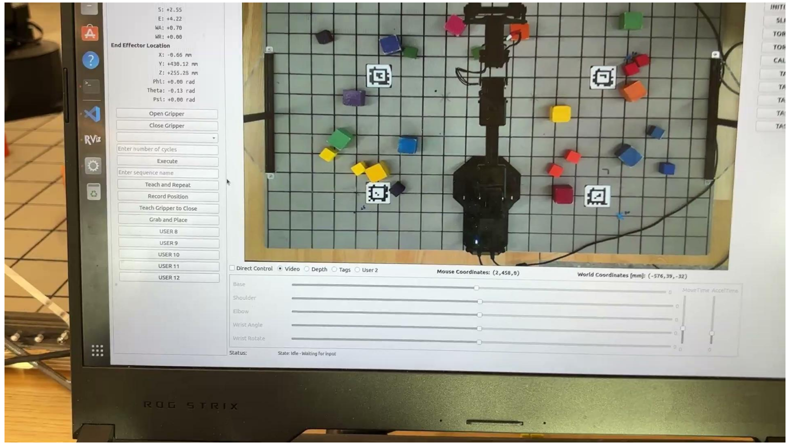
Recording of 1 Surveyor robot mapping the environment

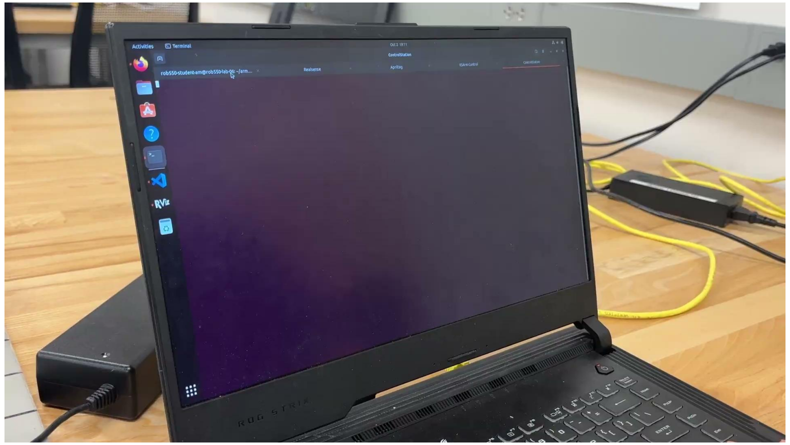
# ROB 550 (Armlab) Final Project

Through this project I practiced my skills using forward/inverse kinematics, coordinate transforms, and computer vision on a 5DOF actuated arm.

My teammate and I used ROS2 and OpenCV's built-in functionalities to find the depth camera's internal and external matrices, which we used to normalize the workspace image and convert coordinates between the camera frame and world frame. We also used OpenCV to filter and segment the image by color and depth values in order to identify small and large blocks, with their centers and orientations.

I calculated the forward and inverse kinematics analytically. I used the [Product of Exponentials method]() for FK by identifying the end effector's matrix in the arm's home pose, then finding the screw vectors generated by each joint. Then, by leveraging the exponential representation of a rotation about axis $\omega$ by angle $\theta$ ($e^{[\omega]\theta}$), I could find the end effector's state given arbitrary joint angles. I computed the inverse kinematics by considering the arm as a 2-link sub-arm and 2-axis wrist joint, with the yaw of the arm determined by the base joint. This greatly simplifies the IK, but I had to take extra care to address degenerate solutions (elbow up vs elbow down).





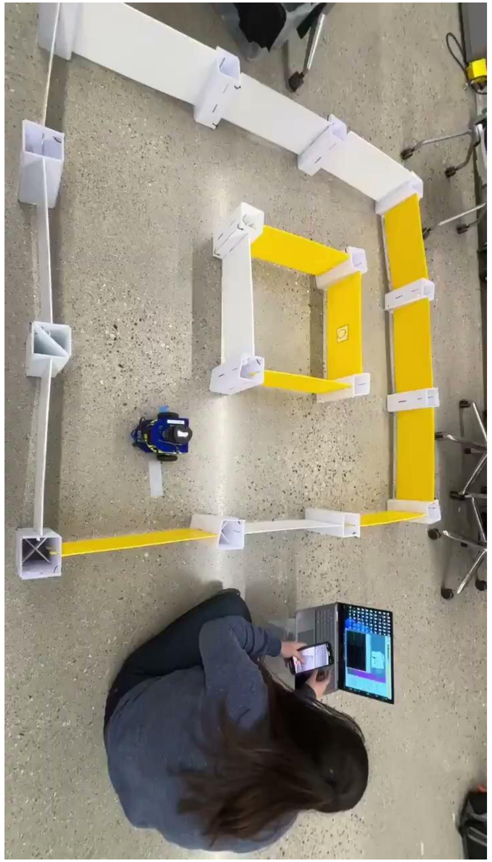
Demonstration of "[Click and Place]()" and [Block Detector]()

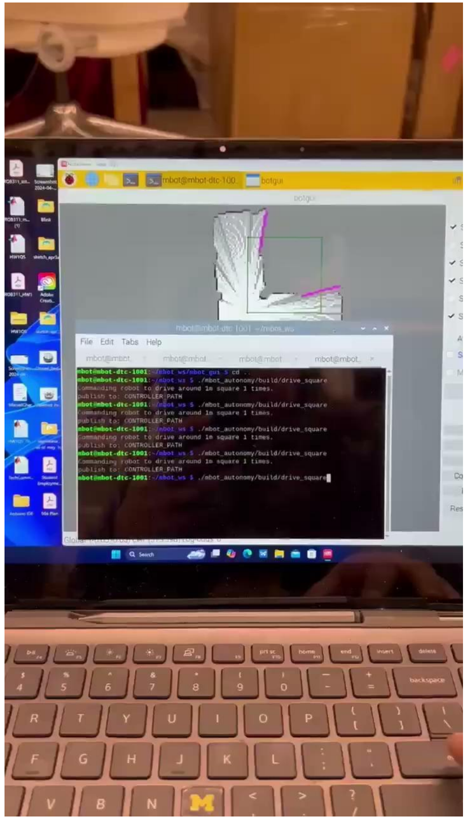
# ROB 330 (SLAM and Navigation) Final Project

The goal of this project is to implement a SLAM algorithm on a two-wheeled robot and demonstrate autonomous mapping and exploration of unknown mazes.

I worked in a group to implement a PID controller for driving, a particle filter to fuse odometry and LiDAR data for position estimation, and a frontier-based exploration algorithm. As the robot drives and accumulates LiDAR scans, it builds an map segmenting its surroundings into a grid of occupancy probabilities. It also has a particle cloud about its estimated position, and propagates particle motion using its motor encoder data (+ Gaussian noise). It then weights particles by how closely their estimated position relative to obstacles match the latest LiDAR scan. By taking the weighted average of the best particles, it generates a more accurate understanding of its position.

The exploration algorithm chooses the next location to explore by identifying clusters of “free” cells adjacent to unknown cells (frontiers) and employs A* with associated costs to penalize long paths and encourage reachability.





```
mbot@mbot-dtc-1001:~/mbot_ws/mbot_gui $ cd ..
mbot@mbot-dtc-1001:~/mbot_ws $ ./mbot_autonomy/build/drive_square
Commanding robot to drive around 1m square 1 times.
publish to: CONTROLLER_PATH
mbot@mbot-dtc-1001:~/mbot_ws $ ./mbot_autonomy/build/drive_square
Commanding robot to drive around 1m square 1 times.
publish to: CONTROLLER_PATH
mbot@mbot-dtc-1001:~/mbot_ws $ ./mbot_autonomy/build/drive_square
Commanding robot to drive around 1m square 1 times.
publish to: CONTROLLER_PATH
mbot@mbot-dtc-1001:~/mbot_ws $ ./mbot_autonomy/build/drive_square
Commanding robot to drive around 1m square 1 times.
publish to: CONTROLLER_PATH
mbot@mbot-dtc-1001:~/mbot_ws $ ./mbot_autonomy/build/drive_square
```

Demonstration of autonomous navigation and mapping of unknown maze