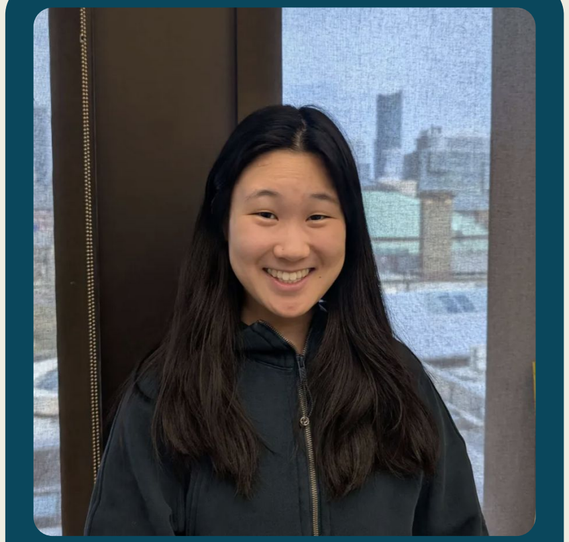
PARSIGHT

FLYRS ROB498 CAPSTONE



Felicia Liu

Luke Yang

Rohan Batchu

Sid Khanna

F L R S

PARSIGHT

FLYRS ROB498 CAPSTONE

ROADMAP

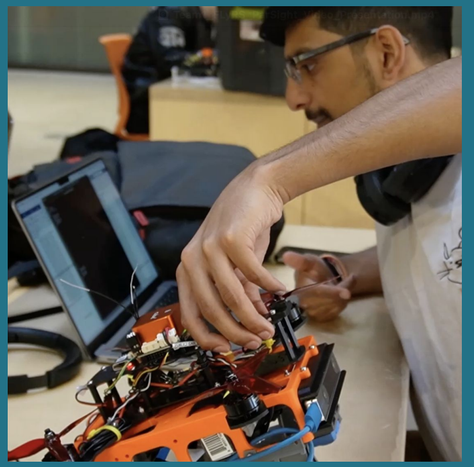
Introduction
- Motivation and Value Proposition
- Problem Statement and Scope

Core Design Functionality
- Objectives, Functions, Metrics
- Minimal Viable Product

Live Demonstration
- Initial Flight Checks
- Flight Process
- General Troubleshooting

Design Components Walk-Through
- Sense Plan Act Paradigm
- Perception and Sensing
- Image Processing and Ball Detection
- Tracking and Path Planning

Evaluation and Testing
- Design Process and Teamwork
- Incremental Testing
- Verification and Validation
- Design Practices Implemented
- Objectives Achieved
- Challenges, Limitations, and Next Steps



INTRODUCTION

MOTIVATION

Growing Senior Golf Demographic

- Players aged 50+ account for 43% of US golfers.

Golf's Dependence on Visual Skill

- Depth perception, contrast sensitivity, fixation ability.

Visual Challenges Faced by Seniors

- "Now? [...] following a drive? Not a scooby!"
- "It practically disappears once it reaches near its apex."

"Many things deteriorate as you get older and right up there at the top of the list is eyesight. I could always follow a golf ball and my playing partners used to depend upon me to find their golf balls in the rough. Now? I can see it take off clearly enough and am fine on most par threes. But following a drive? Not a scooby!"

- Derek Clements (Mon 22 Jul 2024)

EXISTING SOLUTIONS

- Focus on performance tracking, video enhancement, or post-game analysis.
- Does not support real-time visual support for golf.

Shot Tracer

Manually Piloted Drones

Follow-Me Drones

PROBLEM STATEMENT AND SCOPE

Stakeholder Need:
Senior golfers experience age related vision decline.

Existing Solutions:
Focus on broadcast graphics and post-game analysis.

Market Gap:
No solutions currently offer real-time, on-course better ball visibility.

ParSight aims to fill this gap.

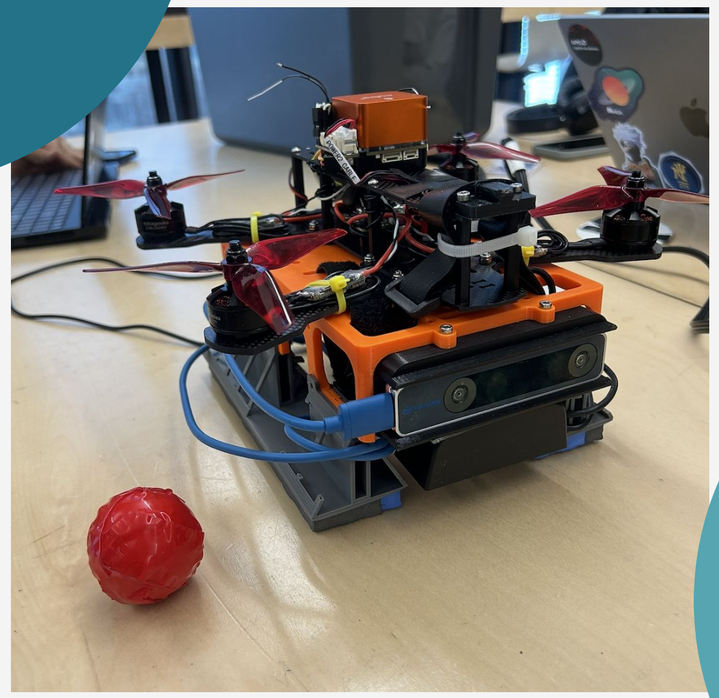
- Autonomous system.
- Drone-based.
- Real-time tracking.
- Enhance visibility.

VALUE PROPOSITION

Autonomous Sports Tracking
- Addresses market gap.

Improves Accessibility and Enjoyment
- Drone serves as larger visual reference.
- Reduces eye strain and tracking difficulty.
- Reduce golf ball search time.

Promotes Autonomy
- Visual aid focus.
- Minimizes effect on traditional golfing experience.



CORE DESIGN FUNCTIONALITY

PROJECT OBJECTIVES - MISSION LEVEL REQUIREMENTS

MLR-001 Tracking

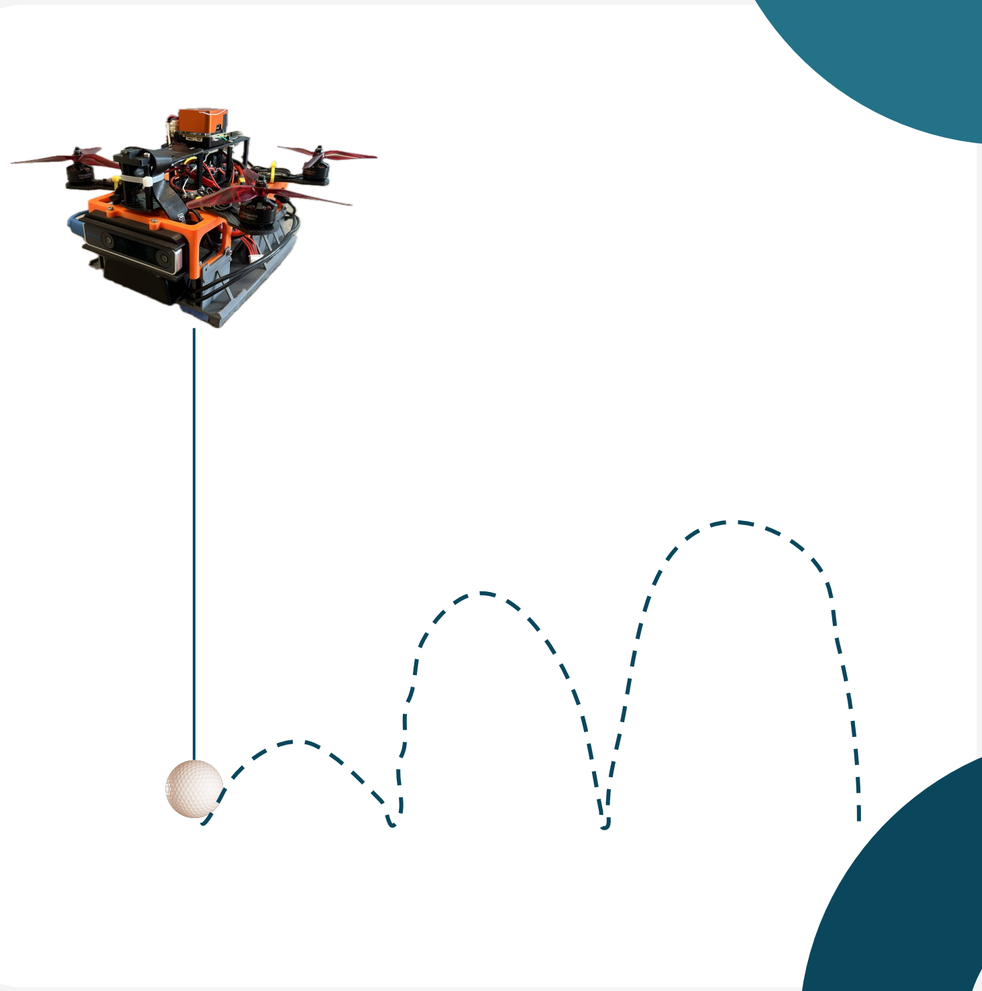
The golf ball shall be tracked from tee-off to rest.

MLR-002 Hovering

The drone shall hover over the golf ball at rest.

MLR-003 Ease of Use

The drone shall be easy to setup and use.



SYSTEM LEVEL FUNCTIONAL REQUIREMENTS

MLR-001: Tracking

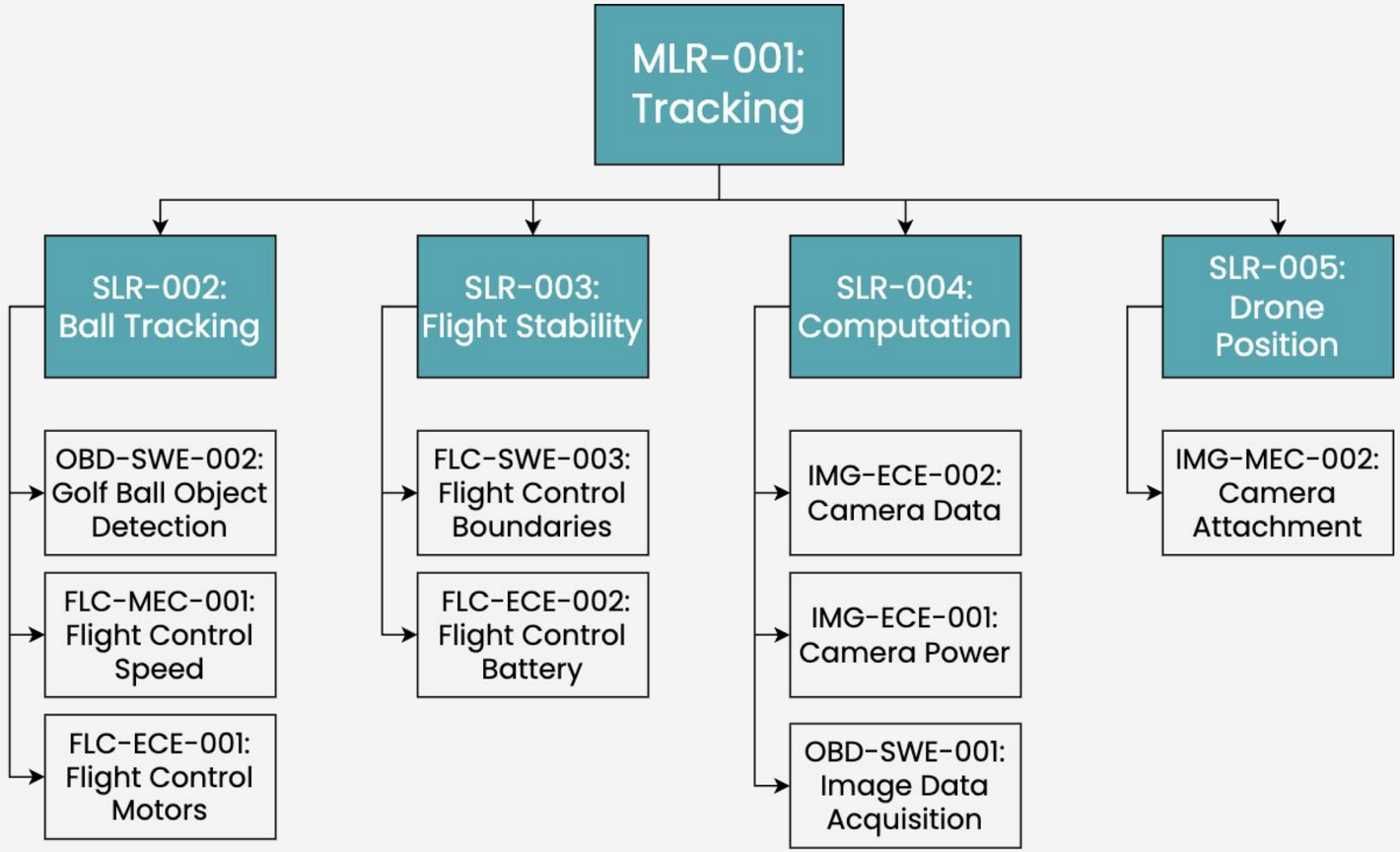
SLR-002: Drone Golf Ball Tracking: The drone shall maintain view of the golf ball throughout flight.

SLR-003 Drone Flight Stability: The drone shall have a safe and stable flight when tracking the golf ball.

SLR-004: Drone Computation: The drone shall have all computation through a Jetson Nano.

SLR-005: Drone Position: The drone will detect the ball’s position from overhead.

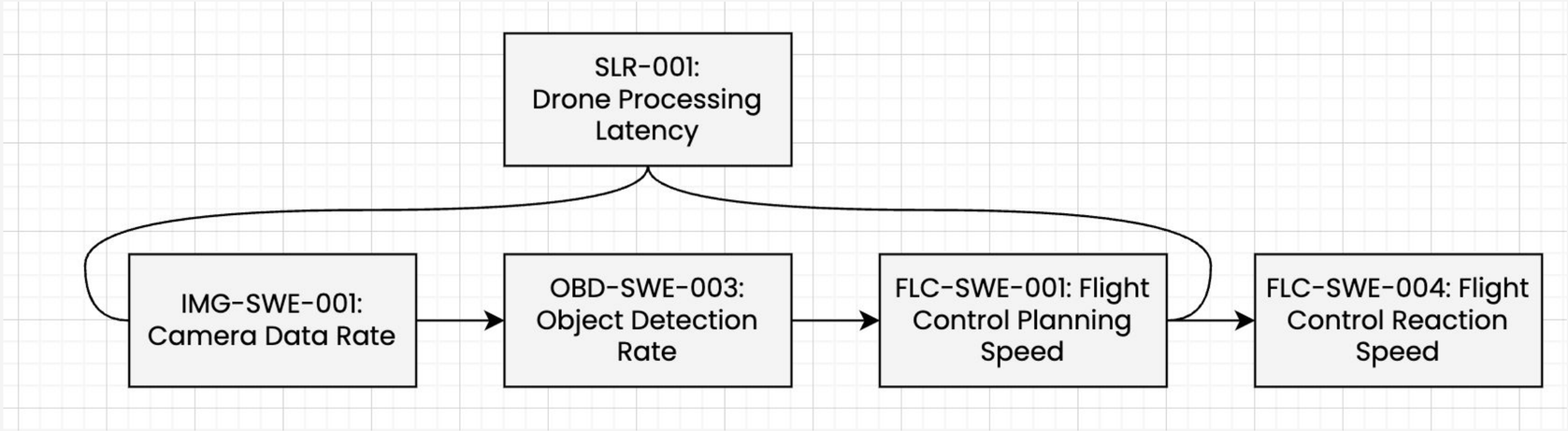
SUBSYSTEM FUNCTIONAL REQUIREMENTS

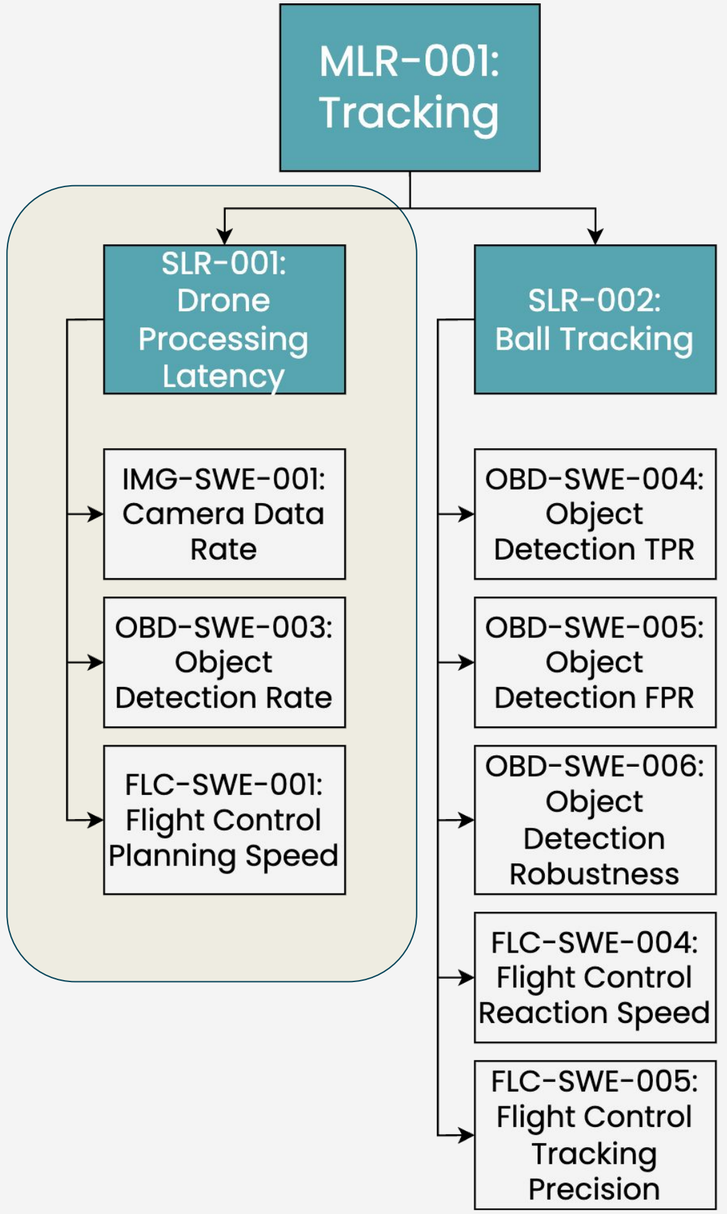


IMG - Perception and Sensing
OBD - Image Processing and Ball Detection
FLC - Tracking and Path Planning

SUBSYSTEM PERFORMANCE REQUIREMENTS

- **MLR-001: Tracking**
  - **SLR-001: Drone Processing Latency**
    - IMG-SWE-001: Camera Data Rate
    - OBD-SWE-003: Object Detection Rate
    - FLC-SWE-001: Flight Control Planning Speed
  - **SLR-002: Ball Tracking**
    - OBD-SWE-004: Object Detection TPR
    - OBD-SWE-005: Object Detection FPR
    - OBD-SWE-006: Object Detection Robustness
    - FLC-SWE-004: Flight Control Reaction Speed
    - FLC-SWE-005: Flight Control Tracking Precision
- **MLR-002: Hovering**
  - FLC-SWE-002: Final Position Convergence
- **MLR-003: Ease of Use**
  - SLR-006: Setup Time Limit


SLR-001:
Drone Processing
Latency
IMG-SWE-001:
Camera Data Rate
OBD-SWE-003:
Object Detection
Rate
FLC-SWE-001: Flight
Control Planning
Speed
FLC-SWE-004: Flight
Control Reaction
Speed


MLR-001:
Tracking
SLR-001:
Drone
Processing
Latency
SLR-002:
Ball Tracking
IMG-SWE-001:
Camera Data
Rate
OBD-SWE-003:
Object
Detection Rate
FLC-SWE-001:
Flight Control
Planning Speed
OBD-SWE-004:
Object
Detection TPR
OBD-SWE-005:
Object
Detection FPR
OBD-SWE-006:
Object
Detection
Robustness
FLC-SWE-004:
Flight Control
Reaction Speed
FLC-SWE-005:
Flight Control
Tracking
Precision

MINIMAL VIABLE PRODUCT: MLR-001 TRACKING

Performance Requirement Code/Name	MVP Metric	Ideal Metrics
IMG-SWE-001: Camera Data Rate	**> 20 Hz**	**> 120 Hz**
OBD-SWE-003: Object Detection Rate	**> 20 Hz**	**> 100 Hz**
OBD-SWE-004: Object Detection TPR	**> 90%**	**> 95%**
OBD-SWE-005: Object Detection FPR	**< 5%**	**< 0.2%**
OBD-SWE-006: Object Detection Robustness	**> 85%**	**> 93%**
FLC-SWE-001: Flight Control Planning Speed	**< 20 ms**	**< 5 ms**
FLC-SWE-004: Flight Control Reaction Speed	**< 150 ms**	**< 90 ms**
FLC-SWE-005: Flight Control Tracking Precision	**< 40 pxls**	**< 100 pxls**
FLC-MEC-001: Flight Control Speed	**TRUE**	**TRUE**
SLR-001: Drone Processing Latency	**< 100 ms**	**< 20 ms**

OTHER PERFORMANCE REQUIREMENTS

MLR-002: Hovering

FLC-SWE-002: Flight Control Final Position Convergence	**< 20 cm**	**< 10 cm**

MLR-003: Ease of Use

SLR-006: Setup Time Limit	**< 5 mins**	**< 1 mins**

LIVE DEMONSTRATION

LIVE DEMONSTRATION

Flight Process:

- Launch Camera Node and Detection Node.
- Confirm detection of golf ball.
- Unkill, arm, switch to Offboard, and launch ParSight.
- Start test once detection is confirmed at flying height.
- Trigger catapult when drone begins tracking.
- Drone should hover over ball once it comes to rest.
- Land and kill the drone to end the demonstration.

Pre-Flight Checks:

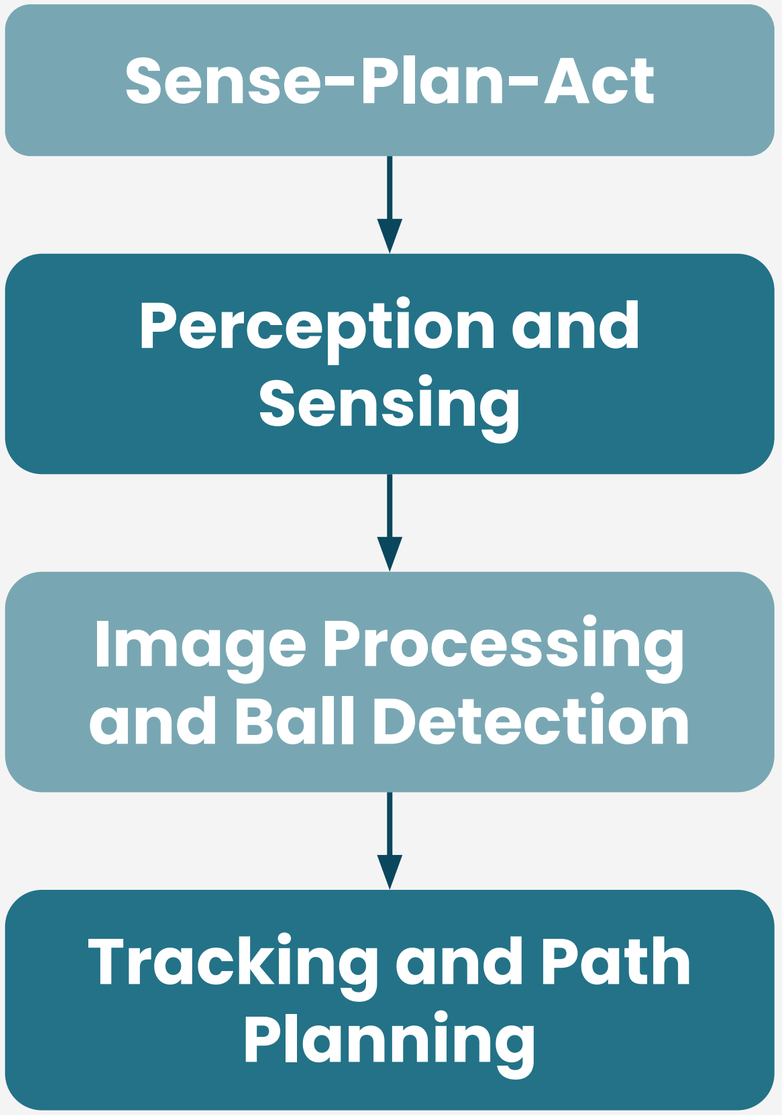
- Make sure wires are clear, net is pulled, and the area's safe.
- Check time synchronization.
- Confirm vision pose and local position alignment.
- Verify Vicon data is reading properly.

General Troubleshooting:

- Relaunch MAVROS.
- Replug drone components.
- Reboot the Cube.
- Swap in a fresh battery.
- Recalibrate if needed.

DESIGN COMPONENTS WALK-THROUGH

COMPONENTS


Sense-Plan-Act
Perception and Sensing
Image Processing and Ball Detection
Tracking and Path Planning

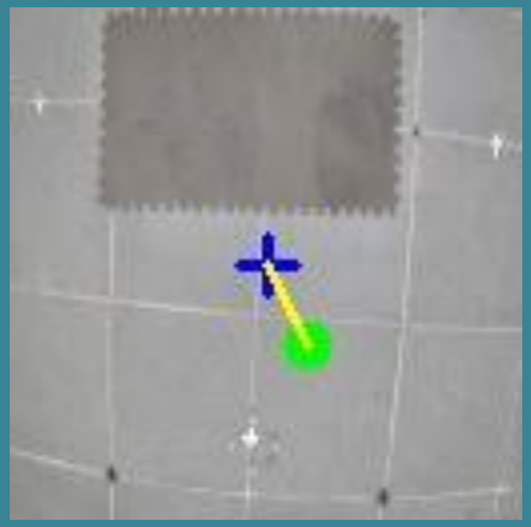
SENSE PLAN ACT PARADIGM

Sense:

- Camera captures live video of the floor.
- Uses color filtering & shape detection to isolate the golf ball.

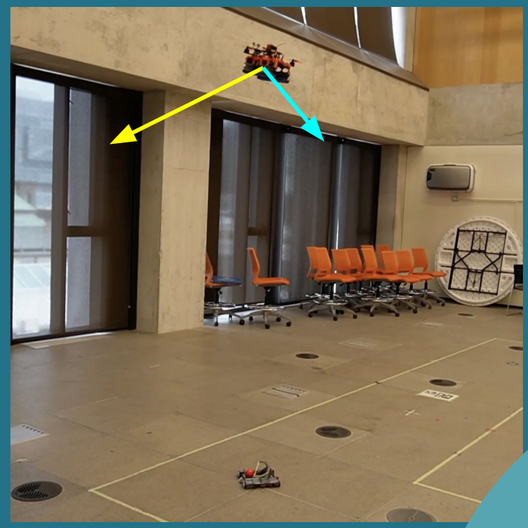
Plan:

- Calculates pixel offset to estimate ball position.
- PD controller computes positional adjustments.



Act:

- Drone setpoints are updated.
- Flight controller moves drone to keep ball centered in image.



INSPIRATION CODE

Skeleton Flight Test Code

```
def handle_abort():
    print('Abort Requested. Your drone should land immediately due to safety considerations')

# Service callbacks
def callback_launch(request, response):
    handle_launch()
    return response

def callback_test(request, response):
    handle_test()
    return response

def callback_land(request, response):
    handle_land()
    return response

def callback_abort(request, response):
    handle_abort()
    return response
```

Skeleton Flight Test Code

Service Callbacks:
callback_launch(self, request, response)
callback_test(self, request, response)
callback_land(self, request, response)
callback_abort(self, request, response)

Service Implementations:
launching_procedure(self)
testing_procedure(self)
landing_procedure(self)
abort_procedure(self)

Subscribers and Publishers:
realsense_callback(self, msg)
vicon_callback(self, msg)
send_setpoint(self)
send_vision_pose(self)

Main Pipeline:
frame_input_callback(self, msg)
full_image_processing(self, frame)
find_object_center(self, frame)
move_drone(self, p_error_x, p_error_y)

Helper Functions:
setup_image_parameters(self, frame)
set_pose_initial(self)
clamp_position(self, position)
set_target_color(self)
rgb_to_hsv(self, rgb)
calculate_pixel_difference(self, x, y)
mini_calculate_golf_ball_metrics(self)

Source: Flight Test Framework
Focus: Modular service call structure, separation of logic.
Outcome: Testable, maintainable modules.

PERCEPTION AND SENSING

Associated Requirements:
IMG-SWE-001: Camera Data Rate
IMG-ECE-001: Camera Power
IMG-ECE-002: Camera Data
IMG-MEC-002: Camera Attachment

IMX219 CSI Camera

Resolution: Up To 4K
Data Rate: Up to 180 FPS
Interface: CSI Port

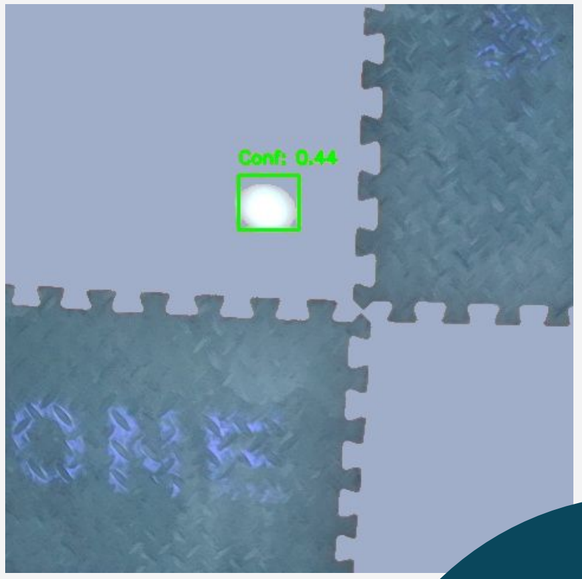
NexiGo N660 USB Camera

Resolution: Up To 1080p
Data Rate: Up to 30 FPS
Interface: USB-A

RESCOPING VISION SYSTEM

Initial Choice - YOLO

- Filter false positives more effectively for white golf ball.
- Jetson can run deep learning.



Major Drawback

- Drains battery power very quickly.
- Slow tracking response because of inference processing time.
- Inconsistent detection (flicker).

Rescope - Red Filter

- HSV-based color filtering using a red coloured golf ball.
- CV2 contour implementation.

Benefits

- Lightweight and fast computation.
- Easier to tune (than retraining NN).
- More stable detection, less flickering.

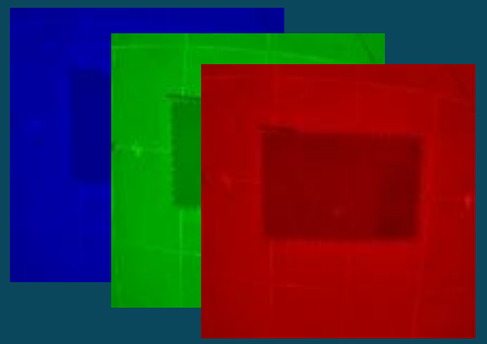
IMAGE PROCESSING AND BALL DETECTION

1. Frame Conversion

- Convert OpenCV RGB image to HSV.
- Invariant and more robust to colour changes.
- Better Control for colour detection.

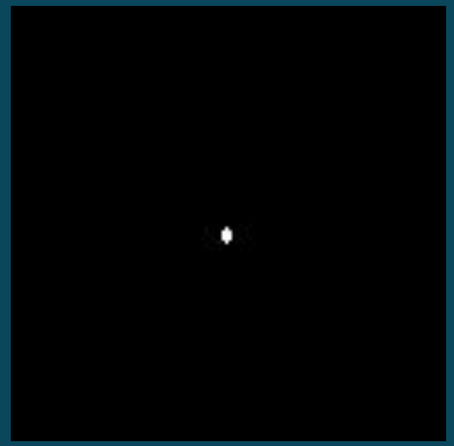
2. Color Removal

- Apply red mask and remove green and blue hues.
- Suppresses noise by other colored objects.
- Cleaner input for detection.



3. Gaussian Blur

- Smooths edges and reduces small noise specks.
- Removes floor texture.
- Improves stability.



4. Contour Scoring and Selection

- Detect all contours in the mask.
- Score each by size and roundness.
- Extract ball center for movement tracking.

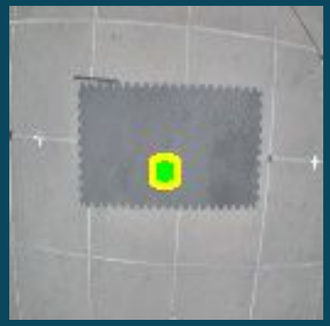
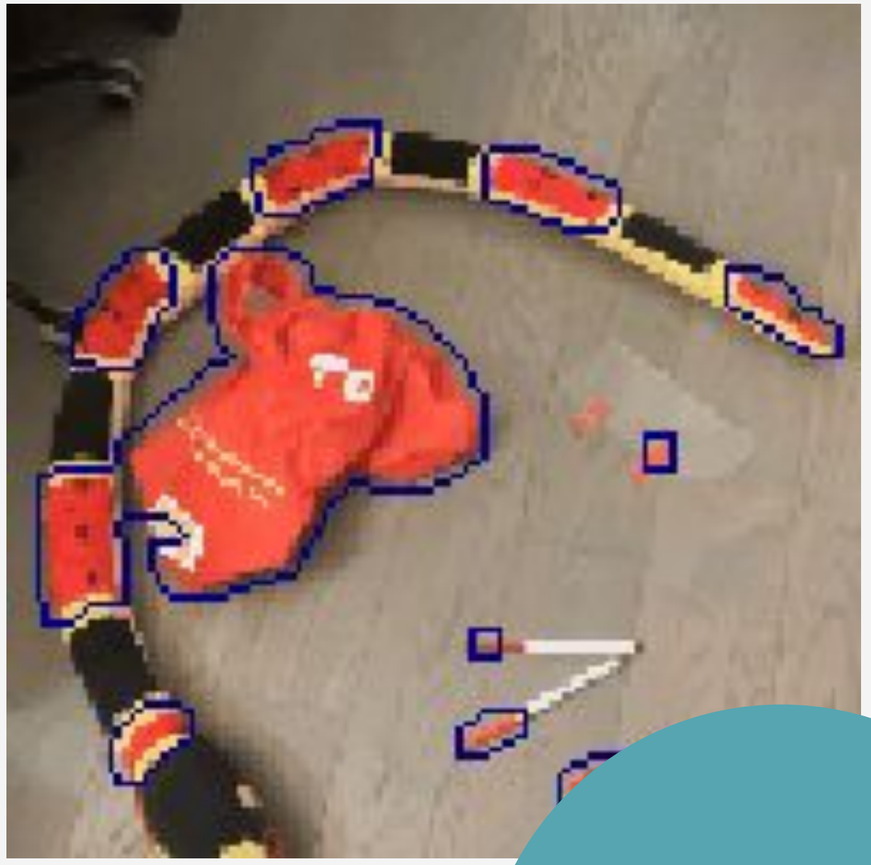


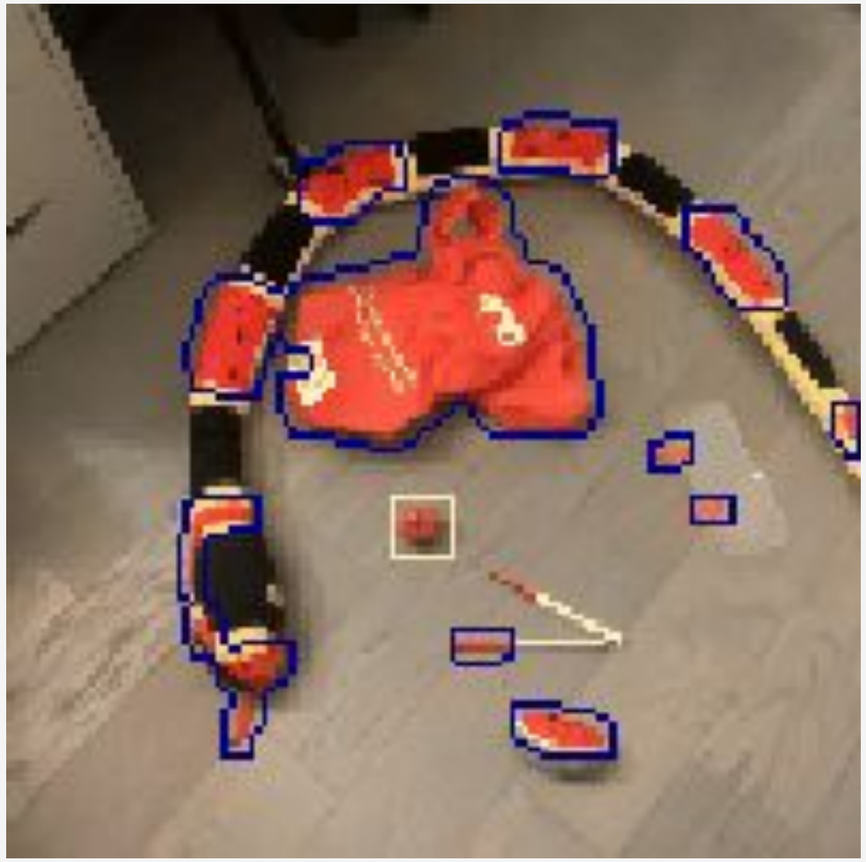
IMAGE PROCESSING AND BALL DETECTION

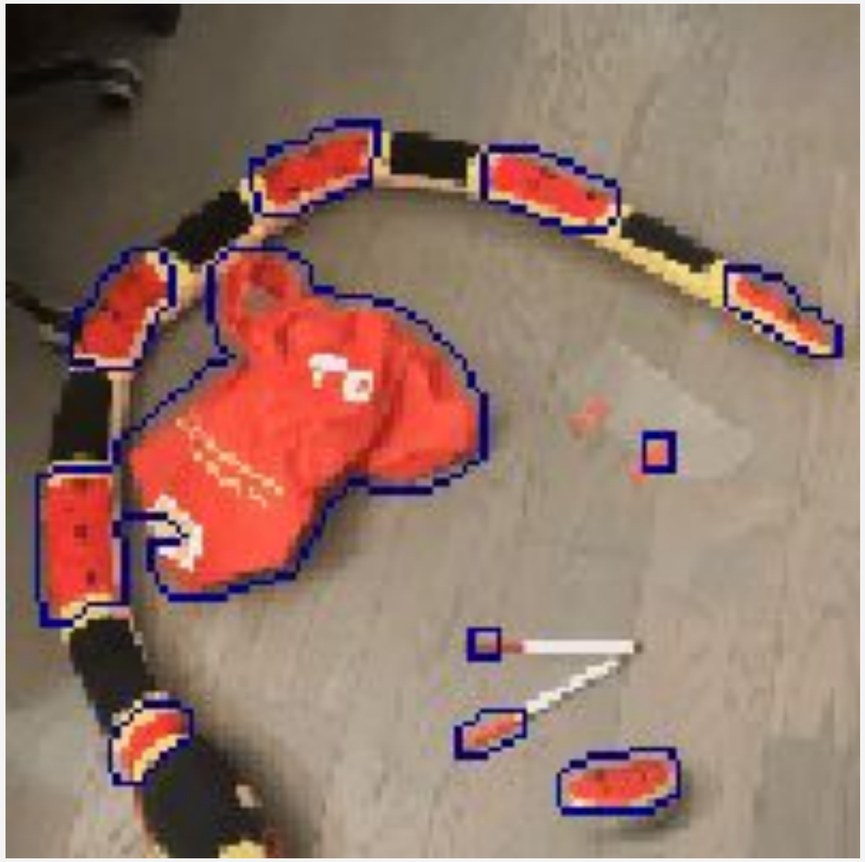


Contour Selection

- Score all contours based on size and roundness.
- Only detects if contour above a set threshold.
- More robust detection.

IMAGE PROCESSING AND BALL DETECTION



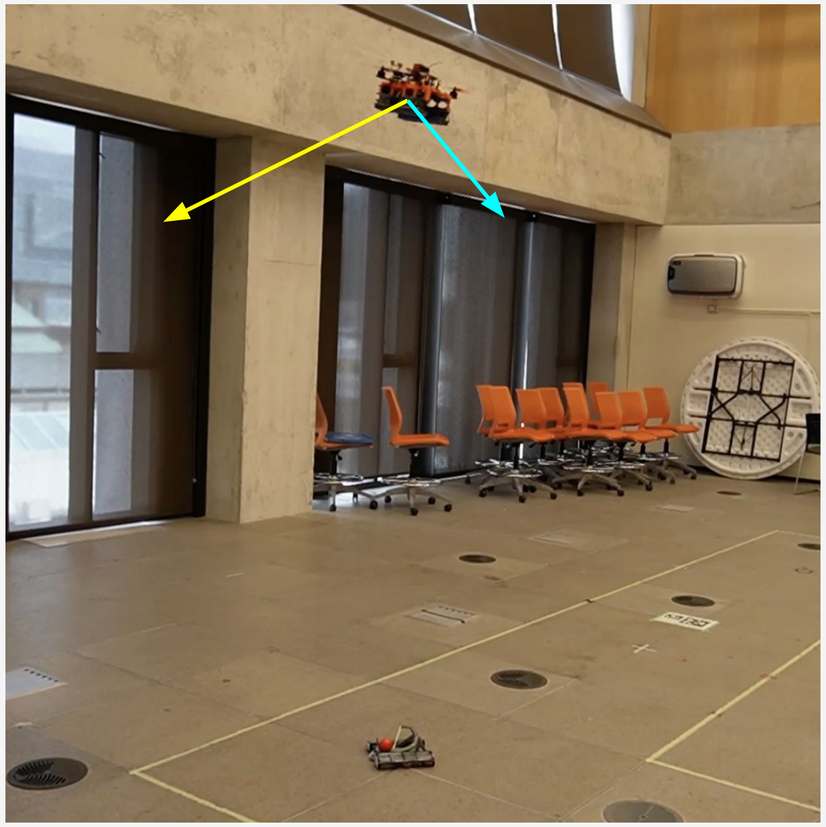


I can do this one if necessary (felicia)

This is the False Positive Detection Becoming More robust

TRACKING & PATH PLANNING

Prior: Image plane center point for the ball's position *(x, y)*.
Goal: Adjust *setpoint_position* to track ball position with feedback.
Method: Image Based Visual Servoing (IBVS) + PD Control.

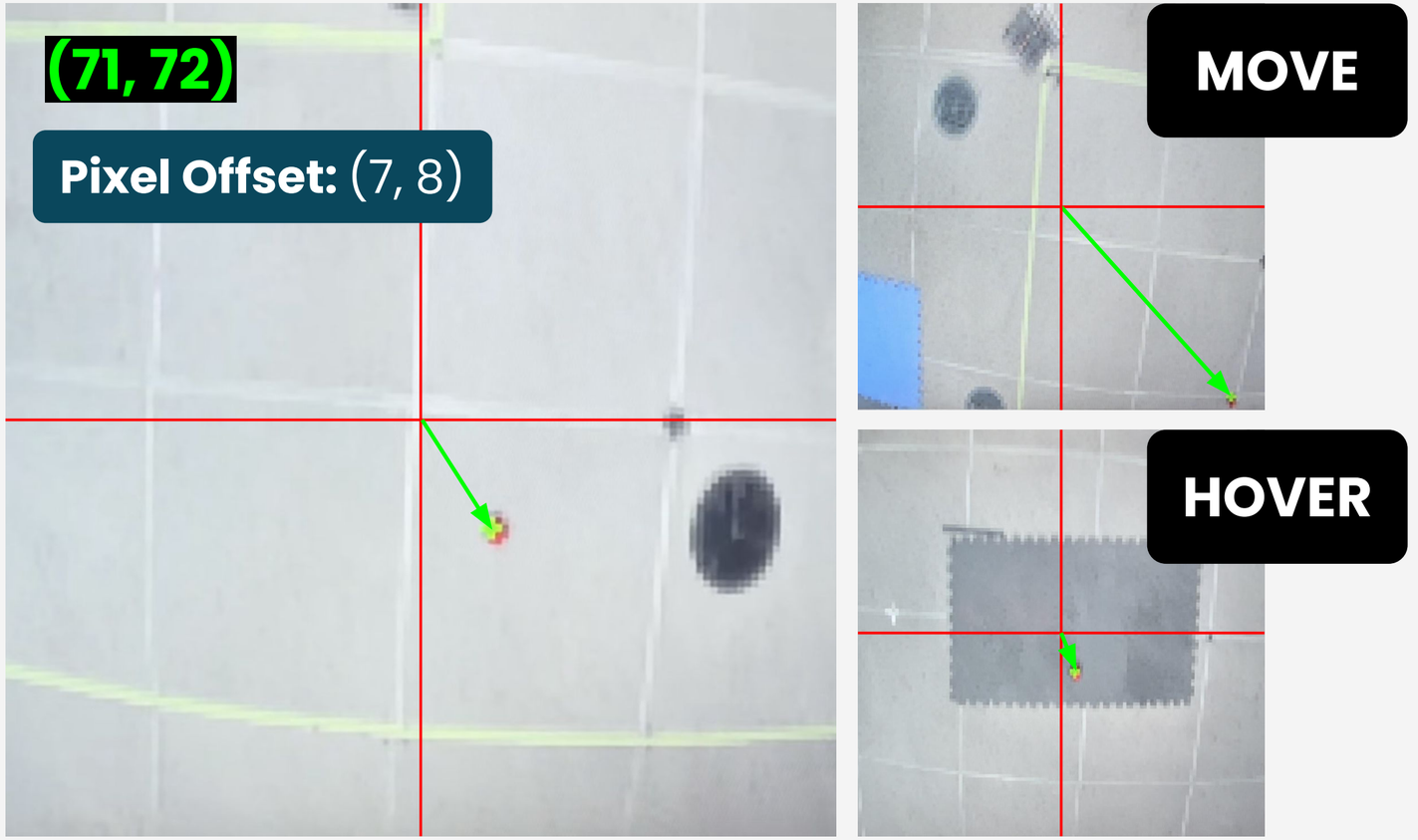


TRACKING & PATH PLANNING

Prior: Image plane center point for the ball's position *(x, y)*.
Goal: Adjust *setpoint_position* to track ball position with feedback.
Method: Image Based Visual Servoing (IBVS) + PD Control.

1. Compute Pixel Offset
Ball (x, y) and frame center reference (x, y).

2. Movement Choice
Small offset vector, no movement.



TRACKING & PATH PLANNING

Prior: Image plane center point for the ball's position *(x, y)*.
Goal: Adjust *setpoint_position* to track ball position with feedback.
Method: Image Based Visual Servoing (IBVS) + PD Control.

1. Compute Pixel Offset
Ball (x, y) and frame center reference (x, y).

2. Movement Choice
Small offset vector, no movement.

3. Position Offset
Calibrate to real world using known ball size.

Pixel Offset: (7, 8)

↓

0.01767 m = 1 pixel

↓

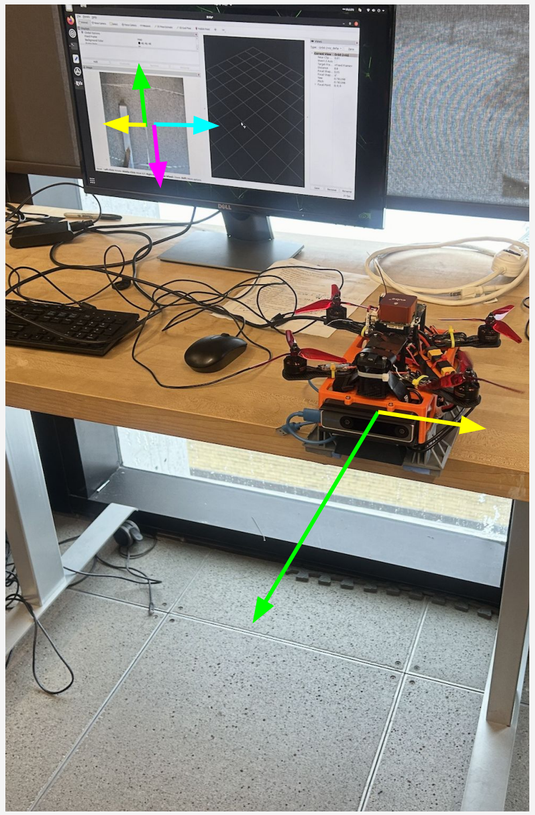
Position Error: 18.78 cm

TRACKING & PATH PLANNING

Prior: Image plane center point for the ball's position *(x, y)*.
Goal: Adjust *setpoint_position* to track ball position with feedback.
Method: Image Based Visual Servoing (IBVS) + PD Control.

4. PD Control
Tuning Kp Kd gains to feed in position error and decide on next position prediction.

```
# calculate derivative component
curr_time = self.get_clock().now()
dt = (curr_time - self.prev_time).nanoseconds / 1e9
if dt == 0: dt = 1e-6  # prevent division by zero
d_error_x = (p_error_x - self.prev_p_error_x) / dt
d_error_y = (p_error_y - self.prev_p_error_y) / dt
# PD control signal
move_x = self.Kp * p_error_x + self.Kd * d_error_x
move_y = self.Kp * p_error_y + self.Kd * d_error_y
# update the drone's position with the scaled values
self.set_position.x = self.position.x - move_y
self.set_position.y = self.position.y - move_x
self.set_position.z = self.desired_flight_height
# save current state for next iteration
self.prev_p_error_x = p_error_x
self.prev_p_error_y = p_error_y
self.prev_time = curr_time
self.t2 = time.time()
```

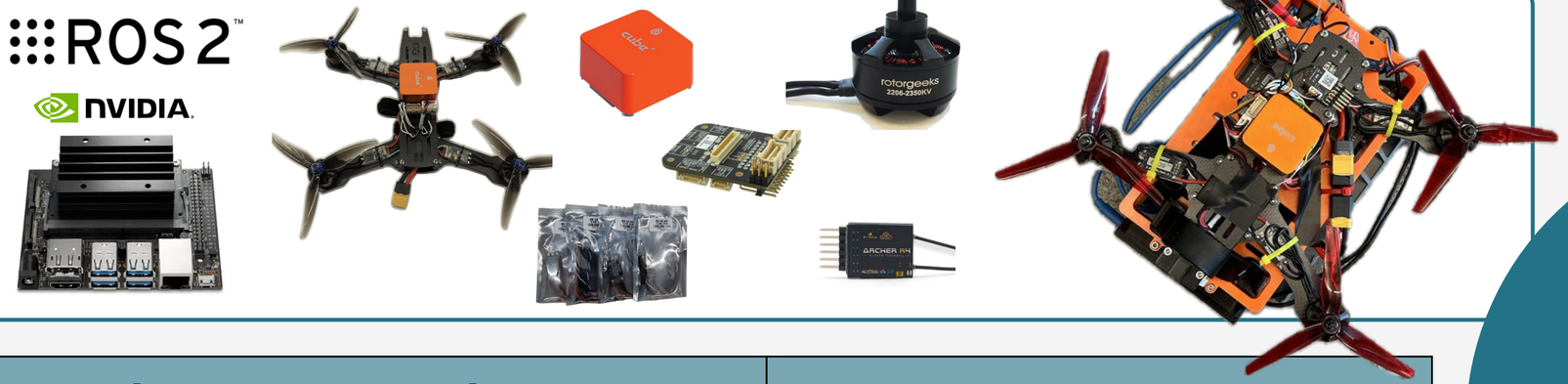


TRACKING & PATH PLANNING

Prior: Image plane center point for the ball's position *(x, y)*.
Goal: Adjust *setpoint_position* to track ball position with feedback.
Method: Image Based Visual Servoing (IBVS) + PD Control.

4. PD Control
Tuning Kp Kd gains to feed in position error and decide on next position prediction.

5. Actuation
Update setpoint locations using frame transform and Cube + will handle movement.



Chose (Design Decision)	Tradeoff
PD Control for fast, responsive tracking with oscillation damping.	No Integral Correction could have potential SS error.
Fixed Altitude Mapping means simplified pixel-to-world conversion.	Scaling Inaccuracy if altitude/resolution changes.

TESTING PROCEDURE

TEAM WORKFLOW

Division of Labour:

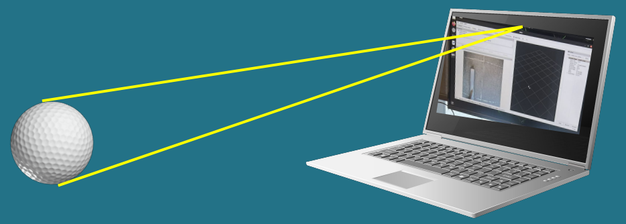
- Development environment/deployment pipeline
- Camera setup and data transmission
- CV and golf ball detection
- Drone tracking algorithm

Continuous Deployment

- Frequent flights, assessment, and adjustments
- Lower altitude and speed

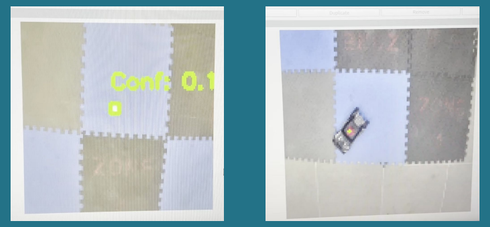
DESIGN PRACTICES IMPLEMENTED

Safety First Approach:
Early tests done on ground/laptop before flying to minimize risk.



Incremental Dev:
Separately validated perception, object detection, and control modules before full integration.

Modular Architecture:
Components like detection can be swapped without reworking the full pipeline.



Context Assumptions:
Used fixed height + Vicon for simpler control in a known environment.

Planning & Documents:
Regular meetings, clear task division, and consistent documentation kept the project organized.

INCREMENTAL TESTING INTRODUCTION

Test Name	Test Type	Control Type	In Flight?
Laptop Detection Test	Detection	-	-
Drone Detection Test	Detection	-	NO
Drone Tracking Test	Tracking	Fixed Step (1 cm)	NO
Drone Detection Slow Movement (S)	Detection	-	YES
Drone Tracking Slow Movement (1cm)	Tracking	Fixed Step (1 cm)	YES
Drone Tracking Slow Movement (P)	Tracking	P Control	YES
Drone Tracking Slow Movement (PD)	Tracking	PD Control	YES
Drone Tracking Fast Movement (PD)	Tracking	PD Control	YES
MVP: Drone Tracking Catapult Launch	Tracking	PD Control	YES

LAPTOP DETECTION TEST

Why?

- Ground-based using laptop.
- Confirm reliability to identify a golf ball in real-time in a variety of different conditions.
- Validate segmentation model accuracy before flying.

Testing Conditions:

- Myhal: accuracy for demonstration and testing.
- Outside: proof of concept for real-world deployment.

Myhal Testing Conditions - Red Robustness
Large Red Tape | Small Red Tape | Red Circle

Different Lighting Conditions
Bright | Dim | Dark

Varying Background Complexities
No Obs | Some Small Obs | Many Obs

Outside on Grass
Proof of Concept

LAPTOP DETECTION TEST (LIGHTING CONDITIONS)

Bright/Normal Conditions

TPR: 97.47%
FPR: 3.56%

Dim Room Lighting

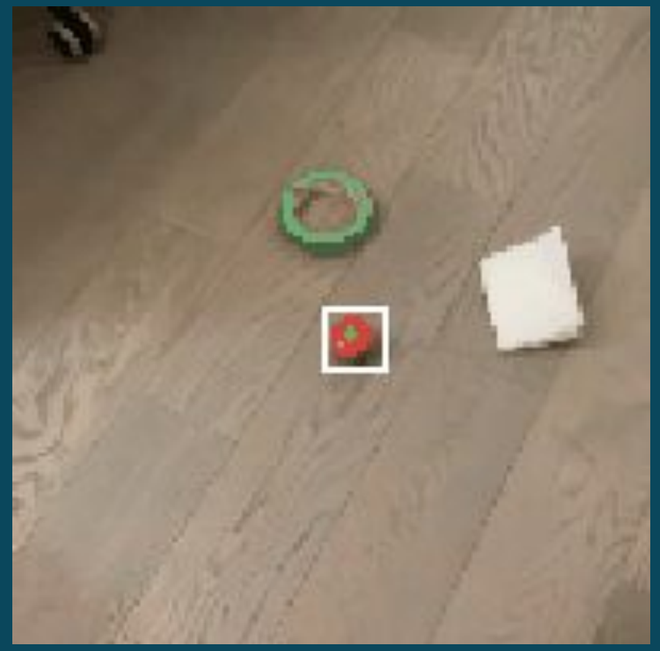
TPR: 92.77%
FPR: 4.00%

Darker Light Conditions

TPR: 95.38%
FPR: 0.00%

LAPTOP DETECTION TEST (BACKGROUND COMPLEXITY)

Clean / No Obstructions

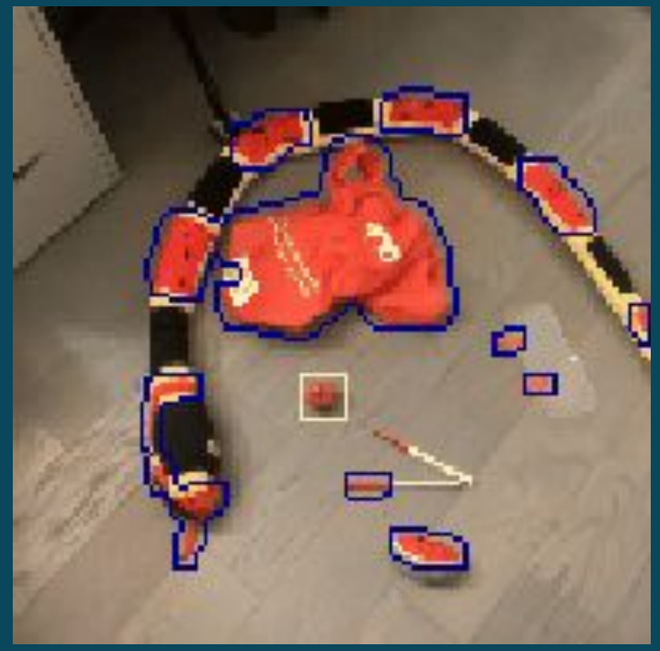


TPR: 100.00%
FPR: 0.00%

Medium Background

TPR: 99.53%
FPR: 0.00%

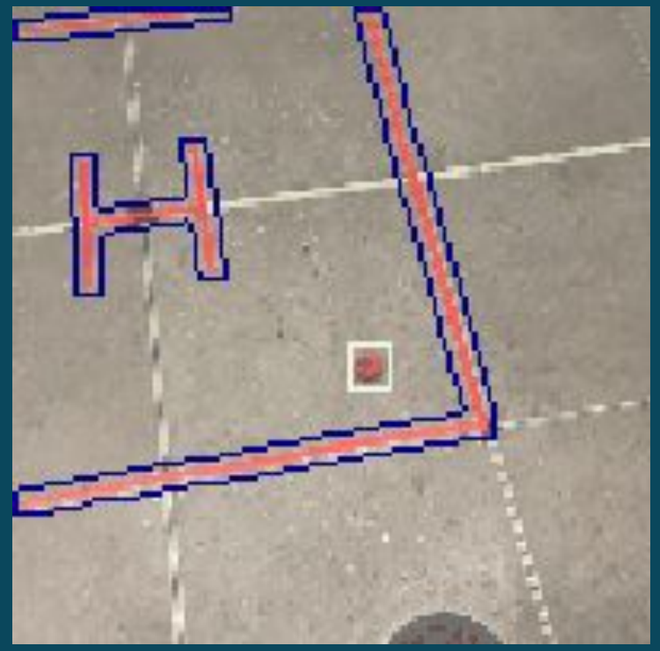
Difficult Background



TPR: 96.53%
FPR: 6.03%

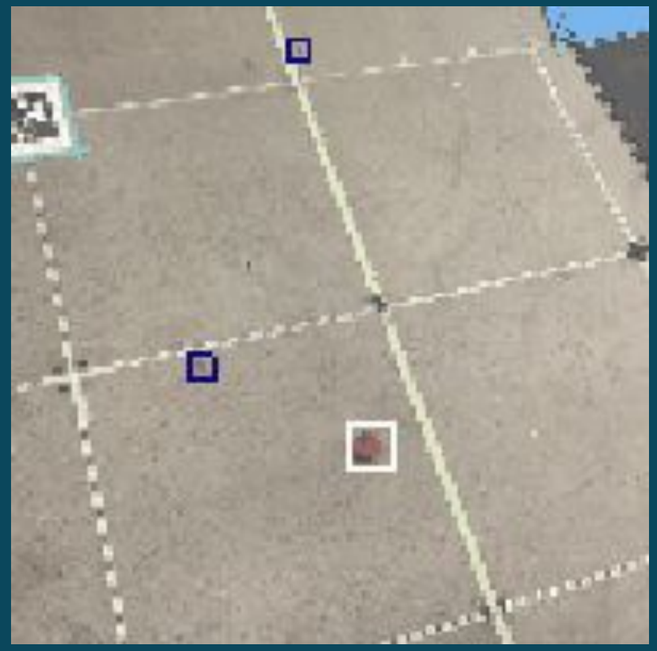
LAPTOP DETECTION TEST - RED ROBUSTNESS

Large Red Tape



TPR: 97.33%
FPR: 3.02%

Small Red Tape



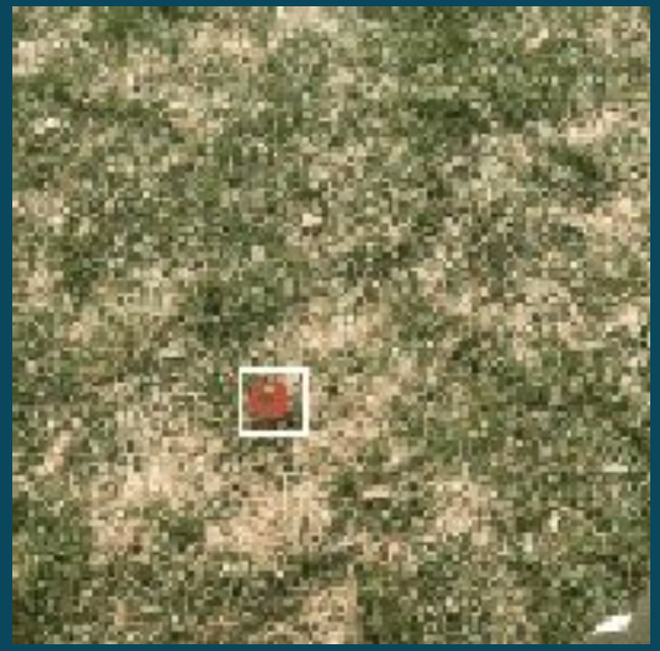
TPR: 97.12%
FPR: 3.10%

Previous Failure Case

TPR: 95.60%
FPR: 0.00%

LAPTOP DETECTION TEST

Red Ball – Front Campus



TPR: 95.54%
FPR: 0.00%

White Ball – Front Campus

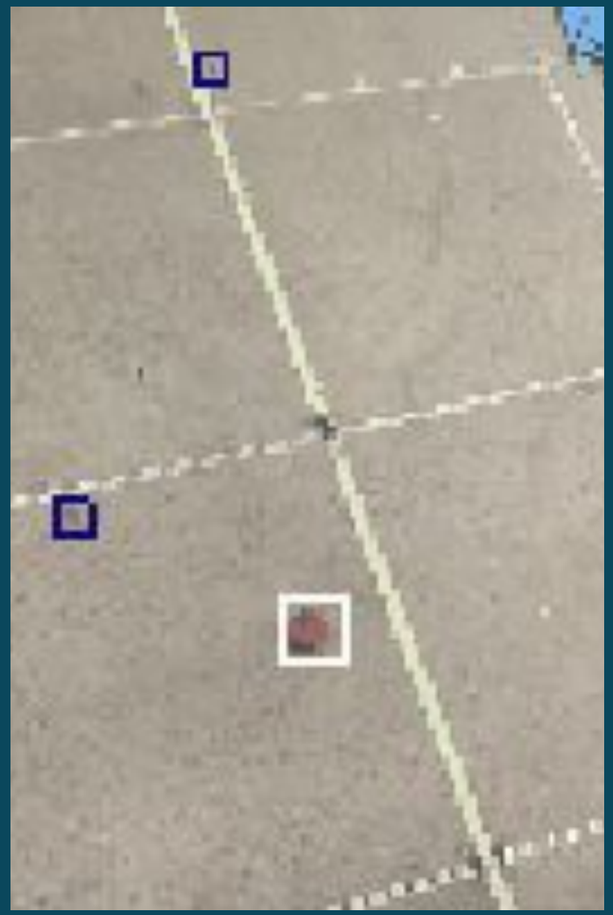
TPR: 100.00%
FPR: 0.00%

LAPTOP DETECTION TEST

Object Detection TPR: **~95%**
Object Detection FPR: **~3%**
Object Detection Robustness: **~95%**

- **Why?** Ground-based using laptop, validate segmentation model accuracy before flying.
- **What?** Identify a golf ball in real-time in many different conditions.
- **Result?** Great TPR and FPR, we can move to using the drone!

Myhal Testing Conditions



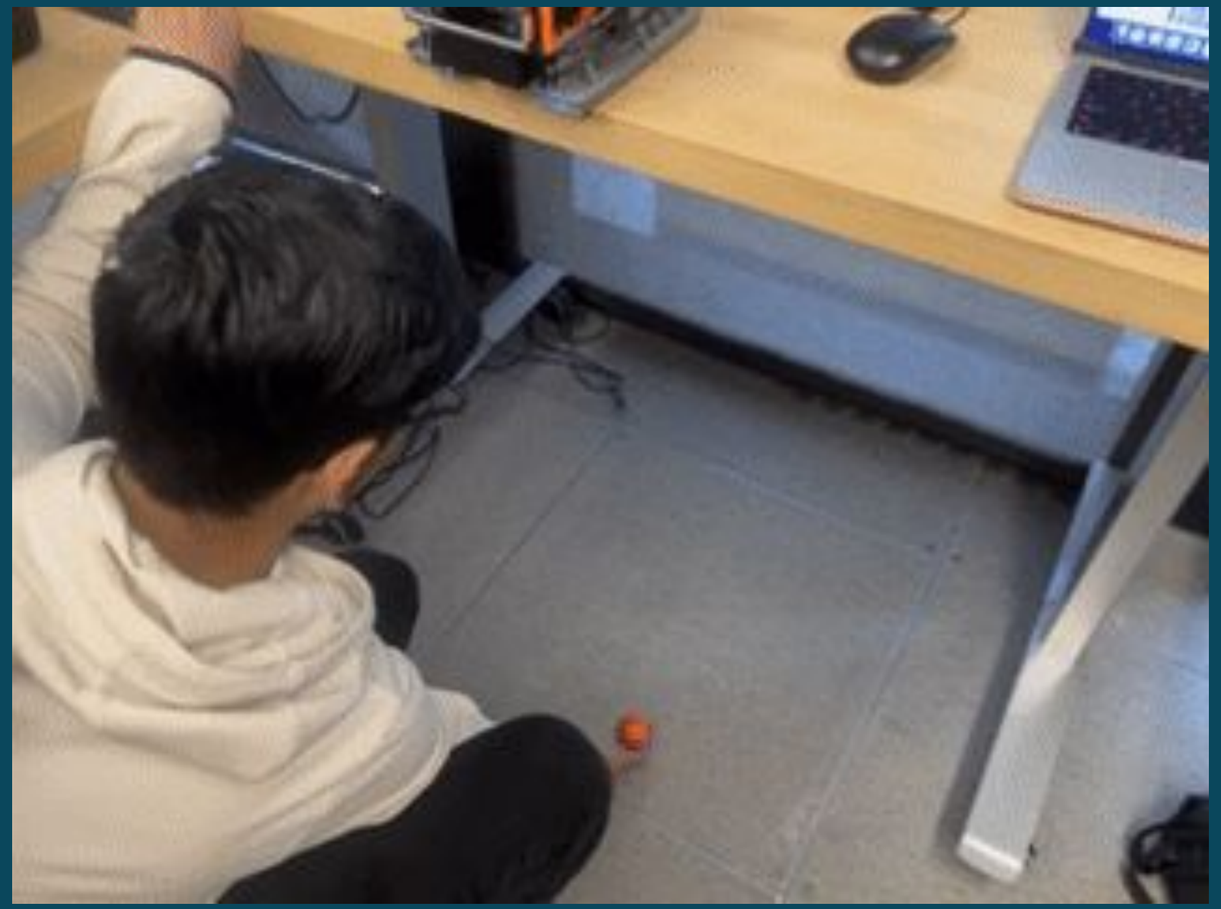
Front Campus Grass

DRONE DETECTION TEST (NO FLIGHT)

Object Detection TPR: **95%**
Object Detection FPR: **3%**
Object Detection Robustness: **95%**
Object Detection Rate: **30Hz**
Camera Data Rate: **30Hz**

- **Why?** Confirm perception pipeline speed and accuracy on drone hardware (Jetson Nano).
- **What?** Drone stationary on table with program object detection running.
- **Result?** Successful object detection that passed all relevant requirements.

Object Detection on Drone Hardware

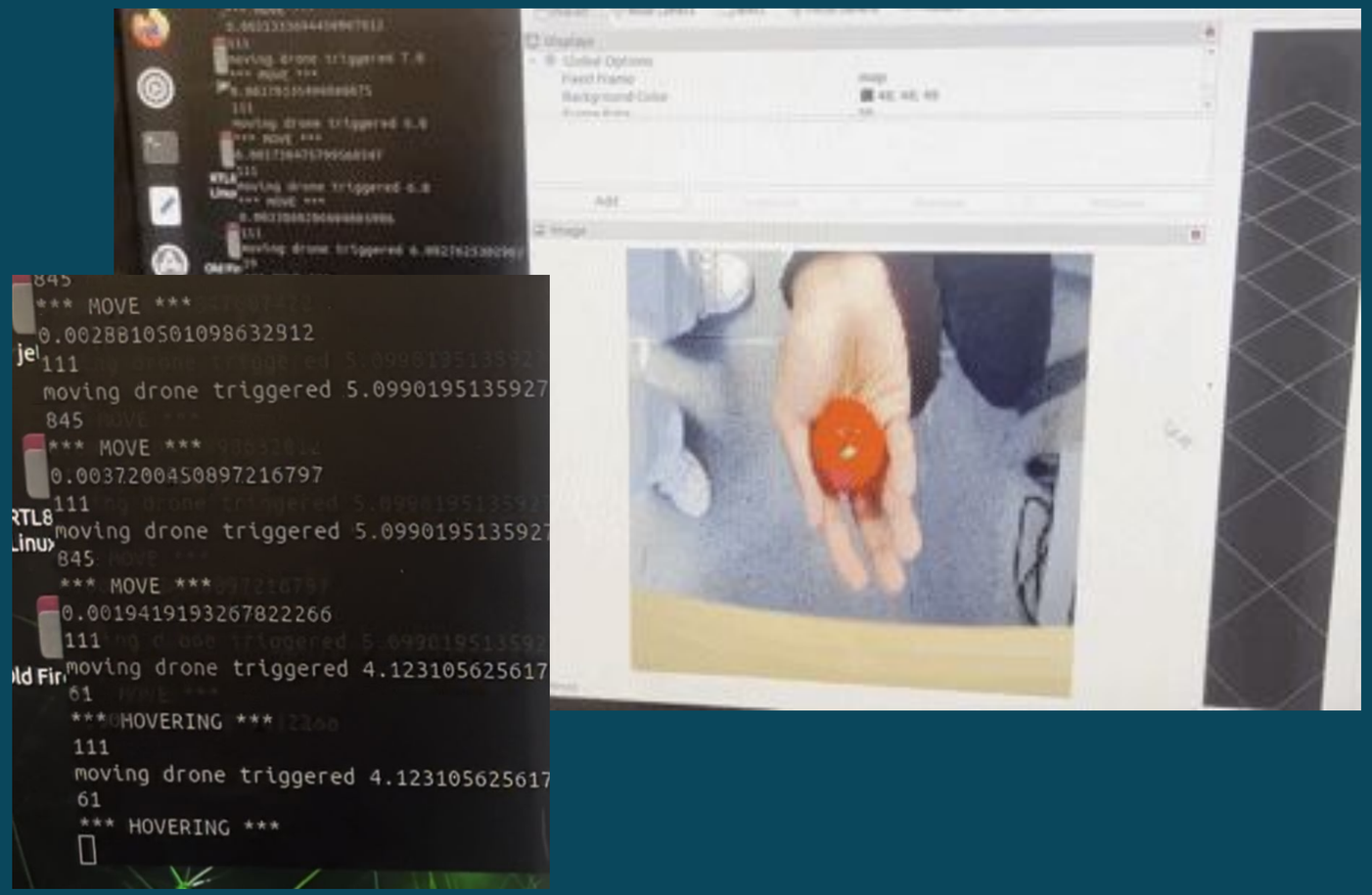


DRONE TRACKING TEST (NO FLIGHT)

Object Detection TPR: **95%**
Object Detection FPR: **3%**
Object Detection Robustness: **95%**
Object Detection Rate: **30Hz**
Camera Data Rate: **30Hz**
Drone Processing Latency: **30.5 ms**

- **Why?** Confirm drone setpoint response to golf ball offsets.
- **What?** Drone stationary on table with object detection with tracking and planning running.
- **Result?** Expected results, close enough says “HOVER” and further pixel disparities require “MOVE”.

Planning and Tracking on Drone Hardware



DRONE DETECTION SLOW MOVEMENT (STATIC HOVER ONLY)

Object Detection TPR: **95%**
Object Detection FPR: **3%**
Object Detection Robustness: **95%**
Object Detection Rate: **30Hz**
Camera Data Rate: **30Hz**
Drone Processing Latency: **30.5 ms**

- **Why?** Confirm reliable in-flight perception before enabling motion.
- **What?** Drone hovers while tracking golf ball pulled by toy car.
- **Result?** No FPs from ground tape, table target lock, fast detection during flying confirmed, safe to proceed with tracking flight.

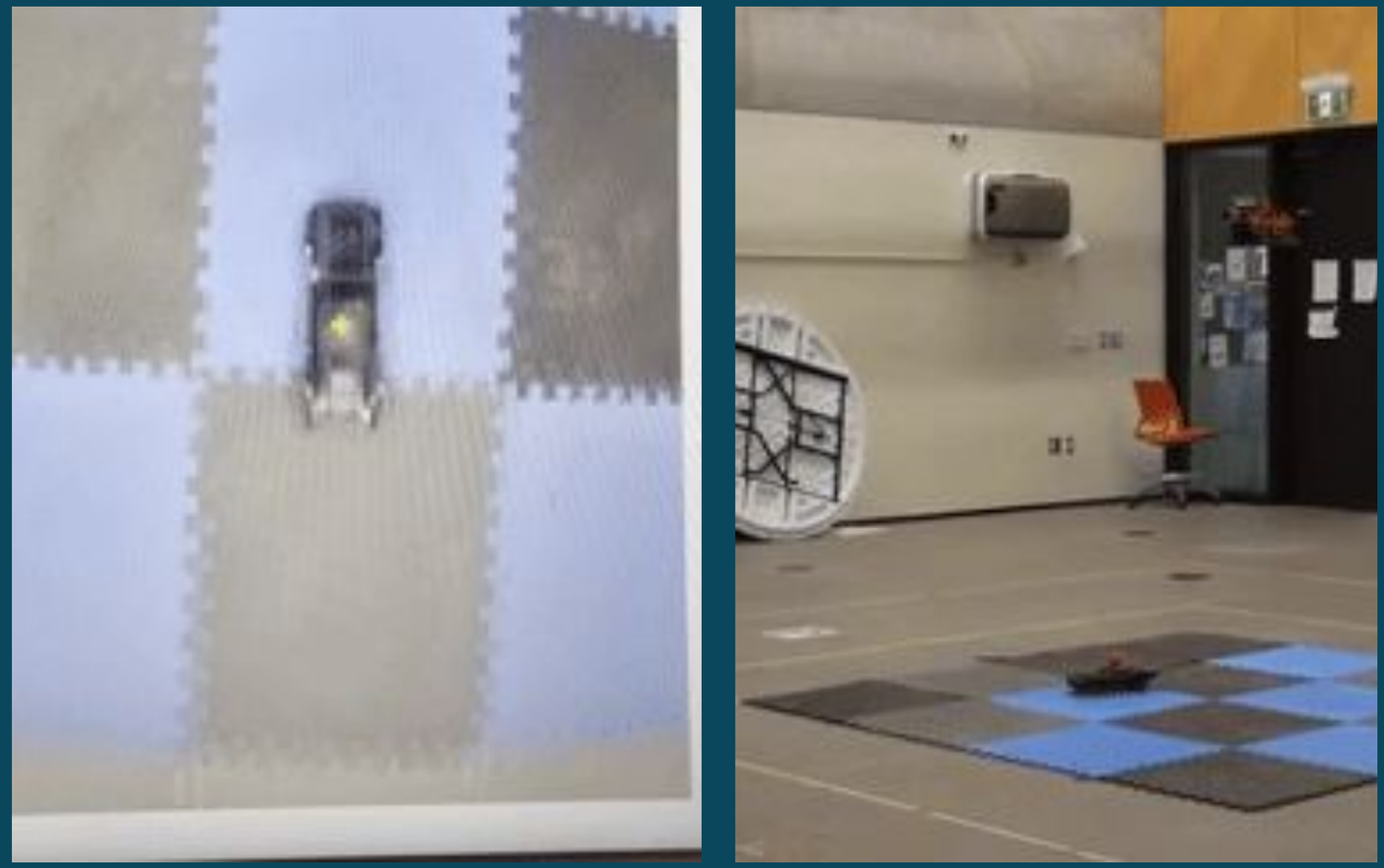
Car Drives Underneath (Perfect Detection)

DRONE TRACKING SLOW MOVEMENT (1 CM INCREMENTS)

Object Detection TPR: **95%**
Object Detection FPR: **3%**
Object Detection Robustness: **95%**
Object Detection Rate: **30Hz**
Camera Data Rate: **30Hz**
Drone Processing Latency: **30.5 ms**
FC Speed: **Not Satisfied**
FC Tracking Precision: **Delayed**
FC Planning Speed: **14.1 ms**
FC Final Position Convergence: **Slow**
FC Reaction Speed: **350 ms**

- **Why?** Test visual servoing accuracy in a controlled, low-risk setup.
- **Safety?** No obstacle avoidance → added fail-safe and boundary limits.
- **What?** Drone moved in fixed 1 cm increments based on pixel error.
- **Result?** System successfully converted pixel offsets into consistent motion.

Slow Tracking Follow (Minimal Overshoot)

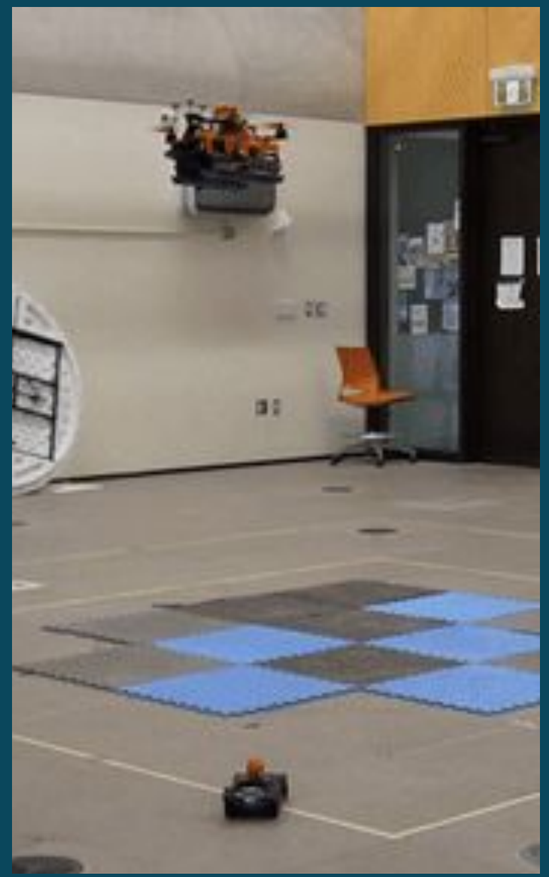


DRONE TRACKING SLOW MOVEMENT (P-CONTROL)

Object Detection TPR: **95%**
Object Detection FPR: **3%**
Object Detection Robustness: **95%**
Object Detection Rate: **30Hz**
Camera Data Rate: **30Hz**
Drone Processing Latency: **30.5 ms**
FC Speed: **Not Satisfied**
FC Tracking Precision: **Overshoots**
FC Planning Speed: **14.1 ms**
FC Final Position Convergence: **Oscillates**
FC Reaction Speed: **350 ms**

- **Why?** Try to improve tracking precision and convergence, using scaled moves.
- **What?** Introduce P-Control to scale movement with size of error.
- **Result?** Exhibits overshooting and oscillatory behaviour, other metrics are not affected.

P-Control Shows Significant Overshooting



DRONE TRACKING SLOW MOVEMENT (PD-CONTROL)

Object Detection TPR: **95%**
Object Detection FPR: **3%**
Object Detection Robustness: **95%**
Object Detection Rate: **30Hz**
Camera Data Rate: **30Hz**
Drone Processing Latency: **30.5 ms**
FC Speed: **Not Satisfied**
FC Tracking Precision: **18.3 px**
FC Planning Speed: **14.1 ms**
FC Final Position Convergence: **5.96 cm**
FC Reaction Speed: **350 ms**

- **Why?** Try to correct overshooting and oscillatory behaviour, we want to dampen velocity approaching the ball.
- **What?** Introduce PD-Control to dampen the system.
- **Result?** Overshooting is less bad, requirements for tracking and convergence metrics are satisfied.

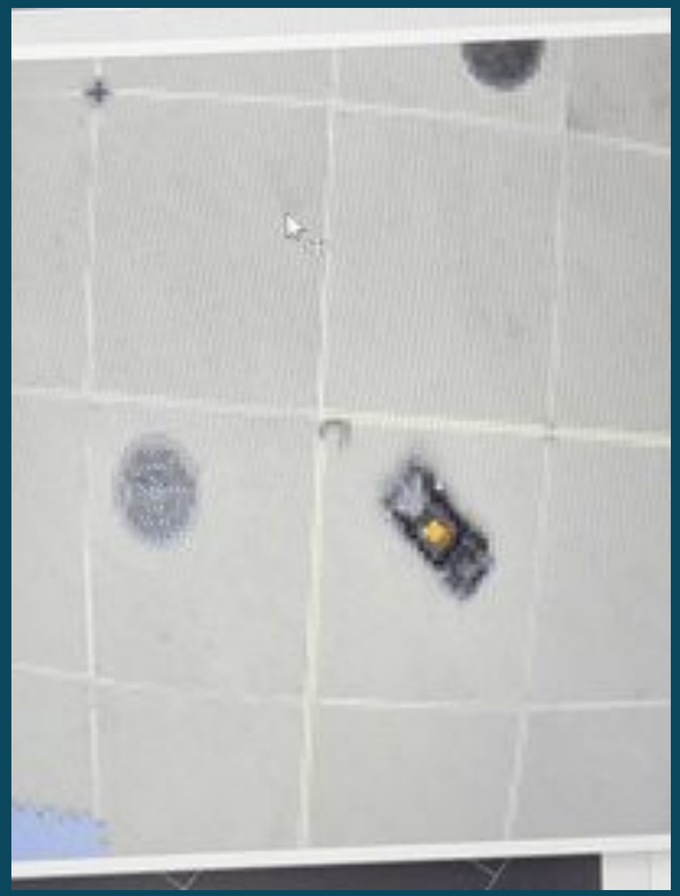
PD-Control Shows Better Stability

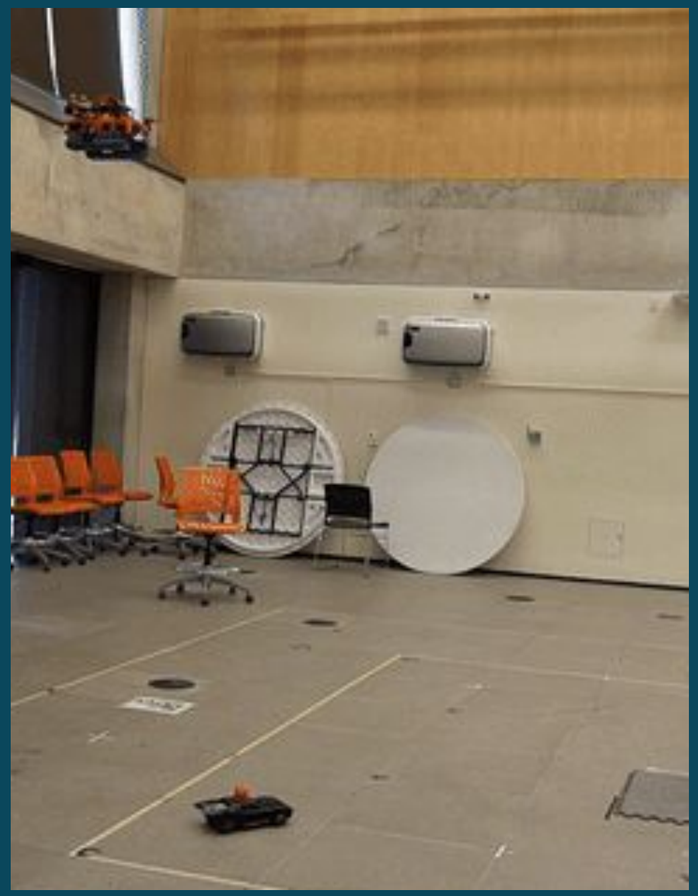
DRONE TRACKING FASTER MOVEMENT (PD-CONTROL)

Object Detection TPR: **95%**
Object Detection FPR: **3%**
Object Detection Robustness: **95%**
Object Detection Rate: **30Hz**
Camera Data Rate: **30Hz**
Drone Processing Latency: **30.5 ms**
FC Speed: **Satisfied**
FC Tracking Precision: **18.3 px**
FC Planning Speed: **14.1 ms**
FC Final Position Convergence: **5.96 cm**
FC Reaction Speed: **350 ms**

- **Why?** Moving up to using catapult, need to increase ball speed tolerance.
- **What?** Increasing car speed to test detection and tracking accuracy.
- **Result?** System successfully tracks golf ball without affecting existing metrics.

PD-Control With Faster Car Acceleration



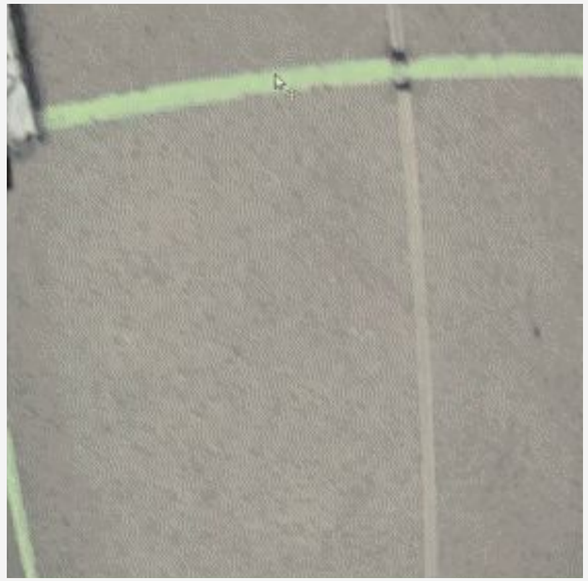


DRONE TRACKING CATAPULT LAUNCH (PD-CONTROL) - MVP

Object Detection TPR: **95%**
Object Detection FPR: **3%**
Object Detection Robustness: **95%**
Object Detection Rate: **30Hz**
Camera Data Rate: **30Hz**
Drone Processing Latency: **30.5 ms**
FC Speed: **Satisfied**
FC Tracking Precision: **44.2 px**
FC Planning Speed: **14.1 ms**
FC Final Position Convergence: **5.96 cm**
FC Reaction Speed: **350 ms**

- **Why?** See if MVP is achieved after all metrics are satisfied with car tests.
- **What?** Replace car with catapult and evaluate detection and tracking.
- **Result?** System successfully tracks golf ball without affecting main existing metrics, tracking precision decreases due to fast acceleration ball launch.

MVP – Full Catapult (Golf Proxy) Test



Gains are Twice too big

Starting Offset:

- Test command starts at frame 00243
- GT: x = 1.97 y = 2.01
- Vicon: x = 2.35 y = 2.13
- Offset: x = -0.38 y = -0.12

Position 1:

- GT: x = 1.26 y = 0.62
- Start frame at 00393
- End frame at 00487
- Avg: x = 1.65 y = 0.68
- Corrected: x = 1.27 y = 0.56
- SS Error (Euclidean): 6.08 cm

Position 2:

- GT: x = -1.21 y = -1.19
- Start frame at 00705
- End fram at 00857
- Avg: x = -0.78 y = -1.10
- Coorected = -1.16 y = -1.22
- SS Error (Euclidean): 5.83 cm

Average SS Error: 5.96 cm

Detection Accuracy: Object Detection TPR
False Positives (merge with background complexity): Objectd Dedection FPR
Lighting Robustness: **(object detection robustness)**
Segmentation FPS: **Object detection rate**
Camera Frame Rate: camera data rate
Processing Latency: E2E (in code only)
Ball Speed Tolerance: **Flight control speed**
Tracking Precision: Flight Control Tracking Precision
Tracking Latency: Flight Control Planning Speed (PD Time)
Convergence: flight control final position convergence
Flight Control Reaction Speed: E2E (look at video frames for when drone moves)

EVALUATION AND DISCUSSION

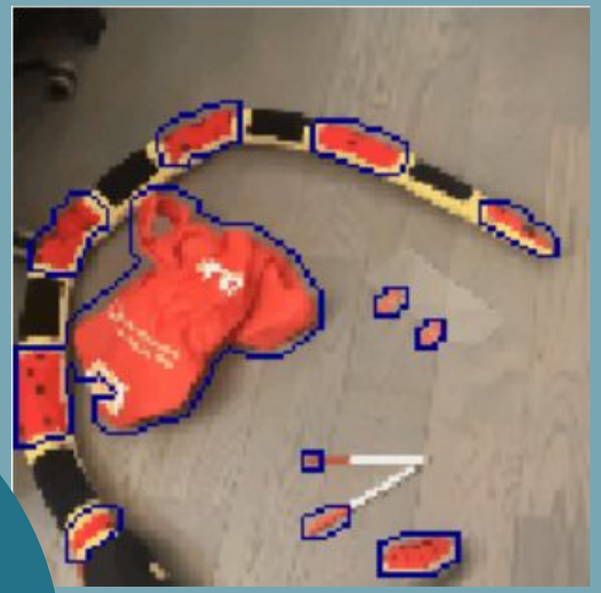
TESTING AND OBJECTIVES ACHIEVED

Performance Requirement Code/Name	MVP Metric	MVP Result
IMG-SWE-001: Camera Data Rate	**> 20 Hz**	**30 Hz**
OBD-SWE-003: Object Detection Rate	**> 20 Hz**	**30 Hz**
OBD-SWE-004: Object Detection TPR	**> 90%**	**95%**
OBD-SWE-005: Object Detection FPR	**< 5%**	**3%**
OBD-SWE-006: Object Detection Robustness	**> 85%**	**95%**
FLC-SWE-001: Flight Control Planning Speed	**< 20 ms**	**14.1 ms**
FLC-SWE-004: Flight Control Reaction Speed	**< 150 ms**	**350 ms**
FLC-SWE-005: Flight Control Tracking Precision	**< 40 pxls**	**44.2 pxls**
FLC-MEC-001: Flight Control Speed	**TRUE**	**TRUE**
SLR-001: Drone Processing Latency	**< 100 ms**	**30.5 ms**

CHALLENGES, LIMITATIONS, AND NEXT STEPS

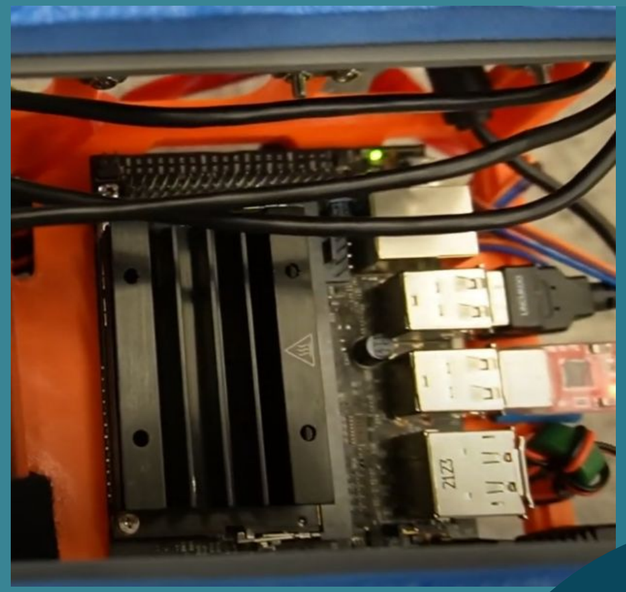
Challenges:

- Stochastic errors after crashes.
- Handling false positives after switching to red filter detector.
- Tuning PD controller.



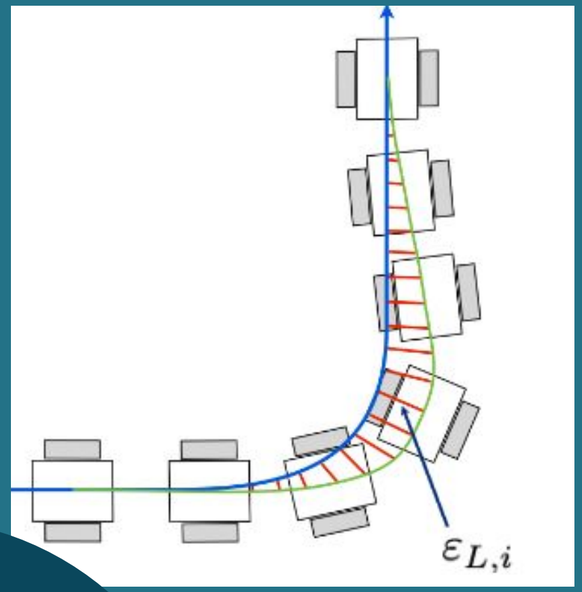
Limitations:

- Drone hardware.
- Camera frame rate.
- Image resolution.



Next Steps:

- Better hardware.
- MPC for better tracking.
- More robust tracker to ignore other golf balls.



CHALLENGES, LIMITATIONS, AND NEXT STEPS

Challenges:

- Fixing our guy after he crashes and debugging where issues lied (realsense stops working, mavros can't match vision pose and setpoint position???)
- Handling false positives after switching to red filter detector (small pieces of tape on the floor was heavily tripping us up)
- Tuning our PD controller. Figuring out the correct proportional control and derivative to fly well, since testing was so limited.

Limitations:

- Drone Hardware (battery dying on us)
- Testing - flight time, needing to be ready for a window, sometimes not getting to
- Frame rate limitations - the camera we chose
- Pixel resolution - more would be great, unfortunately above 256- it stops publishes in a timely way
- processing speeds
- compute could be better
- mounting the camera fixed and it was maybe slightly tilted

Next Steps:

- Get better hardware and try to yolo model again, especially because we fixed frame rate later, there's a chance this could work, that was processing at (67ms delay though inference)
- Determining future golf ball motion with golf ball model and using MPC for better tracking
- Using more robust tracker to ignore other golf balls in frame (eseically useful in the real world, many people playing golf)
- eigen, since the red ball gets smushed and we can see the direction and is


PARSIGHT
FLYRS ROB498 CAPSTONE

All Requirements:

https://docs.google.com/spreadsheets/d/10_khb6EEwtCNaeIO_rd5POkCiUF0DmYJf3SWBCPfAFk/edit?usp=sharing