F1TENTH Autonomous Racing

Spring 2023 Project Presentations

May 9, 2023

Penn Engineering

ZOOM

https://upenn.zoom.us/j/95810963853?pwd=TkZxN2JnRlU0bElmSzVFcUIrenJ6dz09

Penn Engineering

Team 0

Title and team members

Penn Engineering

Team 0 - slides…

Teams add slides after this one

Pitch-Controlled Landing for Airborne Ground Vehicles

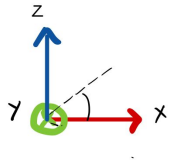
Team 1: Hot Wheels

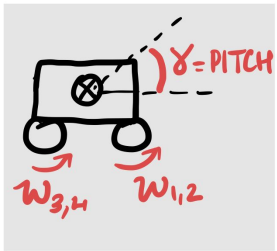
Manasa Sathyan, Mengti Sun, Nicholas Gurnard, Rithwik Udayagiri

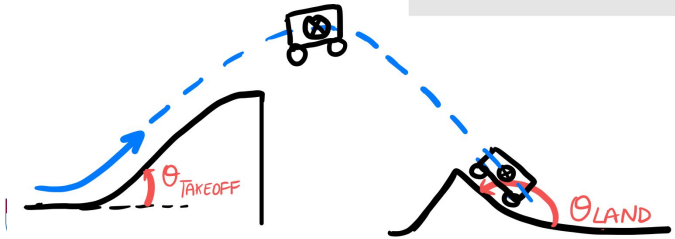
Penn Engineering

Overview

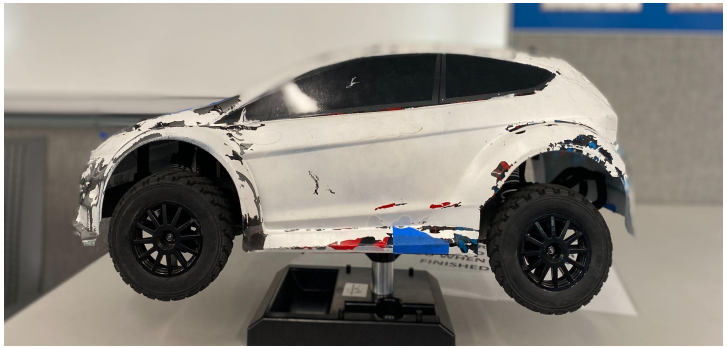
- Take Autonomous Racing outdoors with different terrains
- Control the pitch of the car for optimal landing







Live Demo!!!



Model Dynamics

$$4(I_w\omega_w) + 2(I_a\omega_a) = I_c\omega_c$$

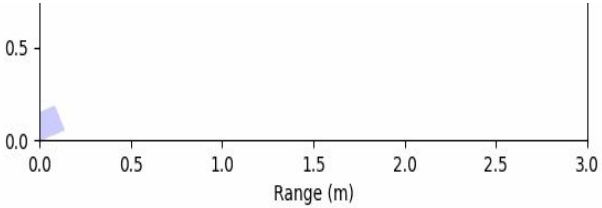
$$\omega_c = \frac{v_c}{r_w}\frac{12 \times 2(m_w r_w^2 + 0.5 m_a r_a^2)}{m_c(l^2 + h^2)}$$

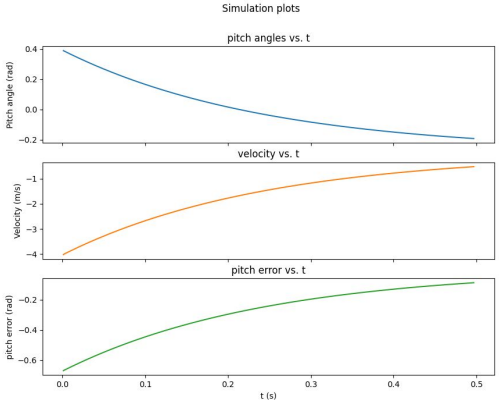
Approach - Simulator

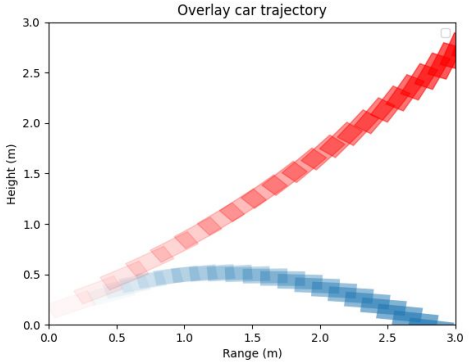
- Calculated pitch error based on the previous commanded velocity (derive the angular velocity based on the dynamical model) and the time difference between them.

$$\psi_t = \psi_{t-1} + \omega_{t-1} dt$$

- Simulated 2D trajectory based on multiple initial angle, take-off velocity and PID values for feasibility analysis.







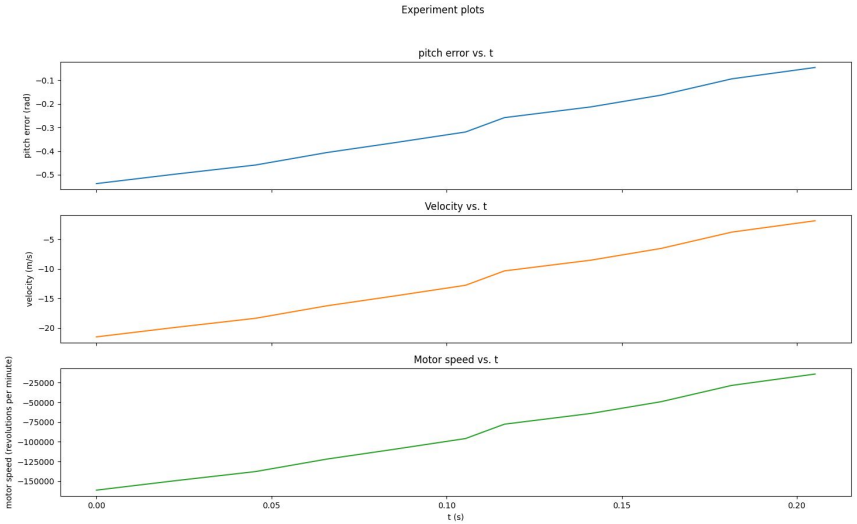
Simulated trajectory with initial angle 22 degrees, take-off velocity 6.5m/s. The left-hand side is the plots of pitch angles, velocity and pitch error with PID control. In the bottom image of right-hand side, the red patches simulate trajectory with no PID and velocity 3.0m/s in the air. The blue patches simulate trajectory with PID control, kp=6.0, kd=1e-5, ki=1e-5

Approach - Real Car

- Safety is #1
  - Removed LiDAR
  - Helmet for car
  - Excessive padding
- Ran multiple tests on different surfaces and collected rosbags

Experiments

- No pitch Control
- Pitch Control - landing onto crash pad
- Pitch Control - landing onto a ramp



Experimental parameters:

- Initial angle: 23 degrees
- Landing angle: -20 degrees
- Distance between ramps: 2.34m
- Take-off velocity: 6.5m/s
- Kp: 40, kd:1e-5, ki: 1e-5

Challenges

- Human errors due to lack of localization (no LiDAR)
- Time consuming experimental setup
- Ensuring safety of the F1Tenth car
- Weather, for outdoor testing

Future Work

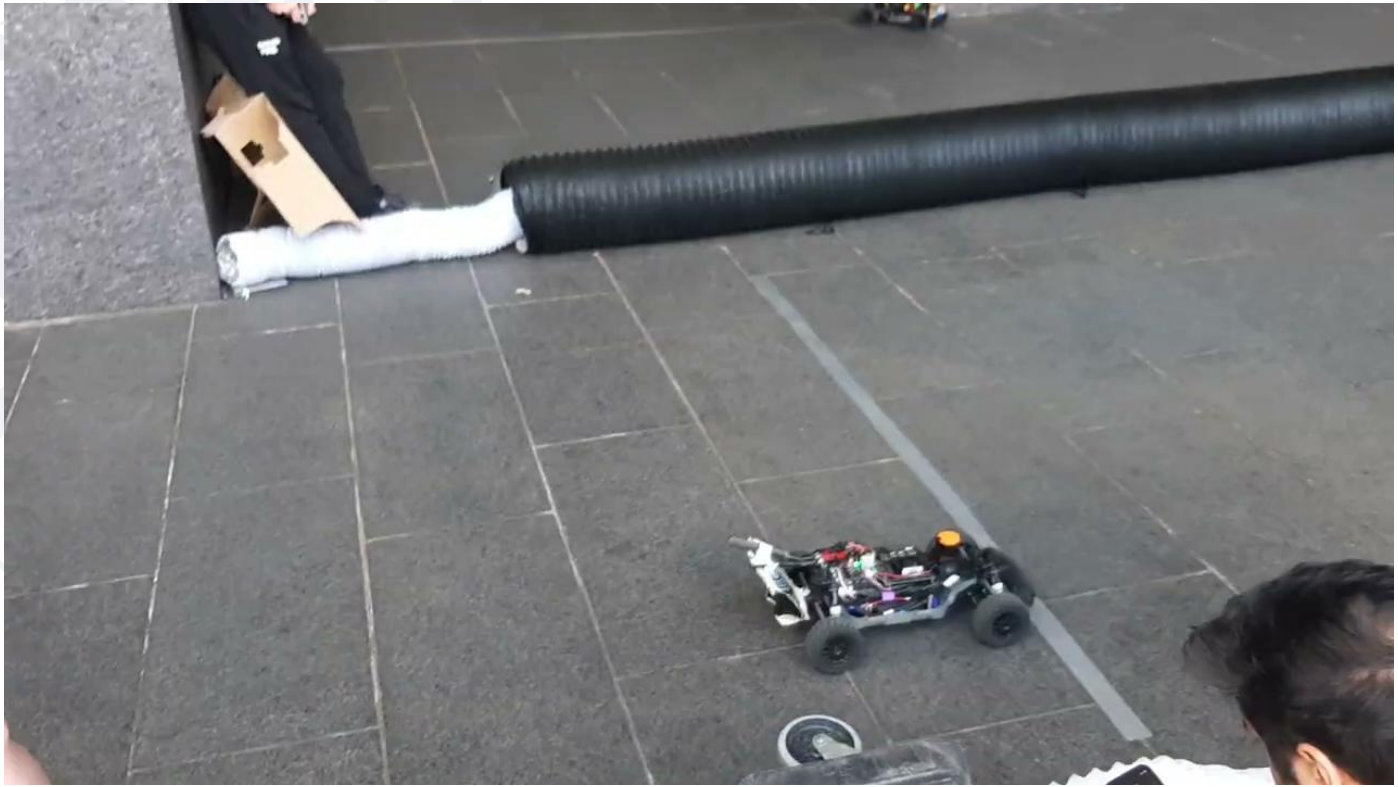
- Learn dynamics of the system
- More Complex Control Strategies
  - MPC
  - LQR
- State Estimation using EKF, UKF
- Full autonomous jumps in race

Team 2

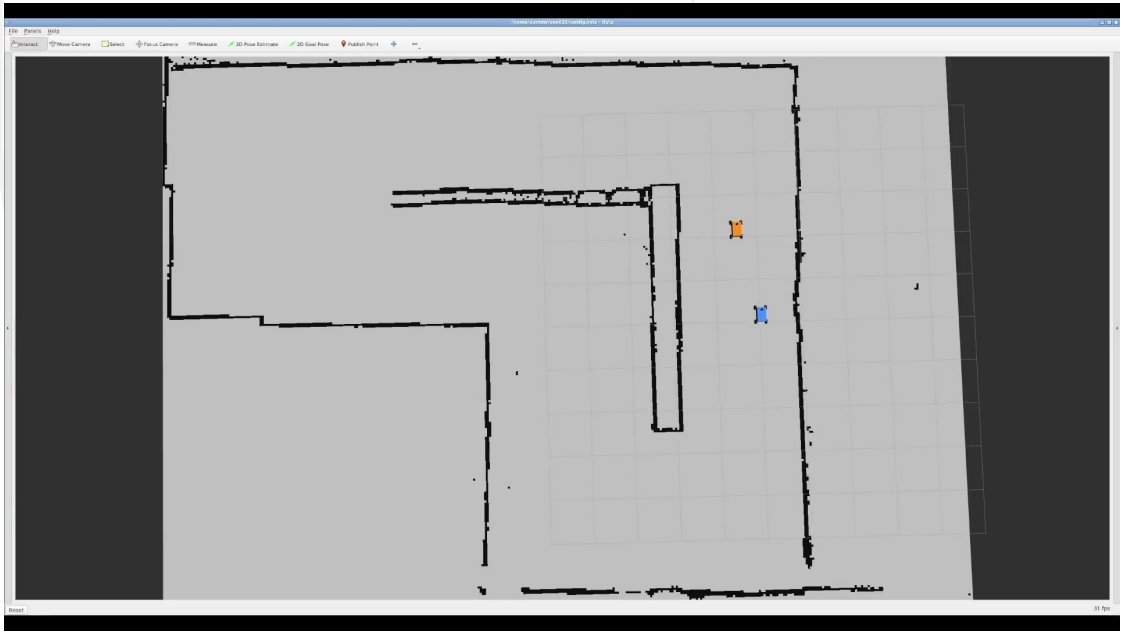
Strategies for Multi-Agent Races

Penn Engineering

Recall what often happened in race 3…



This is how the races could have looked



We break down this into three subsystems

Obstacle Detection

High Level Planning

Low Level Control

Obstacle Detection

High Level Planning

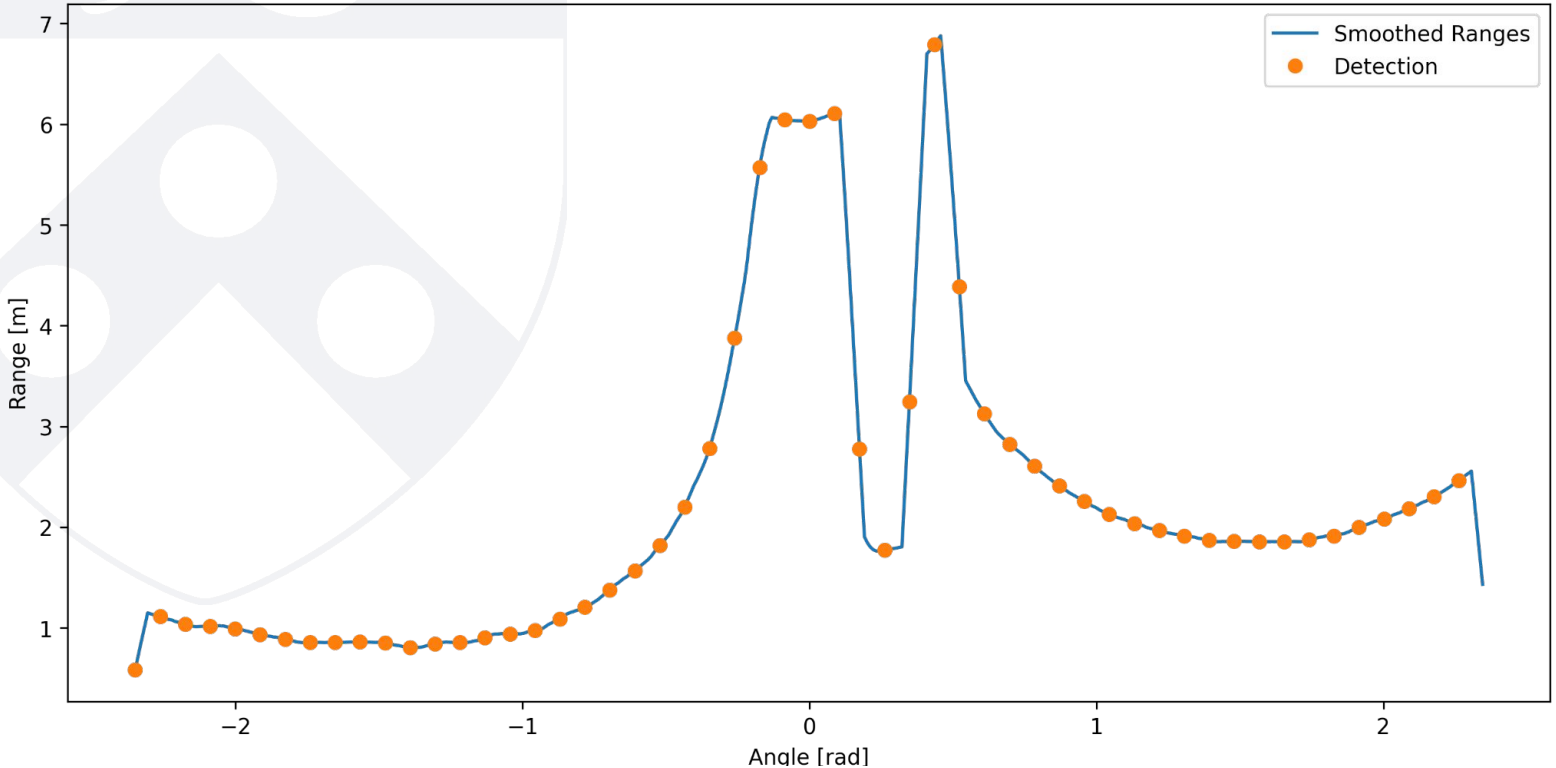
Low Level Control

Downsampling Based Detection

End up with
>30 detections

$$R \approx d \sin(M \Delta\theta)$$

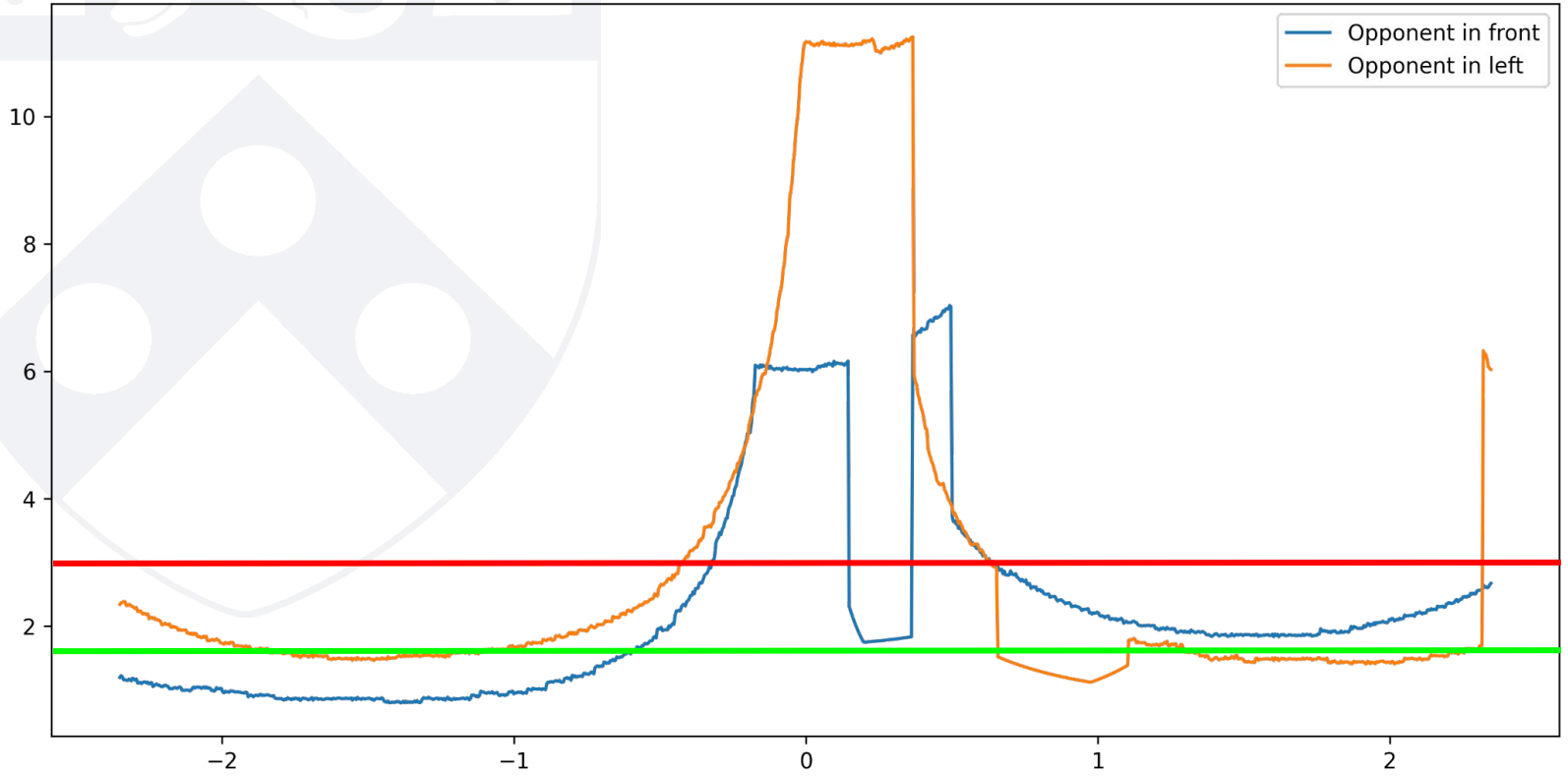
Downsampling Based Detection



Threshold Based Detection

- Inspired by lab 3
- Pick a threshold, eg. 3m
- Consider anything below threshold occupied
- Find rising and falling edges in thresholding

Threshold Based Detection

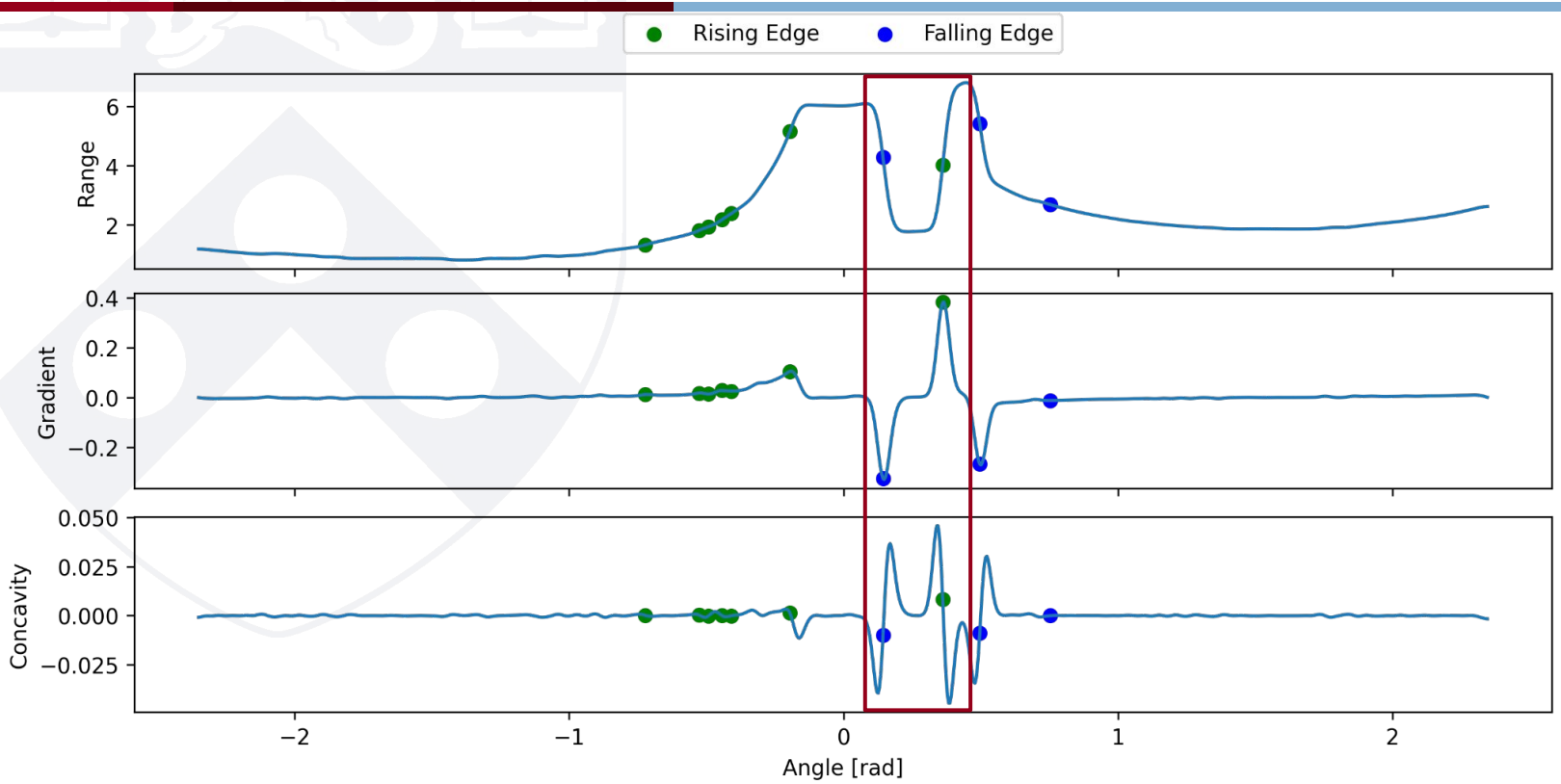


Edge Detection

We can compute edges in three simple steps.

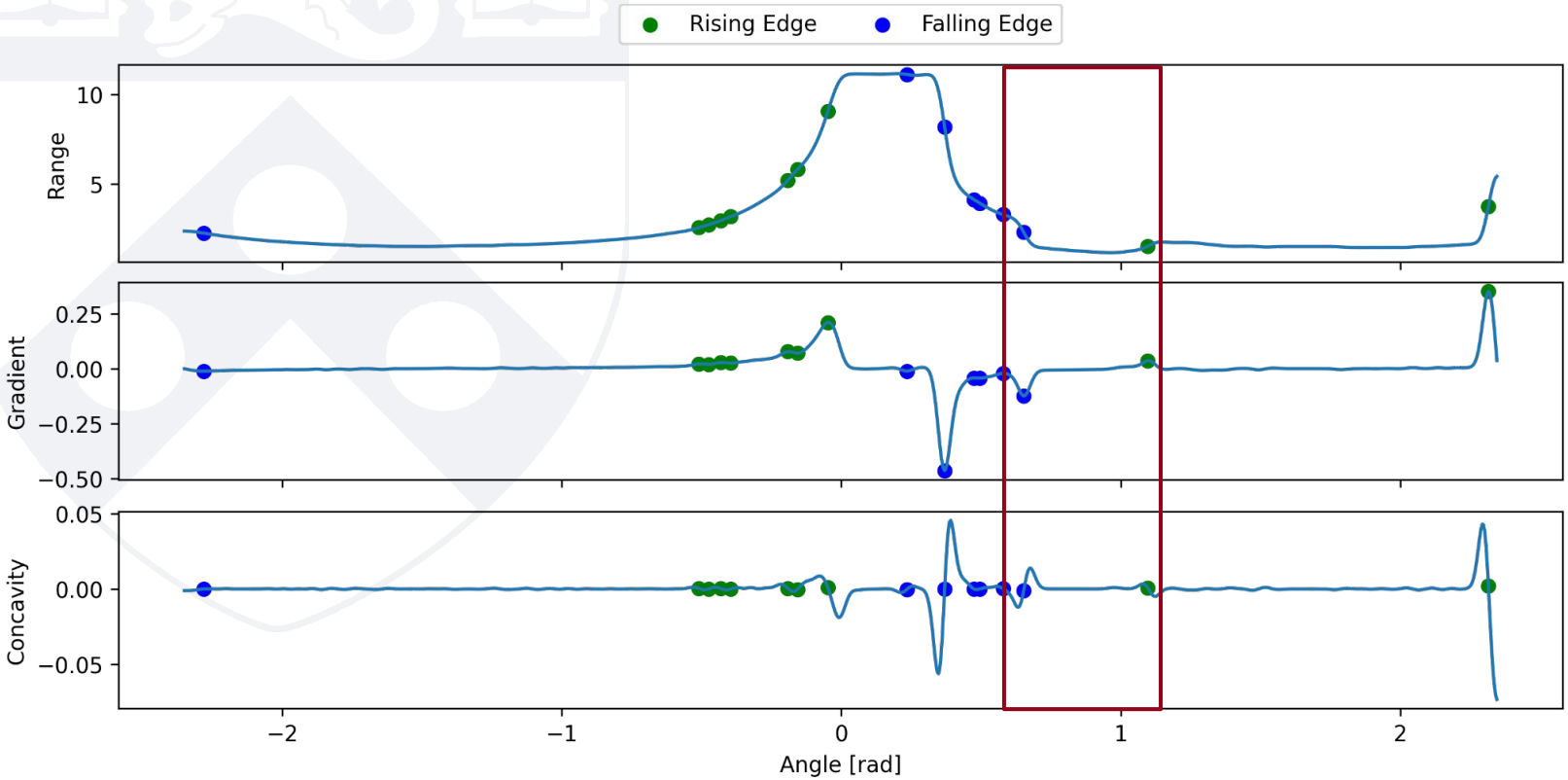
1. Apply Gaussian filter.
2. Find all 2nd derivative zero-crossings.
3. Find all zero-crossings with sufficiently high derivative.

When car is in front

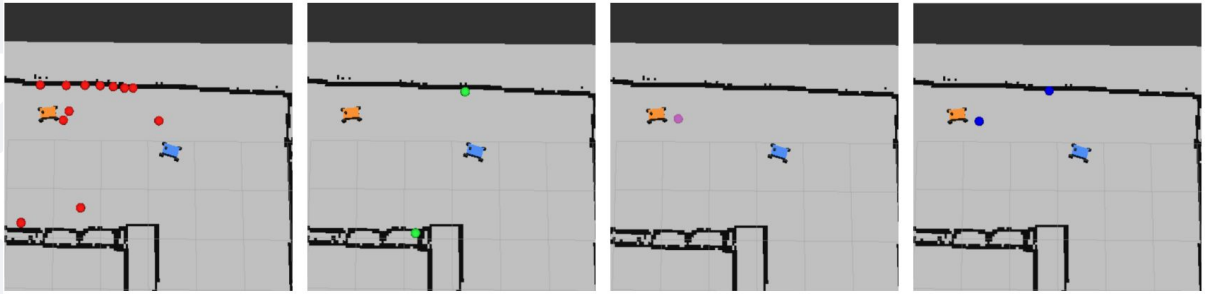


Penn Engineering

When car is to the left

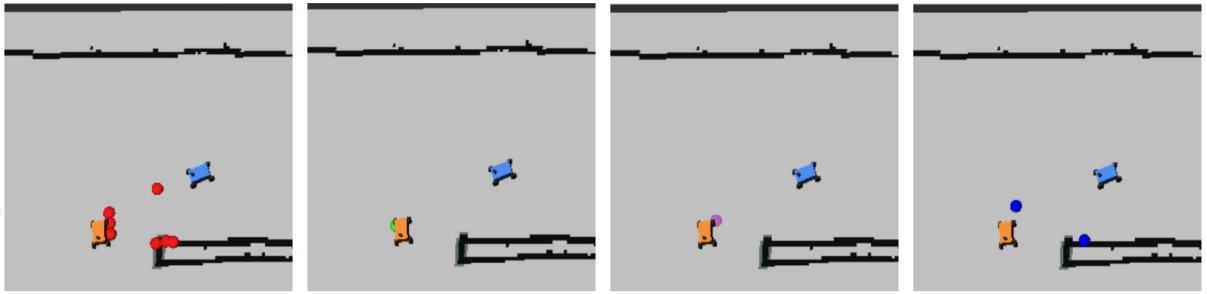


Comparison of approaches



(a) Downsampling based (b) Threshold based (c) Edge extraction based (d) Peak deteciton based

Fig. 3: Comparison of different obstacle detection algorithms at turn 1 in simulation on the Skirkanich map.



(a) Downsampling based (b) Threshold based (c) Edge extraction based (d) Peak deteciton based

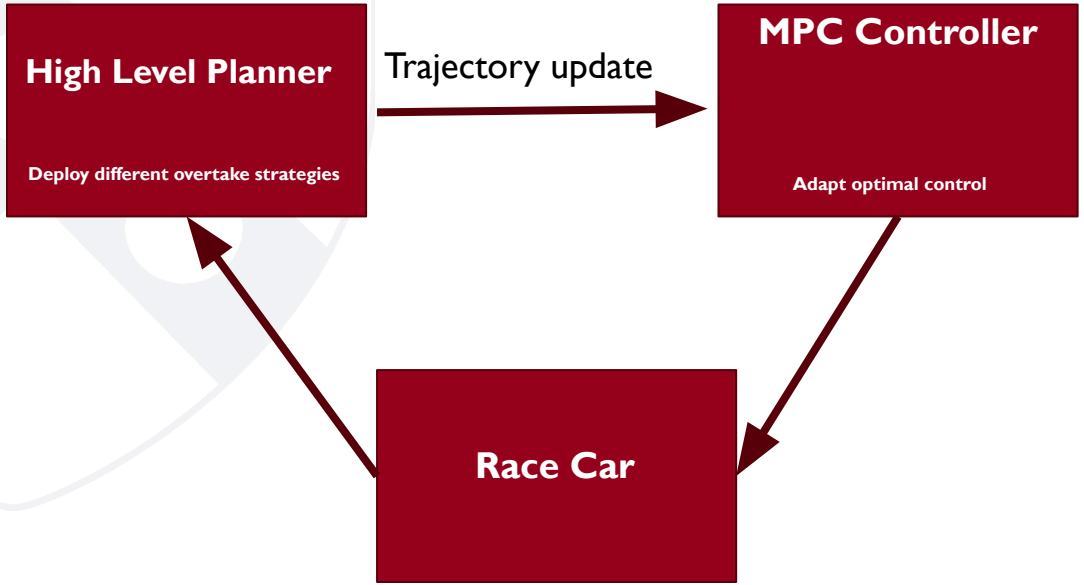
Fig. 4: Comparison of different obstacle detection algorithms at turn 2 in simulation on the Skirkanich map.

Obstacle Detection

High Level Planning

Low Level Control

High Level Planning



High Level Planning

RRT

Switch into RRT mode when detect obstacles

Dual lines

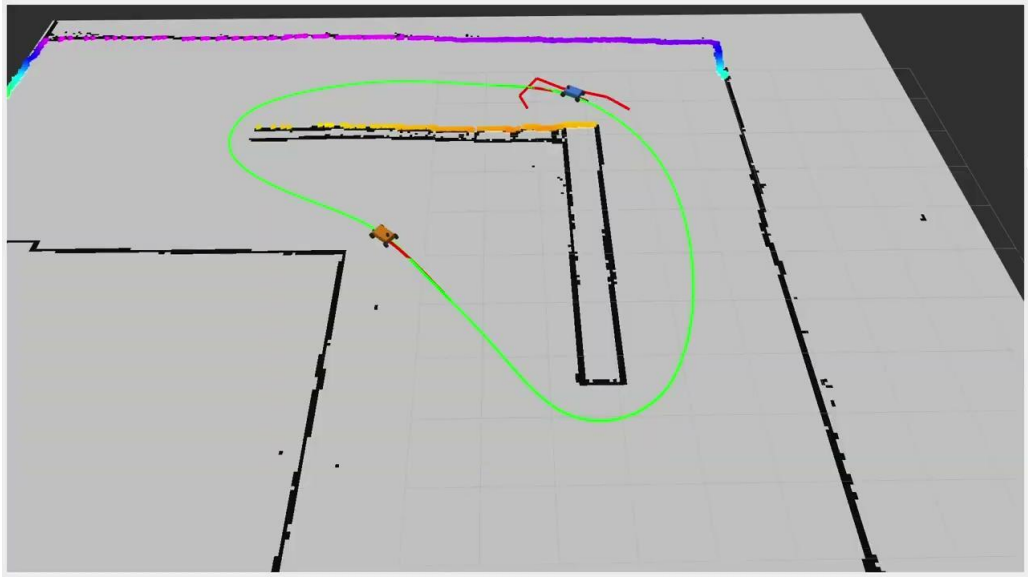
Alternate trajectories when close to obstacles

Graph-based planner

Online trajectory generation based on acquired obstacles

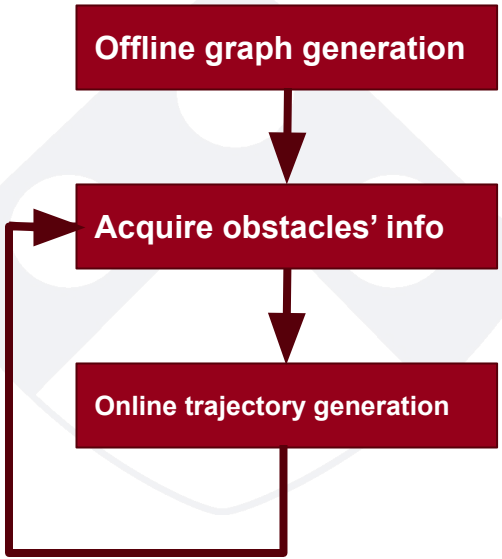
RRT* Overtake

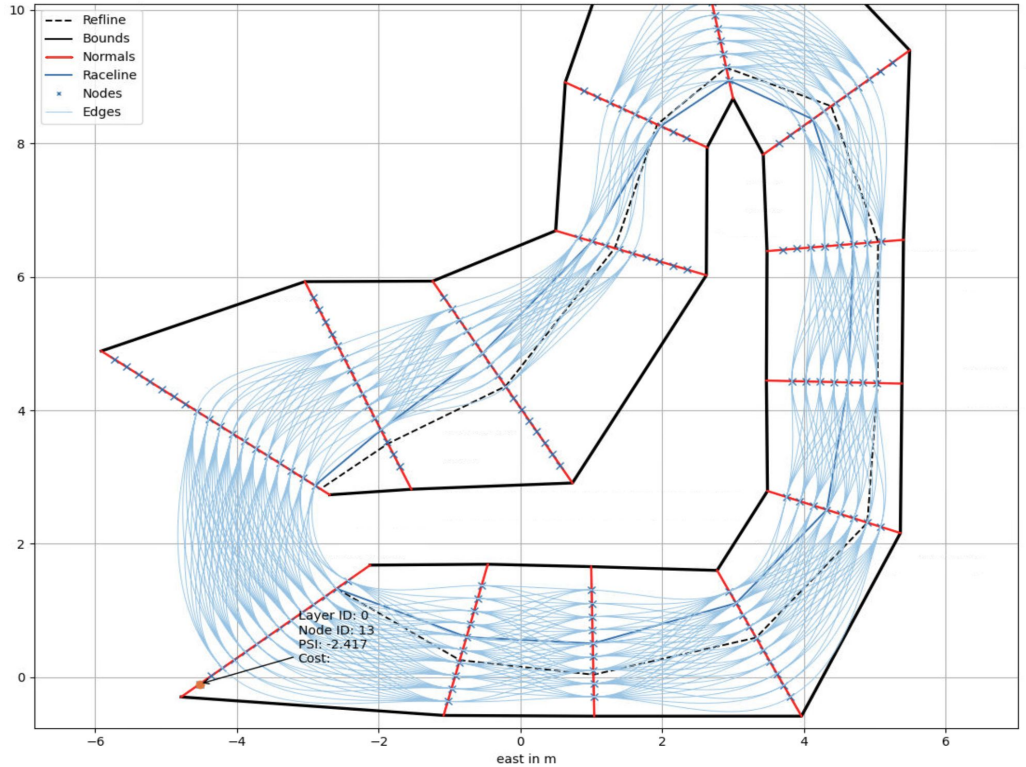
- The ego and opponent car start on an optimal centerline trajectory using MPC, opponent car set at 80% speed
- If the ego car is within threshold distance of opponent switch planner to RRT* with local occupancy map
- Once ego car is > threshold distance away from oppo Switch back to MPC



Penn Engineering

Graph-Based Planner





Penn Engineering

[1] Tim Stahl et al. "Multilayer Graph-Based Trajectory Planning for Race Vehicles in Dynamic Scenarios". In: 2019 IEEE Intelligent Transportation Systems Confernce (ITSC). 2019, pp. 3149–3154.

Offline Graph Generation

- Input a reference trajectory under Frenet frame
- Split trajectory into a number of layers laterally within the bound
- Generate nodes along each layer
- Connect nodes across layers that satisfy dynamic constraints
- Assign cost to each edge

Online Trajectory Generation

- Extract current position on the offline graph
- Remove edges/node occupied by obstacles and potentially occupied
- Search shortest path from current node to a desired planning distance with minimum cost

$$\underset{n_i \in L_l, n_j \in L_{l+1}}{\operatorname{argmin}} \sum_{l=l_s}^{l_e-1} c_{i,j}. \quad [1]$$

- Velocity calculation using forward-backwards solver

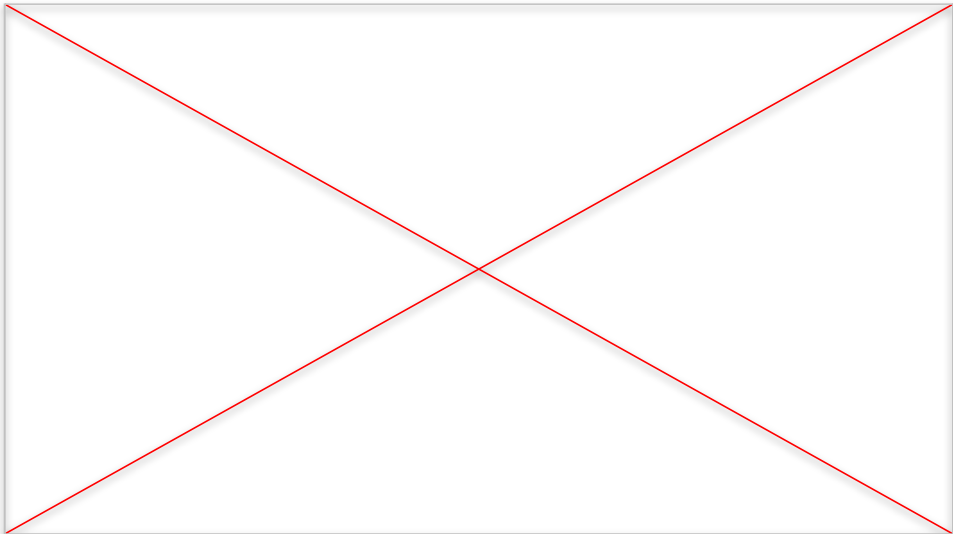
Dual Raceline

- An optimal inner, outer and unrestricted center raceline trajectory are generated offline
- Inner: red, central: green, outer: blue
- The ego and opponent car start on an optimal centerline using MPC to follow the trajectory
- Opponent car set at 80% speed

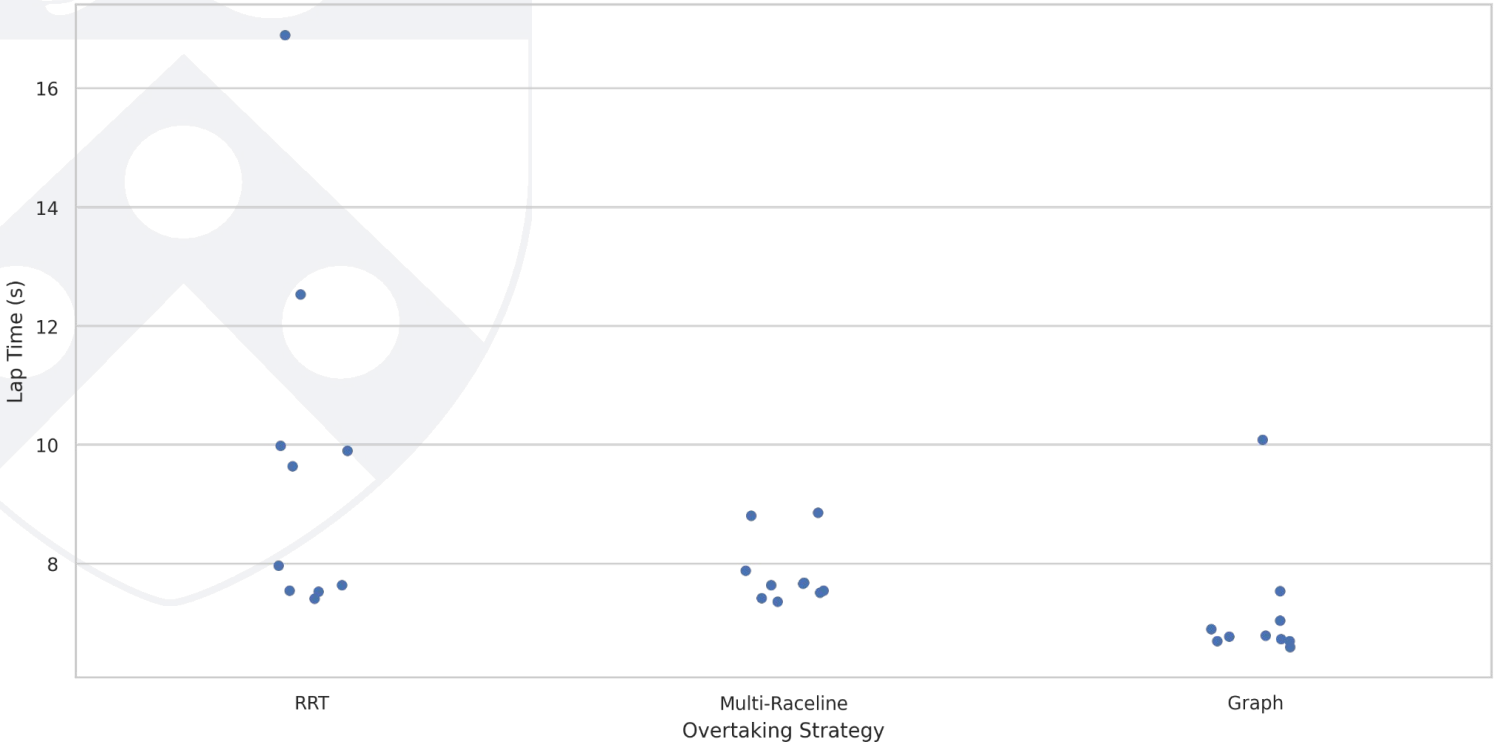
Dual Raceline Algorithm

- If ego is within some threshold distance of opponent:
  - For the next n waypoints on the centerline trajectory
  - Find closest waypoint position in both inner and outer lines
  - Find euclidean distance between center points and inner, outer points
- If min distance inner > threshold
  - Change to inner trajectory
- Elif min distance outer > threshold
  - Change to outer trajectory
- Else match oppo speed and wait

- Hysteresis on passing trajectory
- Timed cooldown on pass attempt



Overtaking Strategy Comparison

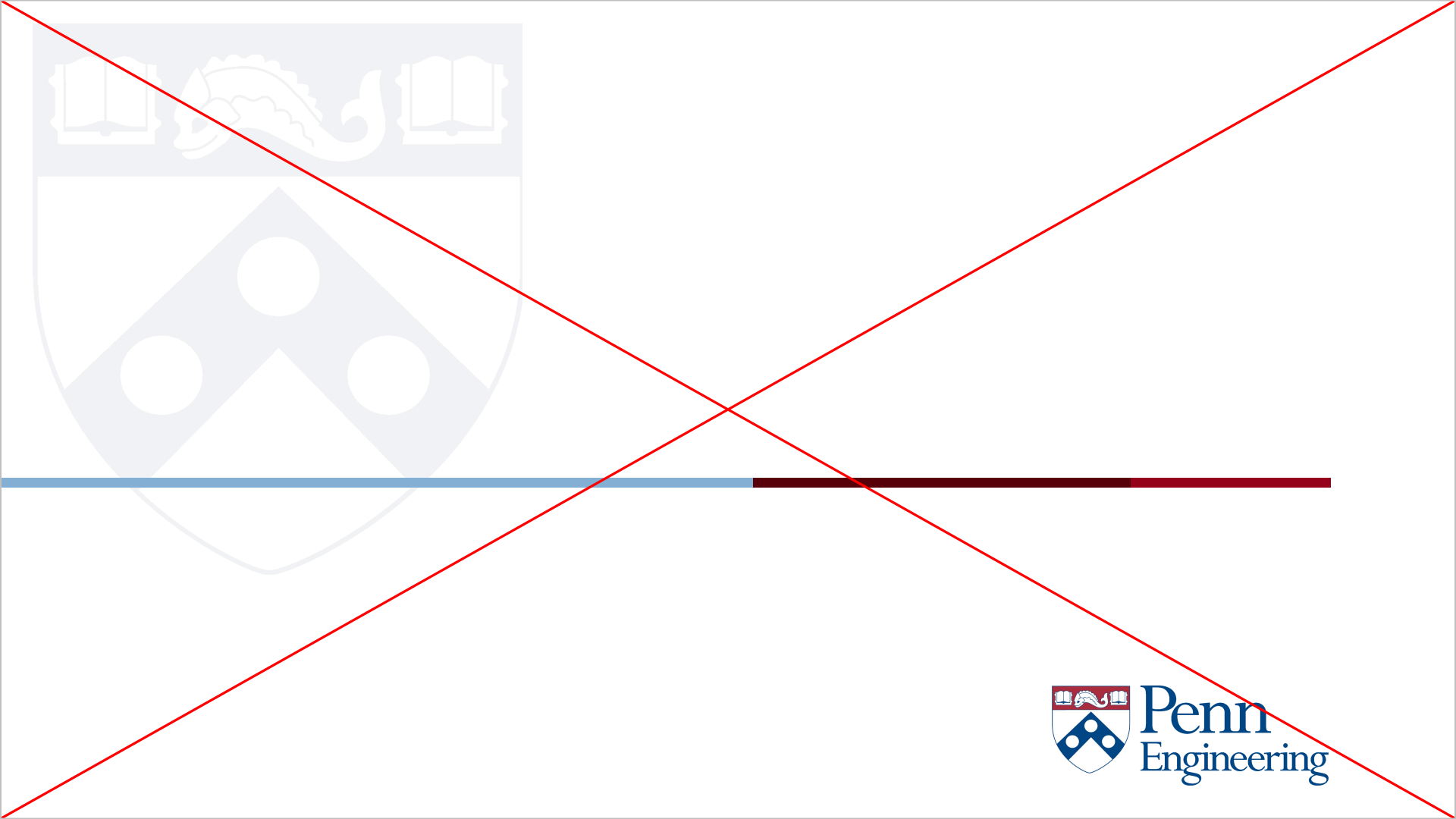


Penn Engineering

Comparison

Planner	RRT	Dual lines	Graph planner
Time(10 laps)	1:37.12	1:18.37	1:11.85
Collision	2	0	1

* Graph based planner does not respect the system dynamics, like the dual-racelines, which were created using global raceline optimization.


Penn
Engineering

Team 3

Range Finder Disciplined Monocular Depth Estimation

Penn Engineering

Project Objectives

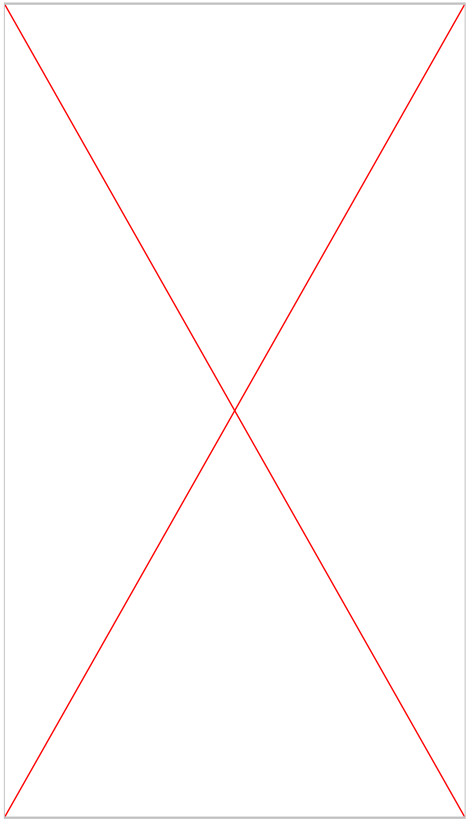
- Lidar is one of the most expensive components on the F1TENTH car
- Develop a lower-cost alternative with usable depth accuracy

- Monocular depth estimation models:
  - Excellent azimuth/elevation accuracy
  - Returns relative depths
- Low-cost laser rangefinders
  - Excellent depth accuracy
  - Azimuth/elevation precision limited to sensor FOV

- Combine strengths of Depth Estimation models and low-cost laser rangefinders through sensor fusion

Summary

- Able to execute follow the gap using vision-based inputs
- Depth Estimation model performed reasonably well
- Sensor fusion outputs currently too “jumpy” to use reliably
  - Can show some degree of absolute depth
  - Would need more integration time to refine implementation

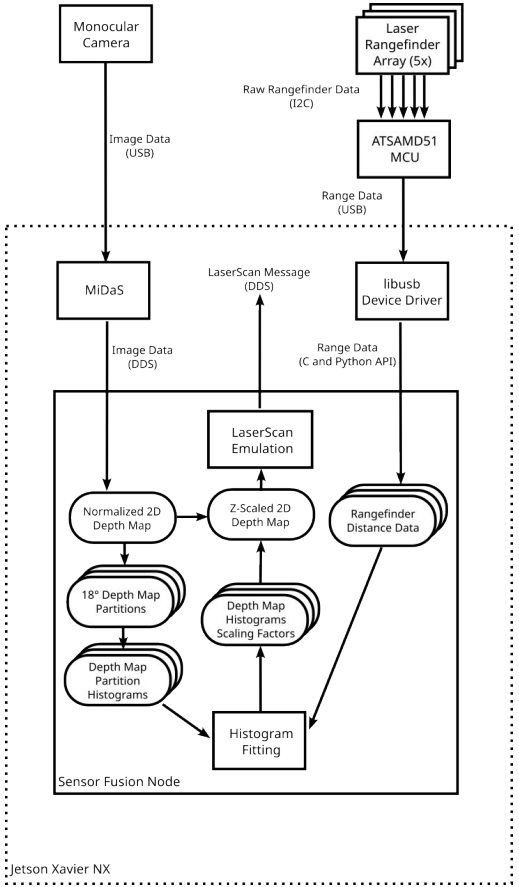


Implementation

Penn Engineering

Sensor Design

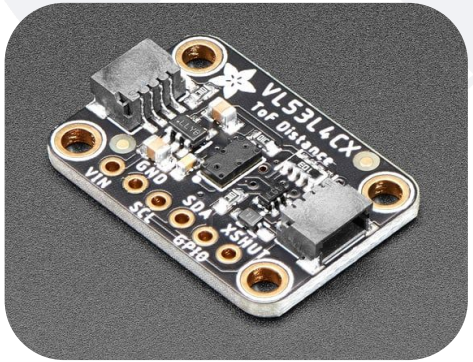
- Receive data from TOF sensor array and camera
- Process camera data with MiDaS
- Provide range measurements via USB
- Fuse MiDaS depth map with direct range measurements
- Emulate a ROS LaserScan message by taking a slice through the fused depth map
- Execute follow-the-gap based on emulated LaserScan

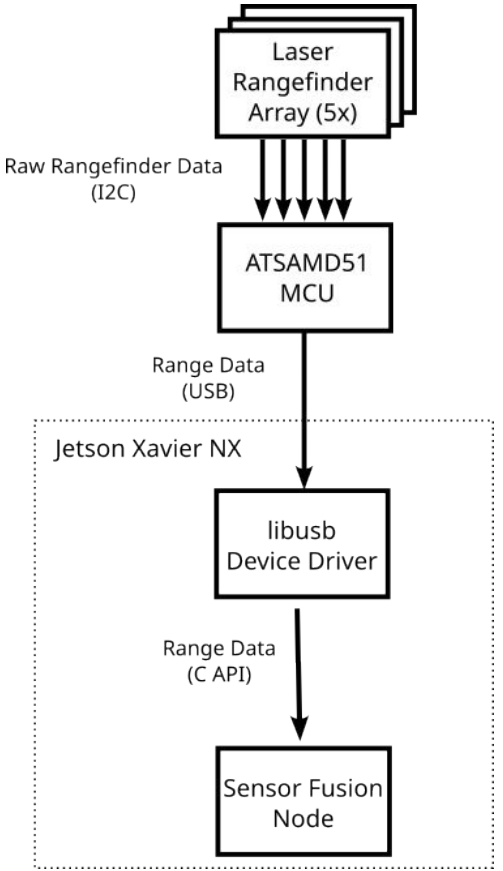


Penn Engineering

VL53L4CX Time-of-Flight Sensor Array

- VL53L4CX: low cost and excellent depth accuracy, but fixed 18-degree FOV
- Array of rangefinders to cover full FOV of the camera
- 3D-printed mount at fixed 18-degree increments
- Sensor –I2C–> SAMD51 –USB–> libusb driver –> fusion node





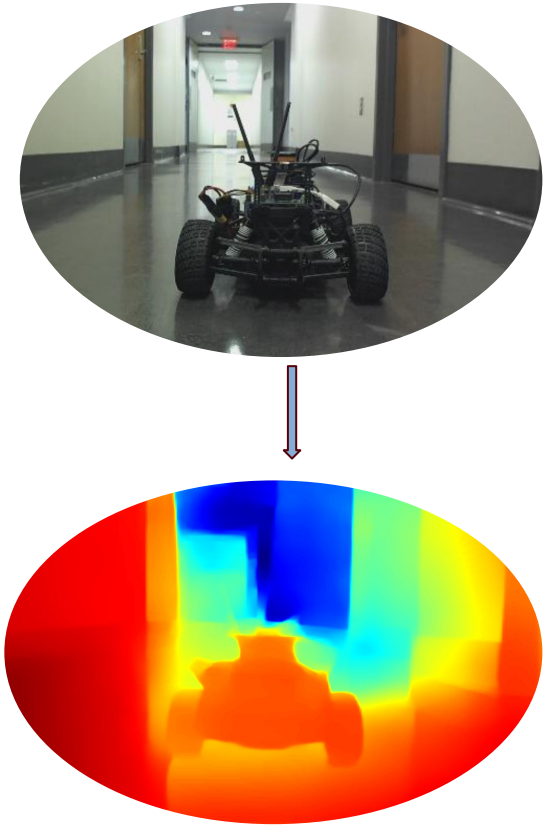
Penn Engineering

Monocular Depth Estimation

Given a Single RGB image, Predict depths of each pixel. Each pixel has a class (0-1) after normalization

Why use Monocular Depths ?

- More Convenient and Cost-effective.
- Stereo depth estimation relies on triangulating points between the two cameras, which can lead to lower resolution in areas with fewer features or where the two views are not well-aligned.
- Monocular depth models can be trained on a wider range of images making it more robust to changes in lighting conditions and other factors.



Adopted Approaches

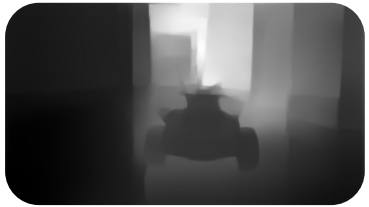
Custom U-Net Model

Input Size: (256,256)

Bad Accuracy !
High Temporal inconsistency

Inference:
Xavier NX: Not Tested

UNet - DensNet-201 Backbone



Input Size: (640,480)

Better Accuracy,
Computationally Expensive !

Inference:
Xavier NX: 4 FPS

MiDaS v2.1 Small

Input Size: (256,256)

Moderate Accuracy,
Light weight & Efficient!

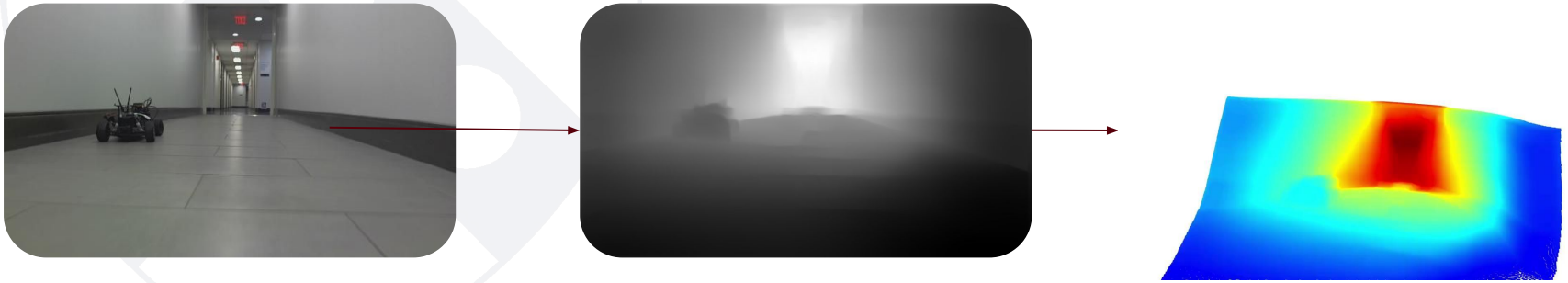
Inference:
Xavier NX: 40 FPS

Transfer Learning -
MiDas v21 small

Built the Levine Dataset with Ground Truth Depth Maps from Real-sense camera for Transfer Learning

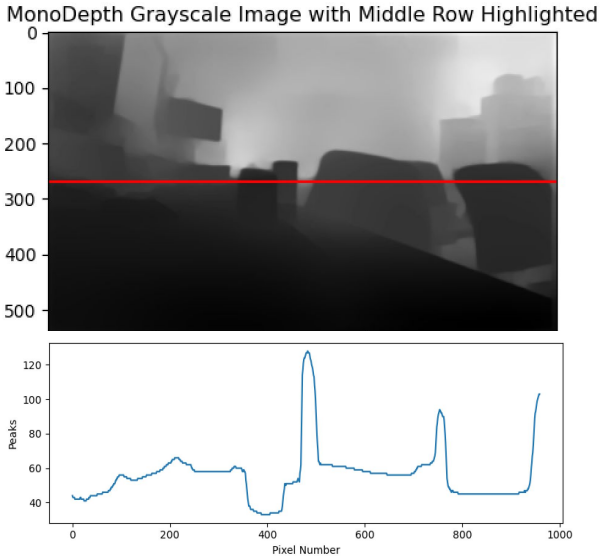
Poor Ground truth depths for re-training :(

Results



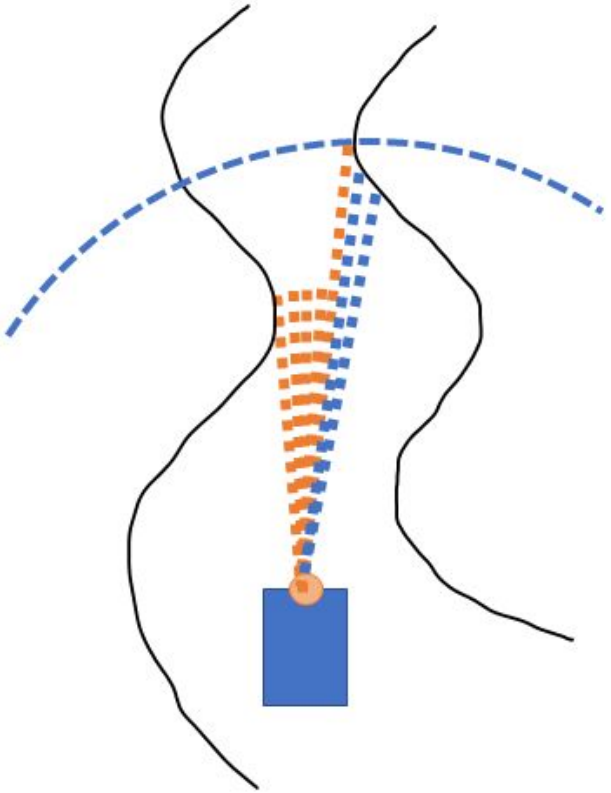
Sensor Fusion: Recovering Absolute Depths

- Step 1 - Correlating Intensity Peaks and Linear Regression
  - Depth Map Overlap
  - Histogram Analysis
  - Obtaining scaling factors and biases for aligning depth map predictions with absolute inverse depth
- Step 2 - Adjusting Depth Values and Sensor Fusion:
  - Adjusting Depth Values
  - Sensor Fusion Algorithm



Gap Following

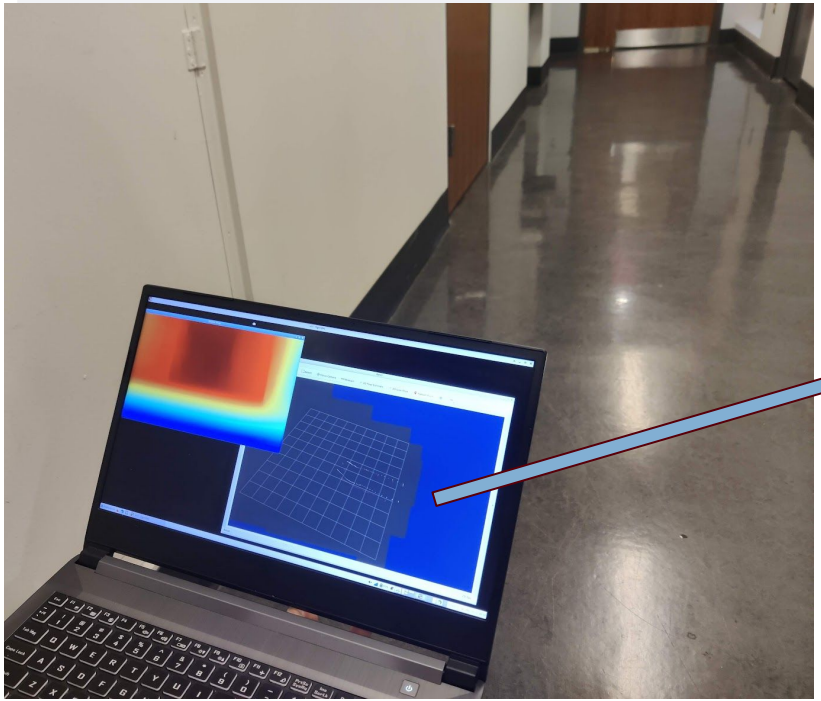
- Replicate 2D lidar data by taking horizontal slice of sensor fusion data
- Input pseudo-lidar data into disparity extension and gap follow algorithm from Lab 4
- Camera FOV of 87° pairs well with gap following in corridors
- Were unsure if disparity extension would work with sensor fusion data, but initial results were promising

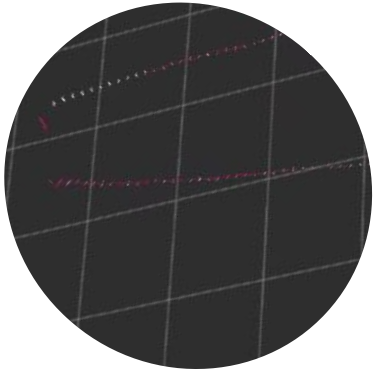


Results !

Penn Engineering

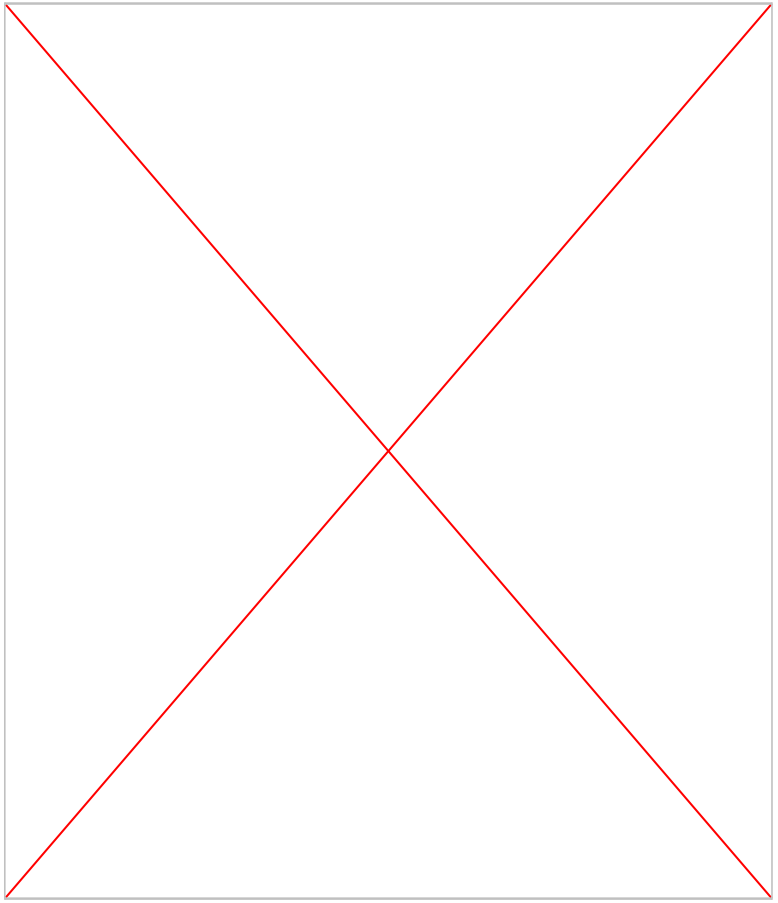
Integrated Follow the Gap





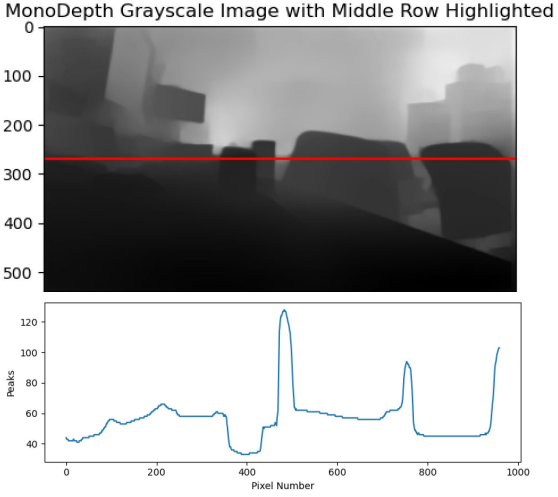
Monocular Depths from MiDaS v2.1

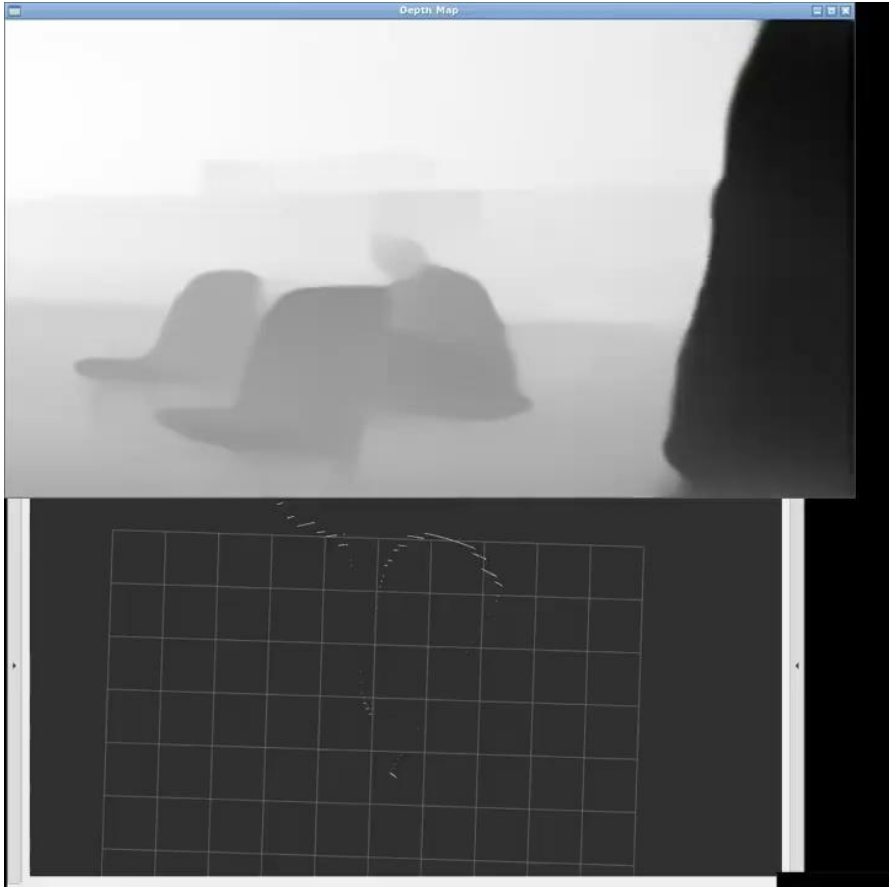
- Somewhat noisy
- “Likes” hands and faces
- Vulnerability to reflections, illusions
- Temporal Inconsistency due to being trained on non consecutive images



Sensor Fusion

- Histogram with smallest point
- Regression approach





Penn Engineering

Time-of-Flight Sensor Performance

- Outputs provided by VL53L4CX were somewhat less "rich" than expected
  - Each sensor typically returns only a single depth, occasionally two or three
    - Fewer sensor fusion inputs
  - Maximum range limited to about 3.5 meters instead of 6 meters
    - Tunable parameters have tradeoff between max range and latency

- F1TENTH car consistently induces failures of I2C interfaces
  - I2C bus locks up when motor activates
  - Unshielded cables of several cm in length between SAMD51 and VL53L4CX breakout board
  - Motors are noisy; likely an EMI problem
  - Limited ability to test fused sensor data on functional platform

Future Work

- Refine association logic in sensor fusion; Possible approaches:
  - Find association producing smallest change in scale
  - Retrain model with rangefinder inputs

- Build Better Dataset for Maps like Levine with a better Depth Camera
- Transfer Learning on Custom Dataset of sequential images for higher accuracy, reducing temporal inconsistency.
- Try better models like DPT-Hybrid, Swin v2.
- Point Cloud Segmentation for Localization based Approaches.

- Eliminate cables for rangefinder array I2C bus; a flex PCB design could potentially:
  - Reduce EMI problems
  - Be more compact mechanically
  - Reduce overall sensor cost ($5 IC vs $15 breakout board)

Bye-Bye to Lidar
Hopefully :)

Mono-cam Localization and Mapping

Team 4

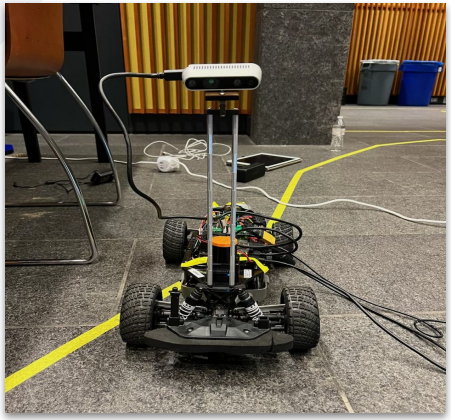
Penn Engineering
University of Pennsylvania

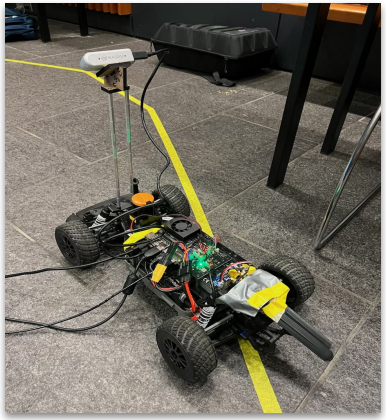
Objective

- Accessible and cost-effective alternative to LiDAR for autonomous racing applications
- Develop a framework for vision based racing for F1/tenth system

Hardware Setup

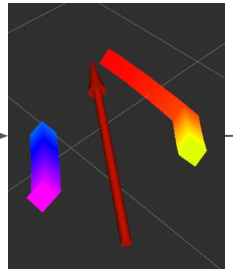
- The camera is mounted at a height of ~40cm for higher field of view.
- The camera also has a deliberate pitch down which is calculated during calibration.





Penn Engineering

Mono-Cam Localization and Mapping (MCLM Pipeline)



Particle Filter

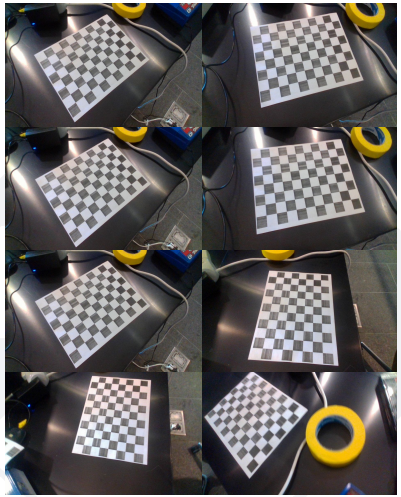
Inferred Pose

Odometry

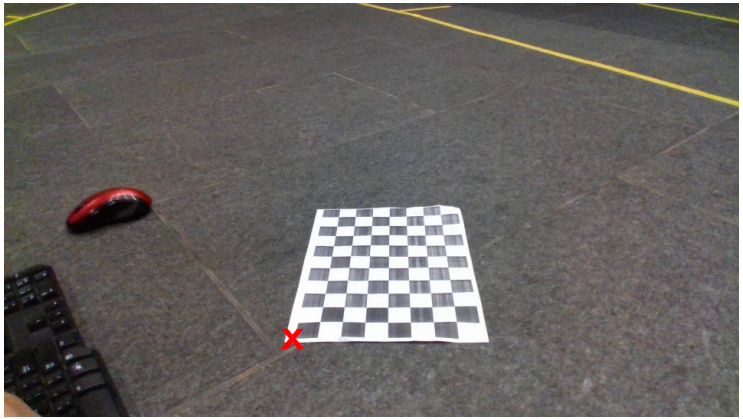
Graph-based SLAM (slam_toolbox)

Global Map

Calibration



Obtain camera intrinsic matrix K



Obtain camera pitch and yaw angles

Track Set-up

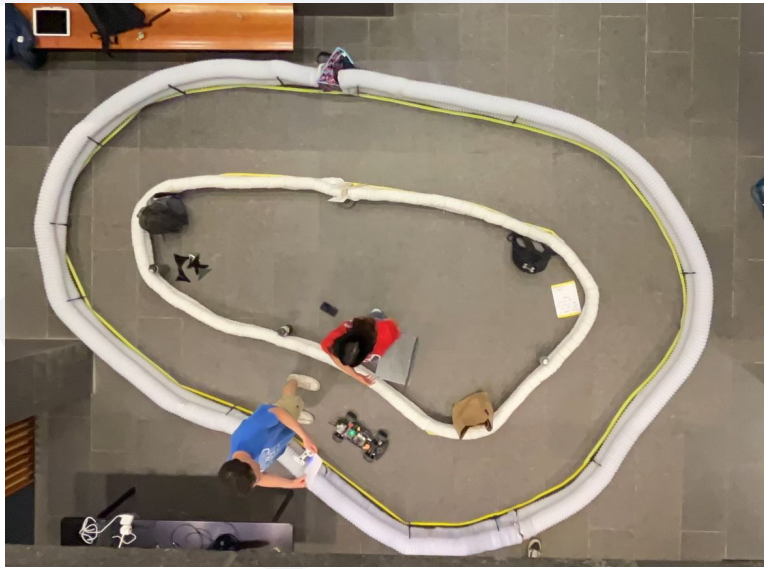
Track with added features for SLAM

Circular Track:
sharp curves

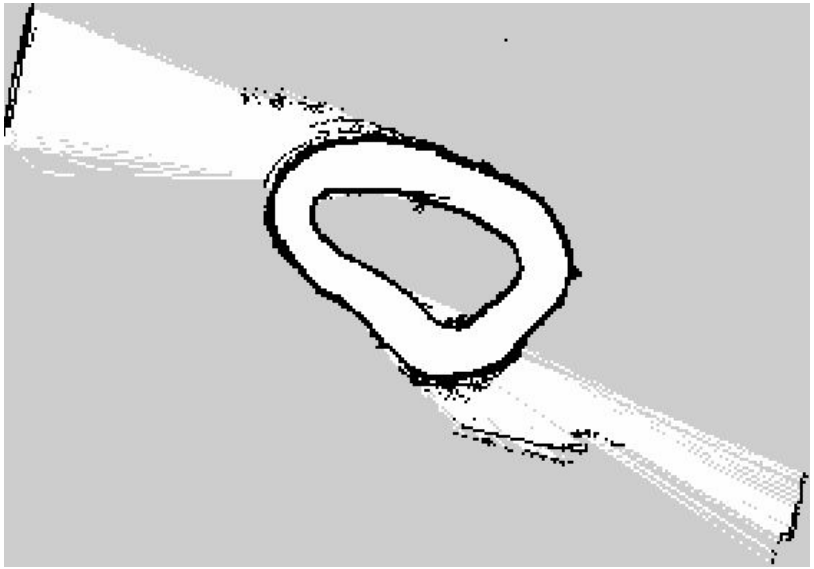
Square Track:
sharp corners

Simple track with Minimal features

Ground Truth Map using Lidar

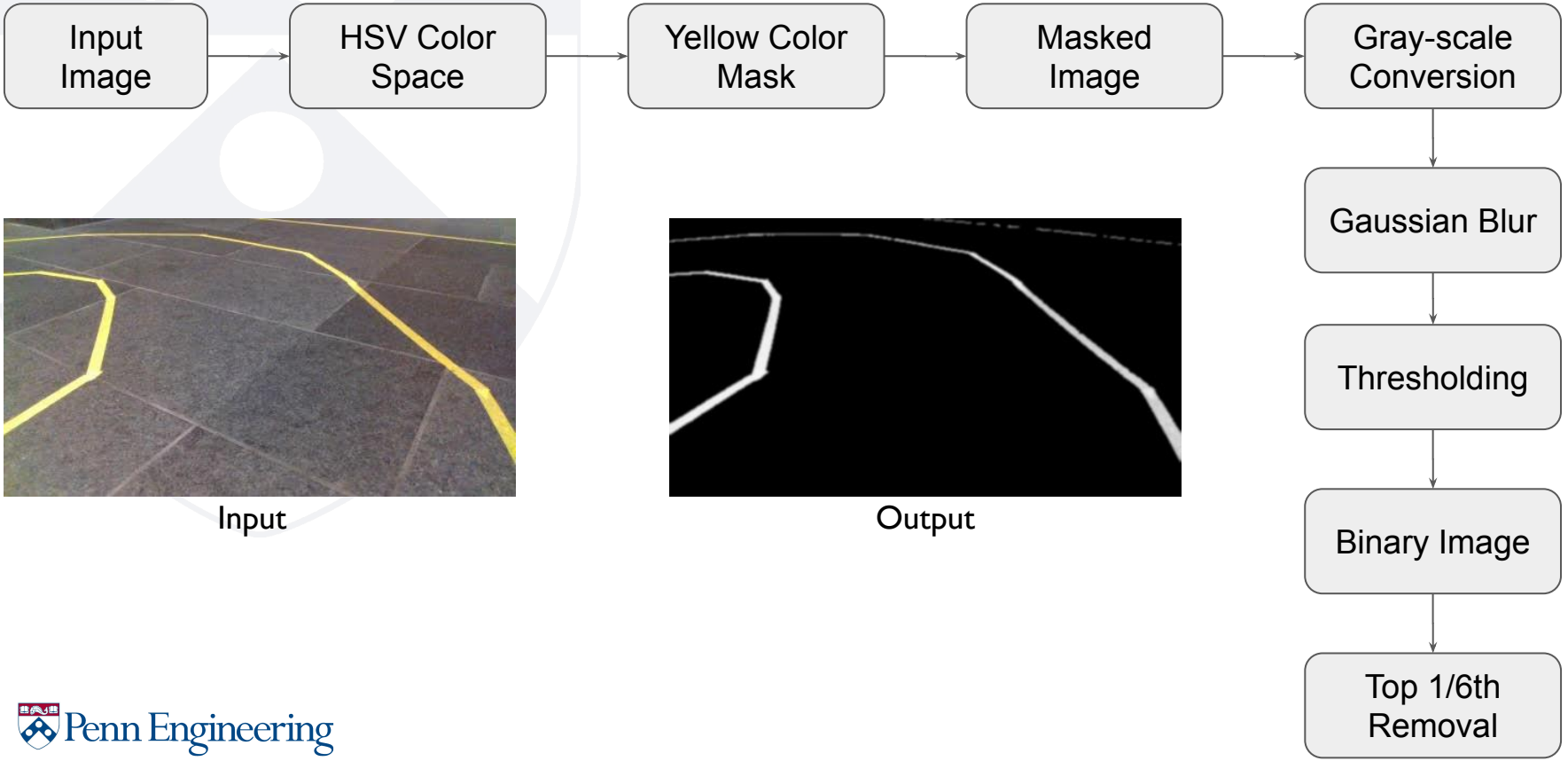


Overhead shot of the track for lidar mapping



Generated Map

Lane Detection



Input

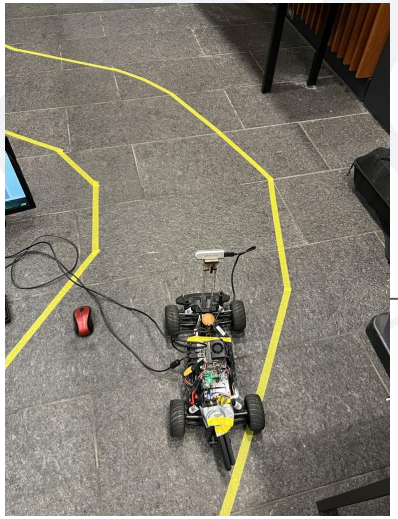
Output

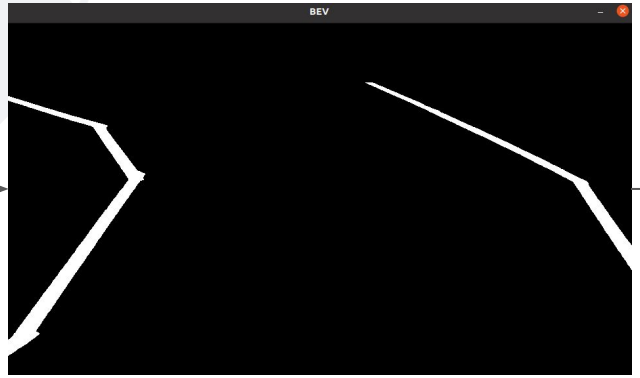
Lidar-like readings

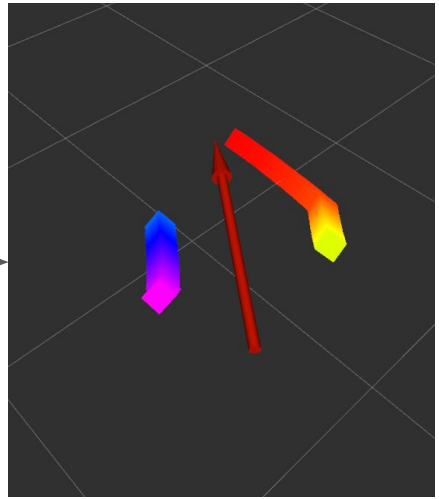
Binary Image
(white for lane)

Calculate distance and angle of white points from car/lidar

Post-process fake lidar readings to arrange and remove overlapping readings



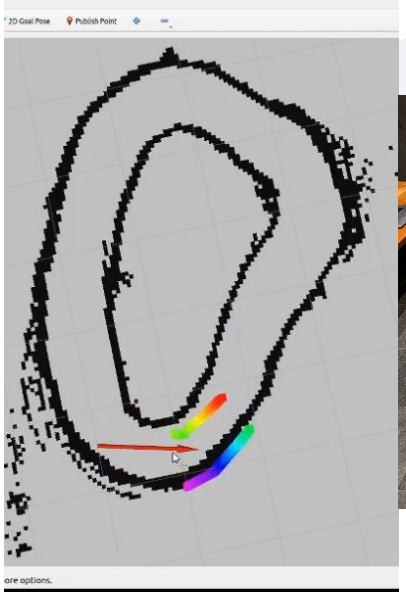


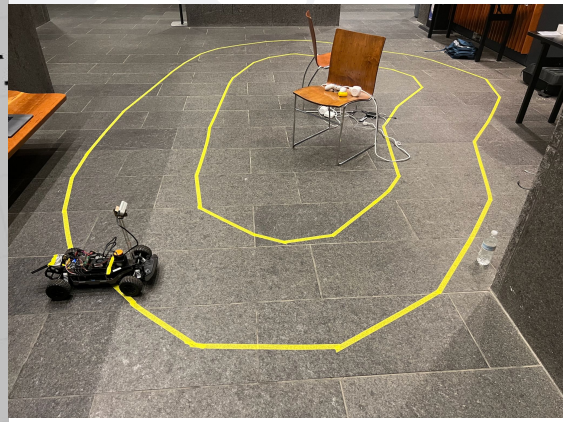


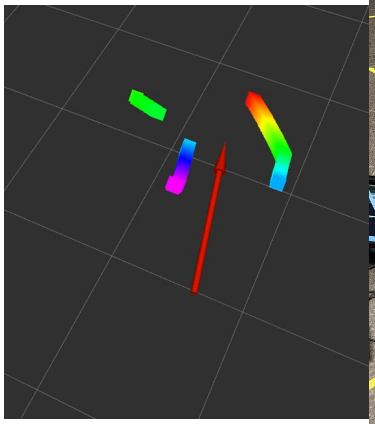
Everything running at ~33Hz

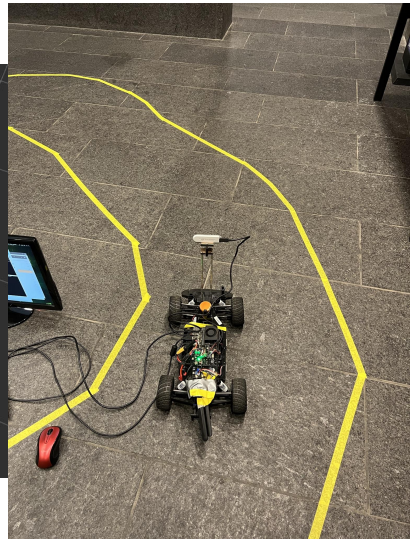
Ghost Lidar

- Challenge: Inner loop blocks itself, b/c of LiDAR-like rays
- Solution: Tell PF and SLAM that car is in front of itself





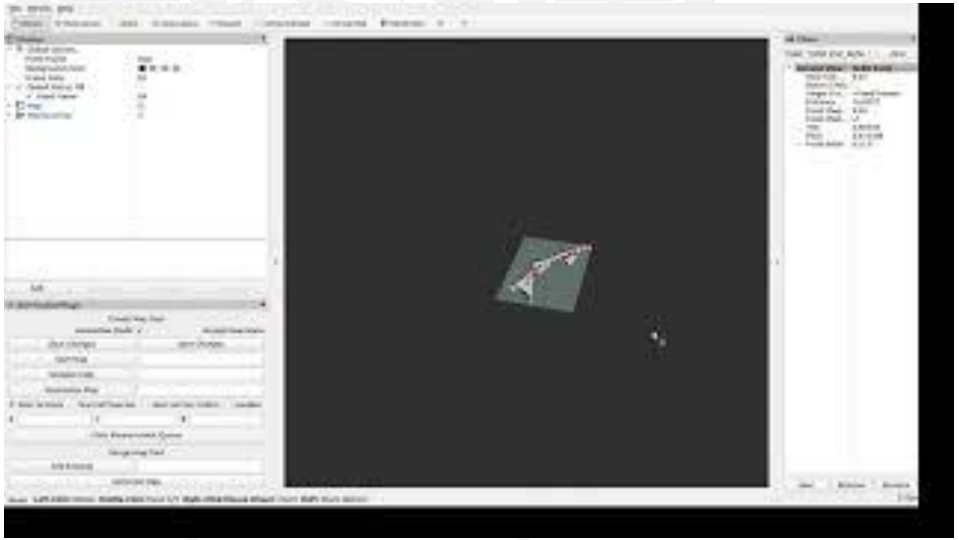




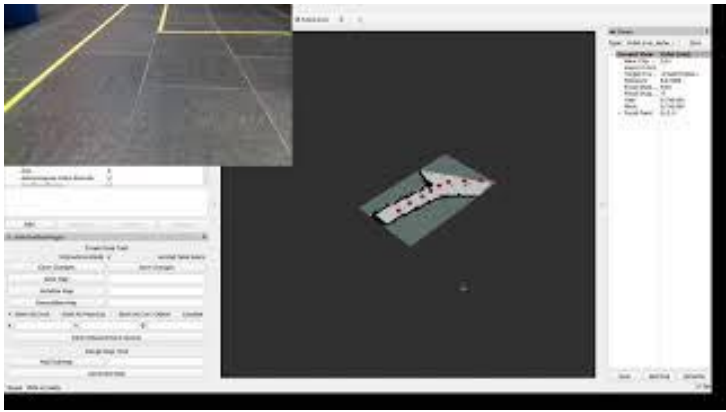
Raising the camera and introducing ghost LiDAR increased the FoV from ~68 to ~230 degrees (Real LiDAR has 270)

Penn Engineering

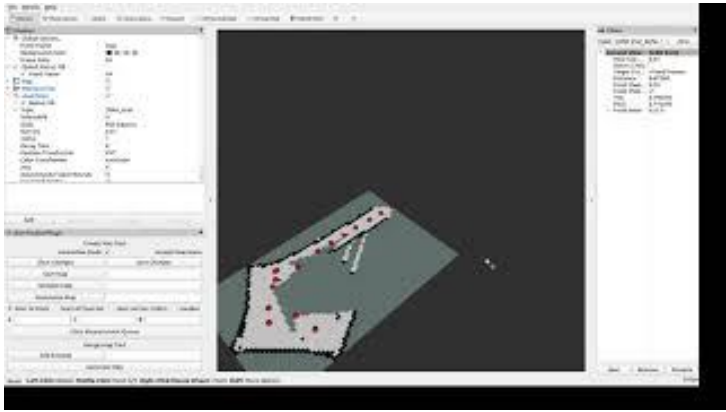
SLAM



Initial Attempt at SLAM



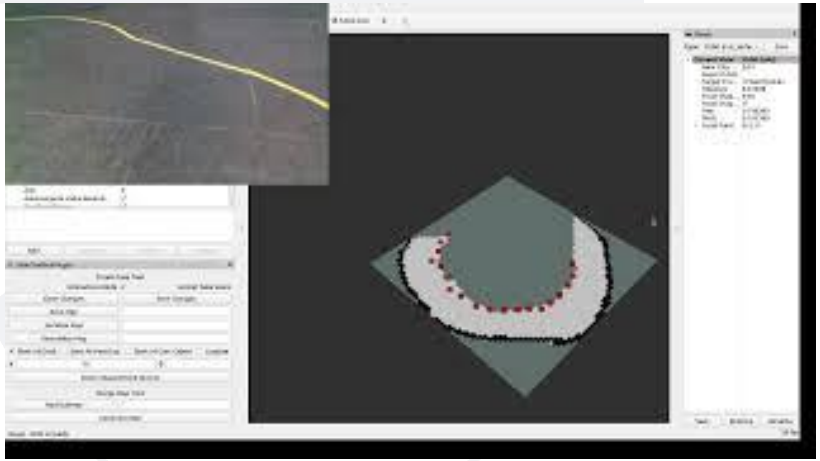
Pre-Tuning and accurate calibration



Post-Tuning and accurate calibration

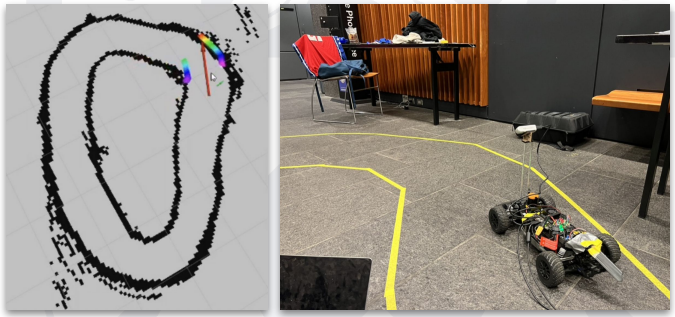
Penn Engineering

SLAM



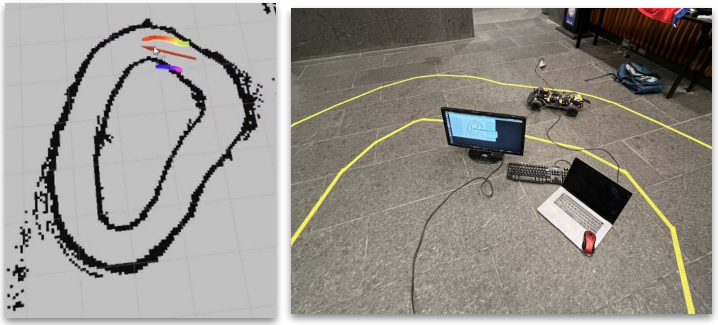
Final Attempt at SLAM on the Circle Track

Particle Filter Localization

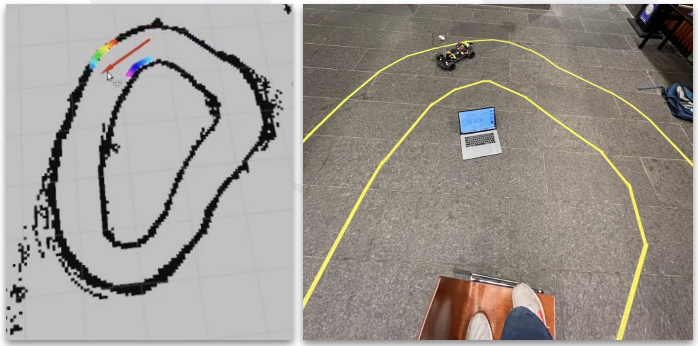


Turn 1

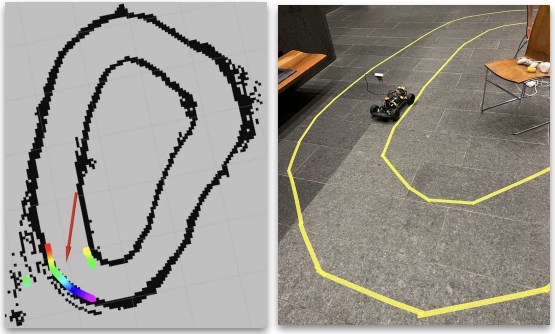
Localization is achieved using particle filter using lidar-like readings on the lidar scan generated map



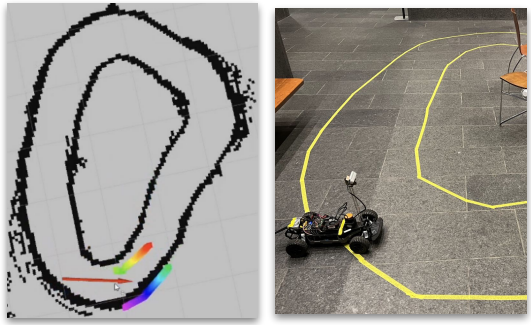
Turn 2



Turn 3



Turn 4



Turn 5

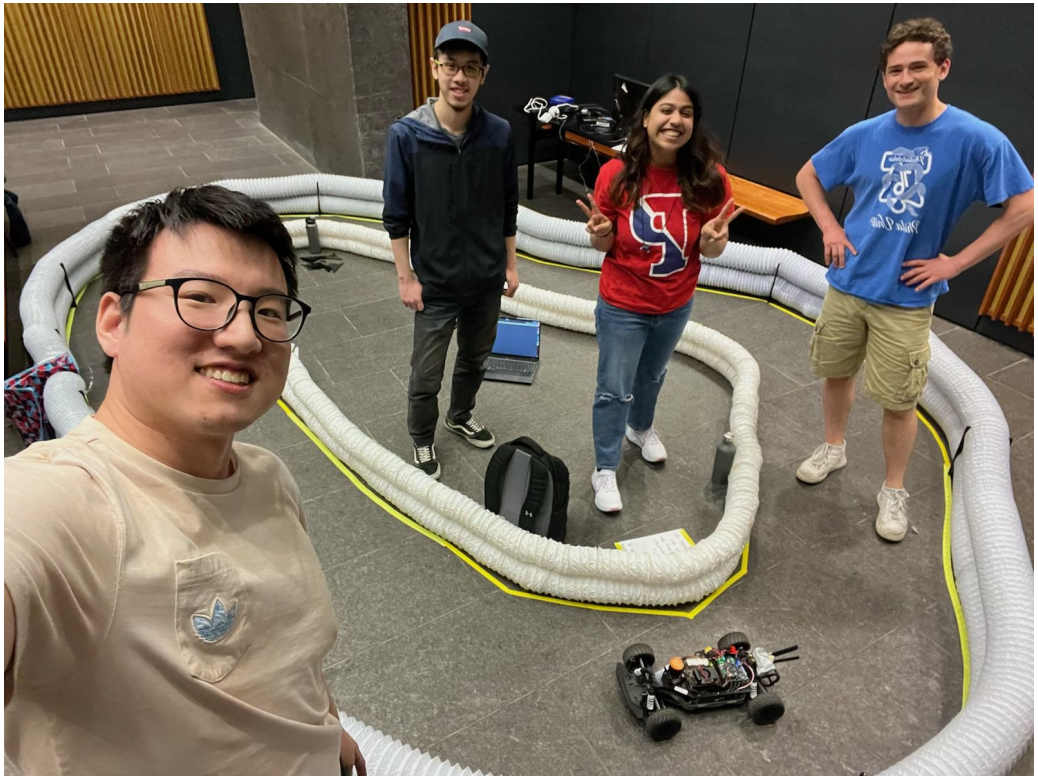
Penn Engineering

Mono-Cam vs LiDAR

	Cost	Speed	Accuracy	Robustness to lighting	Robustness to tilt	Environment Setup
	~$1500	~40Hz	High	Good	Poor	Hard
	<$300	~33Hz	Medium	Medium	Medium	Easy

Learnings and Improvements

- Accurate camera calibration is crucial for improving the performance of SLAM.
- Yaw and pitch tilts of the mounted camera should be taken into account during camera calibration.
- The field of view of the camera can affect the performance of SLAM.
- Odom tuning is important for particle filter performance in SLAM.



Thank you!

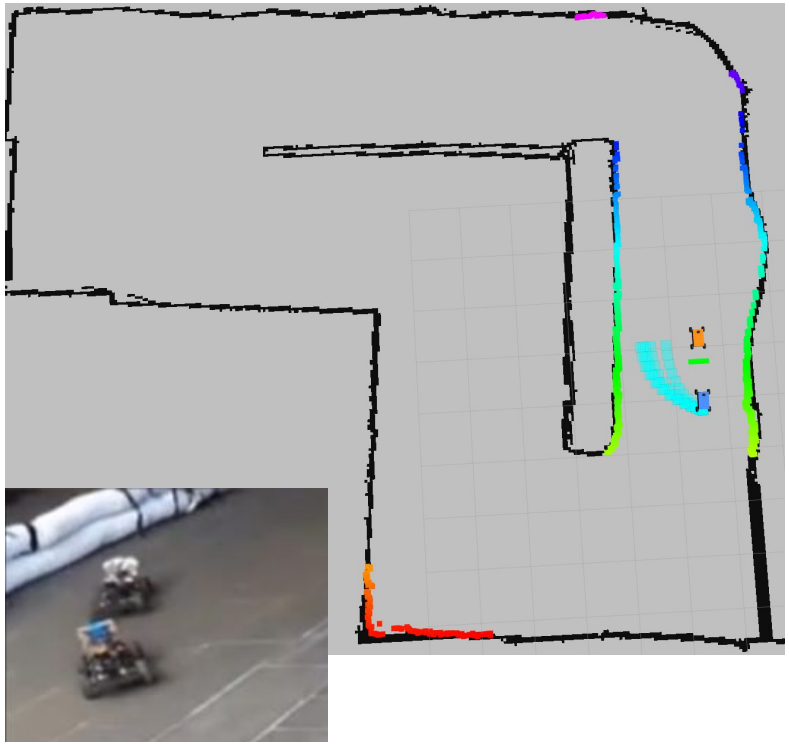
Team 5

Learning based MPC and Trajectory Generation in Frenet Frame for Autonomous Racing

Penn Engineering

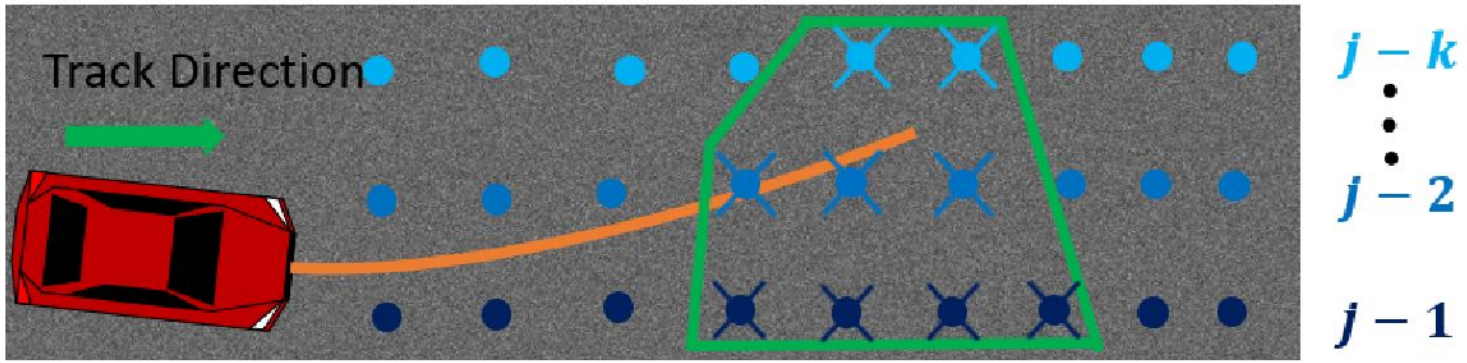
Overview

- The main objective is to race autonomously as fast as possible in presence of static/dynamic obstacles.
- Solution is to implement 2 different control modes.
  - LMPC - Allow algorithm to learn driving faster lap.
  - Frenet - Generate alternate feasible trajectories.



Learning based MPC

- LMPC ensures improved lap time performance.
- Closed loop trajectories are saved and used to update for following laps.
- Constructs a safe set- convex hull of KNNs



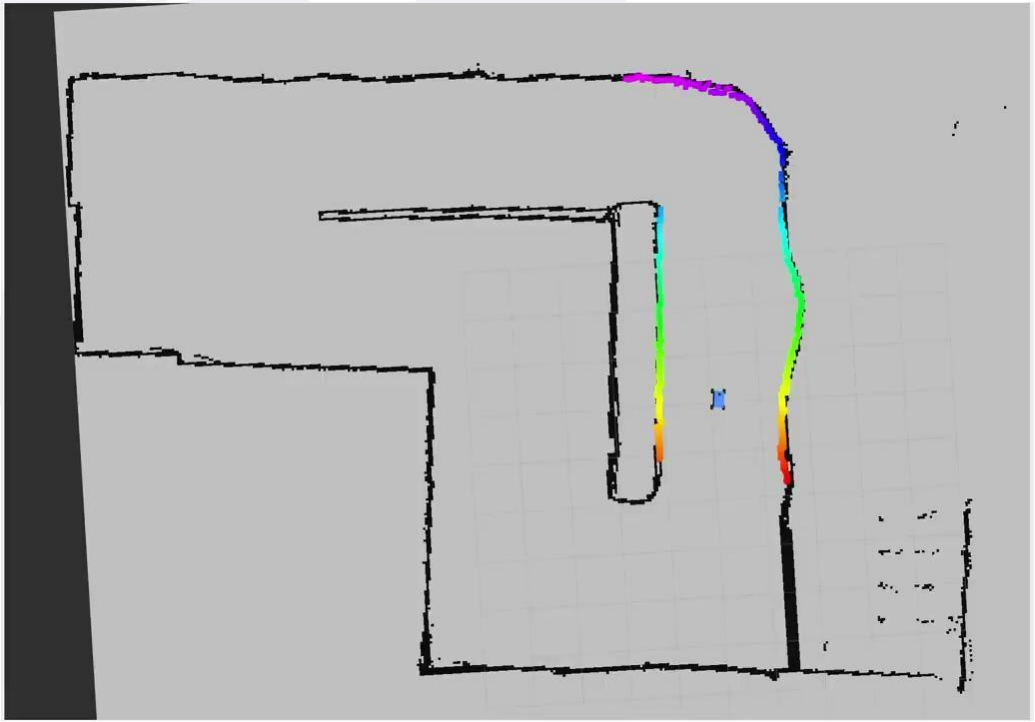
LMPC Problem Construction

- Objective Function:
  - State
  - Control input
  - Slack variables
  - Q Function
- Constraints:
  - Initial state
  - Kinematic model
  - Track
  - Terminal state

$$
\begin{aligned}
\min_{\mathbf{X}_t^j, \mathbf{U}_t^j, \boldsymbol{\lambda}_t^j} \quad & \sum_{k=t}^{t+N-1} (x_{k|t}^j - x_{ref})^T \mathbf{Q} (x_{k|t}^j - x_{ref}) \\
& + u_{k|t}^{jT} \mathbf{R} u_{k|t}^j \\
& + (u_{k|t}^j - u_{k-1|t}^j)^T \mathbf{R_d} (u_{k|t}^j - u_{k-1|t}^j) \\
& + s_1^T \mathbf{Q_{s_1}} s_1 + s_2^T \mathbf{Q_{s_2}} s_2 + \mathbf{J}_l^{j-1} \left(z_t^j \right) \boldsymbol{\lambda}_t^j \\
\text{s.t.} \quad & x_{t|t}^j = x_t^j \\
& x_{k+1|t}^j = A_t x_{k|t}^j + B_t u_{k|t}^j + C \\
& E_{k|t} - s_1 \leq G_{k|t} x_{k|t} \leq F_{k|t} + s_1 \\
& \boldsymbol{\lambda}_t^j \geq 0, \mathbf{1} \boldsymbol{\lambda}_t^j = 1 \\
& - s_2 \leq D^{j-1} \left(z_t^j \right) \boldsymbol{\lambda}_t^j - x_{t+N|t}^j \leq s_2 \\
& x_{k|t}^j \in \mathcal{X}, u_{k|t}^j \in \mathcal{U} \\
& \forall k = t, \cdots, t+N-1 \\
& s_1 \geq 0, s_2 \geq 0
\end{aligned}
$$

where $\mathbf{X}_t^j = \left[x_{t|t}^j, \ldots, x_{t+N-1|t}^j \right] \in R^{d_x \times N}, \mathbf{U}_t^j = \left[u_{t|t}^j, \ldots, u_{t+N-1|t}^j \right] \in R^{d_u \times N}, \boldsymbol{\lambda}_t^j \in R^{4K}.$

Results - Simulation



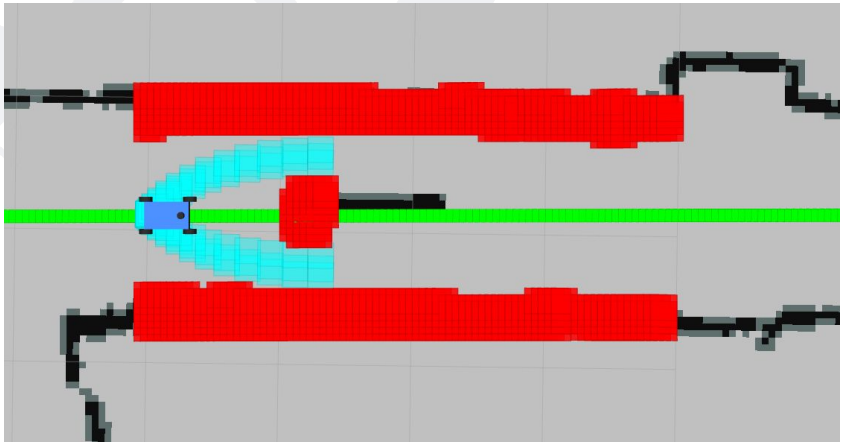
lap number	lap time (s)
1	10.69
2	9.47
3	8.77
4	8.12
5	7.17
6	6.77
7	6.57
8	6.36
9	6.17
10	6.02
11	5.76
12	5.61
13	5.45
14	5.39
15	5.27
16	5.18
17	5.15
18	5.15
19	5.15
20	5.15

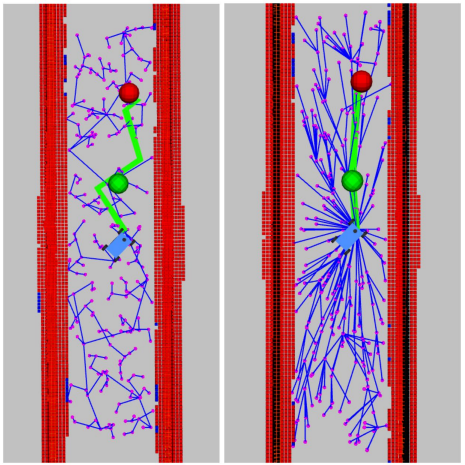
Results - Real World Environment

lap number	lap time (s)
1	12.0
2	10.1
3	9.6
4	9.2
5	9.2
6	9.0
7	8.7
8	8.6

Obstacle avoidance with Frenet

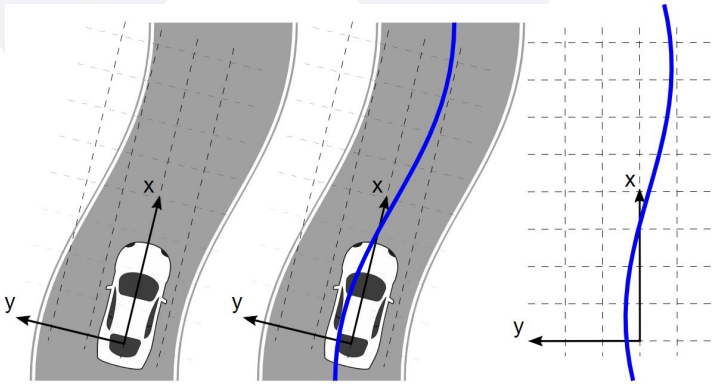
- Collision free paths generated with RRT and RRT* algorithm which may not be optimal for hardware.
- Using Frenet frames for motion planning gives smoother splines.
- They take lateral and longitudinal movement into account.

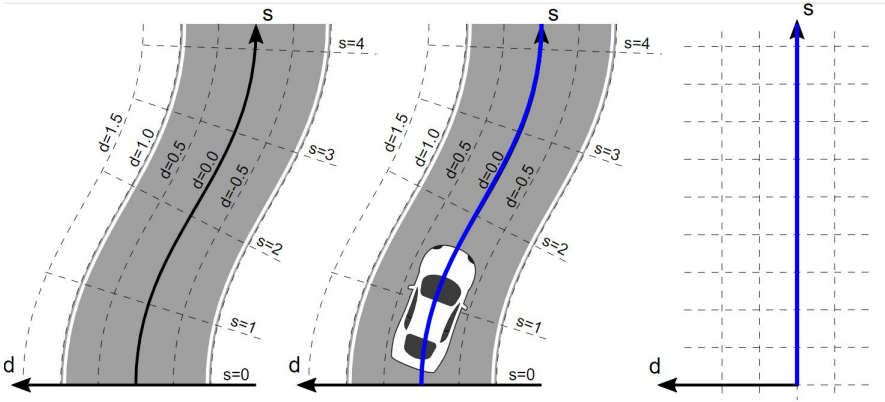




Frenet Frames for Local Path Planning

- Trajectory with the smallest jerk is selected.
- Frenet frame utilizes ‘s’(longitudinal displacement) and ‘d’(lateral displacement) instead of cartesian coordinates .



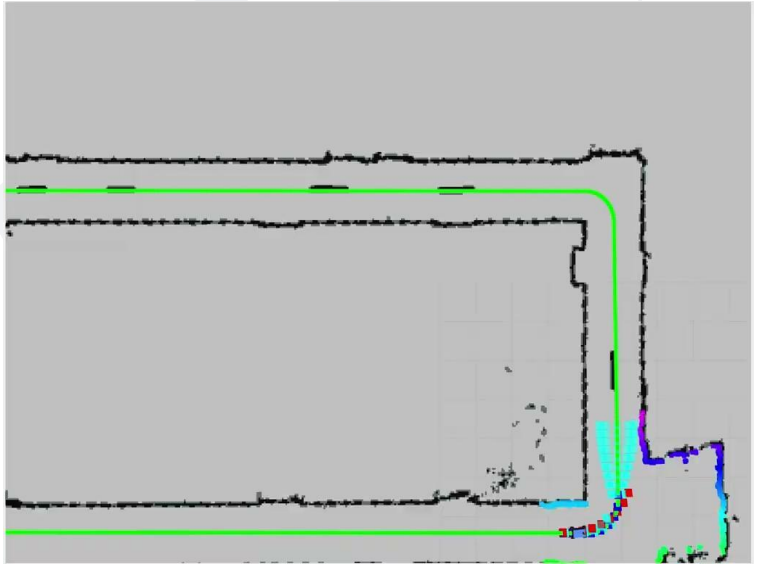


Trajectory Generation in Frenet Frame

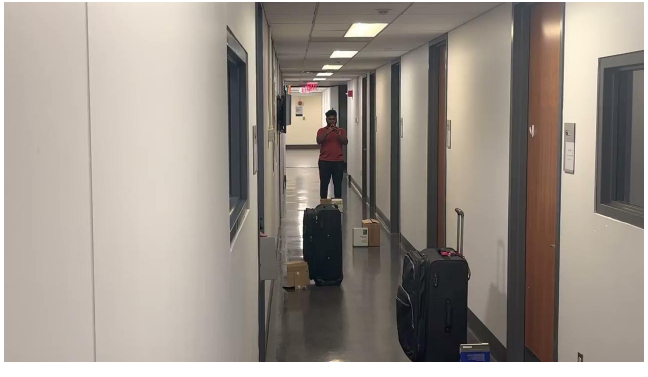
- Determine Frenet state vector $[s,\ \dot{s},\ \ddot{s},\ d,\ \dot{d},\ \ddot{d}]$
- Generate lateral trajectories by fitting a quintic polynomial to $d,\ \dot{d},\ \ddot{d}$ and target deviation at the end of prediction time
- Generate longitudinal trajectories by fitting a quartic polynomial to $s,\ \dot{s},\ \ddot{s}$ and the target velocity at the end of prediction time
- Combine lateral and longitudinal trajectories and assign cost to the trajectories according to the jerk in lateral and longitudinal directions
- Verify the combined trajectories if they have collision with obstacles and assign a large cost for the trajectories with collision
- Select trajectory with the smallest cost

Results

Simulation



On Actual Car



Penn Engineering

Future Scope

- Tune parameters in LMPC for faster learning on hardware.
- Reduce crashes when speed gets faster in later laps of LMPC.
- Improve dynamic obstacle avoidance and overtaking strategy.
- Tune parameters in Frenet Frame trajectory planar to work better at high speed.

Thank you

Questions?

Penn Engineering

Team 6

Reinforcement Learning for Planning

Mingyan Zhou, Sofia Panagopoulou, Gaurav Kuppa, Martin Alijaj

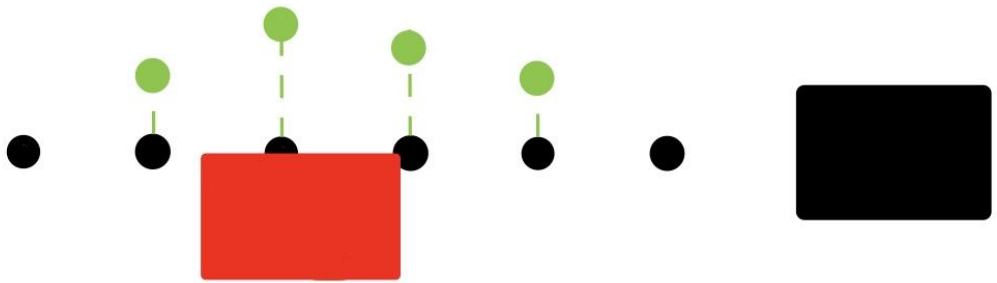
Penn Engineering

Overview

- Pure pursuit is an excellent controller but cannot avoid obstacles
- RRT, RRT* can avoid obstacles but not efficiently
- Use pure pursuit but adapt to obstacles
- Meet our project:
  - Use reinforcement learning to modify our pure pursuit path and adapt to obstacles
  - RL training beforehand (efficient, lower computational planning)

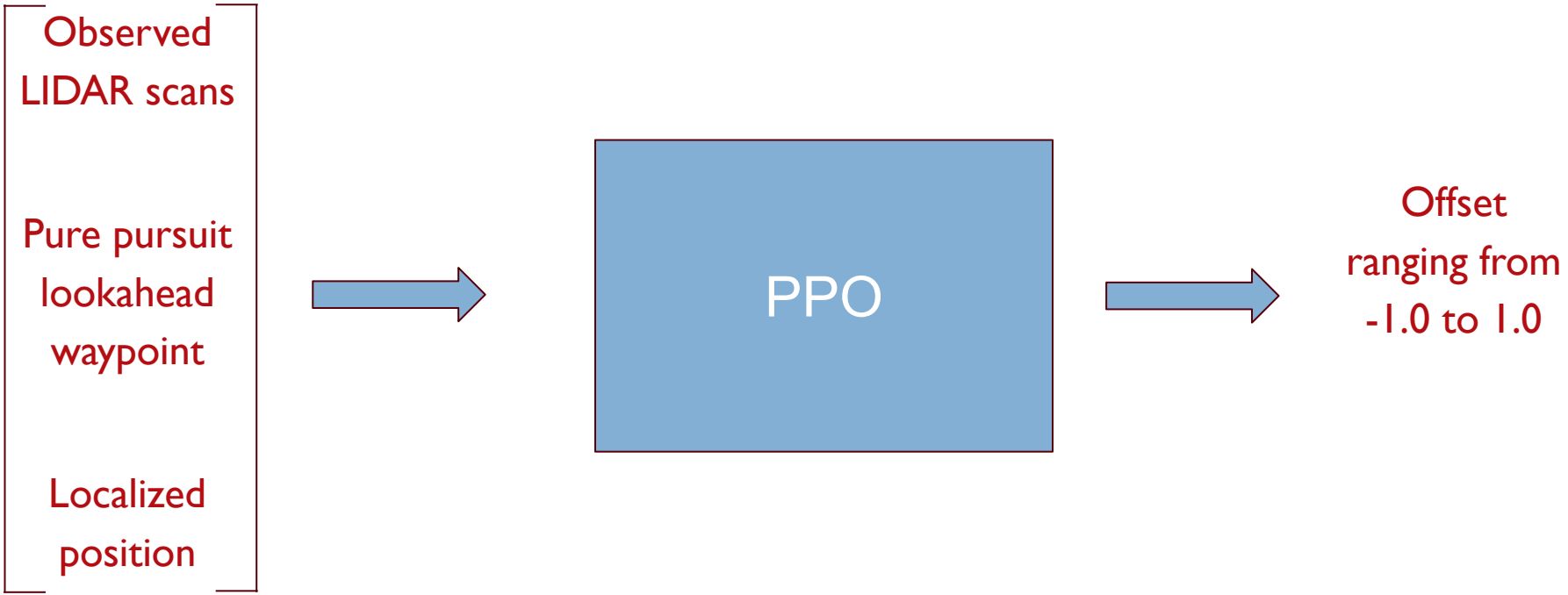
The Idea

- Learn to deviate from pure pursuit waypoints to avoid obstacles
- Generate series of green offsets at every timestep
- Could expand to dynamic obstacles (opponent)



black box: our car
red block: opponent car
black points: pure pursuit waypoints
green points: offsets

Inputs & Outputs To Model



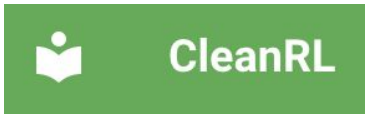
Development Process

- Environment Setup
- Training Model with RL
- Experimental Results in Gym
- Converting for Real World Use
- Experimental Results on Real Car

Environment Setup

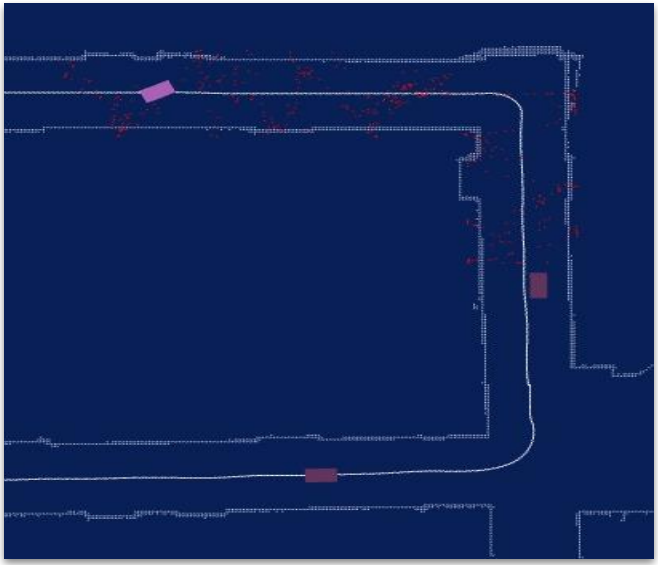
- F1TENTH gym environment (vs ROS)
  - Lowerweight, suitable for training
  - Load map, handle car simulation, gather data to send to model
- CleanRL
  - Training for RL in games
  - Multithreaded, single-file PPO implementation

f1tenth.org

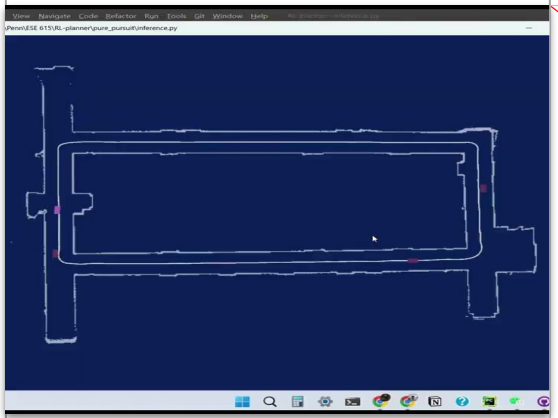
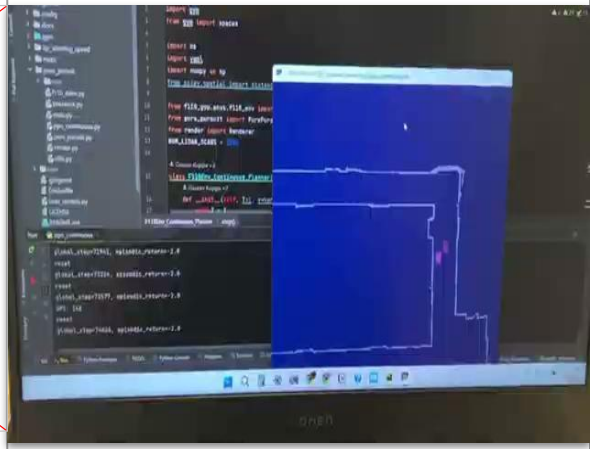


Training Process

- PPO Algorithm
  - Actor-critic method with a continuous action space
  - Policy gradient method that limits update space
- Observations
  - Lidar scans, waypoints, current position
  - Only Lidar scans
- Reward function:
  - Collision, longevity
  - Add maximize minimum Lidar scan, penalize deviation from waypoints



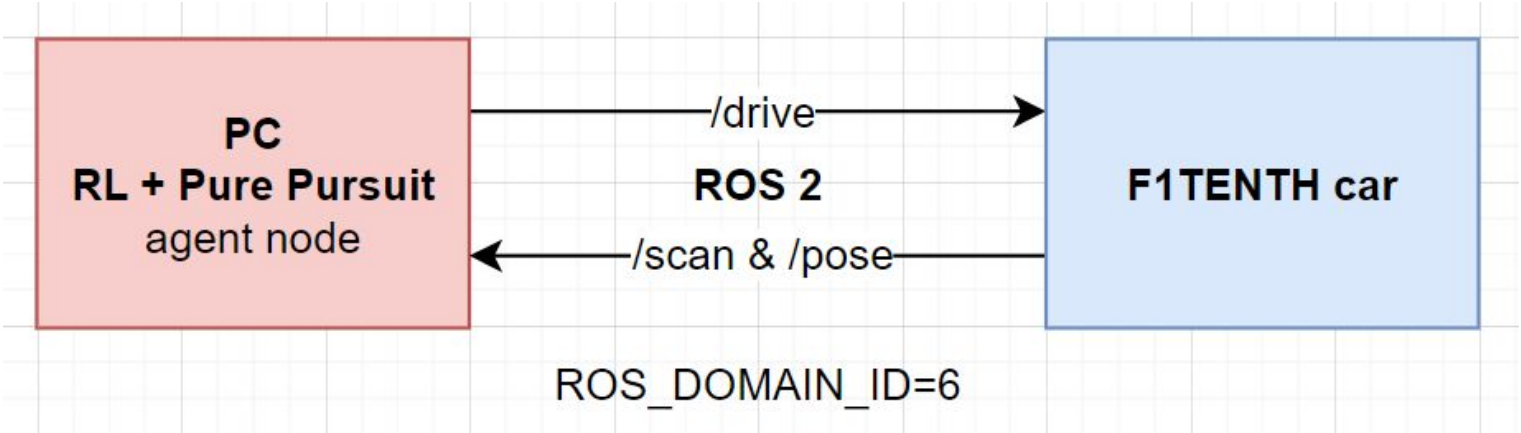
Final Gym Experimental Results

Obstacles	Static	Random Static	Dynamic
Map	Easy	Medium	Difficult
Visual			

Penn Engineering

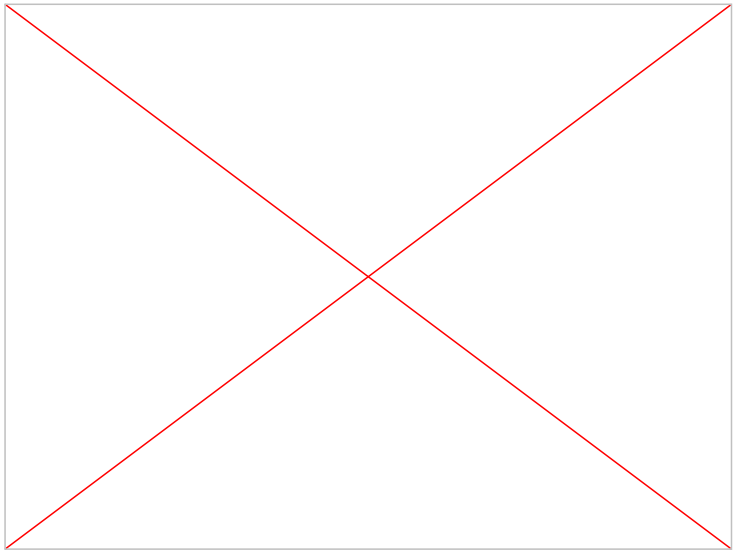
Converting to ROS

- Gym uses **Observation** to save pose, heading, lidar scans, velocity, etc.
  - Agent car can get the observation easily
- On **F1TENTH** car, we adopted the upper-lower communication method
  - Easy to implement



Real Car Results

- Performances are not good… – Sim2Real Gap
- Added Gaussian noise to the Lidar scans during training
- Exactly same setup with the simulation (same obstacle and car locations)
- Need extra time & effort



Project Learning

- You cannot slap RL onto any problem and expect it to work…you must first understand the problem to have RL attempt to solve it
  - A paraphrased quote by Rahul
  - And we fully agree
- Significantly more familiar with using RL to accomplish tasks
  - Importance of reward functions & how they impact training
  - Importance of randomness in training scenarios
- Using and setting up a gym environment
- Huge sim2real gap for RL

Current Limitations & Future Improvements

- Current limitations:
  - Increased dependence on Lidar
  - Car learns by colliding, does not learn avoiding behavior
- Future Improvements:
  - More training
  - Varying conditions such as speed, obstacle count, maps
  - Leverage sim2real gap for all kinds of obstacles
  - Bootstrap using imitation learning

Thank you!

Penn Engineering

Autonomous Control for Car-Trailer System

F1Tenth Final Project Presentation - Team 7

Penn Engineering

Motivation

As self-driving cars become a reality, there will be an increasing demand for adding trailers and extensions to one's autonomous cars.

Moreover, the development of autonomous car-trailer systems can benefit the logistics and transportation industry by improving efficiency, reducing costs, and increasing safety. These systems could reduce the need for human drivers, making transportation more efficient and cost-effective while also reducing the risk of accidents caused by driver error.

Controlling the motion of a car and trailer in a coordinated manner is a complex problem that requires the development of a good control system.

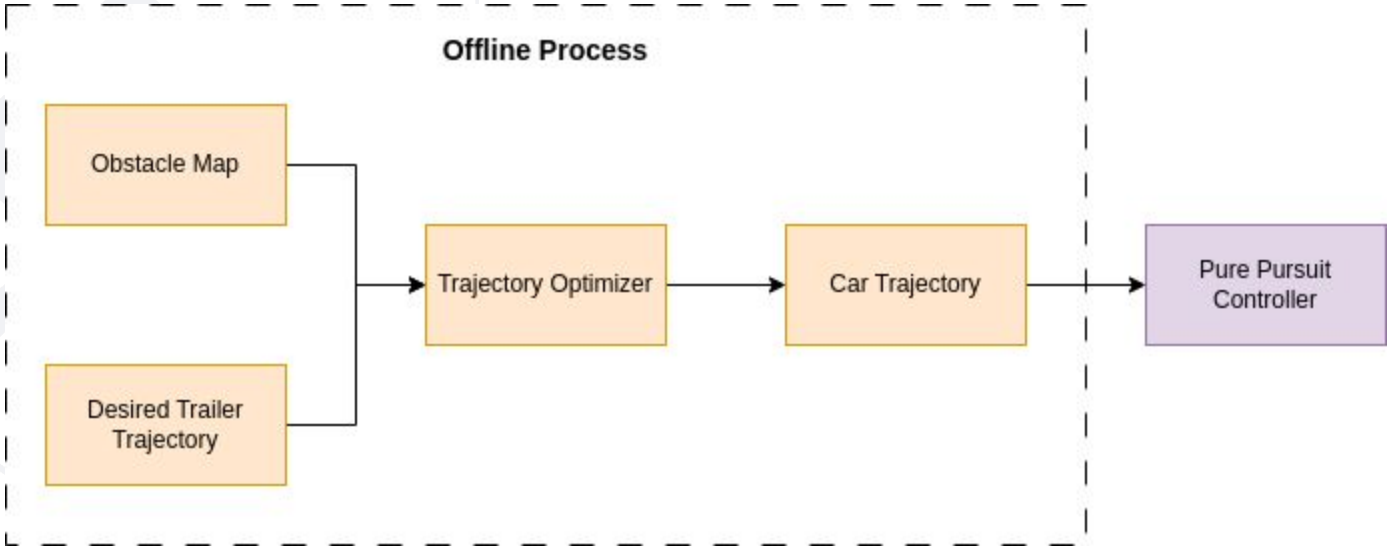
Objective

The objective of the project is to design and build an autonomous single-trailer car that can safely and efficiently navigate through an environment with obstacles.

To achieve this objective, we designed a control system using a non-linear trajectory optimization framework and Pure Pursuit controller that can accurately track the car-trailer system trajectory accounting for the non-linear dynamics of the system.

To test the control system, we placed obstacles like cans in the environment and built a controller that ensures the car avoids the cans while the trailer is controlled to knock-off the cans.

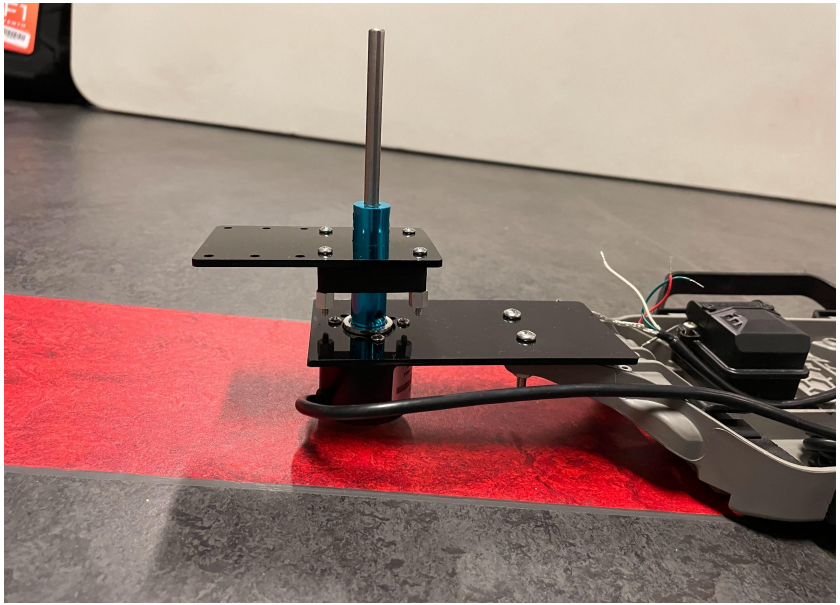
Problem Formulation and Approach



Mechanical Designing of the Trailer

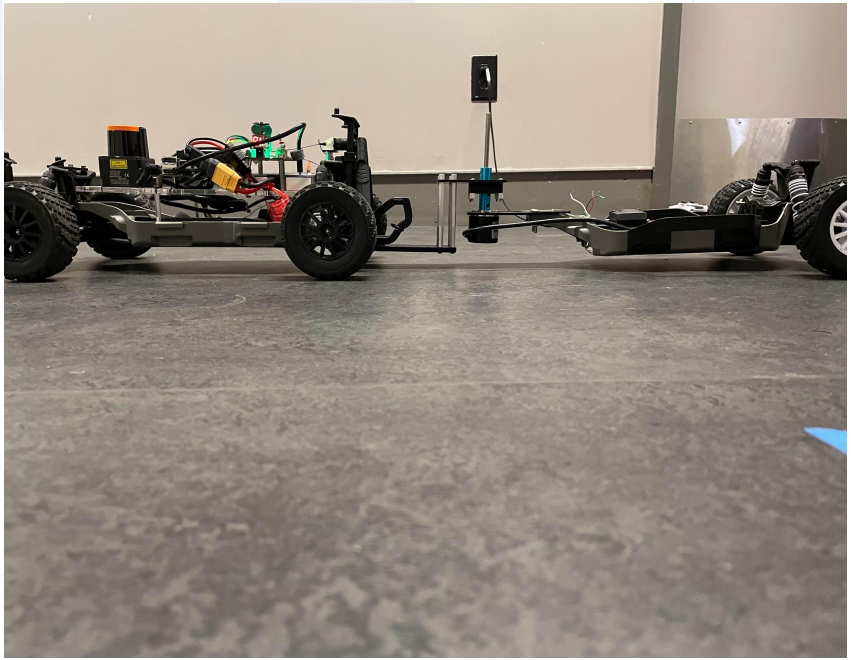
Car Side Attachment

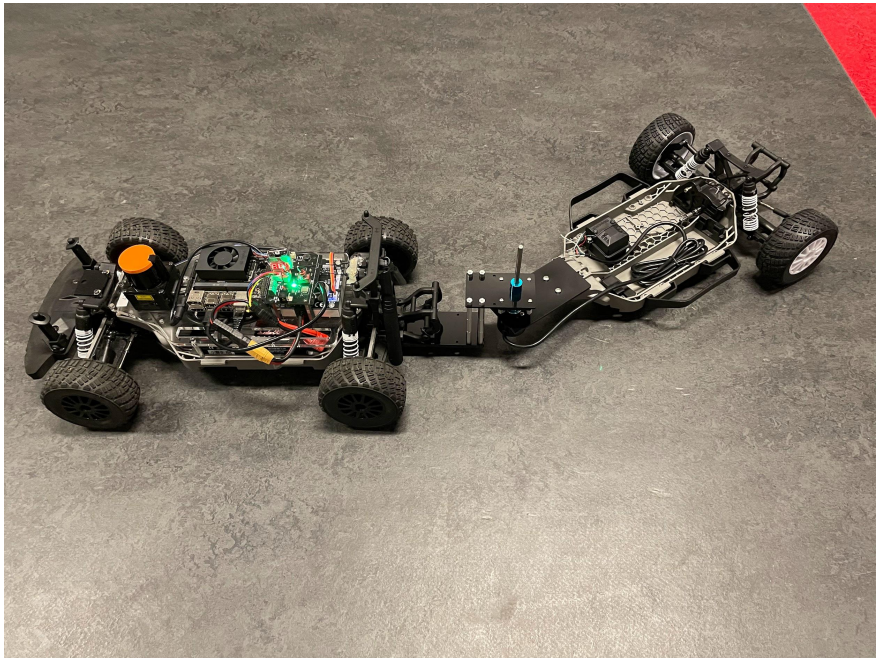
Trailer Side Attachment



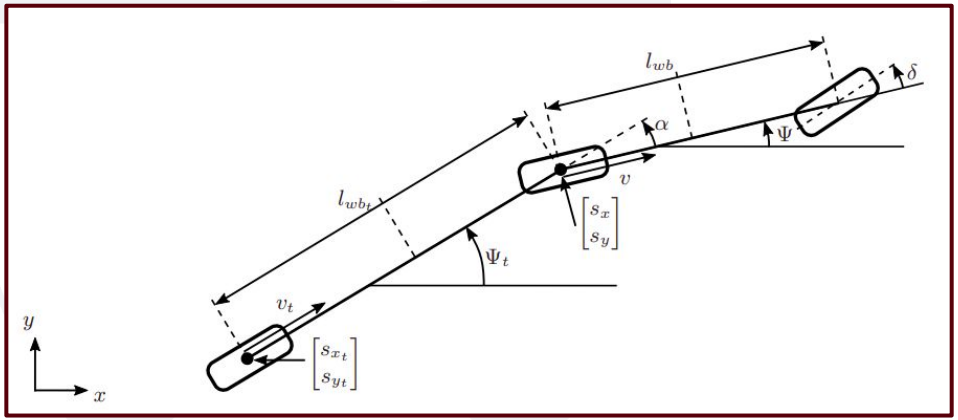
Mechanical Designing of the Trailer

Final System Assembly





Car-Trailer System Model



Kinematic Model:

$$\dot{x}_1 = u_1 \cos(x_3)$$
$$\dot{x}_2 = u_1 \sin(x_3)$$
$$\dot{x}_3 = \frac{u_1}{l_{wb}} \tan(u_2)$$
$$\dot{x}_4 = -u_1 \left(\frac{\sin(x_4)}{l_{wbt}} + \frac{\tan(u_2)}{l_{wb}} \right)$$

State Variables:

$$\mathbf{x}_1 = x_{car}$$
$$\mathbf{x}_2 = y_{car}$$
$$\mathbf{x}_3 = \theta_{car}$$
$$\mathbf{x}_4 = \alpha$$

Control Inputs:

$$\mathbf{u}_1 = v$$
$$\mathbf{u}_2 = \delta$$

We get the trailer coordinates as follows:

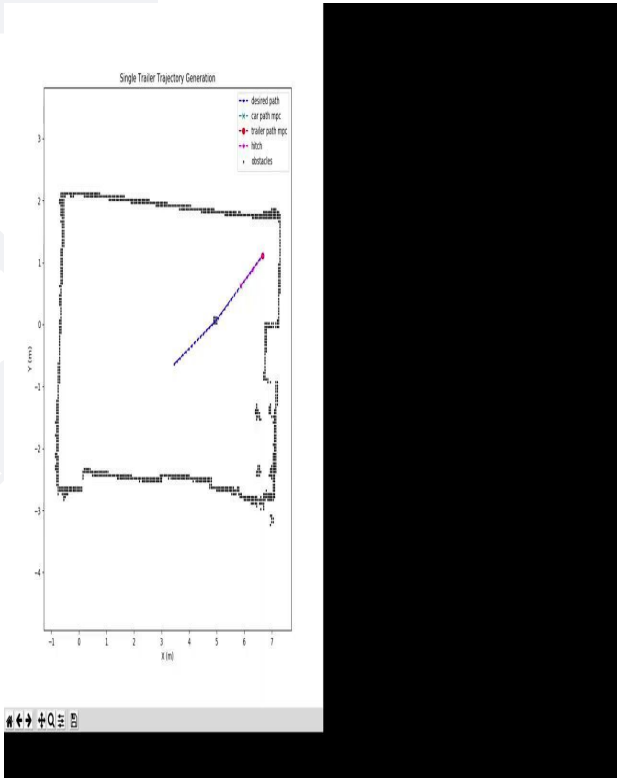
$$p_{trailer}^{world} = T_{car}^{world} \times p_{trailer}^{car}$$

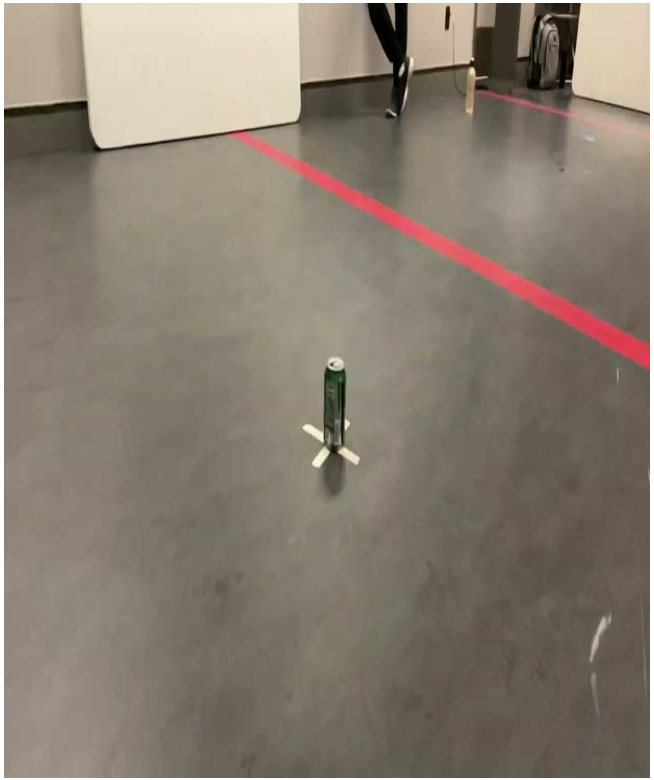
$$p_{trailer} = \begin{bmatrix} \cos(x_3) & -\sin(x_3) & x_1 \\ \sin(x_3) & \cos(x_3) & x_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -l_{wb} - l_{wbt}\cos(x_4) \\ -l_{wbt}\sin(x_4) \\ 1 \end{bmatrix}$$

Non-linear Trajectory Optimization

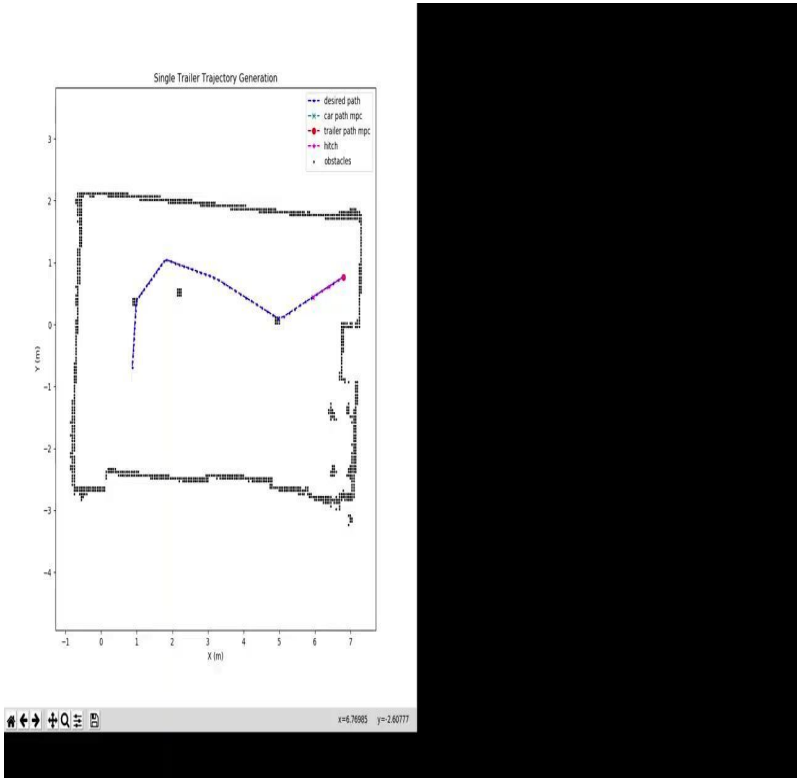
- Variables
  - State: Car Position (x, y, θ), Hitch angle (α)
  - Input: Car Velocity (v, $\boldsymbol{\delta}$, dt)
- Constraints
  - Initial state constraint
  - Input constraints
  - State constraints
  - Dynamics constraints
  - Obstacle constraints
- Objective Function
  - Tracking cost (for trailer)
  - Acceleration cost

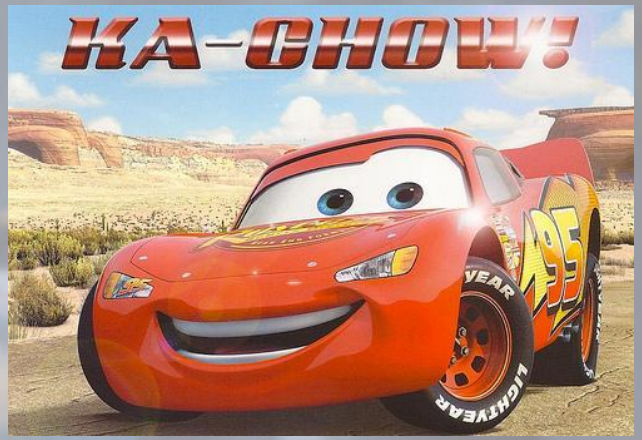
Video Demonstration - Single Can Trajectory





Video Demonstration - 3 Can Trajectory





Team 8 - Kachow Racing

Enhanced Lateral Control for Masterful Defensive Driving

Team members – Colin Hosking, Saibernard Yogendran

Penn Engineering

Questions

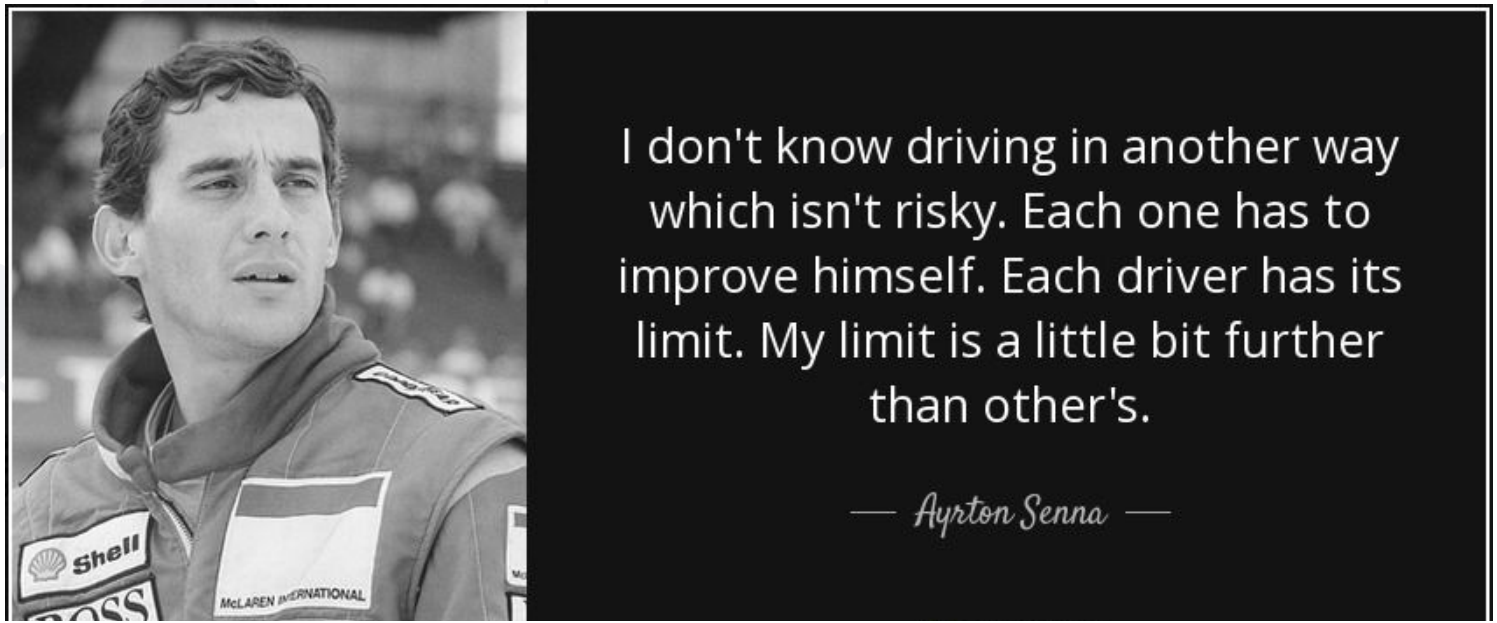
Mastering the Art of Rearview Defense: Formula Drivers' Key to Outsmarting Competitor

Penn Engineering

Credits: https://www.quora.com/Do-F1-cars-have-rear-view-mirror

Mastering the Art of Rearview Defense: Formula Drivers' Key to Outsmarting Competitor

Once upon a time, there lived a ghost !!!!



Penn Engineering

Credits: https://www.azquotes.com/quote/266138

Defensive Racing: The Ever-Evolving Art of Speed and Strategy

Penn Engineering

Credits: Formula 1 Youtube channel

Defensive Racing: The Ever-Evolving Art of Speed and Strategy

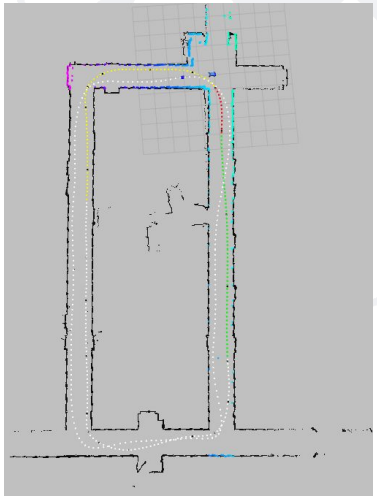
Penn Engineering

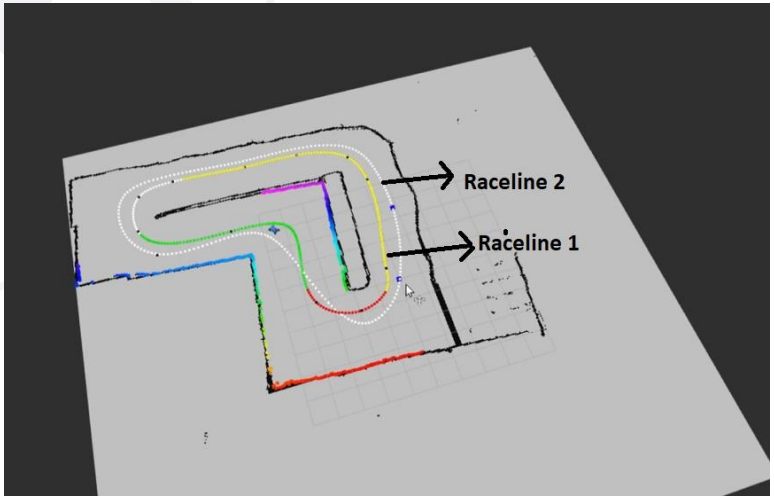
Credits: Formula 1 Youtube channel

Dual Racing Line Strategy: Enhancing Performance on the Track

Why two racing lines?

- Embracing two racing lines proves essential for efficient racing, as relying solely on a trajectory to block opponents can be precarious due to the potential lack of awareness regarding the track surface and conditions.
- Implementing a reference line mitigates this risk and ensures the car's lap time remains consistent. The car seamlessly transitions between these two lines to achieve defensive racing. As track size increases, so does the number of lines, with decisions made based on the position of the competing race cars.



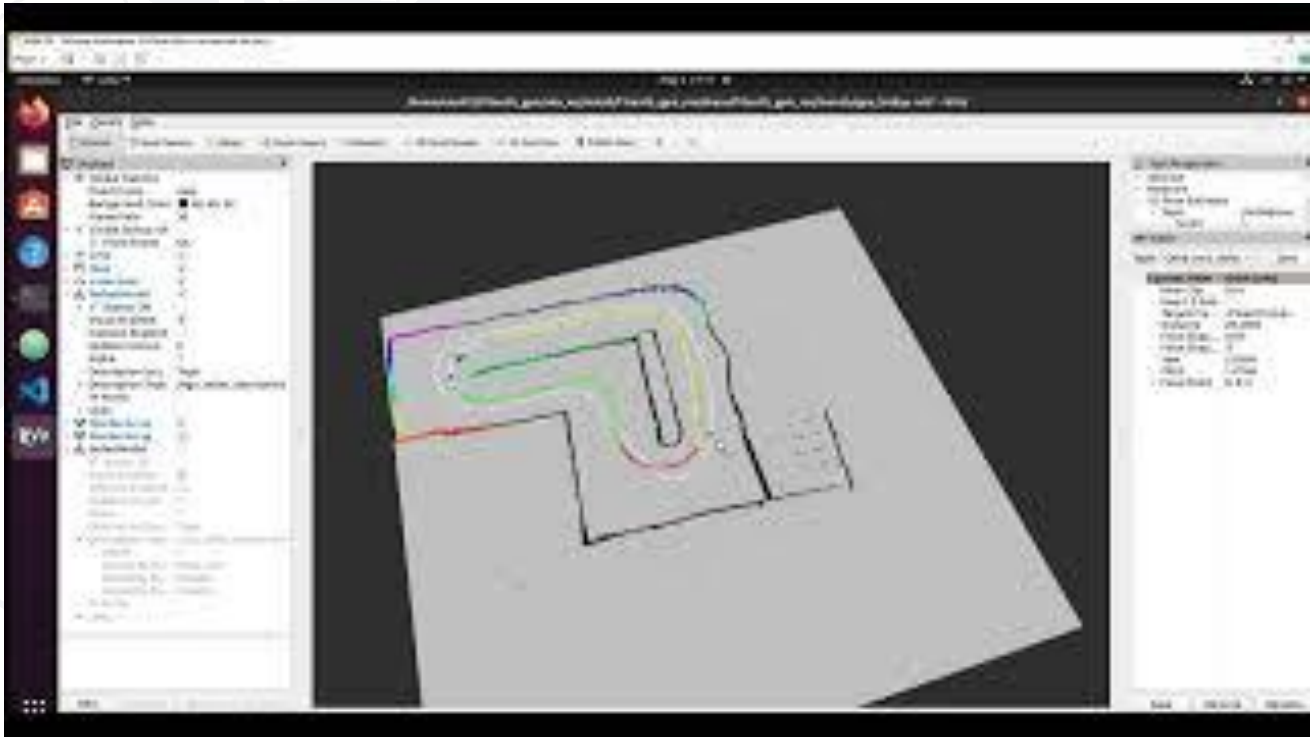



```
Lap time raceline 1: 6.307 seconds
Lap time raceline 2: 6.297 seconds
```

Difference – 1/100th of a second

Penn Engineering

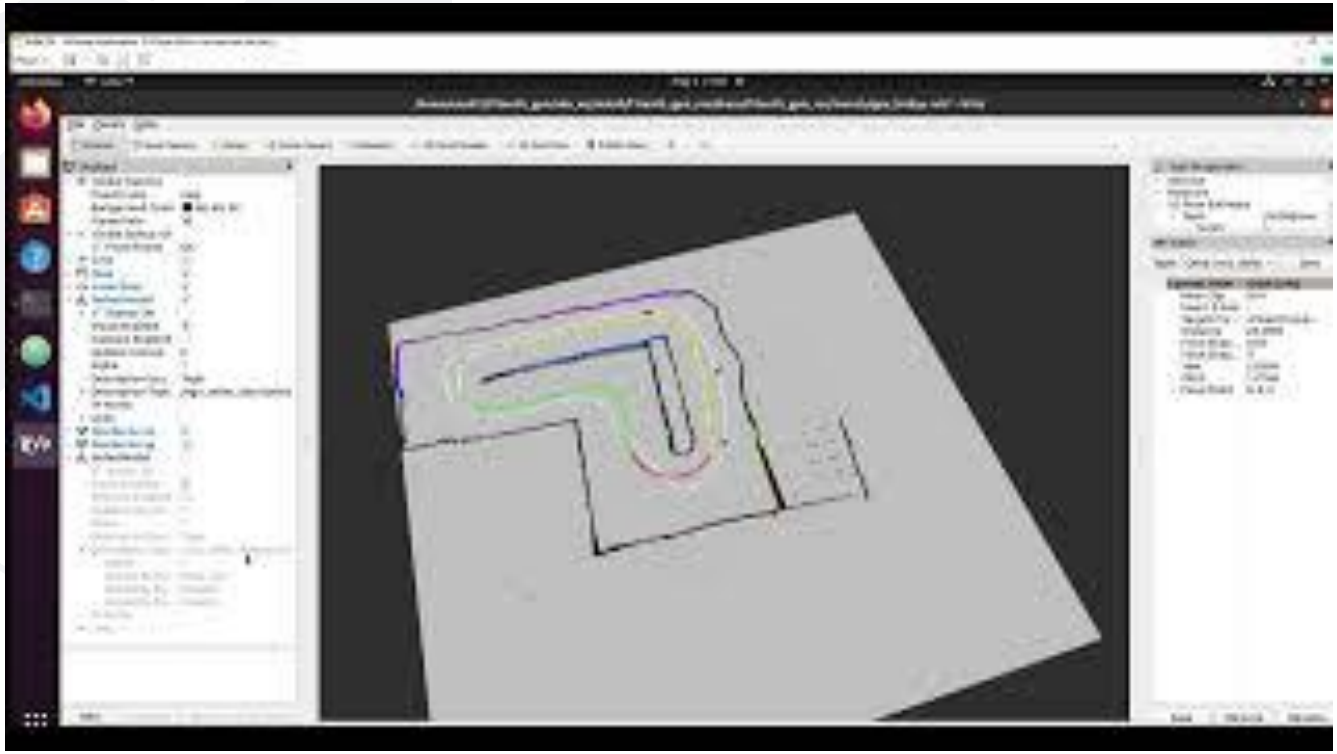
Raceline 1 performance



Penn Engineering

Lap time raceline 1: 6.307 seconds

Raceline 2 performance



```
Lap time raceline 2: 6.297 seconds
```

Enhanced Lateral Controller

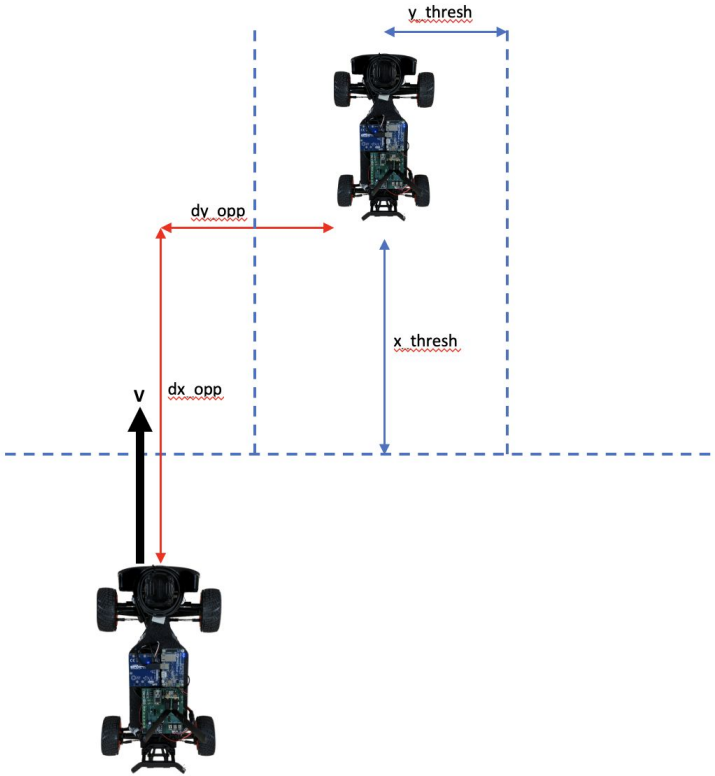
How are we achieving the shift between racing lines?

<u>Controller Strategy:</u>

1. Decision to Switch
2. Calculating Merging Path
3. Steering Angle Considerations

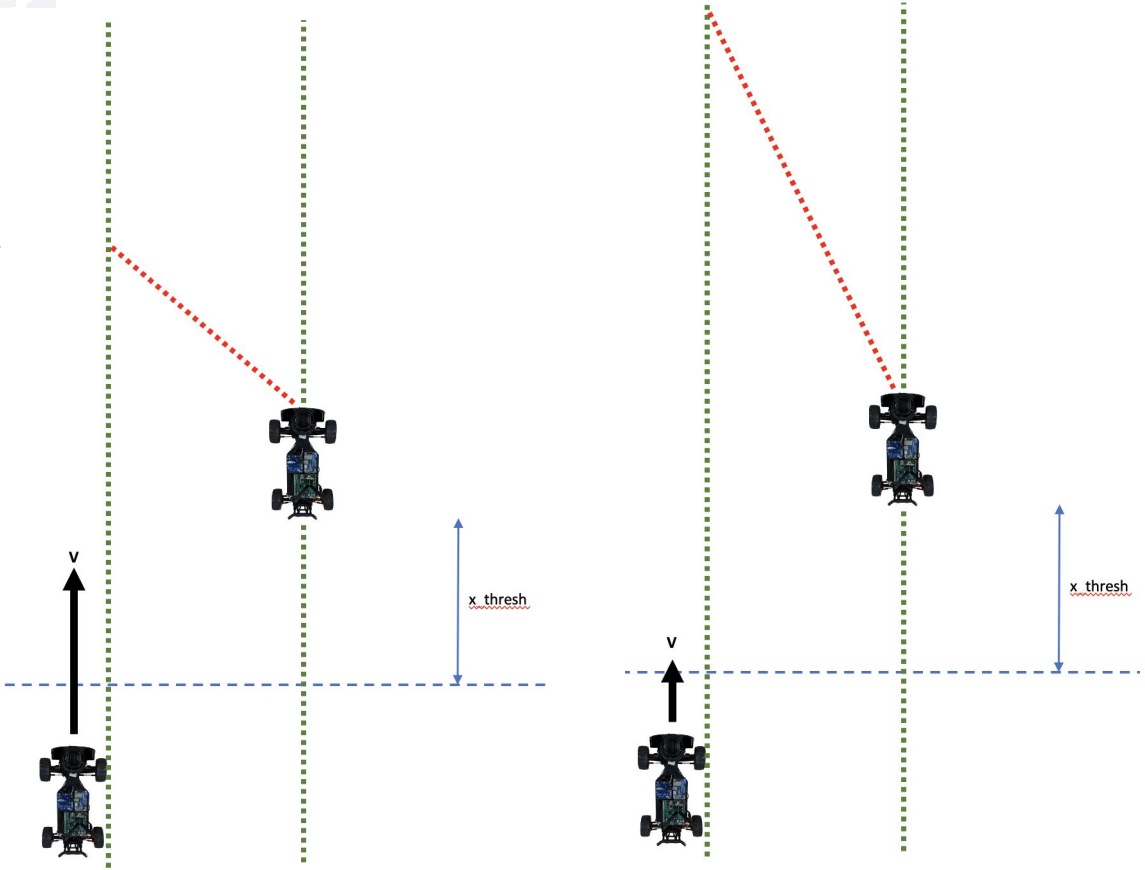
Decision to Switch

- Vision node tracks opponent's lateral offset and distance behind, as well as calculates the opponent's velocity at each timestep
- Tunable parameters of x_thresh and y_thresh
- If the opponent is projected to cross x_thresh in the next timestep, and is laterally outside y_thresh, then switch racing lines



Merge Path

- If the opponent is approaching at a high velocity, our car needs to switch lines faster in order to block it
- The merge distance is a linear function of the opponent's velocity
- Interpolate a cubic spline between current position and merge distance along the next racing line



Penn Engineering

Steering Angle Considerations

Calculating the maximum allowable steering angle while taking the lateral acceleration limits into account.

1) Parameters:

- ay_max: Maximum allowable lateral acceleration (Tested using constant radius test)
- V: Vehicle's longitudinal speed (straight-line speed)
- L: Wheelbase of the vehicle (distance between front and rear axles)
- δ: Steering angle

2) We can use the following equation to relate lateral acceleration (ay), vehicle speed (V), and turning radius (R):

$$ay = V^2 / R$$

3) Calculate the maximum turning radius (R_max) allowed based on the maximum allowable lateral acceleration (ay_max) and vehicle speed (V):

$$R_{max} = V^2 / ay_{max}$$ (V here is the velocity of the car at which is it travelling)

4) Now, we need to establish a relationship between the steering angle (δ) and the turning radius (R). We can use the small angle approximation, assuming small steering angles and low-speed scenarios:

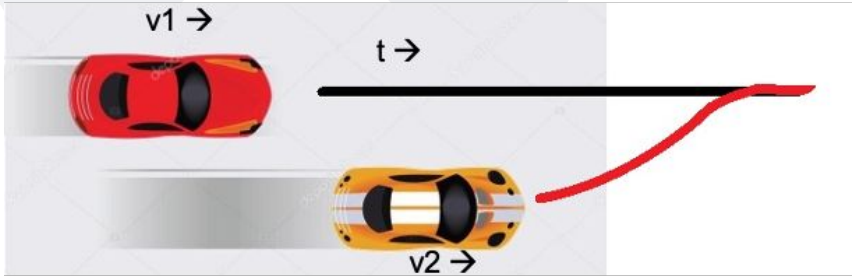
$$R \approx L / \tan(\delta)$$

5) Calculate the maximum allowable steering angle (δ_max) using the maximum turning radius (R_max) and the wheelbase (L):

$$\delta_{max} \approx \operatorname{atan}(L / R_{max})$$

Steering Angle Considerations

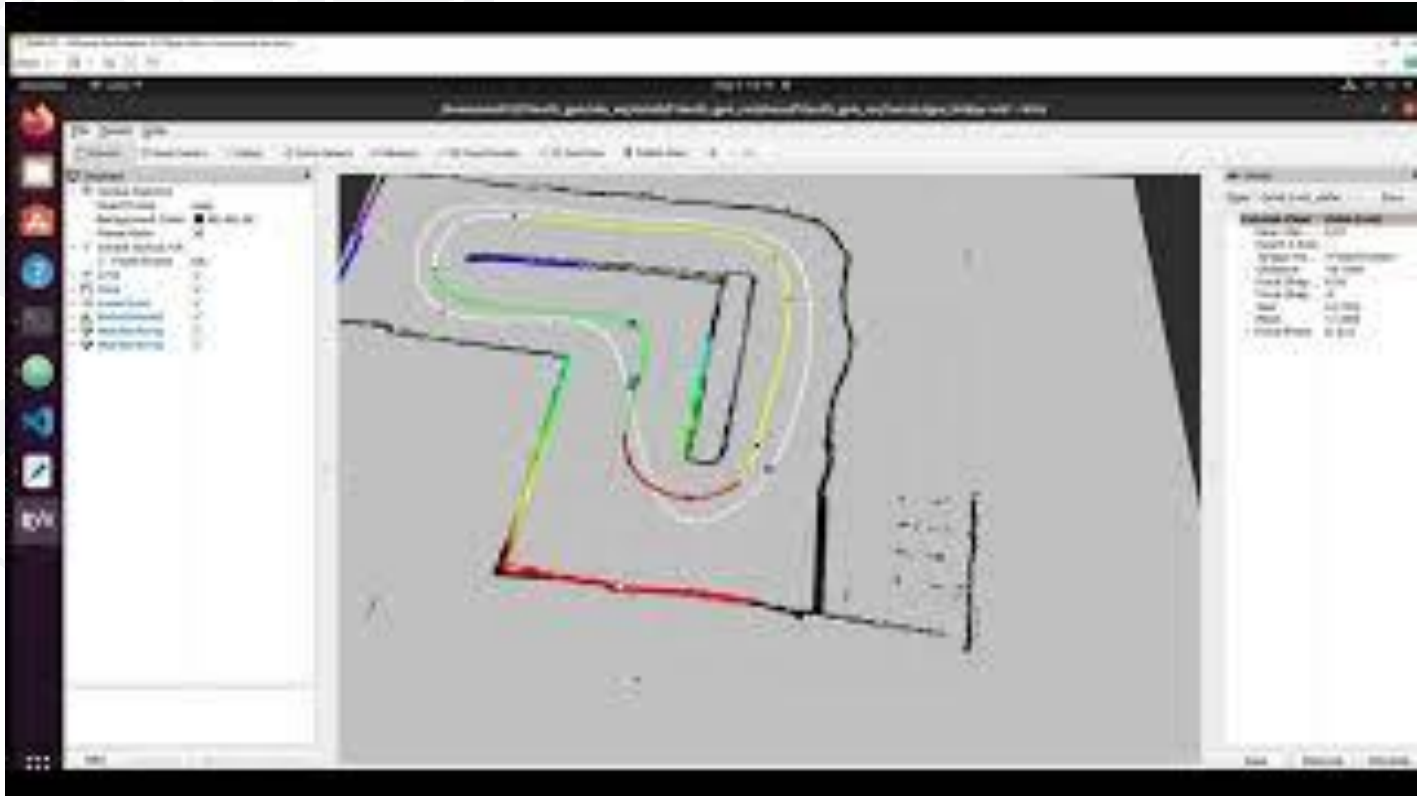
- Added PD controller to smooth out changes in steering angle when merging
- Clamped maximum steering angle when merging according to car's current velocity in order to reduce tire slip



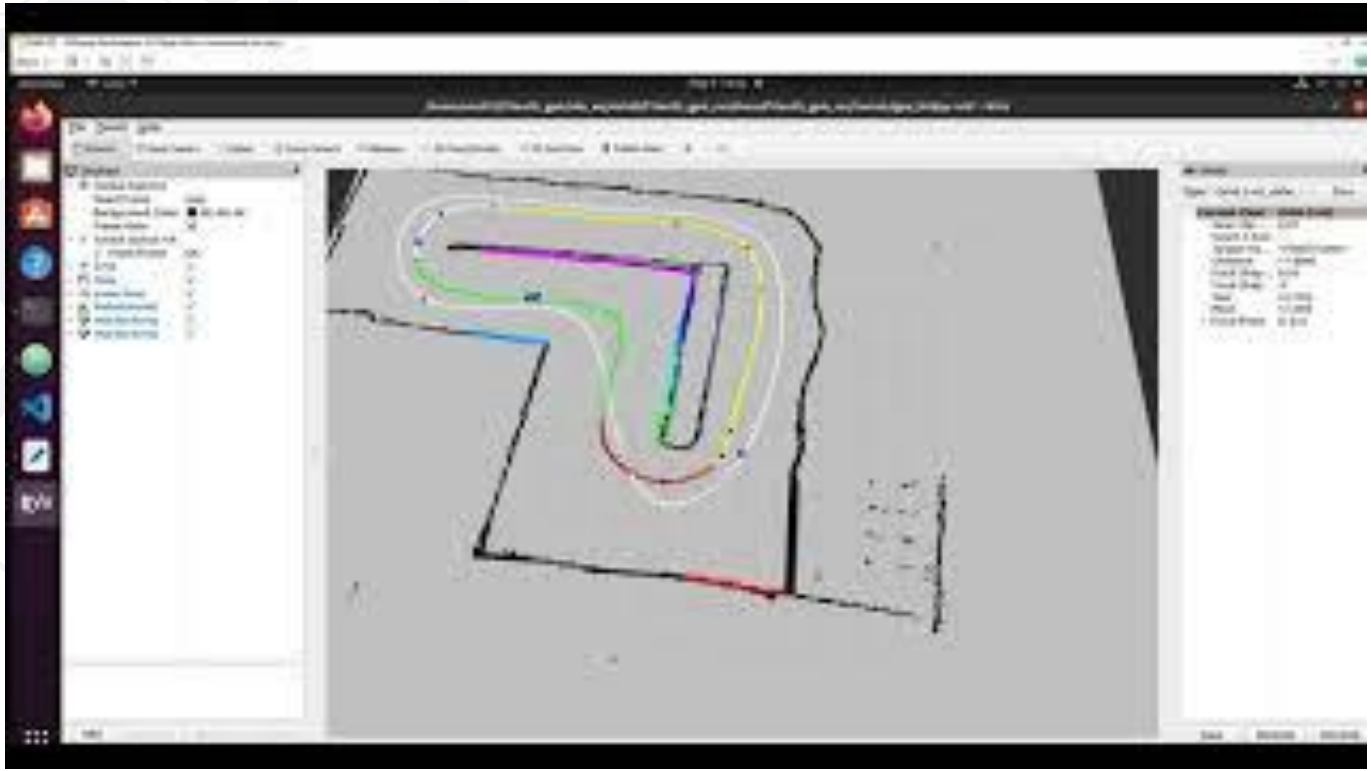
Penn Engineering

Lateral g = (2.4 m/s)^2 / 0.866 = 6.65 = 0.68 g

Without Lateral Control

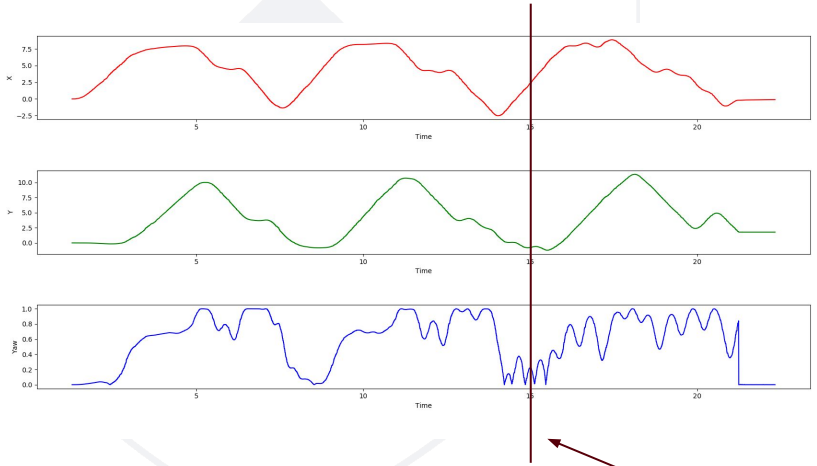


With Lateral Control

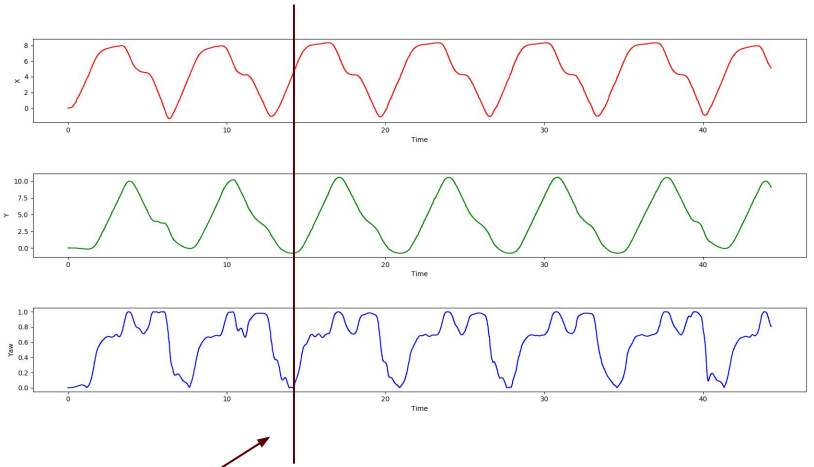


Data analysis

Without lateral control



With lateral control implemented



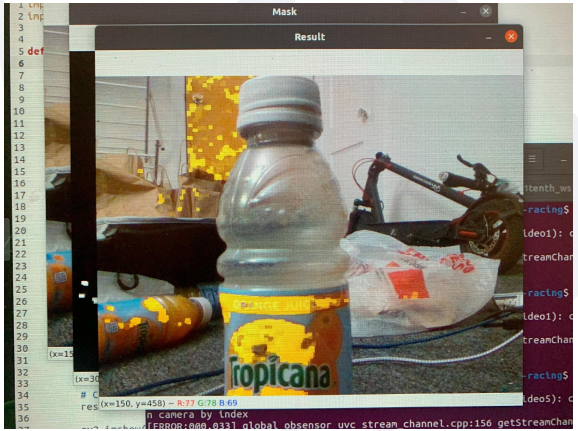
Point where the shift in racing line happens

Vision

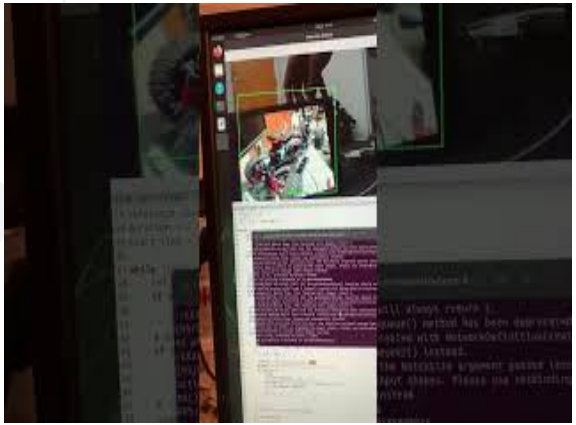
Simple is the best!

Methodology Tested for Detection

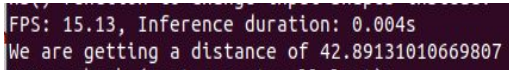
Color based approach



YOLO V5 Detection



April tag detection



Penn Engineering

April tag detection

Procedure:

1) Threshold values for x and y coordinates, tag size, and camera field of view are defined.
2) Image dimensions are obtained from the video capture object, and the focal length in pixels is calculated using the diagonal field of view.
3) A while loop continuously reads frames from the video capture object. a. The frame is converted to grayscale using OpenCV's `cvtColor` function.
4) AprilTags are detected in the grayscale frame using the detector object's `detect` method.
5) Each detected tag is processed, and its center coordinates are calculated.
6) The center of the tag is drawn on the frame, and its coordinates are displayed.
7) The perimeter of the tag is computed, and the distance to the tag is estimated in millimeters.
8) The distance is displayed on the frame.
9) Decisions are made based on whether the tag's center is beyond the threshold and within the distance threshold.

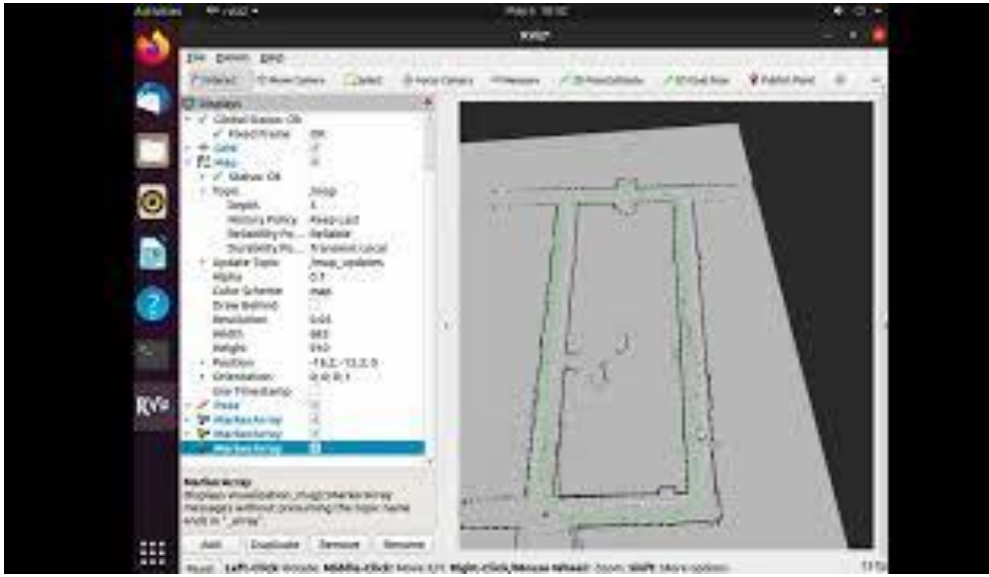
Implementation of lateral controller

WITHOUT LATERAL CONTROL ALGORITHM

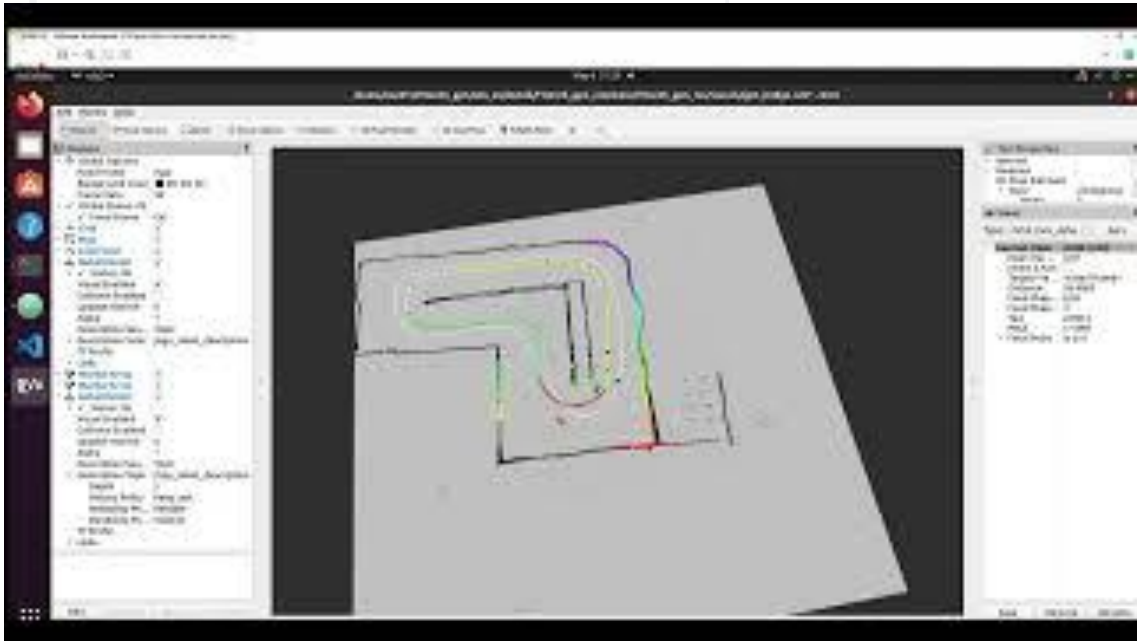
WITH LATERAL CONTROL ALGORITHM

Penn Engineering

Implementation of lateral controller



Defensive racing



Lose some to gain more !

Time spent by each car on the lap:

Ego car timing (original) - 6.308 seconds
Ego race car defensive racing timing - 6.338 seconds

Speed difference between ego and opp car - 1 m/s

Opp car with increased speed (original timing) – 6.1 seconds

Opp car race timing- 6.5 seconds (with our overtake algorithm, changes based on diff approach) increased by 0.4 seconds

Penn Engineering

Future Iterations

1. Test the lateral controller at higher velocities in order to better observe the lateral acceleration limitation
2. Improve opponent detection to get better accuracy - particularly in velocity estimation
3. In real world racing, considerations to manage the car resources, such as energy consumption and tire wear while defending, to ensure optimal performance and reliability throughout the race event. This requires the development of advanced control and optimization techniques that can balance the trade-offs between performance, safety, and resource utilization.

Team 9

Improving the robustness of Local-INN

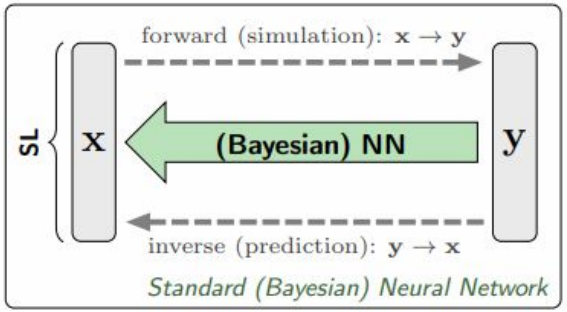
Jason, Jimmy, Patrick, Rohit

Penn Engineering

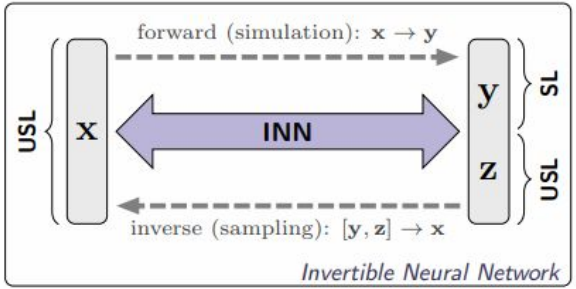
Background

- Invertible Neural Networks (INNs) are bijective function approximators which can achieve both forward and backwards mapping.
- INNs have the capability of being used to solve an inverse problem, which has been utilized in our case to localize the F1Tenth car[1,2]

Standard Neural Network



Invertible Neural Network



Penn Engineering

[1]: Analyzing Inverse Problems with Invertible Neural Networks
[2]: Local_INN: Implicit Map Representation and Localization with Invertible Neural Networks

INN is a cheaper option for localization

- INN can be a superior, drop-in replacement to particle filtering due to its high localization frequency.

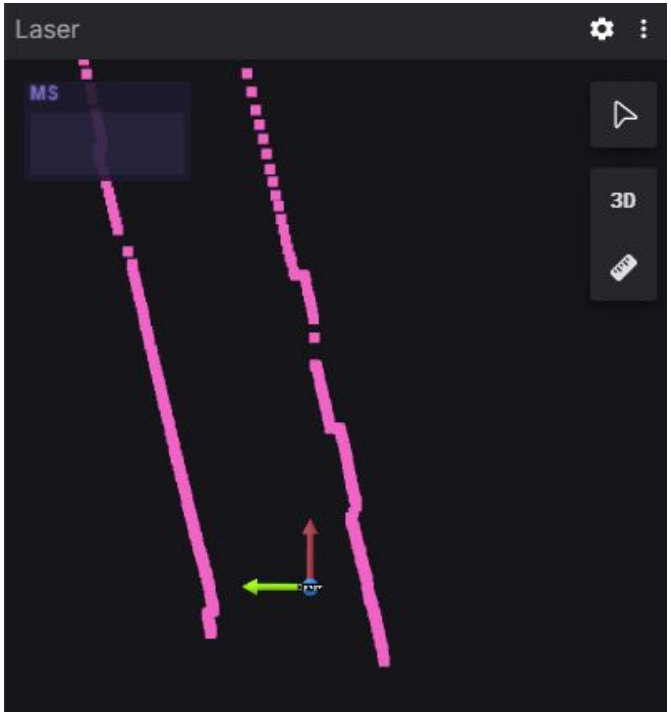
- A comprehensive assessment approach for INN will enable us to conduct a comparative analysis with its rival technologies, thereby facilitating the identification of areas that require enhancement.

- Through this evaluation, we can refine the local INN implementation, ensuring its performance aligns with those of alternative models, as INN offers a more cost-effective and time-efficient solution compared to other localization techniques.

The current INN method is not robust

- Current implementations of INN lose localization at locations with low distinguishable features[2].

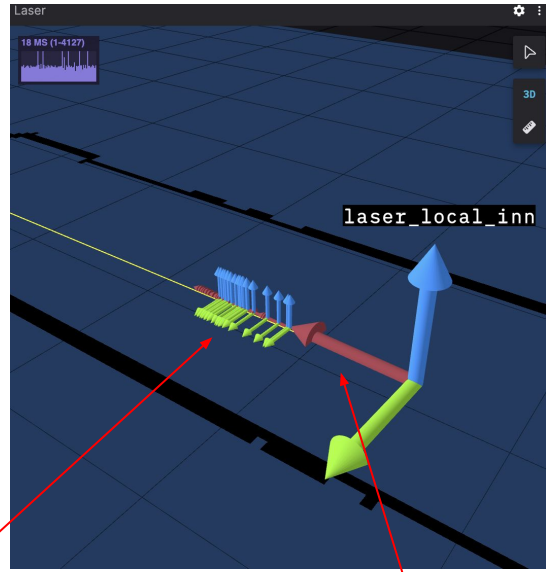
- Without external information like speed and acceleration, the output pose can be highly uncertain



[2]: Local_INN: Implicit Map Representation and Localization with Invertible Neural Networks

Improving Robustness: Sensor Fusion

- Local_INN outputs plausible poses of the car given the laser scan, but outputs can be inconsistent
- Existing EKF solution is insufficient for racing - loses localization at Levine after one lap.
- We aim to improve the robustness of Local_INN by incorporating IMU into our nonlinear sensor fusion technique



Local_INN estimates

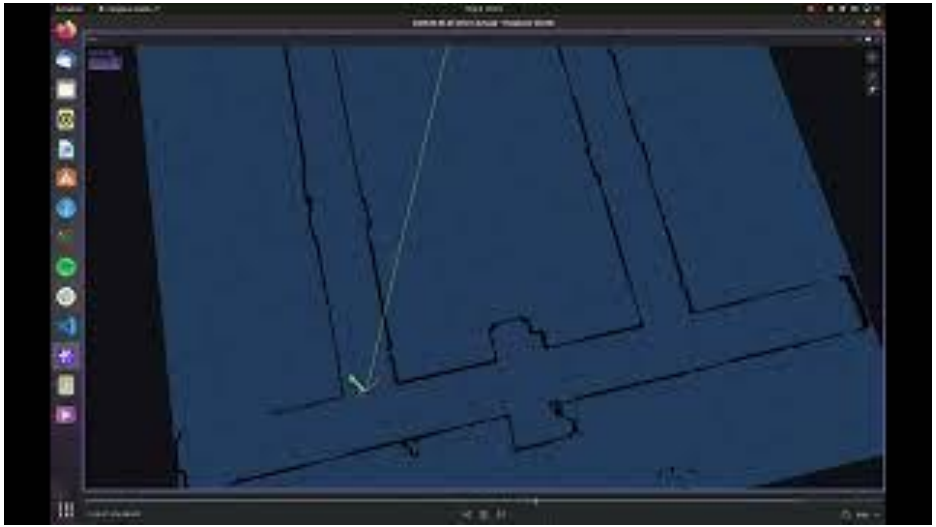
Fused Pose

Fusion Technique

- We use the Unscented Kalman Filter to estimate position, velocity, acceleration, yaw, IMU bias.
- The filter propagates future states using wheel odometry data and nonlinear transformations
- Two Kalman separate updates are used: one using pose from Local_INN, and one using IMU accelerometer data

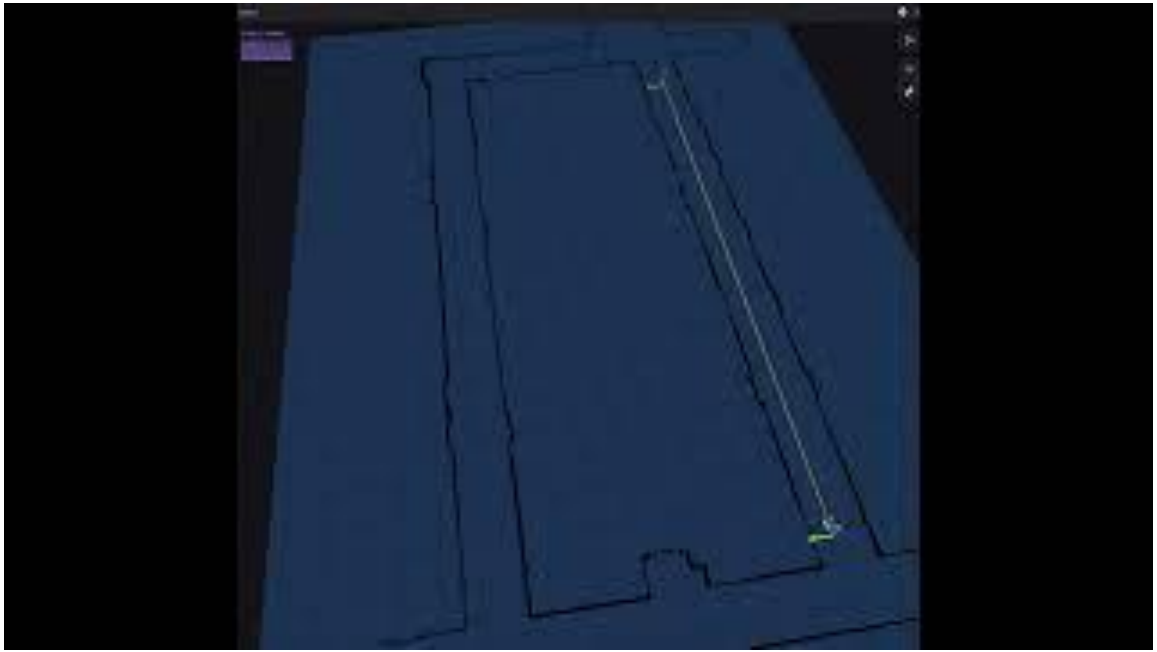
Challenges

- As seen in this video, our team faced difficulties with certain spots of the Levine track.
- We believe that we faced these difficulties for a couple different reasons:
  - Our Unscented Kalman Filter was not active initially
  - Certain portions of Levine track are not distinguishable from the other.



Results

We demonstrate the improved robustness of the localization by running Local_INN at Levine for multiple laps without losing tracking



Penn Engineering

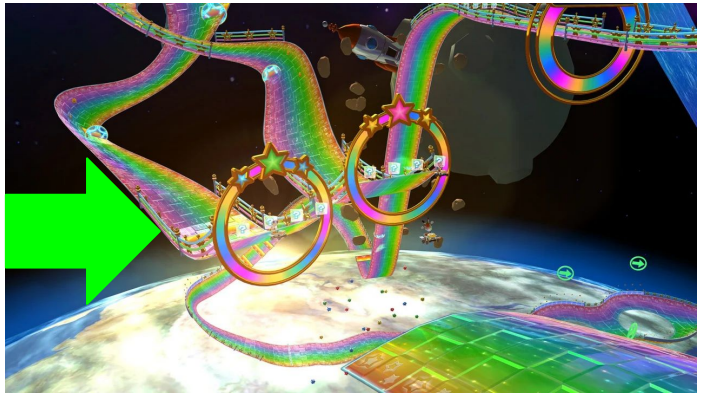
Evaluation

- We believe that we overcame the initial issue of INN breaking at locations with no distinguishable features based on our final run of INN in the Levine track.

- The proper implementation of Unscented Kalman Filter allowed our estimates to be a lot more accurate for localization and updated quickly as our car moved through the halls of Levine!

Future Directions

- Had we had more time to test our implementation of INNs, we would have liked to test on different tracks than Levine hall to see how our code stacked up against different environments.
- We also would have compared INN to differing techniques like ORB SLAM 3.



Thank you!

Penn Engineering

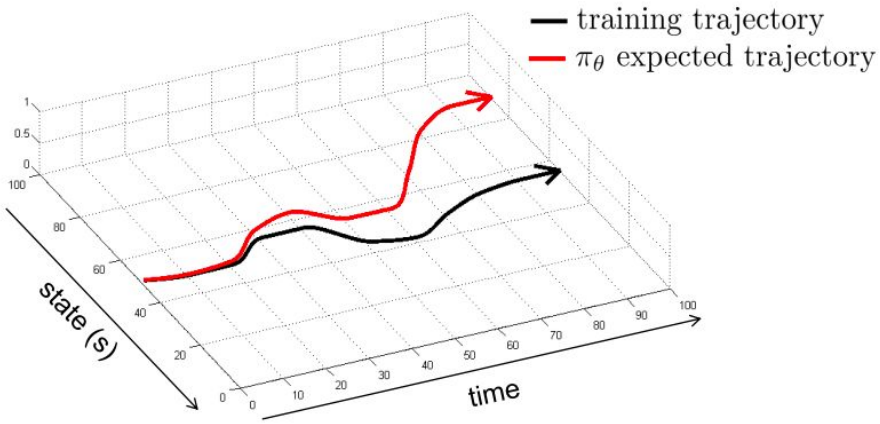
Team 10

Learning CBFs for safe imitation

Penn Engineering

Introduction

Distribution Shift is a big problem in imitation learning



Known Solutions: Collect more data from the expert (DAGGER) or collect more data online (RL)

Another Solution

If the learnt policy takes you out of the training distribution, bring it back into distribution.

- Offline RL (pessimism towards exploration)
- **Control Barrier Functions**

Objective:

1. **Learn a control barrier function to distinguish between safe regions and unsafe regions**
2. **Use CBF to execute a 'fail safe' policy in unsafe regions (everything is learned)**

Neural Lyapunov-Barrier functions (Intro)

A function $V : X \rightarrow R$ is a CLBF if

$$V(x_{\text{goal}}) = 0 \tag{1a}$$

$$V(x) > 0 \; \forall \, x \in \mathcal{X} \setminus x_{\text{goal}} \tag{1b}$$

$$V(x) \leq c \; \forall \, x \in \mathcal{X}_{\text{safe}} \tag{1c}$$

$$V(x) > c \; \forall \, x \in \mathcal{X}_{\text{unsafe}} \tag{1d}$$

$$\inf_u L_f V + L_g V u + \lambda V(x) \leq 0 \; \forall \, x \in \mathcal{X} \setminus x_{\text{goal}} \tag{1e}$$

- The system dynamics are assumed to be control affine, i.e. $\dot{x} = f(x) + g(x)u$
- L_f and L_g is the Lie derivative of V along f(x) and g(x), can be calculated using the formula $\frac{\partial V}{\partial x} f(x)$

CLBF and Policy Learning

Loss Function for learning CLBF

$$L = V(x_{goal})^2 + \frac{a_1}{N_{safe}} \sum_{x \in \mathcal{X}_{safe}} [\epsilon + V(x) - c]_+ + \frac{a_2}{N_{unsafe}} \sum_{x \in X_{unsafe}} [\epsilon + c - V(x)]_+$$
$$+ \frac{a_3}{N_{\text{train}}} \sum_{x \in X} [\epsilon + L_f V(x) + L_g V(x) \pi_{NN}(x) + \lambda V(x)]_+$$

CLBF and Policy Learning

To learn the policy simultaneously, and additional term is added to the loss function

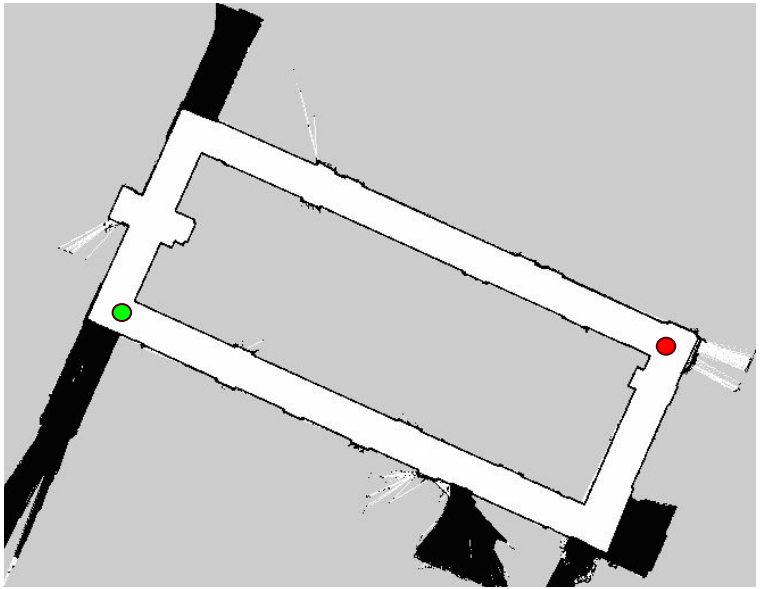
$$L_\pi = ||\pi_{NN} - \pi_{nominal}||^2$$

The combined loss

$$L = L_{CBF} + \alpha L_\pi$$

Problem Formulation

- Green: Start
- Red: Goal
- Nominal controller: Pure pursuit controller
- State: [x,y,steering angle, velocity and yaw]
- Control: [velocity, steering angle]

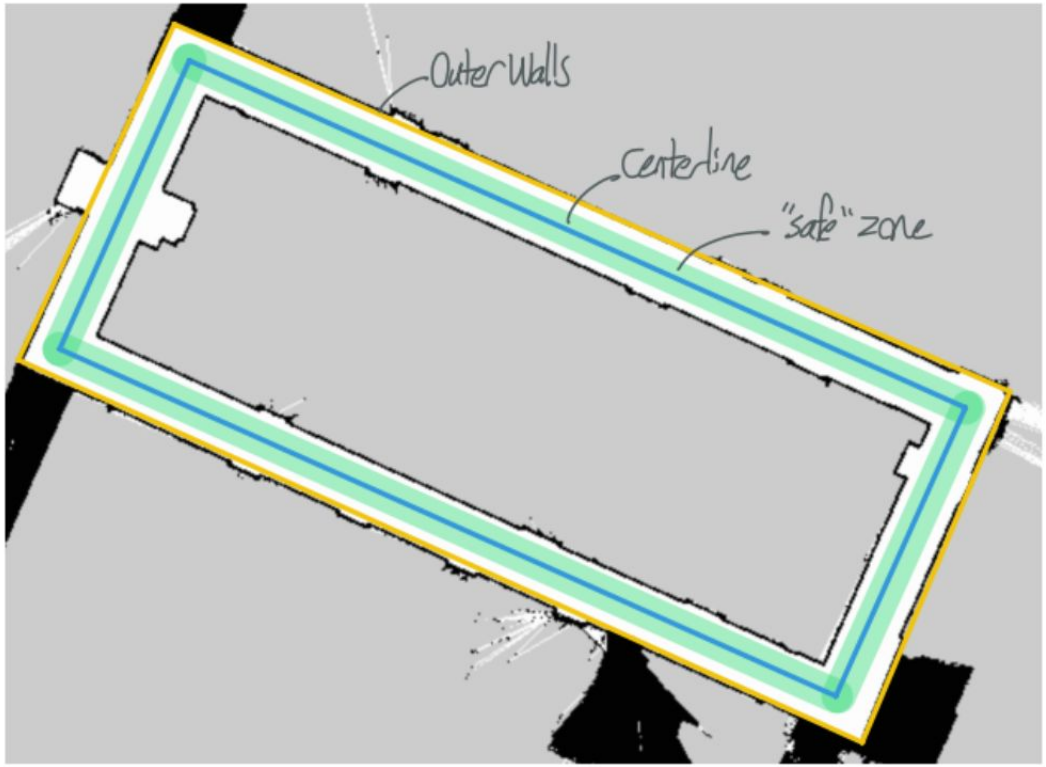


“Safe” Point Validation

Validates given state as “safe”

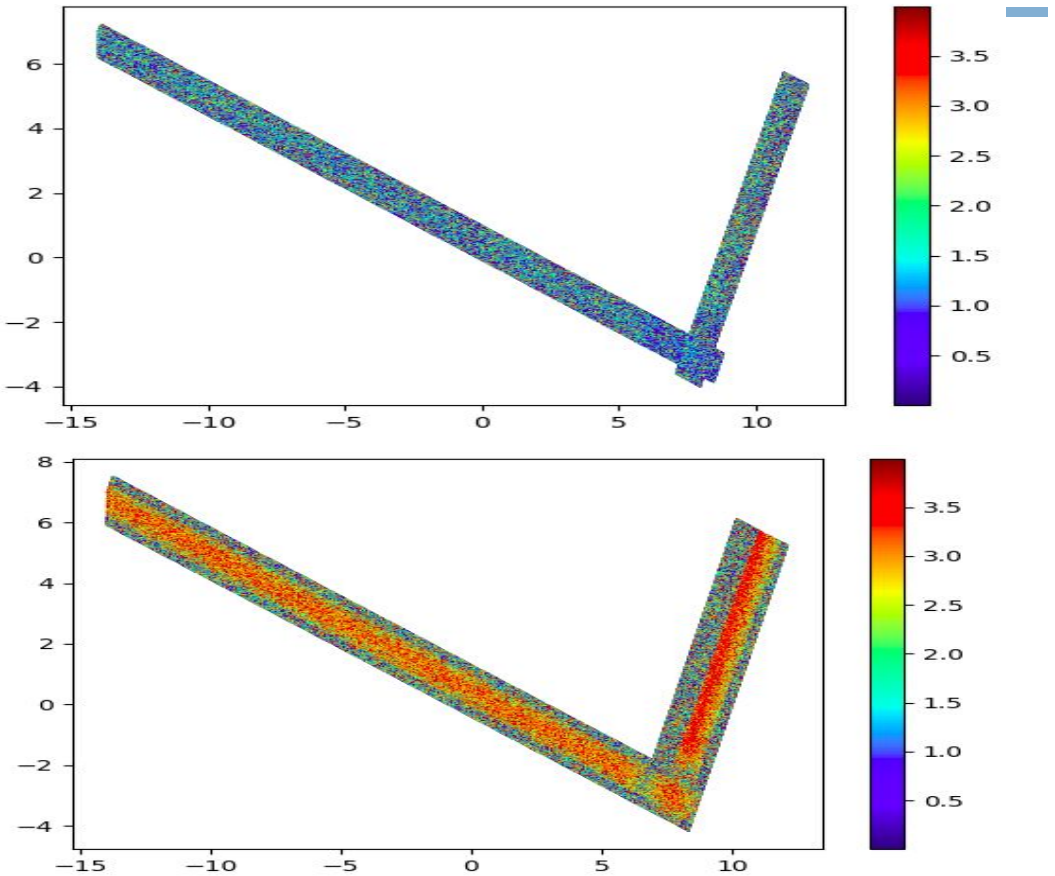
Input: pos, heading, vel, yaw rate
Output: 1 (safe), 0 (unsafe)

How:
(within outer walls)
/\ (within centerline tolerance)
/\ (no critical TTC w/ walls)



Data collection

- Sample 1 million states and label them safe or unsafe
- Compute nominal controller with pure pursuit
- Left top: Safe Right Top: Unsafe Colours: Velocity

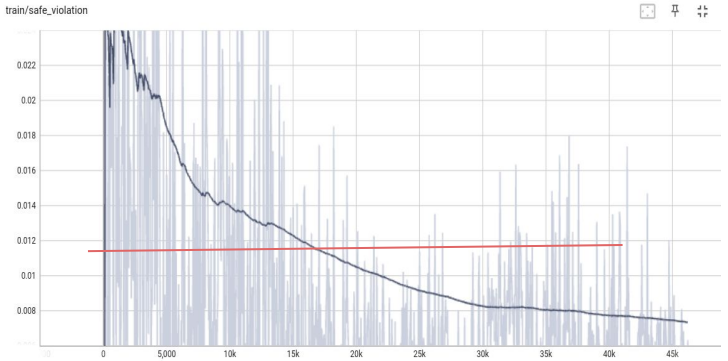


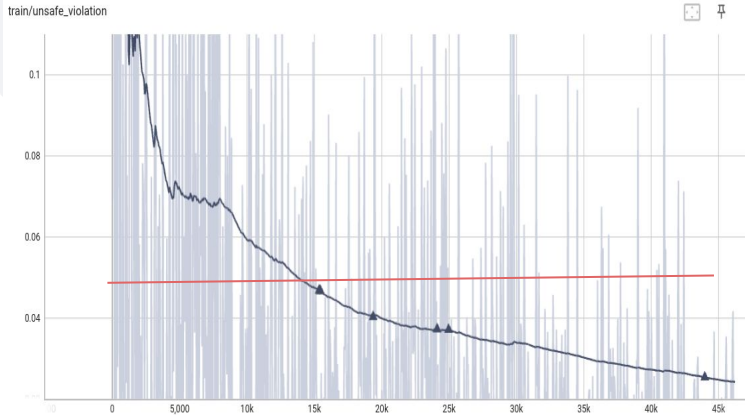
Learning

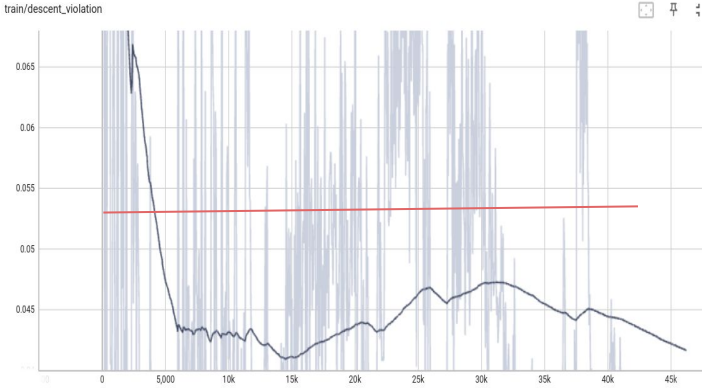
Policy violation (<0.02)

Safe violations (<0.05)

Unsafe violations (<0.05)



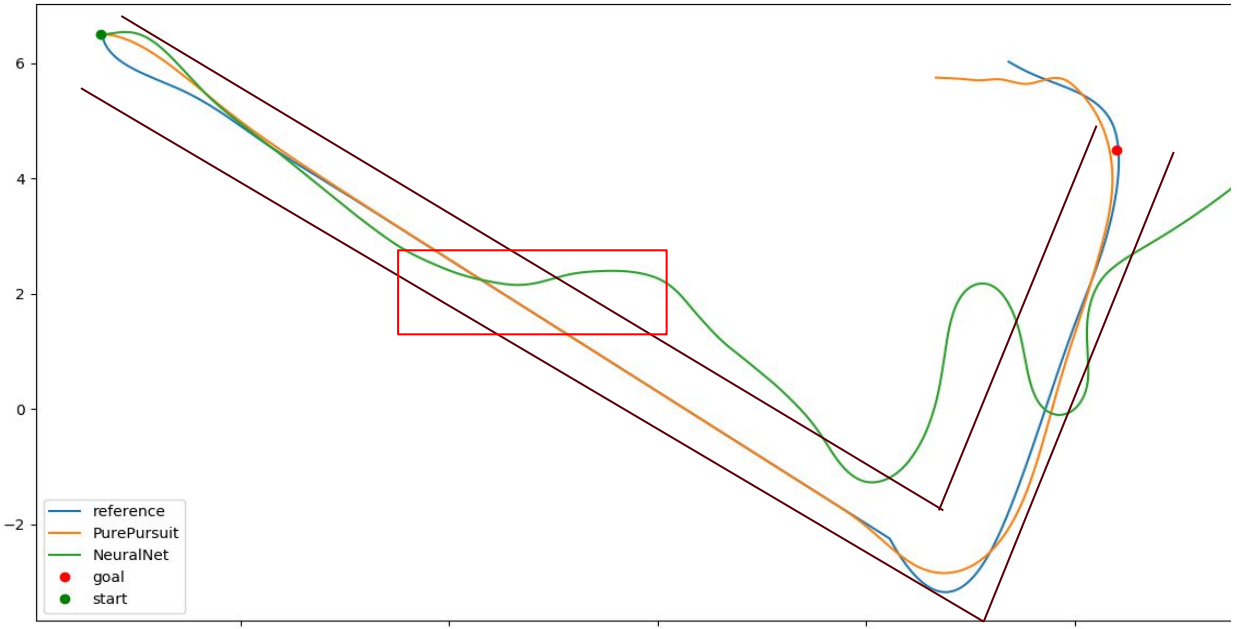




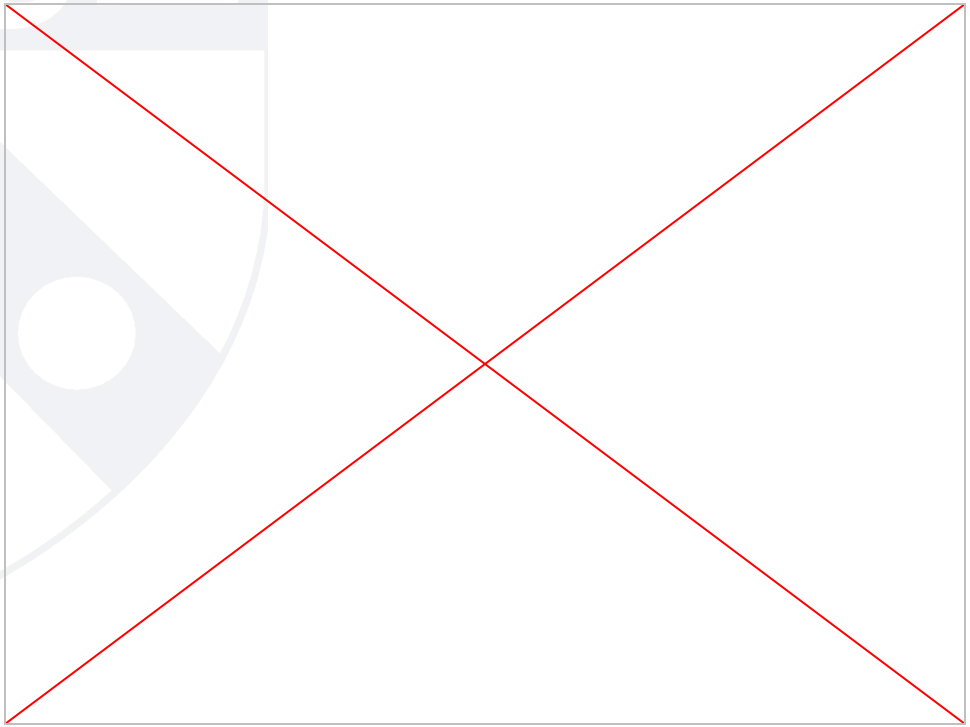
Penn Engineering

Results (Simulated Dynamics)

We still face potential training data issues that lead to un-robust performances in un-safe regions



Sim Results



Real World Results

- We have deployed the tensorrt version of the learnt policy on the real car but real execution is pending simulation verification

- Potentially a data issue where we randomly sample states within the levine hall bounds, it can be biased more towards the expert trajectory

Thank you!

Penn Engineering

Team 11

Penn Engineering

Lightning MPCQueen: Safe Blocking Strategy using Model Predictive Control

Team 11

Sachin Pullil, William Hoganson,
Aurunima Samaddar, Ryan Tong

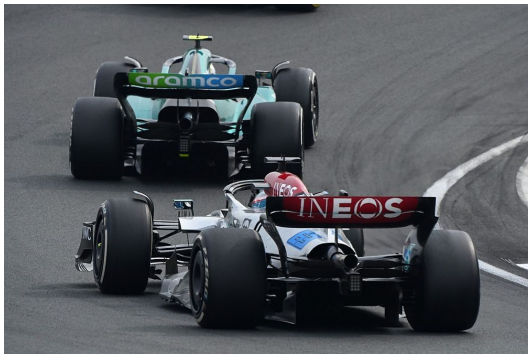
Penn Engineering

Motivation

Position defense: Maintaining lead in the race by taking defensive measures

Blocking: if an opponent car attempts to overtake the lead car, the lead car may swerve to block the predicted path of the opponent's vehicle.

In F1/10: Not utilised often in F1 racing due to the risk on the drivers. F1/10 gives us an unique opportunity to test the validity of blocking in autonomous racing at a lower risk level.



Blocking MPC VS Optimal Raceline MPC Experiment Setup - 1

- In our experiment, **overtaking** car starts at a distance of 2 meters behind **blocking** car.
- Overtaking car speed of 0.66 Blocking car has constant speed of .5 m/s. (handicap of 1.3)
- Overtaking car driven using teleop. Blocking car ran either raceline or blocking MPC.
- Safety node in overtaking car that would automatically decrease the speed if a collision is imminent.

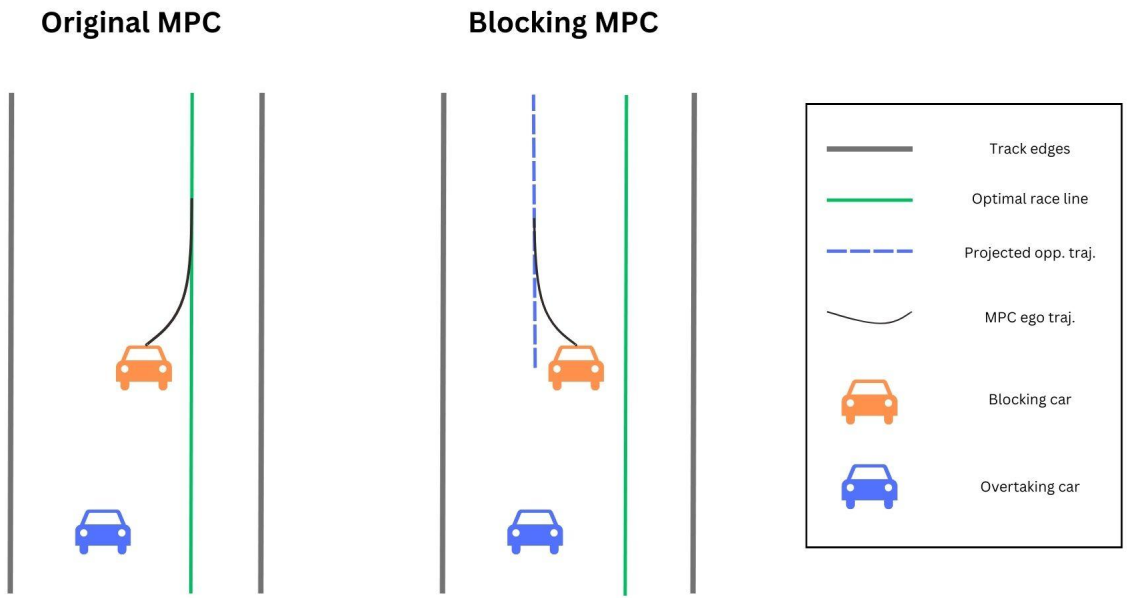
Blocking MPC VS Optimal Raceline MPC Experiment Setup - 1

- Blocking car had costs on change in steering angle to prevent violent steering shifts and improve safety.
- All races conducted in simulation on the straightaway of Levine 2nd floor.
- We ran 10 control races where the lead car was running regular MPC to follow the optimal raceline and 10 races where the lead car was running our blocking MPC.

Extension

Additional physical analysis to determine the maximum overtaking speed where our blocking MPC was effective: 2.68m/s, a handicap of 5.4.

Our MPC Blocking Algorithm



Penn Engineering

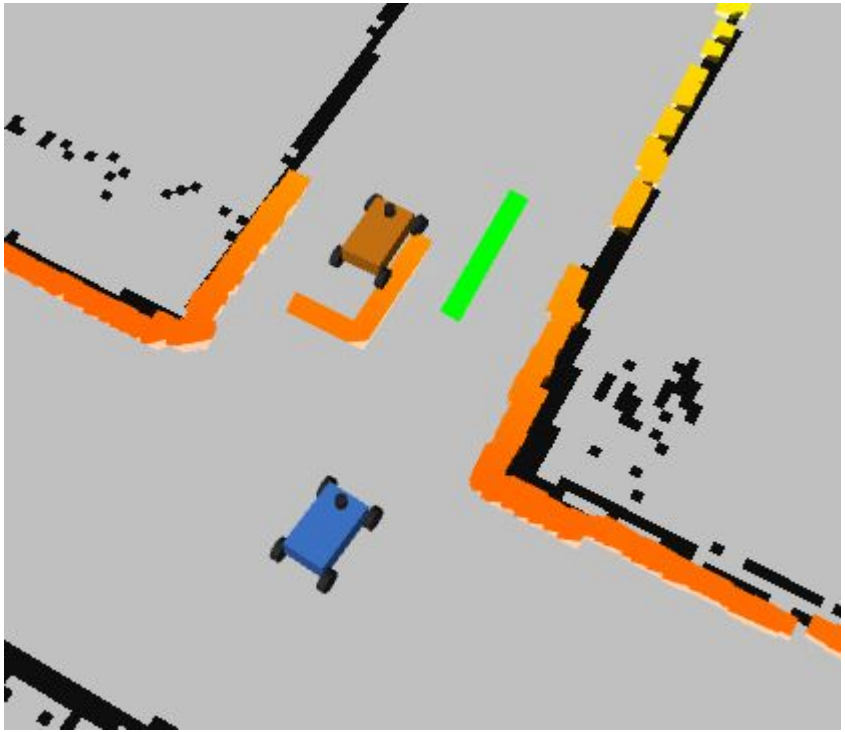
Our MPC Blocking Algorithm

$$(x_N - x_r)^T Q_f (x_N - x_r) + \sum_{k=0}^{N-1} [(x_k - x_r)^T Q_k (x_k - x_r) + u_k^T R u_k + (u_k - u_{k-1})^T R_d (u_k - u_{k-1})]$$

$$\begin{aligned} \text{subj. to} \quad & x_{k+1} = Ax_k + Bu_k, \ k = 0, \ldots, N-1 \\ & x_k \in \mathcal{X}, \ u_k \in \mathcal{U}, \ k = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{aligned}$$

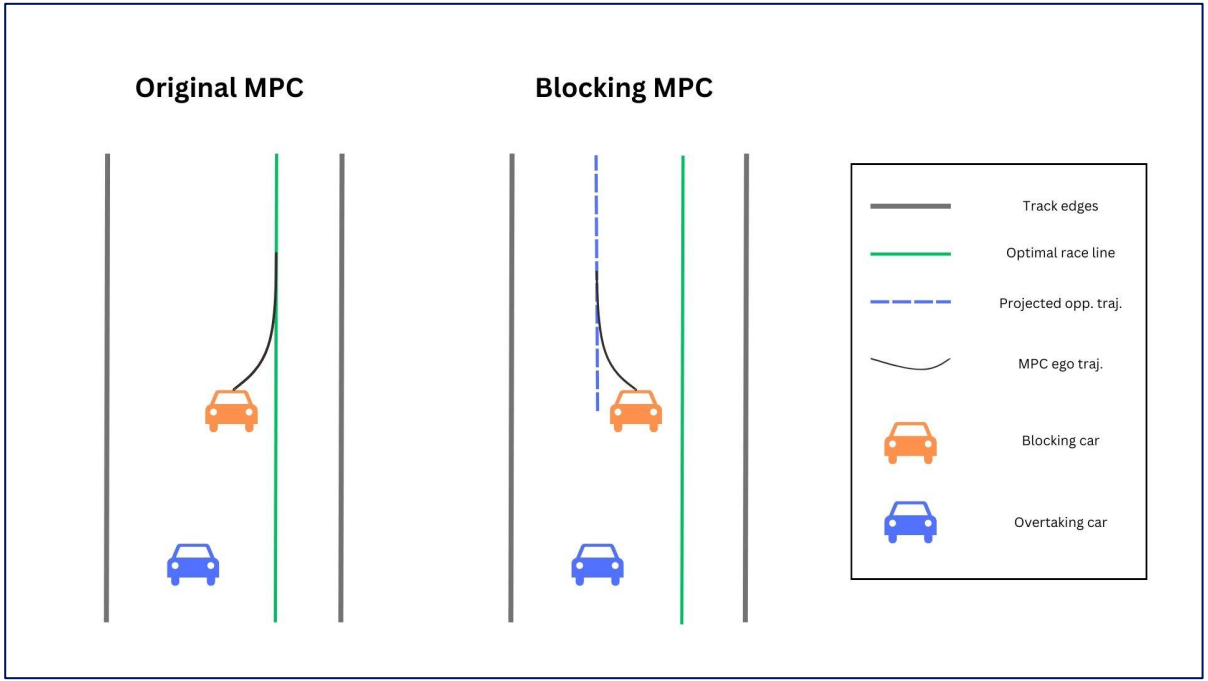
x_r = projected trajectory of the overtaking car for N timesteps.

Our MPC Blocking Algorithm



Penn Engineering

Hypothesis



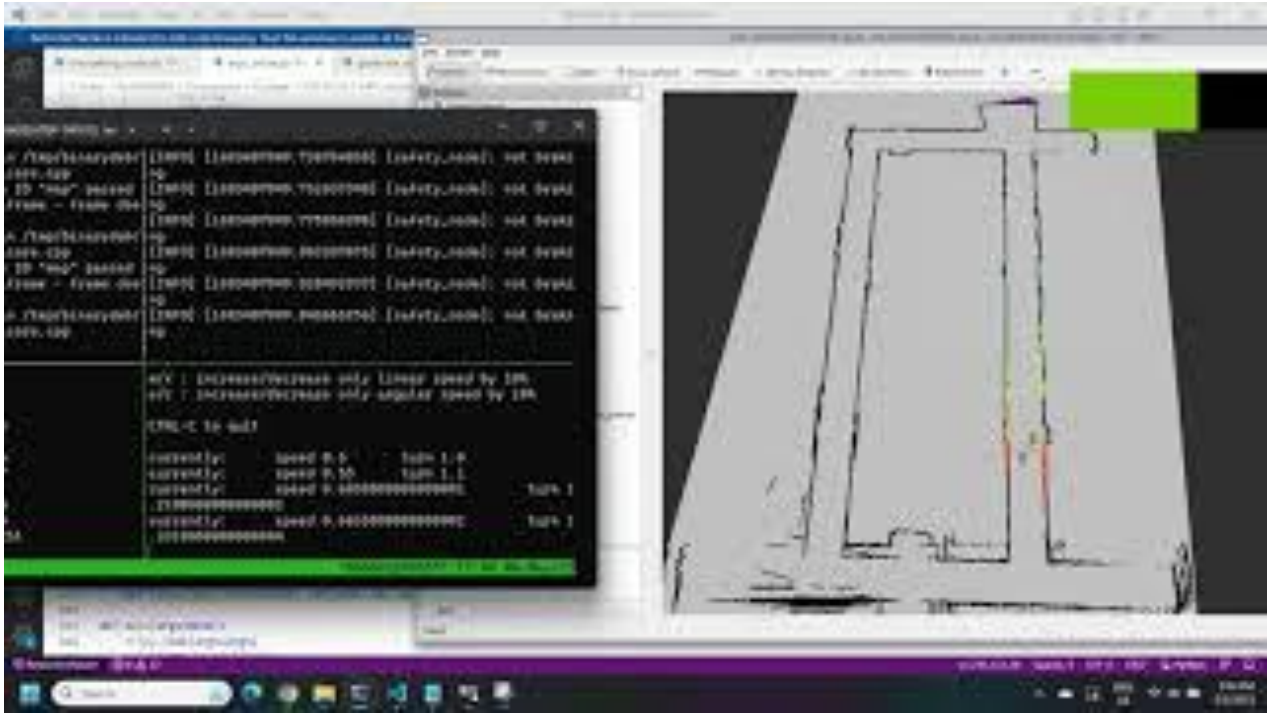
We believe that given the race parameters we outlined, blocking will consistently prevent overtaking more frequently than following the optimal raceline.

Results

Simulation: using a human driver, we successfully overtook a car using raceline MPC 10/10 times. However, we failed to overtake the lead car when it used blocking MPC unless we had a handicap of 5.3. The safety node would activate and force us to slow down to prevent crashing.

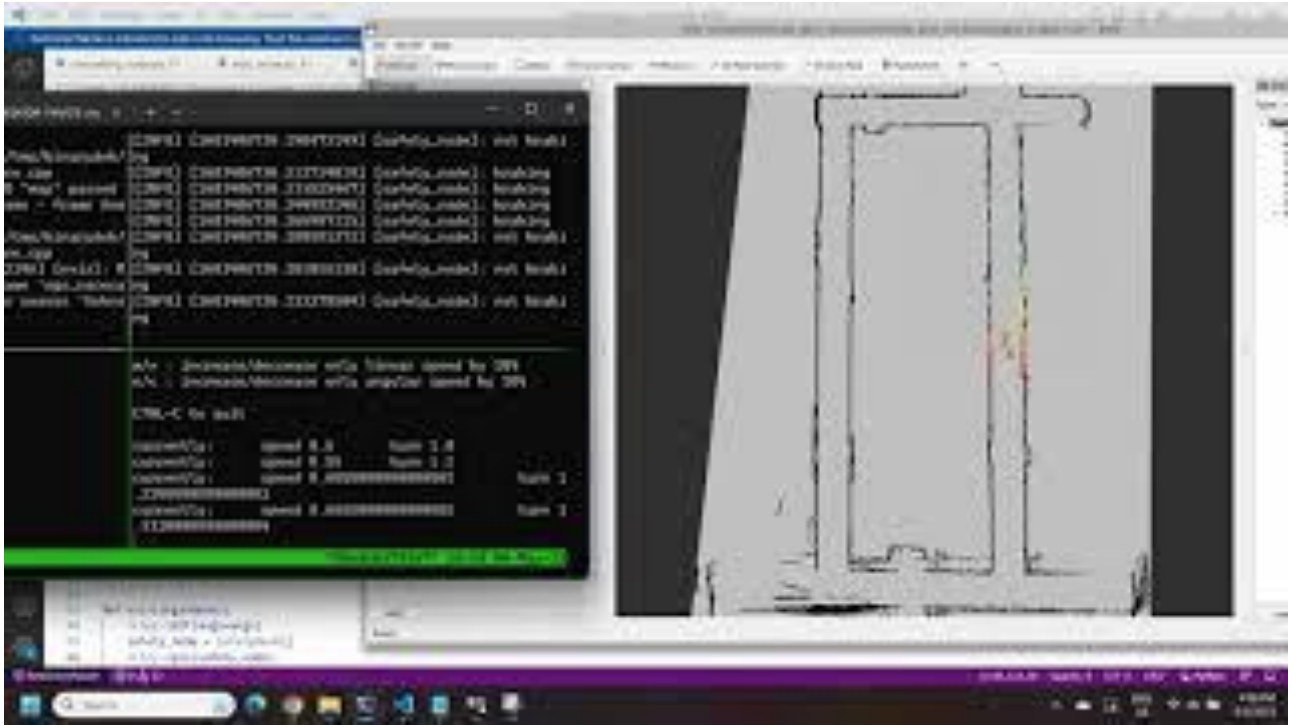
Real car: MPC can successfully track the trajectory of the opponent's car. However, MPC's inherent problem of not returning any solutions if it cannot find an optimal one leads to frequent stops.

Regular MPC vs Manual Overtake - 0.67m/s



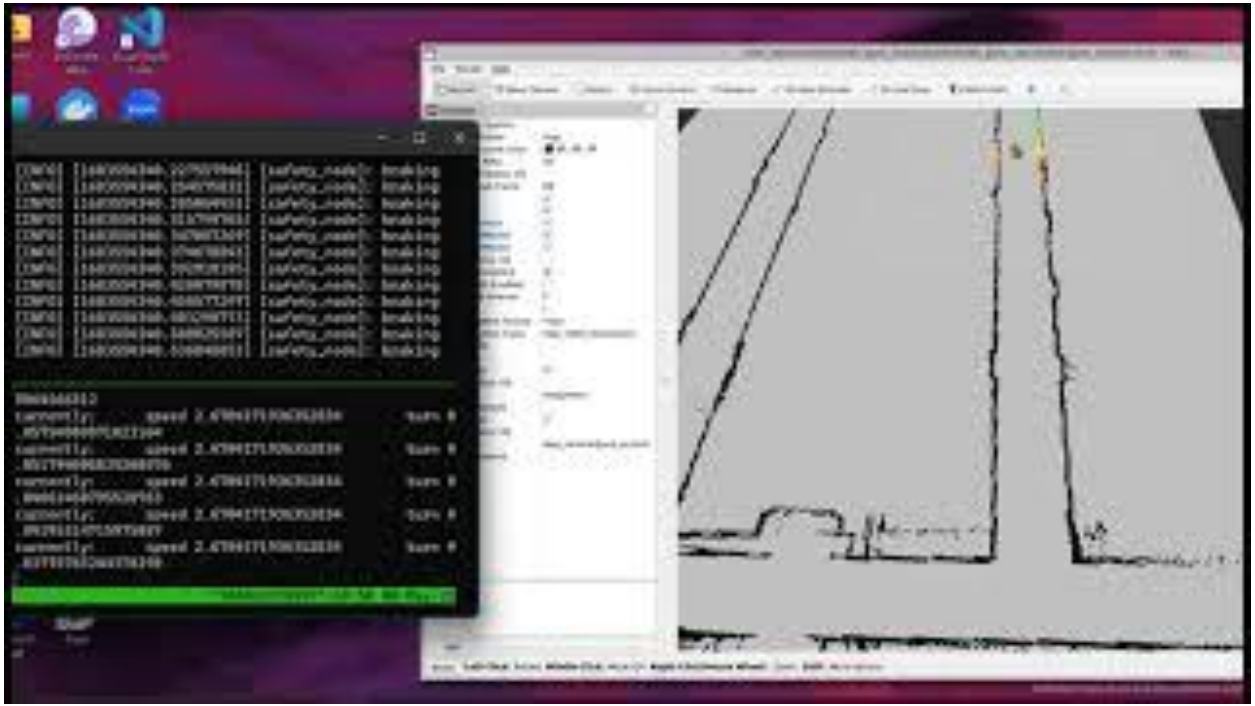
Penn Engineering

Blocking MPC vs Manual Overtake - 0.67m/s



Penn Engineering

Blocking MPC vs Manual Overtake - 2.68m/s



Penn Engineering

Real world

Penn Engineering

Conclusions

From this experiment we were able to determine that **blocking** can be considered a viable racing strategy in autonomous racing.

Safe blocking is realised when we ensure that the participating cars are considerate of collision avoidance and safety node automatically decreases the speed (of the overtaking car) if a collision is imminent.

Future work

- Alternate MPC blocking strategies
    - Combining ideal raceline and blocking
- Blocking on turns
    - Dangerous and more complex
- Automated overtaking
- Robust real world testing

Acknowledgements

- Prof. Mangharam and the teaching team
- Team 6 for letting us borrow their car
- Our team!

Questions