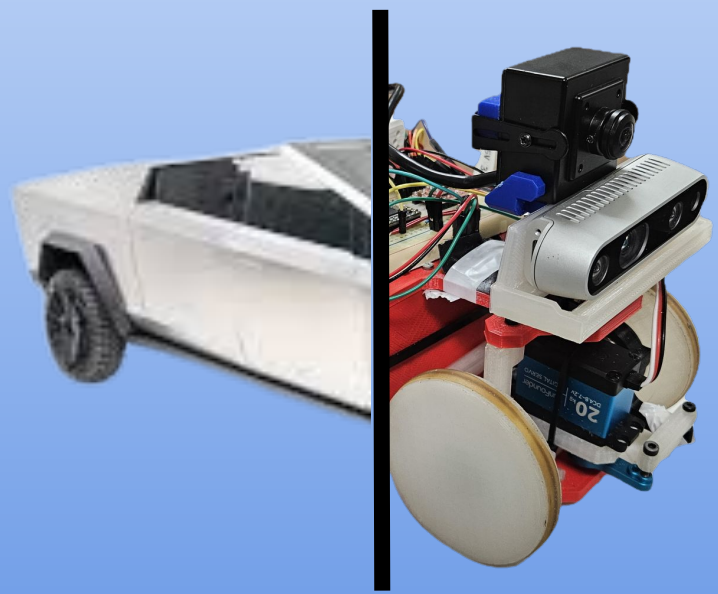
Self Driving Car

Team: Michelle, Chris, Varun, Thierry, and Anjali
Advisors: Ivan Seskar and Jenny Shane



The Team

Michelle Gutwein

Chris Lee

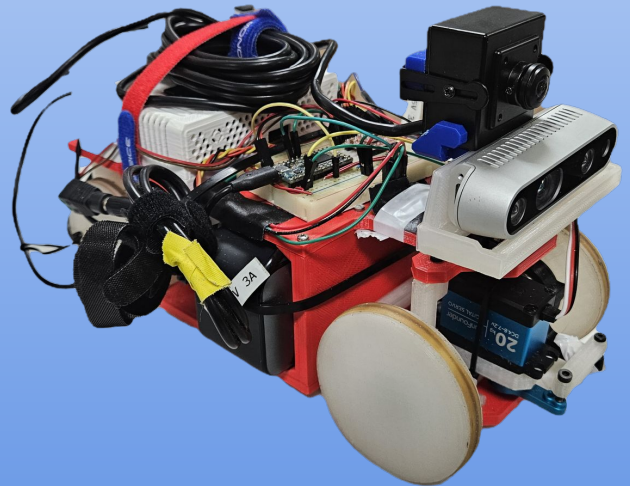
Varun Chirravuri

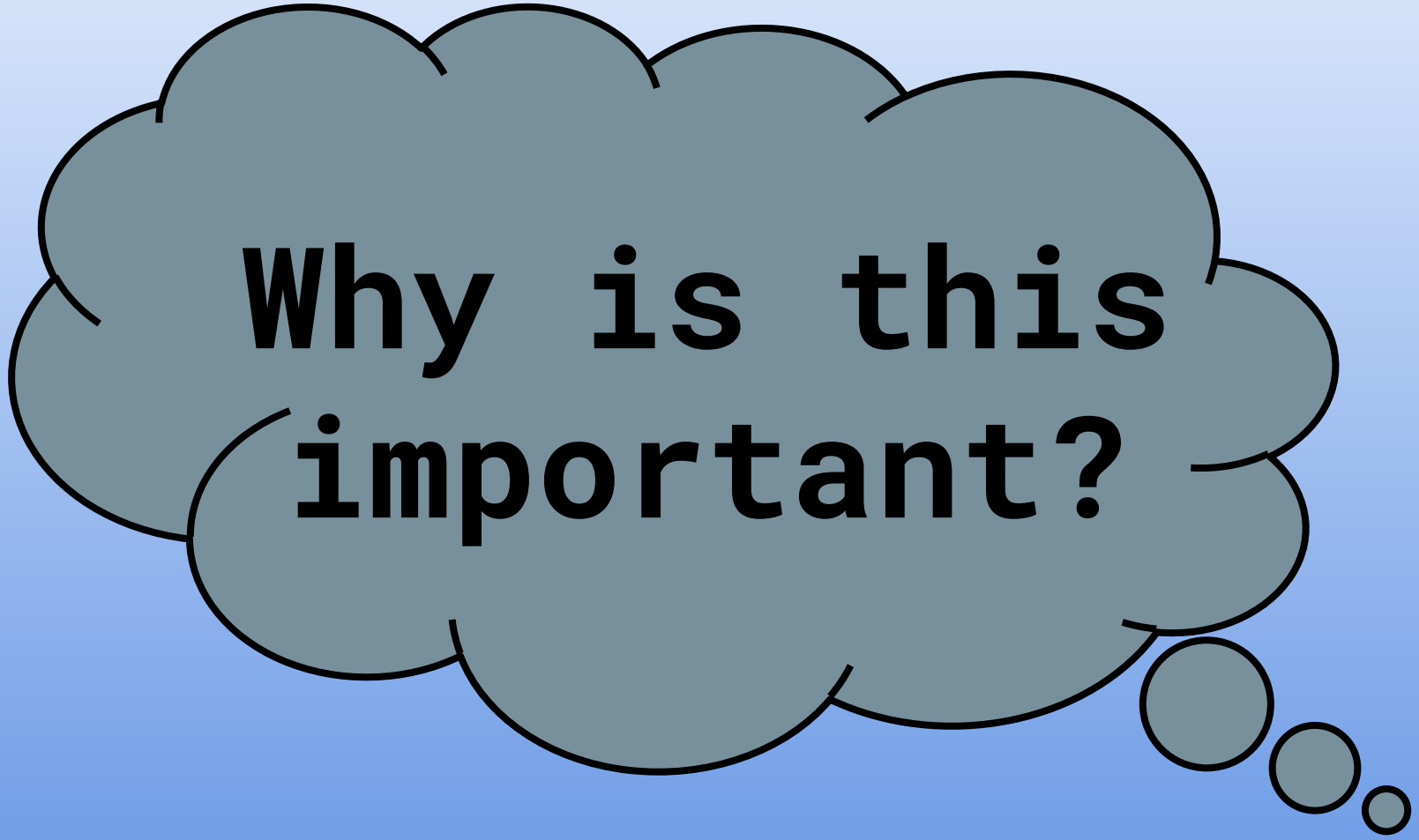
Thierry Antoine

Anjali Kapilavai

Overview

Our goal is to train miniature autonomous cars, using machine learning, to drive in a miniature city, modeled off of real streets in Manhattan. It should be able to navigate from a start to an end point.




Why is this important?

Safety

- Low-speed, small-scale vehicle eliminates risk of harmful collisions
- Avoids real-world hazards like traffic, pedestrians, or property damage
- Enables rapid iteration and failure recovery

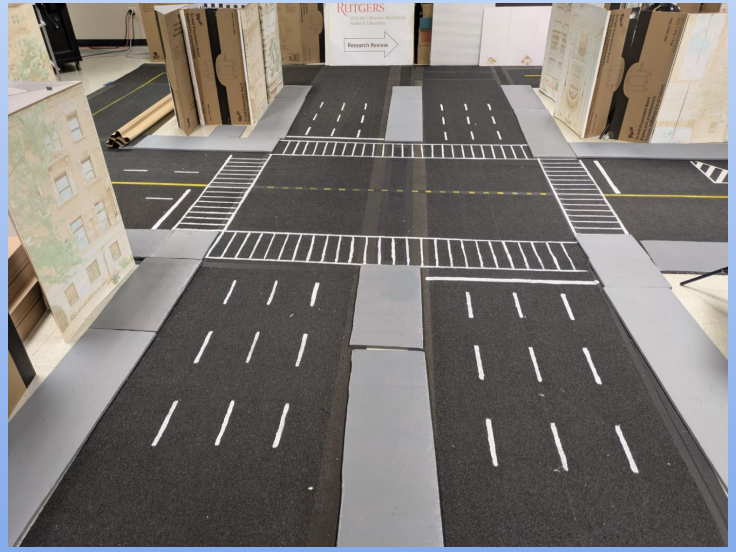
Cost

- Total hardware cost: only $650.83
- Significantly cheaper than using full-sized vehicles, which can cost tens of thousands of dollars
- Enables affordable, scalable testing for research teams and classrooms

Item	Quantity	Total Cost	Narrative/Purpose
Bourns Optical Encoder	1	$51.52	Sensor for distance calculation
Nema 17 Stepper Motor	1	$10.65	Motor to control speed
A4988 Stepper Motor Driver	1	$8.09	Chip to control stepper motor
20kg Servo Motor	1	$19.18	Motor to control steering
Anker 737 Power Bank	1	$85.29	Battery bank to power Jetson Nano and steppe
15V Cable	1	$11.72	Cable from battery to motor
Micro-USB Cable	2	$8.51	Cable to power Teensy 4.0 and Jetson Nano
400-Point Breadboard	1	$7.20	Breadboard for electrical components
Male to Female jumper wire	8		For electrical components
Male to Male jumper wire	20		For electrical components
Wifi Adapter	1	$10.65	WiFi adapter for Jetson Nano for wireless conn
100-pack M2.5 Heat-Set Inserts	1	$18.72	Screw inserts in chassis
50-pack M2.5 Bolts	1	$13.02	Bolts for chassis
50-pack M3 Bolts	1	$10.66	Bolts for Nema 17 Stepper Motor and 20kg Ser
M2 Shoulder Bolts	10	$41.58	Bolts for chassis
10-pack M2 Lock Nuts	1	$9.92	Nuts for chassis
4-pack Ball Bearings	2	$27.70	Ball bearing for wheels
3D Printing Filament	1		Chassis material
Zipties	Numerous		Cable management
Rubberbands	4		Tire traction
		$650.83	

Realism

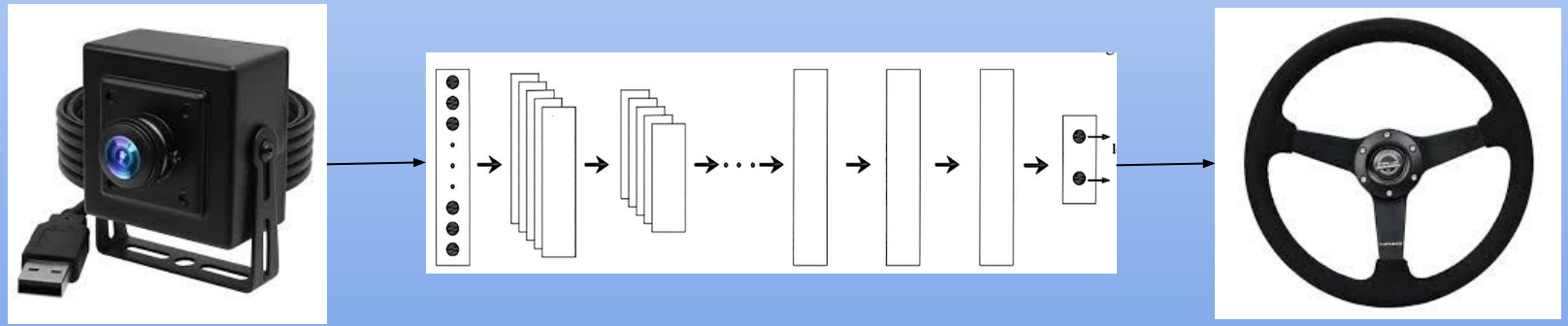
- Provides a scaled environment for path planning
- Helps researchers develop intuition for the challenges of real-world autonomy
- Supports real-time data collection from cameras, IMUs, and other sensors – just like in full-scale autonomous vehicles



How it works

Convolutional Neural Network (CNN)

- Takes in images and labels as an input
- Trained to predict the steering angles based on inputs

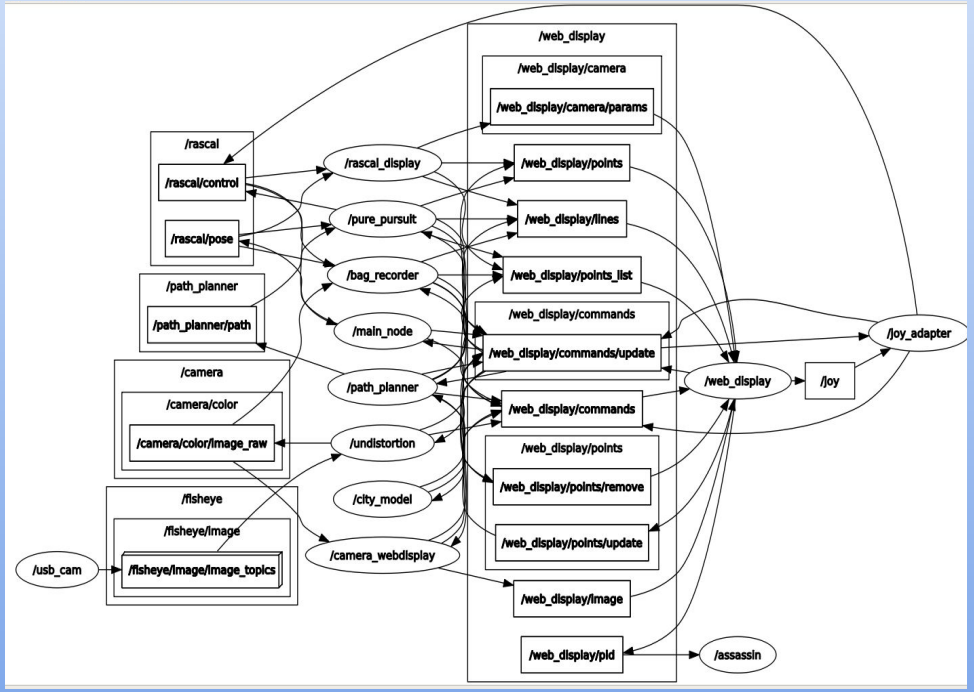


Robot operating system (ROS)

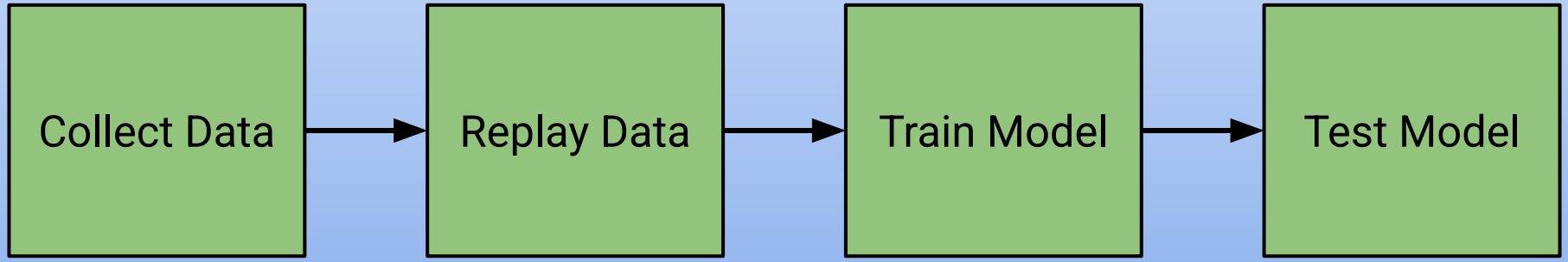
Allows for sending and receiving of data and information between files

Creates Packages with executable code called nodes

Nodes communicate through topics using services and messages



Training Process

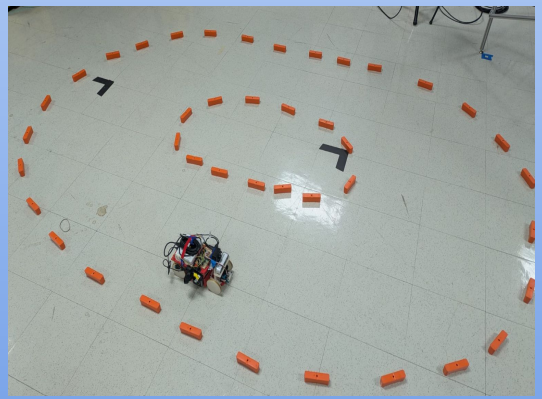


Collected data

Started out simple with orange bricks on a white tile floor

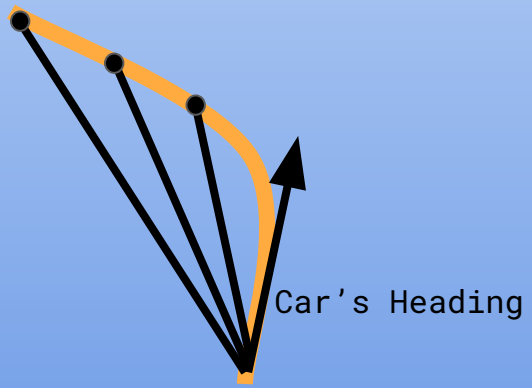
Then with right and left simple turns in the city

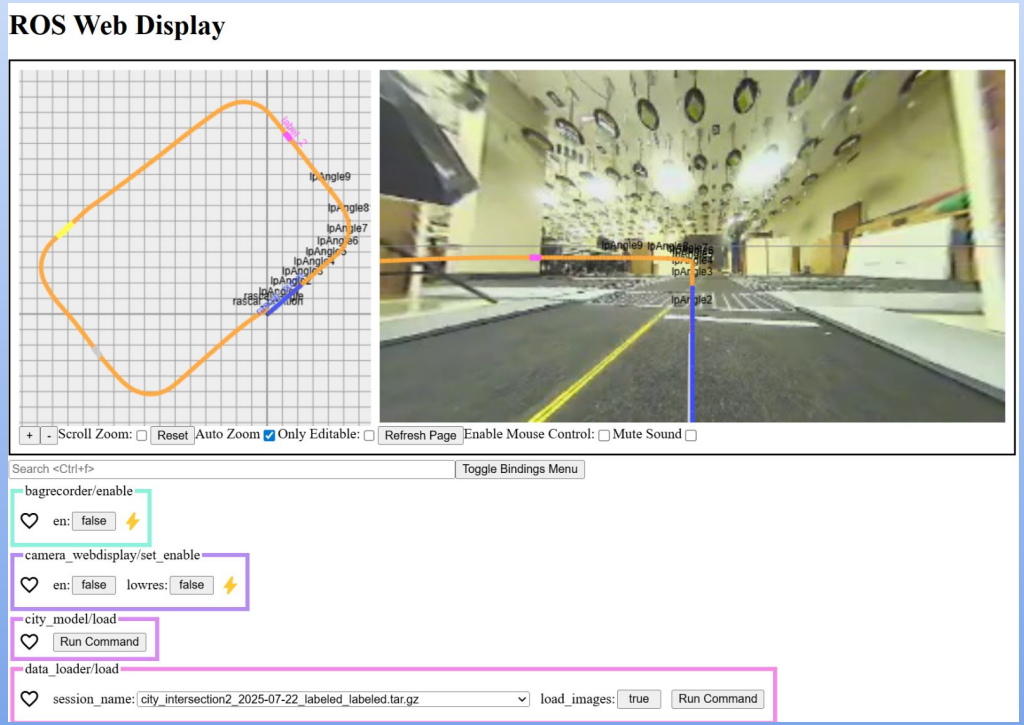
Moved on to the intersections which have more complex turns



Edit and label data

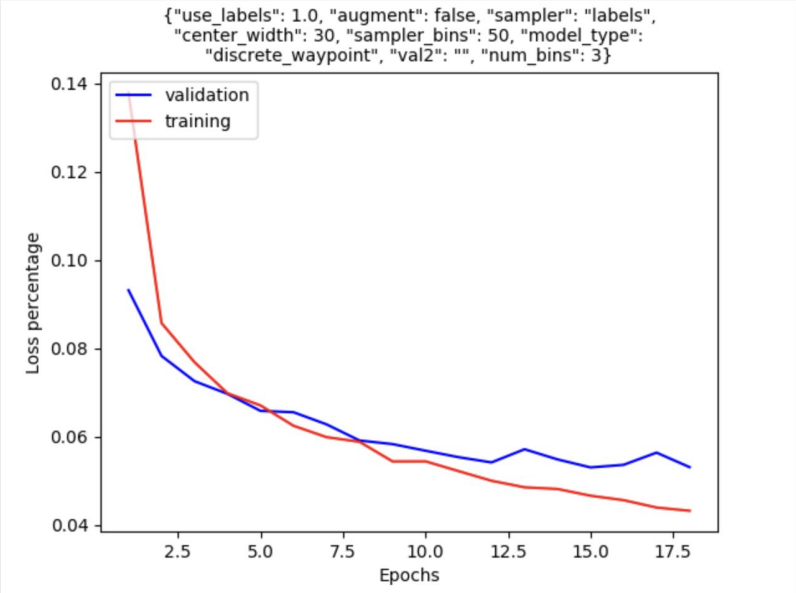
- Delete unwanted segments
- Label the intersections and turns
- Create lookahead angles for training data



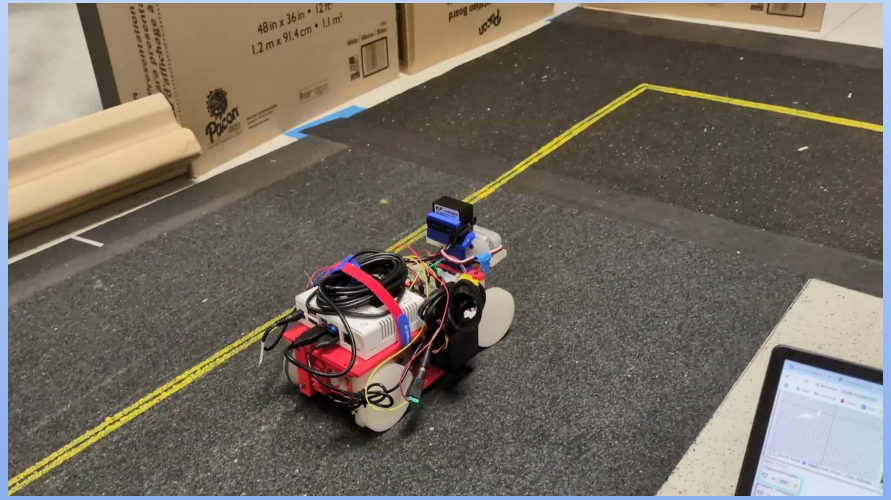


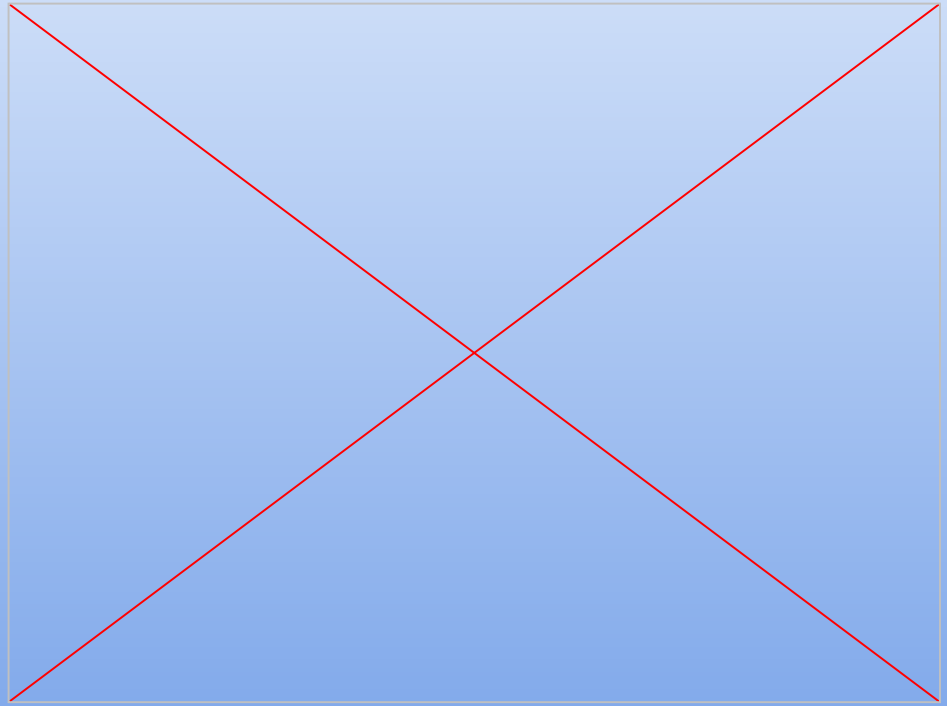
Train Model

- Teach the model the expected steering angles based on the image and labels

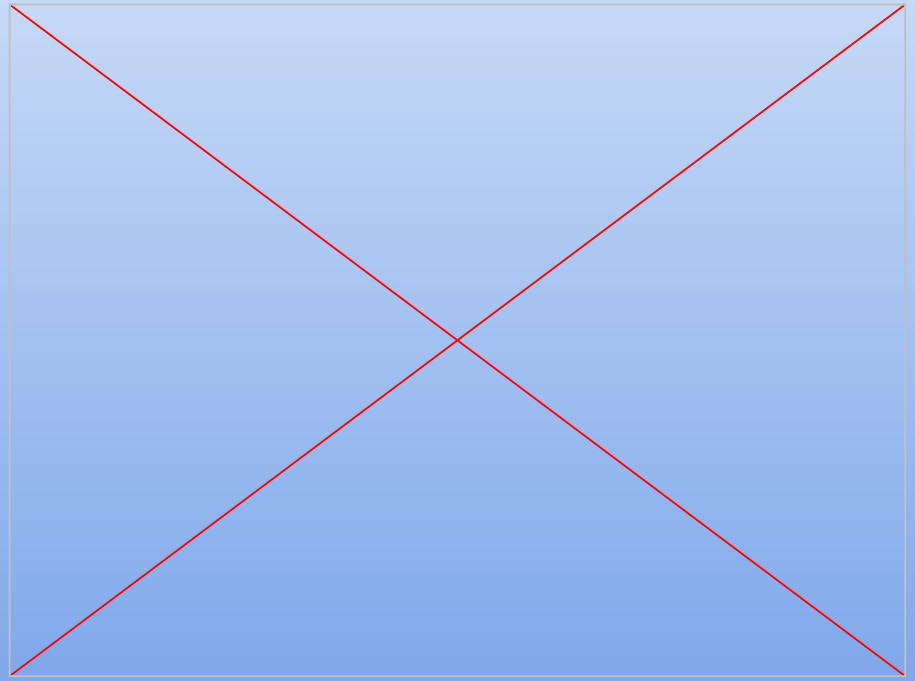


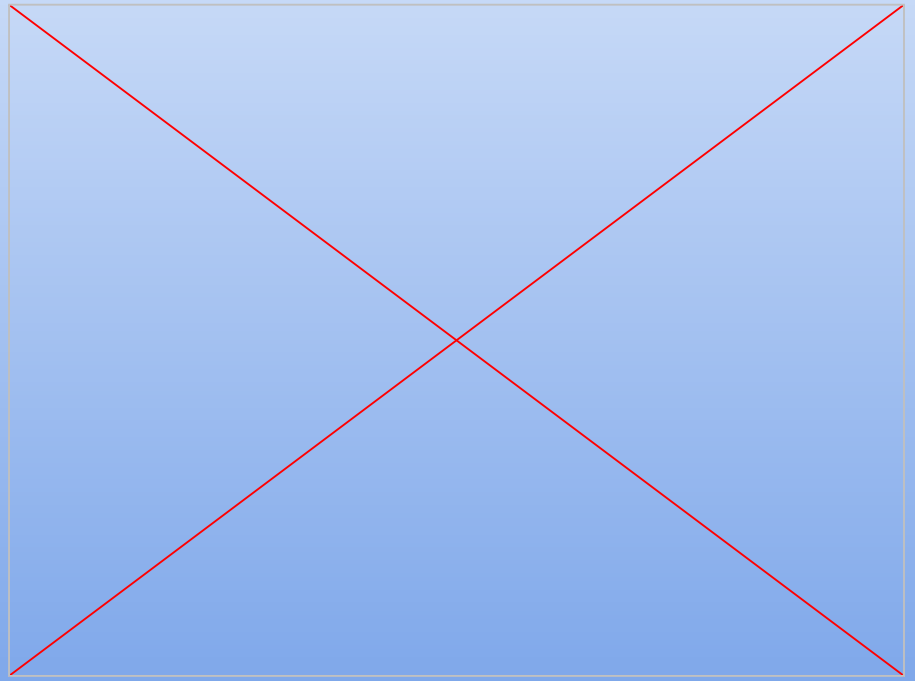
Fisheye





Discrete Waypoint



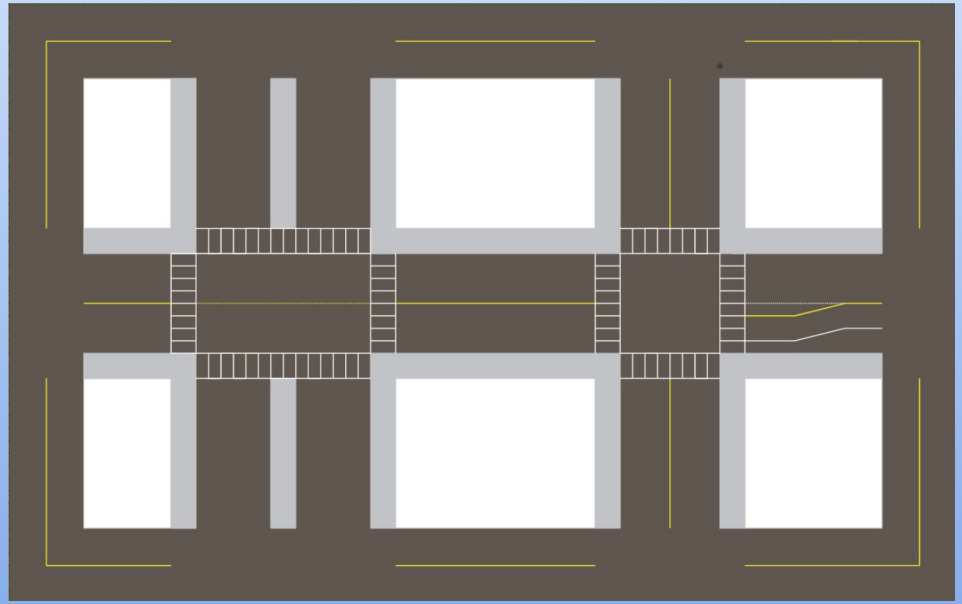


Created a map of the city

Using virtual grid paper (virtual-graph-paper.com)

6-inch grid resolution

ArUco Markers placed on Orbit nodes every 3 ft



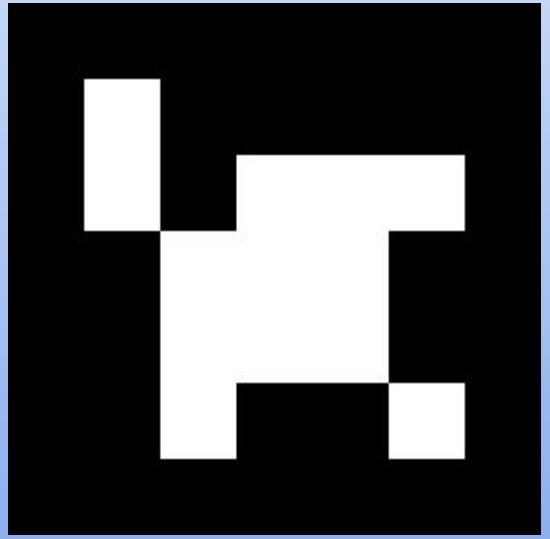
Set up ArUco Markers on each node

Generated ArUco Markers

Printed all the tags and stuck them to the nodes.

The car has a RealSense camera facing upwards to read the ArUco Markers

Created a Python script that can identify each Aruco Marker consistently and determine the camera’s position based on the tags



Refined Model and Environment

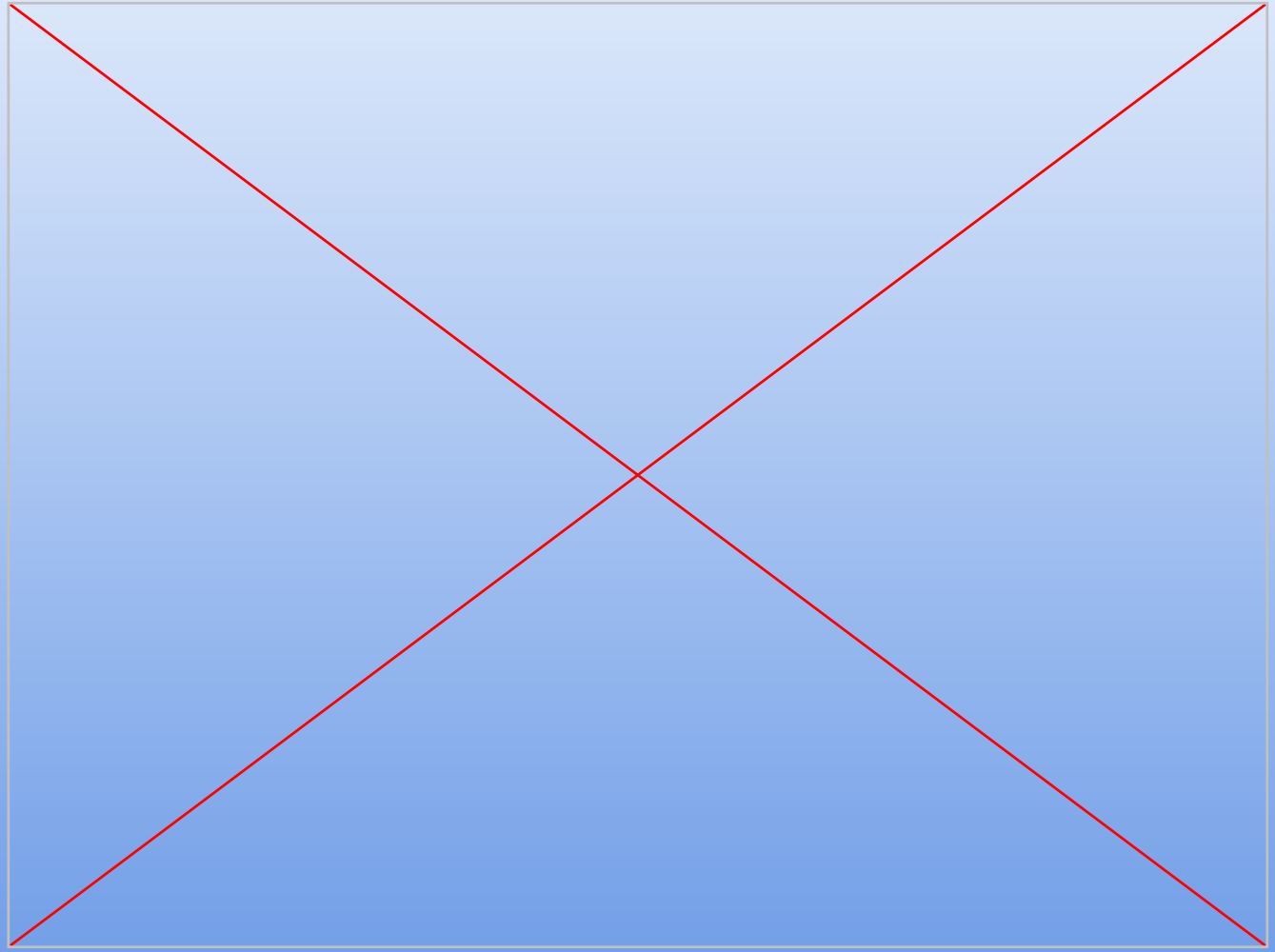
Replaced fisheye Camera

Relabeled data and updated documentation for future groups

Used additional inputs to train a more accurate model

Results

(time lapse)



Thank you!