RUTGERS
WINLAB | Wireless Information Network Laboratory

Self-Driving Vehicular Project 2024

RASCAL: Robotic Autonomous Scale Car for Adaptive Learning

Aaron Cruz[UG], Arya Shetty[UG], Brandon Cheng[UG], Tommy Chu[UG], Erik Nießen[HS], Siddharth Malhotra[HS]

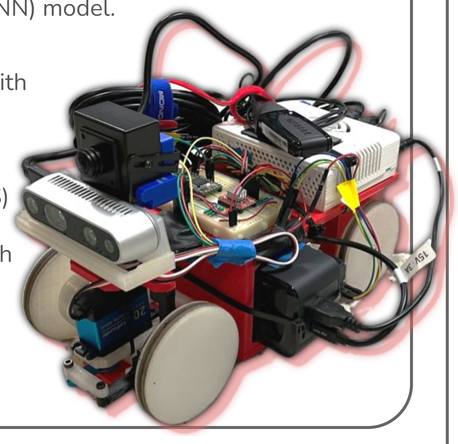
Overview

Self-driving cars typically use a myriad of sensors and cameras to achieve their goal. This project explores the sole use of front-facing cameras and their viability for autonomous driving.

Using a custom-built miniature car, we trained a self-driving neural network by driving throughout the smart city intersection to simulate how it would react to its environment.

Goal:
Assemble and document the creation and training of a 3D printed self-driving car that reacts to the city environment, using a Convolutional Neural Network (CNN) model.

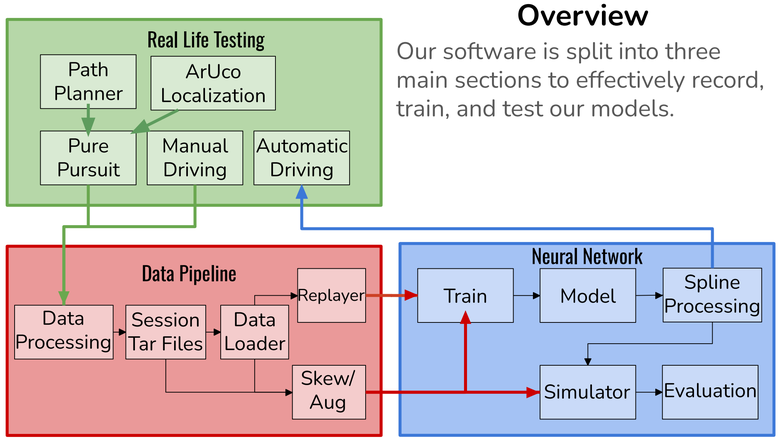
Methodology
- Combine 3D printed chassis with off-the-shelf electronics
- Develop a codebase of nodes and topics that run on the Robot Operating System (ROS)
- Train neural network on driving data using a server with hardware acceleration
- Evaluate trained model by connecting output of neural network to driving control in simulation and physical car



System Architecture

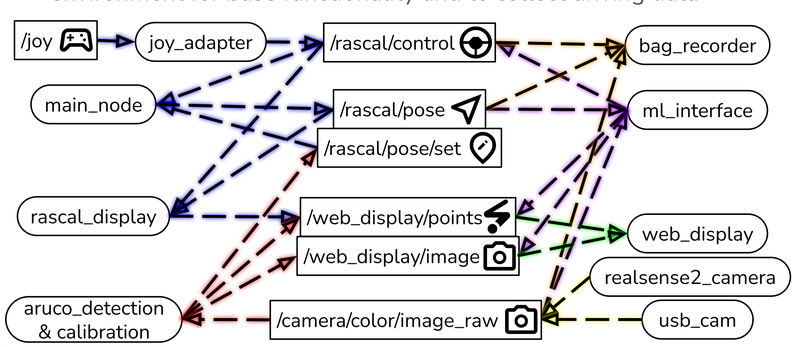
Overview

Our software is split into three main sections to effectively record, train, and test our models.



ROS Nodes and Topics

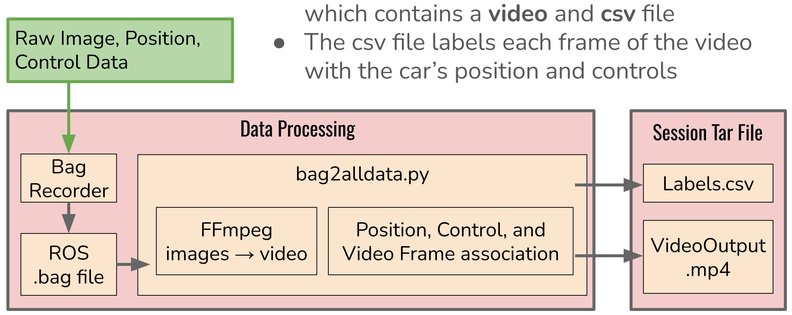
- Software runs on the car's computer, a Jetson Nano, in the ROS environment for base functionality and to collect driving data



Data Pipeline

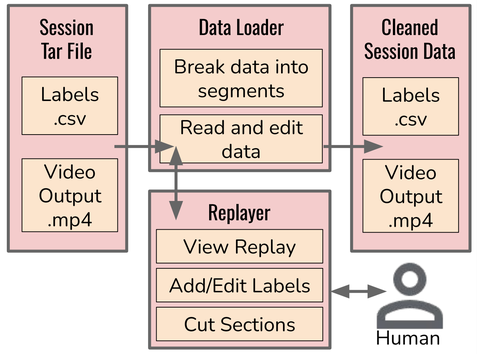
Data Processing

- Raw driving data is recorded into a "session", which contains a **video** and **csv** file
- The csv file labels each frame of the video with the car's position and controls



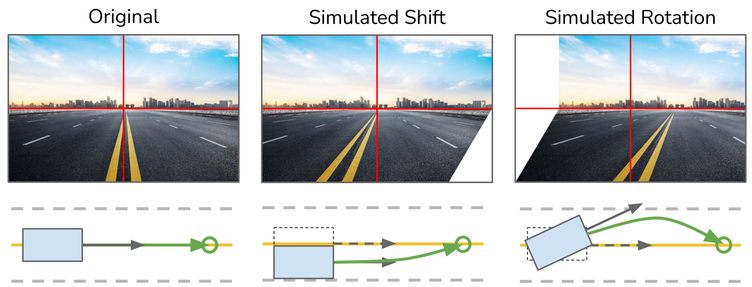
Replaying and Editing

- Compressed tar files are offloaded to a server
- The replayer allows us to **view and edit** the data
- Using this, we manually remove bad data so that the model does not learn from it



Skewing and Augmentation

- Image skewing is used to generate **alternate views** from the car in different positions and orientations (data augmentation)
- This allows for a more varied dataset, allowing the neural network to generalize for a wider range of scenarios

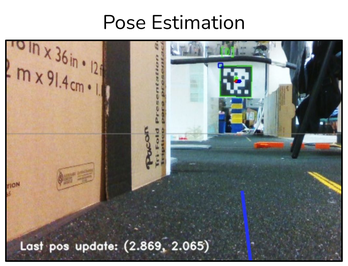


Camera Calibration

Localization

Motivation: As RASCAL drives, slipping and sensor imprecision leads to internal position inaccuracies.

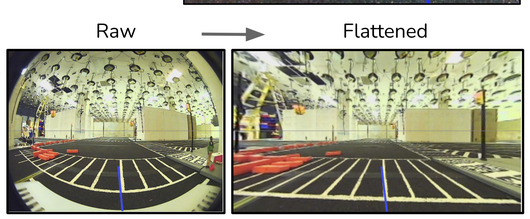
Solution: Detect **ArUco markers** at fixed locations within the city and use them to ground the car's position to the world.



Fisheye Correction

A fisheye camera allows for wider field of view.

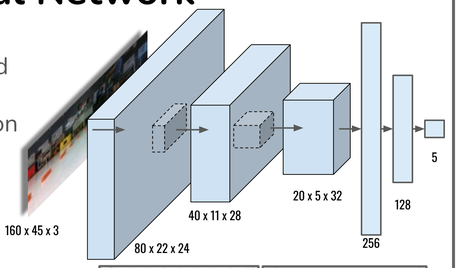
Calibration is needed to **flatten** the raw image.



Neural Network

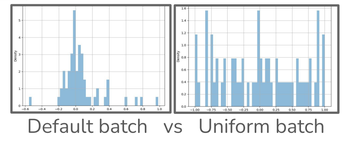
Network Architecture

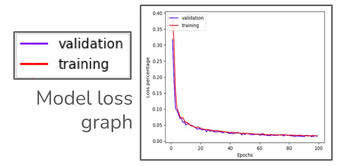
- 3 convolution layers followed by 3 fully connected layers
- Utilizes Leaky ReLU Activation function, Max Pooling, and Dropout between layers
- Outputs **4 angles** to future path points and **speed**

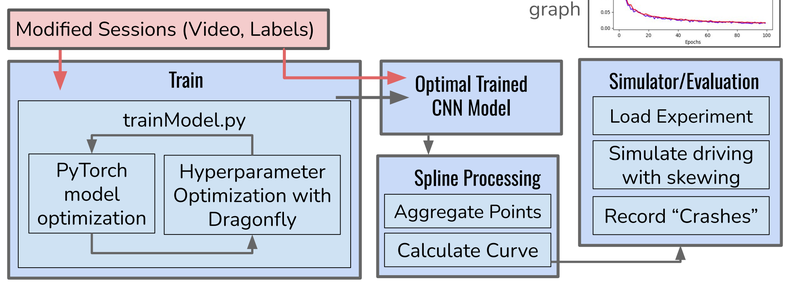


Network Training

- Uniform **batch sampling** prevents dataset bias for driving straight
- Hyperparameter optimization trains multiple models and picks the most **optimal** one
- After training, the model is run in the simulator to evaluate its performance



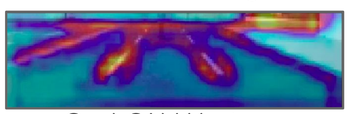




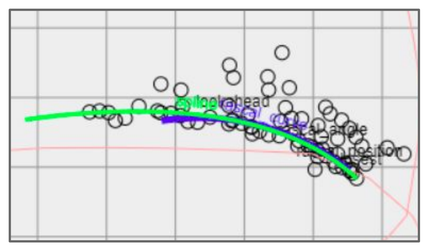
Output Processing

- To achieve a smooth path, we aggregate the output lookahead points of many images and fit a spline
- A Grad-CAM heatmap is overlaid onto the input image to understand what influences the model's decisions

Normalized Image



Grad-CAM Heatmap

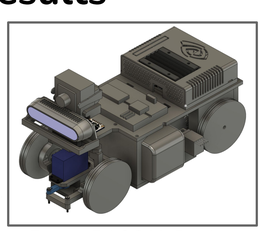


Spline Fit on Aggregated Points

Results

Conclusion

After many iterations of our model and hardware, we were able to precisely drive through the smart city, while obstacles were present, without collisions.



Future RASCAL Design

YOLO detection

Future Work

- Document process and assemble multiple RASCAL vehicles
- Integrate YOLO (You Only Look Once) object detection to control vehicle braking to react to traffic lights, road signs, and pedestrians
- Develop a state machine to declare intent while driving
- Utilize global position and ArUco integration for decision making

WINLAB