RC Car Project

Lane Keeping Control in Autonomous Driving by Computer Vision Approach

SEP 742 – Group 1

Instructor Info:

Siqi Zhao

Students Info:

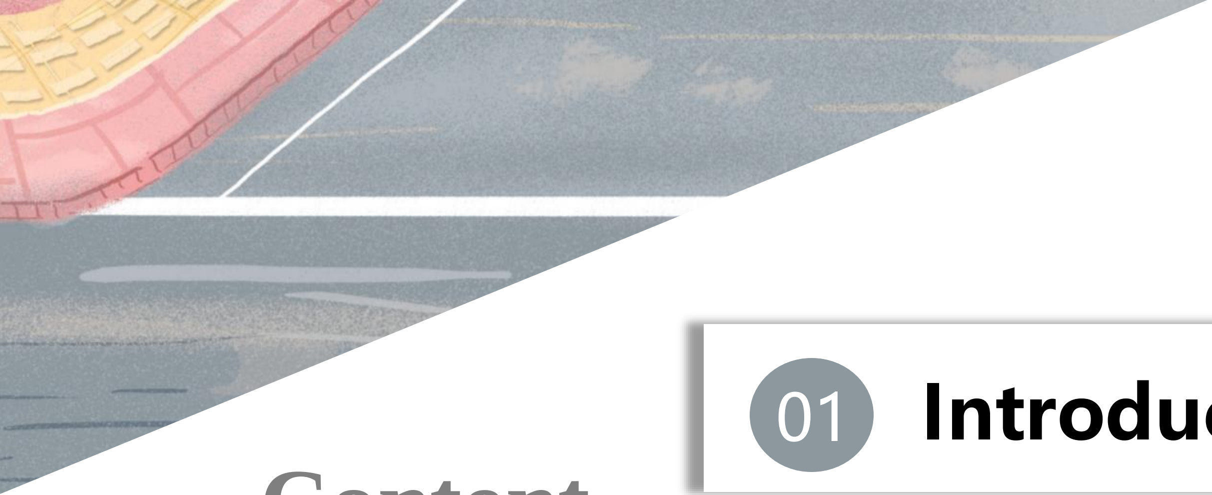
Jian Guan - guanj48@mcmaster.ca

Li Luo - luol35@mcmaster.ca

Ye Chen - chen63@mcmaster.ca

Yifei Zhou – zhouy487@mcmaster.ca

Zhengyang Cui - cuiz38@mcmaster.ca



Content

01 **Introduction**

02 Computer Vision

03 Hardware and Control

04 Summary and Future Work

Project Background

Why Lane-Keeping?

- Core task in autonomous driving
- Improves safety, comfort, and traffic flow
- Critical in highway scenarios

Our Approach

- Use a vision-based system on an RC car
- Focus on detecting lanes and adjusting steering

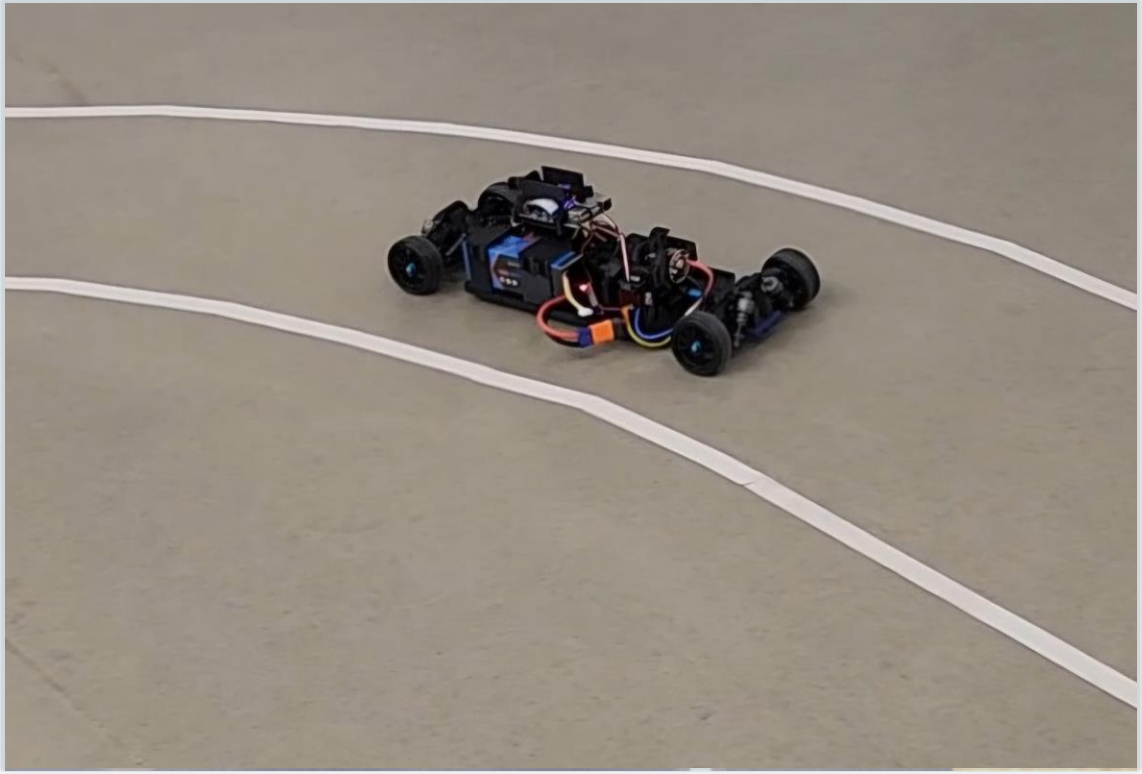
Problem Definition

◆ **Main Tasks**

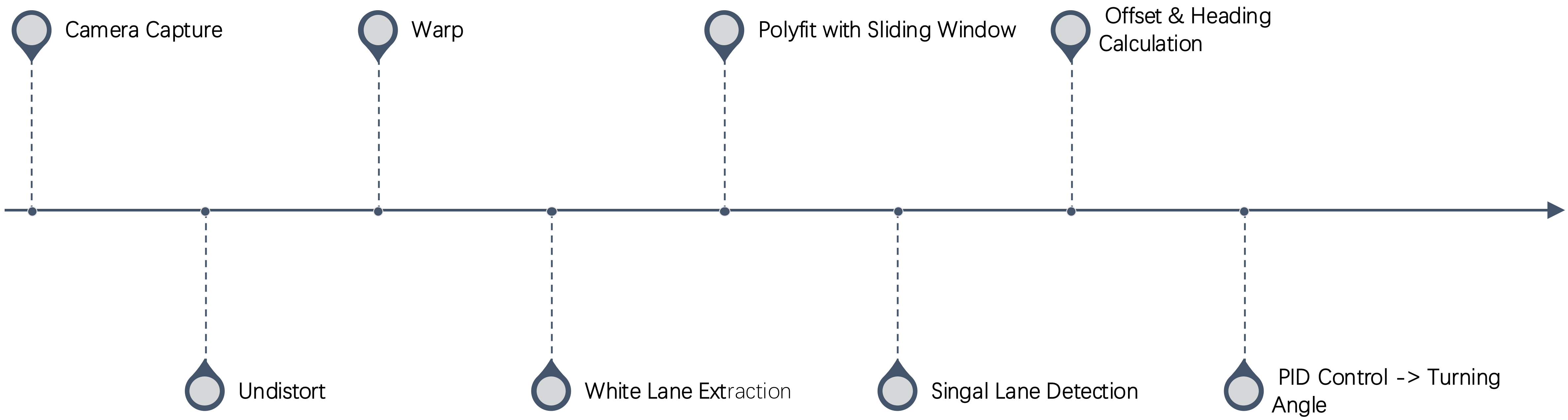
- Detect lane boundaries
- Estimate curvature & position
- Control steering in real-time

◆ **In our setup**

- Maintain constant forward speed
- Adapt to different curve geometries
- Focus on lateral control (not lane change or obstacle avoidance)



System Flowchart



01 Demo (video)

CONTENT

01 Introduction

02 Computer Vision

03 Hardware and Control

04 Summary and Future Work

02 Computer Vision

McMaster University

Scene Preparation

Computer vision-based lane detection systems require precise scene preparation to create a standardized view of the road environment.

Camera Calibration

Aerial View Transform

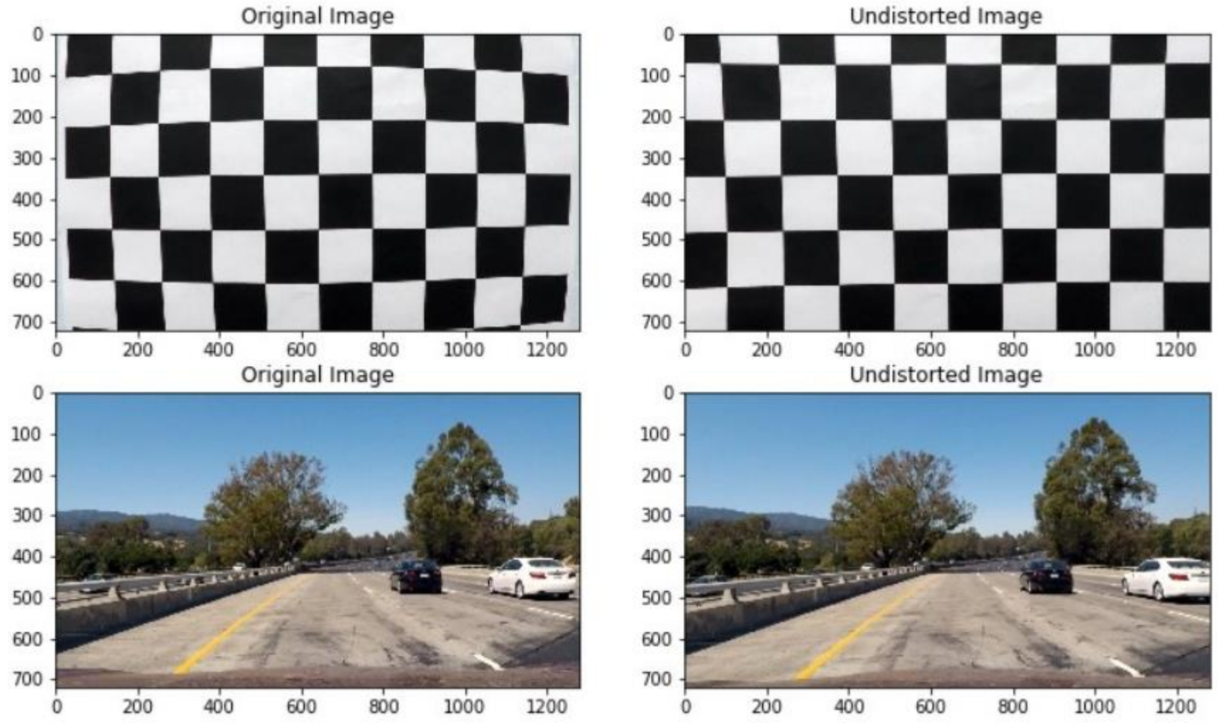
02 Computer Vision

McMaster University

Scene Preparation

Correct lens distortion by establishing the relationship between 3D world coordinates and 2D image coordinates through parameters such as length, point, and distortion coefficients.

Camera Calibration

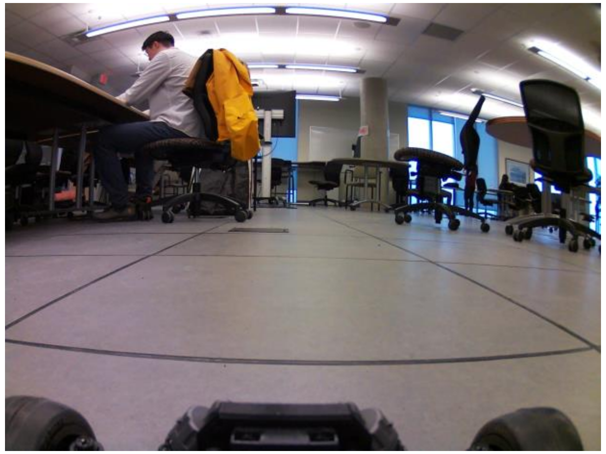


02 Computer Vision

Scene Preparation

Correct lens distortion by establishing the relationship between 3D world coordinates and 2D image coordinates through parameters such as length, point, and distortion coefficients.

Camera Calibration



Original Image

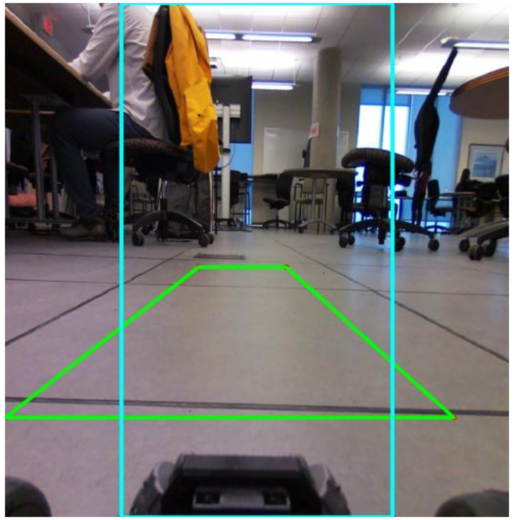
Undistorted

02 Computer Vision

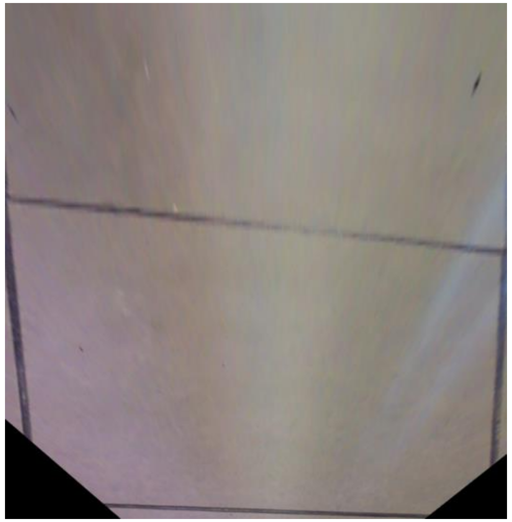
Scene Preparation

A perspective transformation is applied to obtain a bird's-eye view of the road, converting the frontal camera perspective into a top-down view where lane lines appear parallel rather than converging at the horizon.

Aerial View Transformed



a) Original Image



b) Aerial View Transformed

02 Computer Vision

Image Transformation - HSV

HSV converts the input BGR image to HSV space and applies specific thresholds to identify lane markings.

- Color Information (Hue)
- Color Saturation (Saturation)
- Brightness (Value)

(Abbas & Kadhim, 2024;Hillel et al., 2012)

With carefully tuned thresholds, we can detect both white and yellow lane markings on roads and have higher resistance on interferences compared with RGB image.

a) Original Image

b) Image Extracted White-Yellow in RGB

c) Image Applied White-Yellow HSV

02 Computer Vision

Edge Detection- Canny

Lane lines possess a distinct characteristic: they typically form long, continuous borders!

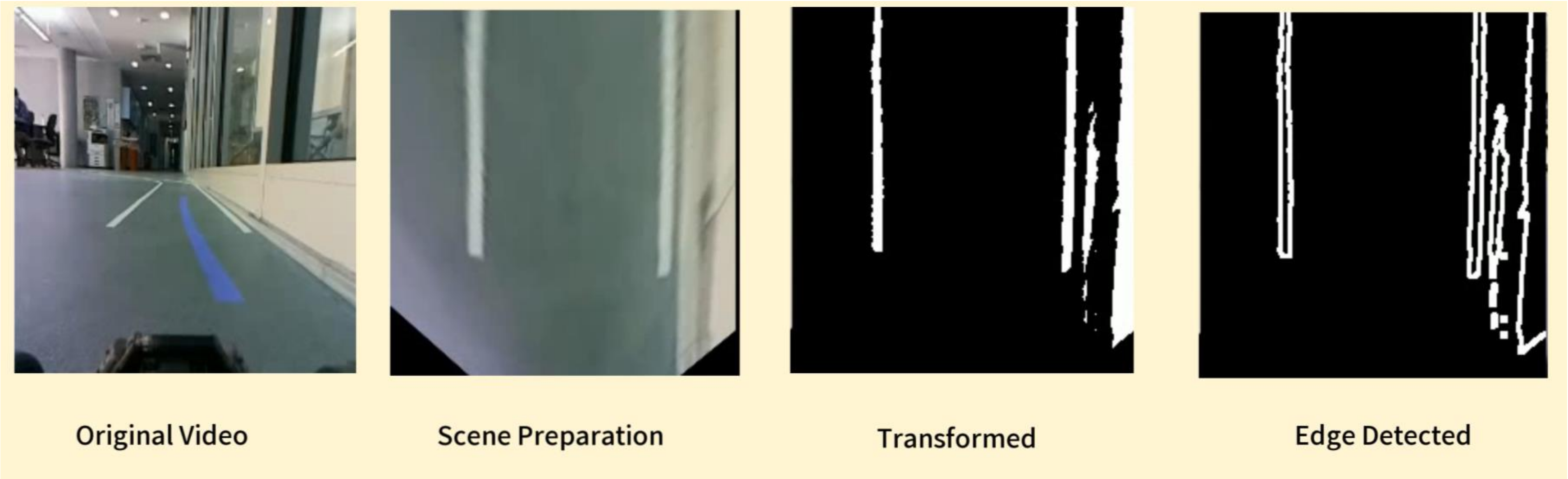
•**Noise Reduction**
Apply Gaussian filter to smooth image and reduce noise

•**Gradient Calculation**
Use Sobel operator to find intensity gradients

•**Double Thresholding on Classified Pixels**
- Strong: above high threshold
- Weak: between low & high
- Non-edge: below low threshold

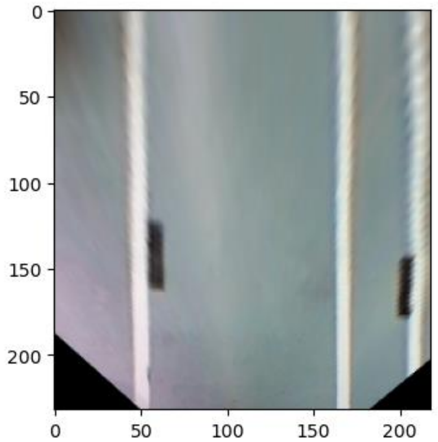
Link weak edges to strong ones if connected.

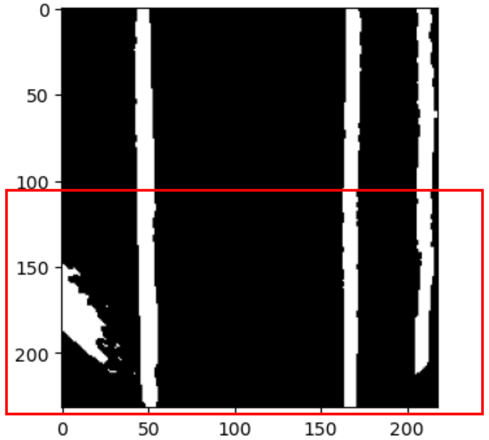
(Malche, 2024; Educative, 2025; Scikit, 2013)



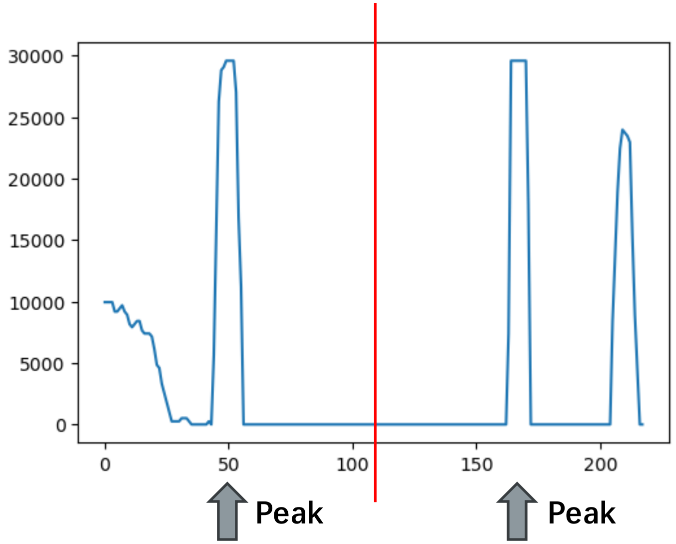
02 Computer Vision

Lane Detection using Sliding Window



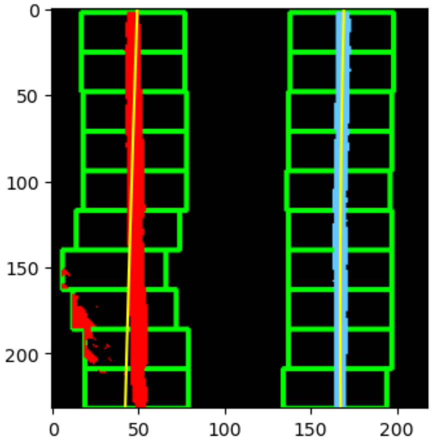


HSV Threshold Filtered



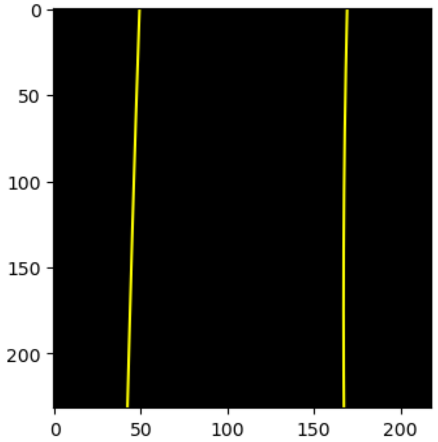
Initial Position Determination

- Build histogram from bottom-half of the image
- Split left and right by mid point
- Select peak points as initial left lane and right lane starting location



Sliding Windows

- Building blocks vertically
- Whole image split into *n* windows
- **Filtering** points – **calculate** average x – **reposition** next window



Polynomial Fitting

- 2-degree polynomial
- y=ax^2+bx+c
- Next sliding window location base on fitting

02 Computer Vision

Lane Geometry – Offset, Heading

Offset

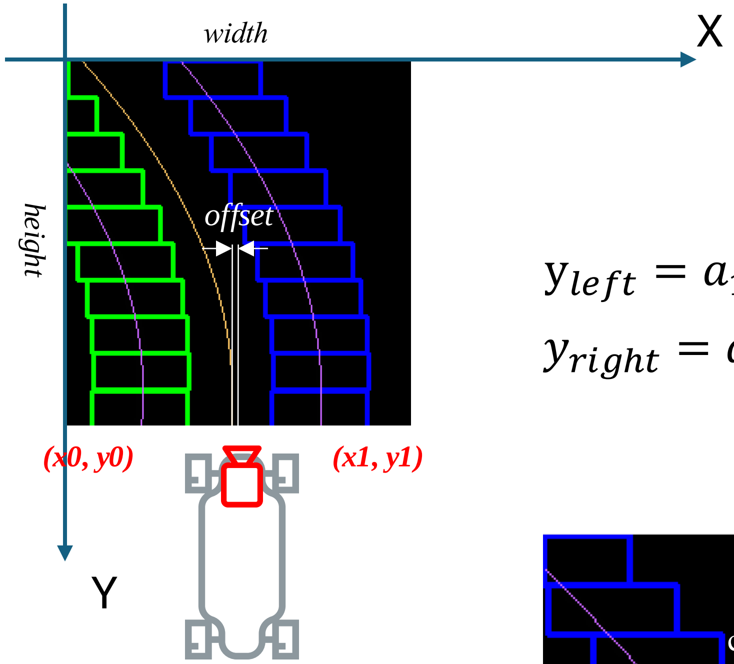
When y=*height*, solve y_{left} ➡ *(x0, y0)*

When y=*height*, solve y_{right} ➡ *(x1, y1)*

y0=y1=*height*

offset = $\frac{width}{2} - \frac{x_0+x_1}{2}$

The camera always point at the mid point of the image!



$$y_{left} = a_1x^2 + b_1x + c_1$$

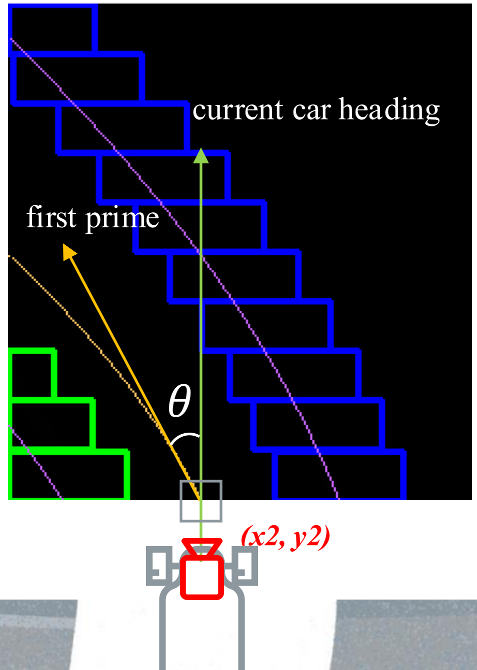
$$y_{right} = a_2x^2 + b_2x + c_2$$

Heading

$$y'(x_n) = (a_1 + a_2)x_n + \frac{b_1+b_2}{2}, \text{ at given point } x_n$$

$$\theta = \arctan(a_1 + a_2)$$

$$y_{middle} = \frac{a_1 + a_2}{2}x^2 + \frac{b_1 + b_2}{2}x + \frac{c_1 + c_2}{2}$$



McMaster University

CONTENT

01 Introduction

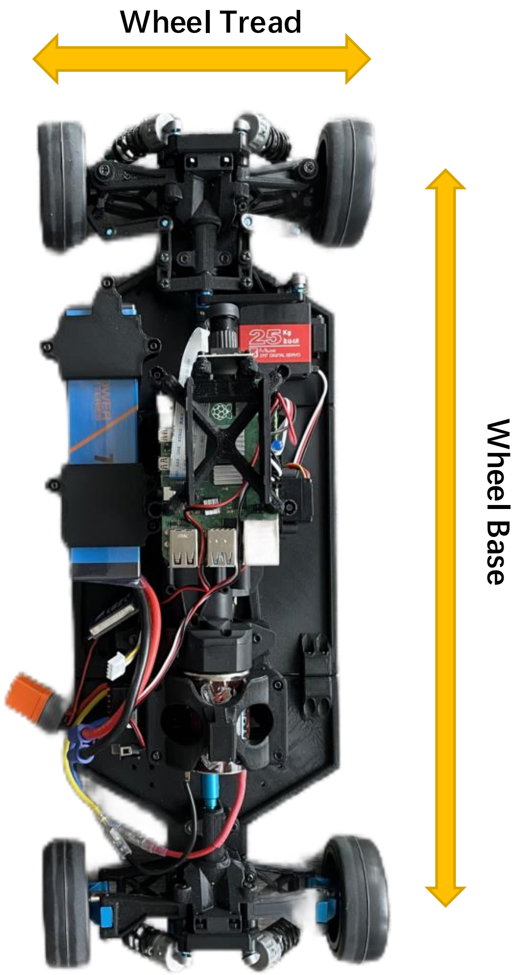
02 Computer Vision

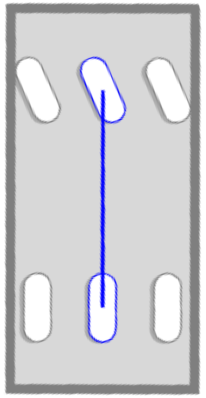
03 Hardware and Control

04 Summary and Future Work

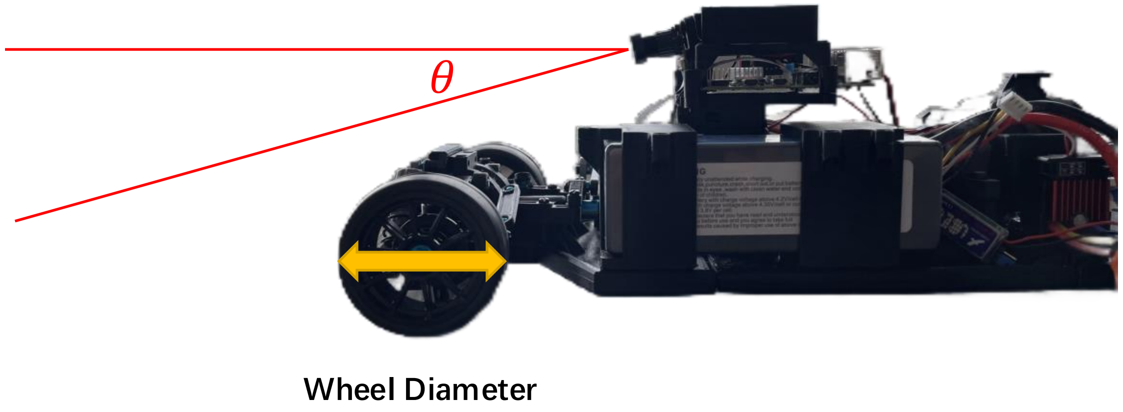
03 Hardware and Control

RC Car Hardware





Two-Wheel Bicycle Model

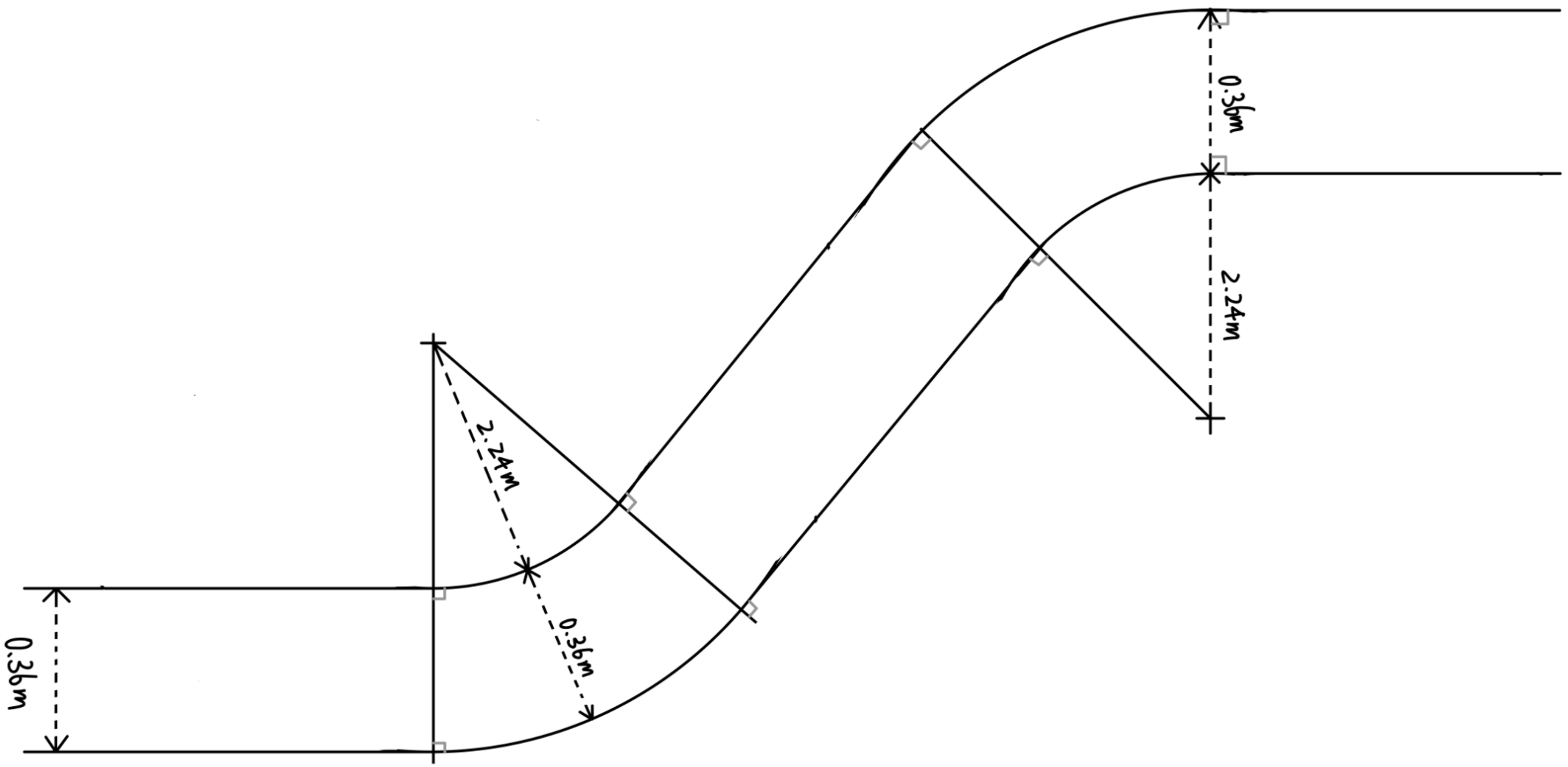


Specification	Value
Wheel Base	0.4 m
Minimum Turning Radius	~2.8 m
Maximum Steering Angle	~8°
Camera Incline Angle	10°
Camera Resolution	Up to 2592x1944

03 Hardware and Control

Track Design

1. Straight Line
2. Left Turn
3. Straight Line
4. Right Turn
5. Straight Line



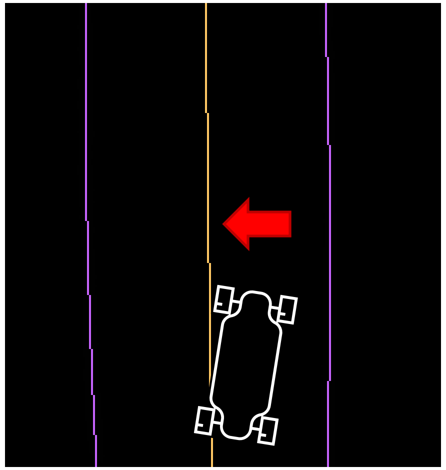
03 Hardware and Control

PID Control

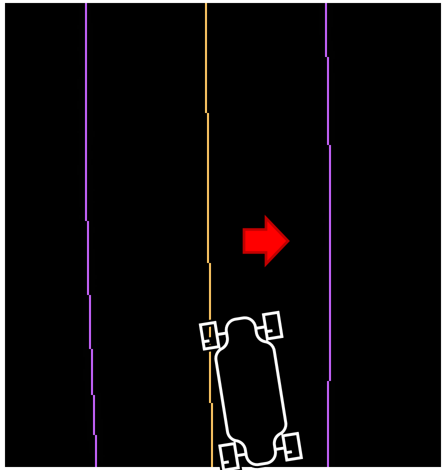
$\boxed{offset \to 0, theta \to 0}$

$$Observation = k * Offset\,(1 - k) * Heading \qquad \theta = tanh(\lambda * Gain) * \omega_{max}$$

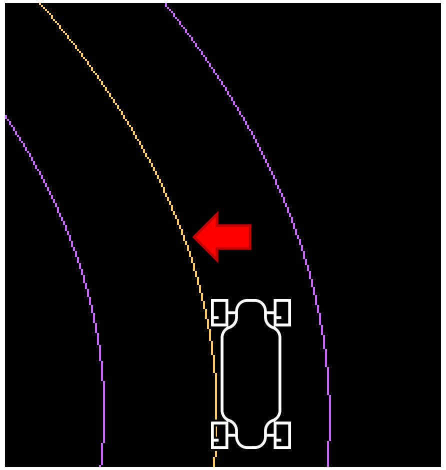
$$PID\ Gain = PID(SetPoint, Observation)$$



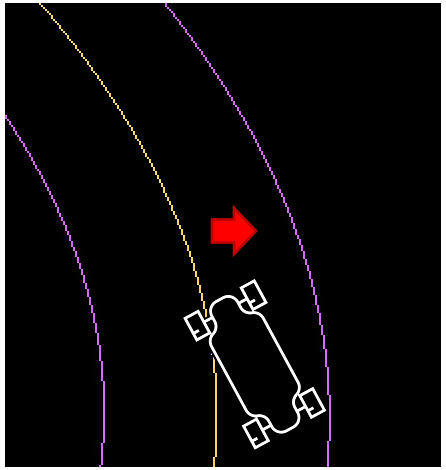
offset > 0, theta > 0



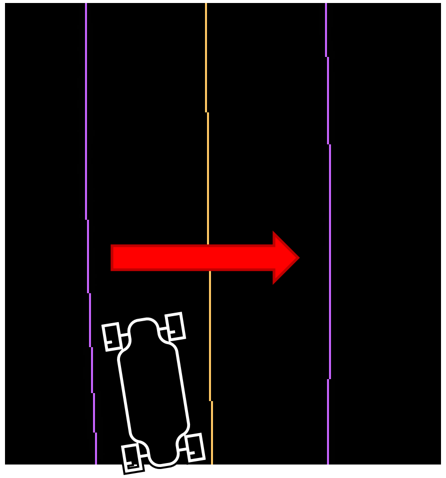
offset > 0, theta < 0



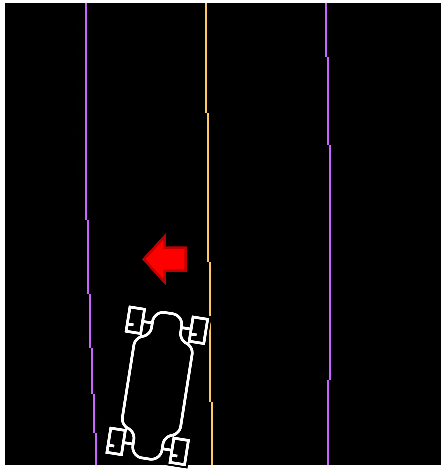
offset > 0, theta > 0



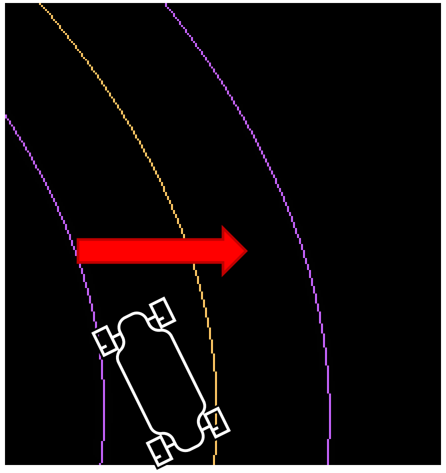
offset > 0, theta < 0



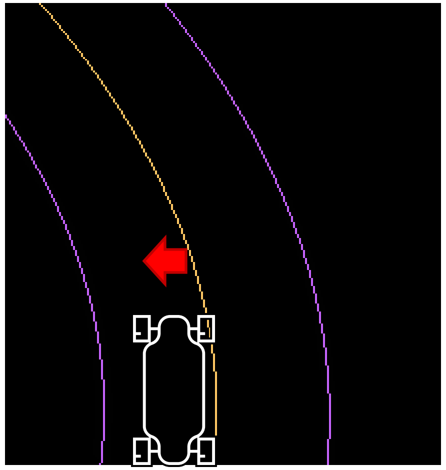
offset < 0, theta < 0



offset < 0, theta > 0



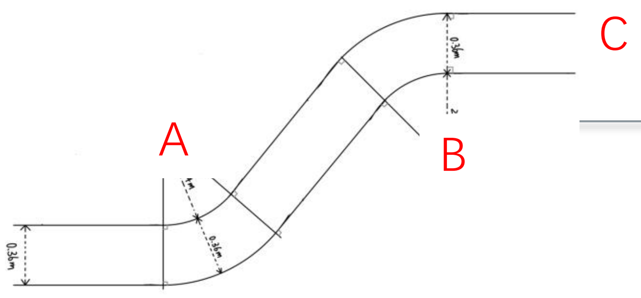
offset < 0, theta < 0

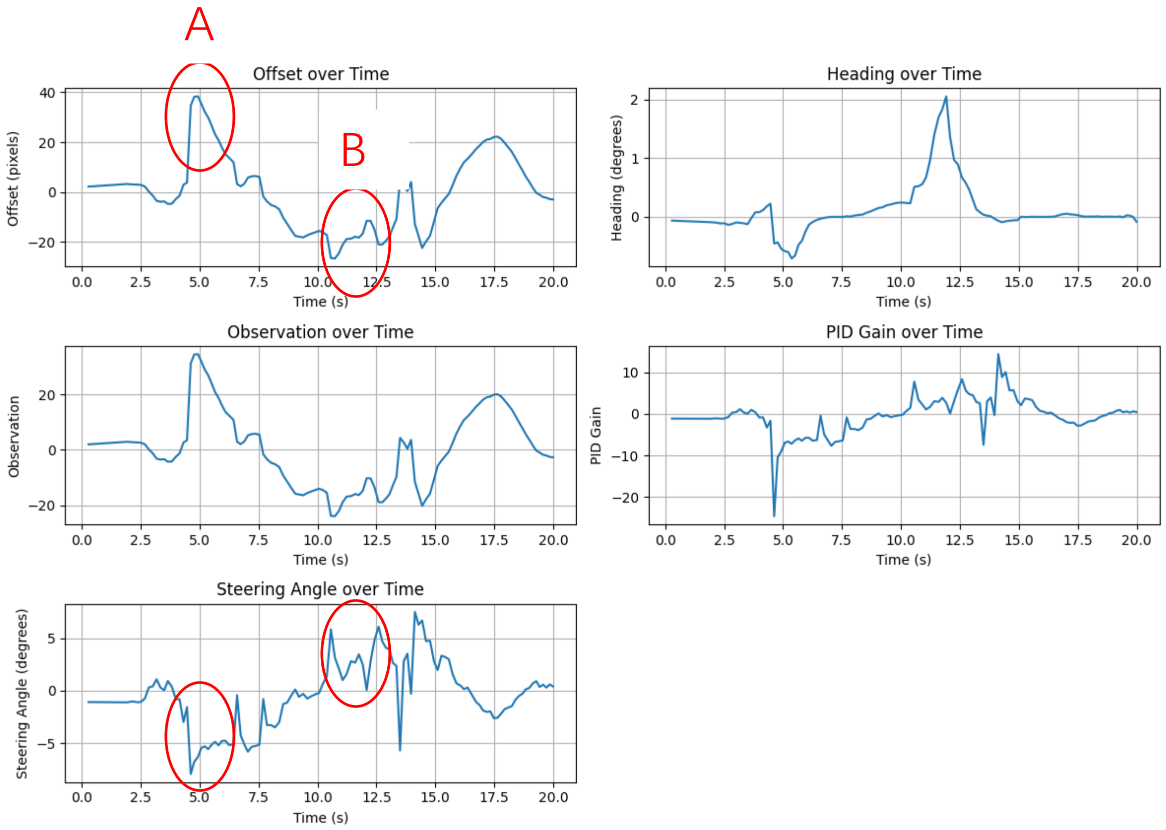


offset < 0, theta > 0

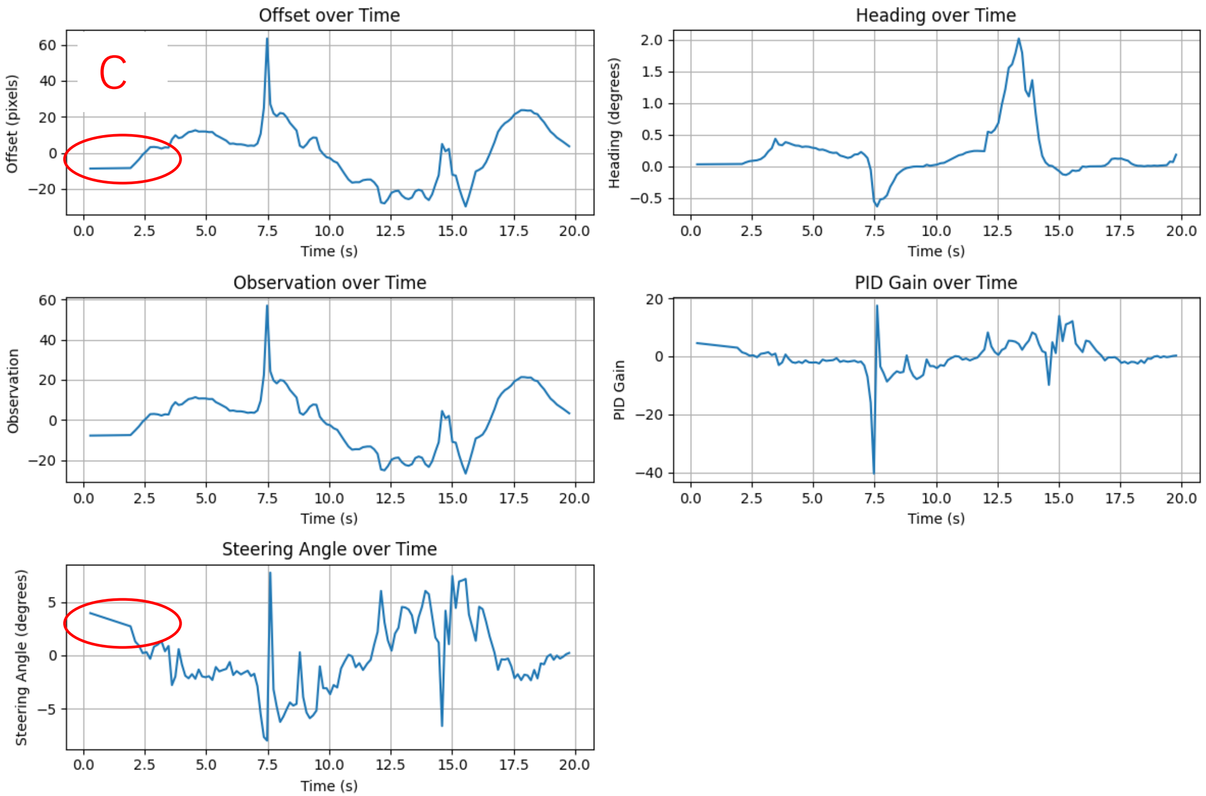
03 Hardware and Control

System Control Analysis

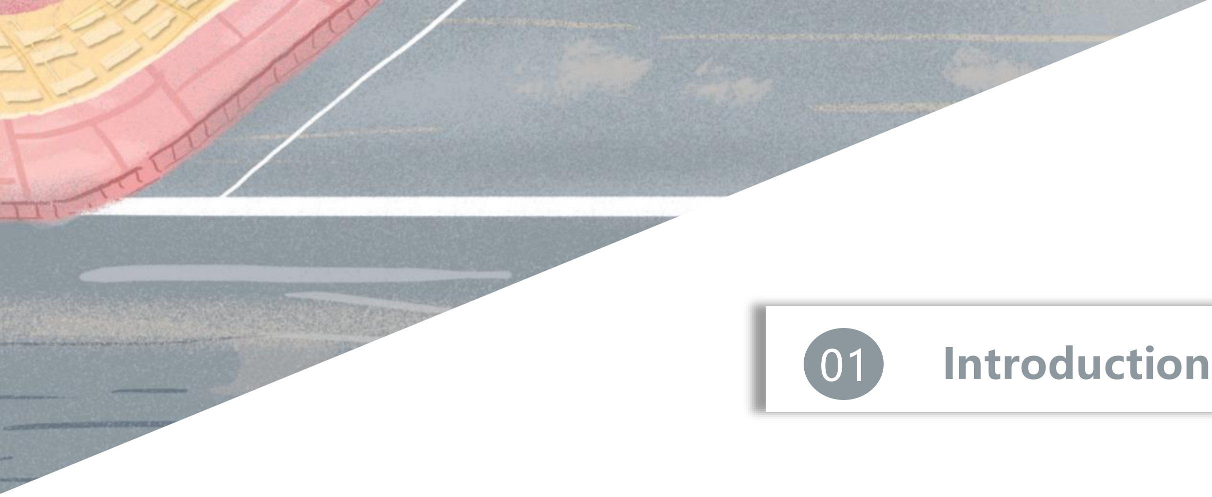




Forward Pass



Backward Pass



McMaster University

CONTENT

01 Introduction

02 Computer Vision

03 Hardware and Control

04 Summary and Future Work

04 Summary

- **Overview**
  - Successfully ran the entire pipeline
  - Very fast, achieving 10 fps, allowing steering adjustments 10 times per second
  - Strong results from PID parameter tuning
  - Accelerated parameter tuning driven by data plots
- **Hardware Limitations**
  - Camera: Narrow FOV (Field of View) causes loss of one lane during turns
  - Vehicle does not go perfectly straight forward, though PID control compensates
- **Software Pipeline**
  - HSV thresholding is not robust; strong illumination (e.g., sunlight or reflections) causes poor filtering
  - Hard-coded corner cases lack flexibility

https://github.com/YeChen-coder/RCCarRelated

04 Future Exploration

Motivation for CNN-Based Steering

Limitations of Traditional Computer Vision

Manual Feature Engineering:
Reliance on edge detection, thresholding, and geometric heuristics.

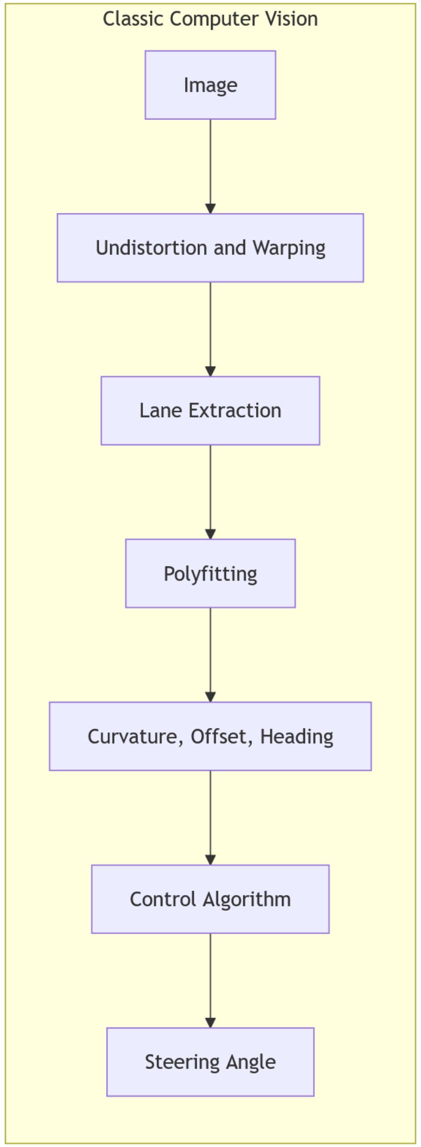
Environmental Sensitivity:
Fails under dynamic lighting (shadows/glare) and structural ambiguities (faded/blocked lane markings).

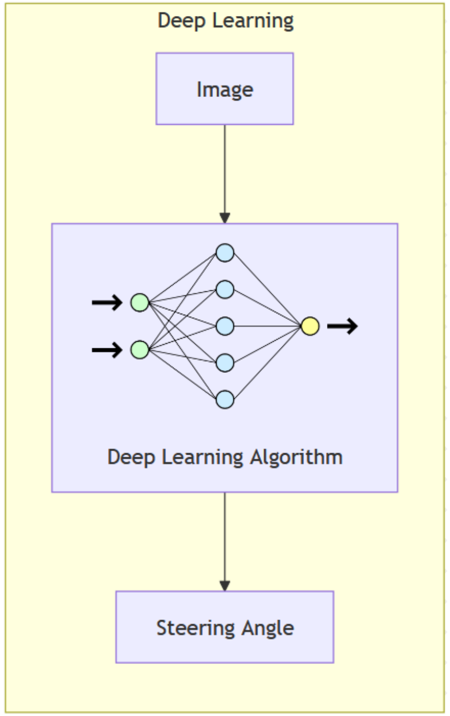
Scalability Challenges:
Requires frequent recalibration (HSV thresholds) for new environments.

Advantages of Deep Learning Solutions

End-to-End Learning:
Direct pixel-to-steering mapping without manual feature engineering.

Robustness:
Learns latent patterns from numerous examples.





04 Future Exploration

Software-based Experiment Setup

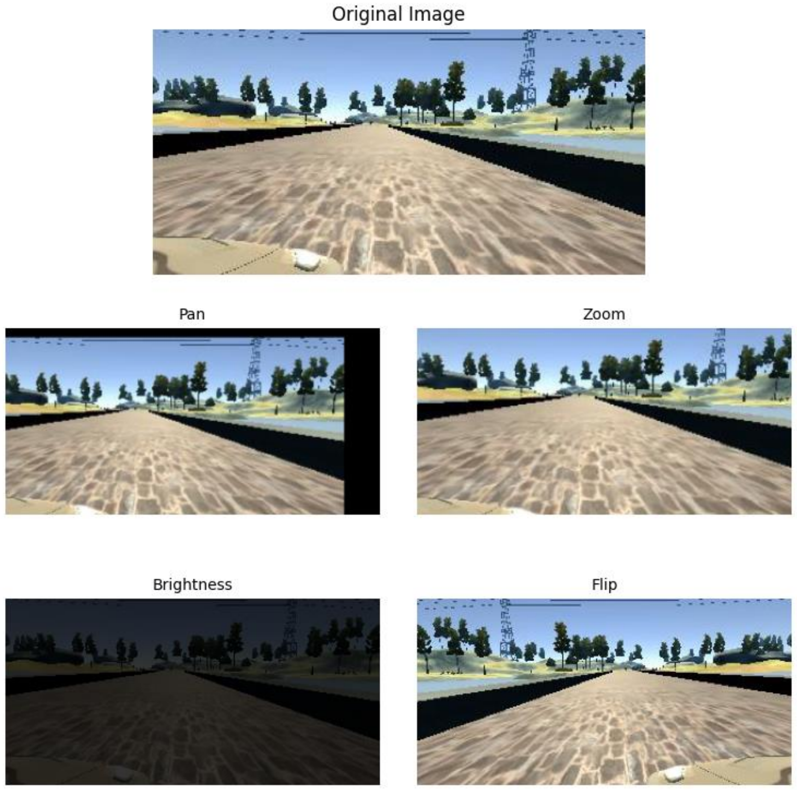
Simulation Environment
Udacity's Self Driving Car Simulator

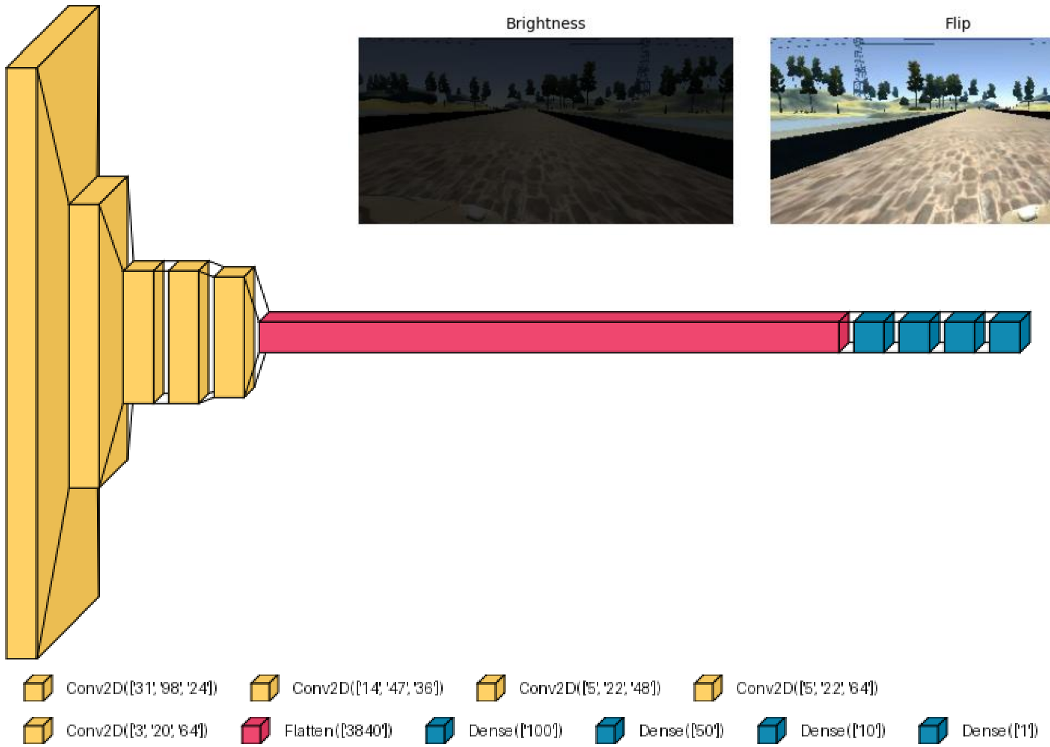
Dataset
https://github.com/rslim087a/track/
Collected from 3 cameras (center, left, right)
Includes steering angles, throttle, reverse, and speed

Data Processing
Bin Balancing
Random Affine Transformation
Side Cams with steering angle correction (±0.15)

Model
NVIDIA PilotNet
5 convolutional layers (ELU activation, dropout)
4 fully connected layers (ELU activation)

Evaluation
Validation Loss (MSE)
Human Observation (lane adherence, recovery from deviation)





04 Future Exploration

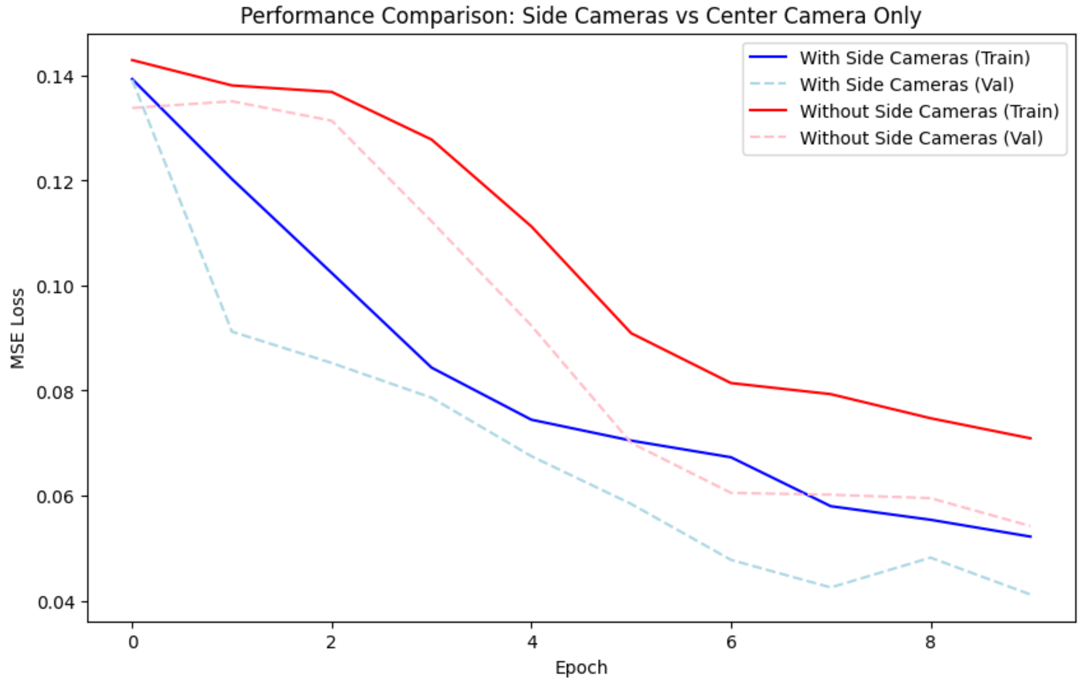
https://github.com/tzway/e2e-selfdrive

Key Challenge in Porting CNN Based Steering to RC Car

Missing Side Cameras
RC car has only a front-facing camera.
No data regarding lateral displacement.

Comparative Experiment: 3-Camera vs. Center-Camera Training
Replicated center cam images as side cam images for center-cam only data to ensure data balance
Trained with same arch (batch=100, lr=1e-3, epochs=10)
Used the same validation set for comparison

Val Loss: 3-Cam (0.041) vs. Center-Cam (0.054)
Observation: 3-Cam recovered more drastically from drifting while the Center-Cam in some circumstances failed to keep in the lane



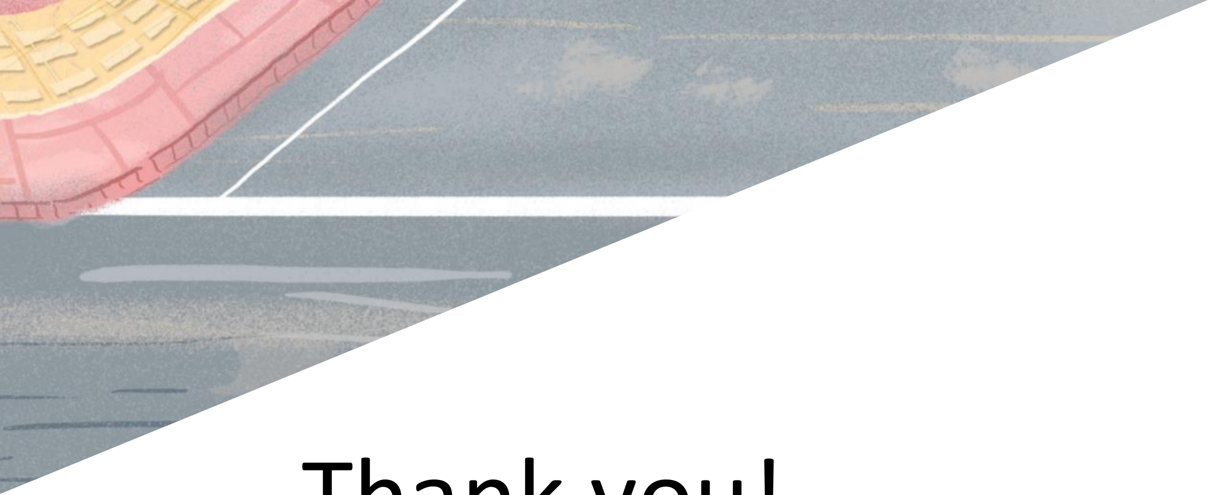
Conclusion and Porting Suggestions
Prioritize sensor diversity, add more cameras, add sensors like IMU to gain spatial information
Adjust the CNN layout based on specific tasks/environments (input image dimensions, output dimensions, depths)
Pair CNN predictions with rule-based corrections for fail-safe maneuvers

04 Future Exploration

RC Car Platform Exploration

Robot Operating System (ROS)

https://www.diyrobocars.com/



Thank you!
Feel free to reach out!

SEP 742 – Group 1

Students Info:

Jian Guan - guanj48@mcmaster.ca

Li Luo - luol35@mcmaster.ca

Ye Chen - chen63@mcmaster.ca

Yifei Zhou – zhouy487@mcmaster.ca

Zhengyang Cui - cuiz38@mcmaster.ca

References

Abbas, A. H., & Kadhim, H. J. (2024, February 27). *Detect Lane Line for Self-Driving Car Using Hue Saturation Lightness and Hue Saturation Value Color Transformation*. International Journal of Online and Biomedical Engineering (ijoe). https://online-journals.org/index.php/i-joe

Hillel, A. B., Lerner, R., Levi, D., & Raz, G. (2012, February 7). *Recent progress in road and Lane Detection: A survey - machine vision and applications*. SpringerLink. https://link.springer.com/article/10.1007/s00138-011-0404-2

Malche, T. (2024, October 31). *Edge detection in image processing: An introduction*. Roboflow Blog. https://blog.roboflow.com/edge-detection/

Educative. (2025). *What is canny edge detection?* https://www.educative.io/answers/what-is-canny-edge-detection

Scikit. (2013). *Canny edge detector*. Canny edge detector - skimage 0.25.2 documentation. https://scikit-image.org/docs/0.25.x/auto_examples/edges/plot_canny.html

Ryan Keenan, Brok, Reichelt, M., & Wong, C. (2017). *CHARLESWONGZX/Advanced-lane-lines: Finding lane lines using sliding window search.* GitHub. https://github.com/charleswongzx/Advanced-Lane-Lines

References

Devane, V., Sahane, G., Khairmode, H., & Datkhile, G. (2021, August 9). *Lane detection techniques using image processing*. ITM Web of Conferences. https://www.itm-conferences.org/articles/itmconf/abs/2021/05/itmconf_icacc2021_03011/itmconf_icacc2021_03011.html

Kuo, C. Y., Lu, Y. R., & Yang, S. M. (2019, April 8). *On the image sensor processing for Lane Detection and control in Vehicle Lane Keeping Systems*. Sensors (Basel, Switzerland). https://pmc.ncbi.nlm.nih.gov/articles/PMC6479783/

Udacity. (2017). *Self-driving car simulator*. GitHub repository. https://github.com/udacity/self-driving-car-sim

Udacity. (2017). *CarND-Behavioral-Cloning-P3.* GitHub repository. https://github.com/udacity/CarND-Behavioral-Cloning-P3