Autonomous Driving Technology Project

Autonomous Navigation on Road

Think safety, think innovation

Raghavendra Rao

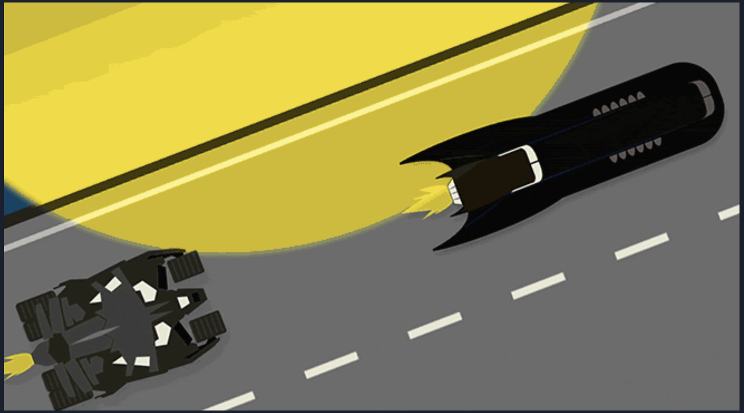
Table Of Content

Overview

"An autonomous car (also known as a driverless car, self-driving car, robotic car) and unmanned ground vehicle is a vehicle that is capable of sensing its environment and navigating without human input."

The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task

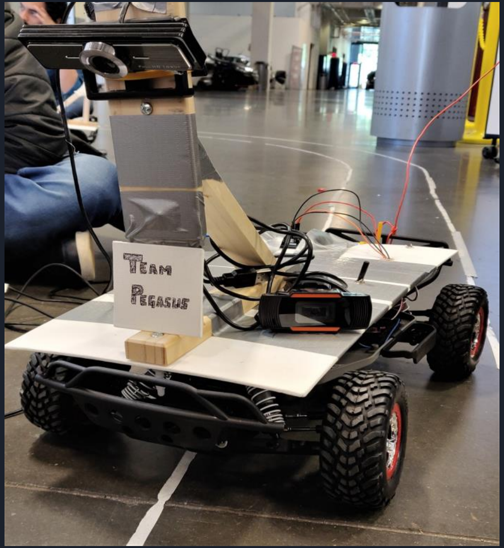
- Input data from visual information
- GPS sensing knowledge to help with navigation
- Sensors and other equipment to avoid collisions
- augmented reality
- SLAM, RTLS, LIDAR, GPS, IMU, Stereo vision
- Deep learning



Project objective

Use Matlab and deep learning to build an autonomously controlled vehicle performing a set of tasks

Understand the functional and conceptual overview

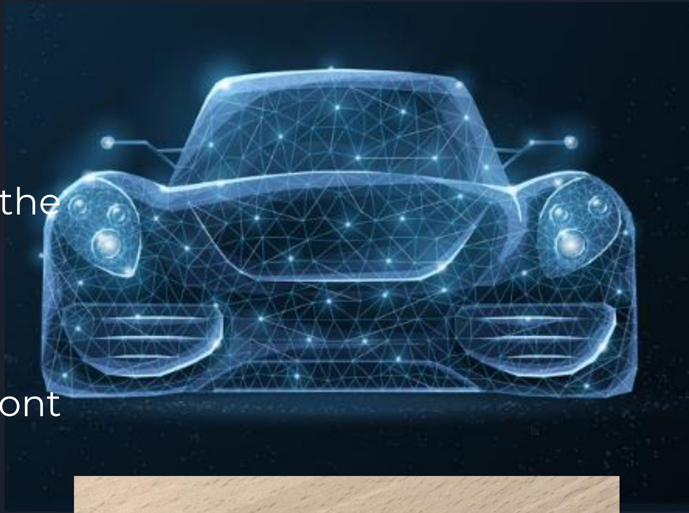


Understanding the problem

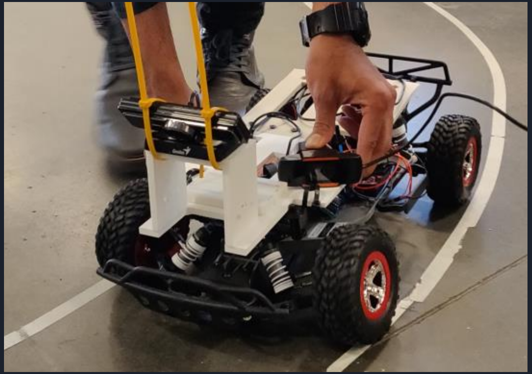
- Autonomously track the lane clockwise using the big camera and one laptop
- Autonomously recognize a stop sign and a school zone sign using the small camera and a second computer
- Send the road sign information from the second computer to the first computer through Wi-Fi communications
- Stop the vehicle at stop sign for 2 seconds and then run at a high speed; Reduce the speed by half at school zone sign

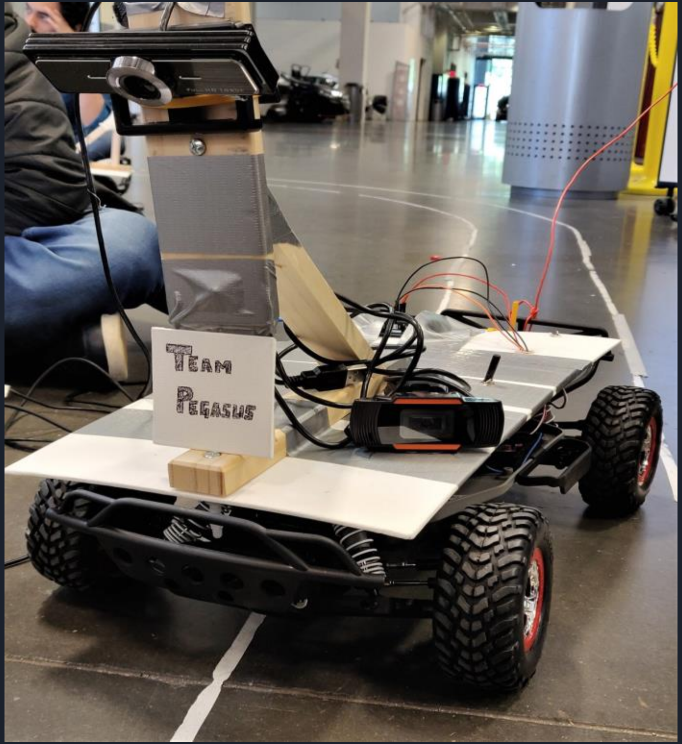
Design of Car

- Platform made out of cardboard, covering the length of the car
- The platform fixed to the car on the four corners
- Camera mounted on a wooden plank at the front edge of the car for lane tracking
- Another camera for sign board detection mounted on the platform at an angle
- Arduino mounted on the platform on the rear side of car



- ➢ Design Challenge
  The camera was initially mounted on the platform
  The problem with this design was the height of the camera
  The height wasn’t sufficient to map the lanes
- ➢ Solution
  The camera was placed at a greater height by mounting it on a wooden plank
  Initial height - 17cm
  Final height - 33cm


Initial Design

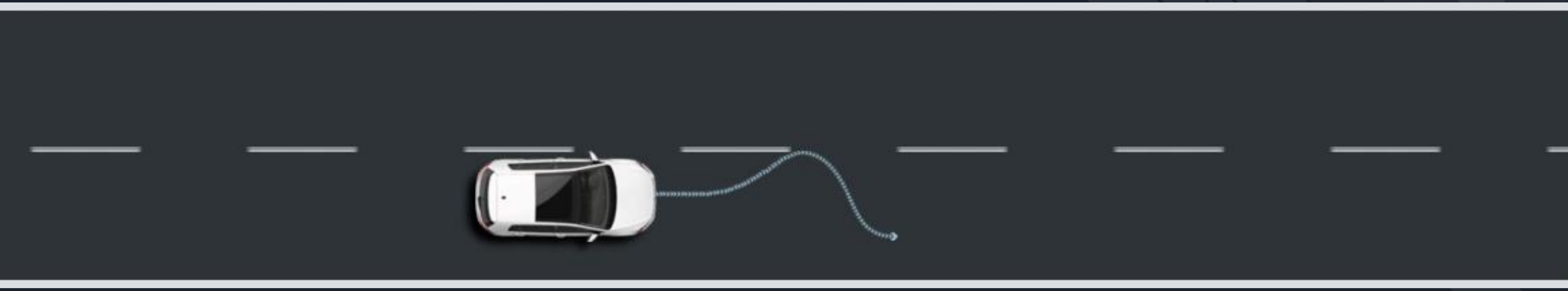


Final Design

LANE DETECTION

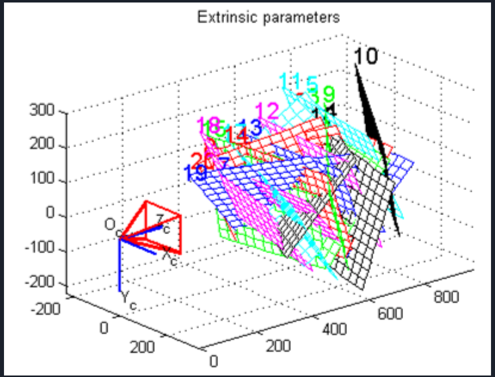
Lane Detection

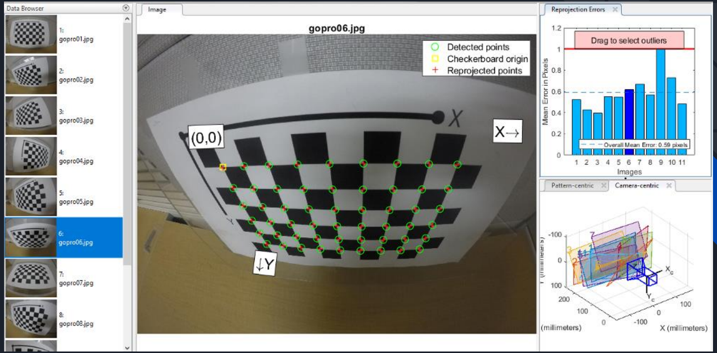
- Camera calibration
- Autonomous lane extraction
- Inverse projection



Camera Calibration

- Found intrinsic parameters using camera calibrator app in MATLAB
  Mean projection error = 0.19
- Found extrinsic parameters using option 1 which is manually calculating the translations and rotations of the camera wrt world frame
  Mounting Height of the Big Camera: 33cm from ground
  Camera angle: 28 degrees





Lane Extraction

Video from camera converted to grayscale image which was used to extract lines through hough transform

1. Masking
2. Kalman Filter
3. Post processing and inverse projection

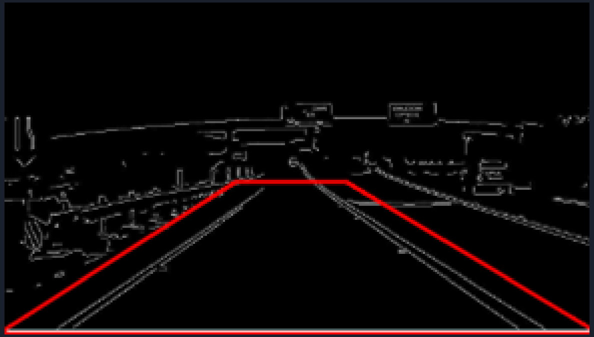
1. Masking

- In the earlier approach, a single masking region was defined. But this created the problem of no line detection on one side mainly at turns.

- Also angular threshold value needed to be constantly iterated.

- Assigning two masking regions for left and right lanes and individually applying hough transform to them rectified that issue. It also enabled to do away with angular threshold value.

- Only one line was extracted from each masking region and polyfitted to get lines of same length.

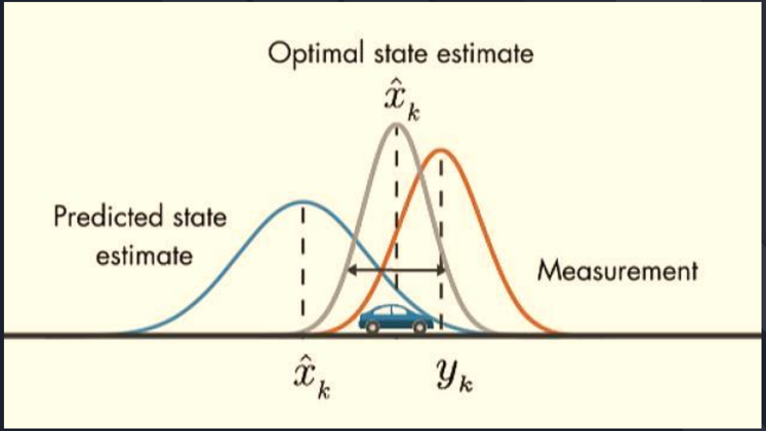


2. Kalman Filter

- Problem
  During lane detection it took 1-2 seconds for the detected lines to stabilize

- Solution
  Kalman filter was introduced to reduce those errors
  With the use of kalman filter the lines stabilized faster instantaneously
  Kalman filter was individually applied to left and right lanes to optimize the values

- Factors obtained by manual iterations
  Q = 0.03
  R = 0.01



3. Post Processing & Inverse Projection

- To make the extracted lanes of the same length, polyfit function was used over the lines. Vertical length for which polyfit needs to be carried out was reduced from the initial so as to make the model more robust and lines more stable.

- Based on camera parameters we carried out inverse projection from pixel frame to camera frame and then to world frame
- Intrinsic parameters

$F_u = 188.58$	$u_o = 323.5$
$F_v = 99.94$	$v_o = 172.5$

- Extrinsic parameters: Based on option 1, extrinsic matrix was calculated with following parameters.

$\theta_x = 30$ deg	$X_o = 0$
$\theta_y = 0$	$Y_o = 0$
$\theta_z = 0$	$Z_o = 0.33$

LANE TRACKING CONTROL

Stanley Controller Design

- The extracted left and right lane markers are represented with two points for each,
- We calculated the centerline of the lane by:

$$X_{cf} = \frac{1}{2}(X_{rf} + X_{lf}),\ Z_{cf} = \frac{1}{2}(Z_{rf} + Z_{lf})$$
$$X_{cn} = \frac{1}{2}(X_{rn} + X_{ln}),\ Z_{cn} = \frac{1}{2}(Z_{rn} + Z_{ln})$$

- Vehicle departure angle:

$$\theta_e = -\arctan\left(\frac{Z_{cf} - Z_{cn}}{X_{cf} - X_{cn}}\right) - 90$$

Controller Design

- We used stanley controller to control the steering of car
- After getting coordinates in world frame, values of θ_e and e_{fa} were calculated

Steps :

1. Corrected zero error of the steering
2. Based on manual iterations we concluded that the car was going straight at a steering value of 0.482
3. Assigned a gain value to θ_e and e_{fa} to check responses of steering at turn
   The gains of θ_e and e_{fa} were found to be +1 and -1 respectively
   Based on this our controller design was $\boldsymbol{\varphi} = 0.5 + k_1\theta_e + k_2 e_{fa}$

Challenges and achievements

- Gain optimization values for angle and departure distance. Initializing with the values of 0.1 and -0.1 respectively for the two parameters, car was not reacting at the terms. We eventually figured out that the gain value had to be increased. Final gain values were fixed at 1 and -1 respectively, making our controller model as follows:

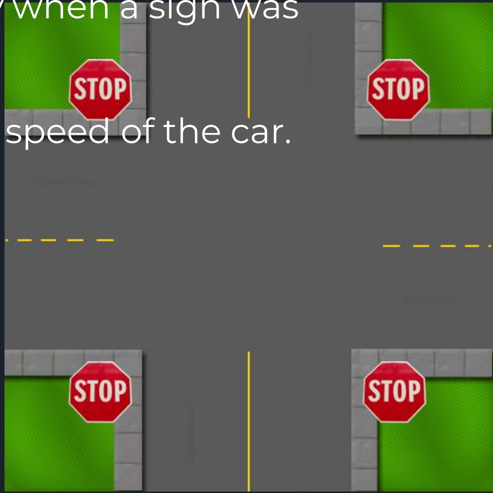
$$\phi=0\cdot5-\theta+e\_fa$$

- Sign detection challenge. Due to constant delay and lag in the sign detection, sign was not being properly detected. Also at times, the stop sign was being detected too early. This led us to find the right balance between fixing the camera resolution and subsequent resizing of the image to detect the signs at appropriate time.

- Also, camera angle had to be iterated and fixed to determine the correct position for detecting both the signs.

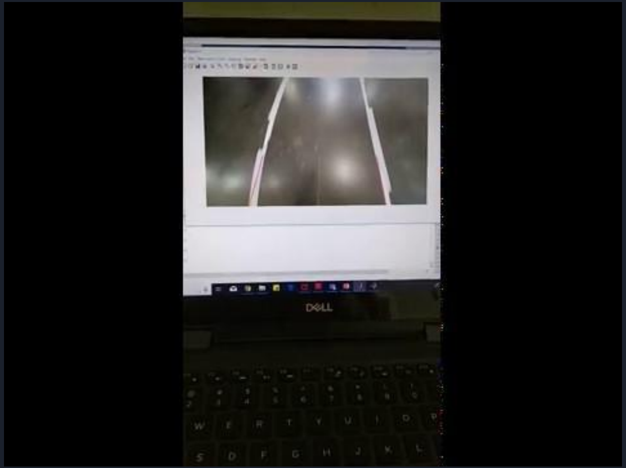
Road sign detection and Wi-Fi communication

- The objective was to stop the vehicle for 2 seconds at the stop sign and to reduce the speed by half after detecting the school sign.
- Deep Learning - Convolutional Neural Network method was used to carry out the sign detection training.
- 89 images for stop sign and 154 images for school sign were analyzed and trained.
- Data from first laptop was transferred to the second laptop only when a sign was detected, at other times there was no transmission.
- The input received from the first laptop was used to control the speed of the car.
- Camera mounting was adjusted to detect the signs properly.



Challenges and achievements

- Stabilizing the detected lines was one of the first issues faced. To resolve this, Kalman filter was implemented and the Q and R values were manually iterated to obtain deflection under acceptable limits in the lines.

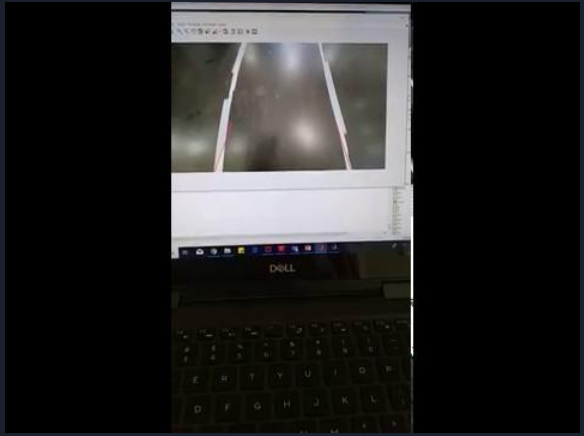


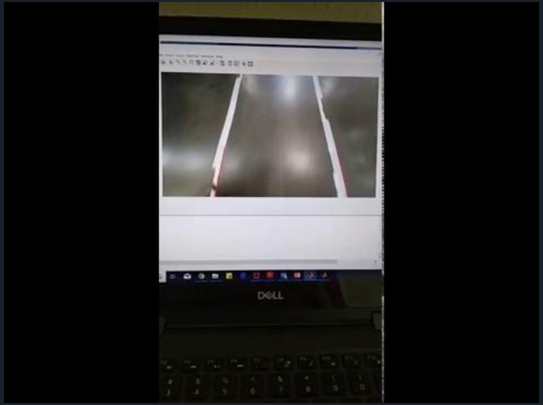
Without kalman filter

With kalman filter

Challenges and achievements

- Non-detection of lines at one side through houghpeaks command. At corners especially, only good lines at one side were being detected. To solve this, left and right regions were separately masked and only one line was extracted from each of the regions. This also helped do away with angular threshold value.


Single masking region

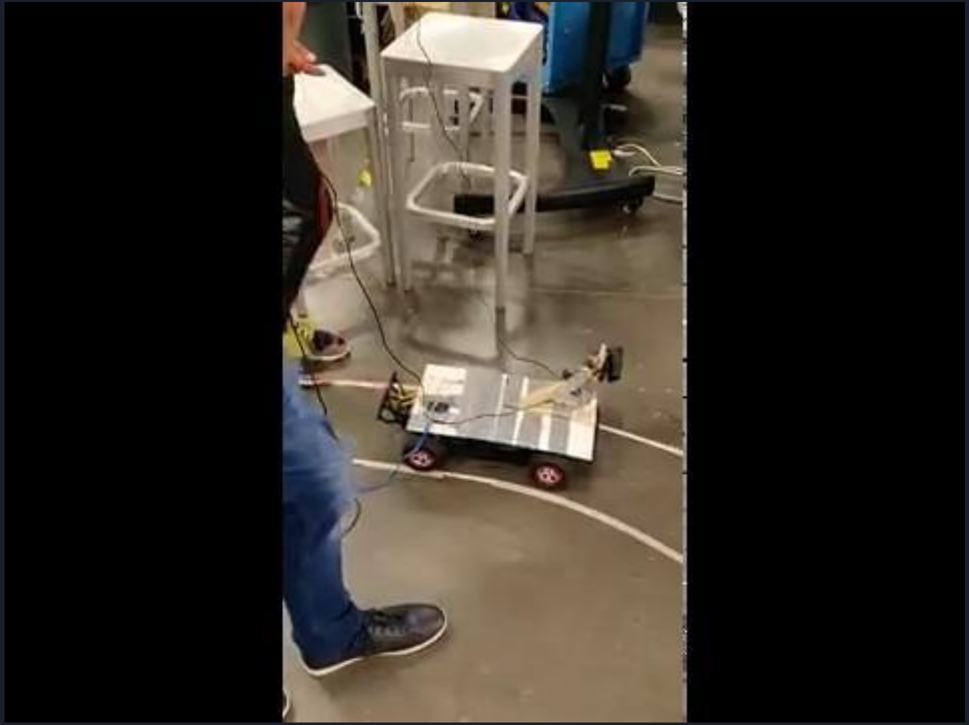

Dual masking region

Conclusion

- After addressing the above problems, the vehicle was able to successfully carry out lane tracking, control and road sign detection.
- The lane tracking was robust and the vehicle didn’t move out of the track during the course of the run.

Robust lane tracking illustration

Conclusion



Project completion