Working Manual

AI Self Driving Car

AI Self Driving Car using Computer Vision and Convolutional Neural Networks

AIM

To develop a miniature prototype of self driving car that can detect lanes and avoid obstacles by using camera and computer vision.

COMPONENTS REQUIRED

1.Hardware

- Chassis
- Motors
- Motor driver
- Arduino UNO
- Raspberry pi
- Fast SD card with size 16GB or more
- V2 NoIR Camera
- Power supply

2. Software

- Rasbian OS (Rasbian Buster) with CSI, SSH, VNC enabled.
- Python3, NumPy, SciPY, MatplotLib
- OpenCV, OpenCV_Contrib
- TensorFlow_Lite
- Pre-Trained Object Detection Model
- Geany Editor
- Arduino IDE
- VNC viewer, VNC Server

Pre-Requisites

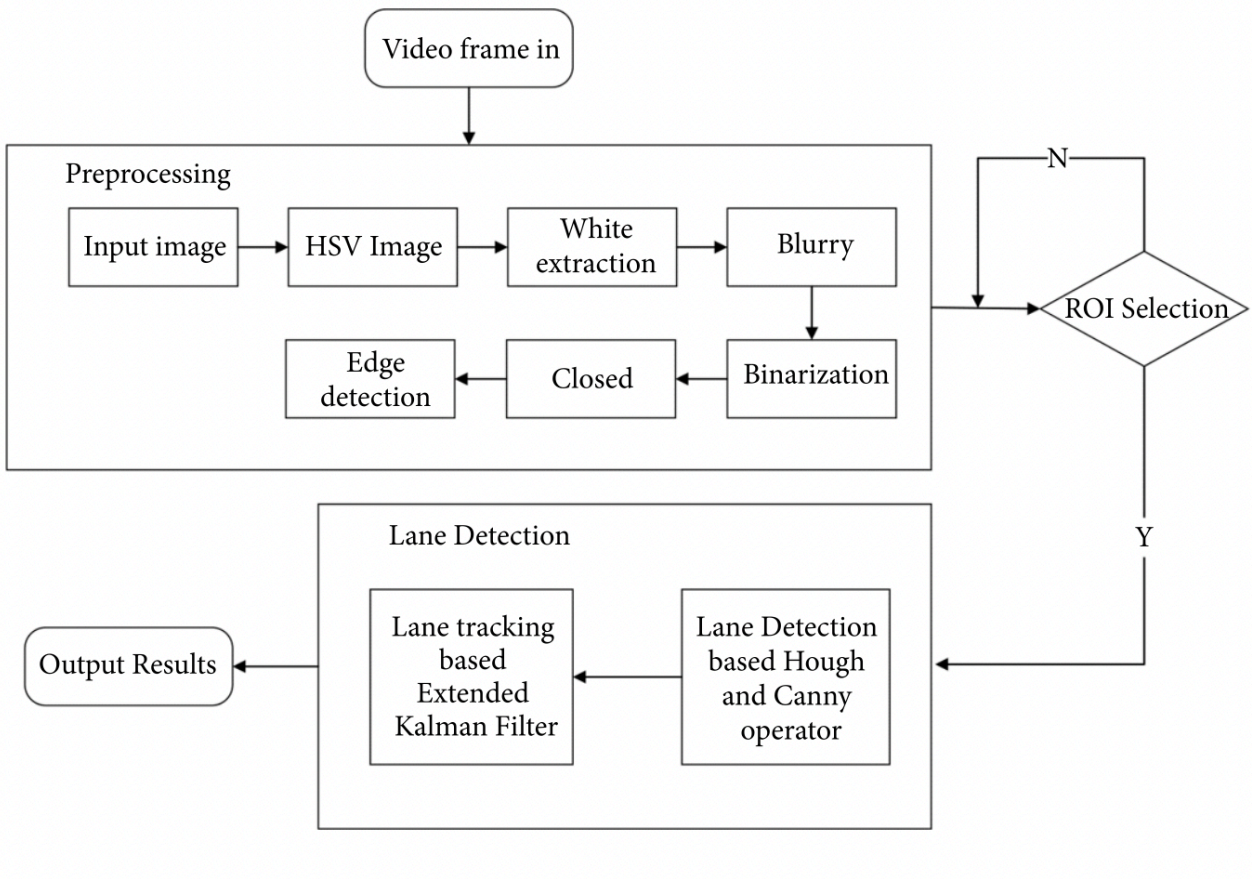
- Basic Networking Skills
- Working with Linux Terminal
- Python3
- TensorFlow

THEORY

A self-driving car (also known as an autonomous car or a driverless car) has no human input and can sense surrounding without any human interactions. Rushing around, trying to get errands done, thinking about the things to be bought from the nearest grocery store has become a part of our daily schedule. Driver error is one of the most common cause of traffic accidents, and with cell phones, in- car entertainment systems, more traffic and more complicated road systems, it isn't likely to go away.

Lane Detection

Lane detection is the task of detecting lanes on a road from a camera. Lane Detection algorithm helps car navigate through the lanes of road, It is achieved by OpenCV library.



Flow Chart Lane Detection Algorithm

Capturing and decoding video file: We will capture the video using VideoCapture object and after the capturing has been initialized every video frame is decoded (i.e. converting into a sequence of images).

Grayscale conversion of image: The video frames are in RGB format, RGB is converted to grayscale because processing a single channel image is faster than processing a three-channel colored image.

Region of Interest: This step is to take into account only the region covered by the road lane. A mask is created here, which is of the same

dimension as our road image. Furthermore, bitwise AND operation is performed between each pixel of our canny image and this mask. It ultimately masks the canny image and shows the region of interest traced by the polygonal contour of the mask.

Reduce noise: Noise can create false edges, therefore before going further, it's imperative to perform image smoothening. Gaussian filter is used to perform this process.

Canny Edge Detector: It computes gradient in all directions of our blurred image and traces the edges with large changes in intensity. For more explanation please go through this article: Canny Edge Detector

Hough Line Transform: The Hough Line Transform is a transform used to detect straight lines. The Probabilistic Hough Line Transform is used here, which gives output as the extremes of the detected lines.

Canny Edge Detection

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

The Canny edge detection algorithm is composed of 5 steps:-

1. Noise reduction;
2. Gradient calculation;
3. Non-maximum suppression;
4. Double threshold;
5. Edge Tracking by Hysteresis.

Noise Reduction

One way to get rid of the noise on the image, is by applying Gaussian blur to smooth it. To do so, image convolution technique is applied with a Gaussian Kernel (3x3, 5x5, 7x7 etc…). The kernel size depends on the expected blurring effect. Basically, the smallest the kernel, the less visible is the blur. In our example, we will use a 5 by 5 Gaussian kernel.

The equation for a Gaussian filter kernel of size (2k+1)×(2k+1) is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

Gradient Calculation

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators. Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highight this intensity change in both directions: horizontal (x) and vertical (y) When the image is smoothed, the derivatives Ix and Iy w.r.t. x and y are calculated. It can be implemented by convolving I with Sobel kernels Kx and Ky, respectively:-

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

Sobel filters for both direction (horizontal and vertical). Then, the magnitude G and the slope θ of the gradient are calculated as follow:-

$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x,y) = arctan\left(\frac{I_y}{I_x}\right)$$

The result is almost the expected one, but we can see that some of the edges are thick and others are thin. Non-Max Suppression step will help us mitigate the thick ones.

Moreover, the gradient intensity level is between 0 and 255 which is not uniform. The edges on the final result should have the same intensity (i-e. white pixel = 255).

Non-Maximum Suppression

Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression too thin out the edges. The algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.

Double threshold

The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant:-

1. Strong pixels are pixels that have an intensity so high that we are sure they contribute to the final edge.
2. Weak pixels are pixels that have an intensity value that is not enough to be considered as strong ones, but yet not small enough to be considered as non-relevant for the edge detection.
3. Other pixels are considered as non-relevant for the edge.
4. High threshold is used to identify the strong pixels (intensity higher than the high threshold)
5. Low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold)

6. All pixels having intensity between both thresholds are flagged as weak and the Hysteresis mechanism (next step) will help us identify the ones that could be considered as strong and the ones that are considered as non-relevant.

Edge Tracking by Hysteresis

Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one, as described below:-

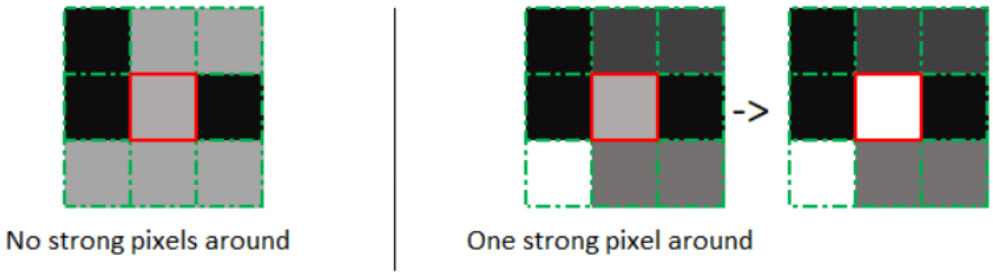


Figure- Edge tracking

Object Detection

Object detection is commonly associated with self-driving cars where systems blend computer vision and various AI technologies to generate a multidimensional representation of the road with all its participants.

Image classification

aims at assigning an image to one of a number of different categories (e.g. car, dog, cat, human, etc.), essentially answering the question “What is in this picture?”. One image has only one category assigned to it.

Object localisation

allows us to locate our object in the image, so our question changes to "What is it, and where it is?".

Object detection

finding all the objects in an image and drawing the so-called bounding boxes around them. In a real-life scenario, we need to go beyond locating just one object but multiple objects in one image. For example, a self-driving car has to find the location of other cars, traffic lights, signs, and humans and take appropriate action based on this information.

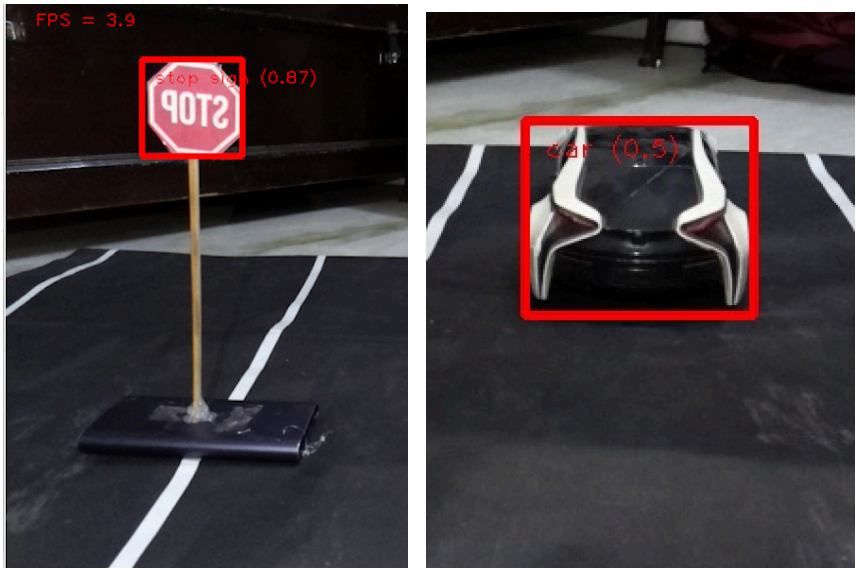


Figure- Object Detection Algorithm

CIRCUIT DIAGRAM

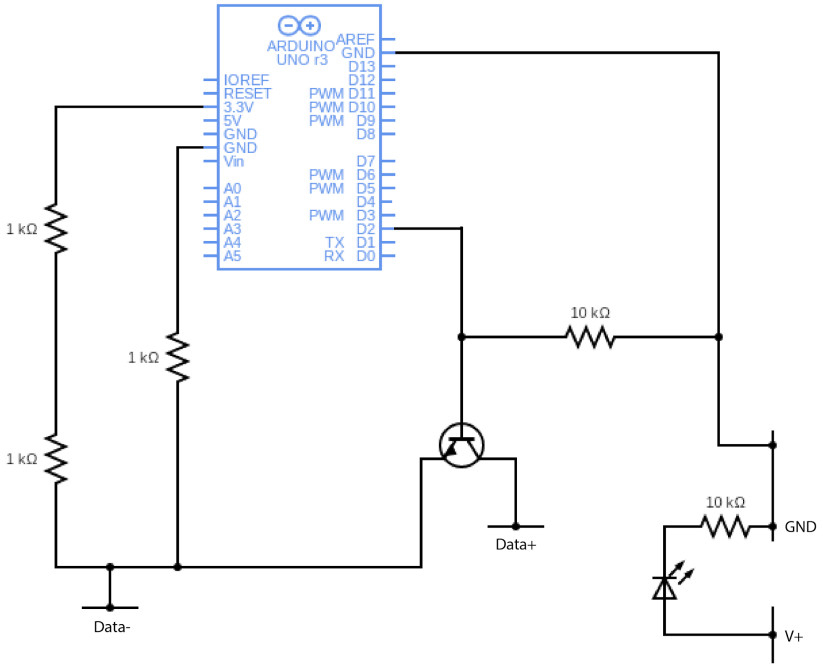


Figure 1- OC3 trigger board

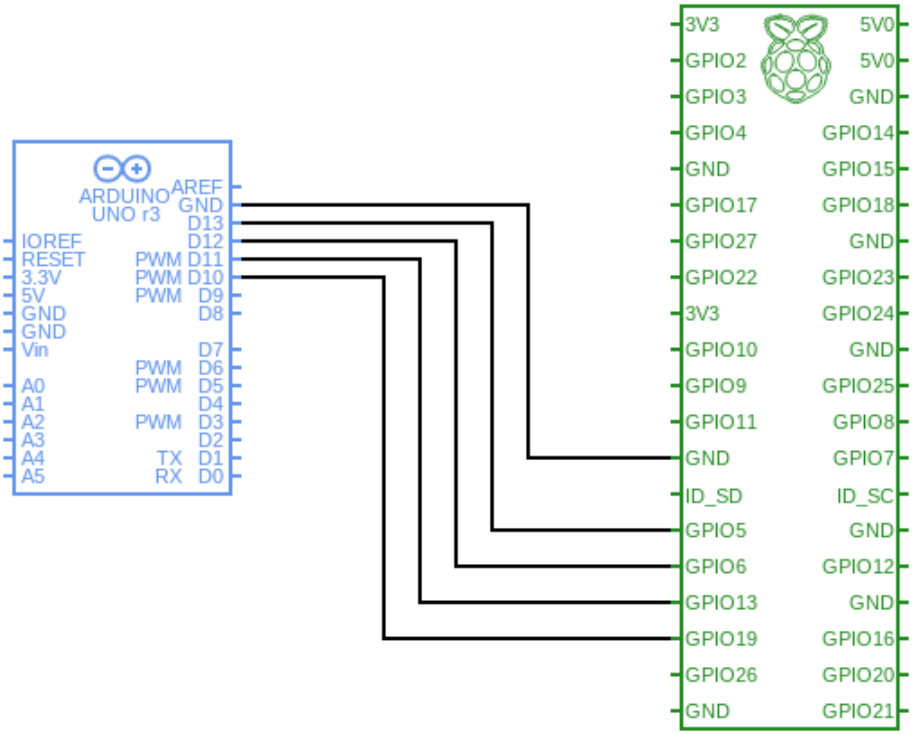


Figure 2- Circuit Connection of Arduino UNO and L298 motor controller

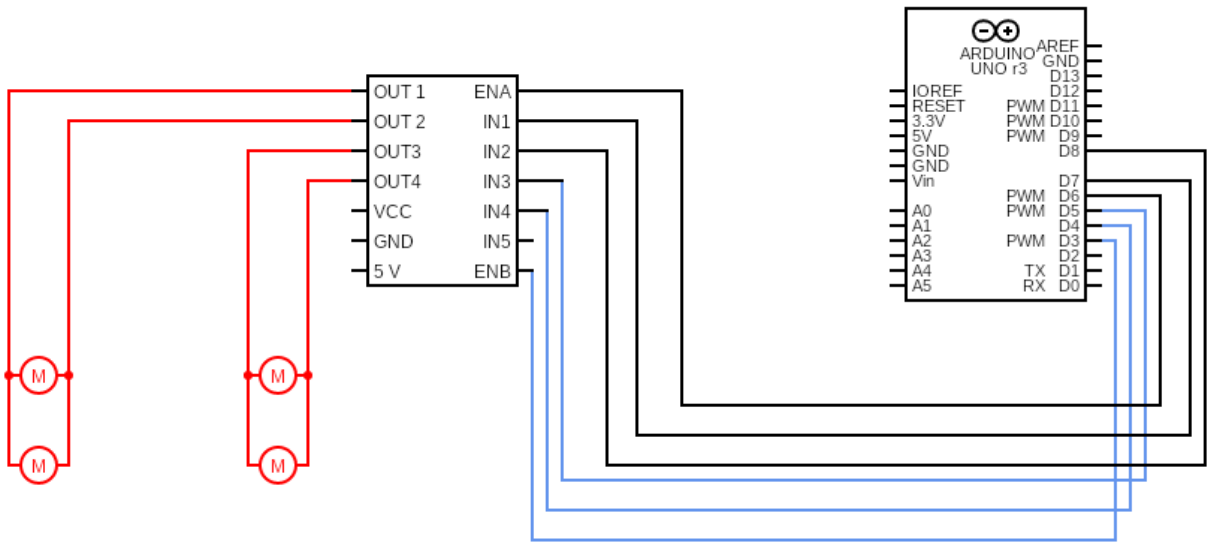


Figure 2- Circuit Connection of Arduino UNO and L298 motor controller

PROCEDURE

1. Connect the Arduino UNO with RaspberryPi using the same pin as shown in Figure 1.
2. Know connect the motor to the Arduino UNO via motor controller L298 as mentioned in the Figure 2.
3. The 4 wheels of the chassis are connected to 4 separate motors parallel.
4. The chassis has two shelves over the wheels separated by 2 inch approx.
5. The IC is fixed on the lower shelf with the help of two 0.5 inch screws.
6. It is permanently connected to the motor wires and necessary jumper wires are drawn from L298 to connect to Arduino UNO.
7. The rest of the space on the lower shelf is taken by power bank 2 which provide the power to run the buck converter using 12V voltage trigger.

8. To control the motor connected to OUT1, OUT2, OUT3, OUT4 of motor driver, Left side motor's ENA, IN1, IN2 are connected to digital pins 5~, 6,7 of Arduino UNO where '~' indicates PWM pins, Right side motor's ENB, IN3, IN4 are connected to digital pins 3~,4,5 of Arduino UNO.

Connection to RaspberryPi

1. Open Terminal in Mac/Open command prompt in Windows.
2. Type ssh user@[IPv4 address of RaspberryPi] and enter user password.
3. Open VNC viewer in Mac/Windows.
4. Make a new connection by entering IP address and user password.
5. After a successful connection the RaspberryPi desktop will open.

Lane Detection Application

1. In RaspberryPi desktop open Terminal.
2. Type cd desktop
3. ./calibration.cpp, or open calibration.cpp from desktop
4. Once Geany Editor opens, place the car on track
5. Click RUN.

Object Detection

1. Open Terminal on RaspberryPi
2. Cd " Location of detect.py dir"
3. Type python3 -r detect.py \ --model "location of Pre-Trained Model"