IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-15, May 2019 E-ISSN: 2347-2693

Self Driving Car Using Deep Neural Networks

Sharmila S[1], Shivaswaroop S[2], Sudhakar M[3], Tejashwini S V[4] , Rajshekhar S A[5*]

[1,2,3,4,5]Department of Computer Science, East West Institute of Technology, Bengaluru, India

Corresponding Author: rajshekhar@ewit.edu

DOI: https://doi.org/10.26438/ijcse/v7si15.171176 | Available online at: www.ijcseonline.org

Abstract— Automation has a wide role in the current generation which can be deployed into cars making them drive on their own by considering the surrounding environment as input parameters. Detection of lanes using canny edge lane detection algorithm helps to detect lanes and ensure the drivable space and have clear information of lane in which the car is moving. Deep Neural Networks helps in deciding the action to be performed by the car (forward, reverse, right, left, stop, and park). This paper covers motion control, path detection and obstacle detection. The results have been achieved by the implementation of Canny Edge Detection Algorithm, Deep Neural Networks Techniques.

Keywords— Deep Neural Network, Canny Edge Detection Algorithm, Obstacle Detection.

I. INTRODUCTION

The number of road accidents is increasing every year. And the road accidents could be caused either by the human mistake or the vehicle error. Example for human errors possibly could be drunk and drive, drowsiness, breaking traffic rules etc and vehicle errors. To overcome these tough situations, self driving car has been implemented so as to reduce the number of accidents on road.

Highway driving and parking assist functions have been implemented in self driving car which is increasing day by day. In this system, a Self Driving Car is developed, which also includes ultrasonic sensors, IR sensors which prevents the vehicle from getting damaged by an obstacle. The Vehicle movements i.e. parking of the vehicle are all assisted by the user. The Motion Control, Obstacle Detection, and Lane Detection help the car to move in the proper path. The Motion Control planning has been implemented which helps in movement of the car. As the rate of road accidents are increasing by the conditions of road it is important to consider it aswell to protect our vehicle from it. The Obstacle Detection module finds the obstacle if present in the travelling path and sends the alert to the vehicle to stop or take diversions.The Lane detection module implemented using the Canny Edge Detection Algorithm helps the vehicle to find the lane lines on the road easily, even in the wide variety of conditions.

Typical Deep Neural Network (DNN) structure is illustrated as shown in Figure 1. Deep learning is the process of applying deep neural network technologies - that is, neural network architectures with multiple hidden layers - to solve problems. **Deep** neural networks are neural networks with one hidden layer minimum (see below). Like data mining, deep learning refers to a **process**, which employs deep neural network architectures, which are particular types of machine learning algorithms.

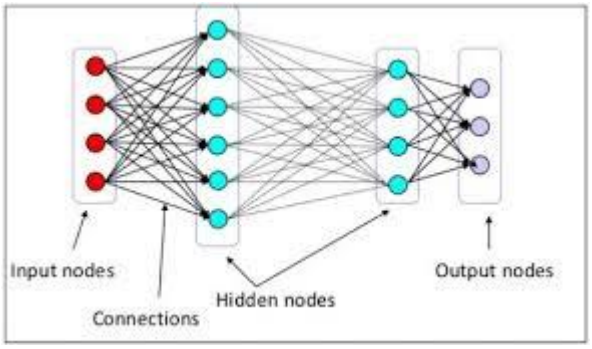


Fig 1.1: Deep Learning key terms

As outlined in Figure 1, MLPC consists of multiple layers of nodes including the input layer, hidden layers (also called intermediate layers), and output layers. Each layer is fully connected to the next layer in the network. Where the input layer, intermediate layers, and output layer can be defined as follows:

- The **input layer** consists of neurons that accept the input values. The output from these neurons is same as the input predictors. Nodes in the input layer represent the input data. All other nodes map inputs to outputs by a linear combination of the inputs with the node's weights **w** and bias **b** and applying an activation function. This can be written in matrix form for MLPC

with K+1 layers as follows : $\mathbf{y}(\mathbf{x}) = \mathrm{f_K}(\ldots \mathrm{f_2}(\mathbf{w}_2^T \mathrm{f_1}(\mathbf{w}_1^T \mathbf{x} + b_1) + b_2) \ldots + b_K)$

- **Hidden layers** are in between input and output layers. Typically, the number of hidden layers range from one to many. It is the central computation layer that has the functions that map the input to the output of a node. Nodes in the i**ntermediate layers** use the sigmoid (logistic) function, as follows :

$$\mathrm{f}(z_i) = \frac{1}{1 + e^{-z_i}}$$

- The **output layer** is the final layer of a neural network that returns the result back to the user environment. Based on the design of a neural network, it also signals the previous layers on how they have performed in learning the information and accordingly improved their functions. Nodes in the **output layer** use softmax function:

$$\mathrm{f}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}}$$

The number of nodes N, in the output layer, corresponds to the number of class.

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through each stages.

1. **Noise Reduction**

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

2. Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows:

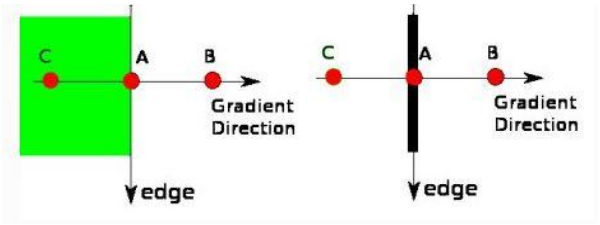
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

3. Non-maximum Suppression

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighbourhood in the direction of gradient as shown below:



Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

In short, the result you get is a binary image with "thin edges".

4. Hysteresis Thresholding

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, *minVal* and *maxVal*. Any edges with intensity gradient more than *maxVal* are sure to be edges and those below *minVal* are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:

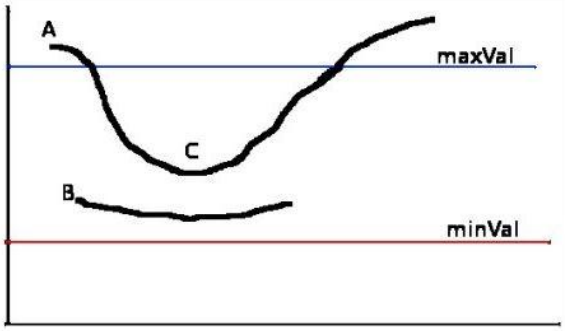


Fig1.2: Hysteresis thresholding

The edge A is above the *maxVal*, so considered as "sure-edge". Although edge C is below *maxVal*, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above *minVal* and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select *minVal* and *maxVal*accordingly to get the correct result.

This stage also removes small pixels noises on the assumption that edges are long lines.

So what we finally get is strong edges in the image.

II. RELATED WORK

[1] Aditya Kumar Jain, "Working model of Self-driving car using convolutional Neural Network, Raspberry Pi and Arduino", where, this paper proposes a working model of self-driving car which is capable of driving from one location to the other or to say on different types of tracks such as curved tracks, straight tracks and straight followed by curved tracks. A camera module is mounted over the top of the car along with Raspberry Pi sends the images from real world to the CNN which then predicts one of the following directions .i.e., left, stop, right or forward.

[2] Mihir Mody," Low Cost and Power CNN/Deep Learning Solution for Automated Driving" In the case of automated driving, one of the key functionality is "finding drivable free space", which is addressed using deep learning techniques like CNN. These CNN networks pose huge computing requirements in terms of hundreds of GOPS/TOPS (Giga or Tera operations per second), which seems beyond the capability of today's embedded SoC. This paper covers various techniques consisting of fixed-point conversion, sparse multiplication, fusing of layers and network pruning, for tailoring on the embedded solution. These techniques are implemented on the device by means of optimized Deep learning library for inference. The paper concludes by demonstrating the results of a CNN network running in real time on TI's TDA2X embedded platform producing a high-quality drivable space output for automated driving.

III. METHODOLOGY

This paper consists of 5 modules, which are,

1. **Image Processing:** Images are captured at the rate of 32 frames per minute using raspberry pi camera. These images are given as input to the raspberry pi on which the canny edge algorithm is applied to detect edges or lanes. And also it detects the angle of curvature of the road so as the car has to adjust its turning radius to ensure that the car is moving in the proper lane.

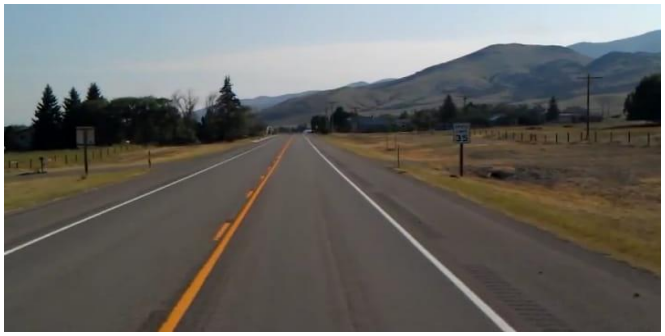


Fig3.1: Lanes before detection

Fig3.2: Lanes after detection

2. Obstacle detection: As every vehicle needs its own driving space, it is very often to meet up an obstacle. Hence in order to avoid collisions with obstacles, the minimum distance has to be maintained between our car and the obstacle at the front. Therefore, we use appropriate sensors (Ultrasonic & Infrared) to ensure there is no obstacle within the range of driving space so that our car can move without any collisions. Where the ultrasonic sensors gives the difference of distance between our car and the obstacle in the front and infrared sensors detects the immediate obstacle around the car and based on these outputs, the motion of the car can be decided(right, left, forward, stop or backward).

3. Deep Neural Networks: It is a neural network with a certain level of complexity and with more than 2 layers. It uses sophisticated mathematical modelling to process data in complex ways. It takes lanes angles and distance to the barricades if exists and also the obstacle distance as inputs and helps in making a decision (turning radius, right, left, forward, backward or stop). It is given with the learning dataset upon which it learns and makes an optimal predictions. It has higher accuracy and precisions about 83.37 percent on all the decisions it makes. This helps is to ensure the proper and safe movement of the car within its lane.

.

4. Motion Planning: The motion of the car is an important aspect to be considered. The motion planning has 4 modules within it where each module defines the direction in which direction the car has to move or what type of motion to be made by the car. The modules such as:

Right(): indicates to turn right.
Left(): indicates to turn left.
Forward(): indicates to move forward.
Reverse(): indicates to move backward.
Stop(): indicates to stop.

5. Parking: There are situations which cannot be predicted. Henceforth on a precautionary measure considering the safety of the passengers of the car there is a parking switch provided which on activation commands the car for emergency and has to park the vehicle towards left as per

Indian traffic rules where it initially determines the lane in which it is running currently and decides either to stop onthe same line(if running in left lane) or to change its lane and then stop(if running in right lane)

IV ALGORITHMS USED

1. CANNY EDGE DETECTION:

This technique is used to detect the edges or the lanes on the road.

Canny edge detection algorithm:

Step I: Noise reduction by smoothing:

Mathematically, the smooth resultant image is given by

$$F(i,j) = G * I(i,j)$$

Step II: Finding Gradients:

In this step we detect the edges where the change in grayscale intensity is maximum. Sobel operators are used to determine the gradients at each pixel of smoothened images.

$$D[I] = \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +3 \end{pmatrix} \quad D[J] = \begin{pmatrix} -1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

These sobel marks are convolved with smoothed image and giving gardients in i and j directions.

$$Gi=Di*F(i,j) \text{ and } Gj=Dj*F(i,j)$$

Therefore edge strength or magnitude of gradient of a pixel is given by:

$$G= (Gi^2 +Gj^2)^{½}$$

The direction of gradient is given by:

$$Θ= \arctan(Gj/Gi)$$

Step III: Non maximum suppressions:

Non maximum suppression is carried out to preserves all local maxima in the gradient image, and deleting everything else this results in thin edges. For a pixel M (i, j):

• Firstly round the gradient direction nearest45°,then compare the gradient magnitude of the pixels in positive and negative gradient directions.

• If the edge strength of pixel M (i, j) is largest than that of E (i, j) and W (i, j), then preserve the value of gradient and mark M (i, j) as edge pixel, if not then suppress or remove.

Step IV: Hysteresis Thresholding:

For a pixel M (i, j) having gradient magnitude G following conditions exists to detect pixel as edge:

- If G >789 than discard the edge.
- If G > than 565 keep the edge.
- If 789 < G < and565 and any of its neighbors in a 3 ×3 region around it have gradient magnitudes greater than565, keep the edge.

2. OBSTACLE DETECTION:

Algorithm 1 Segmentation:Groud

IF $|\overline{H}_i - H_g| > \delta_1$
THEN Grid i belongs to the non-ground object.
ELSE IF $(\sigma_H^2)_i > \delta_2$
THEN Grid i belongs to the non-ground object.
ELSE IF $\Delta H_i < \delta_3$
THEN Grid i belongs to the non-ground object.
ELSE Grid i belongs to the ground.

Algorithm 2 Segmentation:Vehicle-like Objects

IF $l_i < 1.5m$ or $l_i > 7m$
THEN Cluster i belongs to the non-vehicle-like object.
ELSE IF $w_i < 1.5m$ or $w_i > 7m$
THEN Cluster i belongs to the non-vehicle-like object.
ELSE IF $h_i < 1.0m$ or $h_i > 4m$
THEN Cluster i belongs to the non-vehicle-like object.
ELSE Cluster i belongs to a vehicle-like object.

3. DEEPLEARNING:

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function ˆ Q with weights ¯ θ = θ
For episode = 1, M do
Initialize sequence s1=x1 and preprocessed sequenced φ1 = φ(s1) For t =1, T do
With probability ϵ select a random action at
Set s(t+1)=st,at,x(t+1)
Store transition (φt, at, rt, φt+1) in D
Sample random mini batch of transitions (φj , aj , r j,φj+1) from D.
Set yj={rj, if episode terminates at step j+1
{rj + γ max′ aˆ Q(φj+1,a ′; ¯ θ), otherwise
perform a gradient descent step on (yjQ(φj,aj;θ))2 w.r.t. network parameters θ
Every C steps,reset Q'=Q
End for
End for

IV. RESULTS AND DISCUSSION

The model has run within its lane by detecting the right or left edges and drive within the proper lane. It detects the curve in the lane and decides the angle at which the car has to rotate so as to move in the proper lane in order to avoid collisions and in case of emergency the car is provided with emergency parking and also barricades in front of the car.

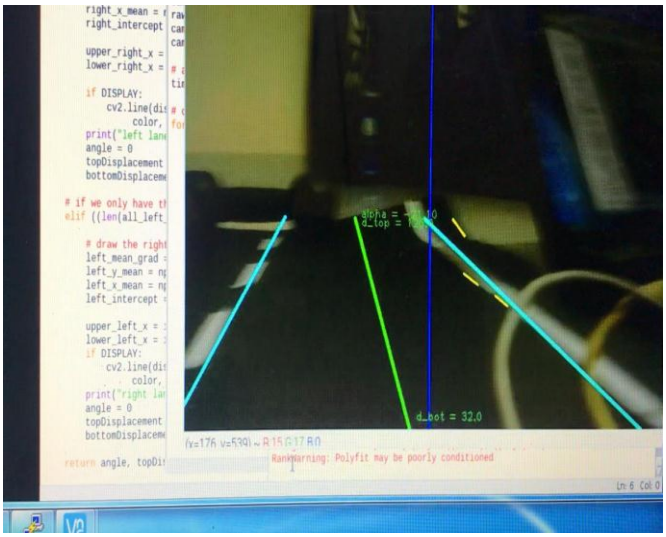


Fig 4.1: Lane detection

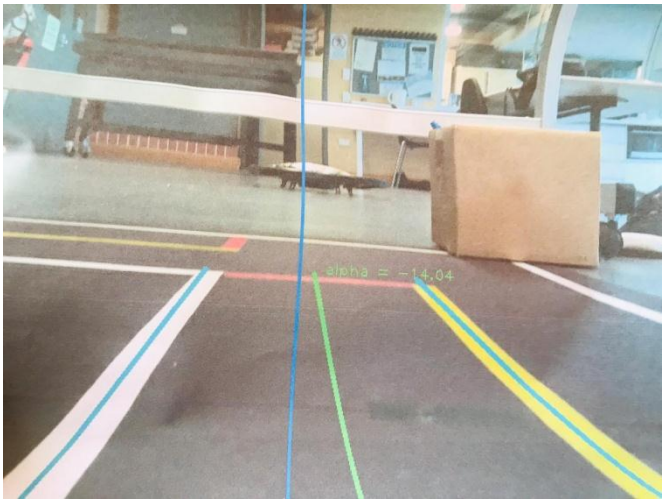


Fig 4.2: Barricade detection

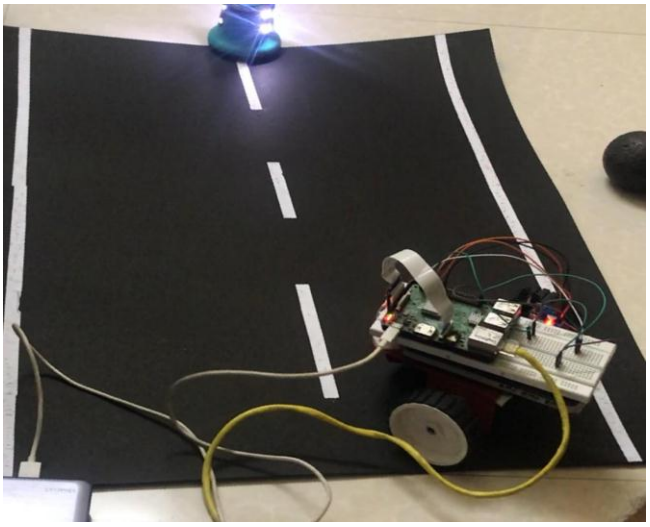


Fig 4.3: Emergency Parking

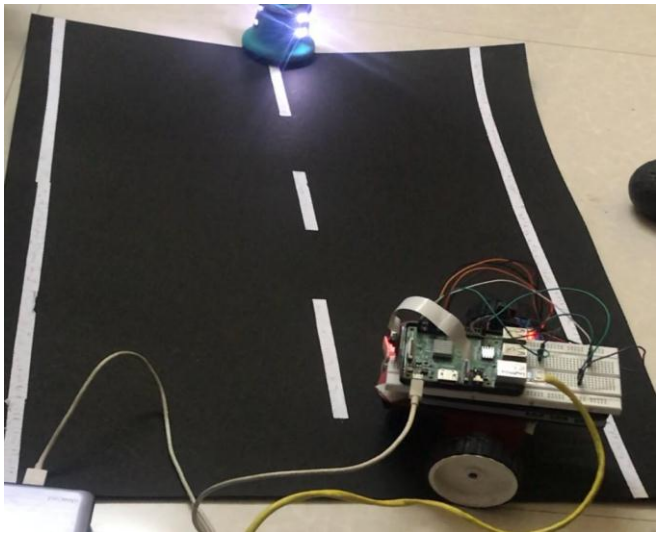


Fig 4.4: Turns left to park

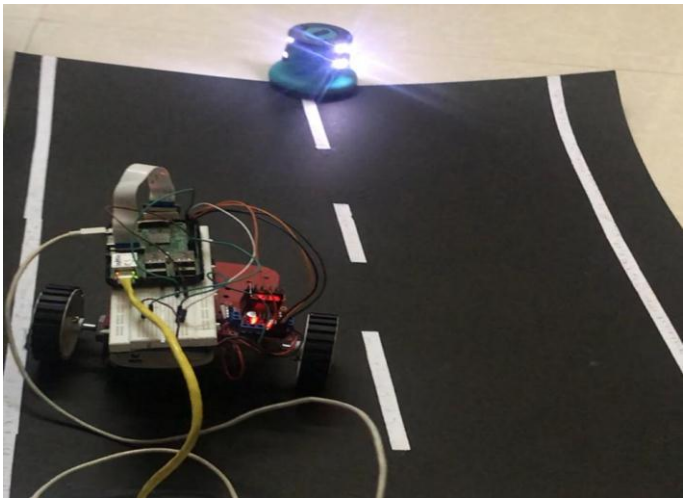


Fig 4.5: Parks on left

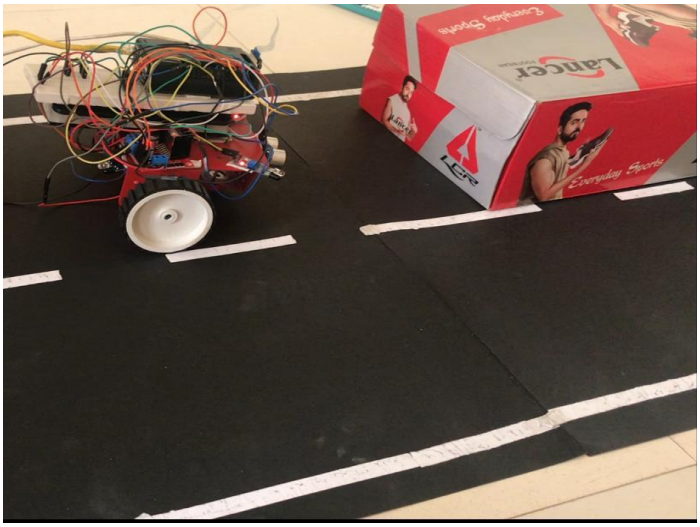


Fig 4.6: Detects obstacle ahead

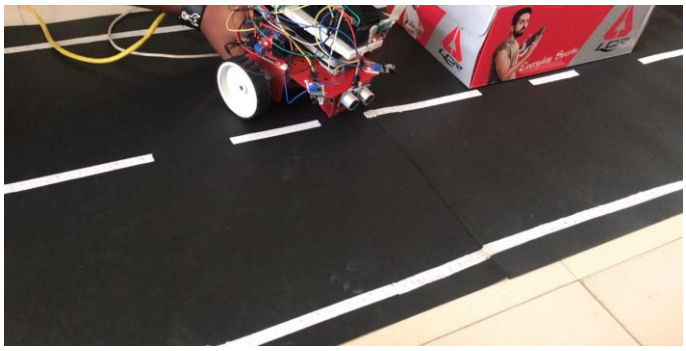


Fig 4.7: changes lane for drivable space

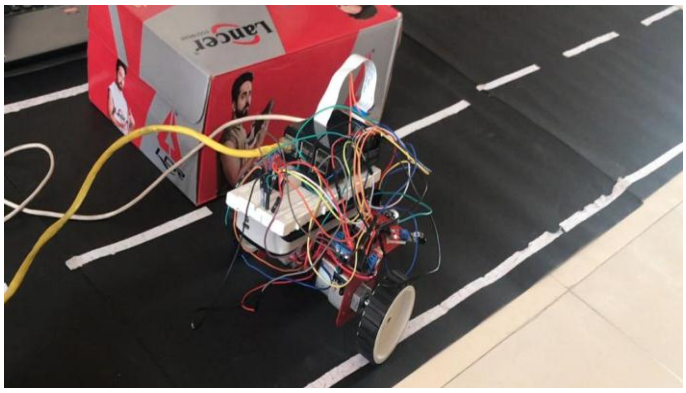


Fig 4.8: Moves on the free lane

V. CONCLUSION AND FUTURE SCOPE

Conclusion: This paper presents the working prototype of self driving car which uses Canny Edge Detection algorithm for detecting lanes and neural networks to make optimal decisions which can help to reduce most of the road risks.

Future work: There are many improvisations that can be brought to this model such as implementing GPS, NLP and so on. Therefore it can be made interactive with the user and keep monitor of every parameters of the car for the safer drive.

VI ACKNOWLEDGMENT

Sharmila S, Shivaswaroop S,Sudhakar M and Tejaswini S V would like to thank **my guide Mr. Rajshekhar S A Associate Professor, CSE Dept.**, East West Institute of Technology for providing the facilities.

Finally, I would like to thank all the Teaching, Technical faculty and supporting staff members of Department of Computer Science and Engineering, East West Institute of Technology, Bengaluru, for their support.

REFERENCES

[1] TU-Automotive, "Driverless vehicles will continue to dominate auto headlines tu automotive [online]," April, 2016, available: http://analysis.tu-auto.com/autonomous-car/driverless-vehicles-willcontinue- dominate-auto-headlines. [Accessed: 10-April-2018].

[2] L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, J. Kindelsberger, L. Ding, S. Seaman, H. Abraham, A. Mehler, A. Sipperley, A. Pettinato, B. Seppelt, L. Angell, B. Mehler, and B. Reimer, "Mit autonomous vehicle technology study: Large-scale deep learning based analysis of driver behavior and interaction with automation," Nov 2017, available:https://arxiv. org/abs/1711.06976.

[3] WHO, "Global status report on road safety 2015. world health organization," 2015.

[4] Saha, Anik, et al. "Automated road lane detection for intelligent vehicles." *Global Journal of Computer Science and Technology (2012).*

[5] Pannu, Gurjashan Singh, Mohammad Dawud Ansari, and Pritha Gupta. "Design and implementation of autonomous car using Raspberry Pi." *International Journal of Computer Applications 113.9 (2015)*

[6] Mohanapriya, R., Hema, L. K., Yadav, D., & Verma, V. K. (2014). Driverless Intelligent Vehicle for Future Public Transport Based On GPS. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 3.*

[7] Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino Aditya Kumar Jain Electronics and Communication Department Dharmsinh Desai University Gujarat, India

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed,D. Anguelov, D.Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on computer Vision and Pattern Recognition, pages 1–9, 2015.

[9] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. arXiv:1712.05877, 2017.

