International Journal of Research in Engineering, Science and Management
Volume 5, Issue 2, February 2022
https://www.ijresm.com | ISSN (Online): 2581-5792

Self-Car Driving using Artificial Intelligence and Image Processing

Mohammed Abdul Muqeet[1*], Rayyan Mohiuddin[2], Adith Thaniserikaran[3], Ikramuddin Ahmed[4], Kamuju Abhi Subrahmanyam[5], Anoohya Podapati[6]

***Abstract*: Self-riding self-sufficient motors are the solution for reinforcing mobility intelligence related to using. This mission presents an effective manner for implementation of a self-using automobile. Proposed paintings are based totally mostly on Artificial Intelligence, Computer Vision and Neural Networks. In our mission, we're the usage of many features which encompass mapping, monitoring and local planning. We can efficaciously create a vehicle that may monitor right lane modifications, parking, and U-activates its own. The extraordinary innovations we're using are obstacles and cut down detection techniques, avenue vehicle tracker, and checking special site visitors' situations. This will make a sturdy self-enough self-driven automobile. It will efficiently reveal right parking allotment, lane adjustments, and automated U-turns. We can do those the use of the obstacle and various reduce detection method, the automobile tracker. Self-driving cars combine a selection of sensors to perceive their environment, inclusive of radar, lidar, sonar, GPS, odometry and inertial length gadgets. Advanced manage structures interpret sensory facts to understand appropriate navigation paths, as well as limitations and applicable signage. Long distance trucking is seen as being on the main fringe of adopting and imposing the technology. We use Artificial Intelligence for recognizing and presenting the course that the self-sufficient car should observe for correct running. Additionally, a driverless car can reduce the time taken to reach the holiday spot because it will take the shortest direction, fending off the website traffic congestion. Human errors may be prevented thereby allowing disabled human beings (even blind humans) to personal their automobile.**

***Keywords*: Self-usage of independent automobiles, Computer Vision, Neural Networks, Artificial Intelligence.**

1. Introduction

This venture is a proposed model to gain driverless vehicles. Research continues to be happening; we are hoping that this simulated model can be included as software program application in actual-life vehicles in the near future to construct a reliable device to growth the Image Enhancement without lack of exceptional, the project can be prolonged and used with special schooling models to increase the accuracy of detecting turns by the use of the car. Our self-enough automobile is able to run itself on any generalized tune with the same schooling and validation accuracy. The system needs to be given enter photographs and optimize for quicker processing. To acquire, monitoring and local making plans. Demonstrate right lane changes, parking, and U-activates its very personal. The exclusive improvements we're the use of are limitations and decrease detection methods, road vehicle tracker, and checking different traffic conditions. Create a strong self-maintaining self-driven car. It will efficaciously exhibit right parking allotment, lane modifications, and automatic U-turns. We can do the ones using the impediment and various reduce again detection strategies, the automobile tracker.

A. Dataset

To gather the facts Udacity, has a car simulator wherein a song is supplied and we will stress the automobile on the tune manually. The simulator has a document button. So, while we click on it the choice includes select a folder then while we start the usage of the auto on the song the simulator shops the photo at each immediately and moreover shops the car steering angle similar to every photo. The automobile simulator has three cameras which save pictures as left, proper and center. For having a right properly-spaced dataset containing all cases and angles, it's far endorsed to take at least take 3 laps in ahead and contrary path on the song.

1) Data preprocessing

Deleting immoderate-frequency dataset values which make the model biased: Since most of the time, we stress via the middle of the music the dataset carries a excessive quantity of 0-diploma guidance angles which may also make the self-driving simulator biased toward predicting 0-diploma attitude so one can lead to the model predicting zero angles and result in a crash. So first we drop some zero-degree mind-set values.

2. Training Set and Validation Set Splitting of Dataset for Better Model Creation and Prevention of Overfitting of Training Data

If we do now not make a validation set then the model will overfit and obtained paintings nicely for generalized tracks. It will nice paintings nicely for the music on which the dataset is created.

A. Augmenting Variations of Images

Images from the dataset are taken and versions of all pix which include zoomed photograph, brightness altered image and flipped photo are delivered to the dataset virtually to make the model more generalized.

*Corresponding author: muqeetwork@gmail.com

B. Preprocessing the Image

The snap shots inside the dataset are all RGB pix so for ease of schooling the model the images are transformed to YUV format. YUV color-areas are extra inexperienced coding and reduce the bandwidth extra than RGB capture can. The photos are also blurred the use of Gaussian blur characteristic of openCV and resized so that unimportant factors which include historical past surroundings are cropped out. Then every pixel is divided with the useful resource of 255 so that all pixels get same priority as pixels with excessive values get pointless precedence. Dividing through 255 will reduce all pixel values to zero or 1.

C. Training the model

The version is trained for a desired quantity of epochs so that its output will always try to remove bias in the model and make it efficient so that it can be generalized for any track when put to practical use.

D. Simulating the Model

We simulated our version using an Open-Source simulator that was made with Unity3D editor and we used Flask backend to establish a actual-time

communication the usage of socket, Io a JavaScript actual time comm library and had been capable of notice the tilt, attitude, coordinate and speed at every instantaneous.

E. Main principle behind

The basic idea of the monitoring is that the automobile is a continuous displacement movement procedure. During the advancement of the car, the change of the lane line is also a non-stop change. This change is meditated inside the slope of the lane line. The slope of the lane line within the frames of the front and rear snap shots aren't much distinctive from the position of the lane line. Therefore, the 2 frames earlier than and after the manipulate are as compared. The slope of the lane line within the center is limited close to the previously detected lane line vicinity. This is the primary concept of tracking. Finding lane strains inside the vicinity of hobby can greatly lessen the quantity of image processing.

F. Working

When the polar attitude of the lane line is in the detection location, the location of the lane line may be speedy and accurately detected. However, while the photo is shifted in a flip, lane exchange or camera role, the lane line without difficulty exceeds the detection area, in order that the outcomes look like a few deviations.

In conventional Hough transform, each factor needs to be traversed at every attitude, that's time ingesting. But the modified Hough remodel is used to carry out transformation at the vanishing point and the restricted pixels round it. And improve the real time performance of the set of rules.

We skilled a custom Convolutional Neural Network. It is a deep neural network. We claim the version to be sequential and add five Convolution2D, we additionally add 4 dropout layers and 4 dense layers to the community. An unmarried flatten layer is introduced to transform the picture matrix to a one-dimensional array. Built basically by means of following the given traits.

We trained a custom Convolutional Neural Network. It is a deep neural community. We claim the version to be sequential and upload 5 Convolution2D, we additionally add four dropout layers and four dense layers to the network. A single flatten layer is added to transform the photograph matrix to a one-dimensional array. Built essentially by means of following the given characteristics: We skilled a custom Convolutional Neural Network. It is a deep neural community. We declare the model to be sequential and add five Convolution2D, we additionally upload four dropout layers and four dense layers to the network. A unmarried flatten layer is delivered to transform the photograph matrix to a one-dimensional array. Built basically through following the given traits:

The Hough Transform (HT) is a virtual image processing approach for the detection of shapes which has more than one makes use of these days. A downside of this method is its sequential computational complexity, mainly while a single processor is used. An optimized set of rules of HT for immediately lines detection in a photo is presented in this article. Optimization is found out with the aid of the usage of a decomposition of the enter picture lately proposed via valuable processing unit (CPU), and the technique referred to as phase decomposition [12].

Why we are relying on Hough transformation?

Most roads on the road are basically instantly, and there are few sharp bends within the curve. Therefore, within the lane detection and monitoring, the Hough remodel is used to hit upon the road and decide the approximate role and shape of the lane. Then determine the deviation direction of the lane via the slope of the lane, after which locate the curve part of the lane. In this manner, the accuracy of the detection of the lane line may be ensured, and there are no critical errors within the detected curve.

The Hough line detection approach is accurate and simple, and the curve detection may be done after including the Vanishing factor monitoring algorithm.

A self-driving vehicle is a vehicle that is able to experience its environment and force without human intervention. Self-using automobiles can experience surroundings the use of many ways like lidar, radar, GPS, digital camera. Biggest advantage of self-driving motors is the decreased range of injuries. If such automobiles are broadly available and engineered well also can keep gas so one can cause lesser pollution. In current length there was incredible amount of development in self-using automobile space that's attracting wide variety of customers commonly because of the benefits it provides [13].

We can efficaciously create a vehicle that may demonstrate proper lane adjustments, parking, and U-activates its personal. The one-of-a-kind improvements that function are boundaries and shrink detection strategies, road car tracker, and checking special traffic conditions. This will make a strong autonomous self-driven car. It will efficiently demonstrate right parking allotment, lane modifications, and automated U-turns. We can do those the usage of the obstacle and diverse cut down detection technique, the automobile tracker. Autonomous vehicles have various advantages over guide motors like fewer site visitors' injuries, clever choice making just to name a few [14].

The maximum normally occurring failures included the failure to stumble on lanes and uncomfortable speed adjustments of the automobile. Additionally, a majority of the drivers emphasized the importance of being alert while driving with self-sustaining features and aware about the restrictions of the contemporary technology. Our fundamental contribution is to offer a photograph of attitudes and experiences in the direction of semi-self-sufficient driving, revealing that some drivers adopting these functions won't understand autonomous driving as unstable, even in an environment with ordinary automation failures [15].

For that depend, it's far essential to outline what exactly is known as a self-reliant automobile (AV) on this report. While the era may be tailored to a brilliant range of different vehicle kinds, here street-primarily based transport is considered specially, protecting diversifications of self-riding technology to personal automobiles and public delivery answers. In that regard the phrases "self-sufficient", "self-riding" and "driver-much less" are used interchangeably as is frequently the case in the current literature in difference to CVs (traditional vehicles). Furthermore, it needs to be defined on which time scope the evaluation addresses. While the 11 technological improvement is already making development, the adoption of self-using automobiles is just beginning. There are lots of situations on how the direction toward big-scale use of AVs will appear like, with technological, societal, prison, and economic barriers [15].

G. Deep Learning

Deep studying is a branch of computer mastering that teaches pc systems to do what comes evidently to human's look at from enjoy. Machine analyzing algorithms use computational techniques to "learn" records right away from records besides counting on a predetermined equation's as a model. Deep gaining knowledge of is specifically appropriate for photo reputation, that is vital for fixing troubles such as facial recognition, movement detection, and many advanced drivers helps applied sciences including self-reliant using, lane detection, pedestrian detection, and self-reliant parking. Deep reading makes use of neural networks to study beneficial representations of points without delay from information. Neural networks blend multiple nonlinear processing layers, the use of clean factors running in parallel and stimulated by using organic disturbing structures.

Unity3D Gaming Engine:

Unity is a move-platform recreation engine developed with the aid of Unity Technologies, first introduced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as a Mac OS X- unique sport engine. As of 2018, the engine was prolonged to assist greater than 25 platforms.

Unity3D is a powerful go-platform 3-D engine and a user-pleasant development surroundings. Easy enough for the newbie and powerful sufficient for the expert; Unity ought to interest anybody who wants to without difficulty create 3-d video games and packages for cell, computing device, the web, and consoles.

The technology of the surroundings in. Xodr is completed using a unfastened utility referred to as OpenRoadEd. Different parts of the roads consisting of straight roads, arcs and spirals are generated using the street settings panel. Later, the roads are linked to each other to shape the entire tune. In this part, the definition of the surroundings may be additionally executed using the available textures or by way of adding the desired textures to the library of snap shots. By saving the geometry, the generated surroundings is transformed routinely to. Xodr layout.

Supervised Learning:

Given a area X that yields a illustration of the picture area containing the photos from the harmony framework. We bear in mind a finite subset of training examples XTrain = x(1), ..., x(n) ⊆ X. Let similarly y (i) be the corresponding ground truth statistics with recognize to x(i), e.G., the guidance angle, brake and throttle. For supervised mastering, the model Mw is skilled with the education set T = (x(i), y (i)) which includes the schooling examples and the corresponding ground fact information. Artificial Neural Networks Artificial Neural Networks (ANNs) are a collection of models in Machine Learning, that are inspired by means of the shape and features of biological neural networks [13]. The principal neural machine inside the mammals' mind consists of a huge number of neurons, which are strongly interconnected. With the assist of this shape, facts are processed and evaluated throughout neurons with the aid of electrochemical interactions. The ANNs are then a try at imposing a comparable way of facts processing on computer systems. We illustrate beneath the inner shape of an artificial neuron, that's the constitutive unit of ANNs. The synthetic neuron gets one or extra inputs xi. All inputs are then increased via corresponding weights wi and summed up together with a threshold time period θ referred to as bias. After that, this input is surpassed thru a non-linear function φ referred to as an activation characteristic to produce a sure output y. We summarize the whole operation carried out via an artificial neuron inside the following method:

$$y = \varphi\left(\sum xi \cdot wi + \theta\right)$$

We can represent the ANN as a version Mw, this is composed of learnable weights w, and activation capabilities φ.

H. Image Processing

Image processing is the approach to transform a picture into digital structure and characteristic operations on it to get an more photograph or extract some useful records from it. Changes that take place in pix are commonly done mechanically and anticipate cautiously designed algorithms. Image processing is a multi-discipliner's field, with contribution from specific branches of technology which all consists of mathematics, physics, optical and electrical engineering.

Flask and Socket IO:

The version is communicated with the simulator through a Flask backend and suing a real time conversation channel hooked up using socket io.

3. Drawbacks in Existing System

Processing power:

First of all, in view that deep mastering requires the sort of excessive stage of computing electricity, a completely powerful "brain" is wanted to address the large facts abilities and processing requirements. Currently, the maximum suitable generation is the so-known as GPU (graphical processing unit), in view that it is designed to deal with heavy picture processing duties (known from for example the pc gaming industry). Currently the groups Nvidia and Intel are on their manner to put themselves as leaders offering the "brains" for the shrewd automobile market. However, it is nonetheless an assignment to have a low-fee GPU that operates within the energy intake and other boundaries, together with heat control, this is required for a marketplace-ready automobile. Moreover, agencies still war with bandwidth and synchronization problems.

Available training facts:

As referred to before, an quit-to-give up gaining knowledge of device especially, requires to be fed a huge quantity of training statistics, so one can expect as many driving situations as feasible and to fulfil a minimum safety requirement. Some claim that at the least one thousand million kilometers of training statistics from sensible street situations are wanted in an effort to make conclusions about safety of the car. Not handiest that, the statistics needs to be numerous enough for it to be beneficial (using one kilometer 1000000000 times back and forth received do the task!)

Safety:

One of the primary demanding situations with protection of deep neural networks is the truth that they are risky under so-known as antagonistic perturbations. For example, minimal modifications in camera snap shots, such are resizing, cropping and the alternate of lighting situations would possibly motive the device to misclassify the image. Additionally, in standard, protection assurance and verification techniques for gadget learning are poorly studied. The winning car safety trendy of ISO26262, does no longer have a manner to define safety for self-studying algorithms which include deep mastering. Hence, there is nonetheless no manner to standardize the protection component yet, because of the fast tempo of contemporary generation. A prominent example of a safety failure is the 2016 Tesla vehicle-pilot accident, where the sensors of the vehicle have been mixed by the sun and the gadget failed to recognize the truck coming from the proper, leading to the crash [9]. This shows that loads still wish to be investigated earlier than we will finish that the modern configuration of a (partly) self-riding car is secure.

Available training facts:

As referred to before, a quit-to-give up gaining knowledge of device especially, requires to be fed a huge quantity of training statistics, so one can expect as many driving situations as feasible and to fulfil a minimum safety requirement. Some claim that at the least one thousand million kilometers of training statistics from sensible street situations are wanted in an effort to make conclusions about safety of the car. Not handiest that, the statistics needs to be numerous enough for it to be beneficial (using one kilometer 1000000000 times back and forth received do the task!).

4. Proposed System

Neural Network Regression Algorithms:

This kind of algorithm is good at predicting events. The Regression Analysis evaluates the relation between 2 or more variables and collate the effects of variables on distinct scales and are driven mostly by 3 metrics:

- The shape of the regression line.
- The type of dependent variables.
- The number of independent variables.

The images (camera or radar) play a significant role in ADAS in actuation and localization, while for any algorithm, the biggest challenge is to develop an image-based model for feature selection and prediction. The repeatability of the environment is leveraged by regression algorithms to create a statistical model of relation between the given object's position in an image and that image. The statistical model, by allowing the image sampling, provides fast online detection and can be learned offline. It can be extended furthermore to other objects without the requirement of extensive human modeling. An object's position is returned by an algorithm as the online stage's output and a trust on the object's presence. The regression algorithms can also be utilized for short prediction, long learning. This kind of regression algorithms that can be utilized for self-driving cars are decision forest regression, neural network regression and Bayesian regression, among others. The neural networks are utilized for regression, classification or unsupervised learning. They group the data that is not labeled, classify that data or forecast continuous values after supervised training. The neural networks normally use a form of logistic regression in the final layer of the net to change continuous data into variables like 1 or 0.

We will be using Google Collaboratory to write our code in Python3 for this project. Google Collaboratory is a free cloud platform where we can write codes and it also supports GPU which makes it a lot faster than any other expensive PC or laptop without buying it thus making it suitable for Artificial Intelligence and Machine learning

Advantages of Proposed System:

- Accuracy will be high
- Picture quality will not be compressed.
- Will be able to rain the car and use the model in any generalized environment.

Algorithms and Approach:

The machine learning algorithms are loosely divided into 4 classes: decision matrix algorithms, cluster algorithms, pattern recognition algorithms and regression algorithms. One category of the machine learning algorithms can be utilized to accomplish 2 or more subtasks. For instance, the regression algorithms can be utilized for object localization as well as object detection or prediction of the movement.

Decision Matrix Algorithms:

The decision matrix algorithm systematically analyzes, identifies and rates the performance of relationships between the sets of information and values. These algorithms are majorly utilized for decision making. Whether a car needs to brake or take a left turn is based on the level of confidence these algorithms have on recognition, classification and prediction of the next movement of objects. The decision matrix algorithms are models composed of various decision models trained independently and, in some way, these predictions are combined to make the overall prediction, while decreasing the possibility of errors in decision making. AdaBoosting is the most commonly used algorithm.

AdaBoosting:

Adaptive Boosting or AdaBoost is a combination of multiple learning algorithms that can be utilized for regression or classification. It overcomes overfitting when compared with any other machine learning algorithms and is often sensitive to outliers and noisy data. In order to create one composite powerful learner, AdaBoost uses multiple iterations. So, it is termed as adaptive. By adding the weak learners iteratively, AdaBoost creates a strong learner. A new weak learner is appended to the entity and a weighing vector is adjusted in order to pay attention on examples that were classified incorrectly in the prior rounds. A classifier that has much higher accuracy than the classifiers of weak learners is the result.

AdaBoost helps in boosting the weak threshold classifier to strong classifier. The above image depicts the implementation of AdaBoost in one single file with understandable code. The function contains a weak classifier and the boosting component. The weak classifier attempts to locate the ideal threshold in one of data dimensions to segregate the data into 2 classes. The classifier is called by the boosting part iteratively and after each classification step, it changes the weights of misclassified examples. Because of this, a cascade of weak classifiers is created and it behaves like a strong classifier.

Clustering Algorithms:

Sometimes, the images acquired by the system are not clear and it becomes difficult to locate and detect objects. Sometimes, there is a possibility of classification algorithms missing the object and, in that case, they fail to categorize and report it to the system. The possible reason could be discontinuous data, very few data points or low-resolution images. The clustering algorithm is specialized in discovering the structure from data points. It describes the class of methods and class of problem like regression. The clustering methods are organized typically by modeling the approaches like hierarchical and centroid-based. All methods are concerned with utilizing the inherent structures in data to organize the data perfectly into groups of maximal commonalities.

K- means, Multi-class Neural Network is the most commonly used algorithm.

K-Means:

K-means Algorithm – The cluster centroids are depicted as crosses and training examples are depicted as dots. (a) Original dataset. (b) Random initial cluster centroids. (c-f) The demonstration of running 2 iterations of k-means. Each training example is assigned in each iteration to the cluster centroid that is closest and then, each cluster centroid is moved to mean of points assigned to it.

Pattern Recognition Algorithms (Classification):

The images obtained through sensors in Advanced Driver Assistance Systems (ADAS) consists of all kinds of environmental data; filtering of the images is needed to determine the instances of an object category by ruling out the data points that are irrelevant. Before classifying the objects, the recognition of patterns is an important step in a dataset. This kind of algorithms are defined as data reduction algorithms. The data reduction algorithms are helpful in reducing the dataset edges and polylines (fitting line segments) of an object as well as circular arcs to edges. Till a corner, the line segments are aligned with the edges and a new line segment will begin after this. The circular arcs align with the line segments' sequences that is similar to an arc. In various ways, the features of the image (circular arcs and line segments) are combined to form the features that are utilized for determining an object. With the PCA (Principal Component Analysis) and HOG (Histograms of Oriented Gradients), the SVM (Support Vector Machines) are the commonly used recognition algorithms in ADAS. The K nearest neighbor (KNN) and Bayes decision rule are also used.

Support Vector Machines (SVM):

SVM are dependent on the decision planes concept that define the decision boundaries. The decision plane separates the object set consisting of distinct class memberships. A schematic example is illustrated below. In this, the objects belong to either RED or GREEN class. A boundary line of separation separates the RED and GREEN objects. Any new object that falls to the left is labeled as RED and it is labeled as GREEN if it falls to the left.

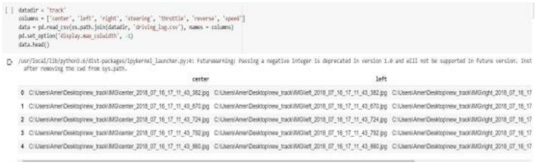
We will be training our model using the algorithms mentioned below:

Regression Algorithms:

This kind of algorithm is good at predicting events. The Regression Analysis evaluates the relation between 2 or more variables and collate the effects of variables on distinct scales and are driven mostly by 3 metrics: The shape of regression line. The type of dependent variables. The number of independent variables. The images (camera or radar) play a significant role in ADAS in actuation and localization, while for any algorithm, the biggest challenge is to develop an image-based model for feature selection and prediction. The repeatability of the environment is leveraged by regression algorithms to create a statistical model of relation between the given object's position in an image and that image. The statistical model, by allowing the image sampling, provides fast online detection and can be learned offline. It can be extended furthermore to other objects without the requirement of extensive human modeling. An object's position is returned by an algorithm as the online stage's output and a trust on the object's presence. The regression algorithms can also be utilized for short prediction, long learning. This kind of regression algorithms that can be utilized for self-driving cars are decision forest regression, neural network regression and Bayesian regression, among others.

Neural Network Regression:

The neural networks are utilized for regression, classification or unsupervised learning. They group the data that is not labeled, classify that data or forecast continuous values after supervised training. The neural networks normally use a form of logistic regression in the final layer of the net to change continuous data into variables like 1 or 0.



```
[ ] datadir = 'track'
    columns = ['center', 'left', 'right', 'steering', 'throttle', 'reverse', 'speed']
    data = pd.read_csv(os.path.join(datadir, 'driving_log.csv'), names = columns)
    pd.set_option('display.max_colwidth', -1)
    data.head()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: FutureWarning: Passing a negative integer is deprecated in version 1.0 and will not be supported in future version. Inst
  after removing the cwd from sys.path.
                                  center                                                            left
0 C:\Users\Amer\Desktop\new_track\IMG\center_2018_07_16_17_11_43_382.jpg C:\Users\Amer\Desktop\new_track\IMG\left_2018_07_16_17_11_43_382.jpg C:\Users\Amer\Desktop\new_track\IMG\right_2018_07_16_17
1 C:\Users\Amer\Desktop\new_track\IMG\center_2018_07_16_17_11_43_670.jpg C:\Users\Amer\Desktop\new_track\IMG\left_2018_07_16_17_11_43_670.jpg C:\Users\Amer\Desktop\new_track\IMG\right_2018_07_16_17
2 C:\Users\Amer\Desktop\new_track\IMG\center_2018_07_16_17_11_43_724.jpg C:\Users\Amer\Desktop\new_track\IMG\left_2018_07_16_17_11_43_724.jpg C:\Users\Amer\Desktop\new_track\IMG\right_2018_07_16_17
3 C:\Users\Amer\Desktop\new_track\IMG\center_2018_07_16_17_11_43_792.jpg C:\Users\Amer\Desktop\new_track\IMG\left_2018_07_16_17_11_43_792.jpg C:\Users\Amer\Desktop\new_track\IMG\right_2018_07_16_17
4 C:\Users\Amer\Desktop\new_track\IMG\center_2018_07_16_17_11_43_860.jpg C:\Users\Amer\Desktop\new_track\IMG\left_2018_07_16_17_11_43_860.jpg C:\Users\Amer\Desktop\new_track\IMG\right_2018_07_16_17
```

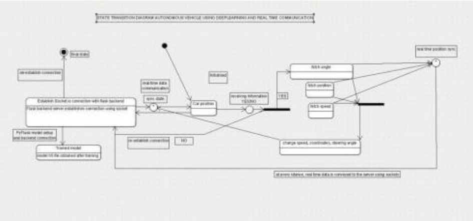
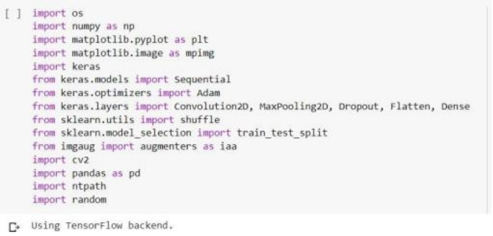
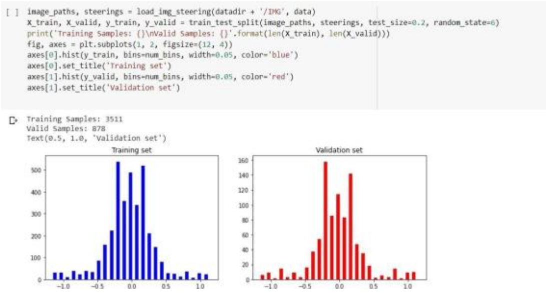

STATE TRANSITION DIAGRAM AUTONOMOUS VEHICLE USING DEEPLEARNING AND REAL TIME COMMUNICATION
final state
de-establish connection
real time data communication
sync state
Establish Socket io connection with flask backend
Flask backend server establishes connection using socket
PyFlask model setup and backend connection
Trained model
model h5 file obtained after training
re establish connection
NO
Car position
initialized
receiving information YES/NO
YES
fetch angle
fetch position
fetch speed
change speed, coordinates, steering angle
real time position sync
at every interval, real time data is conveyed to the server using sockets

Fig. 1. Sequence diagram



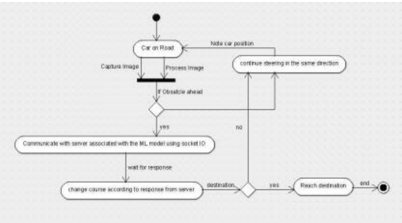
```
[ ] import os
    import numpy as np
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    import keras
    from keras.models import Sequential
    from keras.optimizers import Adam
    from keras.layers import Convolution2D, MaxPooling2D, Dropout, Flatten, Dense
    from sklearn.utils import shuffle
    from sklearn.model_selection import train_test_split
    from imgaug import augmenters as iaa
    import cv2
    import pandas as pd
    import ntpath
    import random
```

```
Using TensorFlow backend.
```

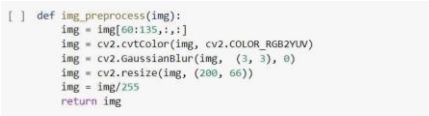


```
[ ] image_paths, steerings = load_img_steering(datadir + '/IMG', data)
    X_train, X_valid, y_train, y_valid = train_test_split(image_paths, steerings, test_size=0.2, random_state=6)
    print('Training Samples: {}\nValid Samples: {}'.format(len(X_train), len(X_valid)))
    fig, axes = plt.subplots(1, 2, figsize=(12, 4))
    axes[0].hist(y_train, bins=num_bins, width=0.05, color='blue')
    axes[0].set_title('Training set')
    axes[1].hist(y_valid, bins=num_bins, width=0.05, color='red')
    axes[1].set_title('Validation set')
```

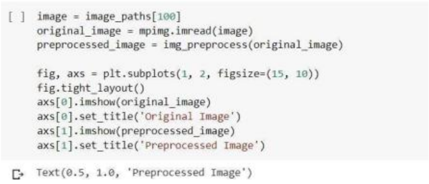
```
Training Samples: 3511
Valid Samples: 878
Text(0.5, 1.0, 'Validation set')
```


Car on Road
Note car position
Capture Image
Process Image
continue steering in the same direction
If Obsatcle ahead
yes
no
Communicate with server associated with the ML model using socket IO
wait for response
change course according to response from server
destination
yes
Reach destination
end

Sample code:

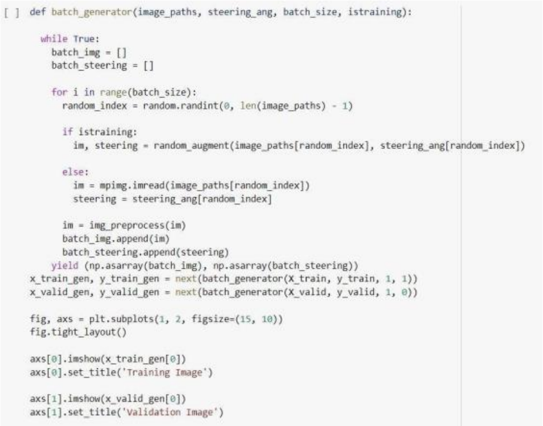


```
[ ] def img_preprocess(img):
        img = img[60:135,:,:]
        img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
        img = cv2.GaussianBlur(img,  (3, 3), 0)
        img = cv2.resize(img, (200, 66))
        img = img/255
        return img
```



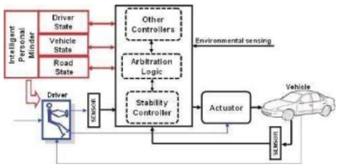
```
[ ] image = image_paths[100]
    original_image = mpimg.imread(image)
    preprocessed_image = img_preprocess(original_image)

    fig, axs = plt.subplots(1, 2, figsize=(15, 10))
    fig.tight_layout()
    axs[0].imshow(original_image)
    axs[0].set_title('Original Image')
    axs[1].imshow(preprocessed_image)
    axs[1].set_title('Preprocessed Image')
```

```
Text(0.5, 1.0, 'Preprocessed Image')
```



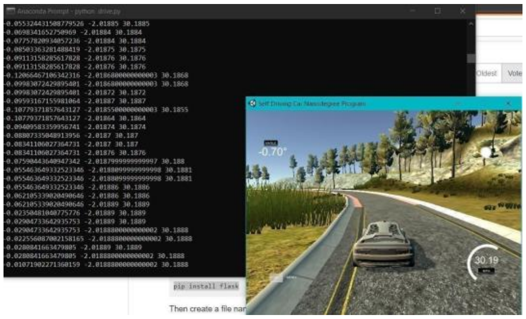
```
[ ] def batch_generator(image_paths, steering_ang, batch_size, istraining):

      while True:
        batch_img = []
        batch_steering = []

        for i in range(batch_size):
          random_index = random.randint(0, len(image_paths) - 1)

          if istraining:
            im, steering = random_augment(image_paths[random_index], steering_ang[random_index])

          else:
            im = mpimg.imread(image_paths[random_index])
            steering = steering_ang[random_index]

          im = img_preprocess(im)
          batch_img.append(im)
          batch_steering.append(steering)
        yield (np.asarray(batch_img), np.asarray(batch_steering))
    x_train_gen, y_train_gen = next(batch_generator(X_train, y_train, 1, 1))
    x_valid_gen, y_valid_gen = next(batch_generator(X_valid, y_valid, 1, 0))

    fig, axs = plt.subplots(1, 2, figsize=(15, 10))
    fig.tight_layout()

    axs[0].imshow(x_train_gen[0])
    axs[0].set_title('Training Image')

    axs[1].imshow(x_valid_gen[0])
    axs[1].set_title('Validation Image')
```

Batch generator:



5. Results

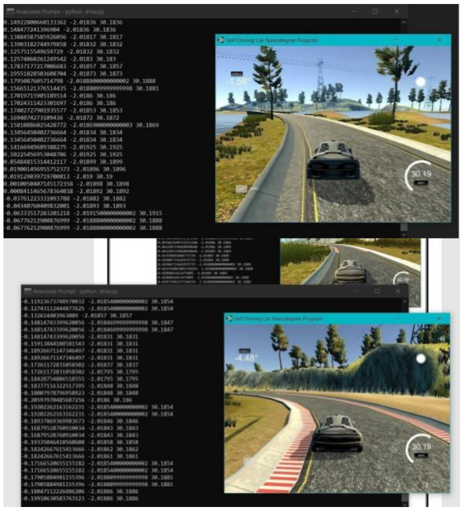
The plot of Loss vs. number of epochs shows that there is a sufficient reduction of training and validation loss after 10 epochs. The training loss was obtained as 0.0343 and the validation loss as 0.0275. This proves that our model can also be tested on any simulator track other than the one we used for training.


Anaconda Prompt - python drive.py
Self Driving Car Nanodegree Program
-0.70°
30.19
Oldest
Vote

```
-0.05532443150879526 -2.01885 30.1885
-0.0698341652750969 -2.01884 30.1884
-0.07757820934057236 -2.01884 30.1884
-0.08503363281488419 -2.01875 30.1875
-0.09113158285617828 -2.01876 30.1876
-0.09113158285617828 -2.01876 30.1876
-0.12066467106342316 -2.0186800000000003 30.1868
-0.09983072429895401 -2.0186800000000003 30.1868
-0.09983072429895401 -2.01872 30.1872
-0.09593167155981064 -2.01887 30.1887
-0.10779371857643127 -2.0185500000000003 30.1855
-0.10779371857643127 -2.01864 30.1864
-0.09409583359956741 -2.01874 30.1874
-0.08807335848913956 -2.0187 30.187
-0.08341106027364731 -2.0187 30.187
-0.08341106027364731 -2.01876 30.1876
-0.07598443640947342 -2.0187999999999997 30.188
-0.05546364933252346 -2.0188099999999998 30.1881
-0.05546364933252346 -2.0188099999999998 30.1881
-0.05546364933252346 -2.01886 30.1886
-0.06210533902049064б -2.01886 30.1886
-0.06210533902049064б -2.01889 30.1889
-0.02350481040775776 -2.01889 30.1889
-0.02904733642935753 -2.01889 30.1889
-0.02904733642935753 -2.0188800000000002 30.1888
-0.02255608700215816 -2.0188800000000002 30.1888
-0.02888416634798805 -2.01889 30.1889
-0.02888416634798805 -2.0188800000000002 30.1888
-0.01071902271360159 -2.0188800000000002 30.1888
```

```
pip install flask
```

Then create a file nar

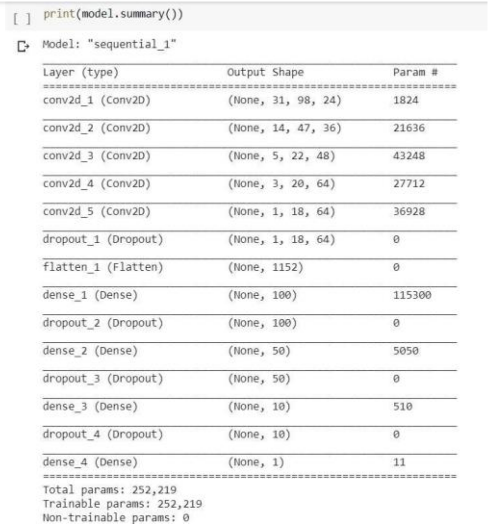

Anaconda Prompt - python drive.py
Self Driving Car Nanodegree Program
30.19
Anaconda Prompt - python drive.py
Self Driving Car Nanodegree Program
-4.48°
30.19

```
0.14922800660133362 -2.01836 30.1836
0.148477241396904 -2.01836 30.1836
0.13884587585926056 -2.01817 30.1817
0.13903182744979858 -2.01832 30.1832
0.1257515549659729 -2.01832 30.1832
0.12574060261249542 -2.0183 30.183
0.17837177217006683 -2.01857 30.1857
0.19551828503608704 -2.01873 30.1873
0.1795087605714798 -2.0188800000000002 30.1888
0.15665121376514435 -2.0188099999999998 30.1881
0.17019715905189514 -2.0186 30.186
0.17024311423301697 -2.0186 30.186
0.17402727901935577 -2.01853 30.1853
0.1694074273109436 -2.01872 30.1872
0.15810886025428772 -2.0186900000000003 30.1869
0.13456450402736664 -2.01834 30.1834
0.13456450402736664 -2.01834 30.1834
0.14166949689388275 -2.01925 30.1925
0.10225456953048706 -2.01925 30.1925
0.05484815314412117 -2.01899 30.1899
0.01900149695575237З -2.01896 30.1896
0.01912903971970081З -2.019 30.19
0.001005040714517235８ -2.01898 30.1898
0.000841146567836403８ -2.01892 30.1892
-0.03761223331093788 -2.01882 30.1882
-0.04340760409832001 -2.01893 30.1893
-0.06333517283201218 -2.0191500000000002 30.1915
-0.06776212900876999 -2.0188800000000002 30.1888
-0.06776212900876999 -2.0188800000000002 30.1888
```

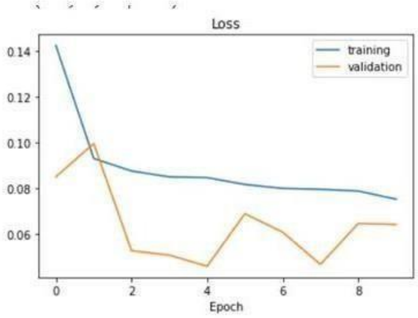
```
-0.11923673748970032 -2.0185400000000002 30.1854
-0.12743112444877625 -2.0185400000000002 30.1854
-0.132614403963089 -2.01857 30.1857
-0.1481474339962005б -2.0184699999999998 30.1847
-0.1481474339962005б -2.0184699999999998 30.1847
-0.1481474339962005б -2.01831 30.1831
-0.15913844100581543 -2.01831 30.1831
-0.18926671147346497 -2.01831 30.1831
-0.18926671147346497 -2.01831 30.1831
-0.17261172831058502 -2.01837 30.1837
-0.17261172831058502 -2.01795 30.1795
-0.18428754806518555 -2.01795 30.1795
-0.18377116322517395 -2.01848 30.1848
-0.18007978796958923 -2.01848 30.1848
-0.20597970485687256 -2.0186 30.186
-0.19202262163162231 -2.0185400000000002 30.1854
-0.19202262163162231 -2.0185400000000002 30.1854
-0.18937869369983673 -2.01846 30.1846
-0.16879528760910034 -2.01843 30.1843
-0.16879528760910034 -2.01843 30.1843
-0.19335046410560608 -2.01858 30.1858
-0.18242667615413666 -2.01862 30.1862
-0.18242667615413666 -2.01861 30.1861
-0.17166520655155182 -2.0185400000000002 30.1854
-0.17166520655155182 -2.0185400000000002 30.1854
-0.17905884981155396 -2.0188099999999998 30.1881
-0.17905884981155396 -2.0188099999999998 30.1881
-0.18847112226486206 -2.01886 30.1886
-0.19910630583763123 -2.01886 30.1886
```

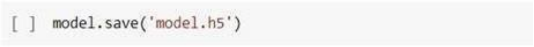
Drive Coordinates are plotted in real time (-) indicates left turn and (+) right turn:

```
-0.0488455705344677 -2.0188300000000003 30.1883
-0.0488455705344677 -2.01879 30.1879
-0.06489957123994827 -2.01879 30.1879
-0.05687129124999046 -2.0187999999999997 30.188
```



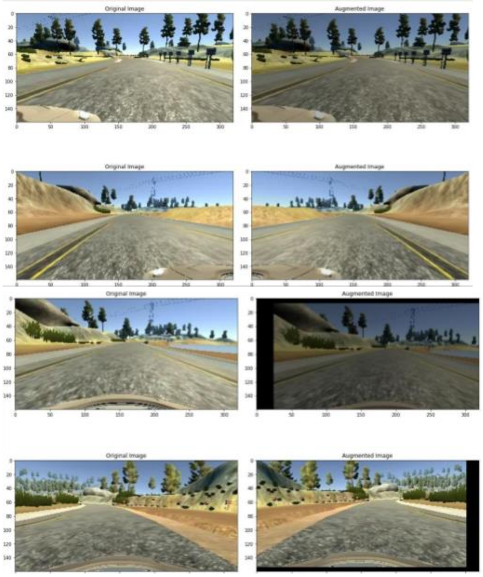
```
[ ] print(model.summary())

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 18, 64)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               115300
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_3 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dropout_4 (Dropout)          (None, 10)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
```





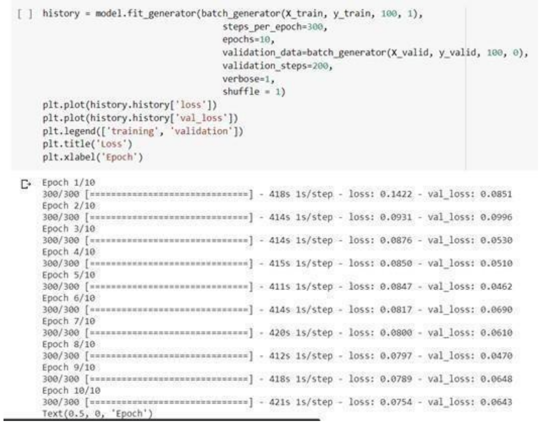
```
[ ] model.save('model.h5')
```

Augmenting variations of images:



```
-0.05687129124999046 -2.0187999999999997 30.188
-0.05891359969973564 -2.0188099999999998 30.1881
-0.0599889270961284б4 -2.01875 30.1875
-0.07196944206953049 -2.01878 30.1878
-0.06476251780986786 -2.0188099999999998 30.1881
-0.05087998881936073 -2.0187999999999997 30.188
-0.05087998881936073 -2.01878 30.1878
127.0.0.1 - - [06/Jun/2020 07:23:57] "GET /socket.io/?EIO=4&transport=websocket HTTP/1.1" 200 0 100.167020
```

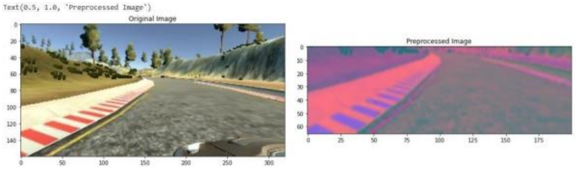
The images in the dataset are all RGB images so for ease of training the model the images are converted to YUV format. YUV color-spaces are a more efficient coding and reduce the bandwidth more than RGB capture can. The images are also blurred using Gaussian blur function of openCV and resized so that unimportant parts such as background scenery are cropped out. Then each pixel is divided by 255 so that all pixels get equal priority as pixels with high values get unnecessary priority. Dividing by 255 will reduce all pixel values to 0 or 1.



```
[ ] history = model.fit_generator(batch_generator(X_train, y_train, 100, 1),
                                  steps_per_epoch=300,
                                  epochs=10,
                                  validation_data=batch_generator(X_valid, y_valid, 100, 0),
                                  validation_steps=200,
                                  verbose=1,
                                  shuffle = 1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.legend(['training', 'validation'])
    plt.title('Loss')
    plt.xlabel('Epoch')

Epoch 1/10
300/300 [==============================] - 418s 1s/step - loss: 0.1422 - val_loss: 0.0851
Epoch 2/10
300/300 [==============================] - 414s 1s/step - loss: 0.0931 - val_loss: 0.0996
Epoch 3/10
300/300 [==============================] - 414s 1s/step - loss: 0.0876 - val_loss: 0.0530
Epoch 4/10
300/300 [==============================] - 415s 1s/step - loss: 0.0850 - val_loss: 0.0510
Epoch 5/10
300/300 [==============================] - 411s 1s/step - loss: 0.0847 - val_loss: 0.0462
Epoch 6/10
300/300 [==============================] - 414s 1s/step - loss: 0.0817 - val_loss: 0.0690
Epoch 7/10
300/300 [==============================] - 420s 1s/step - loss: 0.0800 - val_loss: 0.0610
Epoch 8/10
300/300 [==============================] - 412s 1s/step - loss: 0.0797 - val_loss: 0.0470
Epoch 9/10
300/300 [==============================] - 418s 1s/step - loss: 0.0789 - val_loss: 0.0648
Epoch 10/10
300/300 [==============================] - 421s 1s/step - loss: 0.0754 - val_loss: 0.0643
Text(0.5, 0, 'Epoch')
```

Connecting the model and establishing real time communication using socketIO and Flask web server:

1. We established a flask web server using SocketIO and PyFlask.
2. The simulator was connected to the server using Sockets.
3. Real time data was communicated between the simulation environment and the web server.
4. The machine learning model imported as model.h5



Text(0.5, 1.0, 'Preprocessed Image')

6. Conclusion

This project is a proposed model to attain driverless cars. Research is still going on; we hope that this simulated model will be incorporated as a software in real life cars in the near future. Research is still going on and millions of data scientists and artificial intelligence scientists are working in order to convert this software model into real life application so that cars can become driverless and there would be lesser rule breaks and hopefully lesser road mishaps. Further, the project can be extended and used with different training models to increase the accuracy of detecting turns by the car. We obtained a stable model which when simulated travelled at a max speed of 30km/h on the track, also the angle of tilt was varying between positive and negative for right and left turns respectively and also made sure that it avoided all possible collisions and we were also able to generate positional and angular data at every point in time based on the polar and Cartesian coordinates of the car in the simulated environment. In this way we were able to obtain data in a CSV format which can later be used for self-driving research and conducting various analytics experiments in future.

References

[1] Morris B, Doshi A, Trivedi M. Lane change intent prediction for driver assistance: On-road design and evaluation, Intelligent Vehicles Symposium. IEEE, 2011:895-901.

[2] = Paula M B D, Jung C R. Real-Time Detection and Classification of Road Lane Markings, XXVI Conference on Graphics, Patterns and Images. IEEE Computer Society, 2013:83-90.

[3] Kaur G, Kumar D, Kaur G, et al. Lane Detection Techniques: A Review. International Journal of Computer Applications, 2015, 112(10):4-8.

[4] Dorum O. H, Lynch J. D, Gnedin M. Creating geometry for advanced driver assistance systems: US, US8762046[P]. 2014.

[5] Bottazzi V S, Borges P V K, Stantic B, et al. Adaptive Regions of Interest Based on HSV Histograms for Lane Marks Detection, Robot Intelligence Technology and Applications. Springer International Publishing, 2014:677-687.

[6] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, Lipika Deka,Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions,Transportation Research Part C:Emerging Technologies, Volume 60, 2015, Pages 416-442.

[7] Weiwei Chen, Weixing Wang, Kevin Wang, Zhaoying Li, Huan Li, Sheng Liu, Lane departure warning systems and lane line detection methods based on image processing and semantic segmentation: A review, Journal of Traffic and Transportation Engineering (English Edition), Volume 7, Issue 6, 2020, Pages 748-774.

[8] Narote, Sandipan & Bhujbal, Pradnya & Narote, Abbhilasha & Dhane, Dhiraj. (2017). A Review of Recent Advances in Lane Detection and Departure Warning System. Pattern Recognition. 73.

[9] Duda RO, Hart PE (1975) Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 15(1):11–15.

[10] Muhamad Lazim Talib, Xio Rui, Kamarul Hawari Ghazali, Norulzahrah Mohd. Zainudin, and Suzaimah Ramli. Comparison of Edge Detection Technique for Lane Analysis by Improved Hough Transform.

[11] Satzoda RK, Suchitra S, Srikanthan T (2008). Parallelizing the Hough transform computation. IEEE Signal Process Lett 15:297–300.

[12] Yam-Uicab, R. & López-Martínez, José & Trejo-Sánchez, Joel & Hidalgo-Silva, Hugo & Gonzalez, Sergio. (2017). A fast Hough Transform algorithm for straight lines detection in an image using GPU parallel computing with CUDA-C. The Journal of Supercomputing. 73. 1-20.

[13] Heejun Choi, Travis Teague, and Tanner Luce. 2020. Positive cycle of integrating teaching and research: machine learning self-driving car. J. Comput. Sci. Coll. 35, 7 (April 2020), 74–87.

[14] Rao, Qing & Frtunikj, Jelena. (2018). Deep learning for self-driving cars: chances and challenges. 35-38.

[15] Murat Dikmen and Catherine M. Burns. 2016. Autonomous Driving in the Real World: Experiences with Tesla Autopilot and Summon. In Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications. Association for Computing Machinery, New York, NY, USA, 225–228.