MANIPAL ACADEMY *of* HIGHER EDUCATION

*(Deemed to be University under Section 3 of the UGC Act, 1956)*

**MANIPAL SCHOOL OF INFORMATION SCIENCES**

**(A Constituent unit of MAHE, Manipal)**

# Road Classification using Image Processing and Machine Learning

| Reg. Number | Name | Branch |
|---|---|---|
| 221039017 | Advait Shirvaikar | Embedded Systems |
| 220139025 | Ananth Pai J | Embedded Systems |

**Under the guidance of**

**Dr. Mohan Kumar J**
Associate Professor,
Manipal School of Information Sciences,
MAHE, MANIPAL

**22/12/2022**

MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
*(A constituent unit of MAHE, Manipal)*

## Contents

## LIST OF FIGURES

# Chapter 1
# INTRODUCTION

An autonomous car is a vehicle capable of sensing its environment and operating without human involvement. A human passenger is not required to take control of the vehicle at any time, nor is a human passenger required to be present in the vehicle at all. An autonomous car can go anywhere a traditional car goes and do everything that an experienced human driver does.

Detection of road is one of the main features that incorporates in a self-driving car. Road recognition is one of the major topics of research now a days because of machine learning and computer vision-based researches. Existing road recognition research majorly focuses on structured road with road lane, and very few recognition models for unstructured roads. In general, structured roads are identified by lanes, or a road bounded by edge line pavement markings. Thus, a structured road maybe a single lane road, or might have multiple lanes, all bounded by these pavement markings. Unstructured roads are the roads which does not have a proper lane markings and boundaries. On the other hand, they might have multiple terrains having sand, gravel, stones, etc.

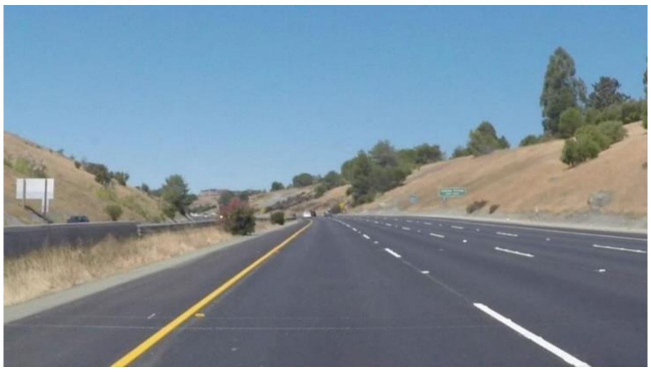


**Figure 1.1** Image of a structured road

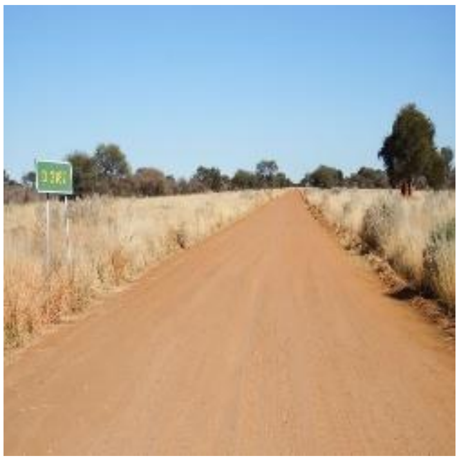


**Figure 1.2** Image of an unstructured road

Based on this knowledge, we adopted a binary classification problem approach. The methodology that we followed to classify the roads consisted of detecting the edge line pavement markings on the road, and if found, classifying that road as a structured road. Similarly, if no lines were found, we classified them as unstructured roads.

To detect these lanes, and classify the road images into the two categories, the images are required to be in a certain format. This is accomplished using OpenCV and python. We used multiple pre-processing techniques such as thresholding, Sobel filtering, perspective warps, Hough lines, Gaussian filtering, Canny edge detection, etc. For object detection, we used the sliding window algorithm. Further, machine learning models will also be used, to classify roads into the two categories.

# Chapter 2
# LITERATURE SURVEY

## 2.1 Analysis of Lane Detection Using OpenCV

Authors have used Canny and Sobel and implemented lane detection techniques. In the lane detection techniques, they have used a camera mounted in front of the vehicle. On the basis of Receiver Operating Characteristic (ROC) curve and Detection Error Trade-off (DET) curve performance of these lane detection techniques has been measured. These curves are the standard parameters for assessing the performance of an algorithm in Computer vision. Confusion matrix has been used for finding FP, FN, TP and TN and on the basis of confusion matrix FP rate, FN rate, TP rate, TN rates have been calculated. Finally, DET and ROC curves have been plotted on the basis of confusion matrix using MATLAB and performance of each method is analysed.

They faced challenges like,

- Different lighting circumstances:

  Changing illumination on road during morning, noon and evening time is a major challenge for lane detection.
- Different weather circumstances:

  Different types of weather include cloudy, windy, rainy, sunny, stormy, foggy and peaceful weather. The weather condition is rarely constant for a given day and varies from time to time.
- Curved roads:

  If some or all part of road gradually diverges in spite of being straight, then it is called curved road. Lane markings also follow the same curve on the road.
- Roads without Lane mark (Unstructured Roads):

  If the road is not having any lane marking, then it is called unstructured road. Lane detection for these roads requires a more robust and effective technique.
- Preceding vehicle, Shadow of tree:

  Due to shadow of the tree on the road and preceding vehicle it is difficult to detect the lane as it changes the illumination and images captured.
- Blockage of visibility:

While travelling, blockage of visibility due to fog, preceding vehicle is also a challenge for lane detection. This blinds the camera sometimes or restricts its view.

- Different colour:

  Due to shadow, day light, evening, colour information of the road is also affected which changes the colour of images getting captured.

- Different texture:

  The variation in texture of the road because of nonuniformity is also a challenge for lane detection.

They concluded by reviewing the theoretical background about lane detection methods and their properties. It has already been stated that DET curves plotted for two methods, the one with greater area under it is better in terms of accuracy (same is with ROC).After analysing the performances of both the methods it is verified that method 1 based on Canny edge detection is better than Sobel operator based lane detection method 2 as it covers greater area.

## 2.2 Unstructured Road Detection based on Contour Selection

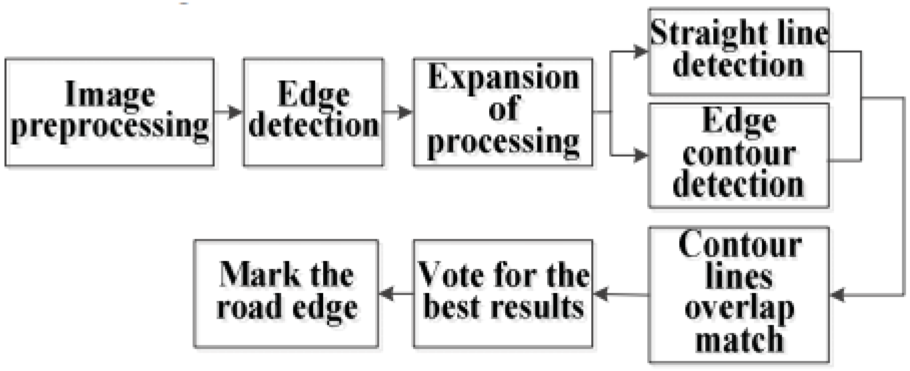


**Figure 2.2.3** Flowchart of the model proposed in this paper

The captured image generally contains noise due to a disturbance. Because of randomness of noise and image signals correlation in space and time, the influence of noise on a pixel will make its grey scale significantly different with its neighbour pixel or with the corresponding pixel next frame. The following edge detection algorithm is mainly based on image intensity of the first and second order derivative, and derivative generally is sensitive to the noise. Therefore, it is necessary to use a filter to improve the performance. Gaussian filter is used to reduce noise.

Gaussian is kind of a linear smoothing filter and can eliminate gaussian noise which has a small error of space a good spatial stability, a small error of space position, is widely used in image processing of the noise reduction process. Gaussian filter is the input array of each pixel point convolution operation with the gauss kernel, the convolution of value as output pixel value. The image after Gaussian filter smooth degree, depends on the standard deviation. It is the output of the pixel in the field of weighted average, at the same time the nearer the centre pixel is higher. Compared with the other filter therefore, its softer, smoother and edge retention is better.

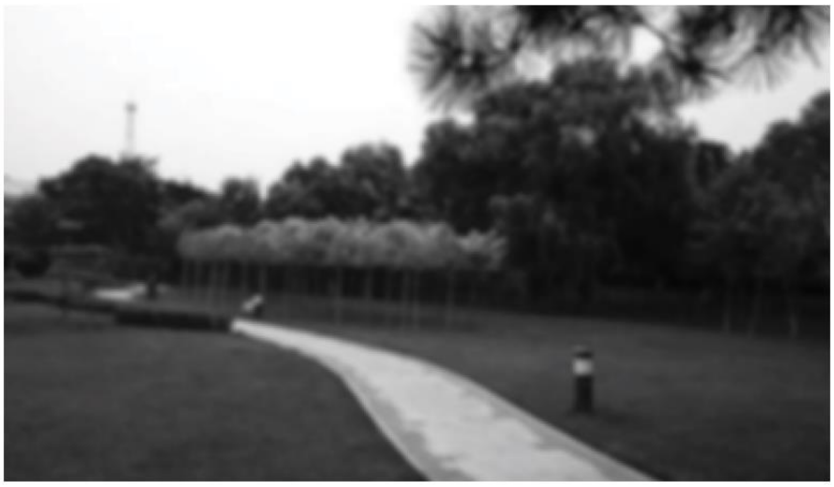


**Figure 2.2.4** Gaussian filter processing

Canny edge detection operator is a multistage edge detection algorithm which was developed by john canny in 1986. principle of canny edge detection operator is to determine the edge pixels of the image by the maximum value of the image signal function. There are three evaluation parameters for optimal edge detection:

1) Low error rate. All edges should be found, and there should be no spurious response. In other words, the detected edges must be as real as possible edges.

2) Image edges should be well positioned. The edges that have been positioned must be as close as possible to the real edges. In other words, the distance between the point marked by the detector and the centre of the true edge should be minimal.

3) Single edge point response. For the real edge points, the detector should return only a point. In other words, the local maximum around the true edge should be minimal. This means that in only a single edge point position.

In this paper, a kind of unstructured road detection based on contour selection algorithm, is improvements in terms of the conventional linear detection method. Test results of our algorithm for some edge complete roads are good, but there is still limitation that this algorithm relies entirely on the final result of canny edge detection. Because the results of the canny detection algorithm in the cracked road often have a lot of irrelevant edges, which will produce a lot of irrelevant contours for the subsequent contour detection. The final detection effect is poor. Therefore, the pre-processing of original image in our algorithm needs to be improved, but also optimize the line and contour matching algorithm to make the output more accurate.

## 2.3 Lane-line Detection Algorithm for Complex Road Based on OpenCV

At present, there are much research on lane line detection methods at home and abroad, which are mainly divided into feature-based and model-based methods. Some methods enhance the contrast between lane line and road by adjusting the CCD brightness, gain and exposure time， then, the seed points of image is selected and classified， after this, Hough transform is performed on the seed points. There are also methods use circular curve lane line model and density-based Hough transformation for lane line recognition. Some researchers also adopted the boundary tracking detection algorithm based on fuzzy clustering to realize lane line recognition when identifying the interested area of lane. However, because the collected lane line image is affected by light, wear, vehicle shade and tree shadow, it is still a big challenge to accurately detect the lane line.

Therefore, a fast detection algorithm for lane line pixels based on combined gradient and colour filter of region of interest is proposed. First, the algorithm uses Sobel edge detection operator to detect the edge information of the structured road based on the high contrast between the lane line and the road surface. The second basis is the colour characteristics of the lane line. Generally speaking, there are only two colours of the lane line, white and yellow, so we can filter these two colours in the colour space to extract the pixel of the lane line. Then, in the region of interest model, a relatively stable lane line extraction method can be obtained by combining edge gradient and colour filter.

The authors followed a methodology which proceeded like this,

- Camera calibration
- Edge detection
- Area of interest
- Colour Thresholding
- Sliding window and polynomial fitting
- Restore to original perspective

In this paper, combined edge method and colour filter are used to detect the lane line, sobel operator and HSL geometric structure are also introduced. The lane line is extracted by sliding window and polynomial fitting, and the real lane line is restored by perspective transformation. The test results of Open CV platform show that the algorithm has better

Realtime performance and anti-interference performance, it can well detect the lane line of its dotted line and the solid line, realize the real-time line marking in the video image, this test method effectively avoid the light, wear, car shade and trees shadow effects on the lane line image. Amount of calculation and robustness of algorithm are highly optimized, this method can be also applied to the safety assistant driving system, or autonomous vehicle system.

# Chapter 3
# OBJECTIVES

- To learn the characteristics of roads in India, and to collect data on the same.
- To implement an algorithm that recognizes structured and unstructured roads, using image processing techniques.
- To create a machine learning model that will classify as well as identify structured and unstructured roads.

# Chapter 4
# TOOLS AND SOFTWARE

## 4.1 Jupyter Notebook

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.

A notebook kernel is a "computational engine" that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist. The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels.

## 4.2 Matplotlib and NumPy

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open-source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project, and you can use it freely. NumPy stands for Numerical Python. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

## 4.3 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

## 4.4 Scikit Learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**.

## 4.5 Pandas

Pandas is a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas has grown into one of the most popular Python libraries. It has an extremely active community of contributors.

Pandas is built on top of two core Python libraries—matplotlib for data visualization and NumPy for mathematical operations. Pandas acts as a wrapper over these libraries, allowing you to access many of matplotlib's and NumPy's methods with less code. For instance, pandas' .plot() combines multiple matplotlib methods into a single method, enabling you to plot a chart in a few lines.

Before pandas, most analysts used Python for data munging and preparation, and then switched to a more domain specific language like R for the rest of their workflow. Pandas introduced two new types of objects for storing data that make analytical tasks easier and eliminate the need to switch tools: Series, which have a list-like structure, and DataFrames, which have a tabular structure.

## 4.6 TensorFlow and Keras

TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google’s Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well. TensorFlow is basically a software library for numerical computation using data flow graphs where:

- nodes in the graph represent mathematical operations.
- edges in the graph represent the multidimensional data arrays (called tensors) communicated between them. (Please note that tensor is the central unit of data in TensorFlow).

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is:

- **Simple** -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- **Flexible** -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- **Powerful** -- Keras provides industry-strength performance and scalability

# Chapter 5
# PROPOSED MODELS

## 5.1 Hough Lines and Threshold masking Model 1

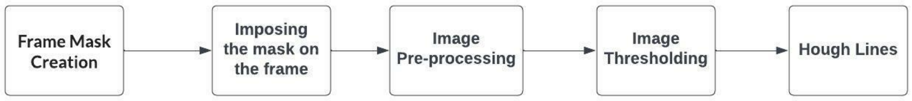


**Figure 5.1.5** Flowchart for the proposed model 1

In this proposed model, the lane detection process was divided into 4 main parts:

1. **Frame Mask Creation:**

   The frame mask consisted of a 2-D NumPy array, with each pixel ranging from 0 (black) to 255 (white). But since the region of interest was in the shape of a polygon, the coordinates of the polygon were used to prepare the mask, while setting all the other pixels to 0. The picture below depicts the mask:

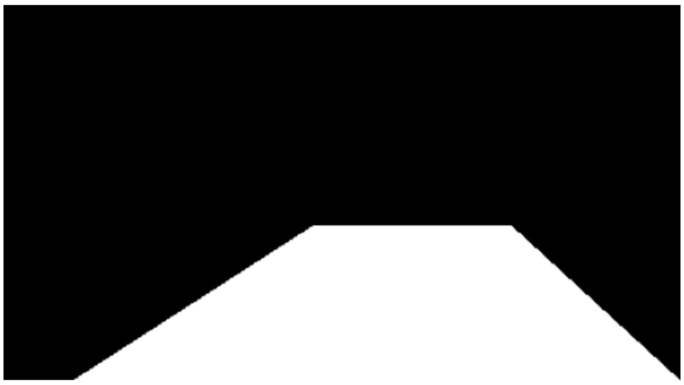


**Figure 5.1.6** Polygon mask creation

2. **Imposing the mask on the frame:**

   Using OpenCV's 'bitwise_and' function, the mask was imposed on the original frame.

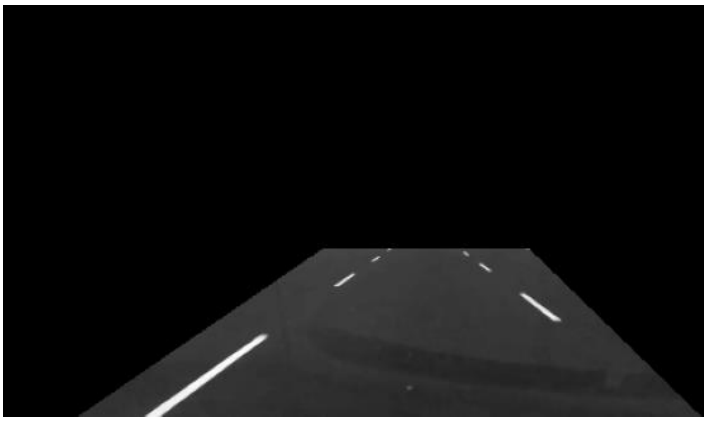


**Figure 5.1.7** Mask imposition on the original frame

3. **Image Thresholding:**

   Removing all the mediator grey pixels to create a perfectly binary image. So that the positions of the lines can be seen perfectly for the upcoming processing.

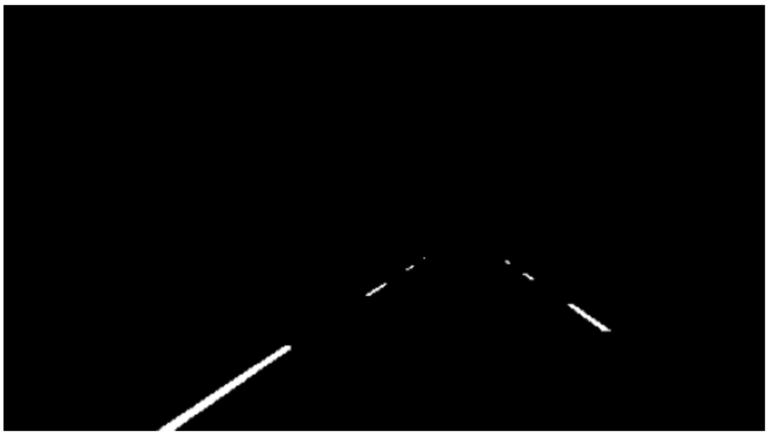


**Figure 5.1.8** Thresholded Image

4. **Hough Line transform:**

   Use the HoughLinesP function which is readily available in the OpenCV library. The function generates a continuous line using previously thresholded image as its input, to find the continuous lines. Now the lines and the original image can be superimposed to get a good view on the output of the model which can be seen below in the image.

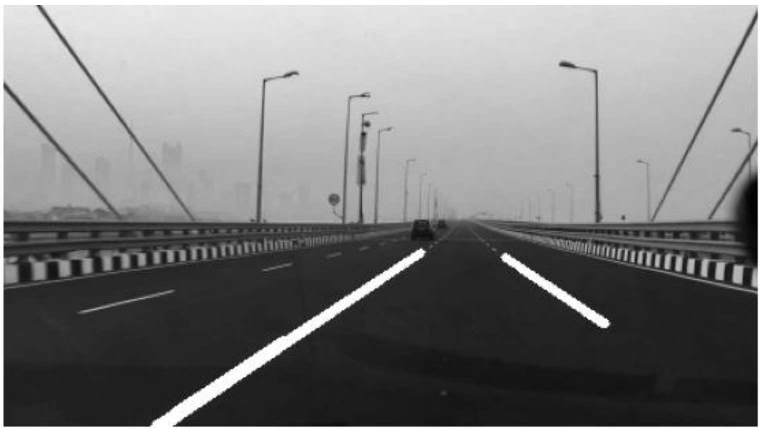


**Figure 5.1.9** Original image imposed with Hough transformed lines

## 5.2 HSV Filtering and Sliding Window search Model 2

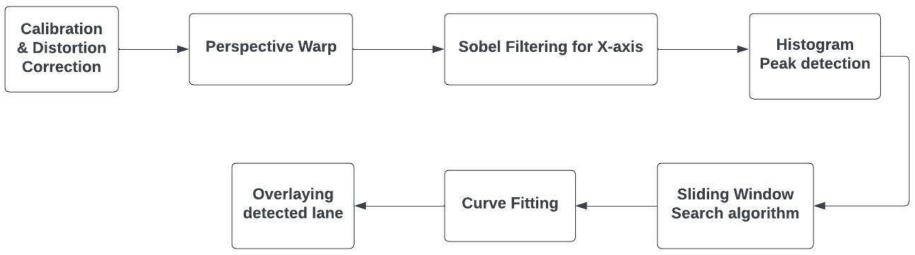

Calibration & Distortion Correction
Perspective Warp
Sobel Filtering for X-axis
Histogram Peak detection
Sliding Window Search algorithm
Curve Fitting
Overlaying detected lane

**Figure 5.2.10** Flowchart of the proposed model 2

This model was developed using multiple filters and object detection algorithm:

1. **Calibration and Distortion correction:**

   To focus incoming light on the camera sensor, camera lenses distort it. Even while it's quite helpful for capturing photographs of our surroundings, they frequently end up slightly misrepresenting light. In computer vision applications, this may lead to inaccurate measurements. Images can provide a distortion model that takes lens distortions into account by being calibrated against a known object. For this, the distortion model was created using 20 images of a checkerboard. The undistorted image was created using OpenCV's calibration and distortion methods. The difference might not be noticeable, but it can have a huge impact on the image processing process.

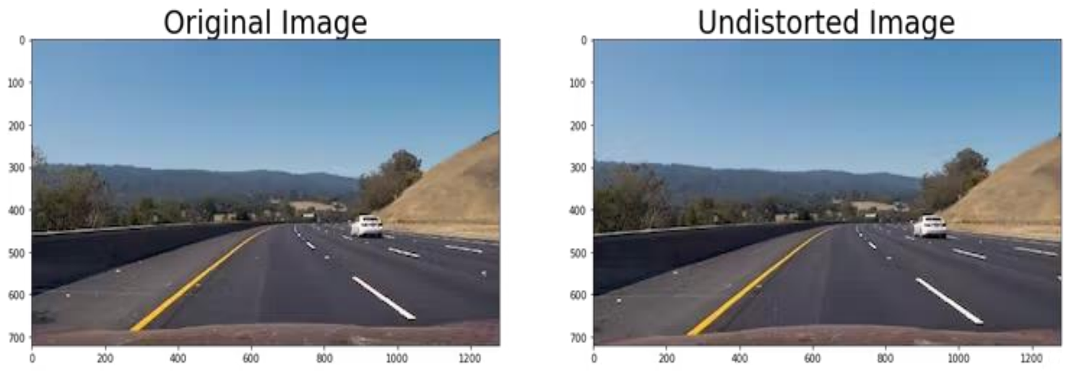


**Figure 5.2.11** Distortion correction applied to a frame

2. **Perspective Warp**

   The perspective warp technique aids in detecting the curved lanes in camera space. By applying a perspective transformation on the image, the birds eye view of the image can be obtained, which makes it easier to process further. By assuming that the lane is on a flat 2D surface, a polynomial can be fir, that can accurately represent the lane in lane space.

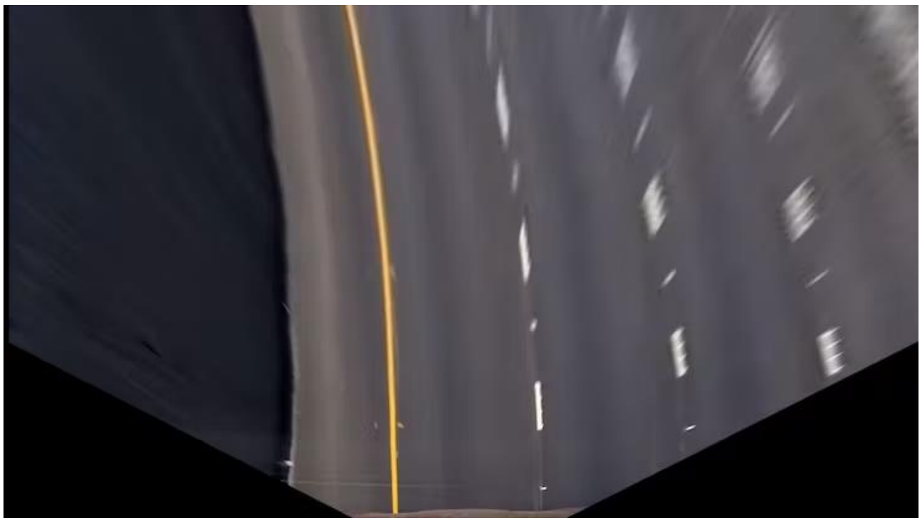


**Figure 5.2.12** Perspective warped/birds-eye view image

3. **Sobel Filtering**

   The Sobel filter is used to detect edges, as they perform a 2-D spatial gradient measurement on an image and so emphasize regions of high spatial frequency that correspond to edges. To monitor the saturation and lightness of the image, the HSL (Hue-Saturation-Lightness) colour space is employed. Both channels are subjected to the Sobel operator, and the gradient with respect to the x-axis is extracted. The pixels that pass the gradient threshold are then added to a binary matrix that represents the pixels in the image. As the entire image is not required, only a portion of the image is selected, which is not blurry or noisy.

**Figure 5.2.13** Zoomed image of road lines

4. **Histogram Peak detection**

   To determine the starting point of the Sliding Window search algorithm, a lane of pixels is required. To detect this, a histogram was used. Each portion of the histogram below displays how many white pixels are in each column of the image. The highest peaks of each side of the image are then taken, one for each lane line. Here's what the histogram looks like:

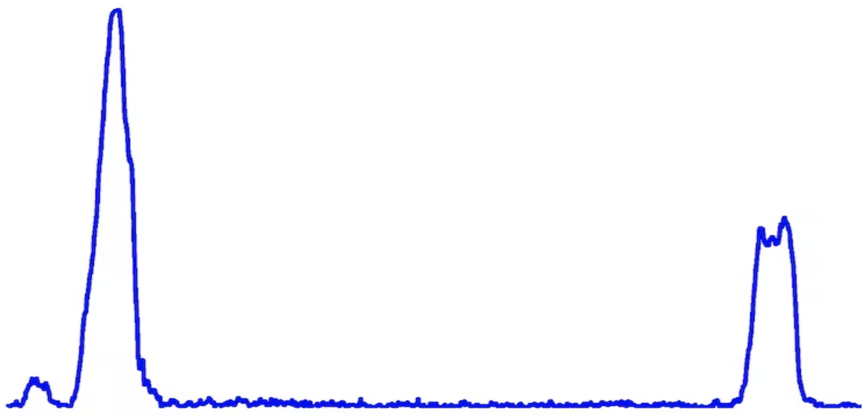


**Figure 5.2.14** Identifying the histogram peaks

5. **Sliding Window search**

   Sliding windows play an integral role in object classification, as they allow us to localize exactly “where” in an image an object resides. A sliding window is a rectangular region of fixed width and height that “slides” across an image, where an image classifier can be applied, to determine if the window has an object of interest. Starting from the initial position, the first window measures how many pixels are located inside the window. If the number of pixels reaches a certain threshold, it shifts the next window to the average lateral position of the detected pixels. If enough pixels are not detected, the next window starts in the same lateral position. This continues until the windows reach the other edge of the image.

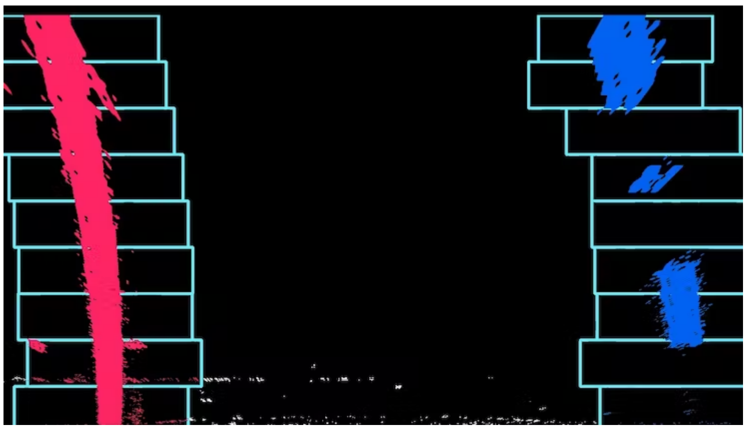


**Figure 5.2.15** Sliding window technique on the edges

### 6. Curve Fitting

Finally, in curve fitting, polynomial regression is applied to the red and blue pixels individually. The figure 4.2.8 depicts the combination of sliding window and curve fitting techniques.

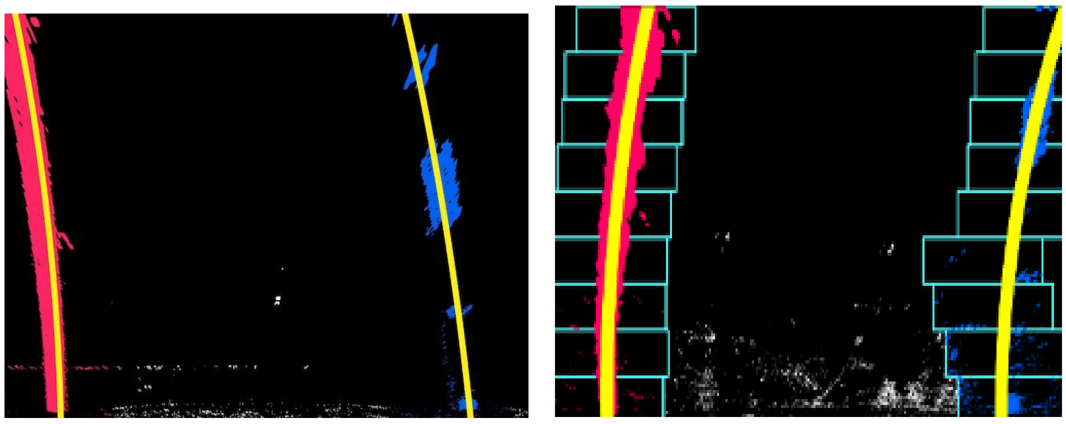


**Figure 5.2.16** Curve Fitting, Sliding Window + Curve Fitting

### 7. Overlaying detected window

An overlay was created, which fills in the detected portion of the lane, which can then be applied on the image.

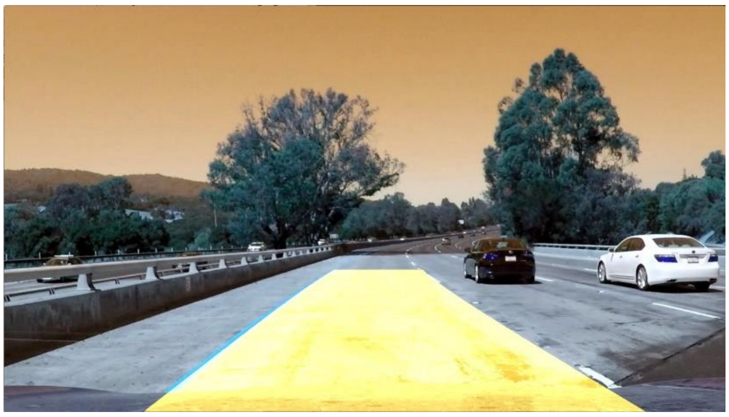


**Figure 5.2.17** Overlaying the detected window on the original image

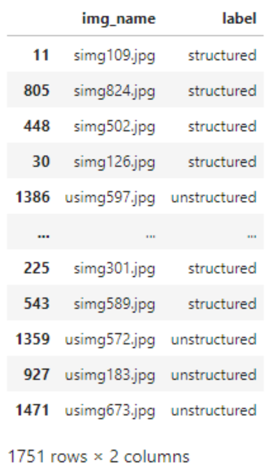
## 5.3 Creation of Machine Learning Model

Since the model 2 was showing more promise than model 1, it is used to design a machine learning model. The model is based on the TensorFlow keras models. From many models available in keras, sequential model is used because it shows optimal output. TensorFlow lite would have been an option if the model was being built as an android application.

The machine learning model will be taking input. The input image will be pre-processed till the Sobel filtering i.e., the image is initially undistorted, followed by Sobel filtering and perspective warping. At this stage, the image which is generated by Sobel filter which is warped can be used as an input for the Machine learning model.

### 1. Creation of Dataset

The bigger dataset was extracted from the site Kaggle. From the dataset, random selection of 1751 images were done, and they were segregated into 2 different classes called structured and unstructured. Using the pandas library, a data frame was created with columns as image name and class labels.



| | img_name | label |
|---|---|---|
| **11** | simg109.jpg | structured |
| **805** | simg824.jpg | structured |
| **448** | simg502.jpg | structured |
| **30** | simg126.jpg | structured |
| **1386** | usimg597.jpg | unstructured |
| **...** | ... | ... |
| **225** | simg301.jpg | structured |
| **543** | simg589.jpg | structured |
| **1359** | usimg572.jpg | unstructured |
| **927** | usimg183.jpg | unstructured |
| **1471** | usimg673.jpg | unstructured |

1751 rows × 2 columns

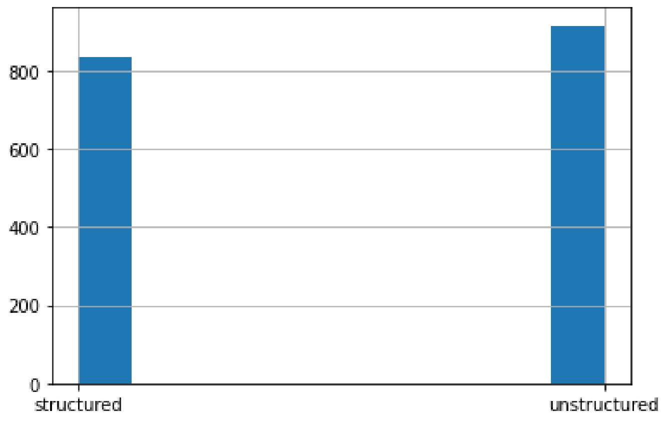


**Figure 5.2.18** Data frame and number of images in each class

## 2. Creation of a sequential model with CNN

The model type that we will be using is Sequential. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. We use the 'add()' function to add layers to our model.

Our first layer is the Conv2D layer. This is the convolution layer that will deal with our input images, which are seen as 2-dimensional matrices. The next layer is the batch normalization layer, that normalizes its inputs. Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. The following layer is the MaxPooling2D layer. It down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by 'pool_size') for each channel of the input. The window is shifted by 'strides' along each dimension. The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/ (1 - rate) such that the sum over all inputs is unchanged.

Once all the three layers are added, the copies of the same layers are again added with different parameters. At every stage, the feature set gets improved, and the machine starts learning the patterns and other aspects on its own.

In between the Conv2D layers and the dense layer, there is a 'Flatten' layer. Flatten serves as a connection between the convolution and dense layers. 'Dense' is the layer type that will be used for output layer. Dense is a standard layer type that is used in many cases for neural networks. The proposed model has 2 nodes in the output layer, one for each possible outcome (structured and unstructured).

The activation is 'SoftMax'. SoftMax makes the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

Compiling the model takes three parameters: optimizer, loss and metrics. The optimizer controls the learning rate. The model is using 'adam' as our optimizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training.

The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

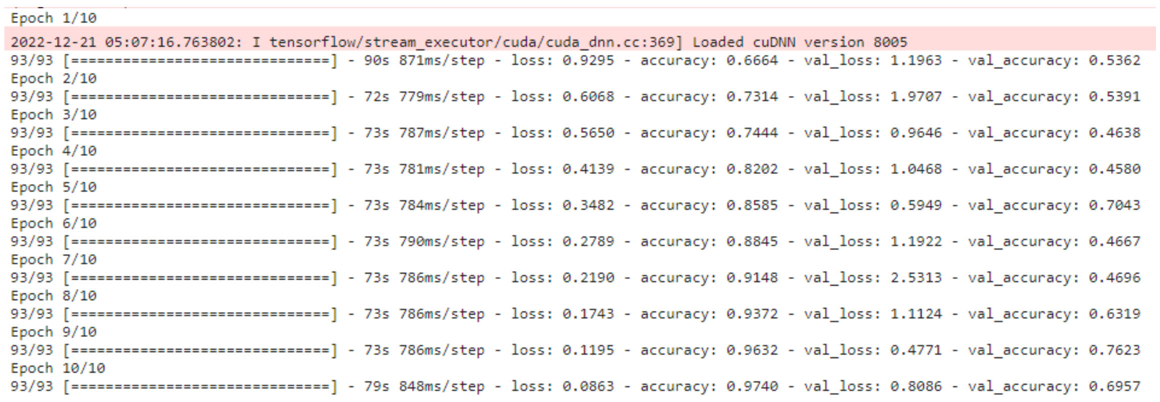
The 'categorical_crossentropy' loss function is the most common choice for classification. A lower score indicates that the model is performing better. To make things even easier to interpret, the 'accuracy' metric is used to see the accuracy score on the validation set when the model is trained.

Keras ImageDataGenerator is used to take the inputs of the original data and then transform it on a random basis, returning the output resultant containing solely the newly changed data.

To train, the 'fit()' function is used on the model with the following parameters: training data (train_generator, from imagedatagenerator), validation data, and the number of epochs.

For the validation data, the train_test_split function is used to split the data set into training and validation data set.

The number of epochs is the number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. For the model, the number of epochs is 10.



```
Epoch 1/10
2022-12-21 05:07:16.763802: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8005
93/93 [==============================] - 90s 871ms/step - loss: 0.9295 - accuracy: 0.6664 - val_loss: 1.1963 - val_accuracy: 0.5362
Epoch 2/10
93/93 [==============================] - 72s 779ms/step - loss: 0.6068 - accuracy: 0.7314 - val_loss: 1.9707 - val_accuracy: 0.5391
Epoch 3/10
93/93 [==============================] - 73s 787ms/step - loss: 0.5650 - accuracy: 0.7444 - val_loss: 0.9646 - val_accuracy: 0.4638
Epoch 4/10
93/93 [==============================] - 73s 781ms/step - loss: 0.4139 - accuracy: 0.8202 - val_loss: 1.0468 - val_accuracy: 0.4580
Epoch 5/10
93/93 [==============================] - 73s 784ms/step - loss: 0.3482 - accuracy: 0.8585 - val_loss: 0.5949 - val_accuracy: 0.7043
Epoch 6/10
93/93 [==============================] - 73s 790ms/step - loss: 0.2789 - accuracy: 0.8845 - val_loss: 1.1922 - val_accuracy: 0.4667
Epoch 7/10
93/93 [==============================] - 73s 786ms/step - loss: 0.2190 - accuracy: 0.9148 - val_loss: 2.5313 - val_accuracy: 0.4696
Epoch 8/10
93/93 [==============================] - 73s 786ms/step - loss: 0.1743 - accuracy: 0.9372 - val_loss: 1.1124 - val_accuracy: 0.6319
Epoch 9/10
93/93 [==============================] - 73s 786ms/step - loss: 0.1195 - accuracy: 0.9632 - val_loss: 0.4771 - val_accuracy: 0.7623
Epoch 10/10
93/93 [==============================] - 79s 848ms/step - loss: 0.0863 - accuracy: 0.9740 - val_loss: 0.8086 - val_accuracy: 0.6957
```

**Figure 5.2.19** Training the model

# Chapter 5
# RESULTS

The machine learning model has different parameters depending on which the output of the model is decided. Firstly, the data frame is divided into batches using ImageDataGenerator function. This will help by dividing the model input into batches based on sizes and classes. Along with the batch creation it can process the image, i.e., convert the image from RGB to Grayscale, which can help in the pre-processing the data before feeding it to the model. Later on the batches of Images are passed through the model which has the weights already set by the developer during the creation phase. These weights and other parameter create a complex neural network which in turn provides the prediction after learning about the features from the Test set.

Epochs are like iterations where the model tries to learn from mistakes and tries to improve itself as it proceeds future. Once the model is trained it can be saved using the .h5 file extension. This model can be later called anywhere to be used as a trained model.

The model will provide its train and validation accuracy as well as loss so that the developer can change the parameters according the requirement.

In the proposed model, batch size was 15 and 5 layers of conv2d were added with weightage of 24, 32, 40, 48, 56 and 64. The dropout was 0.2 and the optimizer was rmsprop. This model gave the best result as showed in the image below.

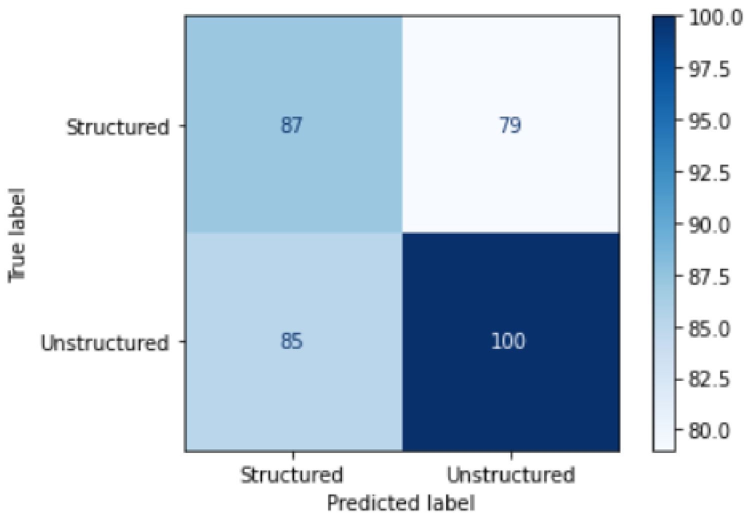


**Figure 5.20** Confusion matrix of the model

As the confusion matrix justifies the model is not yet precise. The model is still detecting a lot of structured images as unstructured and unstructured as structured, i.e., the false positives and false negatives are a lot.

# Chapter 6
# CONCLUSION AND FUTURE SCOPE

The proposed model shows a result which can be further improved based on the image pre-processing that is to be done during the early stages of the model. The validation accuracy is around 75%. This is satisfactory enough at this stage. Further development in the pre-processing and change in parameter of the machine learning model to an even more suitable optimizer and suitable weights can improve the machine learning model's results.

## References

[1] W. Xiang, Z. Juan and F. Zhijun, “Unstructured road detection based on contour selection,” in *4th International Conference on Smart and Sustainable City (ICSSC 2017)*, Shanghai, 2017.

[2] S. K. Vishwakarma, A. and D. S. Yadav, “Analysis of lane detection techniques using openCV,” in *2015 Annual IEEE India Conference (INDICON)*, New Delhi, India, 2016.

[3] A. Sharma, M. Kumar and R. Kumar , “Lane Detection Using Python,” in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2021.

[4] Z. Wang, Y. Fan and H. Zhang, “Lane-line Detection Algorithm for Complex Road Based on OpenCV,” in *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Chongqing, China, 2019.

[5] P. Joshi, “Analytics Vidhya,” 13 May 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/05/tutorial-real-time-lane-detection-opencv/.

[6] kemfic, “github,” github, 22 October 2018. [Online]. Available: https://github.com/kemfic/Curved-Lane-Lines.

[7] kylesf, “github,” 22 December 2016. [Online]. Available: https://github.com/kylesf/Advanced-Lane-Detection.

[8] M. Ameen, “github,” 24 October 2017. [Online]. Available: https://github.com/mohamedameen93/Lane-lines-detection-using-Python-and-OpenCV/commits/master/Lane-Lines-Detection.ipynb.

[9] A. Eijaz , “towardsdatascience.com,” 17 October 2018. [Online]. Available: https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5. [Accessed 20 December 2022].

[10] M. Pahwa, “Kaggle.com,” 15 December 2019. [Online]. Available: https://www.kaggle.com/datasets/manjotpahwa/indian-driving-dataset. [Accessed 26 November 2022].