DEEP LEARNING-BASED VEHICLE BEHAVIOR PREDICTION FOR AUTONOMOUS DRIVING APPLICATION

A PROJECT REPORT

Submitted By

ELANGOVAN D	**(732519104006)**
MONAL PRATHAP S	**(732519104018)**
TAMIZHARASAN M	**(732519104029)**

in partial fulfillment for the award of the degree
of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SHREE VENKATESHWARA HI-TECH ENGINEERING COLLEGE

GOBI-638455

ANNA UNIVERSITY::CHENNAI 600 025

MAY: 2023

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DEEP LEARNING BASED VEHICLE BEHAVIOR PREDICTION FOR AUTONOMOUS DRIVING APPLICATION''** is the bonafide work of **"ELANGOVAN D (732518104006), MONAL PRATHAP S (732518104018), TAMIZHARASAN M (732518104029)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.T.SENTHIL PRAKASH,M.E,Ph.D.,	Dr.T.SENTHIL PRAKASH,M.E,Ph.D.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering
Shree Venkateshwara Hi-Tech Engineering College,	Shree Venkateshwara Hi-Tech Engineering College,
Gobichettipallayam-638455	Gobichettipallayam-638455
Erode(D.T),TN.	Erode(D.T),TN.

Submitted for the University Viva-Voce held on ______________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DEEP LEARNING BASED VEHICLE BEHAVIOR PREDICTION FOR AUTONOMOUS DRIVING APPLICATION

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and Deep regards to our beloved **Thiru.P.VENKATACHALAM**, Chairman, **Thiru.K.C.KARUPANAN**, Secretary, **Thiru.G.P.KETTIMUTHU**,Joint Secretary, and the Management of Shree Venkateshwara Hi-Tech Engineering College who provided all the facilities with the state of art of infrastructure.

We express our earnest sense of gratitude to our beloved Principal **Dr.P.THANGAVEL, ME MBA PhD,** Shree Venkateshwara Hi-Tech Engineering College whose immense help has been a candle in the darkness with regards to the project work.

We are highly grateful to **Dr.S.PRAKASAM, ME MBA PhD,** Vice Principal, for his enthusiasm and support which has been instrumental in the success of the project.

We are very grateful to **Dr.T.SENTHIL PRAKASH ME PhD, Professor and Head Department of Computer Science and Engineering**, for the aspiring suggestion, invaluably constructive criticism and friendly advice during the project work.

We would like to express our profound thanks to our project coordinator **Dr.T.SENTHIL PRAKASH ME PhD,** Professor of Computer Science and Engineering thoughtful words helped us in completing our project successfully.

We wish to express our gratefulness to our project supervisor **Dr.T.SENTHIL PRAKASH ME PhD,** Professor of Computer Science and Engineering for his valuable guidance and constructive suggestions.

Also,we thank all the **Teaching and non-teaching staff**, who have patiently provided us assistance in this project. We also thank to our **classmates** for their encouragement and help during the course of the project.

ABSTRACT

The self-driving cars are also known as autonomous vehicles. This car has the ability to sense around the environment. These sensed parameters are processed and according to it the different actuators in the car will work without any human involvement .An autonomous car work like a normal car but without any human driver. Autonomous cars rely on sensors, actuators, machine learning algorithms and software to perform all the automated functions. The software part is very important for autonomous vehicles. The software architecture acts as a bridge between hardware components and application. The standardized software for automotive cars is AUTOSAR.The AUTOSAR is a standardized architecture between application software and hardware. This standardized architecture provide all communication interfaces, device drivers, basic software and run-time environment. There are two important modules in self-driving cars. They are lane detection and traffic signal detection which works automatically without any human intervention. A machine learning algorithm is proposed in this project. This algorithm is mainly used to train the shape models and helps to detect the shape for traffic sign detection and lane detection. These both tasks are programmed using python with open cv2 library file, numpy library file and though detection technique is used to detect the appropriate circles of the traffic signals .By using all these tools, all the shape models are trained using supervised training algorithm and the detection is performed in such a way to help autonomous cars to detect the lane and traffic sign.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ACRONYM	**ABBREVIATIONS**
AUTOSAR	**AUT**omotive **O**pen **S**ystem **AR**chitecture
SSLA	**S**hape **S**upervised **L**earning **A**lgorithm
ADAS	**A**dvanced **D**river **A**ssistance **S**ystem
LiDar	**Li**ght **D**etection and **R**anging
GPS	**G**lobal **P**ositioning **S**ystem
CNN	**C**onvolutional **N**eural **N**etwork
RGB	**R**ed **G**reen **B**lue
KDD	**K**nowledge **D**iscovery in **D**atabases
ACID	**A**tomicity, **C**onsistency, **I**solation **D**urability
HDFS	**H**adoop **D**istributed **F**ile **S**ystem
CV2	**C**omputer Vision 2
LSTM	**L**ong **S**hort-**T**erm **M**emory

LIST OF FIGURES

CHAPTER 1

1.1 INTRODUCTION

Safety is the important aspect which must be noticed while driving vehicles.In a survey it is been published that more 10 lakhs of people die in the road accidents in a country. The Road accident happened due to Human Errors are about 98%. So, to avoid this all over the world Autonomous Cars are under Research and Development.The term Autonomous Cars is that the car drives itself using various technologies without any Human intervention. For Autonomous Cars Software task development is very much important. The Software architecture acts as a bridge between Hardware Components and Application. The Standardized Software for Automotive cars is AUTOSAR. This Standardized Architecture provide all Communication Interfaces, Device Drivers, Basic Software and Run-Time Environment. There are two important tasks in Autonomous cars they are Lane detection and Traffic Sign Detection. These two tasks are important because many accidents are due to malfunction of these two tasks.A New Algorithm **SSLA (Shape Supervised Learning Algorithm)** is proposed in this project. The Hough Line Transformation is the technique which is used to detect the Traffic Sign Detection.Matplolib and numpy is the library files in python used for Lane Detection. These two techniques are possible by Open CV, numpy libraries in python. The Hough line Transformation is used to detect any shapes. In order to detect the Lane in which the car is to drive is by using various Edge detection techniques which makes use of colors in python.

CHAPTER 2
LITERATURE SURVEY

2.1 D. Helbing and P. Molnár, 1995 “Social force model for pedestrian dynamics,” Phys.Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 51, no. 5, p. 4282.

It is suggested that the motion of pedestrians can be described as if they would be subject to "social forces." These "forces" are not directly exerted by the pedestrians' personal environment, but they are a measure for the internal motivations of the individuals to perform certain actions (movements). The corresponding force concept is discussed in more detail and can also be applied to the description of other behaviours. In the presented model of pedestrian behaviour several force terms are essential: first, a term describing the acceleration towards the desired velocity of motion; second, terms reQecting that a pedestrian keeps a certain distance from other pedestrians and borders; and third, a term modeling attractive effects. The resulting equations of motion are nonlinearly coupled Langevin equations. Computer simulations of crowds of interacting pedestrians show that the social force model is capable of describing the self-organization of several observed collective efFects of pedestrian behaviour very realistically.

2.2 A.Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social LSTM: Human trajectory prediction in crowded spaces,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 961–971

Pedestrians follow different trajectories to avoid obstacles and accommodate fellow pedestrians. Any autonomous vehicle navigating such a scene should be able to foresee the future positions of pedestrians and accordingly adjust its path to avoid collisions. This problem of trajectory prediction can be viewed as a sequence generation task, where we are interested in predicting the future trajectory of people based on their past positions. Following the success of Recurrent Neural Network (RNN) models for sequence prediction tasks, we propose an LSTM model which can learn general human movement and predict their future trajectories. This is contrast to traditional approaches which use hand-crafted functions such as Social forces. We demonstrate the performance of our method on several public datasets. Our model outperforms state-of-the-art methods on some of these datasets. We also analyze the trajectories predicted by our to demonstrate the motion learned by our model.

2.3 A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 2255–2264

Understanding human motion behaviour is critical for autonomous moving platforms (like self-driving cars and social robots) if they are to navigate human-centric environments. This is challenging because human motion is inherently multimodal: given a history of human motion paths, there are many socially plausible ways that people could move in the future. We tackle this problem by combining tools from sequence prediction and generative adversarial networks: a recurrent sequence-to-sequence model observes motion histories and predicts future behaviour, using a novel pooling mechanism to aggregate information across people. We predict socially plausible futures by training adversarial against a recurrent discriminator, and encourage diverse predictions with a novel variety loss. Through experiments on several datasets we demonstrate that our approach outperforms prior work in terms of accuracy, variety, collision avoidance, and computational complexity.

2.4 T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection," Neural Netw., vol. 108, pp. 466–478, Dec. 2018.

As humans we possess an intuitive ability for navigation which we master through years of practice; however existing approaches to model this trait for diverse tasks including monitoring pedestrian flow and detecting abnormal events have been limited by using a variety of hand-crafted features. Recent research in the area of deep-learning has demonstrated the power of learning features directly from the data; in sequence-to-sequence problems such as neural machine translation and neural image caption generation. Motivated by these approaches, we propose a novel method to predict the future motion of a pedestrian given a short history of their, and their neighbors, past behavior. The novelty of the proposed method is the combined attention model which utilizes both "soft attention" as well as "hard-wired" attention in order to map the trajectory information from the local neighborhood to the future positions of the pedestrian of interest. The navigational capability of the proposed method is tested on two challenging publicly available surveillance databases where our model outperforms the current-state-of-the-art methods. Additionally, we illustrate how the proposed architecture can be directly applied for the task of abnormal event detection without handcrafting the features.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the existing work there are Automotive cars which work through Sensors, Actuators and an Embedded System Control. Here the Lane Keeping is very much important in terms of safety measures to prevent road accidents.In the Concept of Lane keeping the LiDar,radar and GPS is used in existing research work to keep the vehicles in the lane.Also,the lane keeping in the existing papers are achieved through ADAS based system with the help of Adaptive cruise control and this technique is performed through Deep Neural Networks.The Simulation part is possible in Carla Software also.

3.1.1 DISADVANTAGES

Deep learning based methods are widely used to perform lane detection, thanks to their robustness in various conditions. Usually the methods are formulated in a non-sequential way, taking individual frames as input, and producing detection output based on a single input frame. Because of this, plus the imperfect detection and the fact that the lane markers are not always well-maintained in the real world, we are constantly left with some shaky and noisy lane detection results. An example of such can be seen in. Due to the damage of the lane marker, the green lane is shorter than it should be. Since it is a curvy road, it is obvious that a naive fix by simply prolonging the lane will introduce large error. For applications like lane keeping and lane switching detection, steady and clear lane positions are needed. To achieve this, a post processing stage in the form of lane filtering in temporal domain is necessary.

3.2 PROPOSED SYSTEM

Where thousands of images are put into training models to get an accurate output model.An Algorithm is proposed in this project which is used to identify the appropriate shape. This Identification is possible through training model. The proposed Algorithm holds Hough line transformation technique which is used to detect any shape. Even the shape is broken also this technique works in an Efficient way. The shape which is detected in turned out in a mathematical form by using various formulae. The maximum Area of the shape is 64480. In this project, the circle shape is required to be detected because Traffic signals are in the shape of circle. Not all circles are detected. It is because the traffic sign is placed on the higher place.

3.2.1 ADVANTAGES

We assume that it is setup to make the baseline horizontal, which assures the horizon in the image, is parallel to the X-axis. Otherwise, we can adjust the image using the calibration data of the camera to make it. Each lane boundary marking, usually a rectangle (or approximate) forms a pair of edge lines. In this project, it was assumed that the input to the algorithm was a 620x480 RGB color image. Therefore the first thing the algorithm does is to convert the image to a grayscale image in order to minimize the processing time. Secondly, as presence of noise in the image will hinder the correct edge detection. Therefore, we apply (F.H.D.) algorithm to make the edge detection more accurate. Then the edge detector is used to produce an edge image by using canny filter with automatic thresholding to obtain the edges, it will reduce the amount of learning data required by simplifying the image edges considerably. Then edged image sent to the line detector after detecting the edges which will produces a right and left lane boundary segment.

CHAPTER 4
SYSTEM SPECIFICATION

4.1 SYSTEM REQUIREMENTS

4.1.1 HARDWARE REQUIREMENTS

• Processor	: AMD Ryzen 5000 series 2.4 GHz.
• Hard Disk	: 50 GB.
• Monitor	: 15 inch VGA Color.
• RAM	: 8 GB

4.1.2 SOFTWARE REQUIREMENTS

• Operating System	: Windows 11
• Platform	: Python Technology
• Tool	: Python 3.6
• Front End	: Python Anaconda Script 2.5
• Back End	: Spyder 3.10

CHAPTER 5
SYSTEM ENVIRONMENT

5.1 ALGORITHM

Convolution Neural Network (CNN) is a Deep Learning algorithm specially designed for working with Images and videos. It takes images as inputs, extracts and learns the features of the image, and classifies them based on the learned features. This blog will be all about another Deep Learning model which is the Convolutional Neural Network. As always this will be a beginner's guide and will be written in such as matter that a starter in the Data Science field will be able to understand the concept, so keep on reading ,

5.1.1 TABLE OF CONTENTS

1. Introduction to Convolutional Neural Network
2. Its Components
   - Input layer
   - Convolutional Layer
   - Pooling Layer
   - Fully Connected Layer
3. Practical Implementation of CNN on a dataset
   - Introduction to CNN

Convolution Neural Network (CNN) is a Deep Learning algorithm specially designed for working with Images and videos. It takes images as inputs, extracts and learns the features of the image, and classifies them based on the learned features.Transform into an expert and significantly impact the world of data science.Thisalgorithmis inspired by the information received from each layer is combined and the image/visual is interpreted or classified.Similarly, CNN has various filters, and each filter extracts some information from the image such as edges, different kinds of shapes (vertical, horizontal, round), and then all of these are combined to identify the image.

- **Feature Extraction:** First 3 layers (input, convolution and pooling) are used for featureextraction
- **Classification:** Last two layers (Fully Connected and output) are used for classification (Lane right or Left)

The convolution operation is responsible for detecting the most important features. The output of the convolution operation is known as a feature map, a convolved feature, or an activation map.

CHAPTER 6
SYSTEM IMPLEMENTATION

6.1 MODULE DESCRIPTION

6.1.1 IMPORT DATASET

The designed system for autonomous vehicle guidance can be successfully used to control the driving simulator. The method for detecting the road boundary provides useful data at a sufficiently high camera resolution in order to enable an adjustment of the steering angle by means of corresponding driving commands.

6.1.1.1 Train Data using CNN:

Step 1: Upload Dataset.

Step 2: The Input layer.

Step 3: Convolutional layer.

Step 4: Pooling layer.

Step 5: Convolutional layer and Pooling Layer.

Step 6: Dense layer.

Step 7: Logit Layer.

You ought to be comfortable with compact Convnets. The CNN is a stacking of alternating Conv2D (with Relu as an activation function) and MaxPooling2D layers, and you'll utilize the same overall structure.However, because you are working with larger images and a more challenging problem, you will need to expand your networks and add the Conv2D + MaxPooling2D stage. This increases the network's capability and further reduces the dimension of the feature maps so that they aren't too big when you get to the Flatten layer. Here, you start with the input of size 150 x 150, and immediately before the Flatten layer, you end up with feature maps of size 7 x 7.You'll terminate the network with a single unit (a Dense layer of size 1) and a sigmoid activation since you're tackling a binary-classification task. This unit will encode the likelihood that the network focuses on one class over another.

6.1.2 ROAD SURFACE AUTONOMOUS

The road surface can be comprised of light or dark pavements or combinations. An example of the variety of road, some roads shows a relatively simple scene with both solid line and dashed line lane markings. Lane position in this scene can be considered relatively easy because of the clearly defined markings and uniform road texture. But in other complex scene in which the road surface varies and markings consist of circular reflectors as well as solid lines the lane detection will not be an easy task. Furthermore, shadowing obscuring road markings makes the edge detection phase more complex. Along with the various types of markings and shadowing, weather conditions, and time of day can have a great impact on the visibility of the road surface.

6.1.3 BEHAVIOR DETECTION

Behavior detection is an important component of advanced driver assistance systems (ADAS) and behavior of vehicles, as it provides information about the road layout and the position of the vehicle within the lane, which is crucial for navigation and safety. The algorithms typically use a combination of computer vision techniques, such as edge detection, color filtering, and Hough transforms, to identify and track the lane markings in a road scene.

6.2 DIAGRAM

6.2.1 USE CASE DIAGRAM

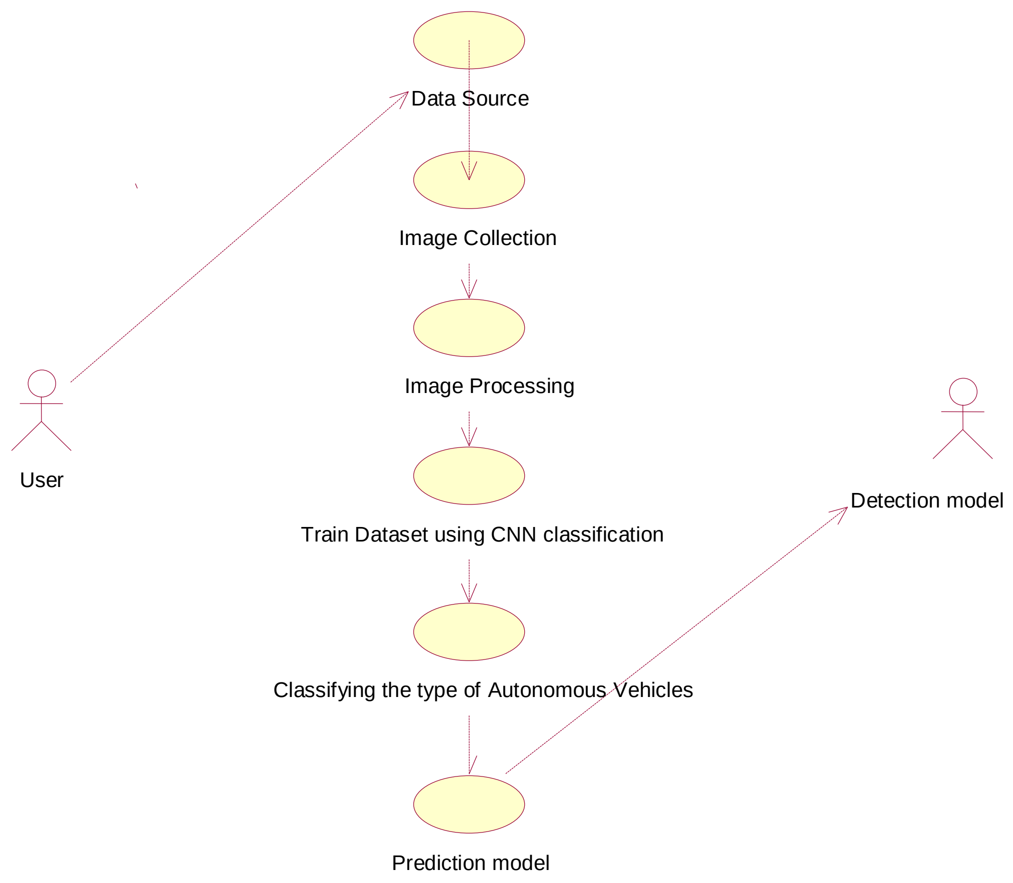

Data Source
Image Collection
Image Processing
User
Train Dataset using CNN classification
Detection model
Classifying the type of Autonomous Vehicles
Prediction model

Fig No 6.2.1-USE CASE DIAGRAM

6.2.2 SEQUENCE DIAGRAM

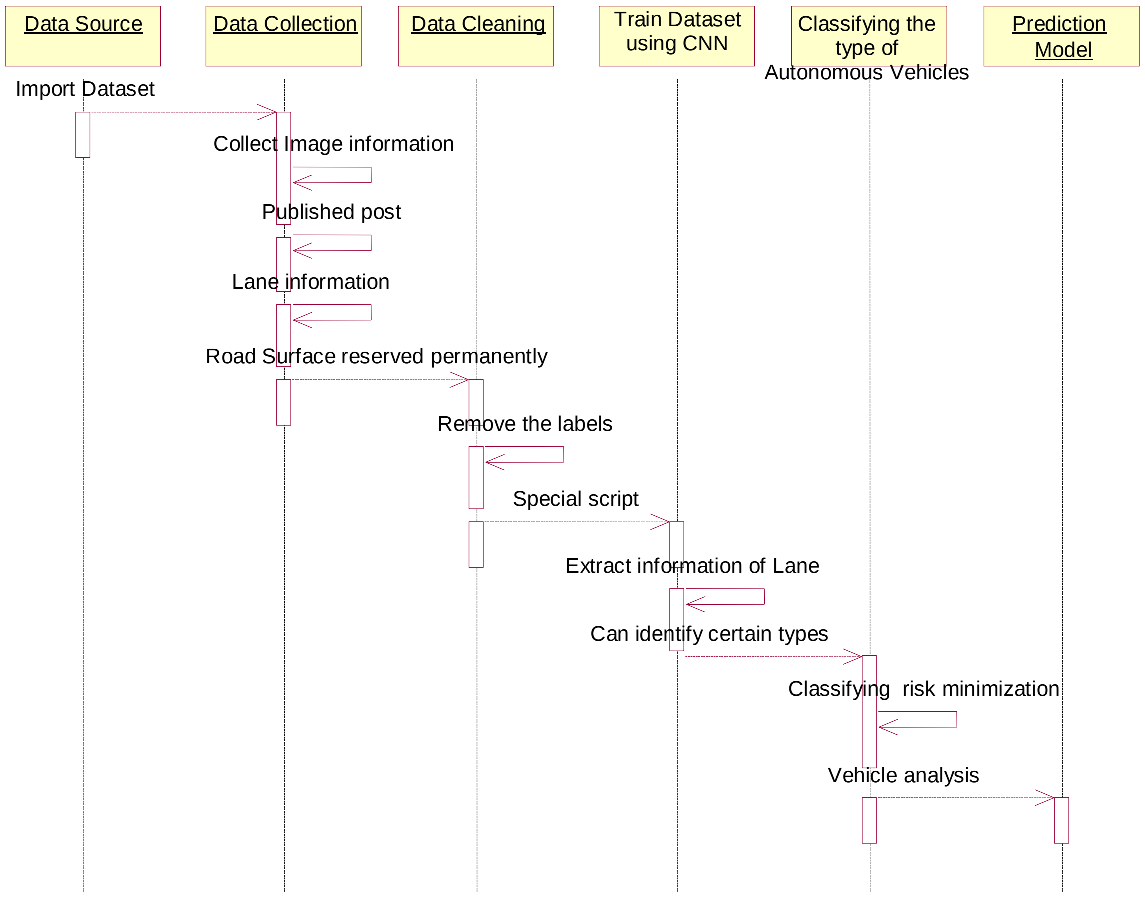

Data Source
Data Collection
Data Cleaning
Train Dataset using CNN
Classifying the type of Autonomous Vehicles
Prediction Model
Import Dataset
Collect Image information
Published post
Lane information
Road Surface reserved permanently
Remove the labels
Special script
Extract information of Lane
Can identify certain types
Classifying risk minimization
Vehicle analysis

Fig No 6.2.2-SEQUENCE DIAGRAM

6.2.3 COLLABORATIVE DIAGRAM

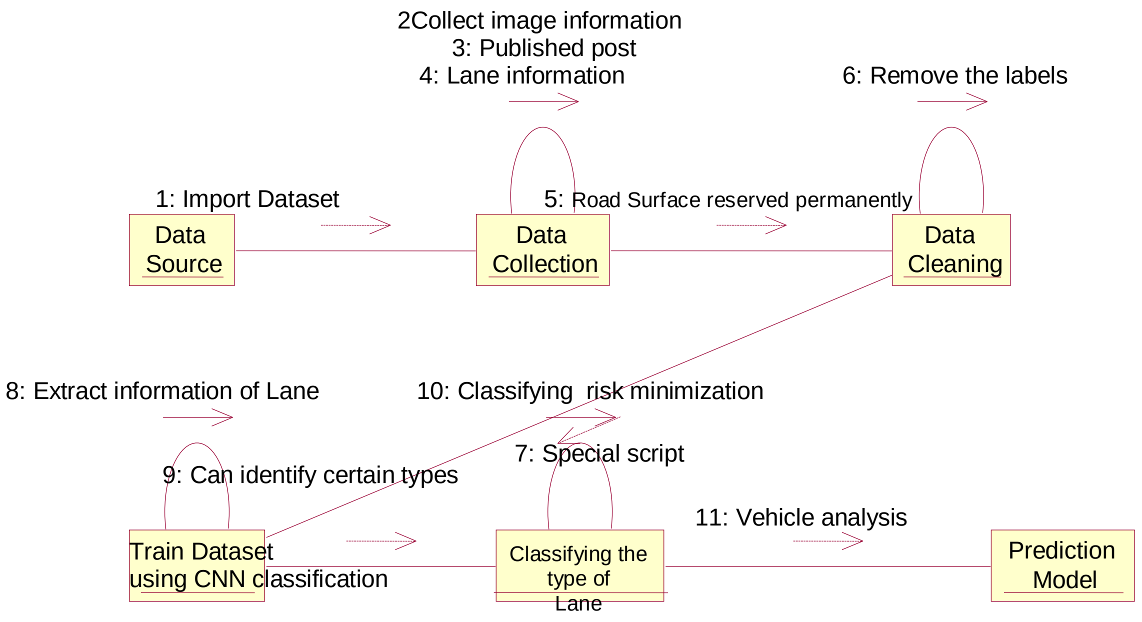

2Collect image information
3: Published post
4: Lane information
6: Remove the labels
1: Import Dataset
5: Road Surface reserved permanently
Data Source
Data Collection
Data Cleaning
8: Extract information of Lane
10: Classifying risk minimization
7: Special script
9: Can identify certain types
11: Vehicle analysis
Train Dataset using CNN classification
Classifying the type of Lane
Prediction Model

Fig No 6.2.3-COLLABORATIVE DIAGRAM

CHAPTER 7
SYSTEM DESIGN

7.1 INTRODUCTION

7.1.1 DATA MINING

Data mining is an interdisciplinary subfield of computer science. It is the computational process of discovering patterns in large data sets ("big data") involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step ,it involves database and data management aspects, data pre-processing, model and inference considerations-interestingness.Metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD. The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection, data preparation, nor result interpretation and reporting is part of the data mining step, but do belong to the overall KDD process as additional steps.These methods can, however, be used in creating new hypotheses to test against the larger data populations. With the fast development of networking, data storage, and the data collection capacity, Big Data are now rapidly expanding in all science and engineering domains, including physical, biological and biomedical sciences. This project presents a HACE theorem that characterizes the features of the Big Data revolution, and proposes a Big Data processing model, from the data mining perspective. This data-driven model involves aggregation of information sources, mining and analysis, user interest modeling, and security and privacy considerations. We analyze the challenging issues in the data-driven model and also in the Big Data revolution.

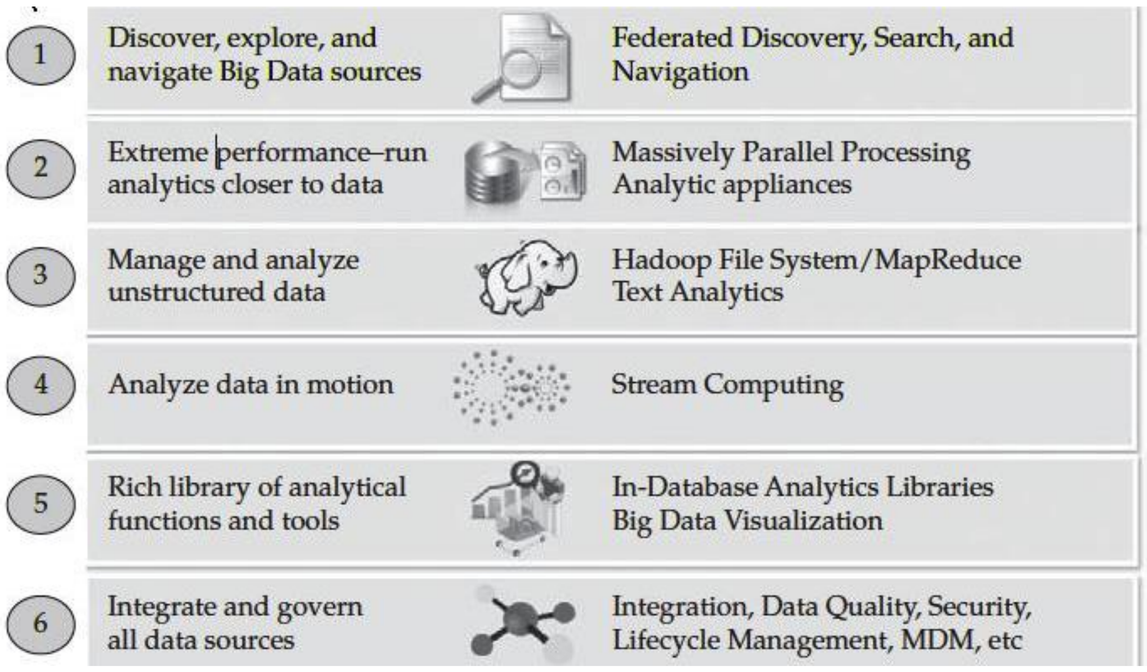
7.1.2 BIG DATA

Big data is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools. The challenges include capture, curation, storage, search, sharing, analysis, and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions.Put another way, big data is the realization of greater business intelligence by storing, processing, and analyzing data that was previously ignored due to the limitations of traditional data management technologies

7.1.2.1 THE FOUR DIMENSIONS OF BIG DATA

1. Volume: Large volumes of data
2. Velocity: Quickly moving data
3. Variety: structured, unstructured, images, etc.
4. Veracity: Trust and integrity is a challenge and a must and is important for big data just as for traditional relational DBs

7.1.2.2 THE BIG DATA PLATFORM MANIFESTO


1
Discover, explore, and navigate Big Data sources
Federated Discovery, Search, and Navigation
2
Extreme performance–run analytics closer to data
Massively Parallel Processing Analytic appliances
3
Manage and analyze unstructured data
Hadoop File System/MapReduce Text Analytics
4
Analyze data in motion
Stream Computing
5
Rich library of analytical functions and tools
In-Database Analytics Libraries Big Data Visualization
6
Integrate and govern all data sources
Integration, Data Quality, Security, Lifecycle Management, MDM, etc

7.1.1 Big Data Platform

7.1.2.3 SOME CONCEPTS

- No SQL (Not Only SQL): Databases that “move beyond” relational data models (i.e., no tables, limited or no use of SQL)
  - Focus on retrieval of data and appending new data (not necessarily tables)
  - Focus on key-value data stores that can be used to locate data objects
  - Focus on supporting storage of large quantities of unstructured data
  - SQL is not used for storage or retrieval of data
  - No ACID (atomicity, consistency, isolation, durability)

7.1.3 HADOOP

- Hadoop is a distributed file system and data processing engine that is designed to handle extremely high volumes of data in any structure.
- Hadoop has two components:
  - The Hadoop Distributed File System (HDFS), which supports data in structured relational form, in unstructured form, and in any form in between
  - The MapReduce programming paradigm for managing applications on multiple distributed servers
- The focus is on supporting redundancy, distributed architectures, and parallel processing

7.1.3.1 SOME HADOOP RELATED NAMES TO KNOW

- **Apache Avro**: designed for communication between Hadoop nodes through data serialization
- **Cassandra and Hbase**: a non-relational database designed for use with Hadoop
- **Hive:** a query language similar to SQL (HiveQL) but compatible with Hadoop
- **Mahout**: an AI tool designed for machine learning; that is, to assist with filtering data for analysis and exploration
- **Pig Latin**: A data-flow language and execution framework for parallel computation
- **ZooKeeper**: Keeps all the parts coordinated and working together

7.1.3.2 WHAT TO DO WITH THE DATA

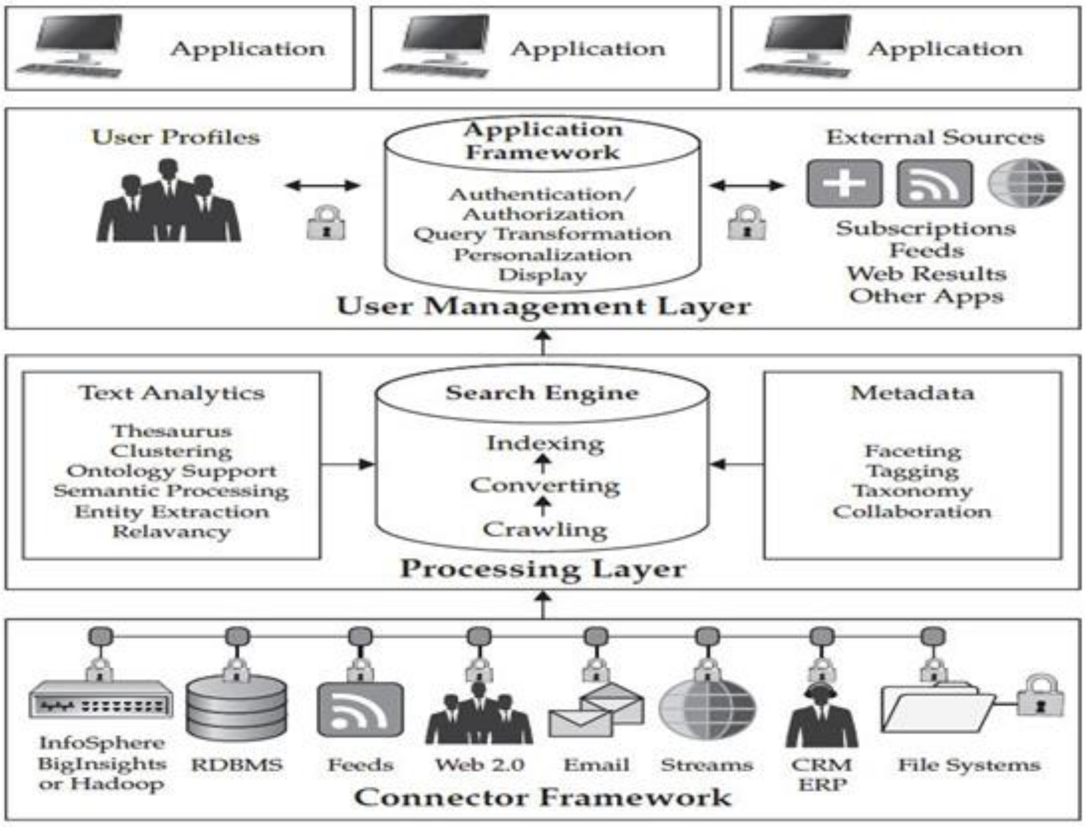


Fig No 7.1.2 Knowledge Discovery in Databases

The Knowledge Discovery in Databases (KDD) process is commonly defined with the stages:

(1) Selection

(2) Pre-processing

(3) Transformation

(4) Data Mining

(5) Interpretation/Evaluation.

It exists, however, in many variations on this theme, such as the Cross Industry Standard Process for Data Mining (CRISP-DM) which defines six phases:

(1) Business Understanding

(2) Data Understanding

(3) Data Preparation

(4) Modeling

(5) Evaluation

(6) Deployment

or a simplified process such as

(1) pre-processing,

(2) data mining,

(3) results validation.

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

7.1.3.3 USAGE AND EXTENSIONS

TensorFlow serves as the core platform and library for machine learning. TensorFlow's APIs use Keras to allow users to make their own machine learning models. In addition to building and training their model, TensorFlow can also help load the data to train the model, and deploy it using TensorFlow Serving.

Integrations

7.1.3.4 NUMPY

Numpy is one of the most popular Python data libraries, and TensorFlow offers integration and compatibility with its data structures. NumpyNDarrays, the library's native datatype, are automatically converted to TensorFlow Tensors in TF operations; the same is also true vice-versa. This allows for the two libraries to work in unison without requiring the user to write explicit data conversions. Moreover, the integration extends to memory optimization by having TF Tensors share the underlying memory representations of NumpyNDarrays whenever possible. ExtensionsTensorFlow also offers a variety of libraries and extensions to advance and extend the models and methods used.For example, TensorFlow Recommenders and TensorFlow Graphics are libraries for their respective functionalities in recommendation systems and graphics, TensorFlow Federated provides a framework for decentralized data, and TensorFlow Cloud allows users to directly interact with Google Cloud to integrate their local code to Google Cloud. Other add-ons, libraries, and frameworks include TensorFlow Model Optimization, TensorFlow Probability, TensorFlow Quantum, and TensorFlow Decision Forests.

CHAPTER 8
SYSTEM TESTING

8.1 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that theSoftware system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.2 TYPES OF TESTS

8.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.2.3 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifyBusiness process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.2.4 SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.2.5 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.2.6 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .

8.3TESTING IMPLEMENTATION

8.3.1 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.3.2 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 9
CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

Although deep learning-based behavior prediction solutions have shown promising performance, especially in complex driving scenarios, by utilizing sophisticated input representation and output type, there are several open challenges that need to be addressed to enable their adoption in autonomous driving applications. Particularly, while most of existing solutions considered the interaction among vehicles, factors such as environment conditions and set of traffic rules are not directly inputted to the prediction model. In addition, practical limitations such as sensor impairments and limited computational resources have not been fully taken into account

9.2 FUTURE ENHANCEMENT

In addition to the vehicle’s states and scene information which both are usually considered in recent works, other visual and auditory data of vehicles, like vehicle’s signaling lights and vehicle horn can also be used to infer about its future behavior.

CHAPTER 10
APPENDICES

10.1 SAMPLE CODE

A.PROCESS.PY

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from camera_calibration import calib, undistort
from threshold import get_combined_gradients, get_combined_hls, combine_grad_hls
from line import Line, get_perspective_transform, get_lane_lines_img, illustrate_driving_lane, illustrate_info_panel, illustrate_driving_lane_with_topdownview
from moviepy.editor import VideoFileClip
input_type = 'image'
#input_type = 'video'
#input_type = 'frame_by_frame'
input_name = 'test_images/test3.jpg'
#input_name = 'test_images/calibration1.jpg'
#input_name = 'project_video.mp4'
#input_name = 'challenge_video.mp4'
#input_name = 'harder_challenge_video.mp4'
# If input_type is `image`, select whether you'd like to save intermediate images or not.
save_img = True
left_line = Line()
right_line = Line()
th_sobelx, th_sobely, th_mag, th_dir = (35, 100), (30, 255), (30, 255), (0.7, 1.3)
th_h, th_l, th_s = (10, 100), (0, 60), (85, 255)
mtx, dist = calib()

def pipeline(frame):
    # Correcting for Distortion
undist_img = undistort(frame, mtx, dist)
    # resize video
undist_img = cv2.resize(undist_img, None, fx=1 / 2, fy=1 / 2, interpolation=cv2.INTER_AREA)
    rows, cols = undist_img.shape[:2]
combined_gradient = get_combined_gradients(undist_img, th_sobelx, th_sobely, th_mag, th_dir)
combined_hls = get_combined_hls(undist_img, th_h, th_l, th_s)
combined_result = combine_grad_hls(combined_gradient, combined_hls)
c_rows, c_cols = combined_result.shape[:2]
    s_LTop2, s_RTop2 = [c_cols / 2 - 24, 5], [c_cols / 2 + 24, 5]
    s_LBot2, s_RBot2 = [110, c_rows], [c_cols - 110, c_rows]
src = np.float32([s_LBot2, s_LTop2, s_RTop2, s_RBot2])
dst = np.float32([(170, 720), (170, 0), (550, 0), (550, 720)])
warp_img, M, Minv = get_perspective_transform(combined_result, src, dst, (720, 720))

searching_img = get_lane_lines_img(warp_img, left_line, right_line)
w_comb_result, w_color_result = illustrate_driving_lane(searching_img, left_line, right_line)
    # Drawing the lines back down onto the road
color_result = cv2.warpPerspective(w_color_result, Minv, (c_cols, c_rows))
lane_color = np.zeros_like(undist_img)
lane_color[220:rows - 12, 0:cols] = color_result
    # Combine the result with the original image
    result = cv2.addWeighted(undist_img, 1, lane_color, 0.3, 0)
info_panel, birdeye_view_panel = np.zeros_like(result),  np.zeros_like(result)
info_panel[5:110, 5:325] = (255, 255, 255)
birdeye_view_panel[5:110, cols-111:cols-6] = (255, 255, 255)

info_panel = cv2.addWeighted(result, 1, info_panel, 0.2, 0)
birdeye_view_panel = cv2.addWeighted(info_panel, 1, birdeye_view_panel, 0.2, 0)
road_map = illustrate_driving_lane_with_topdownview(w_color_result, left_line, right_line)
birdeye_view_panel[10:105, cols-106:cols-11] = road_map
```

```
birdeye_view_panel = illustrate_info_panel(birdeye_view_panel, left_line, right_line)
    return birdeye_view_panel
if __name__ == '__main__':
    # For debugging Frame by Frame, using cv2.imshow()
    if input_type == 'frame_by_frame':
        cap = cv2.VideoCapture(input_name)
frame_num = -1
        while (cap.isOpened()):
            _, frame = cap.read()
frame_num += 1   # increment frame_num, used for naming saved images
            # Correcting for Distortion
undist_img = undistort(frame, mtx, dist)
            # resize video
undist_img = cv2.resize(undist_img, None, fx=1 / 2, fy=1 / 2, interpolation=cv2.INTER_AREA)
            rows, cols = undist_img.shape[:2]
combined_gradient = get_combined_gradients(undist_img, th_sobelx, th_sobely, th_mag, th_dir)
combined_hls = get_combined_hls(undist_img, th_h, th_l, th_s)
combined_result = combine_grad_hls(combined_gradient, combined_hls)
c_rows, c_cols = combined_result.shape[:2]
            s_LTop2, s_RTop2 = [c_cols / 2 - 24, 5], [c_cols / 2 + 24, 5]
            s_LBot2, s_RBot2 = [110, c_rows], [c_cols - 110, c_rows]
src = np.float32([s_LBot2, s_LTop2, s_RTop2, s_RBot2])
dst = np.float32([(170, 720), (170, 0), (550, 0), (550, 720)])
warp_img, M, Minv = get_perspective_transform(combined_result, src, dst, (720, 720))
searching_img = get_lane_lines_img(warp_img, left_line, right_line)
w_comb_result, w_color_result = illustrate_driving_lane(searching_img, left_line, right_line)
            # Drawing the lines back down onto the road
color_result = cv2.warpPerspective(w_color_result, Minv, (c_cols, c_rows))
lane_color = np.zeros_like(undist_img)
lane_color[220:rows - 12, 0:cols] = color_result
```

```
            # Combine the result with the original image
            result = cv2.addWeighted(undist_img, 1, lane_color, 0.3, 0)
info_panel, birdeye_view_panel = np.zeros_like(result),  np.zeros_like(result)
info_panel[5:110, 5:325] = (255, 255, 255)
birdeye_view_panel[5:110, cols-111:cols-6] = (255, 255, 255)
info_panel = cv2.addWeighted(result, 1, info_panel, 0.2, 0)
birdeye_view_panel = cv2.addWeighted(info_panel, 1, birdeye_view_panel, 0.2, 0)
road_map = illustrate_driving_lane_with_topdownview(w_color_result, left_line, right_line)
birdeye_view_panel[10:105, cols-106:cols-11] = road_map
birdeye_view_panel = illustrate_info_panel(birdeye_view_panel, left_line, right_line)
            # test/debug
            cv2.imshow('road info', birdeye_view_panel)
            # out.write(frame)
            if cv2.waitKey(1) & 0xFF == ord('s'):
                cv2.waitKey(0)
            #if cv2.waitKey(1) & 0xFF == ord('r'):
            #    cv2.imwrite('check1.jpg', undist_img)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
cap.release()
        cv2.destroyAllWindows()
    # If working with images, don't use moviepy
elifinput_type == 'image':
img = cv2.imread(input_name)
undist_img = undistort(img, mtx, dist)
        if save_img == True:
            cv2.imwrite('./output_images/01_undist_img.png', undist_img)

undist_img = cv2.resize(undist_img, None, fx=1 / 2, fy=1 / 2, interpolation=cv2.INTER_AREA)
        rows, cols = undist_img.shape[:2]
        if save_img == True:
            cv2.imwrite('./output_images/02_resized_img.png', undist_img)

combined_gradient = get_combined_gradients(undist_img, th_sobelx, th_sobely, th_mag, th_dir)
        if save_img == True:
            cv2.imwrite('./output_images/03_combined_gradient_img.png', combined_gradient)

combined_hls = get_combined_hls(undist_img, th_h, th_l, th_s)
        if save_img == True:
            cv2.imwrite('./output_images/04_combined_hls_img.png', combined_hls)

combined_result = combine_grad_hls(combined_gradient, combined_hls)
        if save_img == True:
            cv2.imwrite('./output_images/05_combined_thresh_result_img.png', combined_result)



c_rows, c_cols = combined_result.shape[:2]
        s_LTop2, s_RTop2 = [c_cols / 2 - 24, 5], [c_cols / 2 + 24, 5]
        s_LBot2, s_RBot2 = [110, c_rows], [c_cols - 110, c_rows]

src = np.float32([s_LBot2, s_LTop2, s_RTop2, s_RBot2])
dst = np.float32([(170, 720), (170, 0), (550, 0), (550, 720)])

warp_img, M, Minv = get_perspective_transform(combined_result, src, dst, (720, 720))
        if save_img == True:
            cv2.imwrite('./output_images/07_warped_img.png', warp_img)

searching_img = get_lane_lines_img(warp_img, left_line, right_line)
        if save_img == True:
            cv2.imwrite('./output_images/08_searching_img.png', searching_img)

w_comb_result, w_color_result = illustrate_driving_lane(searching_img, left_line, right_line)
```

```
        if save_img == True:
            cv2.imwrite('./output_images/09_w_comb_result.png', w_comb_result)
        if save_img == True:
            cv2.imwrite('./output_images/10_w_color_result_img.png', w_color_result)

        # Drawing the lines back down onto the road
color_result = cv2.warpPerspective(w_color_result, Minv, (c_cols, c_rows))
        if save_img == True:
            cv2.imwrite('./output_images/11_color_result.png', color_result)

comb_result = np.zeros_like(undist_img)
comb_result[220:rows - 12, 0:cols] = color_result
        if save_img == True:
            cv2.imwrite('./output_images/12_color_result_crop.png', color_result)

        # Combine the result with the original image
        result = cv2.addWeighted(undist_img, 1, comb_result, 0.3, 0)
        if save_img == True:
            cv2.imwrite('./output_images/13_final_result.png', result)

        cv2.imshow('result',result)
        cv2.waitKey(0)



    # If working with video mode, use moviepy and process each frame and save the video.
elifinput_type == 'video':
white_output = "./output_videos/video_out.mp4"
        frame = VideoFileClip(input_name)
white_clip = frame.fl_image(pipeline)
white_clip.write_videofile(white_output, audio=False)
```

B. PROCESS_VIDEO.PY

```
import cv2
import numpy as np
cap = cv2.VideoCapture('project_video.mp4')

# Check if camera opened successfully
if (cap.isOpened()== False):
print("Error opening video stream or file")

# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, frame = cap.read()
  if ret == True:

    # Display the resulting frame
    cv2.imshow('Frame',frame)

    # Press Q on keyboard to  exit
    if cv2.waitKey(25) & 0xFF == ord('q'):
      break

  # Break the loop
  else:
    break

# When everything done, release the video capture object
cap.release()

# Closes all the frames
```

```
cv2.destroyAllWindows()
import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from camera_calibration import calib, undistort
from threshold import get_combined_gradients, get_combined_hls, combine_grad_hls
from line import Line, get_perspective_transform, get_lane_lines_img, illustrate_driving_lane,
illustrate_info_panel, illustrate_driving_lane_with_topdownview
from moviepy.editor import VideoFileClip
#input_type = 'image'
input_type = 'video'
#input_type = 'frame_by_frame'

input_name = 'test_video/project_video.mp4'
#input_name = 'test_images/calibration1.jpg'
#input_name = 'project_video.mp4'
#input_name = 'challenge_video.mp4'
#input_name = 'harder_challenge_video.mp4'

# If input_type is `image`, select whether you'd like to save intermediate images or not.
save_img = True

left_line = Line()
right_line = Line()

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
#   Tune Parameters for different inputs        #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
th_sobelx, th_sobely, th_mag, th_dir = (35, 100), (30, 255), (30, 255), (0.7, 1.3)
th_h, th_l, th_s = (10, 100), (0, 60), (85, 255)
```

```
# camera matrix & distortion coefficient
mtx, dist = calib()



def pipeline(frame):
   # Correcting for Distortion
undist_img = undistort(frame, mtx, dist)

   # resize video
undist_img = cv2.resize(undist_img, None, fx=1 / 2, fy=1 / 2, interpolation=cv2.INTER_AREA)
   rows, cols = undist_img.shape[:2]

combined_gradient = get_combined_gradients(undist_img, th_sobelx, th_sobely, th_mag, th_dir)

combined_hls = get_combined_hls(undist_img, th_h, th_l, th_s)

combined_result = combine_grad_hls(combined_gradient, combined_hls)

c_rows, c_cols = combined_result.shape[:2]
   s_LTop2, s_RTop2 = [c_cols / 2 - 24, 5], [c_cols / 2 + 24, 5]
   s_LBot2, s_RBot2 = [110, c_rows], [c_cols - 110, c_rows]

src = np.float32([s_LBot2, s_LTop2, s_RTop2, s_RBot2])
dst = np.float32([(170, 720), (170, 0), (550, 0), (550, 720)])

warp_img, M, Minv = get_perspective_transform(combined_result, src, dst, (720, 720))

searching_img = get_lane_lines_img(warp_img, left_line, right_line)

w_comb_result, w_color_result = illustrate_driving_lane(searching_img, left_line, right_line)
```

```
    # Drawing the lines back down onto the road
color_result = cv2.warpPerspective(w_color_result, Minv, (c_cols, c_rows))
lane_color = np.zeros_like(undist_img)
lane_color[220:rows - 12, 0:cols] = color_result

    # Combine the result with the original image
    result = cv2.addWeighted(undist_img, 1, lane_color, 0.3, 0)

info_panel, birdeye_view_panel = np.zeros_like(result),  np.zeros_like(result)
info_panel[5:110, 5:325] = (255, 255, 255)
birdeye_view_panel[5:110, cols-111:cols-6] = (255, 255, 255)

info_panel = cv2.addWeighted(result, 1, info_panel, 0.2, 0)
birdeye_view_panel = cv2.addWeighted(info_panel, 1, birdeye_view_panel, 0.2, 0)
road_map = illustrate_driving_lane_with_topdownview(w_color_result, left_line, right_line)
birdeye_view_panel[10:105, cols-106:cols-11] = road_map
birdeye_view_panel = illustrate_info_panel(birdeye_view_panel, left_line, right_line)

    return birdeye_view_panel



if __name__ == '__main__':

    # For debugging Frame by Frame, using cv2.imshow()
    if input_type == 'frame_by_frame':
        cap = cv2.VideoCapture(input_name)

frame_num = -1

        while (cap.isOpened()):
```

```
        _, frame = cap.read()

frame_num += 1   # increment frame_num, used for naming saved images

        # Correcting for Distortion
undist_img = undistort(frame, mtx, dist)
        # resize video
undist_img = cv2.resize(undist_img, None, fx=1 / 2, fy=1 / 2, interpolation=cv2.INTER_AREA)
        rows, cols = undist_img.shape[:2]

combined_gradient = get_combined_gradients(undist_img, th_sobelx, th_sobely, th_mag, th_dir)

combined_hls = get_combined_hls(undist_img, th_h, th_l, th_s)

combined_result = combine_grad_hls(combined_gradient, combined_hls)

c_rows, c_cols = combined_result.shape[:2]
        s_LTop2, s_RTop2 = [c_cols / 2 - 24, 5], [c_cols / 2 + 24, 5]
        s_LBot2, s_RBot2 = [110, c_rows], [c_cols - 110, c_rows]

src = np.float32([s_LBot2, s_LTop2, s_RTop2, s_RBot2])
dst = np.float32([(170, 720), (170, 0), (550, 0), (550, 720)])

warp_img, M, Minv = get_perspective_transform(combined_result, src, dst, (720, 720))

searching_img = get_lane_lines_img(warp_img, left_line, right_line)

w_comb_result, w_color_result = illustrate_driving_lane(searching_img, left_line, right_line)

        # Drawing the lines back down onto the road
color_result = cv2.warpPerspective(w_color_result, Minv, (c_cols, c_rows))
```

```
lane_color = np.zeros_like(undist_img)
lane_color[220:rows - 12, 0:cols] = color_result

            # Combine the result with the original image
            result = cv2.addWeighted(undist_img, 1, lane_color, 0.3, 0)

info_panel, birdeye_view_panel = np.zeros_like(result),  np.zeros_like(result)
info_panel[5:110, 5:325] = (255, 255, 255)
birdeye_view_panel[5:110, cols-111:cols-6] = (255, 255, 255)

info_panel = cv2.addWeighted(result, 1, info_panel, 0.2, 0)
birdeye_view_panel = cv2.addWeighted(info_panel, 1, birdeye_view_panel, 0.2, 0)
road_map = illustrate_driving_lane_with_topdownview(w_color_result, left_line, right_line)
birdeye_view_panel[10:105, cols-106:cols-11] = road_map
birdeye_view_panel = illustrate_info_panel(birdeye_view_panel, left_line, right_line)



            # test/debug
            cv2.imshow('road info', birdeye_view_panel)
            # out.write(frame)
            if cv2.waitKey(1) & 0xFF == ord('s'):
                cv2.waitKey(0)
            #if cv2.waitKey(1) & 0xFF == ord('r'):
            #    cv2.imwrite('check1.jpg', undist_img)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

cap.release()
        cv2.destroyAllWindows()
    # If working with images, don't use moviepy
elifinput_type == 'image':
```

```
img = cv2.imread(input_name)
undist_img = undistort(img, mtx, dist)
        if save_img == True:
            cv2.imwrite('./output_images/01_undist_img.png', undist_img)

undist_img = cv2.resize(undist_img, None, fx=1 / 2, fy=1 / 2, interpolation=cv2.INTER_AREA)
        rows, cols = undist_img.shape[:2]
        if save_img == True:
            cv2.imwrite('./output_images/02_resized_img.png', undist_img)
combined_gradient = get_combined_gradients(undist_img, th_sobelx, th_sobely, th_mag, th_dir)
        if save_img == True:
            cv2.imwrite('./output_images/03_combined_gradient_img.png', combined_gradient)

combined_hls = get_combined_hls(undist_img, th_h, th_l, th_s)
        if save_img == True:
            cv2.imwrite('./output_images/04_combined_hls_img.png', combined_hls)

combined_result = combine_grad_hls(combined_gradient, combined_hls)
        if save_img == True:
            cv2.imwrite('./output_images/05_combined_thresh_result_img.png', combined_result)
c_rows, c_cols = combined_result.shape[:2]
        s_LTop2, s_RTop2 = [c_cols / 2 - 24, 5], [c_cols / 2 + 24, 5]
        s_LBot2, s_RBot2 = [110, c_rows], [c_cols - 110, c_rows]

src = np.float32([s_LBot2, s_LTop2, s_RTop2, s_RBot2])
dst = np.float32([(170, 720), (170, 0), (550, 0), (550, 720)])

warp_img, M, Minv = get_perspective_transform(combined_result, src, dst, (720, 720))
        if save_img == True:
            cv2.imwrite('./output_images/07_warped_img.png', warp_img)
```

```
searching_img = get_lane_lines_img(warp_img, left_line, right_line)
        if save_img == True:
            cv2.imwrite('./output_images/08_searching_img.png', searching_img)

w_comb_result, w_color_result = illustrate_driving_lane(searching_img, left_line, right_line)
        if save_img == True:
            cv2.imwrite('./output_images/09_w_comb_result.png', w_comb_result)
        if save_img == True:
            cv2.imwrite('./output_images/10_w_color_result_img.png', w_color_result)

        # Drawing the lines back down onto the road
color_result = cv2.warpPerspective(w_color_result, Minv, (c_cols, c_rows))
        if save_img == True:
            cv2.imwrite('./output_images/11_color_result.png', color_result)

comb_result = np.zeros_like(undist_img)
comb_result[220:rows - 12, 0:cols] = color_result
        if save_img == True:
            cv2.imwrite('./output_images/12_color_result_crop.png', color_result)

        # Combine the result with the original image
        result = cv2.addWeighted(undist_img, 1, comb_result, 0.3, 0)
        if save_img == True:
            cv2.imwrite('./output_images/13_final_result.png', result)

        cv2.imshow('result',result)
        cv2.waitKey(0)
    # If working with video mode, use moviepy and process each frame and save the video.
elifinput_type == 'video':
white_output = "./output_videos/video_out.mp4"
        frame = VideoFileClip(input_name)
```

```
white_clip = frame.fl_image(pipeline)
white_clip.write_videofile(white_output, audio=False)
import cv2
import numpy as np
cap = cv2.VideoCapture('./output_videos/video_out.mp4')

# Check if camera opened successfully
if (cap.isOpened()== False):
print("Error opening video stream or file")

# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, frame = cap.read()
  if ret == True:
    # Display the resulting frame
    cv2.imshow('Frame',frame)
    # Press Q on keyboard to  exit
    if cv2.waitKey(25) & 0xFF == ord('q'):
      break
  # Break the loop
  else:
    break
# When everything done, release the video capture object
cap.release()
# Closes all the frames
cv2.destroyAllWindows()
```

10.2 SCREENSHOTS

10.2.1 INPUT PAGE

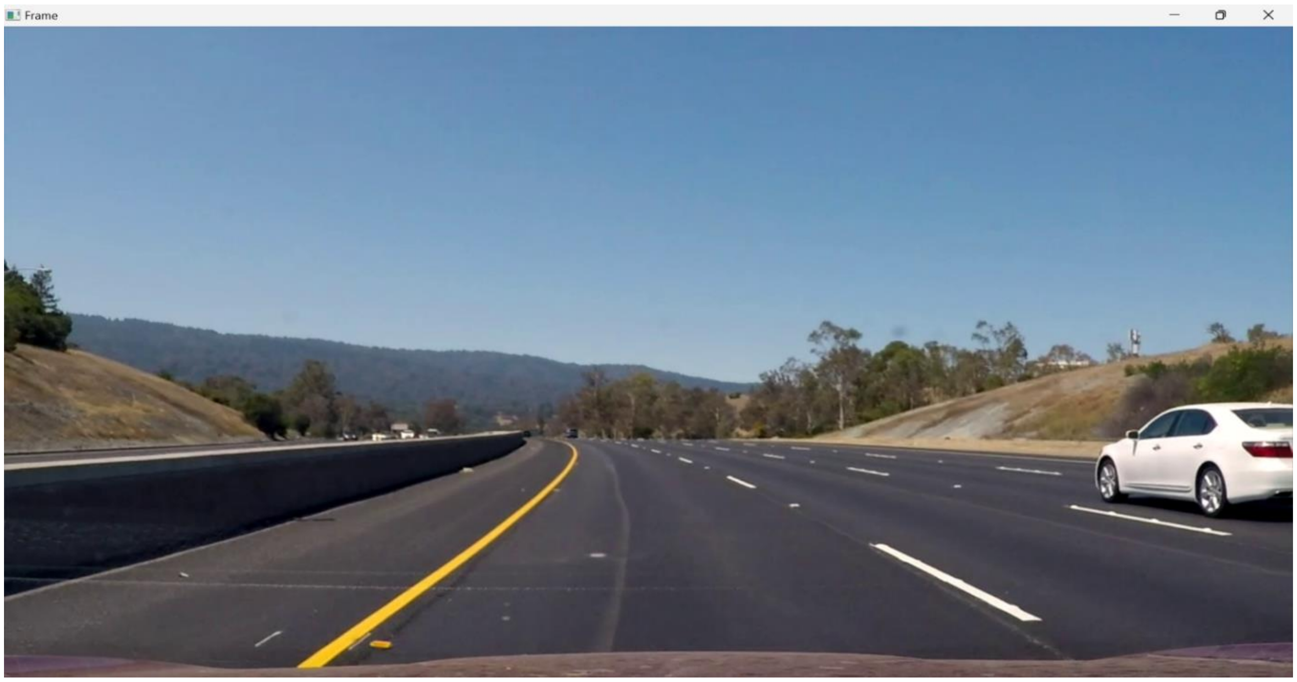
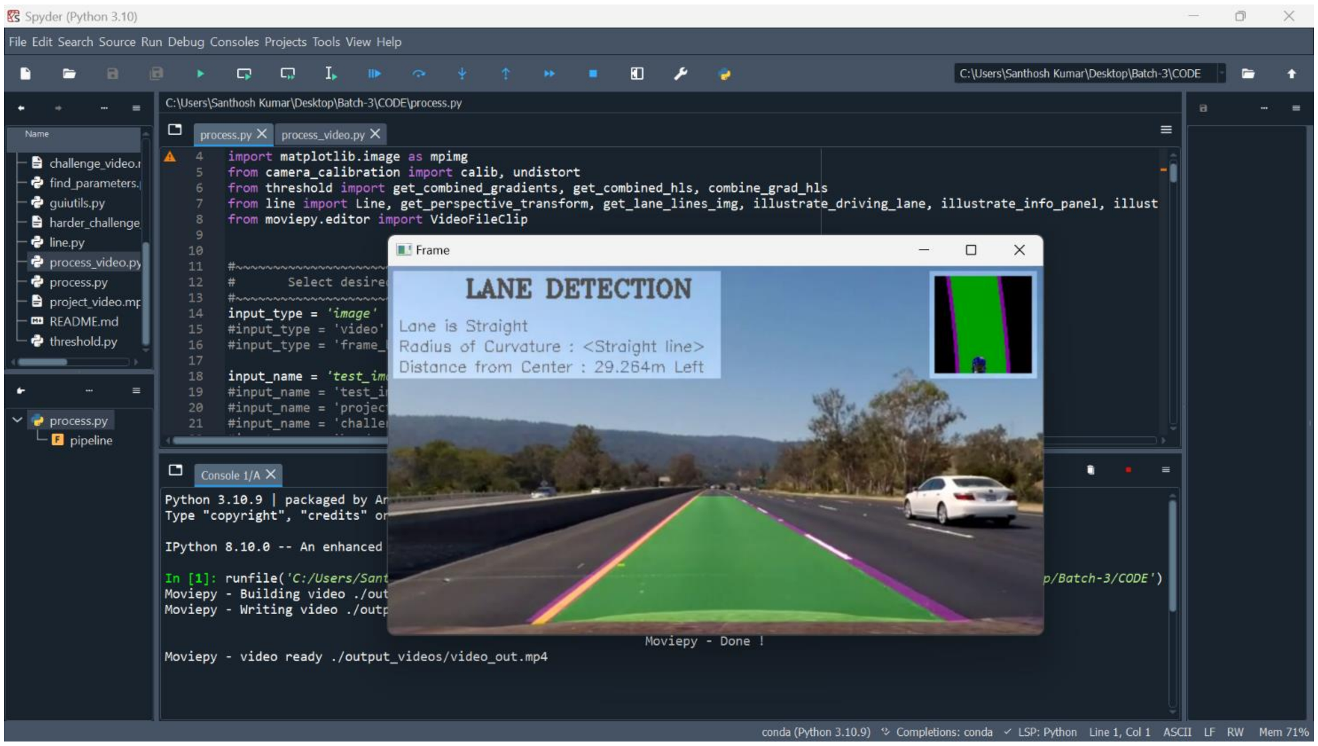


Fig No 10.2.1 INPUT VIDEO IMAGE

In this project we given a video as an input for model building by means of model training. This video should clearly shows the front view of the road by any kind of video capturing equipment.

10.2.2 OUTPUT PAGE



```
import matplotlib.image as mpimg
from camera_calibration import calib, undistort
from threshold import get_combined_gradients, get_combined_hls, combine_grad_hls
from line import Line, get_perspective_transform, get_lane_lines_img, illustrate_driving_lane, illustrate_info_panel, illust
from moviepy.editor import VideoFileClip


#~~~~~~~~~~~~~~~~~~~~~~~
#       Select desire
#~~~~~~~~~~~~~~~~~~~~~~~
input_type = 'image'
#input_type = 'video'
#input_type = 'frame_

input_name = 'test_im
#input_name = 'test_i
#input_name = 'projec
#input_name = 'challe
```

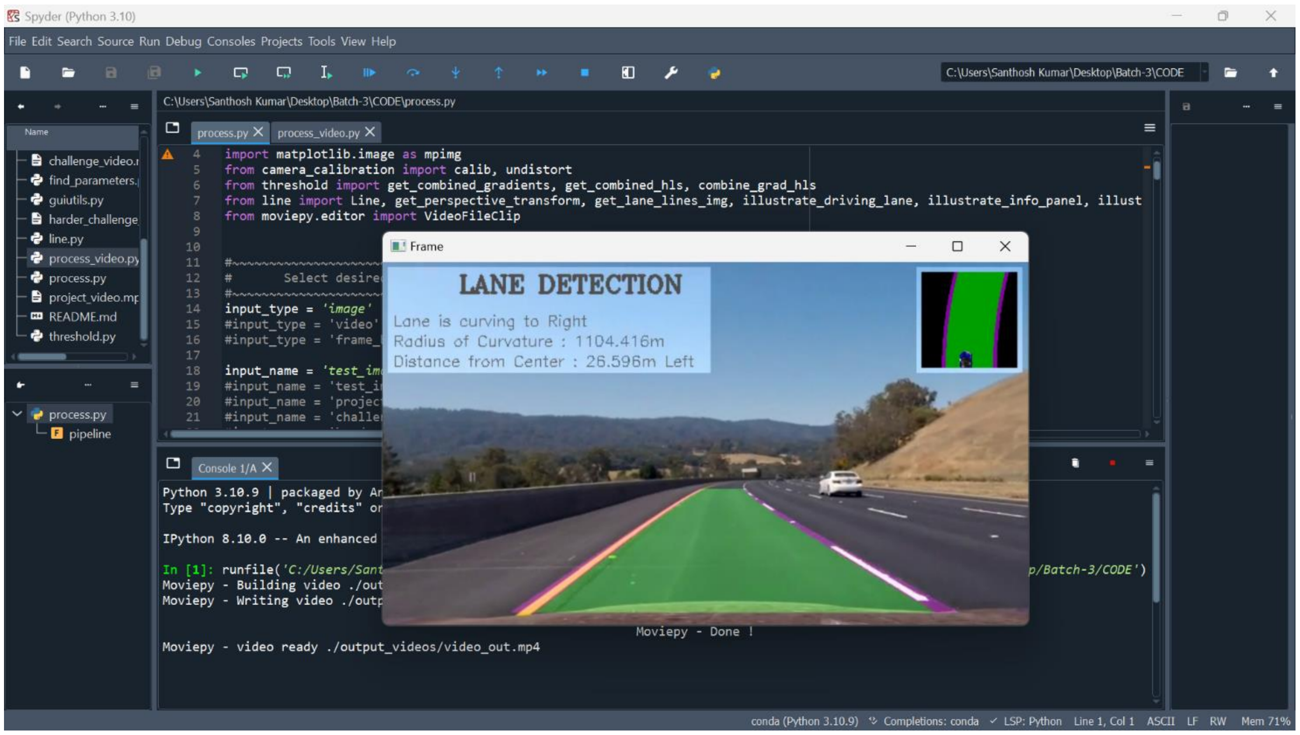
```
Python 3.10.9 | packaged by Ar
Type "copyright", "credits" or

IPython 8.10.0 -- An enhanced

In [1]: runfile('C:/Users/Sant                                                    p/Batch-3/CODE')
Moviepy - Building video ./out
Moviepy - Writing video ./outp

                                        Moviepy - Done !
Moviepy - video ready ./output_videos/video_out.mp4
```

LANE DETECTION

Lane is Straight
Radius of Curvature : <Straight line>
Distance from Center : 29.264m Left

Fig No 10.2.2 OUTPUT IMAGE PREDICT THE STRAIGHT LANE

This is the output of our project, here the trained model detect the lane in front of the car with distinct green color with direction, radius of curvature and the distance from the centre. In this output the image shows straight lane detection.


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Santhosh Kumar\Desktop\Batch-3\CODE
C:\Users\Santhosh Kumar\Desktop\Batch-3\CODE\process.py
process.py
process_video.py
challenge_video.
find_parameters.
guiutils.py
harder_challenge
line.py
process_video.py
process.py
project_video.mp
README.md
threshold.py
process.py
pipeline
Console 1/A
Frame
conda (Python 3.10.9) Completions: conda LSP: Python Line 1, Col 1 ASCII LF RW Mem 71%

```
import matplotlib.image as mpimg
from camera_calibration import calib, undistort
from threshold import get_combined_gradients, get_combined_hls, combine_grad_hls
from line import Line, get_perspective_transform, get_lane_lines_img, illustrate_driving_lane, illustrate_info_panel, illust
from moviepy.editor import VideoFileClip


#~~~~~~~~~~~~~~~~~~~~~~
#        Select desire
#~~~~~~~~~~~~~~~~~~~~~~
input_type = 'image'
#input_type = 'video'
#input_type = 'frame_

input_name = 'test_im
#input_name = 'test_i
#input_name = 'projec
#input_name = 'challe
```

```
LANE DETECTION
Lane is curving to Right
Radius of Curvature : 1104.416m
Distance from Center : 26.596m Left
```

```
Python 3.10.9 | packaged by An
Type "copyright", "credits" or

IPython 8.10.0 -- An enhanced

In [1]: runfile('C:/Users/Sant                                                          p/Batch-3/CODE')
Moviepy - Building video ./out
Moviepy - Writing video ./outp

                                             Moviepy - Done !
Moviepy - video ready ./output_videos/video_out.mp4
```

Fig No 10.2.3 OUTPUT IMAGE PREDICT THE RIGHT CURVING LANE

This is the output of our project, here the trained model detect the lane in front of the car with distinct green color with direction, radius of curvature and the distance from the centre. In this output the image shows right curving lane detection.

10.3 REFERENCES

1) A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese Jun. 2016,, “Social LSTM: Human trajectory prediction in crowded spaces,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 961–971.

2) A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi,Jun. 2018, “Social GAN: Socially acceptable.trajectories with generative adversarial networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 2255–2264

3) A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, 2020 “Human motion trajectory prediction: A survey,” Int. J. Robot. Res., vol. 39, no. 8, pp. 895–935.

4) B. T. Morris and M. Trivedi, 2013, “Understanding vehicular traffic behaviour from video: A survey of unsupervised approaches,” J. Electron. Imag., vol. 22, no. 4, pp. 1–16, doi: 10.1117/1.JEI.22.4.041113.

5) D. Helbing and P. Molnár,1995 “Social force model for pedestrian dynamics,” Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 51, no. 5, p. 4282.

6) M. S. Shirazi and B. T. Morris, Jan. 2017 “Looking at intersections: A survey of intersection monitoring,behaviour and safety analysis of recent studies,” IEEE Trans. Intell. Transp. Syst., vol. 18, no. 1, pp. 4–24.

7) P. V. K. Borges, N. Conci, and A. Cavallaro,Nov. 2013 “Video-based human behaviour understanding: A survey,” IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 11, pp. 1993–2008.

8) T. Fernando, S. Denman, S. Sridharan, and C. Fookes, Dec. 2018,“Soft + hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection,” Neural Netw., vol. 108, pp. 466–478,.[Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608018302648.

9) T.Hirakawa, T. Yamashita, T. Tamaki, and H. Fujiyoshi, Springer-2018, “Survey on vision-based path prediction,” in Distributed, Ambient and Pervasive Interactions: Technologies and Contexts, N. Streitz and S. Konomi, Eds. Cham, Switzerland: pp. 48–64.

10) UK Department for Transport. (May 2016). Research on the Impacts of Connected and Autonomous Vehicles (CAVs) on Traffic Flow: Summary Report. [Online].Available: https://assets.publishing. service.gov.uk/government/uploads/system/uploads/attachment_data/file/530091/impacts-of-connected-and-autonomous-vehicles-on-trafficflow-summary-report.pdf