Chapter 4

SYSTEM DESIGN

4 SYSTEM ARCHITECTURE

A **system architecture** is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A system architecture can consist of system and the sub-systems developed, that will work together to implement the overall system.

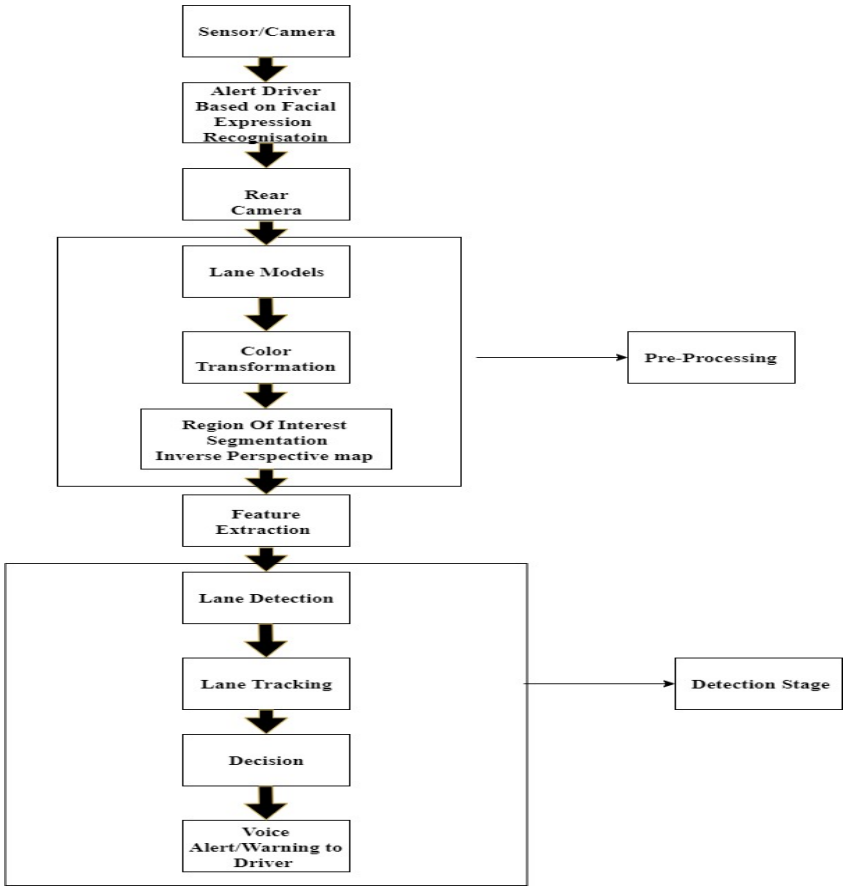

Fig 4.1: System Architecture Diagram

In the above diagram lane change assistant system must detect the frontal lanes and discover the vehicles around the test vehicle. A vision system is utilized including three cameras, two of them are under right and left wing mirrors, one is equipped on the

windscreen of the vehicle. The images from the cameras are used to detect the lanes, detect the vehicles and detect the driver facial expression. The technique is used in facial recognition is support vector machines, canny edge and hough transform techniques are used in lane detection and haar like cascade classifier technique is used in vehicle detection.

4.1 Data flow diagram

A **data-flow diagram** (DFD) is a way of representing a flow of a data of a process or a system. The data flow diagram also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart**.**

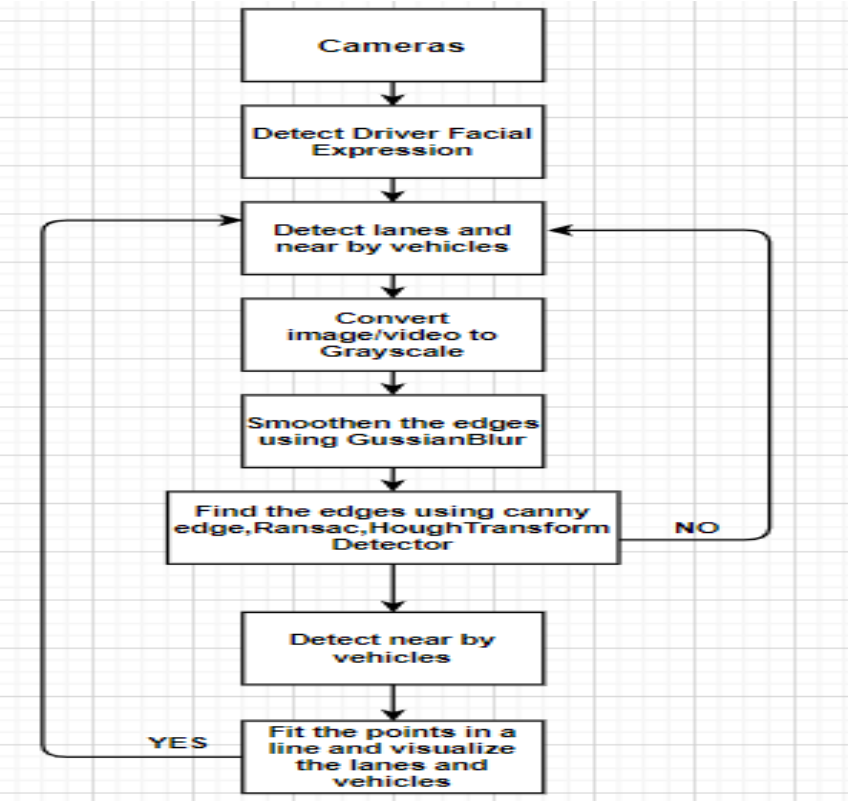

Fig 4.2: Data Flow Diagram

In the above data flow diagram a vision system is utilized including three cameras, two of them are under right and left wing mirrors, one is equipped on the windscreen of

the vehicle. The images from the cameras are used to detect the lanes, detect the vehicles and detect the driver facial expression.

4.2 Use case diagram

A **use case diagram** is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. Use case diagram shows a set of use cases and actors and their relationships. The main contents of a use case diagram are:

- Use Cases
- Actors
- Dependency, generalization, and association relationships

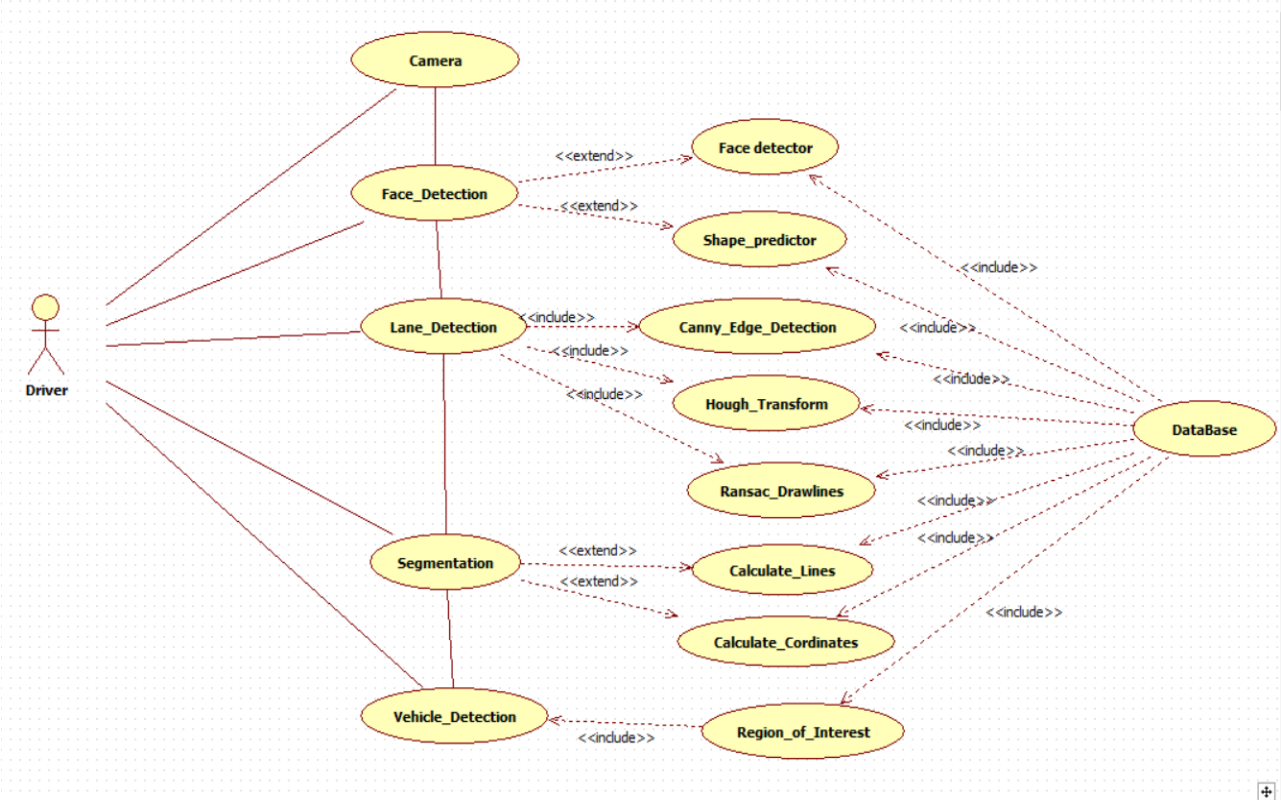

Fig 4.3: Use Case Diagram

In this proposed system use case shows the relationship between the user that is user interface and different use cases between the actions a vision system is utilized including three cameras, two of them are under right and left wing mirrors, one is equipped on the windscreen of the vehicle. The images from the cameras are used to detect the driver facial expressions using support vector machines, detect the lanes using canny edge detector and hough transform, detect the vehicles using haar like cascade classifier.

4.3 Class diagram

A **class diagram** is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modeling. . It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

A class diagram shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams involve global system description, such as the system architecture, and detail aspects such as the attributes and operations within a class as well. The most common contents of a class diagram are:

- Classes
- Interfaces
- Collaborations
- Dependency, generalization, and association relationships
- Notes and constraints

In this class diagram various classes specified as driver, vehicle, camera, database, road and attributes were specified. There are associations between the classes and the images from the cameras are used to detect the facial expression of the driver, detect the lanes and detect the vehicles and these data is stored in the database.

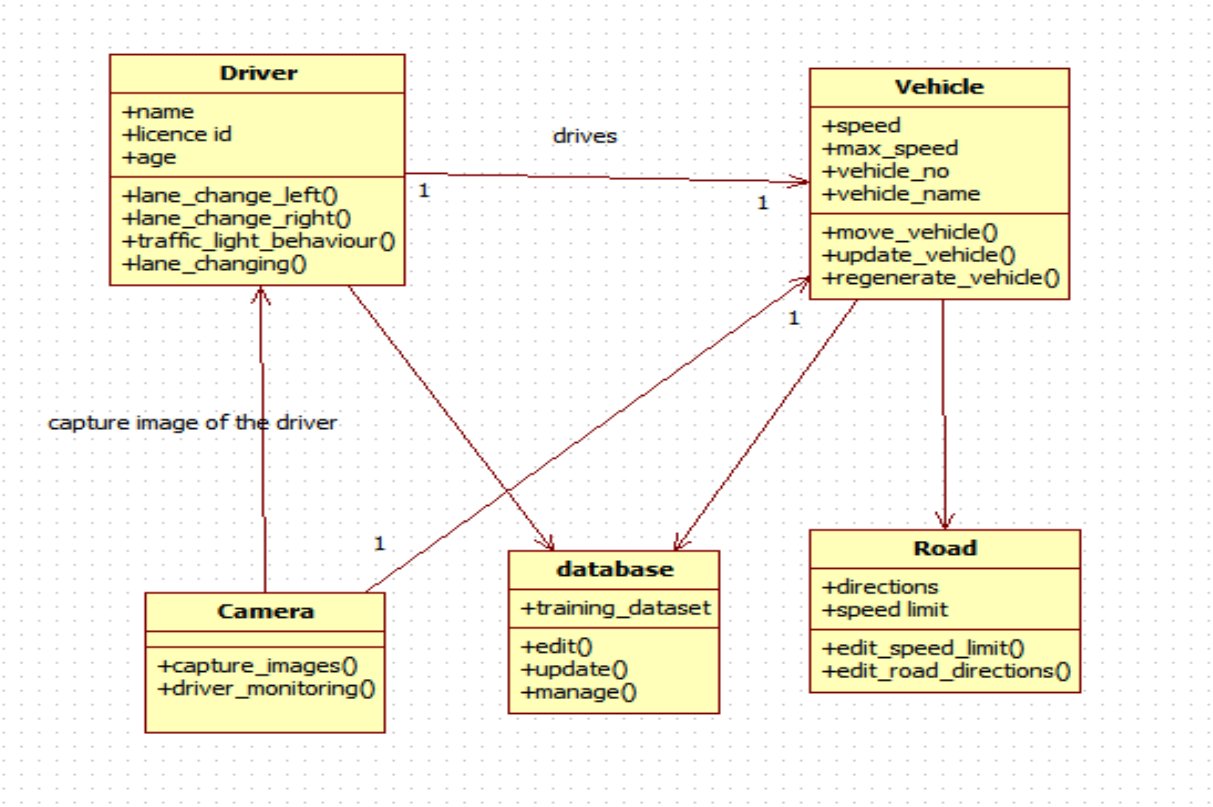

Fig 4.4: Class Diagram

4.4 Behavior diagrams

Behavioral diagrams depict the elements of a system that are independent on time and that convey the dynamic concepts of the system and how they relate to each other.

4.4.1 Sequence diagram

A **sequence diagram** shows object interactions arranged in time sequence. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

Sequence diagram is one kind of interaction diagrams, which shows an interaction among a set of objects and their relationships. The purpose of the Sequence diagram is to document the sequence of messages among objects in a time based view. The scope of a typical sequence diagram includes all the message interactions for a single use case. There may be multiple sequence diagrams per use case, one per use case scenario. The state diagrams commonly contain:

- Objects
- Links
- Messages ·
- Respond Time (especially useful in real-time systems)

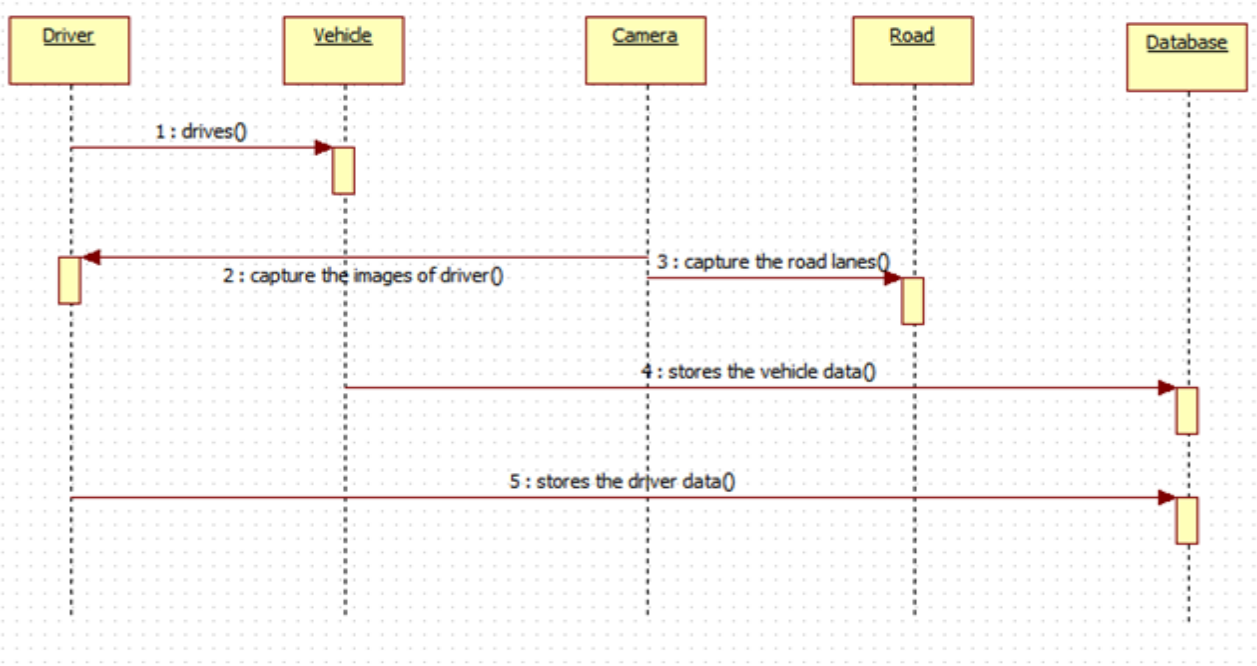

Fig 4.5: Sequence Diagram

In this proposed system driver, vehicle, camera, road and database are referred as objects and the images from the cameras are used to detect the facial expression of the driver, detect the lanes and detect the vehicles and these data is stored in the database.

4.4.2 Collaboration diagram

A collaboration diagram, also known as a communication diagram is an illustration of the relationships and interactions among software objects. A collaboration diagram is required to identify how various objects make up the entire system. Collaboration or communication diagrams are used to explore the architecture of objects inside the system. The message flow between the objects can be represented using a collaboration diagram.

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

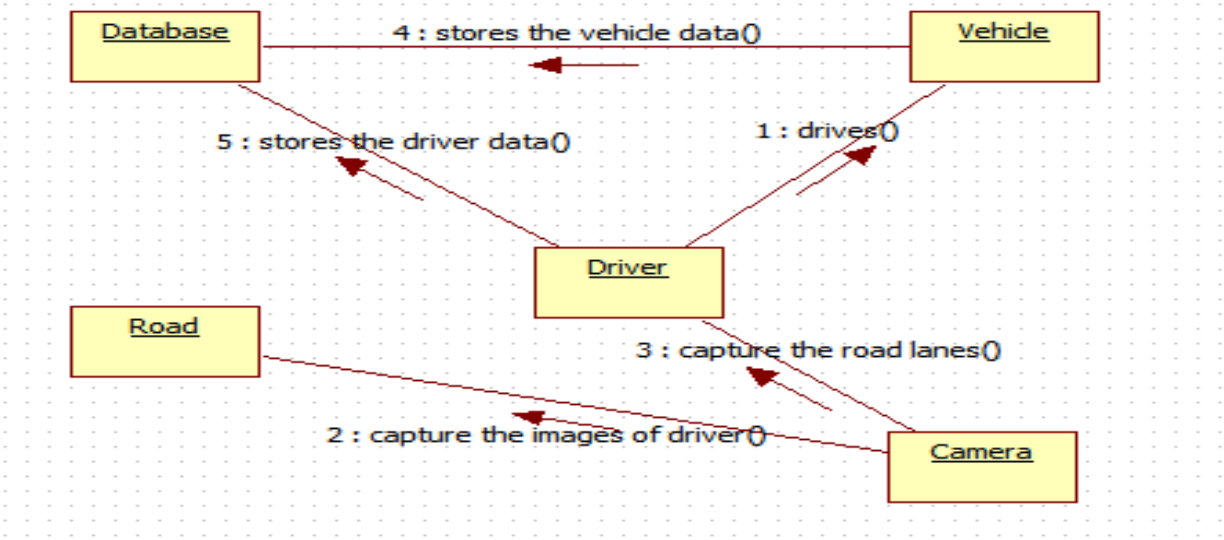

Fig 4.6: Collaboration Diagram

In this collaboration diagram various objects were specified as driver, vehicle, camera, road, database and the images from the cameras are used to detect the facial expression of the driver, detect the lanes and detect the vehicles and these data is stored in the database.

4.4.3 State chart diagram

A **state chart diagram** is used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. It describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

The most important purpose of state chart diagram is to model lifetime of an object from creation to termination. State chart diagrams are important for constructing executable systems through forward and reverse engineering.

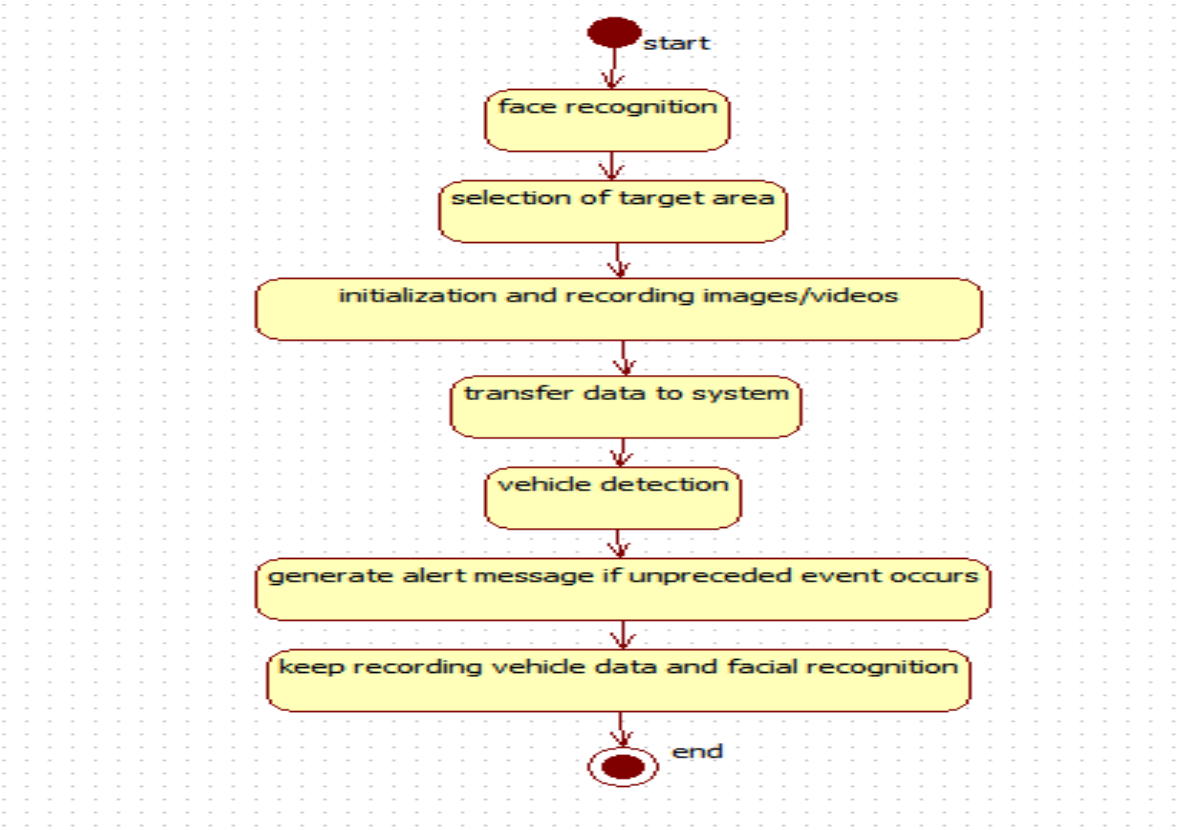

Fig 4.7: State Chart Diagram

In the above diagram a vision system is utilized including three cameras, two of them are under right and left wing mirrors, one is equipped on the windscreen of the vehicle. The images from the cameras are used to detect the driver facial expressions using support vector machines, detect the vehicles using haar like cascade classifier and detect the lanes using canny edge detector and Hough transform.

4.4.4 Activity diagram

Activity diagram is used to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Activity diagram is defined as a UML diagram that focuses on the execution and flow of the behavior of a system instead of implementation. It is also called object-oriented flowchart**.** An activity can be attached to any modeling element to model its behavior. Activity diagrams are used to model,

- Use cases
- Classes
- Interfaces
- Components
- Collaborations

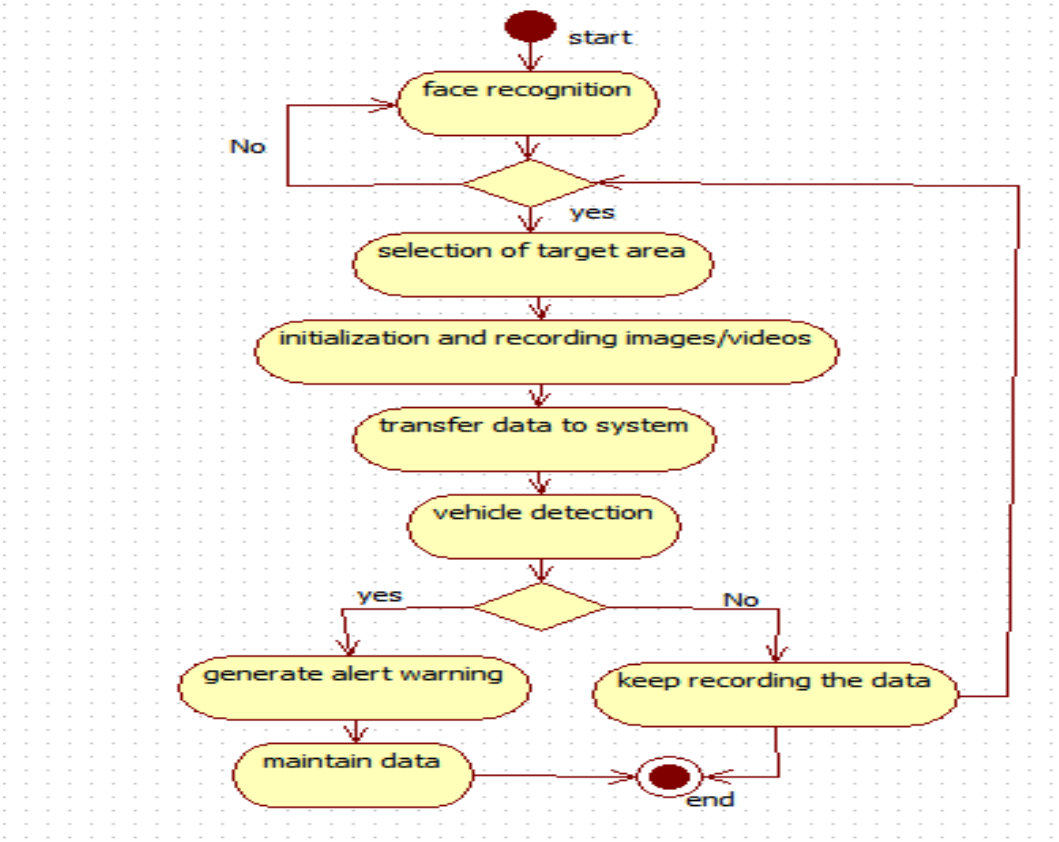

Fig 4.8: Activity Diagram

In the above diagram a vision system is utilized including three cameras, two of them are under right and left wing mirrors, one is equipped on the windscreen of the vehicle. The images from the cameras are used to detect the driver facial expressions using support vector machines, detect the vehicles using haar like cascade classifier and detect

the lanes using canny edge detector and Hough transform and these data is stored in the database.

4.5 Implementation diagram

Implementation diagram specifies the physical components used and also specifies the physical configuration of run time elements.

4.5.1 Component diagram

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Physical aspects are the elements such as executable, libraries, files, documents, etc. which reside in a node. The purpose of the component diagram can be summarized as visualize the components of a system, construct executable by using forward and reverse engineering, describe the organization and relationships of the components.

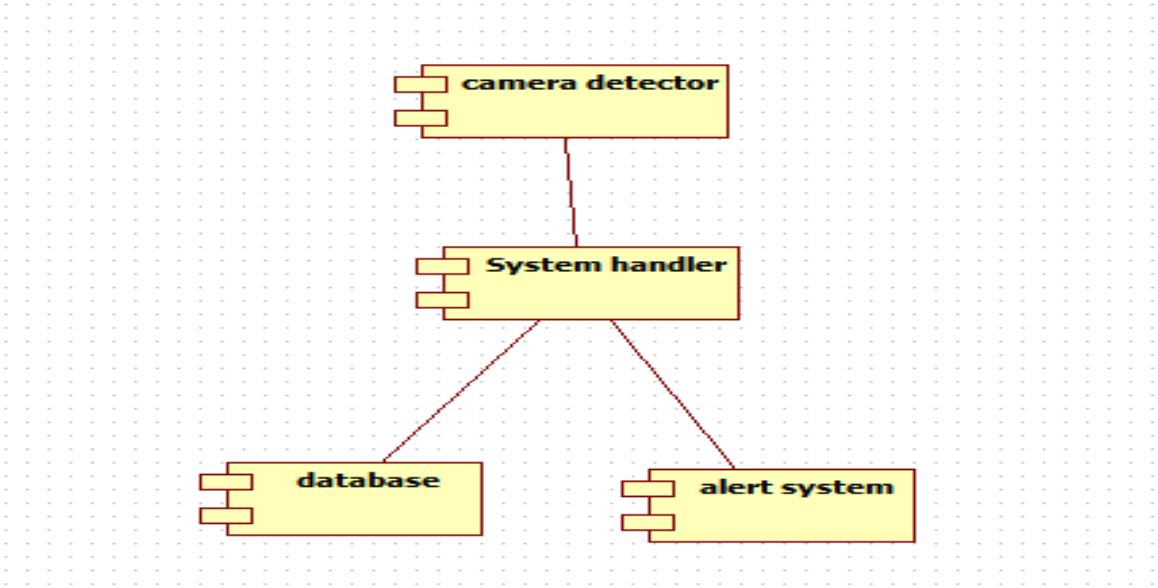

Fig 4.9: Component Diagram

In this component diagram various components were specified as camera detector, system handler, alert system and database. Camera is used to capture the images of driver, road and vehicles. The images from the cameras are used to detect the lanes, detect the vehicles and driver facial expressions because if the driver may be feel sleepy at that time our facial expression will detect and alert the driver with voice assistant otherwise there is

a chance to occur accidents, and these always keep analyze the facial expressions of the driver until he reaches the destination.

4.5.2 Deployment diagram

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships. Deployment diagrams are used for describing the hardware components, where software components are deployed.

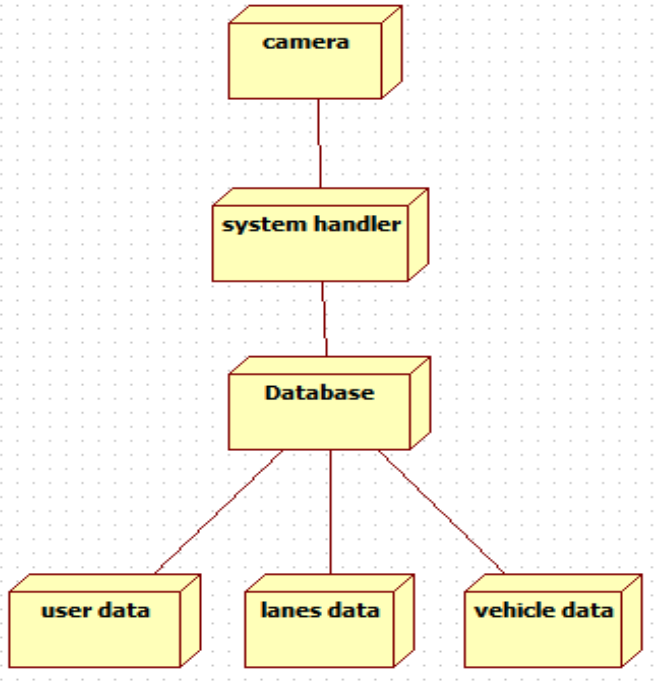

Fig 4.10: Deployment Diagram

In this deployment diagram a vision system is utilized including three cameras, two of them are under right and left wing mirrors, one is equipped on the windscreen of the vehicle. The images from the cameras are used to detect the lanes, detect the vehicles and detect the driver facial expressions and these data is stored in the database.

4.6 Conclusion

In this report, a detailed UML diagram representation for a lane detection, tracking and recognition system for smart vehicles (LTRSV) is given. The UML diagrams used in this article is Architecture Diagram, Data flow Diagram, Use Case Diagram, Class Diagram, Sequence Diagram, Collaboration Diagram, State Chart Diagram, Activity Diagram, Component Diagram and Deployment Diagram.

Chapter 5

IMPLEMENTATION

5.1 Proposed methodology

The proposed methodology is to detect road lanes to reduce road accidents and increase safety through implementing the modules Drowsiness detection, lane detection and vehicle detection. It detects drowsiness of the driver, detect road lanes and vehicles until the vehicle reaches the destination.

5.2 Proposed methodology module

1. Drowsiness detection

2. Lane detection

3. Vehicle detection

5.3 Drowsiness Detection

A computer vision system that can automatically detect driver drowsiness in a real-time video stream and then play an alert if the driver appears to be drowsy, here to recognize the drowsiness it uses the Support Vector Machine algorithm.

The entire drowsiness detection solution is divided into following major modules:

1. Eye shape detector
2. Eye shape predictor

Eye shape detector and Eye shape predictor works on the basis of trained data set given.

A trained data set and the range value is given as an input and the position of the eye is calculated. If it crosses the given range then a voice alert is given to the driver as either wake up or park the car.

Techniques/Algorithms:

The algorithm used in facial recognition is **Support Vector Machine** algorithm**.**

Support Vector Machine

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for either classification or regression analysis. More formally, a support-vector machine constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection.

Formula:

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Each eye is represented by 6 (x, y)-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the eye.

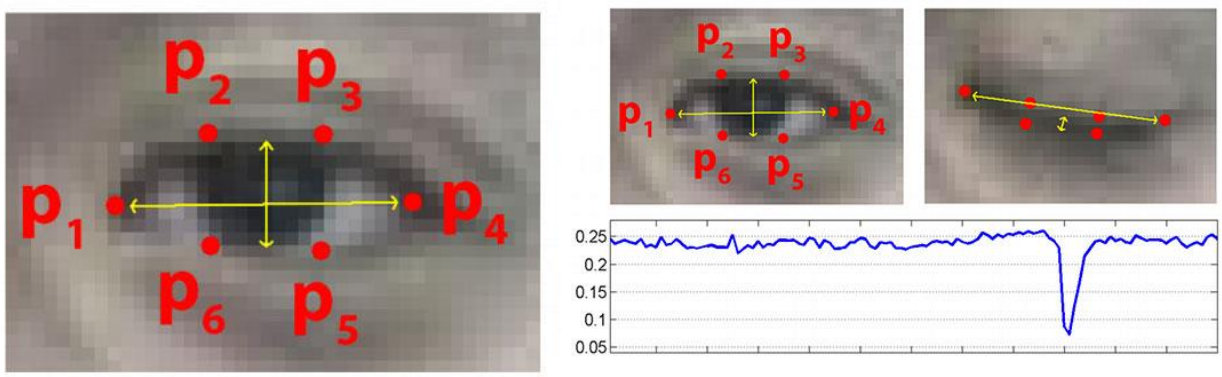

Fig 5.3.1: Drowsiness detection

5.4 Lane detection

Lane detection is one important process in the vision-based vehicle assist system. The results of lane edge detection play an important role in feature-based lane detection. The complicated conditions of road make the correct edge detection of lane markings become very challenging. In order to get an ideal edge of lane markings in road image, a method of lane edge detection based on canny algorithm is proposed. Firstly according to the importance in lane markings recognition, the road image is divided into three regions. Only the regions with useful information are processed. Then by the features of gray distribution and lane markings width, some noises are removed from the image. Using the shape features of lane markings, the lane edges are detected based on canny algorithm. Finally by use of the Hough transform theory, lane detection is achieved

Techniques/Algorithms:

- **Canny Edge Detection Algorithm:** It is used to find the edges of the image and is used for noise reduction and smoothens the image

Formula:

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

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

Where G_x-Derivative in horizontal direction
G_y - Derivative in vertical direction

Algorithm:

```
do Canny(Input_frame);
ReduceNoise = Noise Reduction(Input_frame);
gray = Filter color(ReduceNoise);
blur = GuassianBlur(gray(Arg1,Arg2,Arg3));
Canny=Canny (Blur, Arg1, Arg2);
return Canny;
```

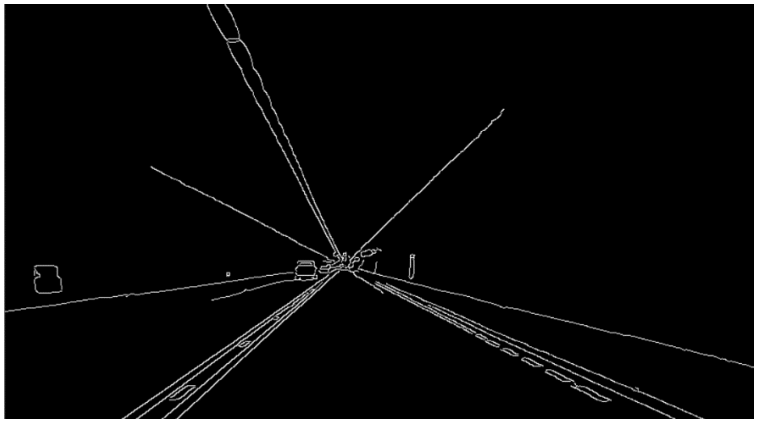

Fig 5.4.1: Canny edge detection

- **Hough Transform:**

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. The classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc.A generalized Hough transform can be employed in applications where a simple analytic description of a feature(s) is not possible. Due to the computational complexity of the generalized Hough algorithm, we restrict the main focus of this discussion to the classical Hough transform. Despite its domain restrictions, the classical Hough transform (hereafter referred to without the *classical* prefix) retains many applications, as most manufactured parts (and many anatomical parts investigated in medical imagery) contain feature boundaries which can be described by regular curves. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise. The lines in the Hough space can be represented in the form of Y=a. x + b, where a & b are numbers.

The Hough technique is particularly useful for computing a global description of a feature(s), given local measurements. The motivating idea behind the Hough technique for line detection is that each input measurement (*e.g.* coordinate point) indicates its contribution to a globally consistent solution (*e.g.* the physical line which gave rise to that image point).

As a simple example, consider the common problem of fitting a set of line segments to a set of discrete image points

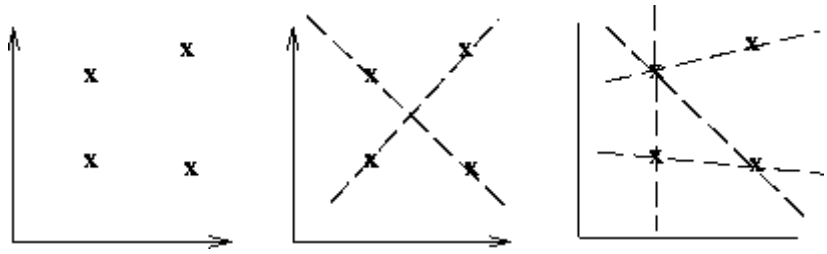

Fig 5.4.2: Hough transform line segments

Algorithm:

```
do segment(Input_frame)
perform bitwise operations and add mask;
return segment;
Calculate_lines(Input_frame.lines)
Left = [];
```

```
Right = [];
for line in lines:
        parameters(Input_frame);
        slope=parameters[0];
        if(slope<0):
                left_lane;
        else
                right_lane;
        return(left_lane,right_lane);
Calculate_coordinates(frame,parameters):
        Find y1, y2, x1, x2 coordinates;
Return coordinates;
def_hough(segment, lines, coordinates and default arguments ):
visualize(hough);
```

Fig 5.4.3: Hough transform

5.5 Vehicle detection:

Vehicle detection is a technology which its aim is to locate and show the vehicle size in digital images. In this technology, vehicles are detected in presence of other things like trees and buildings. Object detection based on digital image processing on vehicles is very important for establishing monitoring system or as alternative method to collect statistic data to make efficient traffic engineering decision. A vehicle counter program based on traffic video feed for specific type of vehicle using Haar Cascade Classifier was made as the output of this research. Firstly, Haar-like feature was used to present visual shape of vehicle, and **AdaBoost** machine learning algorithm was also employed to make a strong classifier by combining specific classifier into a cascade filter to quickly remove background regions of an image. At the testing section, the output was tested over 8 realistic video data and achieved high accuracy. The result was set 1 as the biggest value

for recall and precision, 0.986 as the average value for recall and 0.978 as the average value for precision. It has an important role in many computer vision applications such as vehicle tracking, analyzing the traffic scene and efficient traffic management.

Techniques/Algorithms:

The technique used in vehicle detection is Haar like cascade classifier

- **Haar like cascade classifier**

It is a machine learning object detection algorithm used to identify objects in a video. Firstly Haar-like feature was used to present visual shape of vehicle, and AdaBoost machine learning algorithm was also employed to make a strong classifier by combining specific classifier into a cascade filter to quickly remove background regions of an image.

Formula:

To compute the distance between the test vehicle and the detected vehicle.

[Dis = F*H/Y]

Where F=Focal length

H=Camera height

Y=Image co-ordinate

Fig 5.5.1: Vehicle detection

5.6 Conclusion

Finally this proposed methodology implement the road lane detection, face recognition, for smart vehicles by implementing the above modules and by their respective techniques in order to reduce road accidents and increase safety.