CHAPTER-1

INTRODUCTION

1.1 Introduction:

Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time. By monitoring the eyes, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident. Detection of fatigue involves the observation of eye movements and blink patterns in a sequence of images of a face.

Initially, we decided to go about detecting eye blink patterns using python. The procedure used was the geometric manipulation of intensity levels. The algorithm used was as follows. First we input the facial image using a webcam. Preprocessing was first performed by binarizing the image. The top and sides of the face were detected to narrow down the area where the eyes exist. Using the sides of the face, the center of the face was found which will be used as a reference when computing the left and right eyes. Moving down from the top of the face, horizontal averages of the face area were calculated. Large changes in the averages were used to define the eye area. There was little change in the horizontal average when the eyes were closed which was used to detect a blink. However python had some serious limitations. The processing capacities required by python were very high. Also there were some problems with speed in real time processing. python was capable of processing only 4-5 frames per second. On a system with a low RAM this was even 9 lower. As we all know an eye blink is a matter of milliseconds. Also a drivers head movements can be pretty fast. Though the MATLAB program designed by us detected an eye blink, the performance was found severely wanting. This is where OpenCV came in. OpenCV is an open source computer vision library. It is designed for computational efficiency and with a strong focus on real time applications. It helps to build sophisticated vision applications quickly and easily. OpenCV satisfied the low processing power and high speed requirements of our application. We have used the Haartraining applications in OpenCV to detect the face and eyes. This creates a classifier given a set of positive and negative samples. The steps were as follows:-Gather a data set of face and eye. These should be stored in one or more directories indexed by a text file. A lot of high quality data is required for the classifier to work well. The utility application createsamples() is used to build a vector output file. Using this file we can repeat the training procedure. It extracts the positive samples from images before normalizing and resizing to specified width and height. The Viola Jones cascade decides whether or not the object in an image is similar to the training set. Any image that doesn't contain the object of interest can be turned into negative sample. So in order to learn any object it is required to take a sample of negative background image. All these negative images are put in one file and then it's indexed. Training of the image is done using boosting. In training we learn the group of classifiers one at a time. Each classifier in the group is a weak classifier. These weak classifiers are typically composed of a single variable decision tree called stumps. In training the decision stump learns its classification decisions from its data and also learns a weight for its vote from its accuracy on the data. Between training each classifier one by one, the data points are reweighted so that more attention is paid to the data points where errors were made. This process continues until the total error over the dataset arising from the combined weighted vote of the decision trees falls below a certain threshold.

This algorithm is effective when a large number of training data are available. For our project face and eye classifiers are required. So we used the learning objects method to create our own haarclassifier .xml files. Around 2000 positive and 3000 negative samples are taken. Training them is a time intensive process. Finally face.xml and haarcascade-eye.xml files are created. These xml files are directly used for object detection. It detects a sequence of objects (in our case face and eyes). Haarcascade-eye.xml is designed only for open eyes. So when eyes are closed the system doesn't detect anything. This is a blink. When a blink lasts for more than 5 frames, the driver is judged to be drowsy and an alarm is sounded.

1.2 Implementation

Modules:

1. Driver: Driver is the module to whom security system is developed driver need to run application which will on camera and detect user face and then send image samples for processing.
2. Processing: In processing stage image frame is detected then eye is detected and check standby status and detect how many eyes are there in the frame and then detect driver fatigue.
3. Wakeup application: After processing data is sent to output with status of eye then alarm is on if eyes are closed.

CHAPTER-2

SYSTEM FEASIBILITY

2.1 Introduction:

Why OpenCV?

2.1.1 Eye blink detection using python: -

All the programming languages we had obtained the most proficiency in python. Hence it was no surprise that we initially decided to do the project using python. The procedure used was the geometric manipulation of intensity levels. The algorithm used was as follows.

First, we input the facial image using a webcam. Preprocessing was first performed by binarizing the image. The top and sides of the face were detected to narrow down the area where the eyes exist. Using the sides of the face, the center of the face was found which will be used as a reference when computing the left and right eyes. Moving down from the top of the face, horizontal averages of the face area were calculated. Large changes in the averages were used to define the eye area. There was little change in the horizontal average when the eyes were closed which was used to detect a blink. However MATLAB had some serious limitations. The processing capacities required by MATLAB were very high. Also, there were some problems with speed in real time processing. MATLAB was capable of processing only 4-5 frames per second. On a system with a low RAM this was even lower. As we all know an eye blink is a matter of milliseconds. Also, a drivers head movements can be pretty fast. Though the MATLAB program designed by us detected an eye blink, the performance was found severely wanting.

2.1.2 What Is OpenCV?

OpenCV [OpenCV] is an open source (see http://opensource.org) computer vision library available from http://SourceForge.net/projects/opencvlibrary.

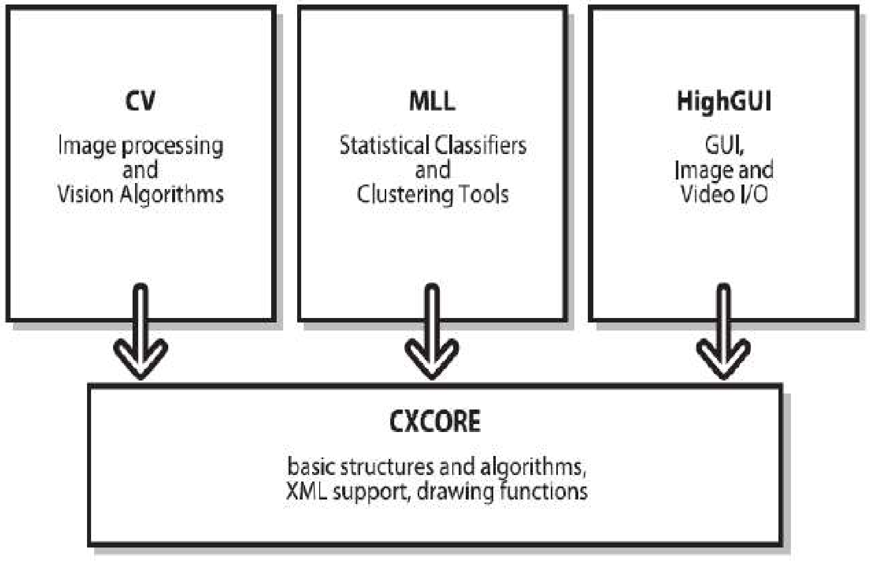
OpenCV was designed for computational efficiency and having a high focus on real-time image detection. OpenCV is coded with optimized C and can take work with multicore processors. If we desire more automatic optimization using Intel architectures [Intel], you can buy Intel's Integrated Performance Primitives (IPP) libraries [IPP]. These consist of low-level routines in various algorithmic areas which are optimized. OpenCV automatically uses the IPP library, at runtime if that library is installed. One of OpenCV goals is to provide a simple-to-use computer vision infrastructure which helps people to build highly sophisticated vision applications fast. The OpenCV library, containing over 500 functions, spans many areas in vision. Because computer vision and machine learning often goes hand-in-hand, OpenCV also has a complete, general-purpose, Machine Learning Library (MLL).

2.1.3 What Is Computer Vision?

Computer vision is the transforming of data from a still, or video camera into either a representation or a new decision. All such transformations are performed to achieve a particular goal. A computer obtains a grid of numbers from a camera or from the disk, and that's that. Usually, there is no built in pattern recognition or automatic control of focus and aperture, no cross-associations with years of experience.

2.1.4 The Origin of OpenCV

OpenCV came out of an Intel Research initiative meant to advance CPU-intensive applications. Toward this end, Intel launched various projects that included real-time ray tracing and also 3D display walls. One of the programmers working for Intel at the time was visiting universities. He noticed that a few top university groups, like the MIT Media Lab, used to have well-developed as well as internally open computer vision infrastructures—code that was passed from one student to another and which gave each subsequent student a valuable foundation while developing his own vision application. Instead of having to reinvent the basic functions from beginning.


CV
Image processing and Vision Algorithms
MLL
Statistical Classifiers and Clustering Tools
HighGUI
GUI, Image and Video I/O
CXCORE
basic structures and algorithms, XML support, drawing functions

2.1.5 OpenCV Structure and Content

OpenCV can be broadly structured into five primary components, four of which are shown in the figure. The CV component contains mainly the basic image processing and higher-level computer vision algorithms; MLL the machine learning library includes many statistical classifiers as well as clustering tools. High GUI component contains I/O routines with functions for storing, loading video & images, while CX Core contains all the basic data structures and content.

2.1.6 Why OpenCV?

Specific OpenCV was designed for image processing. Every function and data structure has been designed with an Image Processing application in mind. Meanwhile, MATLAB, is quite generic. You can get almost everything in the world by means of toolboxes. It may be financial toolboxes or specialized DNA toolboxes. 15 Speedy MATLAB is just way too slow. MATLAB itself was built upon Java. Also Java was built upon C. So when we run a MATLAB program, our computer gets busy trying to interpret and compile all that complicated MATLAB code. Then it is turned into Java, and finally executes the code. If we use C/C++, we don't waste all that time. We directly provide machine language code to the computer, and it gets executed. So ultimately, we get more image processing, and not more interpreting. After doing some real time image processing with both MATLAB and OpenCV, we usually got very low speeds, a maximum of about 4-5 frames being processed per second with MATLAB. With OpenCV however, we get actual real time processing at around 30 frames being processed per second. Sure, we pay the price for speed – a more cryptic language to deal with, but it's definitely worth it. We can do a lot more, like perform some really complex mathematics on images using C and still get away with good enough speeds for your application. Efficient MATLAB uses just way too much system resources. With OpenCV, we can get away with as little as 10mb RAM for a real-time application. Although with today's computers, the RAM factor isn't a big thing to be worried about. However, our drowsiness detection system is to be used inside a car in a way that is non-intrusive and small; so, a low processing requirement is vital.

Thus, we can see how OpenCV is a better choice than MATLAB for a real-time drowsiness detection system.

2.2 Existing System:

There are various methods like detecting objects which are near to vehicle and front and rear cameras for detecting vehicles approaching near to vehicle and air bag system which can save lives after accident is accorded.

2.3 Disadvantage Of Existing System:

Most of the existing systems use external factors and inform user about problem and save user after accident is accord but from research most of the accidents are due to faults in user like drowsiness, sleeping while driving.

2.4 Proposed system:

To deal with this problem and provide an effective system a drowsiness detection system can be developed which can be placed inside any vehicle which will take live video of driver as input and compare with training data and if driver is showing any symptoms of drowsiness system will automatically detect and raise alarm which will alert driver and other passengers.

2.5 Advantages of proposed system:

This method will detect problem before any problem accord and inform driver and other passengers by raising alarm.

In this OpenCV based machine learning techniques are used for automatic detection of drowsiness.

CHAPTER-3

SYSTEM ANALYSIS

3.1 The study of the system

- To conduct studies and analyses of an operational and technological nature, and
- To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

3.1.1 What Is Machine Learning

The goal of machine learning is to turn data into information. After having learned from a gathering of data, we want a machine that is able to answer any question about the data:

What are the other data that are similar to given data?

Is there a face in the image?

What kind of ad will influence the user?

There is usually a cost component, so the question may become:

Of the many products that we can make the money from, which one will most likely be bought by the user if an ad is shown for it?

Machine learning converts data into information by detecting rules or patterns from that data.

3.1.2 Training and Test Set

Machine learning works on data like temperature values or stock prices or color intensities, and in our case face and eye detection. The data is usually preprocessed into features. We might take a database containing 1,000 face images then perform an edge detector on all the faces, and then obtain features such as edge direction, edge intensity, also offset from the face center for every face. We may obtain up to 500 such values for every face or a feature vector of 500 entries. We may then use machine learning to construct some sort of model from the obtained data. If we want to see how the faces fall into various groups (narrow, wide, etc.), after that a clustering algorithm can be the preferred choice. In case we want to learn how to guess the age of a woman from the pattern of the edges that are detected on her face, then we can use a classifier algorithm. To reach our goals, machine learning algorithms can analyze our obtained features and hence adjust the weights, the thresholds, and all the other parameters for maximizing performance set by those goals. This method of parameter adjustment for meeting a goal is what is called learning. It is very important to understand how efficiently machine learning methods can work. This might be a delicate task. Usually, we break up the input data set into a very large training set (i.e. 900 faces, in our project) and a relatively small test set (i.e. remainder 100 faces). Then we can run our classifier on the training set in order to learn for a age prediction model, data feature 18 vectors given. Once done, we can then test our age prediction classifier obtained on the remainder of the images left in the set. The test set is not applied for training; also we don't allow the classifier to see the age labels of the test set. We then run the classifier on all of the 100 faces present in the test set and then record how well the ages predicted by the feature vector match the real ages. When the classifier performs poorly, we may try to add new features to the data or consider other types of classifiers. There are many types of classifiers and various algorithms used for training them. When the classifier performs well, we have a possibly lucrative model that can be used on data in the actual world. This system may be used to set the performance of a video game according to age. After the classifier has been setup, it sees faces that it could not see before and it makes decisions with regards to what it had learned during training. Finally, when we develop a classification system, we usually use a validation data set. At times, testing the entire system at the finish is a big step to take. We usually want to change parameters in the way before preparing our classifier for the final testing. We may do this by splitting the initial 1,000-face data set into 3 parts: i.e. a training set consisting of 800 faces and also a validation set with 100 faces, as well as a test set containing 100 faces. While we're running across the training data, we can perform pretests on validation data in order to see our progress. If we are happy with our output for the validation set, we can run the classifier up on the test set for the final result.

3.1.3 OpenCV ML Algorithms

The machine learning algorithms that are included in OpenCV are given as follows. All the algorithms are present in the ML library apart from Mahala Nobis and K-means, which are present in CVCORE, and the algorithm of face detection, which is present in CV. Mahala Nobis: It is a measure of distance that is responsible for the stretchiness of the data. We can divide out the covariance of the given data to find this out. In case of the covariance being the identity matrix (i.e. identical variance), this measure will be identical to the Euclidean distance. K-means: It is an unsupervised clustering algorithm which signifies a distribution of data w.r.t. K centers, K being chosen by the coder. The difference between K-means and expectation maximization is that in K-means the centers aren't Gaussian. Also the clusters formed look somewhat like soap bubbles, as centers compete to occupy the closest data points. All these cluster areas are usually used as a form of sparse histogram bin for representing the data. Normal/Naïve Bayes classifier: It is a generative classifier where features are often assumed to be of Gaussian distribution and also statistically independent from one another. This assumption is usually false. That's why it's usually known as a ―naïve Bayes‖ classifier. That said, this method usually works surprisingly well. Decision trees: It is a discriminative classifier. The tree simply finds a single data feature and determines a threshold value of the current node which best divides the data into different classes.

The data is broken into parts and the procedure is recursively repeated through the left as well as the right branches of the decision tree. Even if it is not the top performer, it's usually the first thing we try as it is fast and has a very high functionality.

Boosting: It is a discriminative group of classifiers. In boosting, the final classification decision is made by taking into account the combined weighted classification decisions of the group of classifiers. We learn in training the group of classifiers one after the other. Each classifier present in the group is called a weak classifier. These weak classifiers are usually composed of single-variable decision trees known as ―stumps‖. Learning its classification decisions from the given data and also learning a weight for its vote based on its accuracy on the data are things the decision tree learns during training. While each classifier is trained one after the other, the data points are re-weighted to make more attention be paid to the data points in which errors were made. This continues until the net error over the entire data set, obtained from the combined weighted vote of all the decision trees present, falls below a certain threshold. This algorithm is usually effective when a very large quantity of training data is available. Random trees: It is a discriminative forest of a lot of decision trees, each of which is built down to a maximal splitting depth. At the time of learning, every node of every tree is allowed a choice 20 of splitting variables, but only from a randomly generated subset of all the data features. This ensures that all the trees become statistically independent and a decision maker. In the run mode, all the trees get an unweighted vote. Random trees are usually quite effective. They can also perform regression by taking the average of the output numbers from every tree.

Face detector (Haar classifier): It is an object detection application. It is based on a smart use of boosting. A trained frontal face detector is available with the OpenCV distribution. This works remarkably well. We can train the algorithm for other objects by using the software provided. This works wonderfully for rigid objects with characteristic views. Expectation maximization (EM): It is used for clustering. It is a generative unsupervised algorithm It fits N multidimensional Gaussians to the data, N being chosen by the user. It can act as an efficient way for representing a more complex distribution using only a few parameters (i.e. means and variances). Usually used in segmentation, it can be compared with K-means. K-nearest neighbors: It is one of the simplest discriminative classifier. The training data is simply stored using labels. Then, a test data point is classified in accordance to the majority vote of the K nearest data points. K-nearest neighbours is probably the simplest algorithm we can use. It is usually effective but it can be slow. It also requires a lot of memory. Neural networks or Multilayer perceptron (MLP): It is a discriminative algorithm which almost always contains hidden units in between the output and the input nodes for better representation of the input signal. It is slow to train, however it is quite fast to run. It remains the best performer for applications like letter recognition. Support vector machine (SVM): It is a discriminative classifier that is also capable of doing regression. Here, a distance function in between two data points is defined in a higher-dimensional space. (Projecting data onto higher dimensions helps in making the data more likely for linear separation.) Support vector machine (SVM) learns separating hyperplanes which maximally separate all the classes in the higher dimension. This tends to be the best when there 21 is limited data. However, when large data sets are available, boosting or random trees are preferred.

3.1.4 Using Machine Learning in Vision

Usually, most algorithms take a data vector having many features as input. Here, the number of features may number in the thousands. If our task is recognizing a certain kind of object—take for example, a person's face. The first problem that we encounter is obtaining and labeling the training data which falls into positive (i.e. there is a face in the window) and negative (i.e. no face) cases. We soon realize that faces can appear at various scales: i.e. their image might consist of only a few pixels, or we might be looking at an ear which is filling the whole screen. Worse still, faces are usually occluded. We have to define what we actually mean when we say that a face is in the window.

After having labeled the data that was obtained from various sources, we should decide which features we need to extract from these objects. Also, we must know what objects we are after. If the faces always appear upright, there is no reason for using rotation-invariant features and also no reason for trying to rotate the objects before processing. In general, we must try to find features that express a little invariance in the objects. These can be scale-tolerant histograms of gradients or colors or even the popular SIFT features. When we have requisite background window information, we can first remove it in order to help other objects stand out. Then we perform our image processing. This may consist of normalizing the image and then computing the various features. The resulting data vectors are all given the label that is associated with the object, action, or window. Once the data is obtained and converted into feature vectors, we break up the data into training sets, validation sets and test sets. It is advisable to do our learning, validation, and testing using a cross-validation framework. Here, the data is split into K subsets and we run various training (maybe validation) as well as test sessions. Each session consists of various sets of data that take on the roles of training (validation) and test. The test results obtained from these separate 22 sessions are used for averaging to get the final performance result. A more accurate picture of how the classifier performs when deployed in operation can be given by cross-validation. Now that our data is ready, we must choose a classifier. Usually the choice of the classifier is determined by computational, data, and memory requirements. For certain applications, like online user preference modeling, we need to train the classifier quickly. In such a case, nearest neighbors, normal Bayes, or decision trees should be a good choice. When memory is the primary consideration, decision trees or neural networks are used for their space efficiency. When we have time to train our classifier but it needs to run quickly, neural networks can be a good choice, as with normal Bayes classifiers & support vector machines. When we have time to train but require high accuracy, then boosting and random trees are good choices. When we just want an easy and understandable check weather our features are chosen well or not, then decision trees or nearest neighbors should be used. For a good out of the box classification performance, boosting or random trees are tried.

3.1.5 Variable Importance

This is the importance of a particular variable in a dataset. One of the uses of variable importance is for reducing the number of features the classifier needs to consider. After starting with a number of features, we train our classifier to find the importance of each feature in relation to all the other features. We then get rid of unimportant features. Eliminating unimportant features helps in improving speed performance (since it can eliminate the processing taken for computing those features) and also makes training and testing faster. When we don't have sufficient data, which is regularly the case, then eliminating unimportant variables helps in increasing classification accuracy, which in turn yields faster processing and better results.

Breiman's algorithm for variable importance is as follows.

1. A classifier is trained on the training set.
2. A validation or test set is used for determining the accuracy of the classifier. 23
3. For each data point and a chosen feature, a new value for that feature is randomly chosen from among the values that the feature has in the remainder of the data set (known as ―sampling with replacement‖). This helps in ensuring that the distribution of that feature remains the same as in the original data set, however, the actual structure or meaning of that feature is removed (as its value is chosen at random from the remainder of the data).
4. The classifier is trained on the altered set of the training data and then the accuracy of classification measured on the changed test or validation data set. When randomizing of a feature hurt accuracy a lot, then it is implied that the feature is vital. When randomizing of a feature does not hurt accuracy much, then the feature is of little importance and is a suitable candidate for removal.
5. The original test or validation data set is restored and the next feature is tried until we are finished. The result obtained orders each feature by its importance. This procedure used above is built into random trees and decision trees and boosting. Thus, we can use these algorithms to decide which variables we will finally use as features; then we can use the optimized feature vectors to train the classifier.

3.1.6 Tools of Machine Learning

There are a few basic tools which are used in machine learning to know the results. In supervised learning, one of the fundamental problems is knowing just how well the algorithm has performed: i.e. How accurate is it at fitting or classifying the data? For real problems, we must take into account noise, fluctuations and errors in sampling, and so on. In other words, our test or validation data set may not reflect accurately the actual distribution of data. To come closer to predicting the actual performance of the classifier, we use the technique of cross-validation and/or the closely related technique of bootstrapping. In its most rudimentary form, cross-validation involves the division of data into K different subsets. We train on K – 1 of these subsets and then test on the final subset of data (i.e. the ―validation set‖) that we didn't train. We do this K times, in which each of the K subsets gets its turn at becoming the validation set. Then we average the results.

Bootstrapping is somewhat similar to cross-validation, but here the validation set is randomly selected from the training data. The selected points for the round are then used only for testing, not training. Then the process is started again from scratch. We do this N times. Each time we randomly select a new set of validation data and then average the results in the end. This means some and/or many of the data points are reused in various validation sets, hence the results are usually superior compared to cross-validation. There are two other very useful ways of assessing, tuning and characterizing classifiers. One is to plot the receiver operating characteristic (ROC) and the other is filling in a confusion matrix; see the figure. The ROC curve is used to measure the response of the performance parameter of the classifier through the full range of settings of that parameter. Here, the parameter is a threshold. Just to make this more relevant to our project, suppose we are want to recognize blue eyes in an image and we are using a threshold on the color blue as our detector. Setting the blue threshold extremely high would result in the classifier failing to recognize any blue eyes, thereby yielding a false positive rate of 0 but also having a true positive rate also at 0 (i.e. lower left part of the curve in the figure). On the other side, if the blue threshold is set to 0 then all signals count as a recognition. This means that all true positives (the blue eyes) are recognized including all the false positives (brown and green eyes); thus we end up having a false positive rate of 100% (i.e. up for every right part of the curve in the figure). The best possible ROC curve would be the one that follows the y-axis up to 100% and then cuts horizontally over to the up for every right corner. Failing that, the closer that the curve comes to the up for every left corner, the better. We can compute the fraction of area under the ROC curve versus the total area of the ROC plot. It can be a statistic of merit: The closer that the ratio is to 1, the better the classifier. The given figure also shows a confusion matrix. It is just a chart of true and false positives along with true and false negatives. It is a quick way to evaluate the performance of a classifier. In an ideal case, we would see 100% along the NW-SE diagonal and 0% elsewhere. If we have a classifier that is capable of learning more than one class, then the confusion matrix is generalized for many classes and we just have to keep track of the class to which each labeled data point was assigned.

Learning Objects A trained classifier cascade stored in an XMLfile can be used by the cvLoad() function to load it and then cvHaarDetectObjects() to find objects similar to the ones it was trained on to train our own classifiers to detect other objects such as faces and eyes. We do this with the OpenCV haartraining application, which creates a classifier from a training set of positive and negative samples.

3.1.7 Dataset gathering: -

Collect a data set consisting of examples of the object you want to learn These may be stored in one or more directories indexed by a text file in the following format:

<path>/img_1 count_1 x11 y11 w11 h11 x12 y12 . . .

<path>/img_2 count_2 x21 y21 w21 h21 x22 y22 . . .

Each of these lines contains the path (if any) and file name of the image containing the object(s). This is followed by the count of how many objects are in that image and then a list of rectangles containing the objects. The format of the rectangles is the x- and y-coordinates of the upper left corner followed by the width and height in pixels. For optimum performance of the classifier you have to gather a lot of positive samples with very less unnecessary variance in data. The output of this process finally creates a file in the format.

3.2 Script

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

3.3 INTRODUCTION TO PYTHON

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you're not interested in the how's and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

- Open source general-purpose language.
- Object Oriented, Procedural, Functional
- Easy to interface with C/ObjC/Java/Fortran
- Easy-ish to interface with C++ (via SWIG)
- Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

Easy-to-read − Python code is more clearly defined and visible to the eyes.

Easy-to-maintain − Python's source code is fairly easy-to-maintain.

A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases − Python provides interfaces to all major commercial databases.

GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type. This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point. If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program. Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program. With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double. With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values. For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

20 Python libraries

1. Requests. The most famous http library written by Kenneth reitz. It's a must have for every python developer.
2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won't use any other.
3. wxPython. A GUI toolkit for python. I have primarily used it in place of tkinter. You will really love it.
4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.
6. BeautifulSoup. I know it's slow but this xml and html parsing library is very useful for beginners.
7. Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.
8. NumPy. How can we leave this very important library ? It provides some advance math functionalities to python.
9. SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
10. matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
11. Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.
12. Pyglet. A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made

13. pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI’s for my python scripts.
14. pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.
15. Scapy. A packet sniffer and analyzer for python made in python.
16. pywin32. A python library which provides some useful methods and classes for interacting with windows.
17. nltk. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But it’s capacity is beyond that. Do check it out.
18. nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.
19. SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.
20. IPython. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

NumPy

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank.

- Offers Matlab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra etc.
- matplotlib
- High quality plotting library.

Python class and objects

These are the building blocks of OOP. class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something. Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects. When I first learned C and C++, I did great; functions just made sense for me. Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered. Once you learn OOP, you'll realize that it's actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing. One thing to note about Python and OOP: it's not mandatory to use objects in your code in a way that works best; maybe you don't need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want. As you've already seen, Python can do just fine with functions. Unlike languages such as Java, you aren't tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here's a brief list of Python OOP ideas:

- The class statement creates a class object and gives it a name. This creates a new namespace.
- Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.
- Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.
- Calling a class (just like a function) creates a new instance of the class.
- This is where the multiple copies part comes in.
- Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.
- Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it. To elaborate, by sub classing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components. As we've seen, an instance of a class inherits the attributes of that class. However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses. The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all. Operator Overloads Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc. Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use. User-made classes can override nearly all of Python's built-in operation methods

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program's flow, either intentionally or due to errors. They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop. Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python. Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

- Accessing a non−existent dictionary key will raise a KeyError exception.
- Searching a list for a non−existent value will raise a ValueError exception
- Calling a non−existent method will raise an AttributeError exception.
- Referencing a non−existent variable will raise a NameError exception.
- Mixing datatypes without coercion will raise a TypeError exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files. This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions. Your program is usually short enough to not be hurt too much if an exception occurs. Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem. However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception. It operates exactly as a system-caused exception, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing. Because exceptions aren't supposed to happen very often, they aren't processed until they occur. Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions. However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time. Proper use of exceptions can help the performance of your program. The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing. Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions. You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you. They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free. Making your own exceptions involves object-oriented programming, which will be covered in the next chapter**.**

To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class. To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

Testing code

As indicated above, code is usually developed in a file using an editor. To test the code, import it into a Python session and try to run it. Usually there is an error, so you go back to the file, make a correction, and test again. This process is repeated until you are satisfied that the code works. The entire process is known as the development cycle. There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function. You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or can not return one or more values. There are three types of functions in python:

help(),min(),print().

Python Namespace

Generally speaking, a namespace (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and family name (surname). An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems. The same file name can be used in different directories, the files can be uniquely accessed via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace. This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs. Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- global names of a module
- local names in a function or method invocation
- built-in names: this namespace contains built-in functions (e.g. abs (), cmp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language. What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML. This is recommended by the World Wide Web Consortium and available as an open standard. XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone. XML Parser Architectures and APIs The Python standard library provides a minimal but useful set of interfaces to work with XML. The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces. Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document. This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory. Document Object Model DOM API: This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document. SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files. SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possibly comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, templating engines and object.

Web framework resources

When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing. Frameworks is a really well-done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well. what is a web framework? is an in-depth explanation of what web frameworks are and their relation to web servers.

Django vs Flash vs Pyramid:

Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks. This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases. Python's web frameworks benchmarks are a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless. What web frameworks do you use and why are they awesome? is a language agnostic Reddit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks. This user-voted question & answer site asked "What are the best general-purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist

1. Choose a major Python web framework (Django or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.

2. Work through a detailed tutorial found within the resource links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the deployment section to make it accessible on the web.

3.4 SYSTEM REQUIREMENTS

3.4.1 HARDWARE REQUIREMENTS:

➢ System	:	Pentium Dual Core.
➢ Hard Disk	:	120 GB.
➢ Monitor	:	15'' LED
➢ Input Devices	:	Keyboard, Mouse
➢ Ram	:	1GB.

3.4.2 SOFTWARE REQUIREMENTS:

➢ Operating system	:	Windows 7.
➢ Coding Language	:	Python
➢ Tool	:	Anaconda, Visual studio code
➢ Libraries	:	OpenCV

CHAPTER-4

SYSTEM DESIGN

4.1 System Architecture

Architecture Flow:

Below architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.

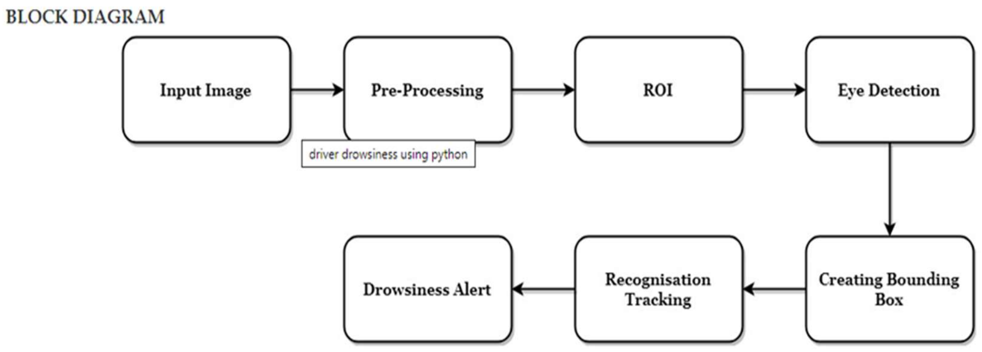


Figure 3.1: Architecture diagram

3-Tier Architecture:

The three-tier software architecture (a three-layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two-tier architecture) by providing functions such as queuing, application execution, and database staging.

The three-tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three-layer architectures a popular choice for Internet applications and net-centric information systems

.

Advantages of Three-Tier:

- Separates functionality from presentation.
- Clear separation – better understanding.
- Changes limited to well define components.
- Can be running on WWW.

- Effective network performance.

SYSTEM DESIGN

System Design Introduction:

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

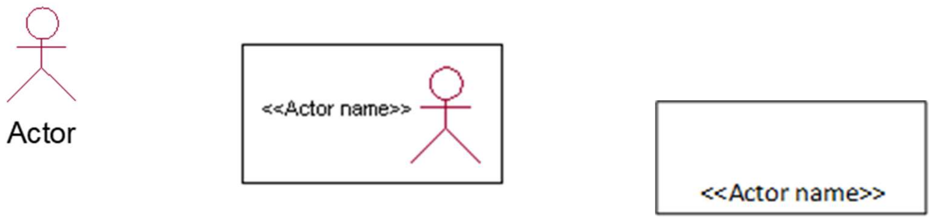
4.2 UML Diagrams

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system?

- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

Questions to identify actors:

- Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

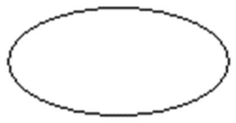
The actors identified in this system are:

a. System Administrator
b. Customer
c. Customer Care

Identification of usecases:

Usecase: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor

Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal

- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar.
- This makes the description less ambiguous
- Questions to identify use cases:
- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What usecases will support and maintains the system?

Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

4.2.1 Construction of Usecase diagrams:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

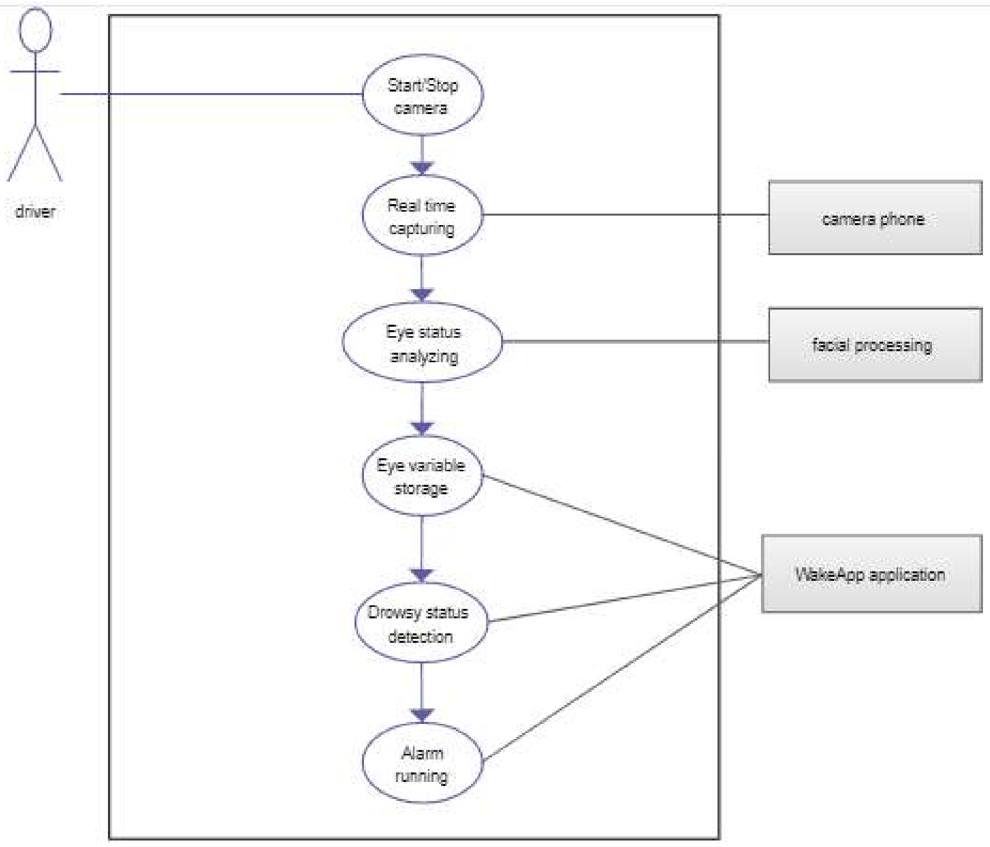


Figure 4.2.1 Use Case Diagram

4.2.2 SEQUENCE DIAGRAMS:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

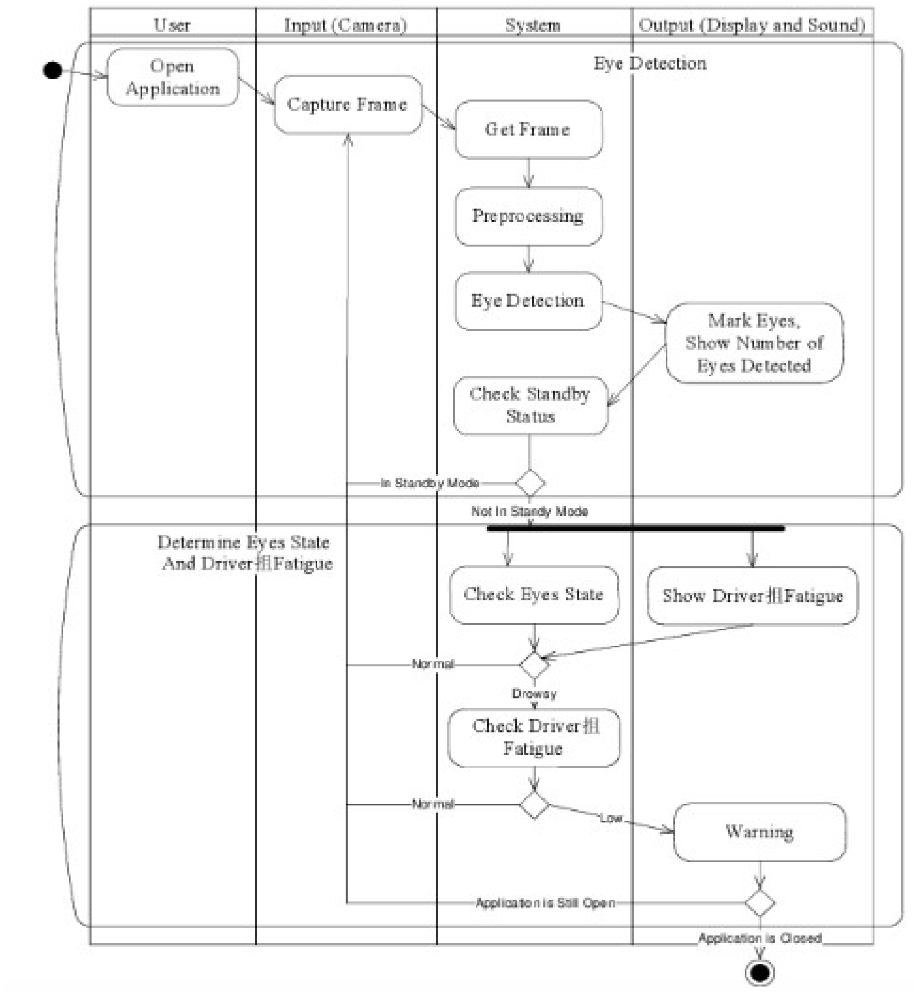


Figure 4.2.2: Sequence diagram

4.2.3. CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) 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 the classes. It explains which class contains information.

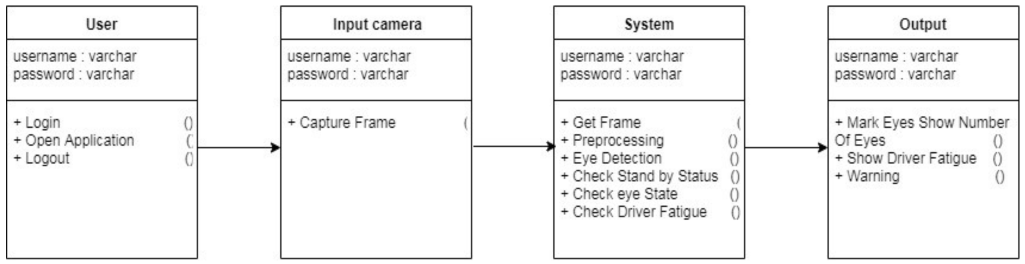


Figure 4.2.3: Class Diagram

4.2.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

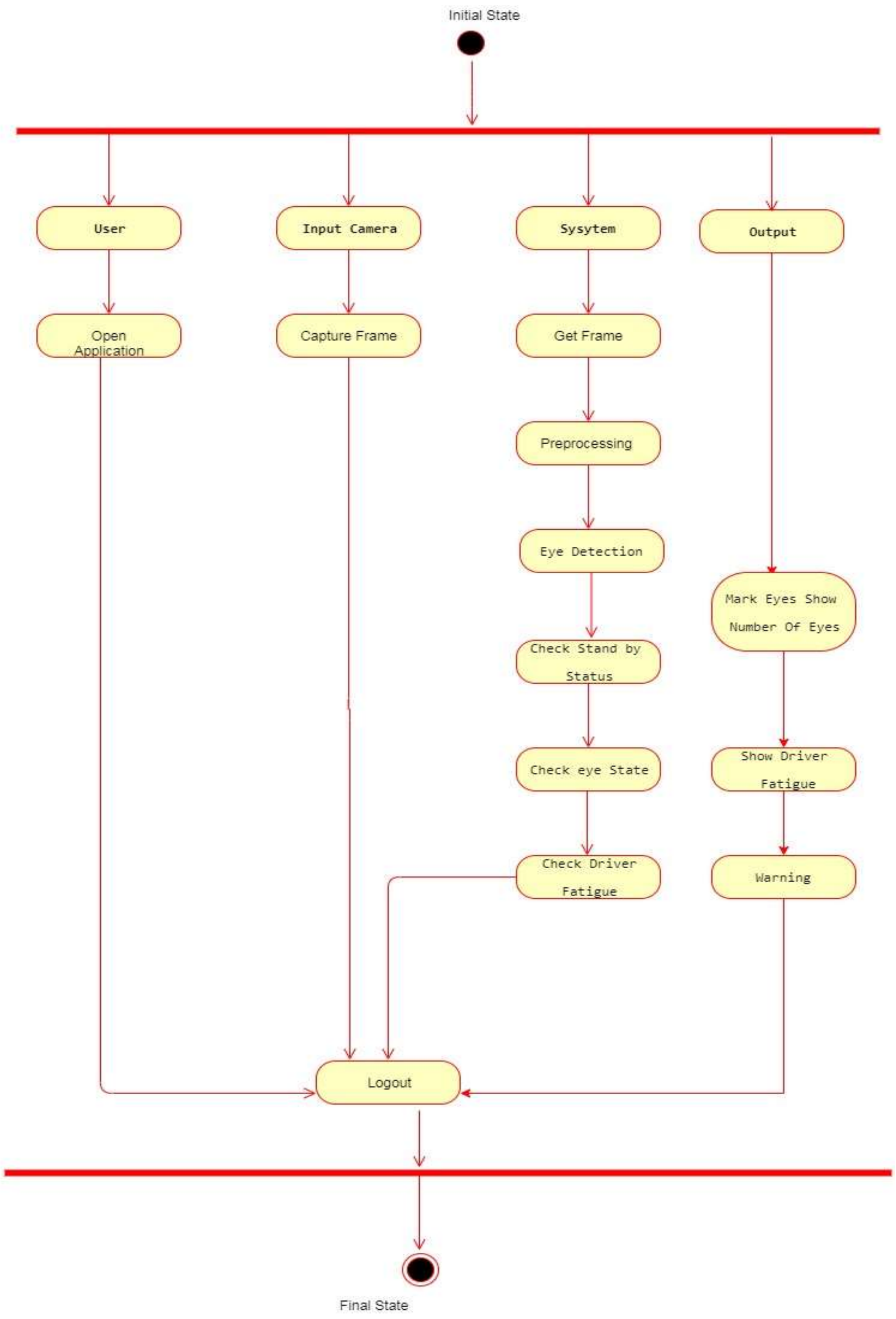


Figure 4.2.4: Activity Diagram

CHAPTER-5

TESTING &VALIDATION

5.1 INTRODUCTION:

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

1. A successful test is one that uncovers an as yet undiscovered error.
2. A good test case is one that has probability of finding an error, if it exists.
3. The test is inadequate to detect possibly present errors.
4. The software more or less confirms to the quality and reliable standards.

5.2. Levels of Testing:

Code testing:

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing:

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing:

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing

2. White Box Testing

BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications. In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

Input
Black Box
Output

The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

- Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
- Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing – Regression testing is done after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure.It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

What do you verify in White Box Testing?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do when testing an application using the white box testing technique:

STEP 1: UNDERSTAND THE SOURCE CODE

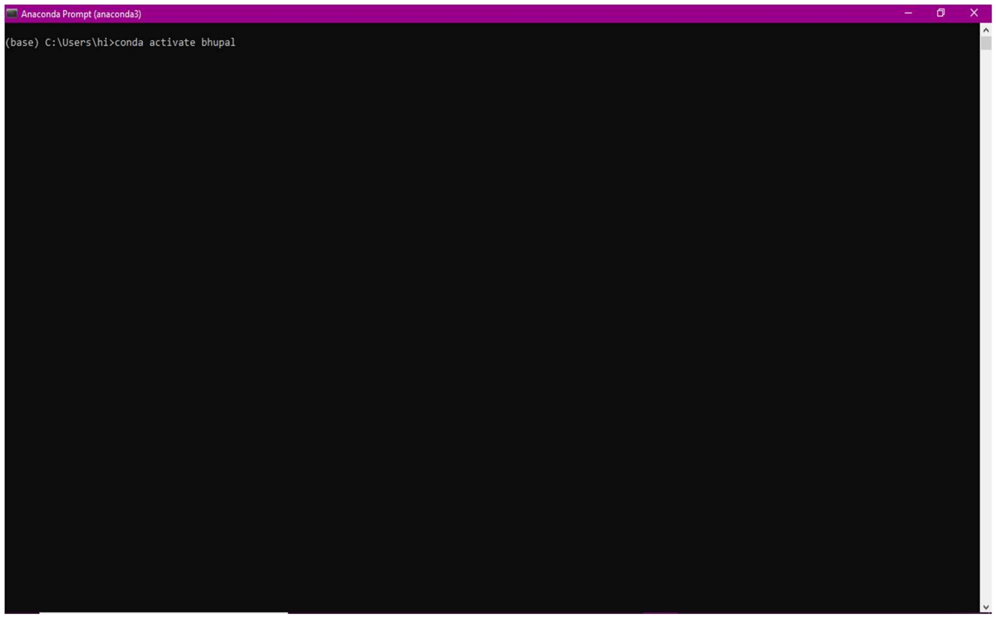
The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2: CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

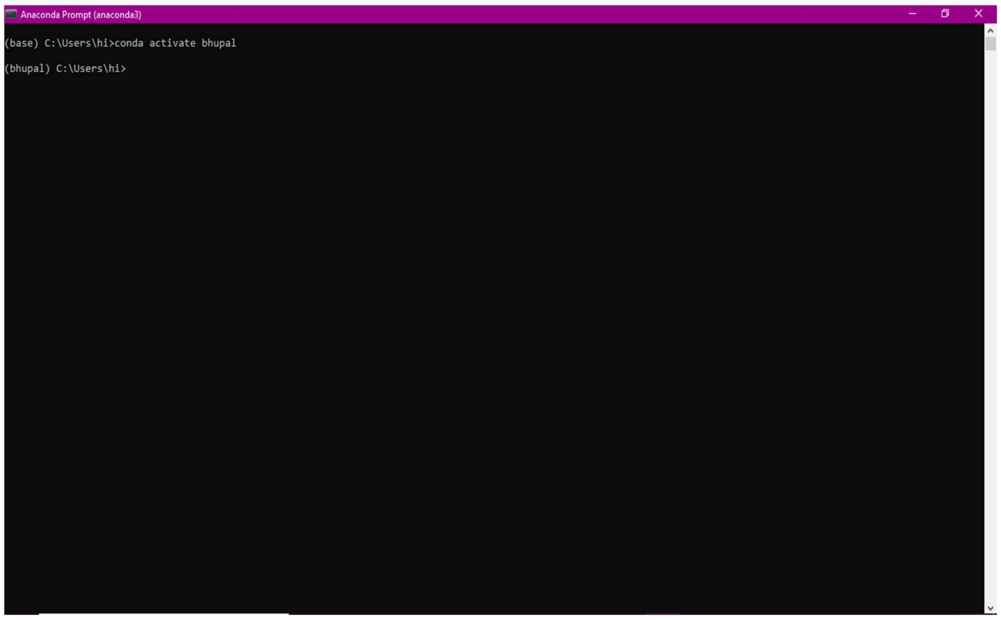
CHAPTER-6

SCREEN SHOTS



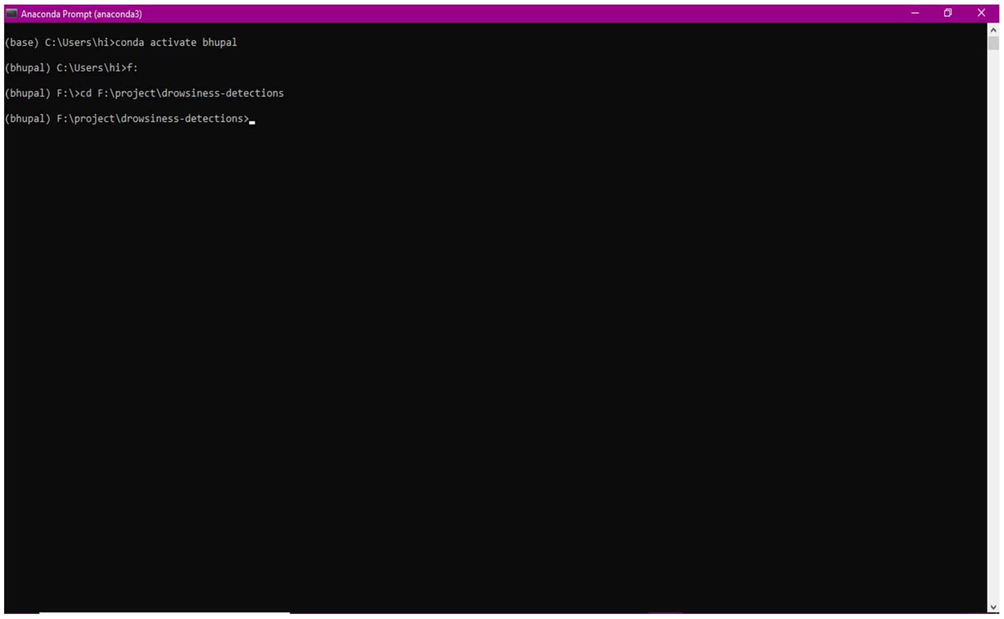
```
(base) C:\Users\hi>conda activate bhupal
```

Screenshot 6.1: open anaconda prompt



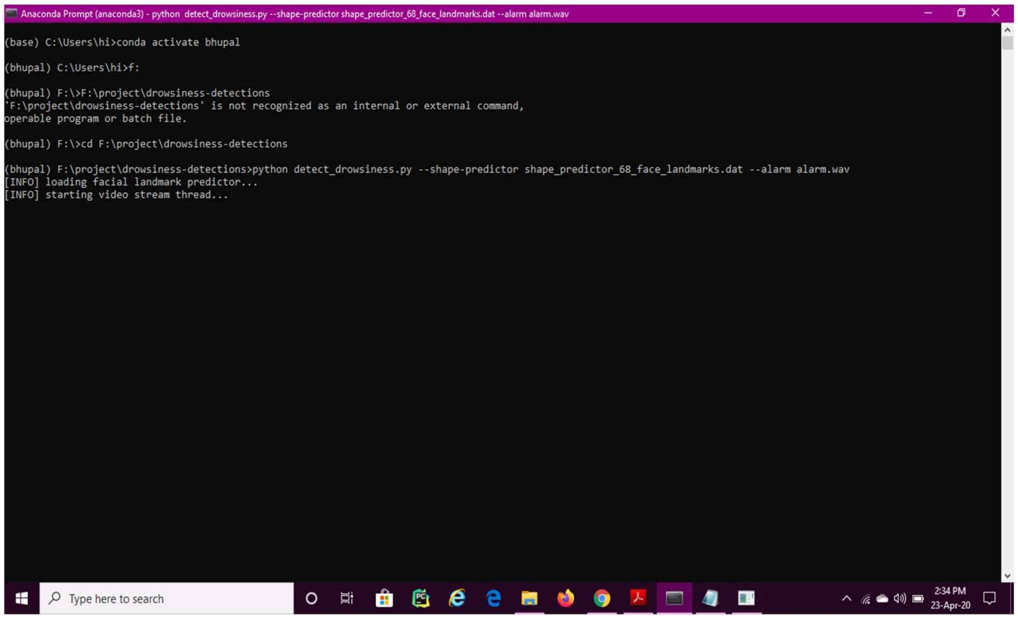
```
(base) C:\Users\hi>conda activate bhupal

(bhupal) C:\Users\hi>
```

Screenshot 6.2: Activate anaconda environment



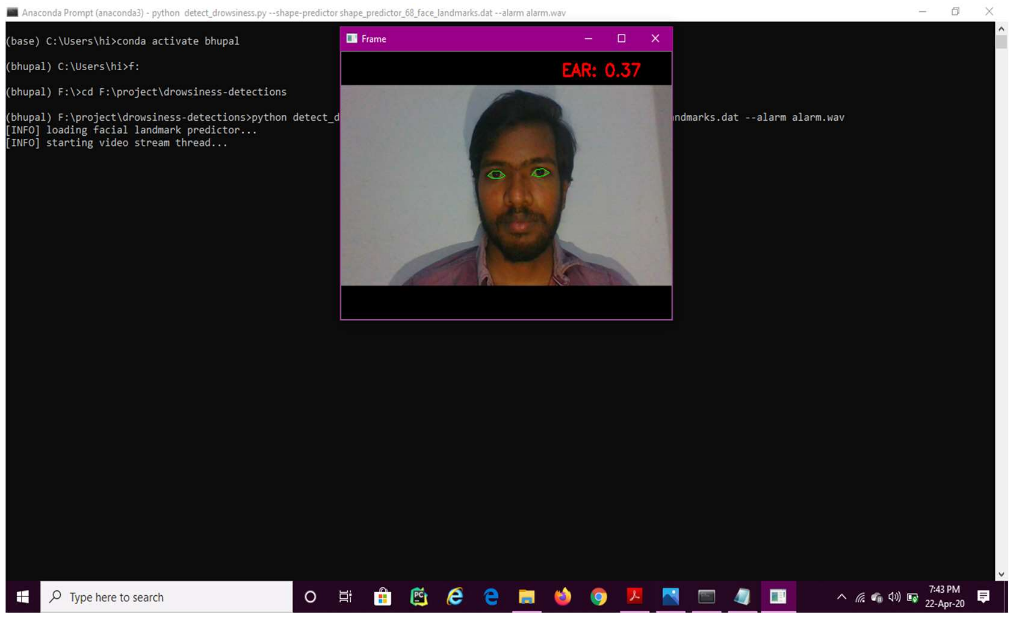
```
(base) C:\Users\hi>conda activate bhupal

(bhupal) C:\Users\hi>f:

(bhupal) F:\>cd F:\project\drowsiness-detections

(bhupal) F:\project\drowsiness-detections>_
```

Screenshot 6.3: set path of the files located


Anaconda Prompt (anaconda3) - python detect_drowsiness.py --shape-predictor shape_predictor_68_face_landmarks.dat --alarm alarm.wav
Type here to search
2:34 PM
23-Apr-20

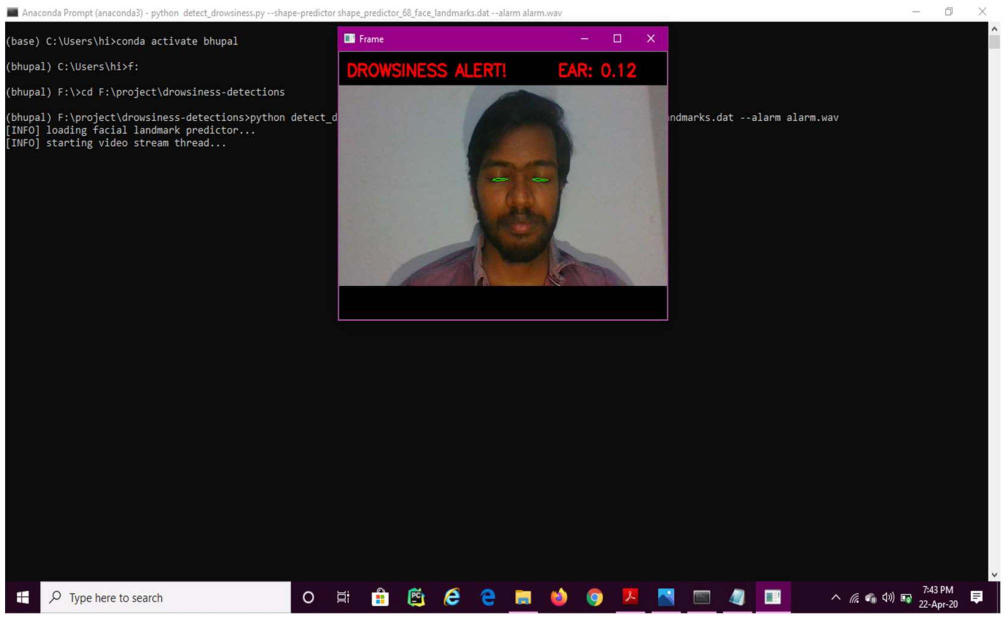
```
(base) C:\Users\hi>conda activate bhupal

(bhupal) C:\Users\hi>f:

(bhupal) F:\>F:\project\drowsiness-detections
'F:\project\drowsiness-detections' is not recognized as an internal or external command,
operable program or batch file.

(bhupal) F:\>cd F:\project\drowsiness-detections

(bhupal) F:\project\drowsiness-detections>python detect_drowsiness.py --shape-predictor shape_predictor_68_face_landmarks.dat --alarm alarm.wav
[INFO] loading facial landmark predictor...
[INFO] starting video stream thread...
```

Screenshot 6.4: Execute the files


Anaconda Prompt (anaconda3) - python detect_drowsiness.py --shape-predictor shape_predictor_68_face_landmarks.dat --alarm alarm.wav
Frame
EAR: 0.37
Type here to search
7:43 PM
22-Apr-20

```
(base) C:\Users\hi>conda activate bhupal

(bhupal) C:\Users\hi>f:

(bhupal) F:\>cd F:\project\drowsiness-detections

(bhupal) F:\project\drowsiness-detections>python detect_d                                        ndmarks.dat --alarm alarm.wav
[INFO] loading facial landmark predictor...
[INFO] starting video stream thread...
```

Screenshot 6.5: Apply facial landmark localization to extract eyes regions from the face


Anaconda Prompt (anaconda3) - python detect_drowsiness.py --shape-predictor shape_predictor_68_face_landmarks.dat --alarm alarm.wav
Frame
DROWSINESS ALERT!
EAR: 0.12
Type here to search
7:43 PM
22-Apr-20

```
(base) C:\Users\hi>conda activate bhupal

(bhupal) C:\Users\hi>f:

(bhupal) F:\>cd F:\project\drowsiness-detections

(bhupal) F:\project\drowsiness-detections>python detect_d                                        ndmarks.dat --alarm alarm.wav
[INFO] loading facial landmark predictor...
[INFO] starting video stream thread...
```

Screenshot 6.6: Compute the EAR and sound an alarm if eyes have been closed for a sufficiently long enough time

CHAPTER-7

CONCLUSION

Future works: -

In the real time driver fatigue detection system, it is required to slow down a vehicle automatically when fatigue level crosses a certain limit. Instead of threshold drowsiness level it is suggested to design a continuous scale driver fatigue detection system. It monitors the level of drowsiness continuously and when this level exceeds a certain value a signal is generated which controls the hydraulic braking system of the vehicle.

Hardware components required-

Dedicated hardware for image acquisition processing and display 47

Interface support with the hydraulic braking system which includes relay, timer, stepper motor and a linear actuator.

Function

When drowsiness level exceeds a certain limit then a signal is generated which is communicated to the relay through the parallel port(parallel data transfer required for faster results).The relay drives the on delay timer and this timer in turn runs the stepper motor for a definite time period .The stepper motor is connected to a linear actuator. The linear actuator converts rotational movement of stepper motor to linear motion. This linear motion is used to drive a shaft which is directly connected to the hydraulic braking system of the vehicle. When the shaft moves it applies the brake and the vehicle speed decreases. Since it brings the vehicle speed down to a controllable limit , the chances of accident occurrence is greatly reduced which is quite helpful for avoiding crashes caused by drowsiness related cases.

Conclusion: -

Thus, we have successfully designed a prototype drowsiness detection system using OpenCV software and Haar Classifiers. The system so developed was successfully tested, its limitations identified and a future plan of action developed.

Code

```
# USAGE
# python detect_drowsiness.py --shape-predictor
shape_predictor_68_face_landmarks.dat
# python detect_drowsiness.py --shape-predictor
shape_predictor_68_face_landmarks.dat --alarm alarm.wav

# import the necessary packages
fromscipy.spatial import distance as dist
fromimutils.video import VideoStream
fromimutils import face_utils
from threading import Thread
importnumpy as np
importplaysound
importargparse
importimutils
import time
importdlib
import cv2

defsound_alarm(path):
      # play an alarm sound
      playsound.playsound(path)

defeye_aspect_ratio(eye):
      # compute the euclidean distances between the two sets of
      # vertical eye landmarks (x, y)-coordinates
      A = dist.euclidean(eye[1], eye[5])
      B = dist.euclidean(eye[2], eye[4])

      # compute the euclidean distance between the horizontal
      # eye landmark (x, y)-coordinates
      C = dist.euclidean(eye[0], eye[3])

      # compute the eye aspect ratio
      ear = (A + B) / (2.0 * C)

      # return the eye aspect ratio
      return ear

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
      help="path to facial landmark predictor")
ap.add_argument("-a", "--alarm", type=str, default="",
      help="path alarm .WAV file")
ap.add_argument("-w", "--webcam", type=int, default=0,
      help="index of webcam on system")
args = vars(ap.parse_args())

# define two constants, one for the eye aspect ratio to
indicate
# blink and then a second constant for the number of
consecutive
```

```
# frames the eye must be below the threshold for to set off
the
# alarm
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 48

# initialize the frame counter as well as a boolean used to
# indicate if the alarm is going off
COUNTER = 0
ALARM_ON = False

# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])

# grab the indexes of the facial landmarks for the left and
# right eye, respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

# start the video stream thread
print("[INFO] starting video stream thread...")
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)

# loop over frames from the video stream
while True:
	# grab the frame from the threaded video file stream,
resize
	# it, and convert it to grayscale
	# channels)
	frame = vs.read()
	frame = imutils.resize(frame, width=450)
	gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

	# detect faces in the grayscale frame
	rects = detector(gray, 0)

	# loop over the face detections
	forrect in rects:
		# determine the facial landmarks for the face
region, then
		# convert the facial landmark (x, y)-coordinates to
a NumPy
		# array
		shape = predictor(gray, rect)
		shape = face_utils.shape_to_np(shape)

		# extract the left and right eye coordinates, then
use the
		# coordinates to compute the eye aspect ratio for
both eyes
		leftEye = shape[lStart:lEnd]
		rightEye = shape[rStart:rEnd]
```

```
           leftEAR = eye_aspect_ratio(leftEye)
           rightEAR = eye_aspect_ratio(rightEye)

           # average the eye aspect ratio together for both
eyes
           ear = (leftEAR + rightEAR) / 2.0

           # compute the convex hull for the left and right
eye, then
           # visualize each of the eyes
           leftEyeHull = cv2.convexHull(leftEye)
           rightEyeHull = cv2.convexHull(rightEye)
           cv2.drawContours(frame, [leftEyeHull], -1, (0, 255,
0), 1)
           cv2.drawContours(frame, [rightEyeHull], -1, (0, 255,
0), 1)

           # check to see if the eye aspect ratio is below the
blink
           # threshold, and if so, increment the blink frame
counter
           if ear < EYE_AR_THRESH:
                COUNTER += 1

                # if the eyes were closed for a sufficient
number of
                # then sound the alarm
                if COUNTER >= EYE_AR_CONSEC_FRAMES:
                     # if the alarm is not on, turn it on
                     if not ALARM_ON:
                          ALARM_ON = True

                          # check to see if an alarm file was
supplied,
                          # and if so, start a thread to have
the alarm
                          # sound played in the background
                          if args["alarm"] != "":
                               t = Thread(target=sound_alarm,
                                    args=(args["alarm"],))
                               t.deamon = True
                               t.start()

                     # draw an alarm on the frame
                     cv2.putText(frame, "DROWSINESS ALERT!",
(10, 30),
                          cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
0, 255), 2)

           # otherwise, the eye aspect ratio is not below the
blink
           # threshold, so reset the counter and alarm
           else:
                COUNTER = 0
                ALARM_ON = False
```

```
		# draw the computed eye aspect ratio on the frame to
help
		# with debugging and setting the correct eye aspect
ratio
		# thresholds and frame counters
		cv2.putText(frame, "EAR: {:.2f}".format(ear), (300,
30),
			cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

	# show the frame
	cv2.imshow("Frame", frame)
	key = cv2.waitKey(1) & 0xFF

	# if the `q` key was pressed, break from the loop
	if key == ord("q"):
		break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

CHAPTER-8

REFERENCES

REFERENCES:

1. http://www.ee.ryerson.ca/~phiscock/thesis/drowsy-detector/drowsy-detector.pdf
2. http://www.cnblogs.com/skyseraph/archive/2011/02/24/1963765.html
3. http://www.scribd.com/doc/15491045/Learning-OpenCV-Computer-Vision-with-the-OpenCV-Library
4. http://opencv.willowgarage.com/documentation/reading_and_writing_images_and_video.html
5. http://www.scribd.com/doc/46566105/opencv
6. Learning OpenCV by Gary Brad ski and Adrian Koehler
7. http://note.sonots.com/SciSoftware/haartraining.html