VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belgaum – 590 018

A PROJECT REPORT ON

“Optimizing Power Consumption using AI”

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING BY

Diwanshu Sharma(1CR16IS124)

Under the guidance of

Mrs. Vidya U

(Associate Professor, Dept. of ISE, CMRIT)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CMR INSTITUTE OF TECHNOLOGY,

BANGALORE-37

Chapter 1

PREAMBLE

1.1 Introduction

The world's transition to clean renewable energy, such as solar, wind, and biofuels, relies on our actions.We must engage in this change by optimizing the power consumed by our homes, buildings, and machines. Indeed, studies have shown that continuously adjusting the latter's operations, and implementing energy-efficiency strategies, could be very beneficial. Energy use may be reduced by 30%. But how can it be achieved? Leveraging artificial intelligence is the answer.

Computer vision is an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.[1][2][3] Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems.

Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, and image restoration.

The Internet of Things (IOT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

The definition of the Internet of Things has evolved due to the convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems. Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), and others all contribute to enabling the Internet of Things. In the consumer market, IOT technology is most synonymous with products pertaining to the concept of the "smart home", covering devices and appliances (such as lighting fixtures, thermostats, home security systems and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers.

Existing System:

1) N. Shribhagat Varma paper on Automatic Electrical Appliances control based on Image Processing:

They design a power management system which will sense if the room is vacant and accordingly turn the lights off. To avoid this sheer loss of money and resources, automated power management system looks to detect whether the room is empty and accordingly switch off the lights and fans. The camera captures live feed of the room and gives it to the computer to process it. The face detection module then detects human presence.

The strength and contribution of this work lies in the combination of a large number of sensors readings which allows deriving higher level semantics as compared to reacting on single sensor readings only.

2) Viola Jones paper on Rubust Real Time Object Detection:

This paper describes a visual object detection framework that is capable of processing images extremely rapidly while achieving high detection rates. There are three key contributions. The first is the introduction of a new image representation called the "Integral Image" which allows the features used by our detector to be computed very quickly. The second is a learning algorithm, based on AdaBoost, which selects a small number of critical visual features and yields extremely efficient classifiers. The third contribution is a method for combining classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. A set of experiments in the domain of face detection are presented. The system yields face detection performance comparable to the best previous systems. It is implemented on a conventional desktop, face detection proceeds at 15 frames per second. Accordingly "Manoj Kumar Asst. Professor, Dept of CSE" they calculated all the various steps done and various results are compared with test cases. Students can be at corner or they can be at in front in a group etc.

Test case I display two students are sitting and their subtracted image is another image also test case II display two students are sitting and their subtracted image is shown in another image. The study shows that this method is helpful in saving electricity.

This method is very cheap, efficient and can reduce wastage of power. This will consistently detect that is there any person in a classroom and auditorium and hence saves electricity.

1.2.1 Drawbacks

1. Although the coder designs an optimised code, the code readability is very inadequate and cannot be modified by the user even if required. Any change to the code requires change in the model itself.
2. The amount of complexity and requirement requested by the code has led to the reduced usage of this application and manual code conversion is preferred

1.3 Proposed System

1. Our aim is to develop a Model that controls the electricity appliances based on the human activity .
2. The Model will take human action as the input to the model and instruct the control system to work accordingly .

1.4 Plan of Implementation

1. The project can be broken down into 4 Major states.
2. The first stage of project is to reorganize the environment.
3. The second stage is based on the certain properties acquired, predict the existence of a human.
4. The third stage is to instruct the control system.
5. The Final stage is Control system will manage the electricity appliances accordingly.

Chapter 2

LITERATURE SURVEY

In order to get required knowledge about various concepts related to the present application, existing literature were studied. Some of the important conclusions were made through those are listed below.

1. **Rapid Object Detection using a Boosted Cascade of Simple Features -** This paper describes a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates
2. **A General Framework for Object Detection** - This paper presents a general trainable framework for object detection in static images of cluttered scenes. The detection technique we develop is based on a wavelet representation of an object class derived from a statistical analysis of the class instances. By learning an object class in terms of a subset of an overcomplete dictionary of wavelet basis functions, we derive a com- pact representation of an object class which is used as an input to a suppori vector machine classifier.
3. **Boosting Image Retrieval** - We present an approach for image retrieval using a very large number of highly selective features and efficient online learning. Our approach is predicated on the assumption that each image is generated by a sparse set of visual “causes” and that images which are visually similar share causes. We propose a mechanism for computing a very large number of highly selective features which capture some aspects of this causal structure (in our implementation there are over 45,000 highly selective features). At query time a user selects a few example images, and a technique known as “boosting” is used to learn a classification function in this feature space.

Chapter 3

THEORITICAL BACKGROUND

Theoretical background highlighting some topics related to the project work is given below. The description contains several topics which are worth to discuss and also highlight some of their limitation that encourage going on finding solution as well as highlights some of their advantages for which reason these topics and their features are used in this project.

3.1 Image Processing

In computer science, digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems. The generation and development of digital image processing are mainly affected by three factors: first, the development of computers; second, the development of mathematics (especially the creation and improvement of discrete mathematics theory); third, the demand for a wide range of applications in environment, agriculture, military, industry and medical science has increased.

3.2 Viola–Jones object detection framework

The Viola–Jones object detection framework is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection.

3.3 Problem description

The problem to be solved is detection of faces in an image. A human can do this easily, but a computer needs precise instructions and constraints. To make the task more manageable, Viola–Jones requires full view frontal upright faces. Thus in order to be detected, the entire face must point towards the camera and should not be tilted to either side. While it seems these constraints could diminish the algorithm's utility somewhat, because the detection step is most often followed by a recognition step, in practice these limits on pose are quite acceptable.

3.4 Components of the framework

3.4.1 Feature types and evaluation

The characteristics of Viola–Jones algorithm which make it a good detection algorithm are:

- Robust – very high detection rate (true-positive rate) & very low false-positive rate always.
- Real time – For practical applications at least 2 frames per second must be processed.
- Face detection only (not recognition) - The goal is to distinguish faces from non-faces (detection is the first step in the recognition process).

The algorithm has four stages:

1. Haar Feature Selection
2. Creating an Integral Image
3. Adaboost Training
4. Cascading Classifiers

The features sought by the detection framework universally involve the sums of image pixels within rectangular areas. As such, they bear some resemblance to Haar basis functions, which have been used previously in the realm of image-based object detection.

However, since the features used by Viola and Jones all rely on more than one rectangular area, they are generally more complex. The figure on the right illustrates the four different types of features used in the framework. The value of any given feature is the sum of the pixels within clear rectangles subtracted from the sum of the pixels within shaded rectangles. Rectangular features of this sort are primitive when compared to alternatives such as steerable filters. Although they are sensitive to vertical and horizontal features, their feedback is considerably coarser.

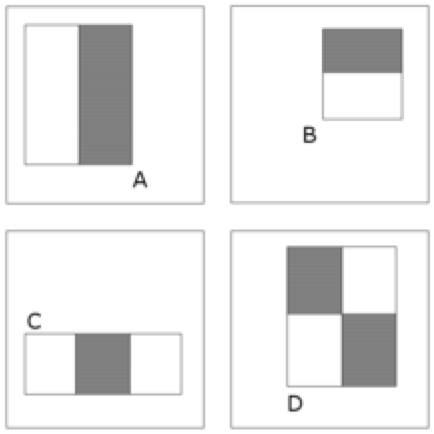


Figure 3.1: Example rectangle features shown relative to the enclosing detection window

3.4.1.1 Haar Features

All human faces share some similar properties. These regularities may be matched using Haar Features.

A few properties common to human faces:

- The eye region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.

Composition of properties forming matchable facial features:

- Location and size: eyes, mouth, bridge of nose
- Value: oriented gradients of pixel intensities

The four features matched by this algorithm are then sought in below Figures.

Rectangle features:

- Value = Σ (pixels in black area) - Σ (pixels in white area)
- Three types: two-, three-, four-rectangles, Viola & Jones used two-rectangle features
- For example: the difference in brightness between the white & black rectangles over a specific area
- Each feature is related to a special location in the sub-window

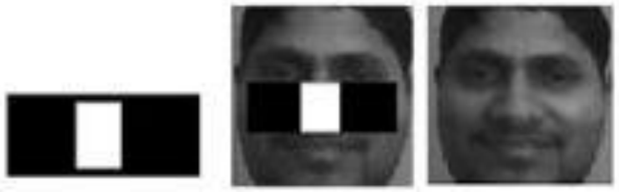


Figure 3.2 Haar Feature that looks similar to the bridge of the nose is applied onto the face

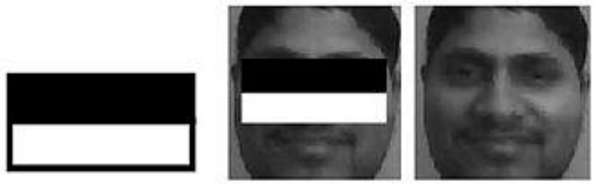


Figure 3.3 Haar Feature that looks similar to the eye region which is darker than the upper cheeks is applied onto a face

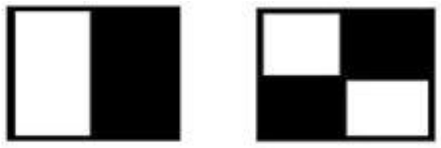


Figure 3.4 3rd and 4th kind of Haar Feature

3.4.1.2 Summed area table

An image representation called the integral image evaluates rectangular features in constant time, which gives them a considerable speed advantage over more sophisticated alternative features. Because each feature's rectangular area is always adjacent to at least one other rectangle, it follows that any two-rectangle feature can be computed in six array references, any three-rectangle feature in eight, and any four-rectangle feature in nine.

3.4.2 Learning algorithm

The speed with which features may be evaluated does not adequately compensate for their number, however. For example, in a standard 24x24 pixel sub-window, there are a total of M = 162,336[4] possible features, and it would be prohibitively expensive to evaluate them all when testing an image. Thus, the object detection framework employs a variant of the learning algorithm AdaBoost to both select the best features and to train classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers.

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

Each weak classifier is a threshold function based on the feature fj.

$$h_j(\mathbf{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$

The threshold value theta j and the polarity sj subset of +or- (1) are determined in the training, as well as the coefficients alpha j.

Here a simplified version of the learning algorithm is reported:

Input: Set of N positive and negative training images with their labels (x^i, y^i). If image i is a face y = -1 0r +1.

1. Initialization: assign a weight w1^I = 1/N to each image i.
2. For each feature fj with j = 1,…..,M.
    1. Renormalize the weights such that they sum to one.

2. Apply the feature to each image in the training set, then find the optimal threshold and polarity theta(j), s(j) that minimizes the weighted classification error. That is

1. $$\theta_j, s_j = \arg\min_{\theta,s} \sum_{i=1}^{N} w_j^i \varepsilon_j^i \text{ where } \varepsilon_j^i = \begin{cases} 0 & \text{if } y^i = h_j(\mathbf{x}^i, \theta_j, s_j) \\ 1 & \text{otherwise} \end{cases}$$

3. Assign a weight alpha(j) to h(j) that is inversely proportional to the error rate. In this way best classifiers are considered more.
4. The weights for the next iteration, i.e. w(j+1)^I , are reduced for the images *i* that were correctly classified.

3. Set the final classifier to

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

3.4.3 Cascade architecture

- On average only 0.01% of all sub-windows are positive (faces)
- Equal computation time is spent on all sub-windows
- Must spend most time only on potentially positive sub-windows.
- A simple 2-feature classifier can achieve almost 100% detection rate with 50% FP rate.
- That classifier can act as a 1st layer of a series to filter out most negative windows
- 2nd layer with 10 features can tackle “harder” negative-windows which survived the 1st layer, and so on...
- A cascade of gradually more complex classifiers achieves even better detection rates. The evaluation of the strong classifiers generated by the learning process can be done quickly, but it isn't fast enough to run in real-time. For this reason, the strong classifiers are arranged in a cascade in order of complexity, where each successive classifier is trained only on those selected samples which pass through the preceding classifiers. If at any stage in the cascade a classifier rejects the sub-window under inspection, no further processing is performed and continue on searching the next sub-window. The cascade therefore has the

form of a degenerate tree. In the case of faces, the first classifier in the cascade – called the attentional operator – uses only two features to achieve a false negative rate of approximately 0% and a false positive rate of 40%. The effect of this single classifier is to reduce by roughly half the number of times the entire cascade is evaluated.

In cascading, each stage consists of a strong classifier. So all the features are grouped into several stages where each stage has certain number of features.

The job of each stage is to determine whether a given sub-window is definitely not a face or may be a face. A given sub-window is immediately discarded as not a face if it fails in any of the stages.

A simple framework for cascade training is given below:

- f = the maximum acceptable false positive rate per layer.
- d = the minimum acceptable detection rate per layer.
- Ftarget = target overall false positive rate.
- P = set of positive examples.
- N = set of negative examples.

```
F(0) = 1.0; D(0) = 1.0; i = 0

while F(i) > Ftarget
   increase i
   n(i) = 0; F(i)= F(i-1)

   while F(i) > f × F(i-1)
      increase n(i)
      use P and N to train a classifier with n(I) features using AdaBoost
      Evaluate current cascaded classifier on validation set to determine F(i) and D(i)
      decrease threshold for the ith classifier (i.e. how many weak classifiers need to accept for strong classifier to accept)
         until the current cascaded classifier has a detection rate of at least d × D(i-1) (this also affects F(i))
   N = ∅
   if F(i) > Ftarget then
      evaluate the current cascaded detector on the set of non-face images
```

and put any false detections into the set N.

The cascade architecture has interesting implications for the performance of the individual classifiers. Because the activation of each classifier depends entirely on the behavior of its predecessor, the false positive rate for an entire cascade is:

$$F = \prod_{i=1}^{K} f_i .$$

Similarly, the detection rate is:

$$D = \prod_{i=1}^{K} d_i .$$

Thus, to match the false positive rates typically achieved by other detectors, each classifier can get away with having surprisingly poor performance. For example, for a 32-stage cascade to achieve a false positive rate of 10^{-6}, each classifier need only achieve a false positive rate of about 65%. At the same time, however, each classifier needs to be exceptionally capable if it is to achieve adequate detection rates. For example, to achieve a detection rate of about 90%, each classifier in the aforementioned cascade needs to achieve a detection rate of approximately 99.7%.

3.5 Using Viola–Jones for object tracking

In videos of moving objects, one need not apply object detection to each frame. Instead, one can use tracking algorithms like the KLT algorithm to detect salient features within the detection bounding boxes and track their movement between frames. Not only does this improve tracking speed by removing the need to re-detect objects in each frame, but it improves the robustness as well, as the salient features are more resilient than the Viola-Jones detection framework to rotation and photometric changes.

Chapter 4

SYSTEM REQUIREMENT SPECIFICATION

A System Requirement Specification (SRS) is basically an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point prior to any actual design or development work. The information gathered during the analysis is translated into a document that defines a set of requirements. It gives the brief description of the services that the system should provide and also the constraints under which, the system should operate. Generally, SRS is a document that completely describes what the proposed software should do without describing how the software will do it. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.

SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an ecommerce website and so on) must provide, as well as states any required constraints by which the system must abide. SRS also functions as a blueprint for completing a project with as little cost growth as possible. SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

4.1 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements: -

Following are the functional requirements on the system:

1. Input must be converted into greyscale and of smaller size.
2. Conditions must be checked before sending command to controller.

4.2 Non Functional Requirement

Non functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:-

- Product Requirements
- Organizational Requirements
- User Requirements
- Basic Operational Requirements

4.2.1 Product Requirements

Platform Independency: Standalone executables for embedded systems can be created so the algorithm developed using available products could be downloaded on the actual hardware and executed without any dependency to the development and modeling platform.

Correctness: It followed a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.

Ease of Use: Model Coder provides an interface which allows the user to interact in an easy manner.

Modularity: The complete product is broken up into many modules and well-defined interfaces are developed to explore the benefit of flexibility of the product.

Robustness: This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with utmost relevancy and correctness

Non functional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security & usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

4.2.2 Organizational Requirements

Process Standards: The standards defined by Gvernement of Indian are used to develop the application which is the standard used to protect user Privacy.

Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

4.2.3 User Requirements

- System require the continuous evaluation using camera.
- In case of multiple user system just check for the condition to be satisfied.
- Output of the computation from code instruct the controller.
- Irrespective of the environment system only check for predefined conditions.

4.2.4 System Configuration

H/W System Configuration:

Processor	-	Pentium –IV
Speed	-	1.1 Ghz
RAM	-	4 (min)
Hard Disk	-	20 GB
Monitor	-	SVGA
Camera	-	full HD Recording

S/W System Configuration:

Operating System	: XP/7/8/8.1/10
Coding Language	: Python, C
Tools	: 1) Spyder IDE
	2) OpenCv framework
	3) Arduino IDE

Chapter 5

SYSTEM ANALYSIS

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

5.1 Feasibility Study

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately this condition does not prevail in practical world. So it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill-conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

5.1.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.1.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.2 Analysis

5.2.1 Performance Analysis

For the complete functionality of the project work, the project is run with the help of healthy networking environment. Performance analysis is done to find out whether the proposed system. It is essential that the process of performance analysis and definition must be conducted in parallel.

5.1.2 Technical Analysis

System is only beneficial only if it can be turned into information systems that will meet the organization's technical requirement. Simply stated this test of feasibility asks whether the system will work or not when developed & installed, whether there are any major barriers to implementation. Regarding all these issues in technical analysis there are several points to focus on:-

Changes to bring in the system: All changes should be in positive direction, there will be increased level of efficiency and better customer service.

Required skills: Platforms & tools used in this project are widely used. So the skilled manpower is readily available in the industry.

Acceptability: The structure of the system is kept feasible enough so that there should not be any problem from the user's point of view.

5.1.3 Economical Analysis

Economic analysis is performed to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. For running this system, we need not have any camera or device which are highly economical. So the system is economically feasible enough.

Chapter 6

SYSTEM DESIGN

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customer’s requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

6.1 System development methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

6.1.1 Model phases

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

Requirement Analysis: This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.

System Design: Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm**,** data structure**,** software architecture etc.

Coding: In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine readable compute code.

Implementation: The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation

Testing: In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

Maintenance: The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

6.1.2 Advantages of the Waterfall Model

6.1.2.1 Clear project objectives.

6.1.2.2 Stable project requirements.

6.1.2.3 Progress of system is measurable.

6.1.2.4 Strict sign-off requirements.

6.1.2.5 Helps you to be perfect.

6.1.2.6 Logic of software development is clearly understood.

6.1.2.7 Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.

6.1.2.8 Less human resources required as once one phase is finished those people can start working on to the next phase.

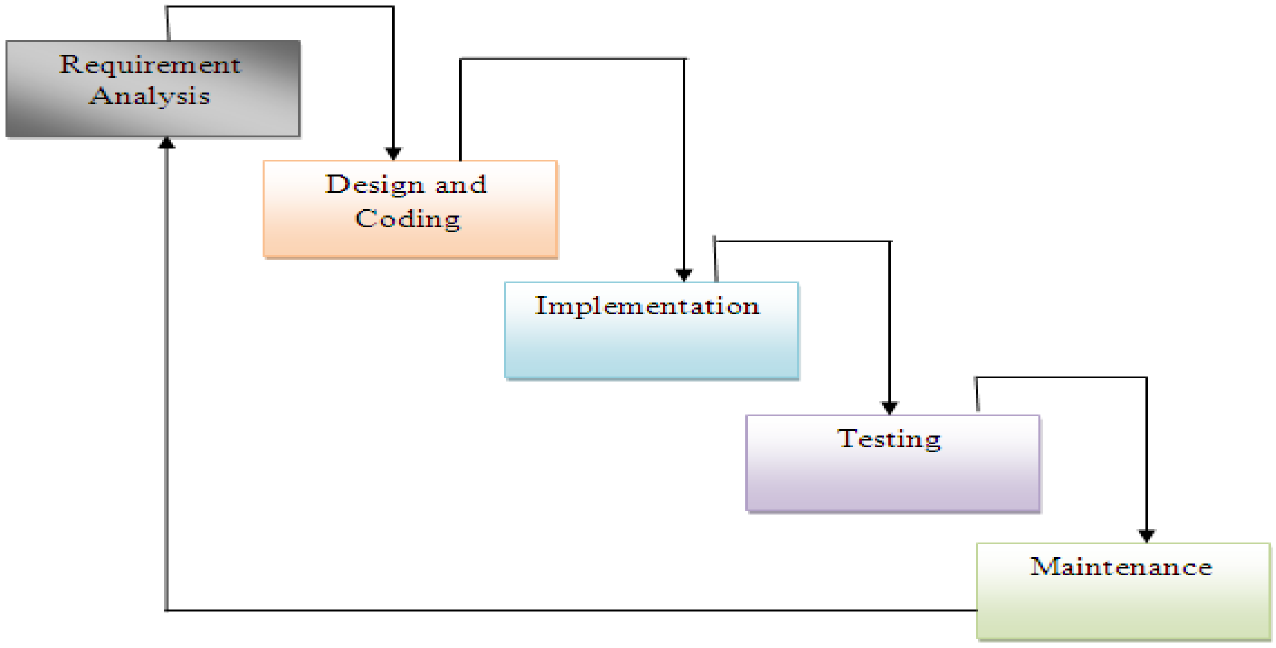
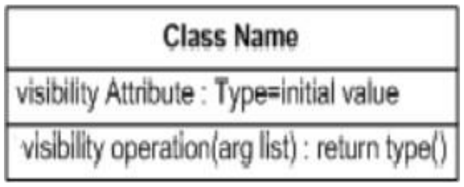
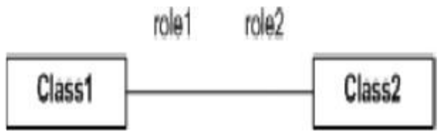
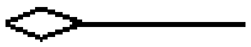
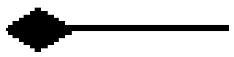
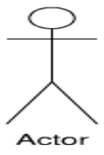


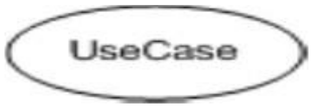
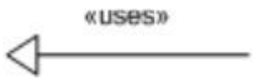
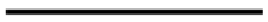
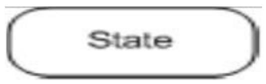
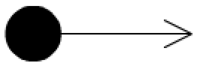
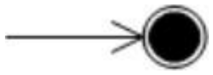
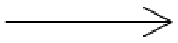
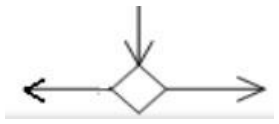
Fig 6.1: Waterfall model

6.2 Design Using UML

Designing UML diagram specifies, how the process within the system communicates along with how the objects with in the process collaborate using both static as well as dynamic UML diagrams since in this ever-changing world of Object Oriented application development, it has been getting harder and harder to develop and manage high quality applications in reasonable amount of time. As a result of this challenge and the need for a universal object modeling language every one could use, the Unified Modeling Language (UML) is the Information industries version of blue print. It is a method for describing the systems architecture in detail.

Easier to build or maintains system, and to ensure that the system will hold up to the requirement changes.

Sl. No	Symbol Name	Symbol	Description
1	Class	 Class Name visibility Attribute : Type=initial value visibility operation(arg list) : return type()	Classes represent a collection of similar entities grouped together.
2	Association		Association represents a static relation between classes.
3	Aggregation		Aggregation is a form of association. It aggregates several classes into a single class.
4	Composition		Composition is a special type of aggregation that denotes a strong ownership between classes.
5	Actor		Actor is the user of the system that reacts with the system.

6	Use Case		A use case is an interaction between system and the external environment.
7	Relation (Uses)		It is used for additional purpose communication.
8	Communication		It is the communication between use cases.
9	State		It represents the state of process. Each state goes through various flows.
10	Initial State		It represents initial state of object.
11	Final State		It represents final state of object.
12	Control Flow		It represents decision making process for object.
13	Decision Box		It represents the decision making process from a constraint.

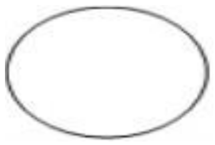
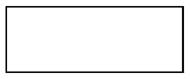
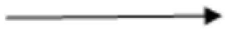
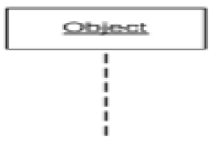
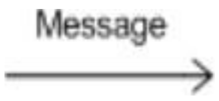
14	Data Process/ State		A circle in a DFD represents a state or process which has been triggered due to some other event or action.
15	External Entity		It represents external entity such as Keyboard, sensors, etc which are used in the system.
16	Transition		It represents any communication that occurs between processes.
17	Object Lifeline		Object lifeline represents the vertical dimension that object communicates.
18	Message		It represents messages exchanged.

Table 4.1: Symbols used in UML

6.3 Data Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

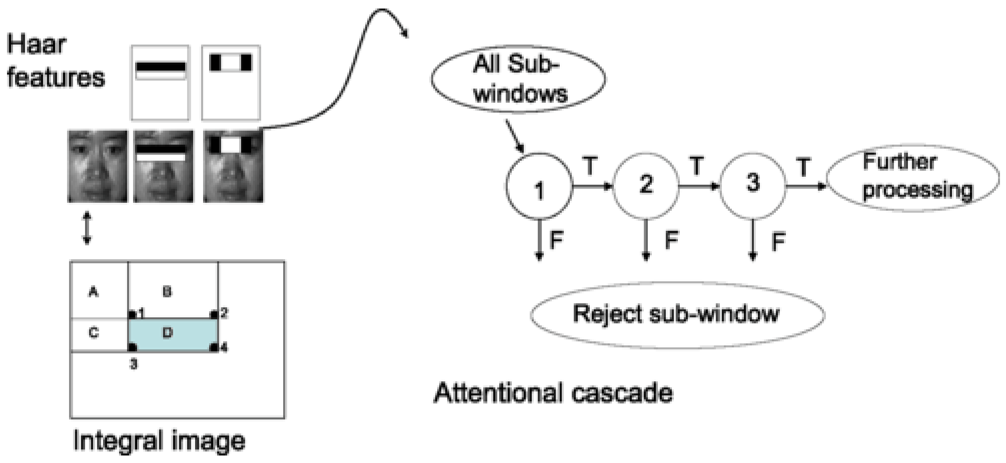


Fig 4.2 Data Flow Diagram

The data flow diagram essentially shows how the data control flows from one module to another. Unless the input filenames are correctly given the program cannot proceed to the next module. Once the correct input filenames are given by the user parsing is done individually for each file. The required information is taken in parsing and an adjacency matrix is generated for that. From the adjacency matrix, a lookup table is generated giving paths for blocks. And the final sequence is computed with the lookup table and the final required code is generated in an output file. In case of multiple file inputs, the code for each is generated and combined together.

6.4 COMPONENT DIAGRAM

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

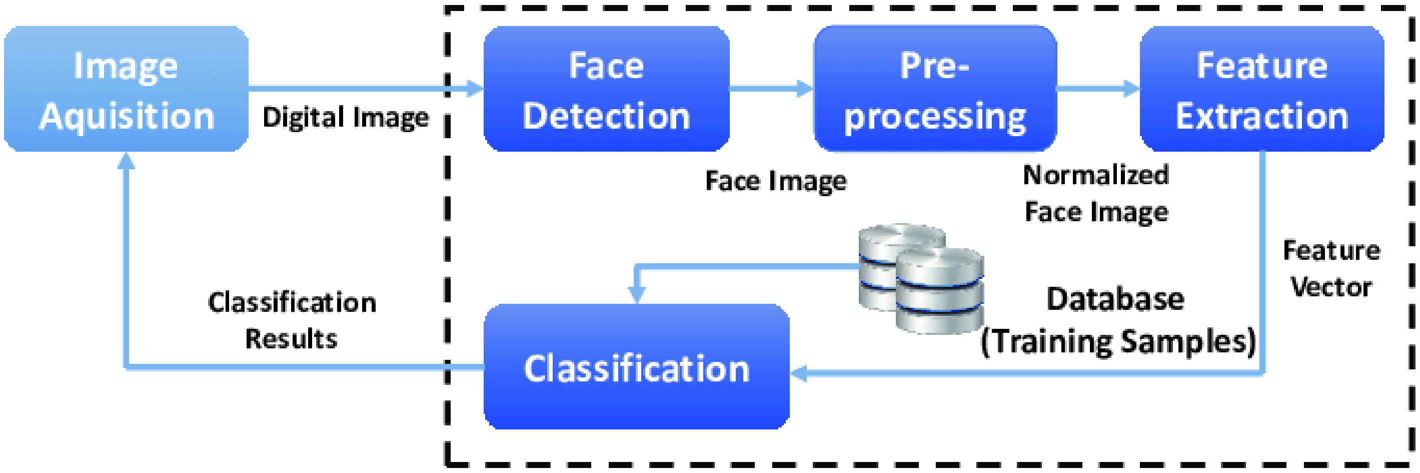


Fig 4.3 Component Diagram

The component diagram for the decentralized system ideally consists of different modules that are represented together via a common module for the user. The user is required to have the input files in the current folder where the application is being used.

It is interesting to note that all the sequence of activities that are taking place are via this module itself, i.e. the parsing and the process of computing the final sequence. The parsing redirects across the other modules till the final code is generated.

6.5 Use case Diagram:

A use case defines a goal-oriented set of interactions between external entities and the system under consideration. The external entities which interact with the system are its actors. A set of use cases describe the complete functionality of the system at a particular level of detail and it can be graphically denoted by the use case diagram.

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

In software and systems engineering, a use case is a list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human, an external system, or time.

In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured in Systems Modeling Language (SysML) or as contractual statements.

The Seqeunce of activities that are carried out are the same as the other diagrams .Use case for this module indicates the users interaction with the system as a whole rather than individual modules .All the encryption mechanisms are carried out via the login page that redirects the user to the particular functionality that he or she wishes to implement .

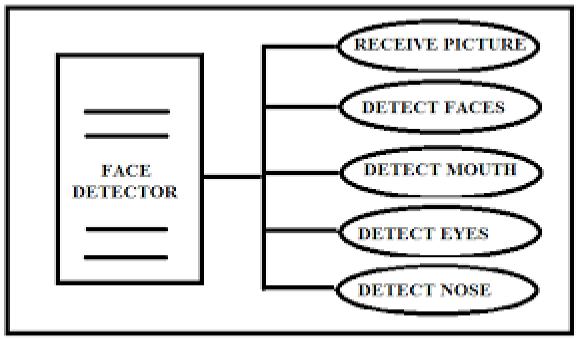


Fig 4.4 Use Case Diagram

6.6 ACTIVITY DIAGRAM

An activity diagram shows the sequence of steps that make up a complex process. An activity is shown as a round box containing the name of the operation. An outgoing solid arrow attached to the end of the activity symbol indicates a transition triggered by the completion.

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organisational processes (i.e. workflows). Activity diagrams show the overall flow of control.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- rounded rectangles represent actions;
- diamonds represent decisions;
- bars represent the start (split) or end (join) of concurrent activities;
- a black circle represents the start (initial state) of the workflow;
- an encircled black circle represents the end (final state).

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

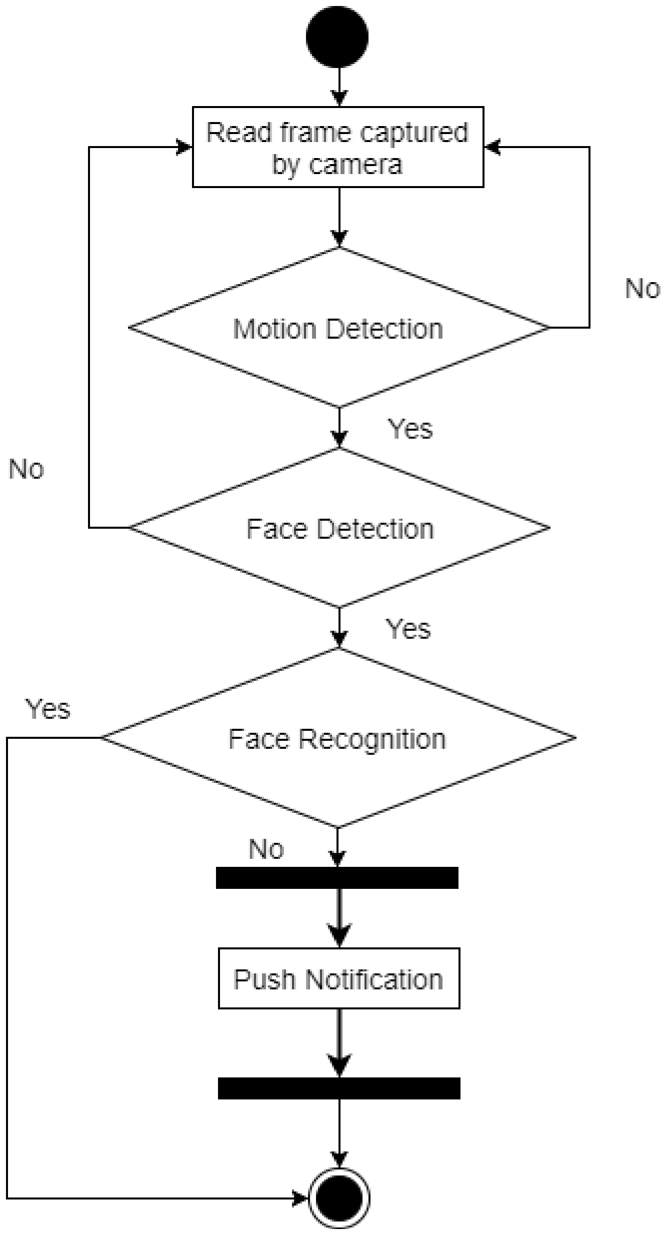


Fig 4.5 Activity Diagram

In the diagram above, the start state is represented before the user inputs file and the end process after the complete final code is generated. The rectangular boxes represent the activities that take place throughout the process and all of these converge to a single end point that marks the end point and also indicates that a successful code generation has taken place.

6.7 Sequence Diagram:

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time, the horizontal dimension represents the objects existence during the interaction.

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

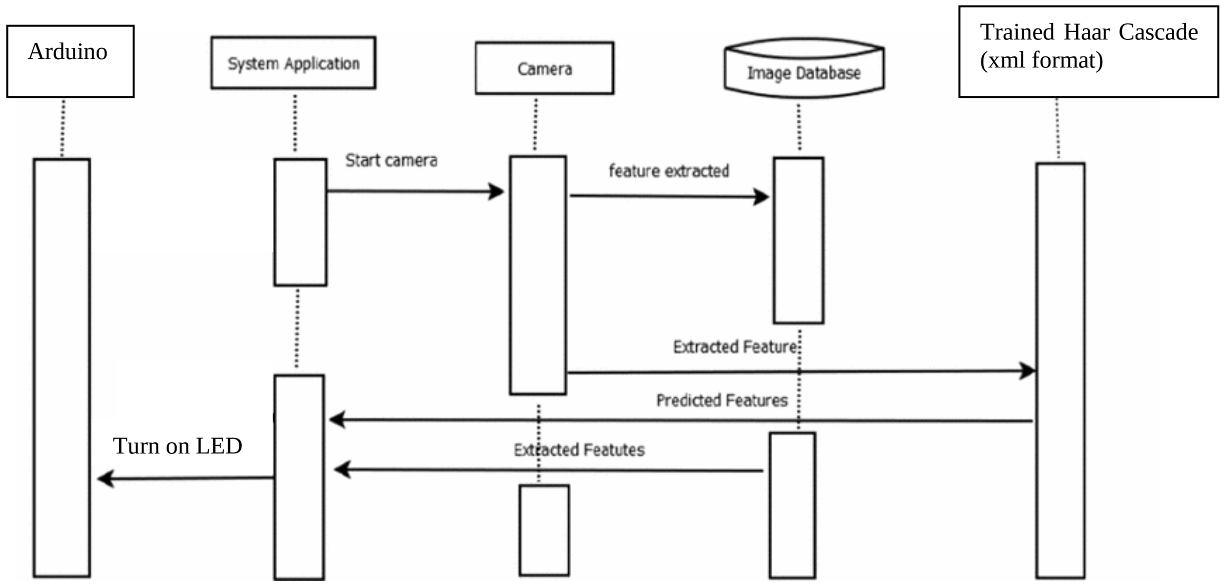


Fig 4.6 Sequence Diagram

The sequence diagram shows the set of events that occur once the input file has been given. The first step is to parse, where the required data will be put into a structure. Then the adjacency matrix is created which will be used when the lookup table is created and also while computing the final sequence. Once the final sequence is generated, the final code is generated which also uses the structure that was created during parsing. The user will get the generated code for the file that he has inputted.

Chapter 7

IMPLEMENTATION

Below Diagram explains how Implementation flow works.

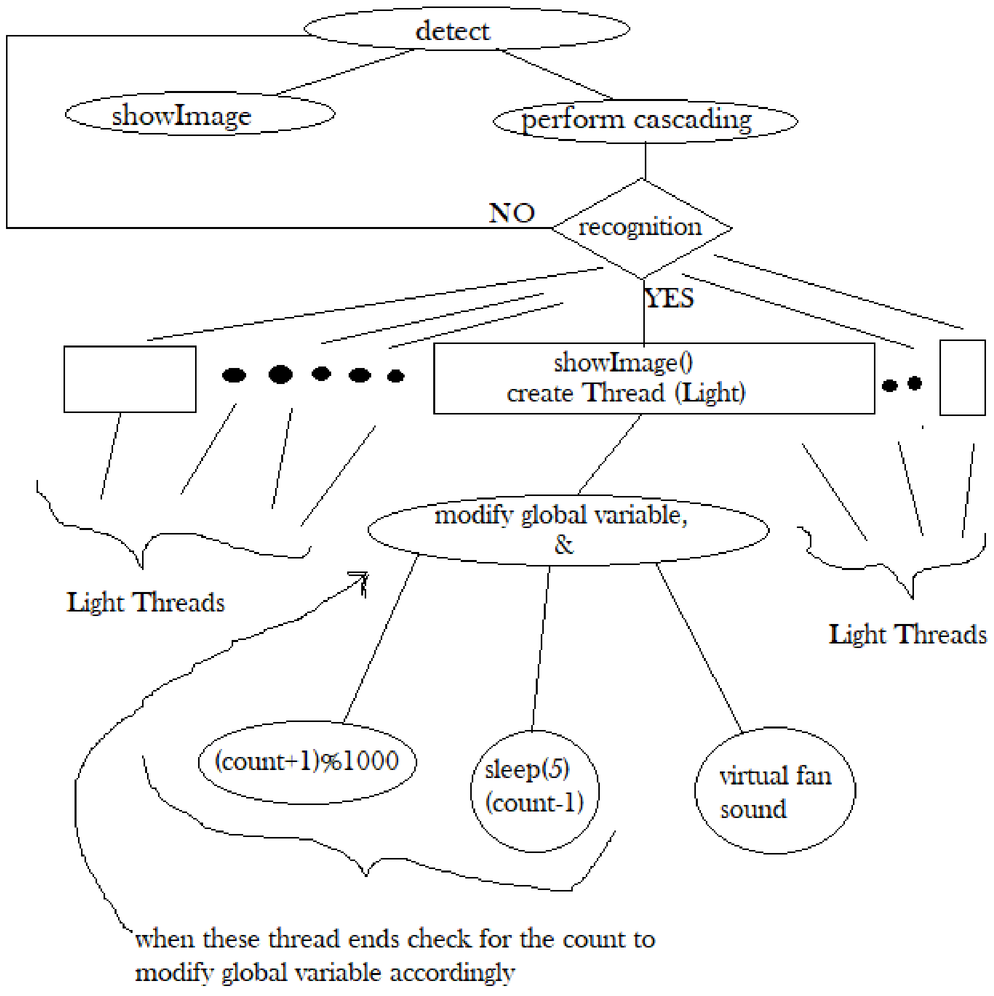


Fig 7.1 Overview of Project Implementation

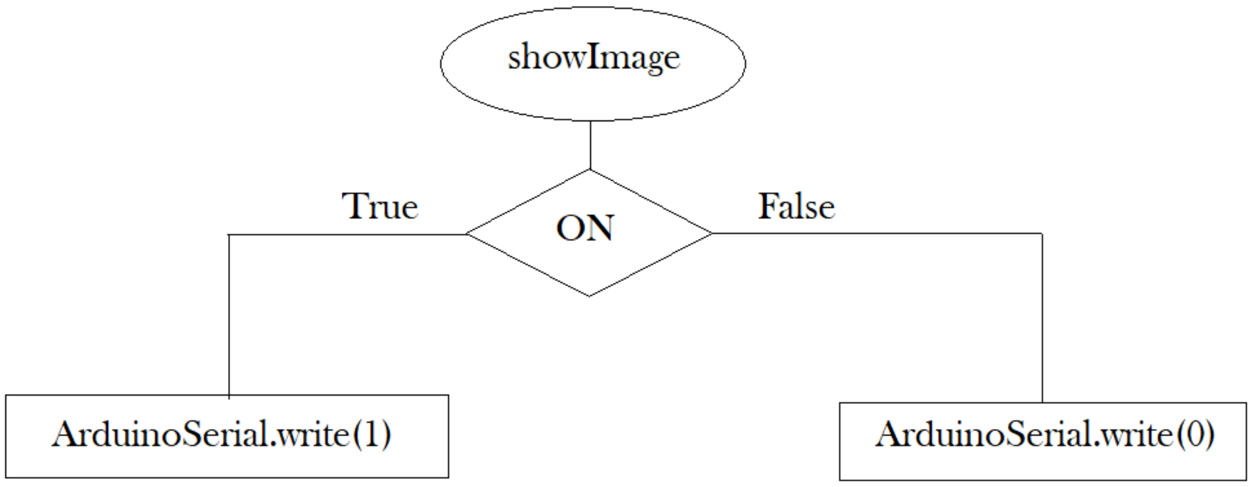


Fig 7.2 Sending data to serial Port

There are 4 major stages in the working of the project where each stage performs an important aspect of the project. The 4 stages are

7.1 Arduino Implementation

Here we will be controlling the LED through Arduino board using Python script.

Inside the setup function we initialize the serial communication at 9600 baud rate and declare that we will be using the built in led as output and turn it low during program start.

```
void setup() {
        Serial.begin(9600); //initialize serial COM at 9600 baudrate
        pinMode(LED_BUILTIN, OUTPUT); //make the LED pin (13) as output
        digitalWrite (LED_BUILTIN, LOW);
 }
```

Inside the *loop* function, we read whatever the data that is coming in serially and assigning the value to the variable "data". Now based on the value of this variable ("data") we toggle the built in led as shown below.

```
void loop() {
        while (Serial.available()){
        data = Serial.read();
        }
        if (data == '1')
        digitalWrite (LED_BUILTIN, HIGH);


        else if (data == '0')
        digitalWrite (LED_BUILTIN, LOW);
}
```

7.2 Recognition

This step is for image recognition, using OpenCv

```
def detect(gray, frame):

  writeOnScreen()
  showImage()

  # for front face
  faces = face_cascade.detectMultiScale(gray, 1.3, 5)
  for (x, y, w, h) in faces:
    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
    t1 = threading.Thread(target=light)
    t1.start()
    writeOnScreen()
    showImage()

# front upper body
upper_body = upper_body_cascade.detectMultiScale(gray, 1.7, 1)
for(upx, upy, upw, uph) in upper_body:
    cv2.rectangle(frame, (upx, upy), (upx+upw, upy+uph), (255, 0, 0), 2)
    t1 = threading.Thread(target=light)



     t1.start()
    writeOnScreen()
    showImage()

# front full body
body = full_body_cascade.detectMultiScale(gray, 1.7, 1)
for(bx, by, bw, bh) in body:
    cv2.rectangle(gray, (bx, by), (bx+bw, by+bh), (255, 0, 0), 2)
    t1 = threading.Thread(target=light)
    t1.start()
    writeOnScreen()
    showImage()

# back upper body
    upper_back = upper_back_part.detectMultiScale(gray, 1.7, 1)
```

```
        for(bx, by, bw, bh) in upper_back:
                cv2.rectangle(gray, (bx, by), (bx+bw, by+bh), (255, 0, 0), 2)
                t1 = threading.Thread(target=light)
                t1.start()
                writeOnScreen()
                showImage()

    return frame
```

7.3 Multi-Threading

As per requirement for every recognition, a thread is create for pre-processing.

```
def light():
    global s
    global r
    global g
    global b
    s = "Lights ON"
    r = 0
    g = 255
    b = 0

    global curr_status
    curr_status = 1

    lock = threading.Lock()

    t1 = threading.Thread(target=add, args=(lock, ));
    t2 = threading.Thread(target=wait, args=(lock, ));
    t1.start()
    t2.start()
    f = threading.Thread(target=startFan(), args=());
    f.start()

    t1.join()
    t2.join()

    if count == 0:
        s = "Lights OFF"
        r = 255
        g = 0
        b = 0

curr_status = 0
```

7.4 Instruct Arduino

Connection of python program with Arduino serial port and sending information as per recognition.

```
import serial #Serial imported for Serial communication

import time #Required to use delay functions

ArduinoSerial = serial.Serial('com3',9600)

#Create Serial port object called arduinoSerialData

time.sleep(2) #wait for 2 secounds for the communication to get established
```

```
def showImage():
    global on
    global curr_status
    if on != curr_status:
        on = curr_status
        bulb = cv2.imread(bulb_image[on], 1)
        cv2.imshow('image', bulb)
        if on == 1:
            ArduinoSerial.write('1') #send 1
            print ("LED turned ON")
            time.sleep(1)
        else:
            ArduinoSerial.write('0') #send 0
            print ("LED turned OFF")
            time.sleep(1)
```

Chapter 8

TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. In the testing stage following goals are tried to achieve:-

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as a solution to the original problem.
- To provide operational reliability of the system.

8.1 Testing Methodologies

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

8.1.1 White box testing

White box testing (clear box testing, glass box testing, and transparent box testing or structural testing) uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. While white box testing is applicable at the unit, integration and system levels of the software testing process, it is typically applied to the unit. While it normally tests paths within a unit, it can also test paths between units during integration, and between subsystems during a system level test.

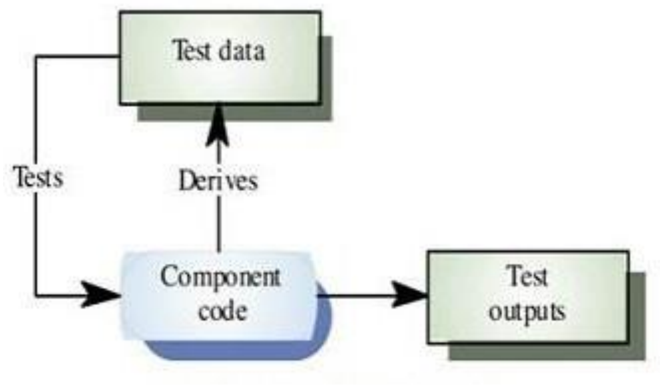


Fig 8.1 White Box Testing

Though this method of test design can uncover an overwhelming number of test cases, it might not detect unimplemented parts of the specification or missing requirements, but one can be sure that all paths through the test object are executed. Using white box testing we can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structure to assure their validity

8.1.1.1 Advantages of White Box Testing

a. To start the white box testing of the desired application there is no need to wait for user face (UI) to be completed. It covers all possible paths of code which will ensure a thorough testing.

b. It helps in checking coding standards.

c. Tester can ask about implementation of each section, so it might be possible to remove unused/deadlines of codes helps in reducing the number of test cases to be executed during the black box testing.

d. As the tester is aware of internal coding structure, then it is helpful to derive which type of input data is needed to test the software application effectively.

e. White box testing allows you to help in code optimization

8.1.1.2 Disadvantages of White Box Testing

a) To test the software application a highly skilled resource is required to carry out testing who has good knowledge of internal structure of the code which will increase the cost.

b) Updating the test script is required if there is change in requirement too frequently.

c) If the application to be tested is large in size, then exhaustive testing is impossible.

d) It is not possible for testing each and every path/condition of software program, which might miss the defects in code.

e) White box testing is a very expensive type of testing.

f) To test each paths or conditions may require different input conditions, so in order to test full application, the tester need to create range of inputs which may be a time consuming.

8.1.2 Black box testing

Black box testing focuses on the functional requirements of the software. It is also known as functional testing. It is a software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs.

The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications. It enables us to derive sets of inputs that will fully exercise all functional requirements for a program.

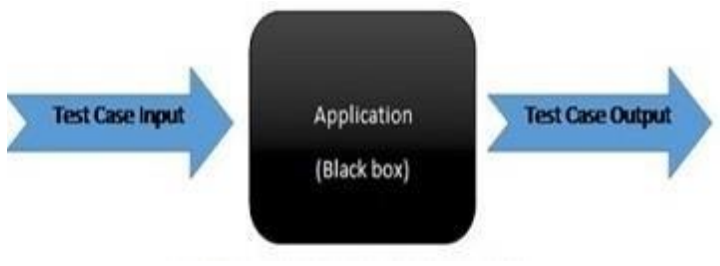


Fig 8.2 Black Box Testing

Black box testing is an alternative to white box technique. Rather it is a complementary approach that is likely to uncover a different class of errors in the following categories:-

- Incorrect or missing function.
- Interface errors.
- Performance errors.
- Initialization and termination errors.
- Errors in objects.

8.1.2.1 Advantages of Black Box Testing

- The test is unbiased as the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.
- Test cases can be designed as soon as the specifications are complete.

8.1.2.2 Disadvantages of Black Box Testing

- The test inputs need to be from large sample space. That is, from a huge set of data this will take time.

- Also it is difficult to identify all possible inputs in limited testing time. So writing test cases is slow and difficult.
- Chances are more that there will be unidentified paths during this testing.

8.2 Unit Testing

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

8.3 System Testing

This information contributes towards reducing the ambiguity about the system. For example, when deciding whether to release a product, the decision makers would need to know the state of the product including aspects such as the conformance of the product to requirements, the usability of the product, any known risks, the product’s compliance to any applicable regulations,

Software testing enables making objective assessments regarding the degree of conformance of the system to stated requirements and specifications.

System testing checks complete end-end scenarios, as a user would exercise the system. The system has to be tested for correctness of the functionality by setting it up in a controlled environment. System testing includes testing of functional and nonfunctional requirements. It helps to verify and validate the system. All components of system should have been successfully unit tested and then checked for any errors after integration.

8.4 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confident that the product quality is meeting its goals. This is an “umbrella activity” that is applied throughout the engineering process**.** Software quality assurance encompasses:-

- Analysis, design, coding and testing methods and tools
- Formal technical reviews that are applied during each software engineering
- Multi-tiered testing strategy
- Control of software documentation and the change made to it.
- A procedure to ensure compliance with software development standards.
- Measurement and reporting mechanisms.

8.4.1 Quality Factors

An important objective of quality assurance is to track the software quality and assess the impact of methodological and procedural changes on improved software quality. The factors that affect the quality can be categorized into two broad groups:

- Factors that can be directly measured.
- Factors that can be indirectly measured

These factors focus on three important aspects of a software product

- Its operational characteristics
- Its ability to undergo changes
- Its adaptability to a new environment.

- Effectiveness or efficiency in performing its mission
- Duration of its use by its customer.

8.5 Functional test

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

Functional testing is centered on the following items:

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

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

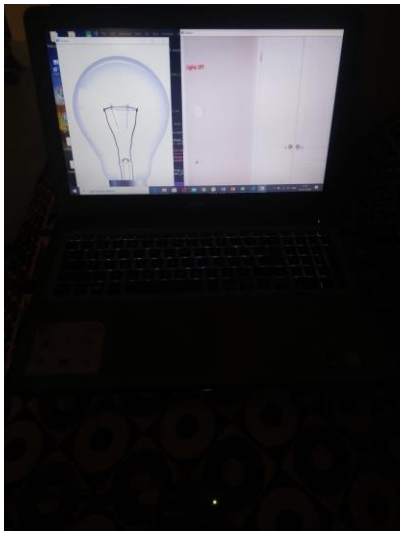
Chapter 9

RESULT AND PERFORMANCE ANALYSIS

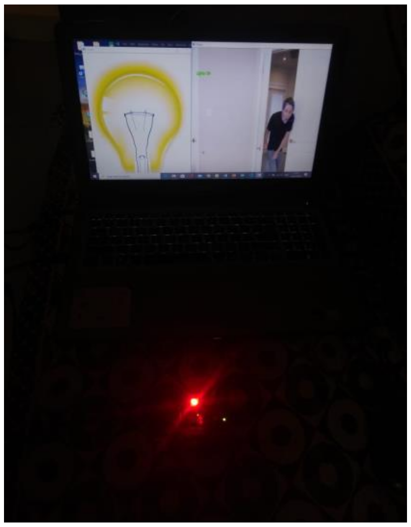
SNAPSHOTS OF OUTPUT

In this section snapshots showing the performance of the final code that is generated is shown.

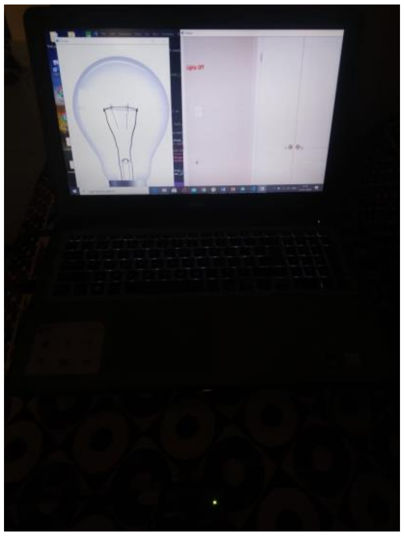
1) **When there is no–one in room:** lights are off.



2) **Person enter inside room:** light turns on automatically.



3) **Person went outside room:** light turns off automatically.



Summary

This chapter gives a graphic view of the execution of the system. The output screens show how the recognition and further hardware implementation work. Based on the analysis the proposed protocol performs well in terms of accuracy, dependency and usability.

Chapter 10

CONCLUSION AND FUTURE SCOPE

10.1 FUTURE SCOPE

- Many highly advance algorithm can be used to make system even more accurate.
- Same prototype can be used for robbery detection and calling police for help.
- More hardware can be attached, based on the requirements.

10.2 CONCLUSION

In conclusion, the project termed “Optimizing Power Consumption using Artificial Intelligence”, was successfully completed in 3 months.More Advance Algorithm will be a better option for this model to be more accurate. We plan on taking this project forward and deploying it in many more organizations and research institutions working in this field in the near future. I am hopeful to make a difference in Saving electricity using AI, even if it’s in a small way.

REFERENCES

[1] https://www.udemy.com/

[2] https://www.superdatascience.com/pages/computer-vision

[3] https://www.researchgate.net/publication/3766402_General_framework_for_object_detection

[4] https://en.wikipedia.org/wiki/Viola-Jones_object_detection_framework

TABLE OF CONTENTS

Title **Page No.**

Abstract………………………………………………………………………………. **i**

Acknowledgement……………………………………………………………………… **ii**

List of Figures………………………………………………………………………... **vii**

List of Tables………………………………………………………………………… **viii**

Chapter 1 **1 – 4**

PREAMBLE………………………………………………………………….…..... **1**

1.1 Introduction……………………………………………………………….. 1

1.2 Existing system……………………………………………………………… 2

1.2.1 Drawbacks………………………………………………………………. 4

1.3 Proposed System………………………………………………………………. 4

1.4 Plan of Implementation………………………………………………………. 4

Chapter 2 **5-5**

LITERATURE SURVEY…………………………………………………………….. **5**

Chapter 3 **6-13**

THEORETICAL BACKGROUND………………………………………………….. **6**

3.1 Image Processing……………………………………………………………… 6

3.2 Viola-Jones Object detection framework………………………………………. 6

3.3 Problem description…………………………………………………………….. 7

3.4 Components of the framework………………………………………………….. 7

3.4.1 Feature types and evaluation………………………………………………. 7

3.4.1.1 Haar Features…………………………………………………………….. 8

3.4.1.2 Summed area table……………………………………………………….. 10

3.4.2 Learning Algorithm……………………………………………………….... 10

3.4.3 Cascade architecture………………………………………………………….. 11

3.5 Using Viola-Jones for object tracking…………………………………………….. 13

Chapter 4 **14-17**

SYSTEM REQUIREMENTS SPECIFICATION………………………………... **14**

4.1 Functional Requirement...………….…………………………………………….. 15

4.2 Non Functional Requirements....…………………………………………………. 15

4.2.1 Product Requirements………………………………………….......... 16

4.2.2 Organizational Requirements…………………………………….……... 17

4.2.3 User Requirements…………………………………………….………... 17

4.2.4 System Configuration…………………………………………………… 17

Chapter 5 **19-21**

SYSTEM ANALYSIS ……………………………………………………………… **19**

5.1 Feasibility Study…………………………………………………………………. 19

5.1.1 Economical Feasibility………………………………………………… 20

5.1.2 Technical Feasibility……………………………………………....... 20

5.1.3 Social Feasibility…………………………………………………………. 20

5.2 Analysis ……………………………………………………………………..... 21

5.2.1 Performance analysis ………………………………………………… 21

5.2.2 Technical analysis …………………………………………………….. 21

5.2.3 Economical analysis …………………………………………………... 21

Chapter 6 …………………………………………………………………. **22-33**

SYSTEM DESIGN……………………………………………………………………. **22**

6.1 System development methodology.……………………………………………… 22

6.1.1 Model phases.………………………………………………………….. 22

6.1.2 Advantages of Waterfall Model.……………………………………….. 23

6.2 Design using UML.……………………………………………………………… 24

6.3 Data Flow Diagram.…………………………………………………………... 28

6.4 Component Diagram………………………………………………………….. 29

6.5 Use Case Diagram…………………………………………………………….. 29

6.6 Activity Diagram……………………………………………………………....... 31

6.7 Sequence Diagram.…………………………………………………………… 33

Chapter 7…………………………………………………………………. 35-40

IMPLEMENTATION…………………………………………………………… 35

7.1 Arduino Implementation………………………………………………………. 36

7.2 Recognition………………………………………………………………........ 37

7.3 Multi-Threading………………………………………………………………… 39

7.4 Instruct Arduino………………………………………………………………… 40

Chapter 8…………………………………………………………………….... 42-48

TESTING………………………………………………………………………. 42

8.1 Testing Methodologies………………………………………………………… 42

8.1.1 White Box Testing………………………………………………………… 42

8.1.1.1 Advantages of White Box Testing…………………………………. 43

8.1.1.2 Disadvantages of White Box Testing……………………………… 44

8.1.2 Black Box Testing………………………………………………………… 44

8.1.2.1 Advantages of Black Box Testing…………………………………. 45

8.1.2.2 Disadvantages of Black Box Testing………………………………. 45

8.2 Unit Testing……………………………………………………………………. 46

8.3 System Testing………………………………………………………………… 46

8.4 Quality Assurance……………………………………………………………… 47

8.4.1 Quality Factors……………………………………………………………….. 47

8.5 Functional Test………………………………………………………………… 48

Chapter 9………………………………………………………………………………… 49-50

RESULTS AND PERFORMANCE ANALYSIS………………………………….. 49

9.1 Screenshots of output………….. ……………………………………………….. 49

9.2 Summary…………………….. ……………………………………………… 50

Chapter 10…………………………………………………………………………… 51-51

CONCLUSION AND FUTURE SCOPE………………………………………….. 51

10.1 Future Scope………………………………………………………………. 51

10.2 Conclusion………………………………………………………………… 51

LIST OF FIGURES

Figure No.	Title	Page No.
Fig 3.1	Example rectangle features shown relative to the……………… enclosing detection window	8
Fig 3.2	Haar Feature that looks similar to the bridge of the nose……… is applied onto the face	9
Fig 3.3	Haar Feature that looks similar to the eye region which is…….. darker than the upper cheeks is applied onto a face	9
Fig 3.4	3rd and 4th kind of Haar Feature……………………………….	9
Fig 6.1	Waterfall model………...	24
Fig 6.2	Data Flow Diagram……………………………………………	28
Fig 6.3	Component Diagram………………………………………….	29
Fig 6.4	Use Case Diagram…………………………………………….	30
Fig 6.5	Activity Diagram……………………………………………...	32
Fig 6.6	Sequence Diagram…………………………………………….	33
Fig 7.1	Overview of Project Implementation………………………….	35
Fig 7.2	Sending data to Serial Port……………………………………	36
Fig 8.1	White Box Testing…………………………………………….	42
Fig 8.2	Black Box Testing……………………………………………..	44

LIST OF TABLES

Table No.	Title	Page No.
Table 4.1	Symbols used in UML………………………..................	25