**Chapter 1**

**INTRODUCTION**

**1.1 Objective**

This project aims to solve one of the most common problems faced by the University: wastage of electricity in form of unnecessarily running fans and lights. We frequently find that even when no one is in the classroom, the fans and lights in the room are all switched on. A similar scenario can be found in hostels too, where some students often lock the room when they are not present and don’t bother switching off fans and lights. A solution to this ubiquitous problem class room automation.

**1.2 Motivation**

Electrical energy is the most widely used form of energy. Being most widely used, it is also seen wasted frequently. Keeping energy conservation and carbon footprints in mind, we aim to develop a system which can reduce the wastage of electricity in our campus. Thus we investigated and found one of the major sources of this wastage to be unnecessarily switched on appliances, primarily fans and lights. We found numerous cases of this in both classrooms as well as hostel rooms. Thus our aim is to develop a system that can automatically switch off fans and lights in case no one is present in the room. We further extend our project by only switching on fans and lights in the vicinity of a person rather than switching on all of them.

**1.3 Background**

The project aims at developing a system that will use a camera to identify the presence as well as the position of a person sitting in a classroom. The system will be based on a Raspberry Pi which will be coded in Python, will use OpenCV for image processing and a NoIR camera for taking images.

A face detection algorithm will be used to detect faces of the people present in the classroom and a distance measurement algorithm will be used to measure the distance between the camera and the person.

The distance of the fans and lights from the camera will be pre-coded into the system. Since most of the classrooms (in a particular building) have the same arrangement of fans and lights, pre measured distances can directly be loaded into the system. Using the above two pieces of data, the system will calculate the fans and lights nearest to the person and switch it on. It will also switch the lights and fans off if there is no one in the room.The above process will be repeated for each person in case several people are there in the room.

**Flowchart**

**Fig1.1**

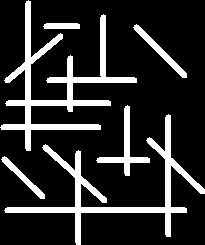
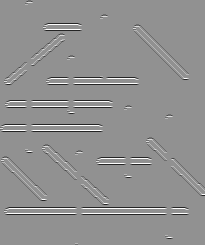
**Detection algorithms**

There are many types of algorithms used to detect the face

We are going to discuss about Viola jones face detection algorithm:

* Haar features
* Integral image
* Adaboost
* Cascading

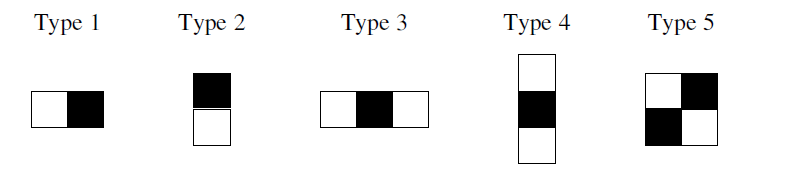
**Haar features:**

**Fig1.2**

The output image (right) has high intensity at pixels where the convolution kernel pixel pattern matches perfectly with the input image

Haar features are similar to these convolution kernels which are used to detect the presence of that feature in the given image



**Fig1.3**

Each feature results in a single value which is calculated by subtracting sum of pixels under white rectangular from pixels under black rectangle

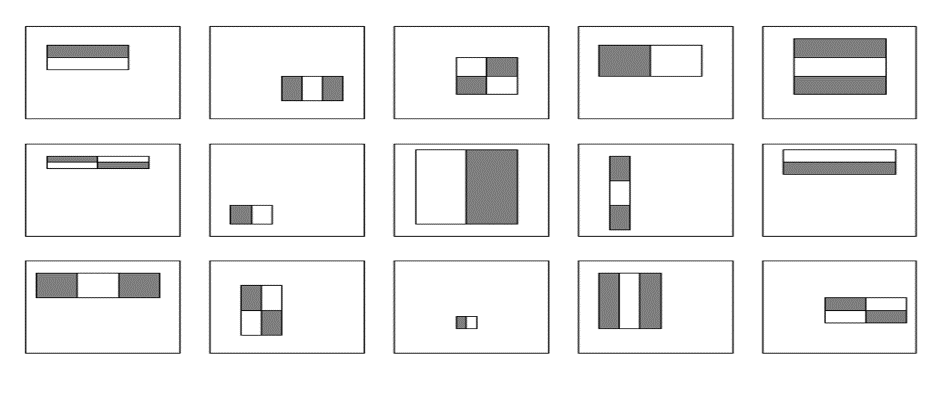
**Integral image:**

Since it is clear that huge number of these rectangular haar features have to be evaluated each time Viola Jones have come up with a neat technique to reduce the computation rather than summing up all pixel values under the black and white rectangles every time.

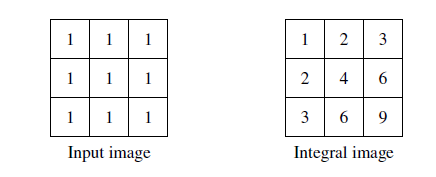
They have introduced the concept of integral image to find the sum of all pixels under a rectangle with just 4 corner values of the integral image.

Viola jones algorithm uses a 24x24 window as the base window size to start evaluating these features in any given image.

If we consider all possible parameters of the haar features like position, scale and type we end up calculating about 160,000+ features in this window

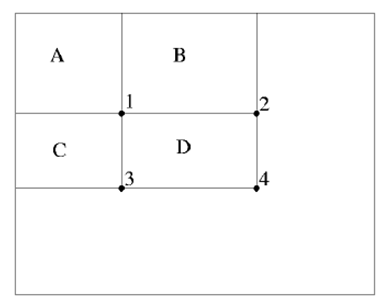


In an integral image the value at pixel (x, y) is the sum of pixels above and to the left of (x, y)



**Fig1.4**

Integral image allows for the calculation of sum of all pixels inside any given rectangle using only four values at the corners of the rectangle.



**Fig 1.5**

Sum of all the pixels

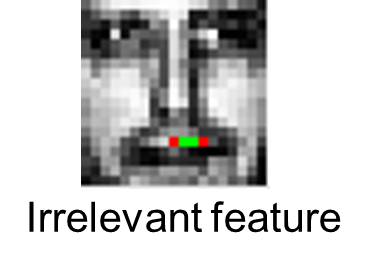
D= 1+4-(2+3)

=A+ (A+B+C+D)-(A+C+A+B) =D

**Adaboost:**

As stated previously there can be approximately 160,000 + feature values within a detector at 24x24 base resolution which need to be calculated. But it is to be understood that only few set of features will be useful among all these features to identify a face.

Adaboost is a machine learning algorithm which helps in finding only the best features among all these 160,000+ features. After these features are found a weighted combination of all these features in used in evaluating and deciding any given window has a face or not. Each of the selected features are considered okay to beincluded if they can at least perform better than random guessing (detects more than half the cases).



**Fig1.6**

These features are also called as weak classifiers. Adaboost constructs a strong classifier as a linear combination of these weak classifiers.

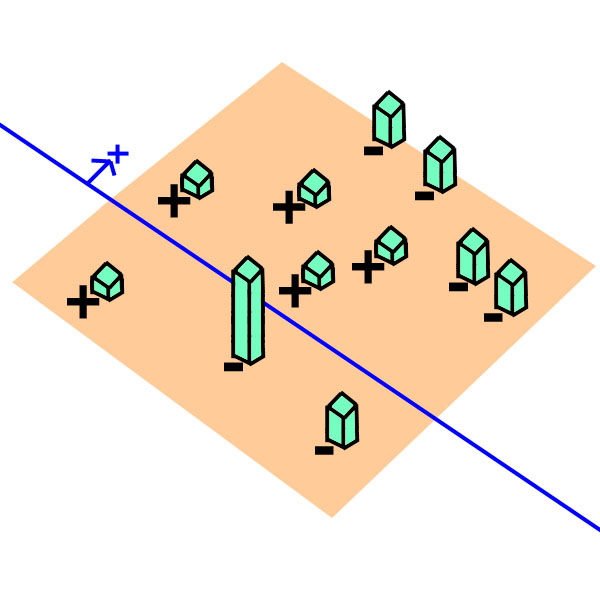
Strong classifier weak classifier

AdaBoost starts with a uniform distribution of “weights” over training examples.

Select the classifier with the lowest weighted error (i.e. a “weak” classifier)

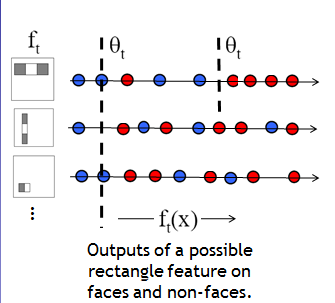
Increase the weights on the training examples that were misclassified.

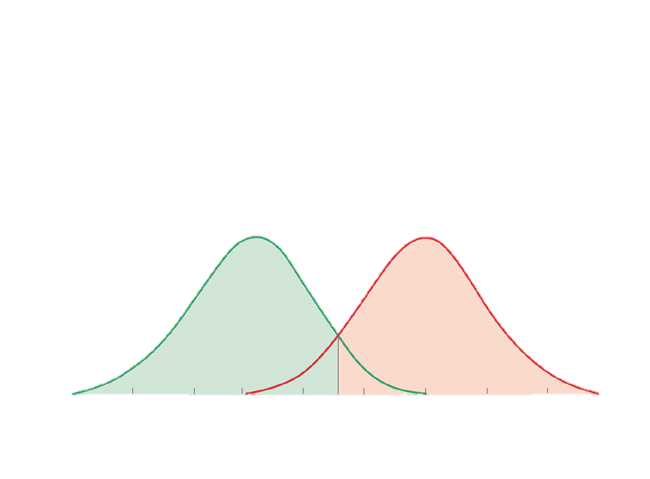




**Fig 1.7**

Adaboost finds the single rectangular feature and threshold that best separates the positive (faces) and negative (non faces) training examples, in terms of weighted error.

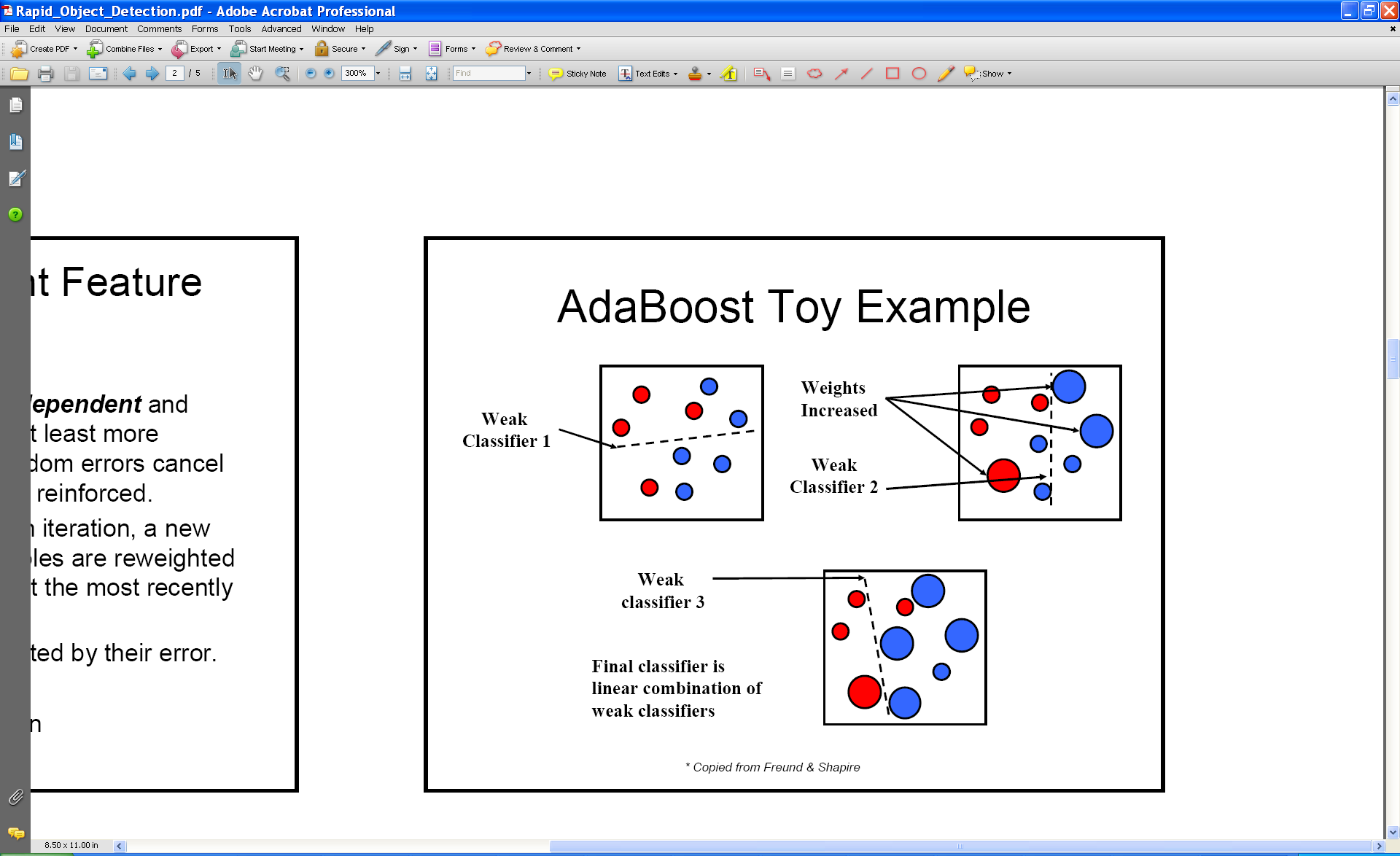




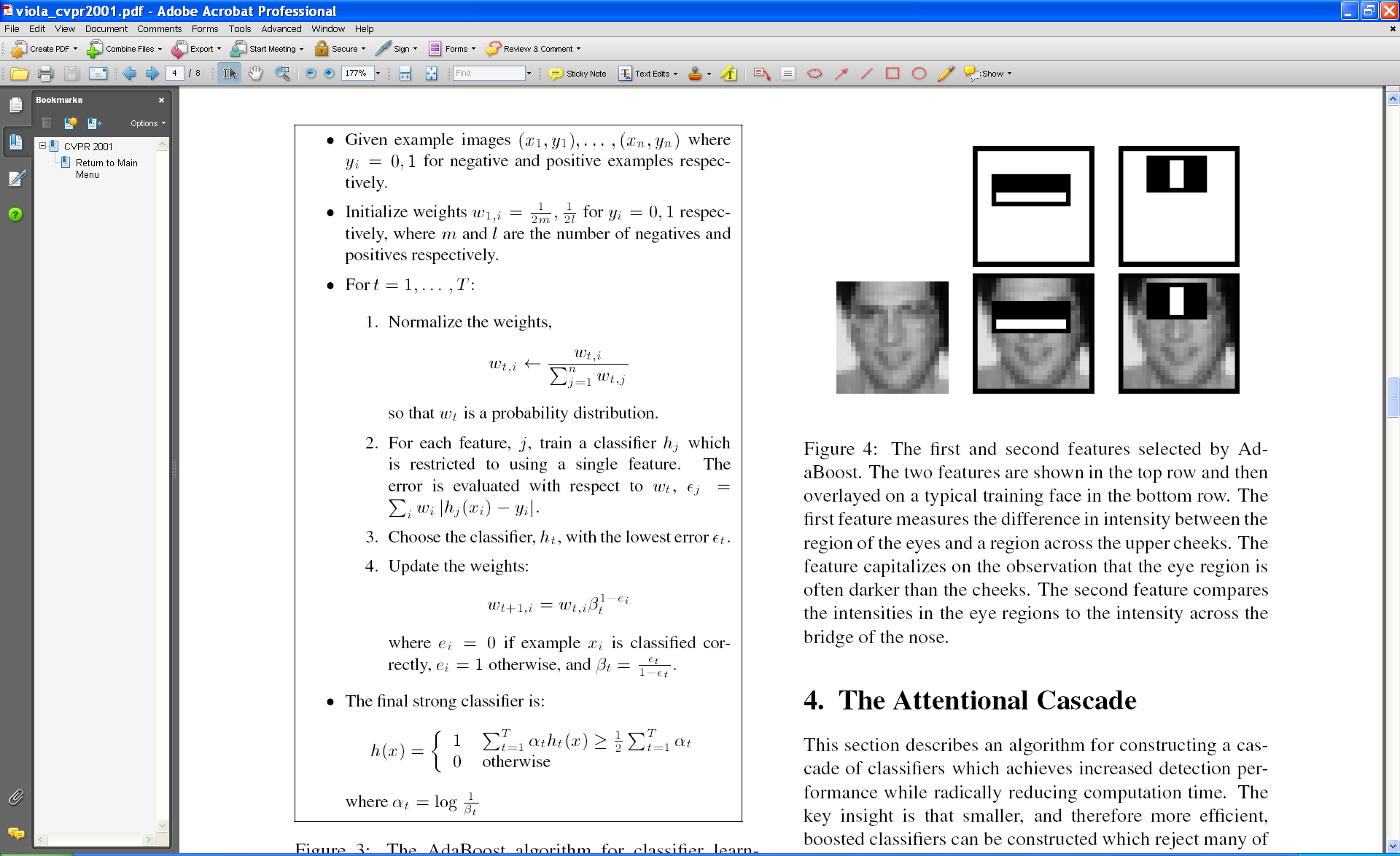
A gaussian weak classifier

**Fig1.8**

Start with uniform weights on training examples



For T rounds:

* Evaluate weighted error for each feature, pick best
* Re-weight the examples: Incorrectly classified -> more weight Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

**Cascading:**

The basic principle of the Viola-Jones face detection algorithm is to scan the detector many times through the same image – each time with a new size.

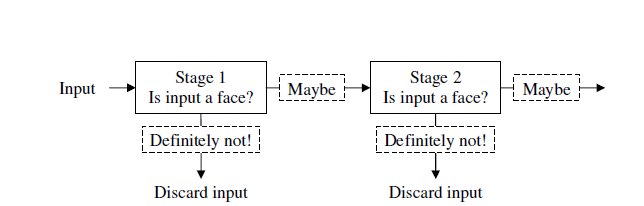
Even if an image should contain one or more faces it is obvious that an excessive large amount of the evaluated sub-windows would still be negatives (non-faces).

So the algorithm should concentrate on discarding non-faces quickly and spend more on time on probable face regions.

Hence a single strong classifier formed out of linear combination of all best features is not a good to evaluate on each window because of computation cost.

Therefore, a cascade classifier is used which is composed of stages each containing 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 used 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 stage.



**Fig 1.9**

**Training cascade:**

To design a cascade, we must choose:

* Number of stages in cascade (strong classifiers).
* Number of features of each strong classifier.
* Threshold of each strong classifier (the  in definition)

Strong classifier definition:  **,**

where**,** 

Since finding optimum combination is extremely difficult. Viola & Jones suggested a heuristic algorithm for the cascade training.

**AdaBoost:**

* User selects values for *f*, the maximum acceptable false positive rate per layer and *d*, the minimum acceptable detection rate per layer.
* User selects target overall false positive rate *Ftarget*.
* *P* = set of positive examples
* *N* = set of negative examples
* *F*0 = 1.0; *D*0 = 1.0; *i* = 0

While *Fi* > *Ftarget*

*i*++

*ni* = 0; *Fi*= *Fi-1*

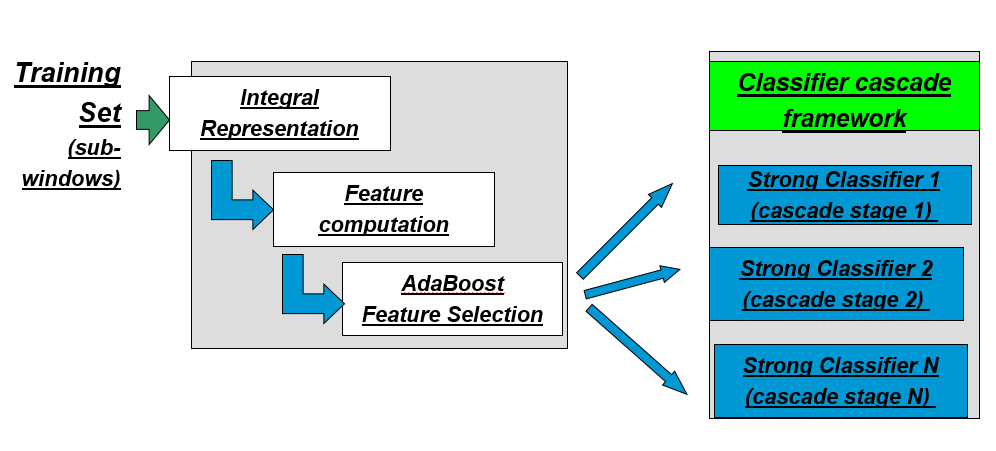
while *Fi* > *f* x *Fi-1*

* + *ni*++
  + Use *P* and *N* to train a classifier with *ni*features using AdaBoost
  + Evaluate current cascaded classifier on validation set to determine *Fi*and *Di*
  + Decrease threshold for the ith classifier until the current cascaded classifier has

a detection rate of at least *d* x *Di-1* (this also affects *Fi*)

*N* = ∅

If *Fi* > *Ftarget* then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set *N.*



**Fig1.10**

Manual Tweaking:

* Select fi (Maximum Acceptable False Positive rate / stage)
* Select di (Minimum Acceptable True Positive rate / stage)
* Select Target (Target Overall False Positive rate)
* Until Target is met:

Add new stage:

Until fi, di rates are met for this stage:

Keep adding features & train new strong classifier

**Chapter 2**

**PROJECT DESCRIPTION AND GOALS**

This project is divided into three stages with the goals set as follows:

**Stage 1:** Testing code on a PC

* Installing Python, OpenCV and other libraries required for the project.
* Developing the code.
* Testing the code on PC.

**Stage 2:** Configuring and connecting the Raspberry Pi to PC

* Installing Raspbian on Raspberry Pi and configuring Xming and Putty on PC and modifying Raspberry Pi’s configuration files to connect it to PC.
* Installing Python, OpenCV and other required libraries on Raspberry Pi and integrating them with Python.

**Stage 3:** Testing code on Raspberry Pi

* Connecting and configuring the camera
* Modify the code to work with Raspberry Pi’s environment.
* Test the code on Raspberry Pi

The project aims at developing a system to automatically detect the presence and location of people inside a classroom and automatically switch on the fans and lights near to the person/people. It will also automatically switch off all fans and lights when no one is present and switch them all on if there more than a threshold number of people present in the class. By doing this we intend to reduce wastage of electricity in the campus, thus also reducing greenhouse emissions and conserving the already fast-depleting non-renewable energy resources such as fossil fuels.

**Chapter 3**

**LITERATURE REVIEW**

**3.1 OpenCV**

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

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 7 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV’s deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

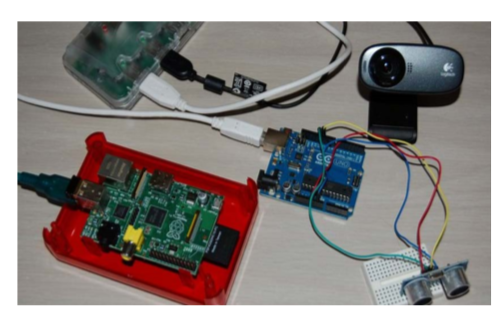
It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, [Android](http://opencv.org/platforms/android.html) and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured [CUDA](http://opencv.org/cuda.html) and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

**3.2 OpenCV Simple Motion Detection**

This project is a program use OpenCV to detect motion and save pictures. Publish by Cédric Verstraeten on website: www.cedricve.me on February 5th, 2013. This program introduction an algorithm use OpenCV to compare different between two images.

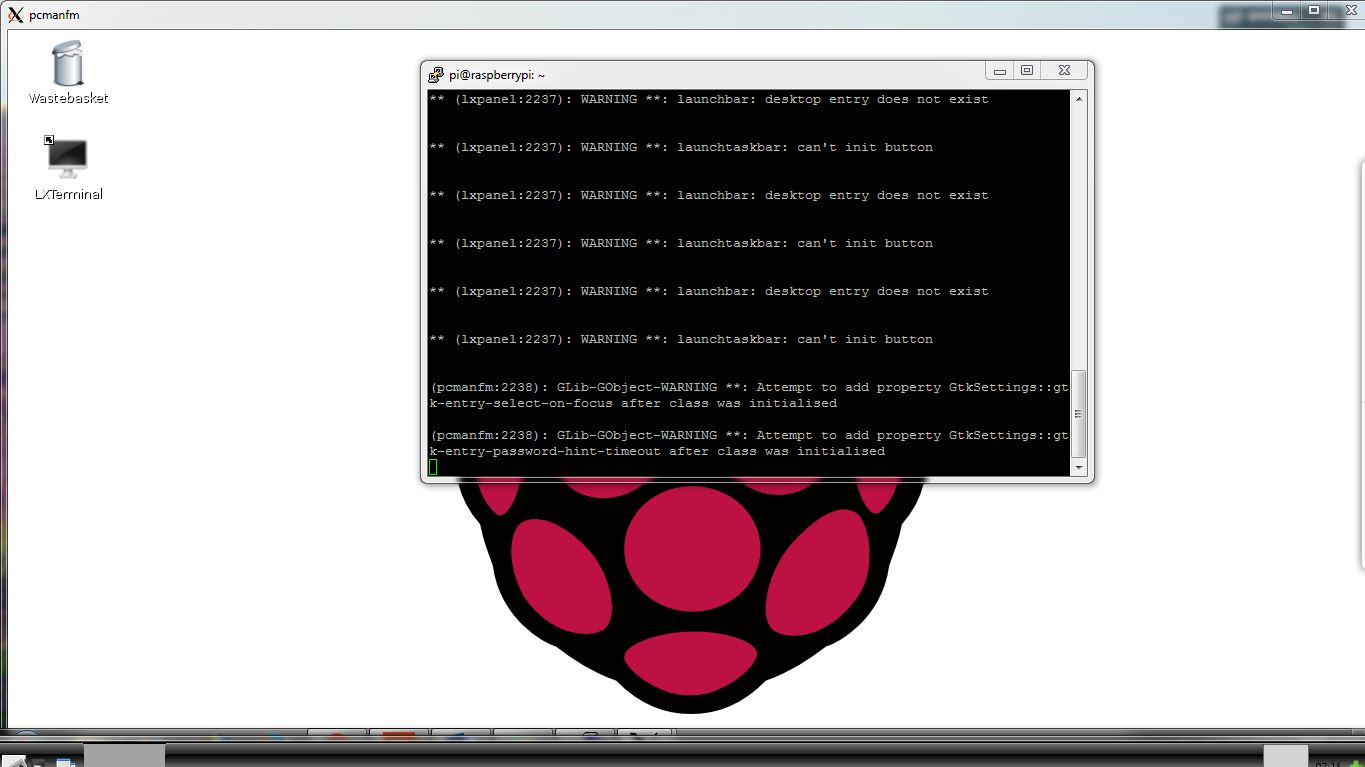
The algorithm is supposed have 2 images, the images are a taken with some delay c between them. If we compare every pixel of the 2 images and they’re all the same, we could say the 2 images are same. But if they don’t, we could say there something happens during the delay time c. Maybe someone place an object in front of the camera or passing.

**3.3 Home Automation with Raspberry Pi**

****

The projects publish by Rupa Dachere on March 16, 2013 in Technology. This project builds to solve a problem where people (gardeners, contractors and neighbor) would come by one’s house when not around.

This project use Arduino to detect motion, use Raspberry Pi and Webcam to capture pictures and send the pictures through e-mail.

**Control Raspberry Pi from Laptop**

This project was posted Raspians.com. This project installs Xming and PuTTY on the Laptop. A slight change in the Pi’s config.txt file is needed (just add the IP address of the network). After enabling X11 forwarding on Putty, users can control the Pi via an Ethernet cable. Putty provides the terminal interface while Xmig gives the GUI in the form of windows on the screen on giving the command “startlxde” via putty terminal.

**3.4 Motion Detection and Object Tracking in Image Sequences**

**3.4.1 Introduction**

Artificial intelligence is an important topic of the current computer science research. In order to be able to act intelligently a machine should be aware of its environment. The visual information is essential for humans. Therefore, among many different possible sensors, the cameras seem very important. Automatically analyzing images and image sequences is the area of research usually called ’computer vision’. This thesis is related to the broad subject of automatic extraction and analysis of useful information about the world from image sequences. The focus in this thesis is on a number of basic operations that are important for many computer vision tasks. These basic steps are analyzed and improvements are proposed.

**3.4.2 Motion Detection and Object Tracking in Image Sequences**

This is a Ph.D. thesis write by Zoran ˇZivkovi´c on June 5 2003. This thesis had introduction algorithm of how to detect motion using image.

The algorithm is a static camera observing a spot is a common case of a monitor system [12, +applicable hypothesis is that the images of the spot without the invade objects exhibit some regular behavior that can be well depict by a statistical model. If we have a statistical model of the spot, an invade object can be detected by spotting the parts of the image that don’t fit the model. This process is usually known as “background subtraction”.

Usually a simple bottom-up way is applied and the spot model has a probability density function for each pixel divided. A pixel from a new image is considered to be a background pixel if its new value is well depicted by its density function. For example, for a static spot the simplest model could be just an image of the spot without the invade objects. The next step would be, for example, to forecast appropriate values for the change of the pixel intensity levels from the image since the change can vary from pixel to pixel. However, pixel values often have complex layout and more elaborate models are needed. In this project, consider two popular models: the parametric Gaussian mixture and the non-parametric k nearest neighbors (k-NN) estimate.

The spot could change from time to time (suddenly or slow illumination changes, static objects deleted etc.). The model should be frequently updated to incarnate the most current situation. The main problem for the background subtraction algorithms is how to automatically and efficiently update the model. This project analyzes the results from the literature and extracts some basic principles. Based on the extracted principles we recommend, analyze and compare two efficient algorithms for the two models: Gaussian mixture and k-NN estimate. The Gaussian mixture density function is a popular flexible probabilistic model. A Gaussian mixture having a fixed number of components is constantly updated using a set of heuristic equations. Based on the results from the previous chapter of this thesis and some additional approximations we propose a set of theoretically supported but still very simple equations for updating the parameters of the Gaussian mixture. The important improvement compared to the previous approaches is that at almost no additional cost also the number of components of the mixture is constantly adapted for each pixel. By choosing the number of components for each pixel in an on-line procedure, the algorithm can automatically fully adapt to the scene. We propose an efficient algorithm based on the more appropriate nonparametric k-NN based model. The both algorithms have similar parameters with a clear meaning and that are easy to set. This project also suggests some typical values for the parameters that work for most of the situations. Finally, we analyze and compare the two proposed algorithms**.**

**3.5 Introduction to OpenCV**

This literature was written by Vadim Pisarevsky, the Senior Software Engineer of Intel Corporation, Software and Solutions Group. This literature introduce general information of OpenCV, Getting Started with OpenCV, Modules Description, interaction with Intel IPP, Python Interface, Some Usage Examples and Summary.

What is OpenCV? OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly direct at real time computer vision, open up by Intel, and now stand by Itseez and Willow Garage. Free for use under the open source BSD license. The library is cross-platform. It is central mainly on real time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself.

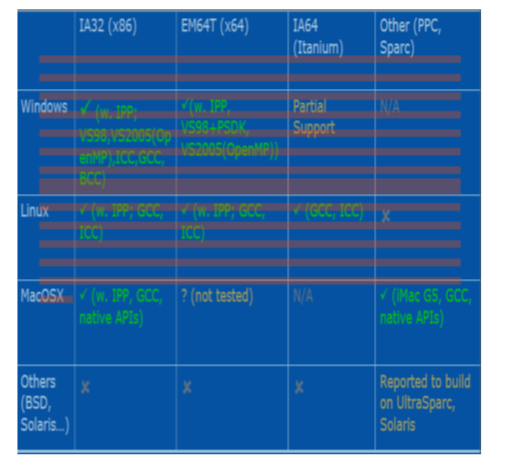
Why we need OpenCV? Now, computer Vision Market is large and still growing. But there is no standard API (like OpenGL and DirectX in graphics, or OpenSSL in cryptography), most of CV software is of 3 kinds:

1. Research code (slow, unstable, independent/incompatible data types for every library/toolbox)

2. Very expensive commercial toolkits (like Halcon, MATLAB+Simulink,)

3. Specialized solutions bundled with hardware (Video surveillance, Manufacturing control systems, Medical equipment …)

So, we need a standard library that simplifies development of new program and solutions much easier. So, Intel develop OpenCV, and Intel optimization for Intel Platforms that creates new usage models by achieving real-time performance for quite “heavy” algorithms and makes Theft problem has troubled people for years, but most motion detector have same problem, such as expensive, large size, hard to install and immovable after installed. To solve this problem, I can design a Motion Detection Monitoring System Use Raspberry Pi and USB webcam for hardware module and writes a program can detection motion and takes pictures and can check by web browser under local network stall and immovable after installed. To Theft problem has troubled people for years, but most motion detector have same problem, such as expensive, large size, hard to I Theft problem has troubled people for years, but most motion detector have same problem, such as expensive, large size, hard to install and immovable after installed.

****

What OpenCV can do? OpenCV library is actively used by a large number of companies and research centers, and versatile.

**3.6 OpenCV Functionality Overview**

1. “Smart” windows

2. Image I/O, rendering

3. Processing keyboard and other events, timeouts

4. Track bars

5. Mouse callbacks

6. Video I/O

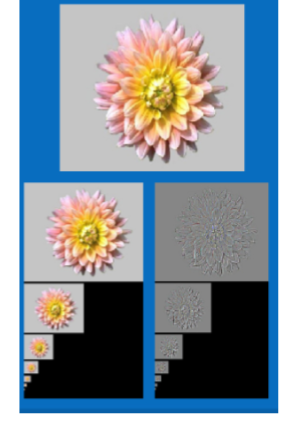
7. Matrix and Image Arithmetical and Logical Operations

8. Linear algebra, DFT/DCT, vector math

9. Dynamic structures

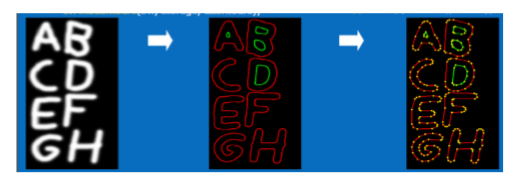
10. Drawing functions

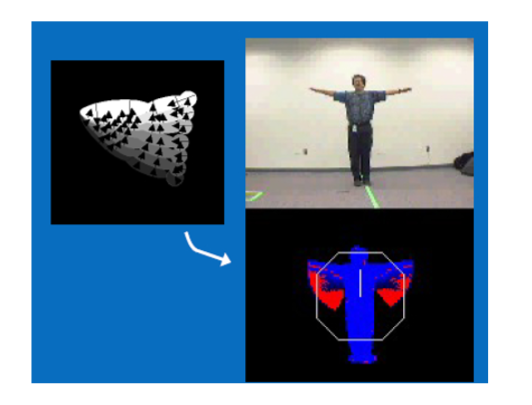
11. XML I/O

12. Basic Image Processing: Image filters, Morphology, Pyramids, Color space conversions, Geometrical transformations, Histograms.

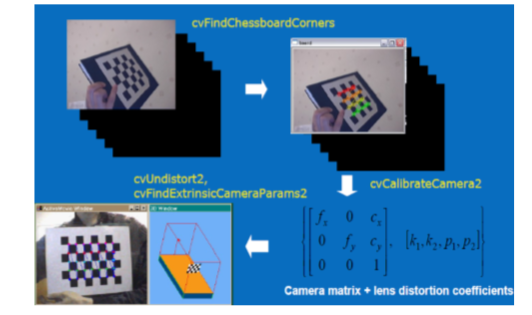
13. Advanced Image Processing and Feature extraction: Corner detection, canny edge detector, Hough transform, Distance transform, Watershed, In painting…



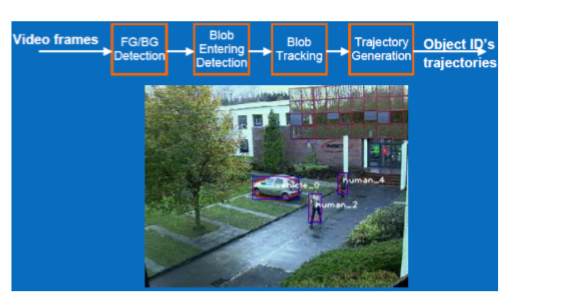
14. Shape analysis and Computational geometry: Image moments, contours, Delaunay triangulation and Voronoi Tesselation.

15. Motion Analysis: Optical flow, Object tracking (Meanshift & CAMSHIFT), Motion templates, and Kalman filters

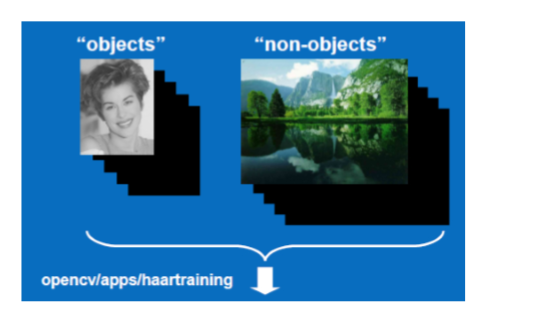
16. Camera calibration, Epipolar geometry



17. Advanced Blob tracking (Video Surveillance module)



18. Object detection (Haar classifier)





OpenCV is a long time project and has accepted reputation because of Good quality. And together with IPP it is used in different parts and for different kinds of programs, from small study projects to large advertisement products. Intel is still standing behind of OpenCV. OpenCV will continue to develop; the further grow will be program driven.

**Chapter 4**

**METHODOLOGY**

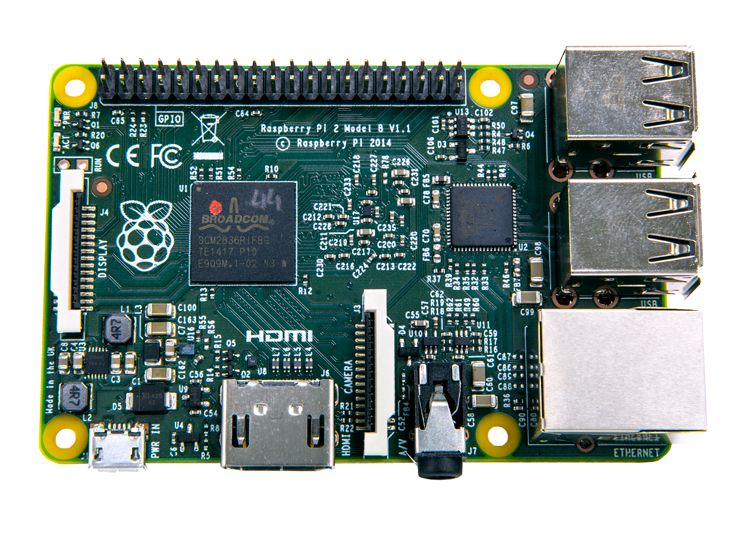
**4.1 Introduction**

The following sections will discuss the importance and usage of each hardware and software components included in this project as well as the working principle and algorithm of the system.

**4.2 Raspberry Pi 2**

Raspberry Pi is a series of credit card-sized single board computers developed in UK by the Raspberry Pi Foundation. The Raspberry Pi 2 Model B is the second iteration of the series which replaced its predecessor, the Raspberry Pi 1 Model B+ in February 2015. Due to its small size and low price, its processing power is very low when compared to modern personal computers, but enough to do many complicated operations such as image processing. Code optimization is very important for it to handle real time operations demanding such processing power as its hardware isn’t very powerful. The following are the technical specifications of the Raspberry Pi 2 Model B:

* A 900MHz quad-core ARM Cortex-A7 CPU
* 1GB RAM
* 4 USB ports
* 40 GPIO pins
* Full HDMI port
* Ethernet port
* Combined 3.5mm audio jack and composite video
* Camera interface (CSI)
* Display interface (DSI)
* Micro SD card slot
* Video Core IV 3D graphics core

Also, the Raspberry Pi 2 only needs a 5V power source that can deliver 500mA current, making it possible to be operated via USB.

This makes it a very powerful portable “mini-computer”. The presence of several different modules that can add to the functionality of the Pi as well as the open source nature of its OS and programs make it a very versatile device. Thus it can be used for several different applications without needing much modifications. An ARMv7 CPU allows it to even run the full range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, as well as Microsoft Windows 10.

This project uses image processing which demands a lot of processing power. The large number of libraries present for Raspberry Pi as well as its high processing power make it an ideal device for our project. We are using the official Raspbian OS provided by the developers on the RPi.

**4.3 Raspberry Pi NoIR Camera**

The NoIR (No Infrared) Camera is one of the numerous modules available for Raspberry Pi. As the name suggests, it is an Infrared Camera, i.e. it lacks an infrared filter which makes it much more sensitive to IR than normal cameras. This makes it ideal for low-light or night photography, wildlife photography and several other scientific applications such as monitoring health of plants. The following are the specifications and features of the camera:

* No IR filter - Great for Low Light Conditions
* 5MP Omni vision 5647 Camera Module
* 1.4 µm X 1.4 µm pixel with OmniBSI technology for high performance (high sensitivity, low crosstalk, low noise)
* Optical size of 1/4" (3.2 mm x 2.4 mm sensor size)
* Image Control Functions:

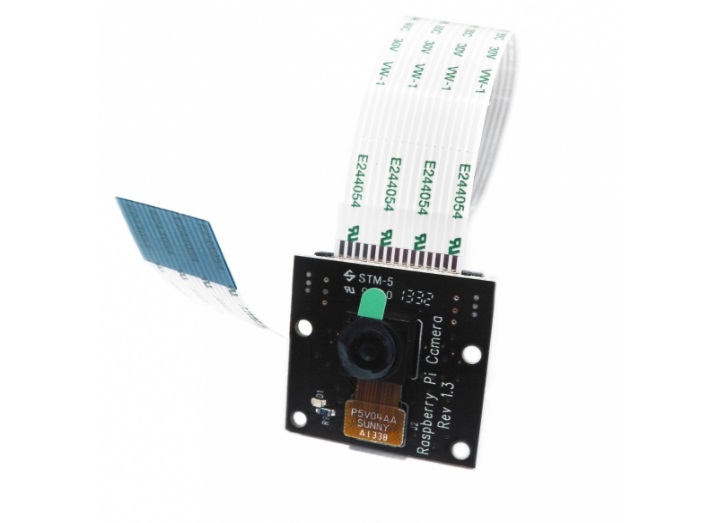
- Automatic exposure control (AEC)

- Automatic white balance (AWB)

- Automatic band filter (ABF)

- Automatic 50/60 Hz luminance detection

- Automatic black level calibration (ABLC)

* Programmable controls for frame rate - AEC/AGC 16-zone size/position/weight control, mirror and flip, cropping, windowing, and panning
* Still Picture Resolution: 2592 x 1944 pixels
* Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90fps Recording
* Size: 20 x 25 x 9mm
* Weight 3g

The reason we selected NoIR camera for our project is primarily to overcome the drastic problems low illumination can have on the face detection algorithm. This is of prime importance in out project as there will frequently be cases where the system has to work in low-light conditions. While the colours in normal illumination get washed out, it has very little if not no effects on the face detection algorithm. Also the high resolution of the camera enables us to be able to detect smaller faces in the image, though it does add significantly to the processing power needed. The presence of different modes allows us to decrease the resolution if the high resolution is not necessary too.

**4.4 OpenCV**

OpenCV (Open Source Computer Vision) is a library of computer functions primarily aimed at real-time computer vision and image processing, originally developed by Intel and now maintained by Itseez. It is a cross platform library and free to use under the open-source BSD license. The large number of image processing functions offered by OpenCV make it one of the, if not the most used image processing library for computer vision projects. It is primarily coded in C++ and its primary interface is C++ but it has bindings in Python, Java and MATLAB, making it easy to use across different platforms.

As our project includes a lot of image processing, this was the ideal library we needed. It has in-built cascades for face detection. The cascades are not limited to only faces but also cover several other facial features such as ears, eyes, eyes with glasses, upper body, etc. Thus multiple cascades can be used in parallel to enable a higher detection accuracy, though this will need significantly more processing power. In addition to having the earlier mentioned features, OpenCV also has an in-built detector which can be trained to detect objects. The detector takes in positive images (containing the object) and negative images (not containing the object) and trains a cascade. The accuracy of the trained cascade depends on the number of images taken for positives and negatives as well as the images themselves. To get a reasonably high accuracy thousands of positives and 5-10 times more negative images are required, making the task of training the cascade a daunting one.

**4.5 Python**

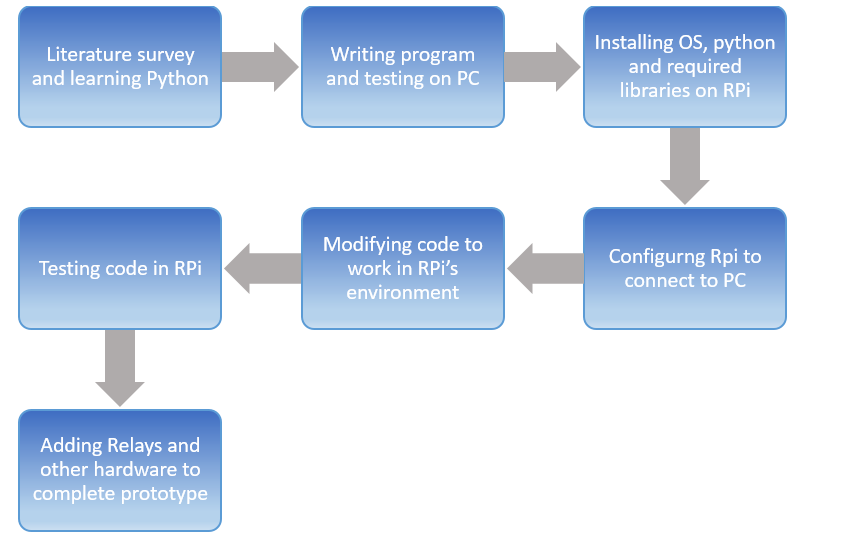
Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems. Due to the large scale usage of python, it has a vast collection of libraries available for innumerable applications.

As we needed to learn a new language for this project, we intended to take a language which would be easy to learn and code, while also offering versatility. Python offered all that we needed and was quite easy to learn and understand. The simplicity of the language helped us speed up the progress of the project significantly, giving us the time to do more and better things for our project. Also OpenCV had bindings in python, giving us more reasons to choose the language over others.

**4.6 Setting up connection between Raspberry Pi and PC**

For configuring the Raspberry Pi, we would normally require I/O peripherals such as keyboard, mouse, Wi-Fi adapters and a monitor. As students having only laptops, getting hands on such peripherals meant either a trip to the lab or buying all of them, which would prove to be expensive and also bulky to move around, greatly reducing the portability of the system. Thus we searched the internet for a way to configure the RPi to use the I/O devices of the laptop for the RPi. For this we have used two software’s: Putty and Xming. The former establishes a network connection between the laptop and the RPi via an Ethernet cable and starts a session with a terminal window. This requires us to add a specific IP address (IP address in the range of the network we are using to connect) to the config.txt file of the RPi and use the same IP to configure a session on Putty. The terminal window allows us to login to the Pi’s OS and do whatever we can with the terminal. The latter, Xming, is used to start a GUI session with the RPi. Entering the command “startlxde” into the terminal window opened by Putty opens up Xming windows which show the Raspbian Desktop and the taskbar. With this we can navigate using the GUI of Raspbian, which is much more convenient for us. The Raspberry Pi was powered using the USB port of the system. The former establishes a network connection between the laptop and the RPi via an Ethernet cable and starts a session with a terminal window. This requires us to add a specific IP address (IP address in the range of the network we are using to connect) to the config.txt file of the RPi and use the same IP to configure a session on Putty.

**4.7 Design Stages**



**4.8 Project working flow chart**

****

**Programming flow Chart**

Get image from camera

Search for faces

No

Faces Found?

Yes

Get Coordinates, size, number of faces

Calculate distance of faces from camera and subtract from distance of fans for all faces

Calculations for all faces done?

No

Change flag of fans based on difference in distances

Yes

Switch on/off fans based on flags

**4.9 Working algorithm**

We have used Voila-jones face detection algorithm in this project to detect faces. The detector takes samples of different sizes (window size) from the image and compares them to the face detection cascade given as input to the program. Once a face is detected, the function outputs the coordinates of the face (in the image) as well as the width and height of the face. The coordinates of the face on the image are mapped to actual coordinates of the face in the room. This is done by dividing the actual width of the visible room (in cm) and width of the image (2592 pixels) to get a constant with unit of cm/pixel. This constant can then be used to know the horizontal and vertical position of the faces in the real room. The following formula is used to calculate the distance between the camera and the face (in cm):

From the technical specifications of the camera, we can find out the focal length and sensor size, thus the “object size in image”. From a study of average sizes of human head, we know that an average adult measures about 16 cm from ear to ear (reference: <https://en.wikipedia.org/wiki/Human_head#/media/File:HeadAnthropometry.JPG>). Using this as the “Object Size”, we can know the approximate object distance (distance of the face) from the camera.

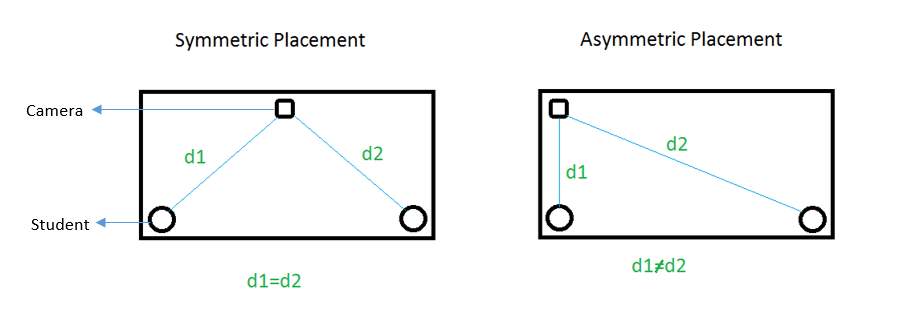
Once the distance of the face is known, it is compared to the distances of fans and lights (pre-stored in the code). The two distances are subtracted and the absolute value is taken. If the absolute value is less than 100, i.e. the distance between the face and the fan is less than 100 cm, the flag of the corresponding fan is set to 1. This process is repeated for all fans for each face. If a flag is already set to 1 by any of the above iterations, it is not set back to 0. Next the array containing the flags is read and the appropriate fans and lights are switched on via relays and Raspberry Pi’s GPIO pins. The flags from the analysis of the previous image shouldn’t interfere with the flags from the current analysis. For this 2 arrays are maintained, one for the current flags and the other for previous image flag array. This helps use note the changes in status of flags.

The number of people also has a direct effect on the flags. For 0 faces detected, all flags will be turned to 0 and for number of faces greater than 6 (more than 6 people present), all the flags will be set to 1. Thus all fans and lights will be switched off in case there is no one in the room while all of them will be switched on if there are more than 6 people in the room.

To overcome the problem of variable illumination and low light conditions, which can cause severe problems for face detection leading it to fail, we are using IR camera which has excellent low light performance and performs well in well-lit conditions as well.

As the images taken from the camera are very high resolution (for face detection), to ensure real-time operation of the system and reasonably fast processing, the minimum and maximum window size for face detection have been calculated thru trial and error. This has resulted in a significant drop of processing time from more than 10 seconds per image to less than 5 seconds per image.

**4.10 Camera Positioning**

****The camera will be setup near one of the corners of the classroom. This asymmetric positioning of the camera will ensure that each face has a different distance from the camera. In contrast, a symmetric positioning, such as at the centre of the wall, will result in two or more people having the same distance from the camera, thus confusing the algorithm and producing undesired results.

But due to the asymmetrical positioning of the camera, half of the viewing angle (65o) will be wasted. Also, though it does reduce equal distance conflicts, it doesn’t eliminate it completely. As shown in the results, the already less viewing angle of the Pi camera, when halved, only covers 1 column of benches. Thus to cover the entire classroom while avoiding such equal distance problems, we would require 3 such systems per classroom (1 system per column of benches). Each system will only be connected to the fans and lights in that specific column, ignoring the other fans and lights in the room. This will solve the problem of equal distances as well as covering the entire width of the classroom.

**Chapter 5**

**RESULTS**

In this chapter we will discuss the results we got from the project. Pictures, screen shots of the program and output can be found in this section. The following pictures show the output of the system in different situations.

In low light conditions:

As it is clearly visible, the detected faces are highlighted by green rectangles. We have assumed 5 fans in this scenario. The terminal window in the bottom right shows the values of different parameters:

1st Line: Number of faces detected in the picture – 4

2nd Line: Distance of the faces from the camera (in cm) – 405, 327, 265 and 226.

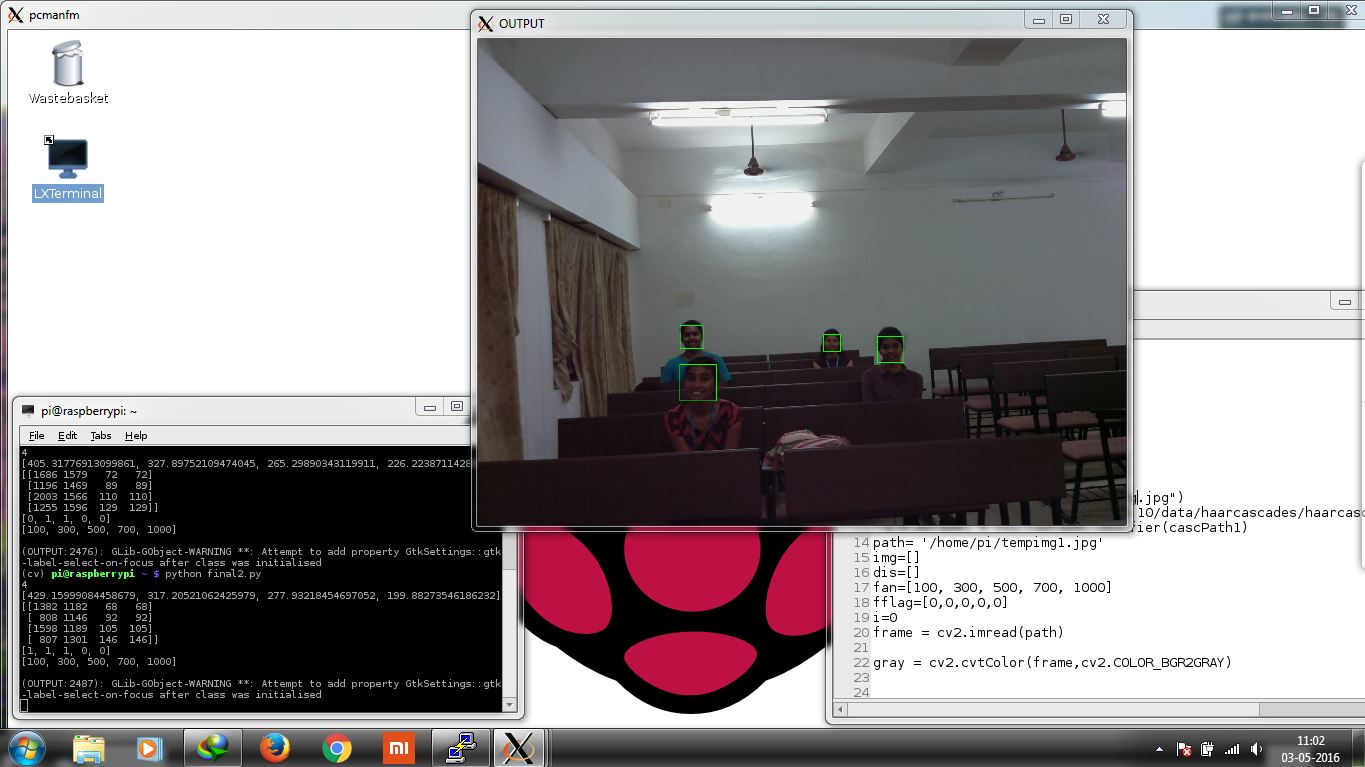
3rd Line: The coordinates of the faces as well as the dimensions of the rectangle-

4th Line: The flags indicating the status of each fan (1 - fan is on, 0 – fan is off) – [0, 1, 1, 0, 0]

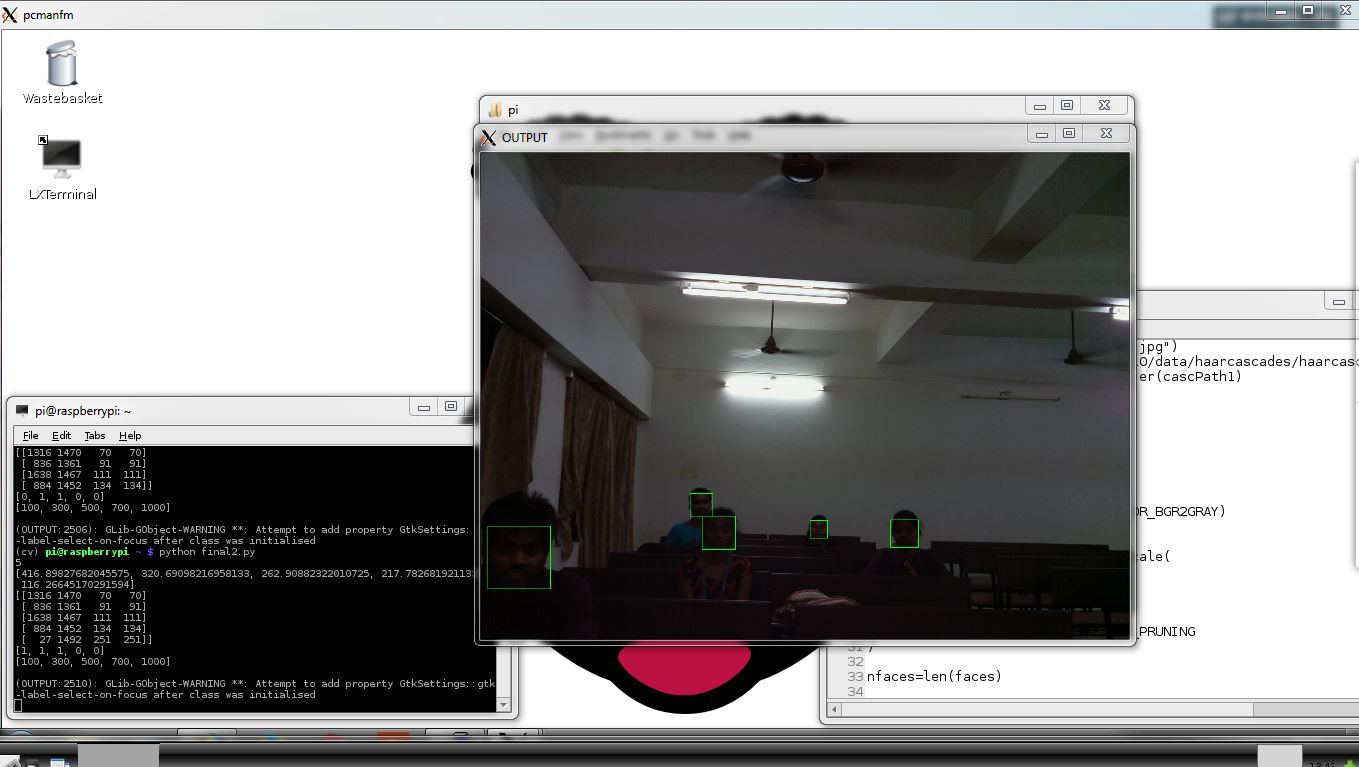
5th Line: Assumed distance of the 5 fans from the camera (in cm) – [100, 300, 500, 700, 1000]

From the results in the above image, we can see that the algorithm is working well. It detects 4 face, highlights them, calculates distance from each fan and then decides which fans should be off and which should be on based on the difference in distance from the fans and the faces.

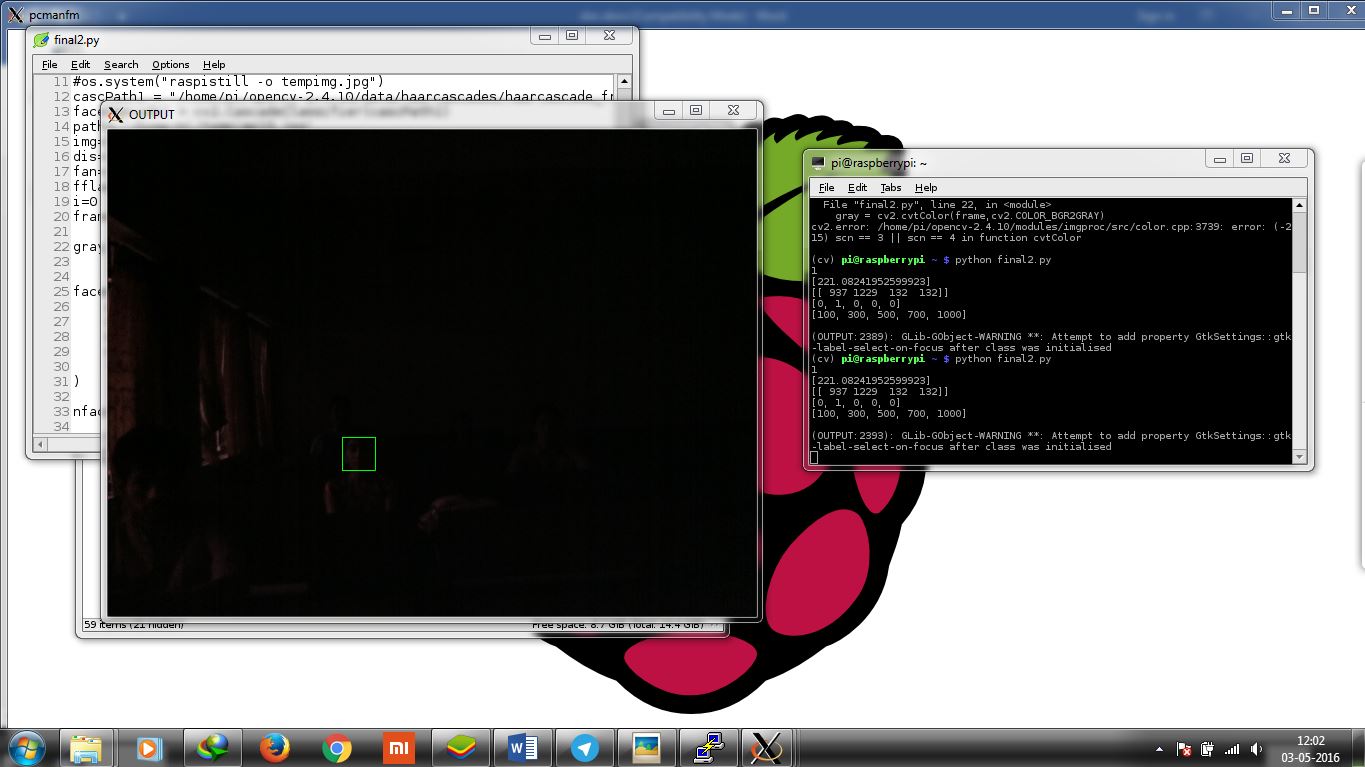
The following are a few more images demonstrating the same results at different distances, number of people, and lighting conditions.

In a well illuminated classroom:

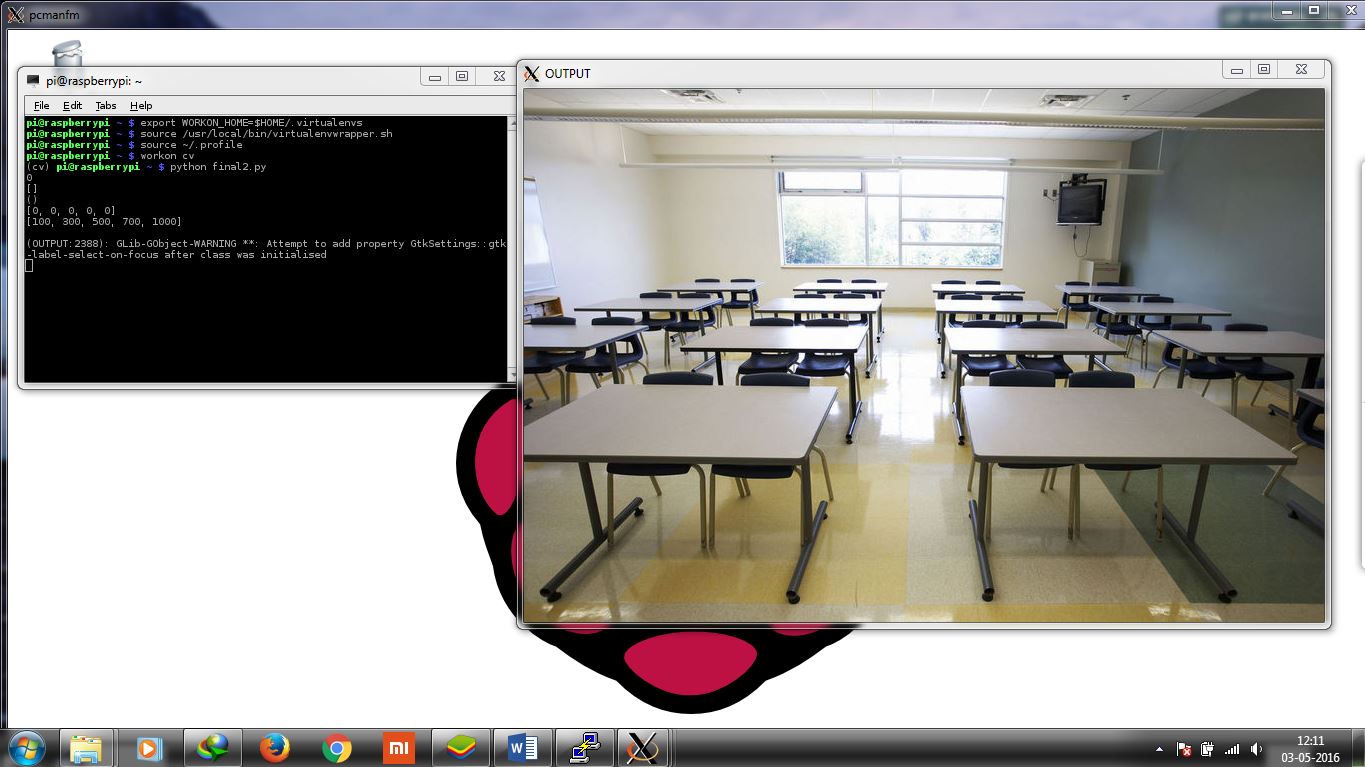
With only 1 person:

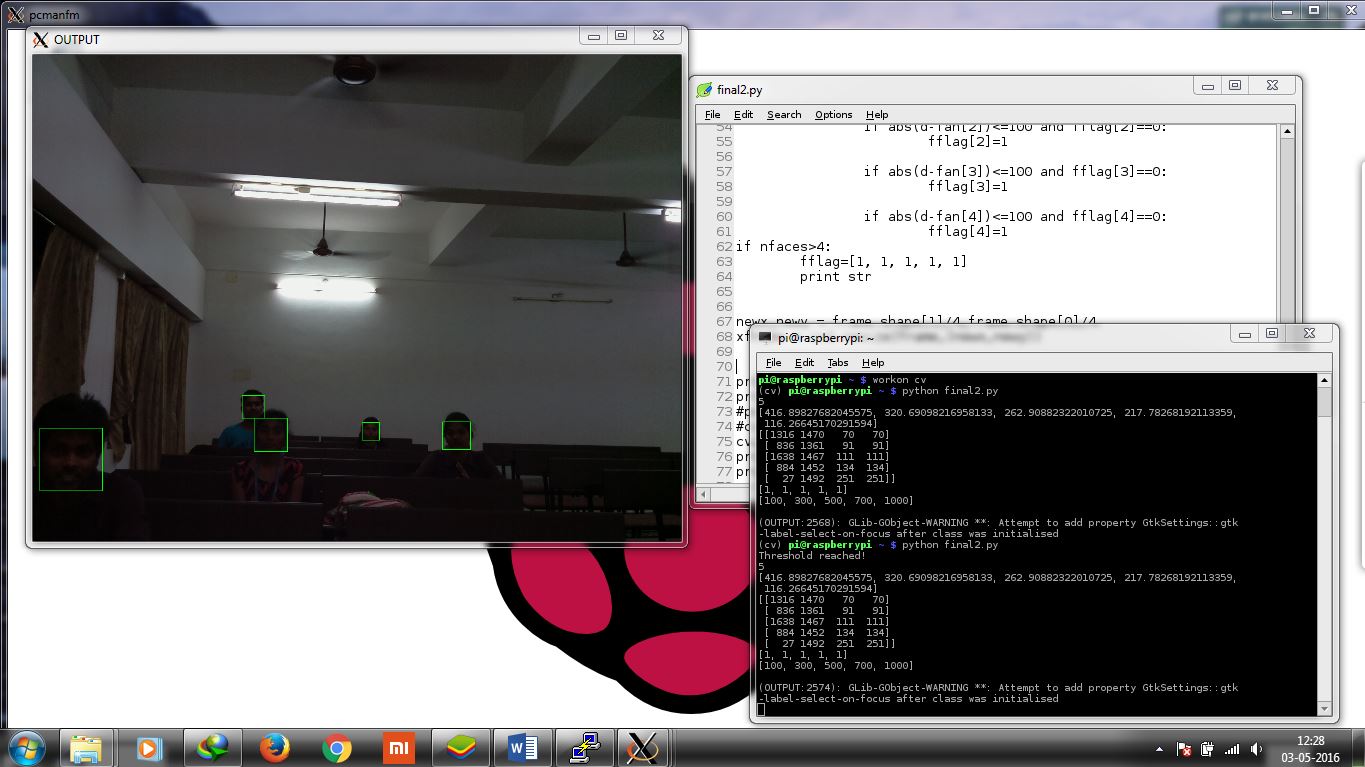
Low light condition:

Complete Darkness:

As we can see from this picture, the camera doesn’t work well in complete darkness. Adding IR lights to the system can solve this problem.

The following picture has not been taken from the pi camera, but from the internet, just to demonstrate the results of having no one in the classroom (empty classroom).

Empty classroom

To demonstrate the result of having number of people more than the threshold (the no. of people needed for all fans and lights to be switched on), we have decreased the threshold from 6 to 5. The following picture shows the output:

The message “Threshold reached!” in the terminal shows that 5 (threshold no.) people are present in the class. The same image processed earlier gave different flag values at output. This shows that the threshold function works.

**Chapter 6**

**Limitations and Future Scope**

The system we designed is working as intended and is giving good results. Though it still has a lot of limitations, which will be discussed in the next section, it works perfectly in controlled environments.

This project has thought us many things, academic and non-academic, like learning a new language, debugging programs, searching the internet for solutions of different problems, overcoming limitations by using different approach to the problem or changing the hardware involved. It also introduced us to and helped us gain a lot of knowledge about Raspberry Pi and Python, both of which are powerful tools which and we believe this will definitely help us in our future endeavors.

**Future scope**

As mentioned in the previous section, this project still has a lot of limitations. One of the primary limitations of the project, as it is clear from the results, is the face detection algorithm itself. Despite there being more than 2 people in the room, the algorithm only detects 2 faces. It is mainly because of the orientation of the faces. The cascade file used for face detection in the program only works on frontal view of faces. Thus any other tilted orientation of the face will make the algorithm fail at detecting the face, making the system itself fail. A possible solution to this is to use multiple cascades for detecting various orientations of the face. But this will need much more processing power than the current requirement, increasing the overall hardware requirement of the system. A more robust face detection algorithm is also a possible solution for this problem.

The second limitation of this system would be the size of the classroom. If the classroom is too big, the camera’s field of view may not be able to cover all of the width of the classroom, thus unable to detect the people at the edges. The length of the room is also a cause for concern as the farther a person is from the camera, the smaller the face becomes, and the harder it becomes for the algorithm to detect it. We will need to decrease the “window size” and the “min. neighbors” of the face detection window in the function in order to detect smaller faces. But doing this will also increase the number of false positives in the results, leading the system to detect “faces” where there are actually no faces. This will result in unnecessary wastage of electricity, going against the very purpose of this project. This problem also can be overcome with a more robust and accurate face detection algorithm or by using background subtraction to extract only the changes in the background. This will result in the subtracted output only containing the students in the class, as the “background” image will contain an empty classroom. A face detection analysis on such an image will have drastically low false positives and take lesser time to process due to most of the pixels in the image being black. This limitation can also be overcome by placing more than one camera in the classroom such that the faces of the students farther to the first camera are nearer to the 2nd camera, and thus the “window size” and “min. neighbors” need not be reduced for face detection.

The last limitation of this project would be the update speed. The image processing requires a lot of time, about 20s on a full resolution (2592 x 1944 pixel) image taken from the Pi Camera. Thus the results are not instantaneous and there is significant lag present in the system. This can be solved by increasing the processing power of the system by getting better performance hardware or by optimizing the code, or by scaling the image down to a lower resolution before processing it. We have scaled the image to reduce the latency and have succeeded in getting it down to 5s by resizing the original image to ¼ its size. But this compromises face detection accuracy as smaller faces will be harder to detect on the resized low resolution image.

**Chapter 7**

**Conclusion**

We have successfully designed a system using Raspberry Pi & NoIR Camera as hardware and Python & OpenCV as software used. The system we designed is working as intended and is giving good results in controlled environments, though it still has a lot of limitations and improvements to be done for making it workable in a

real-time and practical scenario.

The system was tested in controlled environments and pictures of the classroom were taken and given as input to the program. The output image as well as the results have been posted in the results section. The output image shows the detected faces highlighted by green rectangles and results give the various other information such as distance, number of faces, status flags of the fans etc. The accuracy of the system depends on the accuracy of the face detection algorithm used. Since we have only used frontal face cascade due to limitations in processing power and latency, the accuracy in a general scenario is about 50%, whereas it is above 90% in controlled situations.

This project has thought us many things, academic and non-academic, like learning a new language, writing and debugging programs, searching the internet for solutions for different problems, overcoming limitations by using different approach to the problem or changing the hardware involved. It also introduced us to and helped us gain a lot of knowledge about Raspberry Pi and Python, both of which are powerful tools which we believe will definitely help us in our future endeavors.

**REFERENCES**

1. Price, Peter.(2011). "Can a £15 computer solve the programming gap?". BBC Click.
2. Moorhead, Joanna.(2012). "Raspberry Pi device will reboot computing in schools'". The Guardian (London).
3. Gary Bradski. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. ISBN-13:978-059616130
4. Daniel Lellis Baggio. (2012). Mastering OpenCV with Practical Computer Vision Projects. ISBN-13: 978-184517829
5. Carlo Tomasi Takeo Kanad Detection and Tracking of Point Feature Technical Report CMU-CS-91-13
6. NicolaiPetkov·EaswarSubramanian Motion detection, noise reduction, texture suppression, and contour enhancement by spatiotemporal Gabor ﬁlters with surround inhibition Published online: 25October2007
7. Motion\_Detection\_Programming\_Guide\_V1.1 GM8126
8. Price, Peter “Can a £15 computer solve the programming gap?". BBC Click. 2 July 2011.
9. "David Braben on Raspberry Pi". Edge. 25 November 2011. 8 December 2011.
10. Moorhead,Joanna “Raspberry Pi device will 'reboot computing in schools'". The Guardian (London). 20 January 2012.
11. "Interview with Eben Upton – Raspberry Pi Founder". International Business Times. 19 March 2012.
12. Linux news showing the first release of Debian Squeeze for Raspberry running on QEMU". Linuxnewshere.com. 16 July 2012.
13. Raspberry Pi maker says code for ARM chip is now open source". Ars Technica. 3 November 2012.
14. Shead, Sam. "Raspberry Pi delivery delays leave buyers hungry (and angry)". ZDNet. 18 October 2012.
15. Vallance, Chris. "Raspberry Pi bids for success with classroom coders". BBC News. 29 February 2012.
16. Alan Winston(2002). OpenVMS with Apache, WASD, and OSU: The Nonstop Webserver (HP Technologies).
17. Face Detection and Recognition using OpenCV, Article, http://shervinemami.info/faceRecognitio n.html, published by Shervin Emami, 2010.
18. Seeing with OpenCV, Article, http://www.cognotics.com/opencv/servo\_2007\_series/part\_1/index.html, published by Robin Hewitt, 2010.
19. OpenCV Homepage, http://opencv.willowgarage.com
20. Face Recognition Homepage, <http://www.face-rec.org/algorithms>
21. Wikipedia, Three-dimensional face recognition, http://en.wikipedia.org/wiki/Threedimensional\_face\_recognition
22. Wikipedia, Active appearance model, http://en.wikipedia.org/wiki/Active\_appe arance\_model
23. Computer Vision Papers, <http://www.cvpapers.com>
24. Wikipedia, Histogram equalization, http://en.wikipedia.org/wiki/Histogram\_ equalization

# **Appendix A**

**Code**

Import cv2

Import time

Import sys

Import os

Import RPi.GPIO as gpio

gpio.setup (2, gpio.OUT)

gpio.setup (3, gpio.OUT)

gpio.setup (4, gpio.OUT)

gpio.setup (17, gpio.OUT)

gpio.setup (27, gpio.OUT)

f= [2, 3, 4, 17, 27]

def dist(w):

d=10\*0.36/(w\*0.00012336)

return(d)

os.system("raspistill -o tempimg.jpg")

cascPath1 = "/home/pi/opencv-2.4.10/data/haarcascades/haarcascade\_frontalface\_default.xml"

faceCascade1 = cv2.CascadeClassifier(cascPath1)

path= '/home/pi/tempimg.jpg'

img=[]

dis=[]

fan=[100, 300, 500, 700, 1000]

fflag=[0,0,0,0,0]

i=0

nframe = cv2.imread(path)

newx,newy = nframe.shape[1]/4,nframe.shape[0]/4

frame = cv2.resize(nframe,(newx,newy))

gray = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

faces = faceCascade1.detectMultiScale(

gray,

scaleFactor=1.1,

minNeighbors=5,

minSize=(100,100),

flags=cv2.cv.CV\_HAAR\_DO\_CANNY\_PRUNING

)

nfaces=len(faces)

for (x, y, w, h) in faces:

cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 0),2)

if i<nfaces:

img.append([w,h])

i=i+1

for (w,h) in img:

dis.append(dist(w))

if nfaces>0 and nfaces<6:

for d in dis:

if abs(d-fan[0])<=100:

fflag[0]=1

if abs(d-fan[1])<=100:

fflag[1]=1

if abs(d-fan[2])<=100:

fflag[2]=1

if abs(d-fan[3])<=100:

fflag[3]=1

if abs(d-fan[4])<=100:

fflag[4]=1

if nfaces>5:

fflag=[1, 1, 1, 1, 1]

if fflag[0]==1:

gpio.output(f[0], True)

else:

gpio.output(f[0],False)

if fflag[1]==1:

gpio.output(f[1], True)

else:

gpio.output(f[1],False)

if fflag[2]==1:

gpio.output(f[2], True)

else:

gpio.output(f[2],False)

if fflag[3]==1:

gpio.output(f[3], True)

else:

gpio.output(f[3],False)

if fflag[4]==1:

gpio.output(f[4], True)

else:

gpio.output(f[4],False)

print dis

print img

cv2.imshow('OUTPUT', frame)

print faces

print fflag

while cv2.waitKey(1) & 0xFF != ord('q'):

time.sleep(0.1)

cv2.destroyAllWindows()

**Curriculum vitae**

Name : K. HIRANYA GARBHA

Father’s name : Kumar Diwakar Kashyap

Date of Birth : 25/02/1995

Nationality : INDIAN

Sex : MALE

Permanent Address : DNo. 46-13-6/c, Mosque Street, Danvaipeta,

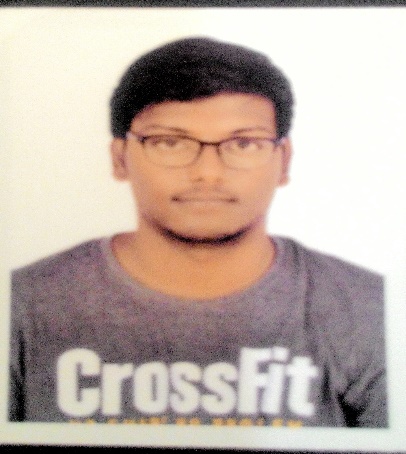
Rajahmundry, Andhra Pradesh.

Phone Number : +919597452935

Mobile : +919949208977

Email Id : khiranyag@gmail.com

**Curriculum vitae**

****

Name : B. SANDEEP KUMAR

Father’s name : B. Ramesh

Date of Birth : 01/05/1994

Nationality : INDIAN

Sex : MALE

Permanent Address : B1-104, nagole x road, bandlaguda,

Hyderabad, Telangana

Phone Number : 8870900660

Mobile : 9701156632

Email Id : burrasandeepkumar1994@gmail.com