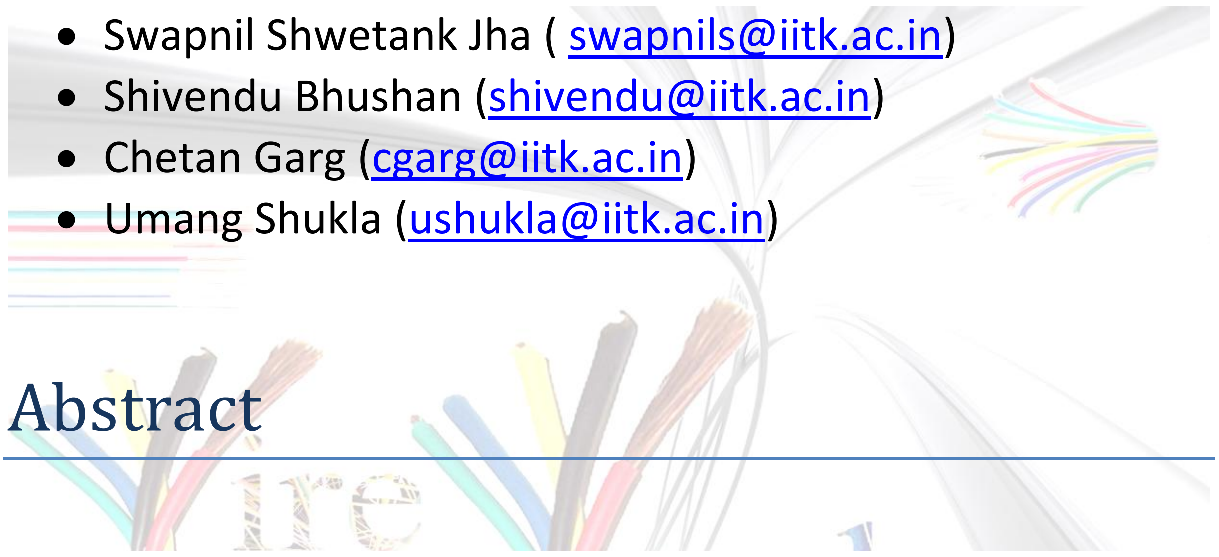
Panoramic view

-----Wireworks

Team Members:

- Swapnil Shwetank Jha (swapnils@iitk.ac.in)
- Shivendu Bhushan (shivendu@iitk.ac.in)
- Chetan Garg (cgarg@iitk.ac.in)
- Umang Shukla (ushukla@iitk.ac.in)



Abstract

Our project aims at simulating an environment which makes the user to visualize things as if he's present at the place which the 'panoramic' image shows. We desire to achieve this using the gaze tracking of the user and synchronising the movement of mouse cursor accordingly. We will need a pupil tracking algorithm that is near perfect so that whatever section the user wishes to view on the panoramic image comes in the frame of the screen.
Eye detection is very important for automatic gaze tracking. Here, we implement an eye detection algorithm which woks under active infrared (IR) illumination. By making use of the bright pupil effect and a sophisticated thresholding method, the pupil candidates can be effectively located. The pupil candidates are then verified using the Haar Classifiers in OPENCV. We have used MATLAB as the coding platform.

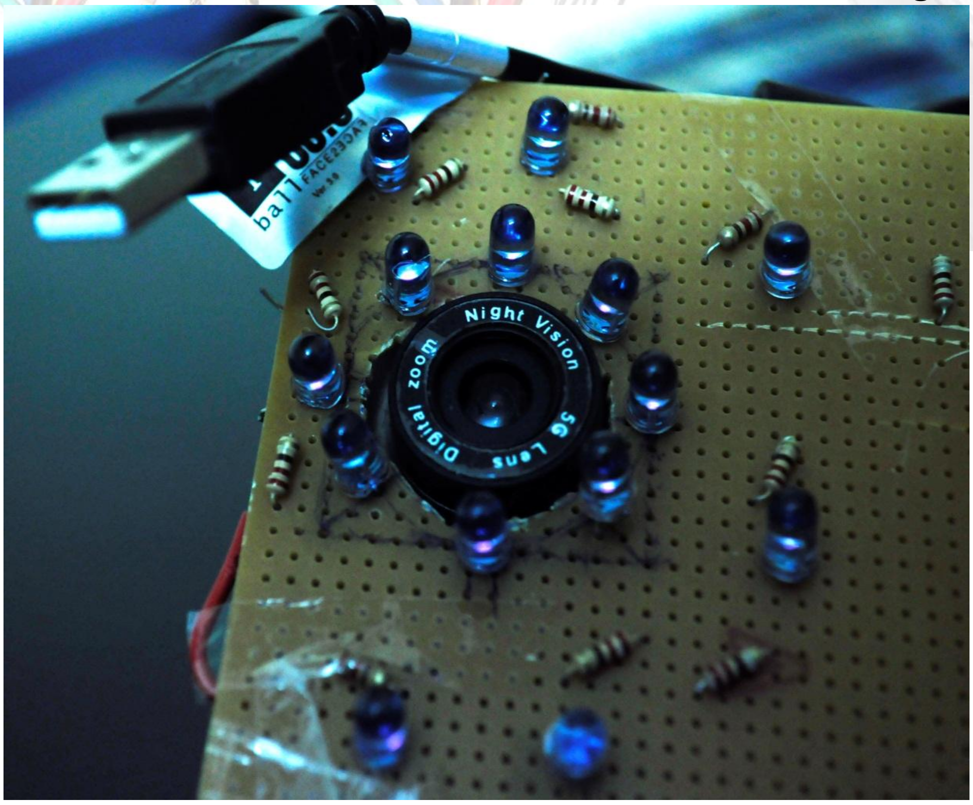
Motivation

We were inspired by the movie "Mission Impossible 4" where the hero tracks the eyes of the security guards with the help of a movable camera and creates a false illusion in their minds to showcase what he intends them to see. We were moved by this and thought why not use it otherwise i.e. not to misguide but just to help the user see for himself what he intends to see. The first idea that we got was to implement this on a 'panorama'. With just the tracking of the pupils we thought of implementing the screen movement to suit the interest of the viewer.

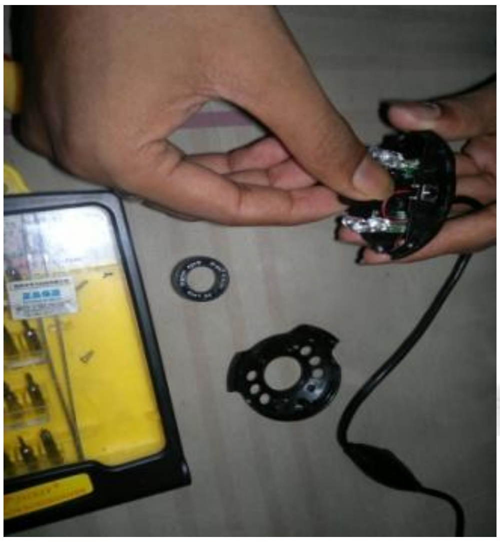
Hardware required

- A normal webcam
- Screw driver
- Film strip (negative)
- IR Led(s)

  All in all, the IR camera with the Leds looks something like this:-

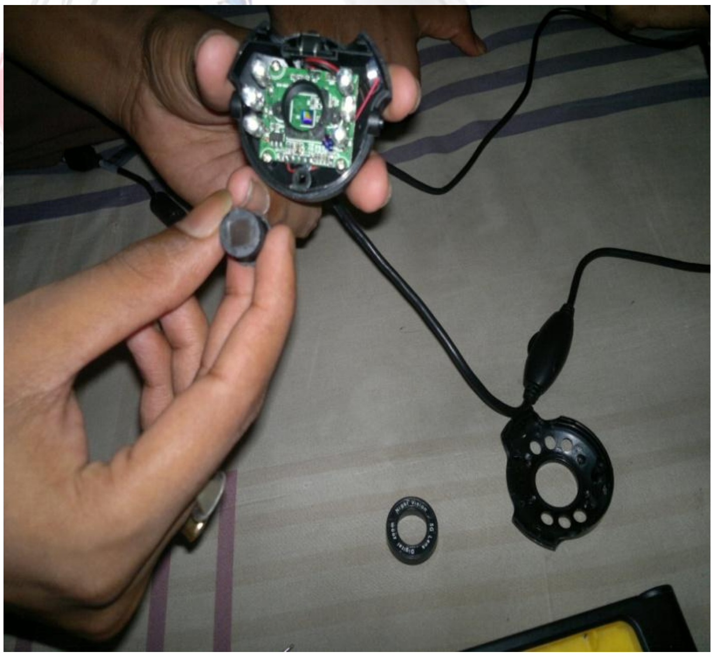


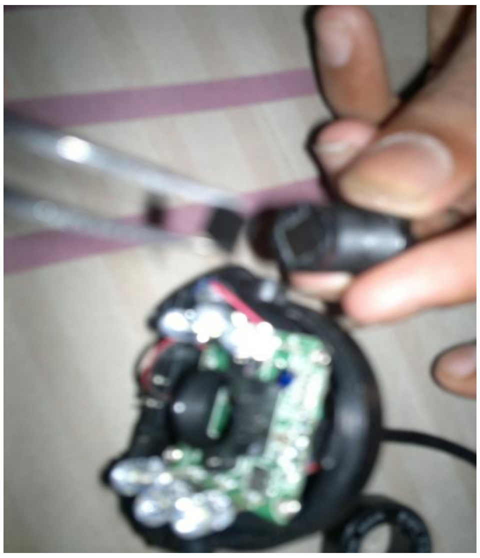
Making of "IR camera"



- Remove the crappy plastic stand.
- Remove the screws.
- Unscrew the lens assembly from it's holder. The focus on most webcams is achieved using the screw thread that is also used to hold the lens in. Turning it enough times will unscrew it completely.

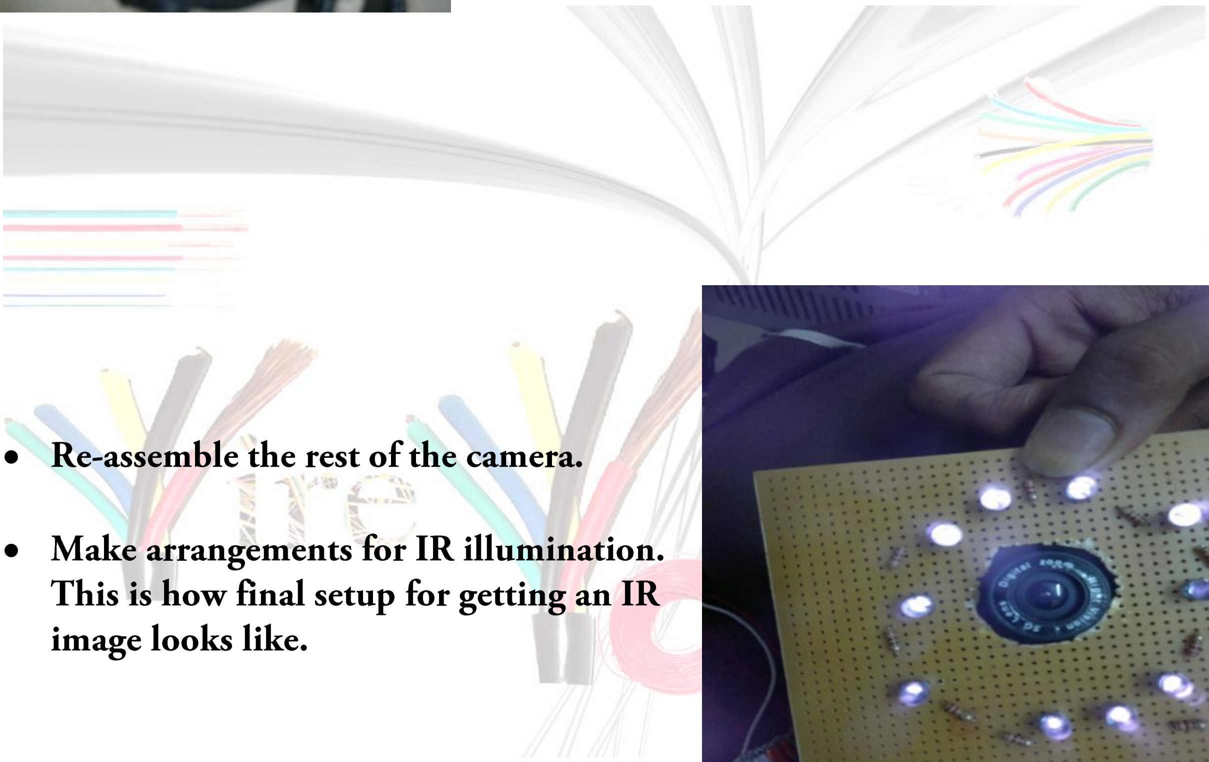
- Looking at the lens assembly in the picture you can see a small square of glass stuck in the back. Though it appears clear in the picture above, it has a red tint to the eye. "This piece of glass is the Infra Red Filter. It stops IR light getting through to the sensor. For our purpose this is bad so remove this piece of glass."





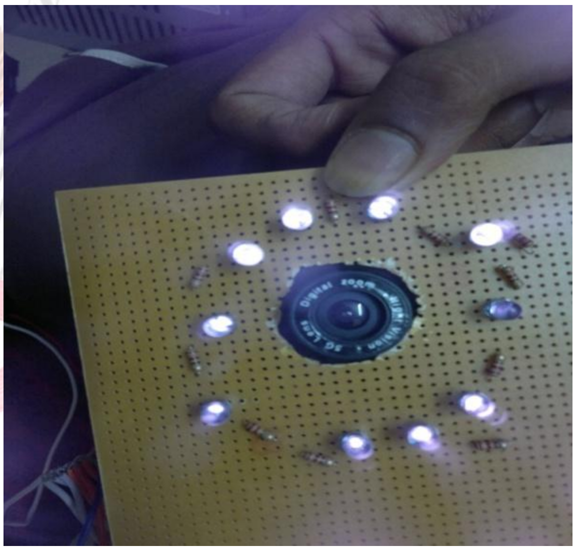
- Find a bit of absolute black photographic film, you can usually find a bit before the real photo's start. Cut two small bits out similar in size to the IR filter that you just removed.

- Fit the two bits of film where the old filter was present.

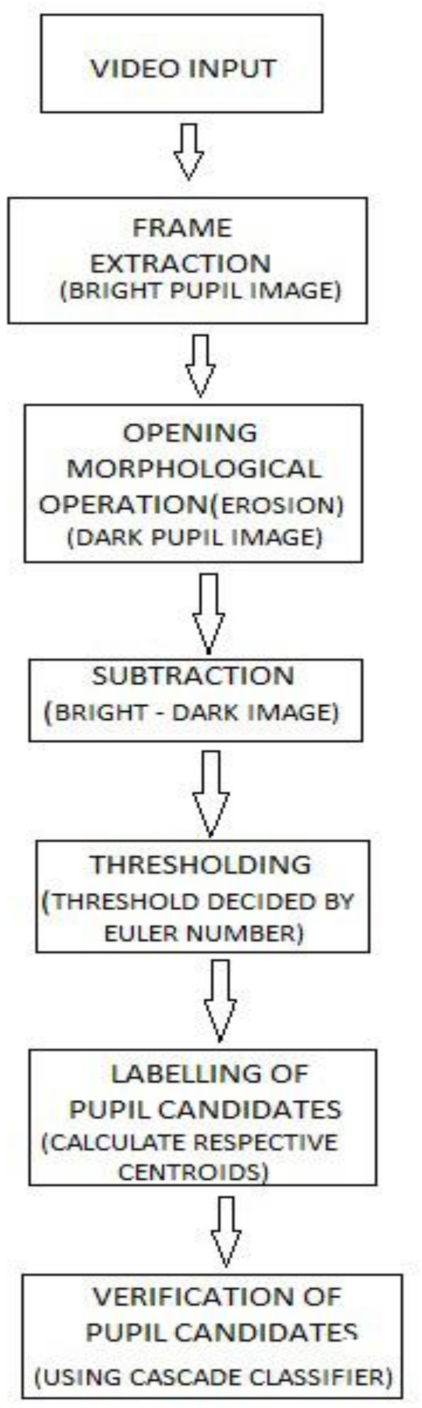


- Re-assemble the rest of the camera.

- Make arrangements for IR illumination. This is how final setup for getting an IR image looks like.



Proposed Algorithm


VIDEO INPUT
FRAME EXTRACTION (BRIGHT PUPIL IMAGE)
OPENING MORPHOLOGICAL OPERATION(EROSION) (DARK PUPIL IMAGE)
SUBTRACTION (BRIGHT - DARK IMAGE)
THRESHOLDING (THRESHOLD DECIDED BY EULER NUMBER)
LABELLING OF PUPIL CANDIDATES (CALCULATE RESPECTIVE CENTROIDS)
VERIFICATION OF PUPIL CANDIDATES (USING CASCADE CLASSIFIER)

This is the flowchart of processes involved in our "eye tracking algorithm."

Summary

“The method that we would be using relies on the fact that the pupil is ideally the darkest part of the face.”
We would be using the following two tools:-

1. The ‘bright pupil effect’[1]
2. thresholding method

With the help of these, we can effectively locate the pupil candidates [2]
Next, comes the verification part, Here, an appearance model (i.e. matching with a ‘database’) is used to verify the pupil candidates.

“This kind of approach needs a bright image and a dark image, and carries out pupil thresholding on the difference image. With aid of the Euler number, our thresholding method makes a good trade-off of accuracy and computational complexity. An appearance model is then used to verify the candidates, which outperforms the approach of template matching.”

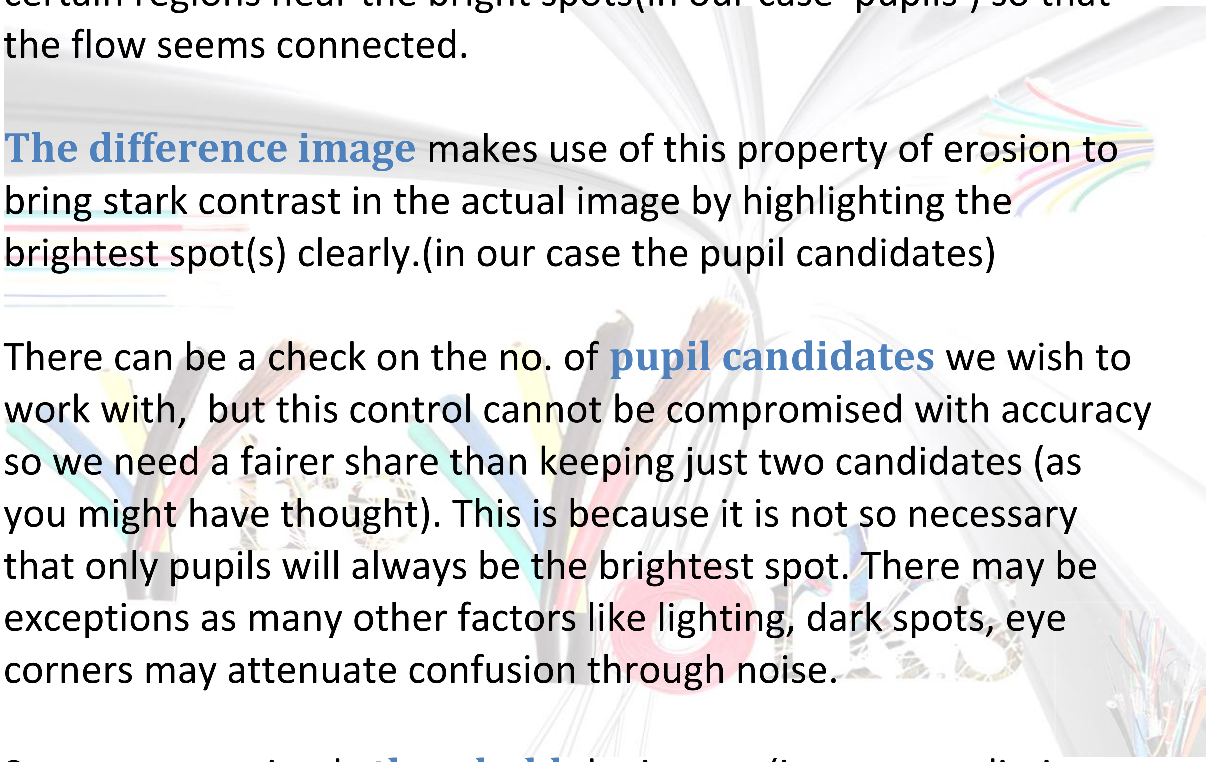
1 When an IR lighting source is placed close to the axis of the camera, the pupils look unnaturally bright because of the reflection of the blood-rich retina. This is the well known bright pupil effect.

2 Pupil Candidates are the points that are candidates for being the pupils. They are the bright spots which can be mistaken for pupils.

Detailing

Dark pupil image is obtained by applying opening morphological operation. Of the two available operations namely Erosion and Dilation, we chose Erosion.

The eroded image just brings about continuity in the colours of the image so that the change is not abrupt. This involves reconfiguring certain regions near the bright spots(in our case 'pupils') so that the flow seems connected.



The difference image makes use of this property of erosion to bring stark contrast in the actual image by highlighting the brightest spot(s) clearly.(in our case the pupil candidates)

There can be a check on the no. of **pupil candidates** we wish to work with, but this control cannot be compromised with accuracy so we need a fairer share than keeping just two candidates (as you might have thought). This is because it is not so necessary that only pupils will always be the brightest spot. There may be exceptions as many other factors like lighting, dark spots, eye corners may attenuate confusion through noise.

So, we appropriately **threshold** the image. (i.e. set up a limit on the no. of pupil candidates). For this we make use of the interesting 'euler no' of the image.

We convert the subtracted image to binary and use its euler number to appropriately threshold the image. At this point, a natural question would be what is this euler number and how does it help to threshold. Well! both answers are cleared if we go through their definitions:-

Euler no: the difference between the no of objects in a figure and the number of holes in it. (in our case, this will simply be the number of connected points or to be more precise a fairly indicative idea of the number of pupil candidates in a figure).

Thresholding: the process of setting up a threshold limit(L) such that all pixels with value more than L drop down to zero(become white) while those with less than L become black.

I guess this demonstrates how **Euler no.** could help us threshold which could finally give us a binary image that has just some white spots in black background as 'pupil candidates'.

<u>Euler number calculation</u> with simple global thresholding:-

1. Select an initial threshold (T) [in MATLAB, this no. is on a scale of 0 to 1]
2. Threshold the image using T.
3. Calculate the Euler number(n) of the binary image,

• If s < 0 or s > 1, stop;
• If euler.no_min _< n < euler.no_max, stop;
• If n < euler.no_min, then (s = s − Δs, go to Step 2;)
• If n > euler.no_max, then (s = s + Δs, go to Step 2;)

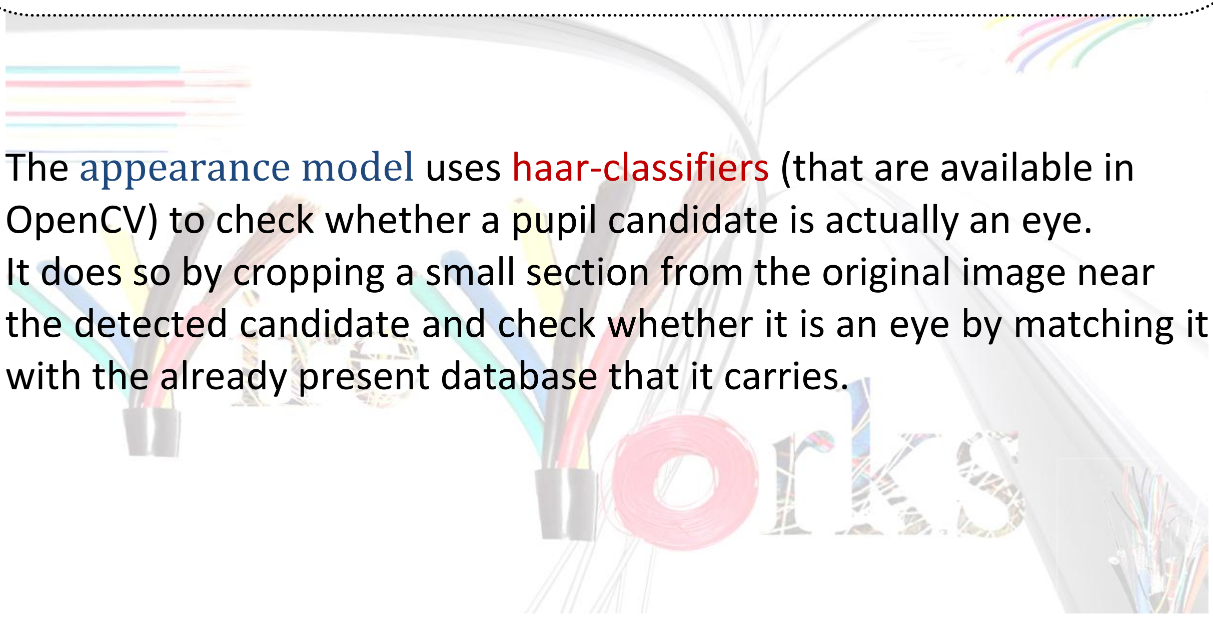
where 'euler.no_min' and 'euler.no_min' are predefined parameters.

The advantage of our method is that the number of pupil candidates can be controlled by parameters 'euler.no_min' and ' euler.no_max'.
Thanks to the Euler number, a simple global thresholding strategy is sufficient.

We used the following values in our code:-
euler.no_min= 2
euler.no_max= 4
initial_threshold(s) = .4(on a scale of 0 to 1)
limit(Δs)= .005

Once, we are done with this part we need to get to the actual eye. We'll make use of an appearance model for this.

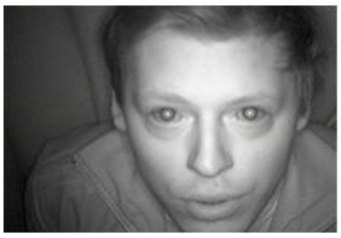
But before,
we need to get to the co-ordinates of the centre of all regions marked as pupil candidates and label them as well so that we could refer to them with their labels and extract the correct candidate using the appearance model label by label.
(to put it in simple terms: suppose, after appearance model we get to two of the (maximum) four candidates. We would still need to find out their exact centres to refer them at that pixel. So, we need to find the 'centroid' of each candidate)



The appearance model uses haar-classifiers (that are available in OpenCV) to check whether a pupil candidate is actually an eye.
It does so by cropping a small section from the original image near the detected candidate and check whether it is an eye by matching it with the already present database that it carries.

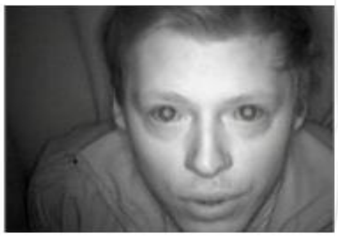
The **flowchart** on the 'next page' would make the flow of the concepts discussed so far, a lot more clearer.

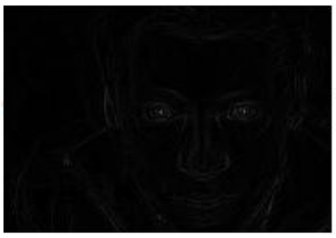
Video input



Bright pupil image

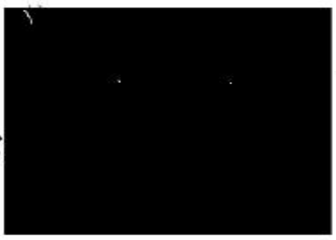
Eroded image

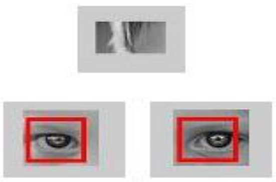




Difference image

Pupil candidates





Verification

y detected

Two directional gaze estimation

For viewing an panoramic image we need to move the mouse left and right, and to determine the need of moving left and right we estimate the direction of gaze.

Because of simplicity of the case we have to deal with, we can use the difference in pupil positions in successive frames to assess the gaze.

When viewing in two different directions, co-ordinate of pupil is likely to change, and depending on whether the x-coordinate of the pupil increases or decreases one of the two directions is identified as the direction of gaze.

Implementation of Panoramic View

The panorama rolls in to so as to bring the desired area of image in the view of the screen.

To achieve this, mouse pointer is synchronised with the direction of gaze according to the need of viewing.

The desired section (right or left) of the panoramic image is then dragged into the frame.

Acknowledgement

We would like to thank the **club coordinators** for bestowing so much faith on us by entrusting us with a summer project. They were ever ready for any kind of support.
Apart from them, our mentor **'Anuraag Pandey'** has also been extremely helpful guiding us all the way through.
We would like to specially mention **'Tanmay Gupta'** without whose support the project wouldn't have reached it's final form.
Also, **prof. Simant Dube** and **prof. Amitabh Mukherjee** for their extended support through mails.

References

1. [Bright pupil effect] http://www.ti1.tu-harburg.de/Mitarbeiter/ehemalige/zhao/ICPR2006.pdf
2. [Making of IR camera] http://www.hoagieshouse.com/IR/
3. [Including OpenCV in matlab] http://blog.cordiner.net/2010/02/15/opencv-viola-jones-object-detection-in-matlab/