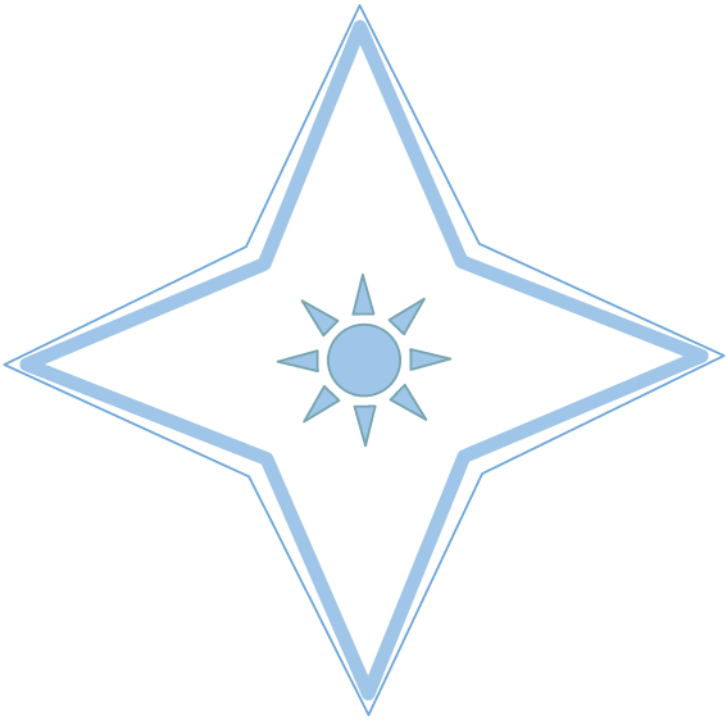
i Mouse

Human Computer Interaction via Head & Eye movements

Submitted for IBM Technical Web Contest 2011 Phase II



Apurva Gupta
Student, IIT Kanpur
C-438, Hall 10
+91 8765069141
apurva@iitk.ac.in

Rabi Shanker Guha
Student, IIT Kanpur
E-112, Hall 5
+91 7607448696
rabisg@iitk.ac.in

Megha Nawhal
Student, IIT Kanpur
G-109, GH - I
+91 9450037711
meghaa@iitk.ac.in

i Mouse

Human Computer Interaction via Head & Eye movements

Apurva Gupta
Student, IIT Kanpur
C-438, Hall 10
+91 8765069141
apurva@iitk.ac.in

Rabi Shanker Guha
Student, IIT Kanpur
E-112, Hall 5
+91 7607448696
rabisg@iitk.ac.in

Megha Nawhal
Student, IIT Kanpur
G-109, GH - I
+91 9450037711
meghaa@iitk.ac.in

Problem Description

With cities getting smarter and services automated, human-computer interaction is an indispensable aspect of everyday life. As most popular medium of interaction are keyboard and mouse, modern software interfaces are designed to work with hands. Thus, these interfaces are rendered unusable to people with hand disabilities. Since we all cannot do perfectly fine with keyboards, joysticks and mouse, investigating the possibility for eye-cursor control and its associated computer interactions becomes interesting. Although commercial eye trackers are available in market, they are not very practical for using at home due to either high costs(x000$) or difficult setup procedures. Thus, we need to make a system which uses cheap components easily available in market such as webcams, IR leds, goggles and has open source software allowing any user to calibrate/implement his own system.

In this paper, we describe a non-intrusive realtime interaction technology, which relies on eye and face movements to control a mouse cursor. Using 'off-the shelf' hardware, open source libraries and a laptop processor, we achieve real time eye-cursor control. We have described the hardware setup and algorithms used in our implementation of this technology. We call such a system eye Mouse or iMouse.

There is empirical evidence that using an eye based interface actually can be faster than traditional selection devices like mouse and joystick[2]. But available systems which implement this technology are either intrusive in nature or too CPU intensive to be executed as a background process. For example [1] requires the user to mount a head gear for tracking eyes and [3] requires supervised learning to determine gaze of user on screen. In comparison, our technique uses a normal webcam mounted on computer monitor and targets mouse control rather than determination of gaze (user can have coarse mouse control with head movements and fine controls with eyeball movements in our method).

Available Eye-tracking devices

Many open source eye trackers are available with different specialities but have the above stated problems. For example ITU gaze tracker [6] is an offline system which packs all processing equipment into a backpack allowing the user to carry system anywhere. This system can be utilized to identify gaze patterns in scenarios such as driving, buying in market or painting. Hence it can be used for researches aiming to classify changes in patterns with learning. This intrusive system consists of a head mounted webcam and thus is not practical for normal usage at home. Hence we aim to build a system and an algorithm which is non-intrusive and uses only off-shelf components such as normal laptop webcam and IR sensors to control cursor.

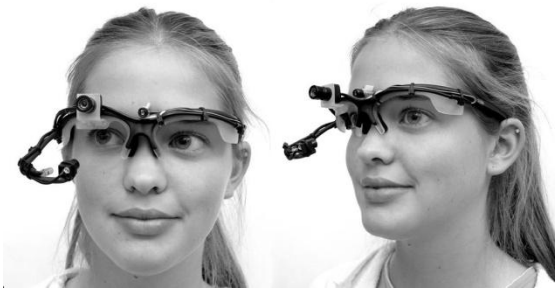


Figure : *ITU's intrusive gaze tracker which uses a mounted head gear to track gaze*

Design and Implementation

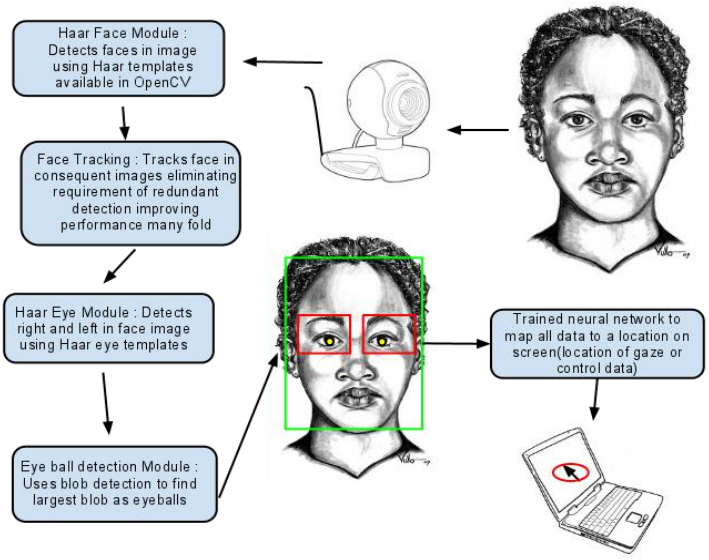
Eye mouse - A device to control mouse cursor on computer screen utilizing low resolution web cameras easily available in market. Purpose of device is to help people with disability in hand who cannot use a hand held mouse. Other such products available in market utilize head gear, which requires mounting on head and is also very costly. We take a detour from this approach to make such a device with targeted cost of Rs. 1000 per piece. To make this device we utilize principles of image processing, machine learning , computer vision and 3D modelling.

Principle of device

- A webcam with resolution of 640x480 captures image of person sitting in front at 30Hz.
- We process this image to obtain following data:
    a. L_f = Location of face - using frontal face *Haar features* as described in research paper Viola Jones[attached]. As Haar is CPU intensive process, we use *CAMShift* to speed up the process as described in paper CAMSHIFT[attached]
    b. L_e = Location of eyes - again using eye Haar features
    c. L_{eb} = Location of eyeballs - Using image processing : Searching for largest black blob inside the detected eye in step (b)
    d. Using data obtained in (a), (b) to derive face orientations in space
    e. Utilizing a neural network we determine gaze point of user on screen
    f. In one another approach we use a 3D model of face to determine the gaze point
    g. To implement control efficiently we utilize a bilevel coarse and fine control approach , where user moves head for coarse control and eyeball for for fine pixel to pixel control
    h. To use the camera in dark we fix 2 IR leds available in market near the camera and remove IR filter on it. As eyeballs reflect IR, it gets easily detected in dark too.

In our hardware setup, we use a 30 fps webcam with resolution of 640x480 pixels. In every image frame we repeat steps 1-6. Hence, the algorithm is required to process 30 images per second while keeping CPU usage to its minimal.
After obtaining image from webcam we approach the problem of determining cursor position in following steps:

1. Track user's face (**F**)(or dominant face if there are multiple),
2. Determine location of eyes(**E**)
3. Determine location of eyeballs(**EB**)
4. Estimate distance of face from screen(**d**) ,tilt angle of face(**T**)
5. Determine velocity vectors(**V**) of face and eyeballs
6. Estimate new cursor location using (**T,E,EB,V,d**) where **T,E,EB,V** are two dimensional vectors and **d** is a scalar



Step of algorithm

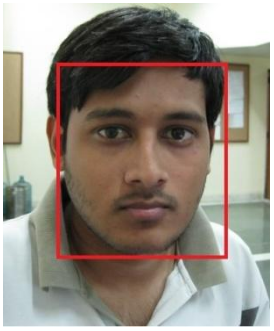
Result achieved

Background

- The aim of this project was development of a software module that would enable users to control the cursor according to eye movements.
- The first step in this project was using Image Processing to extract the eyeball in an given frame obtained from any image capturing source(in this case, a webcam).
- The library used for this task was OpenCV and the programming language used was C/C++.

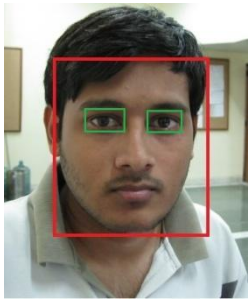
Face

- The first step in our algorithm was finding a human face in a given frame.
- Using HAAR feature extraction we extract the face in a every multiples of 30 frame and track it for rest of frames
- This method provides a faster way of recognizing faces as methods only relying on HAAR feature extraction are slower and more memory intensive.



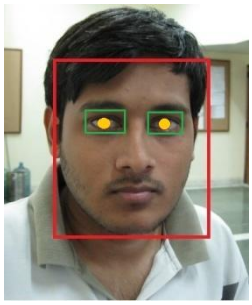
Eye

- After detecting the face in any given frame we analyze that specific part of the frame to look for eye using HAAR features and detect eyeballs

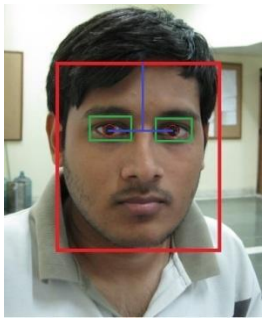


Eye Ball

- Once we detect the eyes, comes the part of the design where we have to detect the eyeballs. For this there exits no direct algorithm or known method as different people have differently approached this problem previously.
- First of all we separated the red channel from the detected regions of eyes as eyeballs stand out distinctively in the red channel.
- Then we binarised the image using a adaptive threshold which adjusts according to the lighting conditions.
- Once we have the binarised image we find the pixel having maximum number of black neighbors in an apt x apt rectangle where apt is equal to 12% of the size of the eye
- The pixel obtained is the center of the eyeball and around it we create a circle of radius aperture

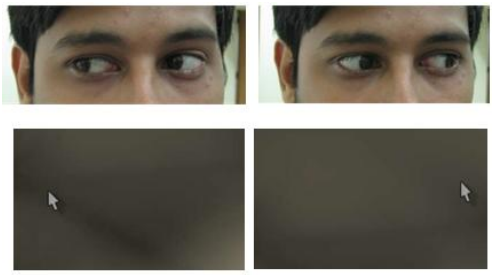


Head Tilt and Distance



- Finding out the tilt of the head and the distance form the screen also constitues an important part of our project since coarse movements are used to drastically navigate the mouse cursor.
- The method for finding the head tilt uses a simple but intuitive method.
- When the head is tilted sideways the distance between the center of eye and the edges of the face change as our webcam cam only produce a 2D representation of the image it is seeing.
- We exploit this property alongwith basic concepts of space geometry to find out the tilt of the head(as a ratio from the condition when he/she was directly facing the camera)

Co-ordinates



- Once we obtain the center of the eyeball we focus on controlling the mouse according to the eye movements.
- Here, we divide the eye movement into two categories for fine and coarse movements.
- Fine movements are those in which the eyeballs move without considerable difference in position of the eye. These are used to move the cursor slightly in the direction of movement of eyeball.
- When the eye moves, as a result of head movements the cursor moves a considerable distance according to a calibrated scale which is explained in the following lines.

GUI

- The frontend of this module i.e. the GUI consits of a profile manager which allows individual users to create their profiles and calibrates the known constants according to their faces.
- Based on the complexion and facial features like the size of their eye, it sets the variables used in our main module namely the threshold and aperture size,etc for obtaining eyeball and tilt of the head.
- When an users uses this profile manager he has the option to either load his profile(if he has cretaed one already) or create a new profile.
- If the user decides to create a new profile he/she is asked to sit straight and the face detected by our algoritm is shown to him.When the user confirms that his face is dectected correctly and in an upright position the calibration module creates his/her profile and stores the variables accordingly.

Training Module/Customization Module

- The Training Module helps calibrate the mouse movements.
- When a user is creating his profile he/she is asked to follow a constantly moving rectangle on the screen.
- This helps us to understand by how much does he/she move his head to change his area of focus.
- Depending on that our module shifts the cursor when he/she uses our module.

Mathematical description of algorithm

1.1 Location of Face

We use a combination of Viola-Jones face detection(or Haar classifiers)[4] and CAMShift tracking[5] algorithms to obtain face location vectors. Viola-Jones face detection has high accuracy but consumes a lot of CPU cycles. whereas CAMshift tracks given object in an image, and is designed to consume low number of CPU cycles[5]. In combined form Haar is used every *n+1* frames and its output face is tracked in next *n* frames using CAMShift :

1. Haar detection on first frame to obtain face of user which is saved as image $\mathbf{I}_f$.
2. Then CAMShift tracks $\mathbf{I}_f$ for next *n* frames. Let location of tracked face be **F,** where F represents a rectangle containing the face.
3. Repeat 1 and 2

Standard implementations of these functions can be found in OpenCV library (which has been used in our tests too). On an image of size 640x480 OpenCV implementation of Viola-Jones takes 150 ms average time to detect single face on an 1.8Ghz, Dual Core machine. Whereas our own OpenCV implementation of combined method takes only 14ms time per frame on same dataset. Thus, we can determine face location in 70 frames per second but accuracy drops from 97% to 95% (on a dataset of 1700 frames).

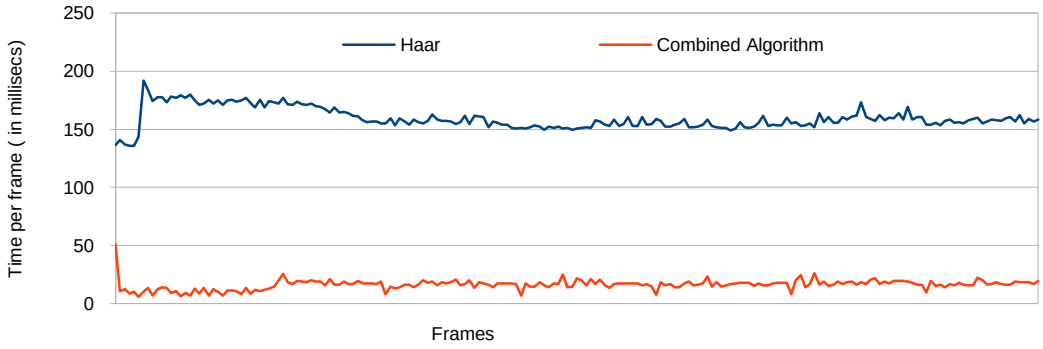


Figure 1: Time comparison of proposed method with Haar

1.2 Location of Eyes on face and face angle

Using Haar classifier for Eyes on image at **F,** we obtain location of both eyes. Let these locations be **E1** and **E2,** where **E1** represents rectangle containing eye 1 (x,y,w,h). The tilt angle of face can be expressed as :

$$\phi = \tan^{-1} \frac{(E1.y - E2.y)}{(E1.x - E2.x)} \qquad (1)$$

Also, the distance between two eyes :

$$\mathbf{d} = \sqrt[2]{(E1.y - E2.y)^2 + (E1.x - E2.x)^2} \qquad (2)$$

1.3 Location of Eyeballs

After location of both eyes, eye images $\mathbf{I_E}$ are processed to obtain location of eyeballs in following steps :

1. Obtain the skin color from face image $\mathbf{I_F}$ obtained in 2.1
2. Convert all skin colored pixels to white in $\mathbf{I_E}$
3. Apply threshold on $\mathbf{I_E}$ such that 12% of pixels are below threshold (approximate percentage sum of eyebrow and eyeball areas). Binarize $\mathbf{I_E}$ about this threshold to get $\mathbf{BI_E}$
4. As eyeball is darkest part it always comes below threshold and remains black.

In $\mathbf{BI_E}$, as eyebrows are thinner as compared to eyeball, (Figure 2) eyeball can be detected using a moving window of size approximately equal to eyeball. Eyeball size is estimated from size of eye obtained in 2.2. We use an O(kn) dynamic algorithm for moving window :

1. Construct a black moving window(**W**) of size $k \times k$ placed at (x,y) on $\mathbf{BI_E}$ where k is estimated diameter of eyeball
2. Calculate correlation of **W** with image pixels of $\mathbf{BI_E}$ using $C(n) = \sum_j^k . \sum_i^k BI_E(i+x, j+y)^2$ as **W** is zero everywhere
3. Store correlation for (x,y)th pixel in another image
4. Move window from (x,y) to (x+1,y). The correlation change can be calculated by looking at xth column of previous and x+k+1th column of recent window: $C(n+1) = C(n) - \sum_j^k BI_E(x, j+y)^2 + \sum_i^k BI_E(x+k+1, j+y)^2$
5. Determine position with highest correlation. This will be the location of eyeball.

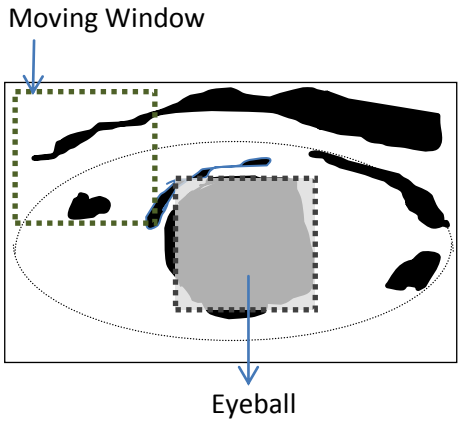


Figure 2: Largest black round blob detection using moving window

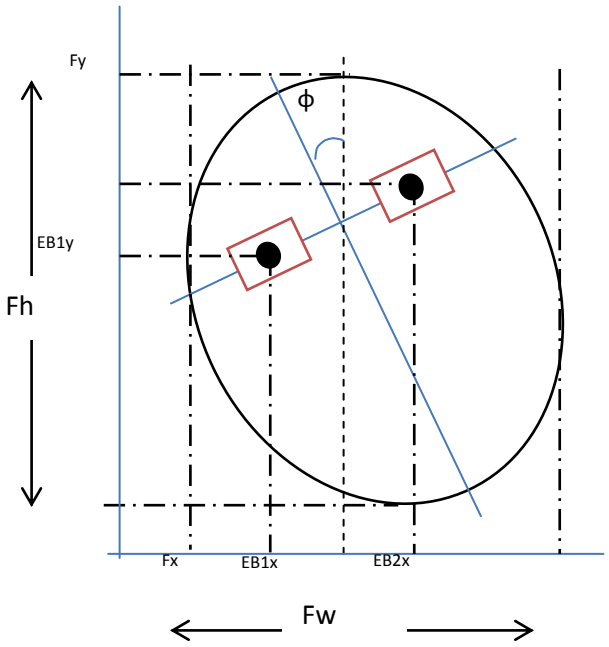


Figure 3: Frontal View of face after eye and eyeball detection

1.4 Cursor control methodology

Once we have obtained locations of eyeballs, eyes, face and tilt angles of face, we estimate movement in head and eye among frames. We divide movements into two categories, *coarse* and *fine*, where *coarse* movements are determined by changes in location and orientation of head whereas *fine* movements result as location of eyeball relative to head changes. If cursor location is $C(x, y)$, then $\Delta C(x, y)$ is determined using a geometric 3D head-eye model as shown in figure below.

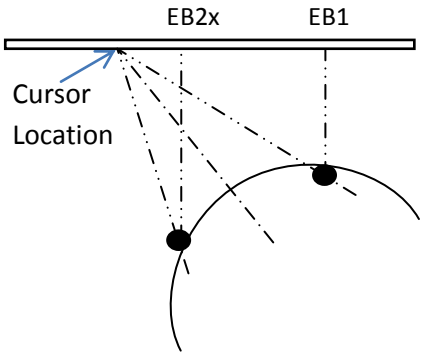


Figure 4: Top View of head model

2. IMPLEMENTATION

Using OpenCV we have implemented above described algorithms. Our system controls cursor in real time with 20% of CPU usage on 1.8Ghz Core 2 Duo machine and accuracy of control upto 2 dpsi (dots per square inch) in coarse control and 4 dpsi in fine control.

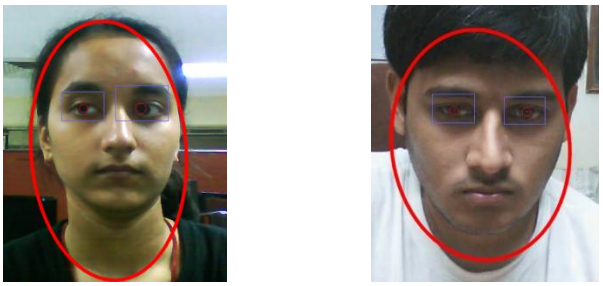


Figure 5: Eyes and eyeballs detected from webcam images

3. FUTURE WORK

As of now our system is able to control cursor with very high speed and low CPU usage. But its precision is very low. The system can be used for pressing 8 buttons on a screen using applications such as Gazetalk. But we need to improve on the precision to enable it for usage as a normal mouse. We have following ideas for further improvement in precision:

- Use camera with less noise
- Use better machine training model which permits for learning based on feedback. If possible an optimized implementation of a Backprop neural network will be better than using a 3D head model

For any information on the project or to obtain the codes contact at apurva@iitk.ac.in or rabisg@iitk.ac.in or meghaa@iitk.ac.in

4. REFERENCES

[1] Marc Eaddy, Gábor Blaskó, Jason Babcock, Steven Feiner, My Own Private Kiosk: Privacy-Preserving Public Displays. *3rd International Semantic Web Conference* (ISWC2004)

[2] Arne John Galenstrup, "How is eye-gaze interface control different", http://www.diku.dk/hjemmesider/ansatte/panic/

[3] David B¨ack , "Neural Network Gaze Tracking using Web Camera" , *LiTH-IMT/MI20-EX--05/414—SE*, Linköpings universitet

[4] Paul Viola, Michael Jones, Robust Real-time Object Detection, In *2nd international workshop on Statistical and Computational Theories of Vision* (2004)

[5] G.R. Bradski, Computer video face tracking for use in a perceptual user interface, Intel Technology Journal, Q2 1998

[6] BABCOCK, J. S., AND PELZ, J. B. 2004. Building a lightweight eyetracking headgear. In *Proceedings of the 2004 symposium on Eye tracking research & applications, ACM*, San Antonio, Texas, 109-114.

[7] Cohen, A. S. (1983). Informationsaufnahme beim Befahren von Kurven, Psychologie für die Praxis 2/83, Bulletin der Schweizerischen Stiftung für Angewandte Psychologie