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

ABSTRACT

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. 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.

KEYWORDS: Mouse, Computer Vision, Image Processing, Machine learning.

1. INTRODUCTION

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. 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).

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 real scalar.

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.

2. ALGORITHM

2.1 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).

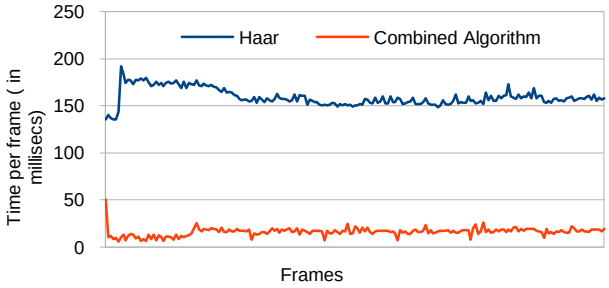

Figure 1: Time comparison of proposed method with Haar

2.2 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 :

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

2.3 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

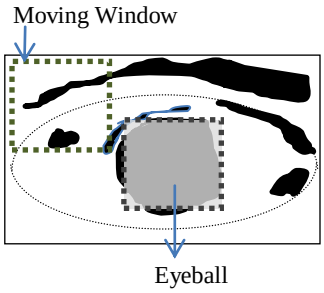

Figure 2: Largest black round blob detection using moving window

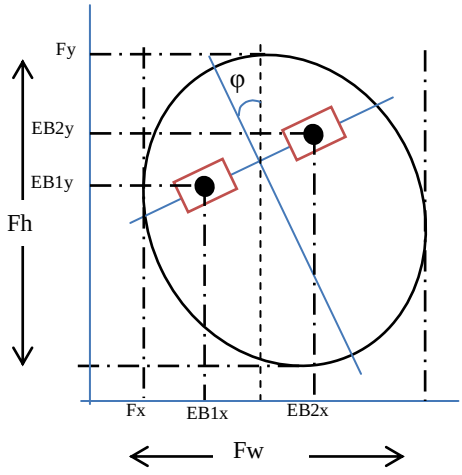

Figure 3: Frontal View of face after eye and eyeball detection

2.4 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. Due to space constraints complete equations of model cannot be described here.

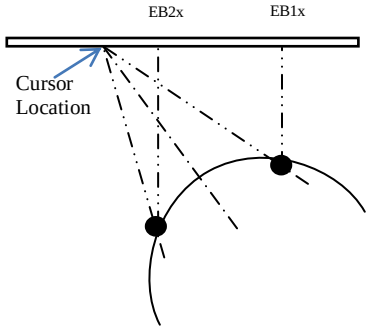

Figure 4: Top View of head model

3. 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 4 dpsi (dots per square inch) in coarse control and 8 dpsi in fine control.

Figure 5: Eyes and eyeballs detected from webcam images

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