Human Computer Interaction via Head and Eye movements

Apurva Gupta, Rabi Shanker Guha, Megha Nawhal

IIT Kanpur

Introduction

Social Problem

Disability

A person with physical disability in hand cannot use traditional methods of interaction with computers such as keyboard, mouse. Similarly, ALS patients have full body paralyses except in facial and eye muscles. Hence we are presented with requirement of a device which works using eye and head movements.

Cost factor

Available Eye tracking devices cost in lakhs of rupees as they use high accuracy cameras and proprietary algorithms. We present a method to construct an open source eye mouse utilizing only a web camera, infrared sensors and open programming libraries. Estimated cost of device construction is below Rs.1000 and even lower for mass manufacture.

Technical Problems

Detection

Detection of head, eye, iris in real-time from a low resolution camera. Accurate positioning of cursor in real-time requires processing 30 frames or $2x10^8$ bits/second. Since, mouse must work as a background process and not hamper other programs, CPU usage is to be kept at minimum(<10 %).

Learning

Conversion of data obtained to cursor location. Popularly used method is to use a machine learning technique which learns relationship between eye movement, head pose and cursor position based on training data (TD). TD is user dependent and its accuracy determines precision of eye mouse. Hence for each user separate training and high precision camera is required.

Previous work

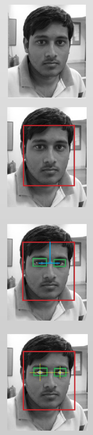
Our work builds on previous attempts of ITU to build a tracker which achieved a typing rate of 6 words per minute using only gaze tracking. But as shown in picture, they required a head gear (and a backpack with equipment for heavy image processing) to be mounted on user. This head gear computes face angles and face distance which we have estimated only from face images hence eliminating requirement of head gear.

Algorithm

Real time face and Eye detection

1. Location of Face

We use a combination of Viola-Jones face detection and CAMShift tracking algorithm 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. In combined form Haar is used every *n+1* frames and its output face is tracked in next *n* frames using CAMShift.



2. Location of Eyes on face and face angle

Using Haar classifier for eyes on image obtained above , we obtain location of both eyes. These locations are used to compute the tilt of face. Distance between eyes is used to estimate distance of face relative to initial distance

3. Location of Iris

$\mathbf{I_E}$'s are processed to obtain location of Iris in following steps. Obtain the skin color from face image $\mathbf{I_F}$. Convert all skin colored pixels to white in $\mathbf{I_E}$. Apply threshold on $\mathbf{I_E}$ such that 12% of pixels are below threshold (approximate percentage sum of eyebrow and iris areas). Binarize $\mathbf{I_E}$ about this threshold to get $\mathbf{BI_E}$. As iris is darkest it always comes below threshold and remains black. In $\mathbf{BI_E}$, as eyebrows are thinner as compared to eyeball, eyeball can be detected using a moving window of size approximately equal to iris. Iris size is estimated from size of eye.

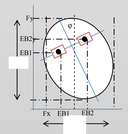
Iris detection and bi-level cursor control

4. Iris Detection

We use an O(kn) dynamic algorithm for moving window. Construct a black moving window $\mathbf{W}$ of size $k \times k$ placed at (x,y) on $\mathbf{BI_E}$ where k is estimated diameter of eyeball. Then calculate correlation of $\mathbf{W}$ with image pixels of $\mathbf{BI_E}$ using: $C(n) = \sum_i \sum_j BI_E(i+x, j+y)^2$

as $\mathbf{W}$ is zero. Store correlation for $(x,y)^{th}$ pixel in another image. Move window from (x,y) to (x+1,y). The correlation change can be calculated by looking at x^{th} column of previous and $x+k+1^{th}$ column of recent window:

$$C(n+1) = C(n) - \sum_j BI_E(x, j+y)^2 + \sum_j BI_E(x+k+1, j+y)^2$$



Determine position with highest correlation. This will be the location of Iris.

5. Bi-level cursor control

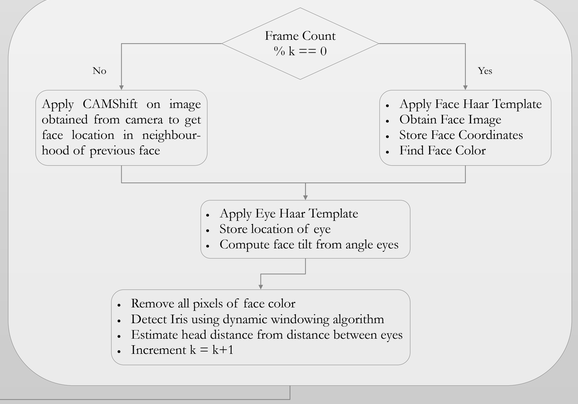
Other eye-mouse use a neural network to obtain gaze location using head and eye coordinates which requires different training for each person. To calculate cursor location, real time head and relative eyeball locations are scaled and compared with their maximum possible y and x coordinates. Head movements are given higher weight than eyeball. This allows user to have coarse and fine control with head and eyeball movements.

Design

Setup

Cognitive learning

When we first start using a normal mouse with hands, it takes some time for brain to familiarize with motor movement patterns required to control cursor in a desired manner. Approach used in other eye-mouse is to follow gaze location of user. But we divulge from this and in our design user learns over time, various head and eye movements required to control the location. As our brain can learn more complicated patterns than a machine, it makes eye mouse more easy and intuitive to use and saves computation time too.

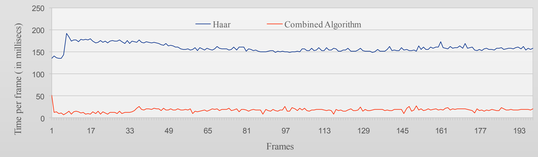


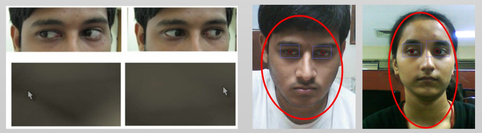
Hardware Requirements

- 30 Hz webcam, resolution 3K
- 2Ghz CPU, 512 MB RAM
- IR emitters

Rip the IR filter off the webcam. This will help in detecting Iris in night as it lights up in IR light

Implementation and Results





Improvement in time

A standard implementation of Haar Face detection from OpenCV vs. combined tracking time of Eye+Face in our algorithm. 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 (on a dataset of 1700 frames). But control is still realtime, since we have 30 datapoints per second and human reaction time is greater 100ms. 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.

Conclusion and Future Work

Available systems for head and eye based interaction are either intrusive in nature or too CPU intensive to be executed as a background process. As of now our system is able to control cursor with high speed and low CPU usage. But its precision is still 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 lesser noise and improve on IR resolution
- Feedback analysis to remove frames with error in detection
- Follow standard API. This will allow other software to interface with eye-mouse and use cursor data as from a normal mouse.