A Novel Method of Head Tracking Cursor Control System Using Open CV

Nikhil.K, Tripty Singh, Mithun R
Department of Computer Science
Amrita School of Engineering Bengaluru, India
Email: nikhilkoyyodan@gmail.com, tripty_prashant@yahoo.in, mithunchembra@gmail.com

Abstract—This project aims at controlling the cursor moving on the desktop by monitoring head movements. Imaging is done by a camera fixed at a key position, which is used to determine the relative shift in the head position of the user. Once the current visual orientation of the user is determined, the calculation involves conversion from the image space to 3D Cartesian co-ordinates. A led head gear is attached to the person head to track the movements of the head which may contain four leds initially we will set a calibration position and based on the movements we can exactly track in which direction motion of the head occurred and based on that we will simulate cursor movements by comparing adjacent frames, and we can perform various mouse operations like left click, right click, navigation based on blink of the eye and head movements.

Index terms: computer vision, Human computer interaction, Head tracking, Open cv

I. INTRODUCTION

Head tracking and movement based interaction using computer vision techniques have the potential to become an important component in perceptual user interfaces. The main intention of this work is to help people who are quadriplegic and non verbal (for example from cerebral palsy, traumatic brain injury or stroke) who have limited voluntary motions. Some people can move their head, some can blink or wink voluntarily while some can move only, say, tongue. Family, friends, and other care providers usually detect these motions visually.

Many computer access methods have been developed to help people who are quadriplegic: external switches, devices to detect small muscle movements or eye blinks, head pointers,

Infrared or near infrared camera based systems to detect eye movements, electrode based systems to measure the angle of the eye, even systems to detect features in EEG. These have helped many people access the computer and have made tremendous improvements in their lives. Still, there are some with no reliable means to access the computer. It will be helpful if it is possible to develop computer vision systems that is economical and will work under normal lighting to provide computer access to people who are quadriplegic and nonverbal.

II. LITERATURE SURVEY

Presently people use optical mouse and touch pads for navigating through personal computers and laptops. But if these devices experience a technical break down then there is no way we could do the navigation on the system. Keyboards may prove to be inefficient in this case as we are highly dependent on these tools. We could operate a system whose keyboard is Malfunctioning with a mouse but the other way around is really not easy. It is also not easy for elders and disabled persons to use the mouse.

Much research effort has been expended on locating and tracking heads and recognizing pose and facial expressions from video. Researchers have also explored the use of head tracking to help disabled people to move the cursor of a computer in the desired direction.

James Gips, Margrit Betke, and Peter Fleming developed the Camera Mouse [2]. They proposed a system that uses a camera to visually track the tip of the nose or the tip of a finger or some other selected feature of the body and move the mouse pointer on the screen accordingly. The system involved two computers: the vision computer, which does the visual tracking, and the user’s computer, which runs a special driver and any application software the user wishes to use.

Margrit Betke, James Gips, Peter Fleming proposed the visual tracking algorithm [3], based on cropping an online template of the tracked feature from the current image frame and testing where this template correlates in the subsequent frame. The location of the highest correlation was interpreted as the new location of the feature in the subsequent frame.

3D head tracking using non-linear optimization by James Paterson, Andrew Fitzgibbon [4], addresses the
specific problem of model-based tracking with a generic deformable 3D head model. A collection of head models is obtained from a 3D scanner, registered and parameterized to give a generic head model which is linearly parameterized by a small number of parameters.

Head tracking by glasses detection was carried out by Klaus Voss, Homero V. Rios Figueroa, Joaquin Pena Acevedo[5], which was based on glasses detection and does not require camera calibration. The method provided five degrees of freedom with head tracking. It provided the relative position of the head from the camera and its rotation.

An improved technique for 3D head tracking under varying illumination was proposed by Marco La Cascia, Stan Sclaroff, and Vassilis Athitsos [6], where the head is modeled as a texture mapped cylinder. Tracking is formulated as an image registration problem in the cylinder's texture map image. The resulting dynamic texture map provides a stabilized view of the face that can be used as input to many existing 2D techniques for face recognition, facial expressions analysis, lip reading, and eye tracking.

Black and Yacoob [7] use a rectangular planar patch under Affine transformation as a face model. Similar patches are attached to the eyebrows and the mouth. They follow the movements of the underlying facial patch but detect differential movements of their facial parts. Affine motion, like any 2D model, has its limitations because it has no concept of self-occlusion occurring at the sides of the head and around the nose. Affine transformations also distort the frontal face image when they are used to model larger rotations.

Azarbayejani et al. [8] use feature point tracking projected on an ellipsoidal model to track the head position. Feature point tracking has the drawback that tracking fails when the feature points are lost due to occlusions or lighting variations. New feature points are acquired, but only at the cost of excessive error accumulation.

Jebara and Pentland [9] also use feature point tracking, but with automatically located head features like eyes and mouth corners. The 3D position of the feature points is estimated using a structure from motion technique that pools position information over the image sequence with an extended Kalman filter. The estimate of the feature point position is filtered using Eigen faces to restrict the measurements to match an expected facial geometry.

### III. RELATED WORKS

At present people use optical mouse and touch pads for navigating through personal computers and laptops. But if these devices experience a technical break down then there is no way we could do the navigation on the system. Keyboards may prove to be inefficient in this case as we are highly dependent on these tools. We could operate a system whose keyboard is malfunctioning with a mouse but the other way around is really not easy. It is also not easy for elders and disabled persons to use the mouse.

In the proposed system overcomes the difficulty of the existing system by making use of Human-Computer Interaction (HCI) and Computer Vision (CV) which has proven to be a milestone in the field of science and technology.

### IV. PROPOSED MODEL

1) Acquiring images done by a camera fixed at a key position.
2) Continuous images are taken at a rate of 10 fps and the head will be detected by the presence of LED spectacles.
3) Head movements are tracked to detect and determine the relative shift in the head position. The movement of the head which is detected will provide the data for calculating the spatial coordinates and the cursor is controlled for movement accordingly.

### V. IMPLEMENTATION OF HEAD TRACKING ALGORITHM

```
  Image capture
   ↓
 Splitting channels → R,G,B Colour space
   ↓
 Image enhancement → Histogram based technique
   ↓
 Image smoothing → Gaussian blurring
   ↓
 Thresholding → Grey scale value
   ↓
 Image scaling → Interpolation technique
   ↓
 Edge detection → Canny edge detection
   ↓
 Contour detection → Approximation technique
   ↓
 Hough transform → Circle detection
   ↓
 Simulate cursor movement
```

---

ISSN (Print): 2278-5140, Volume-3, Issue-2, 2014
Image capture

A webcam is a video camera that feeds its image in real time to a computer or computer network. Unlike an IP camera (which uses a direct connection using Ethernet or Wi-Fi), a webcam is generally connected by a USB cable, FireWire cable, or similar cable. The laptop integrated web camera with a 0.3 MP (VGA 640x480) sensor, which has all the camera drivers built in into the Operating System (Ubuntu).

Splitting channels

Images are stored in memory in various different colour spaces. A grayscale picture just needs intensity information how bright is a particular pixel. The higher this value, greater the intensity. Current displays support 256 distinct shades of gray, each one just a little bit lighter than the previous one. So for a grayscale image, all that is needed is one single byte for each pixel. One byte (or 8-bits) can store a value from 0 to 255, and thus it would cover all possible shades of gray.

The actual colour data of an image is stored as arrays of values, known as channels. There are several kinds of channel modules, such as RGB, CMYK and HSV. Among them, RGB is the most widely used model, in which any pixel of a colour image is represented by its red, green and blue components.

Sometimes, the information we are interested in resides on a single colour channel only. In this case, we can consider splitting the colour channels from the original image and process the colour channels separately. If a user wants to extract an object whose colour is mainly green from its background, they can break the input image into its RGB channels and then operate on the green channel. Since only the green colors in the image contain information, we can totally remove all other tones. Green tones appear as white (shades of light gray, more towards white) in the Green channel and black (shades of dark gray) in other channels. Thus after removing the unwanted dark areas, we split the image into its channels and use only the RED channel from now on. After processing the colour channels, we can opt to recombine them to get a new, processed colour image.

There are times when it is not convenient to work with a multichannel image. In such cases, we can use split() in OpenCV[14] to copy each channel separately into one of several supplied single-channel images.

Image enhancement

The principal objective of image enhancement is to process an image so that the result is more suitable than the original image for specific applications. Image enhancement is basically improving the interpretability or perception of information in images for human viewers and providing better input for other automated image processing techniques. The principal objective of image enhancement is to modify attributes of an image to make more suitable for a given task and a specific observer. During this process, one or more attributes of the image are modified. The choice of attributes and the way they are modified are specific to a given task. Moreover, observer-specific factors, such as the human visual system and the observer’s experience, will introduce a great deal of subjectivity into the choice of image enhancement methods.

Histogram processing is the act of altering an image by modifying its histogram. The histogram is a graph showing the number of pixels in an image at each different intensity value found in that image[13]. The Histogram of digital image with the intensity levels in the range [0,L-1] is a discrete function.

\[ h(r_k) = n_k \]

where

- \( r_k \) is the intensity value.
- \( n_k \) is the number of pixels in the image with intensity \( r_k \).
- \( h(r_k) \) is the histogram of the digital image with Gray Level \( r_k \)

Histograms are frequently normalized by the total number of pixels in the image. Assuming a \( M \times N \) image, a normalized histogram.

\[ p(r_k) = k/\text{MN} \]

where

- \( k=0,1,2,3,\ldots,\text{L-1} \) is related to probability of occurrence of \( r_k \) in the image.
- \( p(r_k) \) gives an estimate of the probability of occurrence of gray level \( r_k \).

The sum of all components of a normalized histogram is equal to 1. Histograms are simple to calculate in software and also lend themselves to economic hardware implementations, thus making them a popular tool for real-time image processing.

Brightness/Contrast enhancements

Contrast enhancements improve the perceptibility of objects in the scene by enhancing the brightness difference between objects and their backgrounds. Contrast enhancements are typically performed as a contrast stretch followed by a tonal enhancement, although these could both be performed in one step. A contrast stretch improves the brightness differences uniformly across the dynamic range of the image, whereas tonal enhancements improve the brightness differences in the shadow (dark), midtone (grays), or highlight (bright) regions at the expense of the brightness differences in the other regions.

Contrast stretch
A high-contrast image spans the full range of gray-level values; therefore, a low contrast image can be transformed into a high-contrast image by remapping or stretching the graylevel values such that the histogram spans the full range. The contrast stretch is often referred to as the dynamic range adjustment (DRA). The simplest contrast stretch is a linear transform that maps the lowest gray level GLmin in the image to zero and the highest value GLmax in the image to 255 (for an eight-bit image), with all other gray levels remapped linearly between zero and 255, to produce a high-contrast image that spans the full range of gray levels.

This linear transform is given by
\[
g'(x, y) = \text{INT}\left\{\frac{255}{\text{GLMAX} - \text{GLMIN}}[g(x, y) - \text{GLMIN}]\right\} \quad \text{........... (3)}
\]
where the INT function returns the integer value.

Image smoothing

Smoothing, also called blurring, is a simple and frequently used image processing operation. Smoothing is also important when we wish to reduce the resolution of an image in a principled way. There are many reasons for smoothing, but it is usually done to reduce noise or camera artifacts or to produce a less pixelated image. Image smoothing is a key technology of image enhancement, which can remove noise in images. So, it is a necessary functional module in various image-processing software.

Gaussian Blurring

A Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the Bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales.

Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function. This is also known as a two-dimensional Weierstrass transform. By contrast, convolving by a circle (i.e., a circular box blur) would more accurately reproduce the Bokeh effect. Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian blur has the effect of reducing the image’s high-frequency components; a Gaussian blur is thus a low pass filter.

The equation of a Gaussian function in one dimension is
\[
G(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-x^2/2\sigma^2} \quad \text{.................. (4)}
\]
We have also assumed that the distribution has a mean of zero (i.e. it is centered on the line x=0).

Thresholding is a non-linear operation that converts a gray scale image into a binary image where the two levels are assigned to pixels that are below or above the specified threshold value. Thresholding may be viewed as an operation that involves tests against a function \( T \) of the form
\[
g(x, y) = \begin{cases} 
1, & \text{if } f(x, y) > T \\
0, & \text{if } f(x, y) \leq T
\end{cases} \quad \text{.................. (5)}
\]
Where \( f(x, y) \) is the gray level at the point \((x, y)\). In image processing, thresholding is used to split an image into smaller segments, or junk, using at least one color or gray scale value to define their boundary. A possible threshold might be 40% gray in a gray scale image: all pixels being darker than 40% gray belong to one segment, and all others to the second segment. It’s often the initial step in a sequence of image-processing operations.

Image scaling

Image scaling is the process of resizing a digital image. Scaling is a non-trivial process that involves a trade-off between efficiency, smoothness and sharpness. With bitmap graphics, as the size of an image is reduced or enlarged, the pixels which comprise the image become increasingly visible, making the image appear “soft” if pixels are averaged, or jagged if not. Image size is most commonly decreased (or subsampled or down sampled) in order to produce thumbnails. Enlarging an image (up sampling or interpolating) is generally

Common for making smaller imagery fit a bigger screen in full screen mode, for example. In zooming a bitmap image, it is not possible to discover any more information in the image than already exists, and image quality inevitably suffers. However, there are several methods of increasing the number of pixels that an image contains, which even out the appearance of the original pixels. Down sampling and up sampling are two fundamental and widely used image operations, with applications in image display, compression, and progressive transmission. Down sampling is the reduction in spatial resolution while keeping the same two-dimensional (2D) representation. It is typically used to reduce the storage and/or transmission requirements of images. Up sampling is the increasing of the spatial resolution while keeping the 2D representation of an image. It is typically used for zooming in on a small region of an image, and for eliminating the pixelation effect that arises when a low-resolution image is displayed on a relatively large frame.

Edge detection

The purpose of edge detection in general is to significantly reduce the amount of data in an image, while preserving the structural properties to be used for further image processing. Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in
pixel intensity which characterize boundaries of objects in a scene. Classical methods of edge detection involve convolving the image with an operator (a 2-D filter), which is constructed to be sensitive to large gradients in the image while returning values of zero in uniform regions. There is an extremely large number of edge detection operators available, each designed to be sensitive to certain types of edges.

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work. He was very successful in achieving his goal and his ideas and methods can be found in his paper, “A Computational Approach to Edge Detection”[15]. In his paper, he followed a list of criteria to improve current methods of edge detection. The first and most obvious is low error rate. It is important that edges occurring in images should not be missed and that there be no responses to non-edges. The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to have only one response to a single edge. This was implemented because the first 2 were not substantial enough to completely eliminate the possibility of multiple responses to an edge.

OpenCV has a Canny edge detection algorithm based on “A Computational Approach to Edge Detection” [16].

Contour detection
People can differentiate between various objects by observing their shape, colour, texture and features. To impart this intelligence into a system, it needs to implement techniques like contour detection to help the system in recognizing the shapes of objects. The term contour can be defined as an outline or a boundary of an object. Contour detection deals with detecting various objects in an image/video specifically. Use of contour detection in image processing is to locate objects and their boundaries in images/videos. It helps in determining the shapes of various objects while differentiating them from each other and also helps in detecting the motion of objects while in a video. It is used as a preliminary step for corner detection as it improves the accuracy of the detected corners.

Hough transform
The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. It is used for circle detection.

VI METHOD TO SIMULATE CURSOR FUNCTIONS
A mouse performs many functions such as:
- Cursor movement
- Click-Drag to select
- Left Click
- Right Click

We develop a method of logical simulation of these functions using the data obtained from the head tracking algorithm mentioned previously. The cursor is controlled by translating the head movements. Left and Right clicks are performed based on other gestures. Due to ease of usage we use the blinking of the eye. A left/right blink will be detected and used to trigger a left/right click respectively. If both eyes blink at the same time, it should not trigger any events, must be over-ridden.

The head tracking algorithm eventually calculates the pixel co-ordinate positions of the four corners enclosing the Region of Interest. These values form the input data and are necessary for further processing.

From the previous sections we infer that the corners are actually the centres of the four circles detected using Hough circle transform. These values are stored contiguously in an array. (0,0)

VI.1 REGION OF INTEREST
An image consists of various objects or regions, of which only few will contain the information required for a certain image processing application. Such an area is called a Region of Interest. Defining a proper Region of Interest prior to the processing tasks is highly recommended. It reduces the time required to process and also improves efficiency.

In our method, all the information is contained in the area bounded by the corners. Thus our Region of Interest is the polygon formed by joining these four points.
VI.2 EXTRACTING THE REGION OF INTEREST

The four corners have to be sorted based on position. To do this, first the centroid (mass center) must be calculated, which is the average of the x and y coordinates of the corners.

\[ M(\text{Xc}, \text{Yc}) = \frac{(x + y)}{2} \]  
where,

- \( M(\text{Xc}, \text{Yc}) \) represents the Mass center given by taking the average of x and y coordinates of corners.

Now, with the centroid as origin, a 2D Cartesian plane is visualised. The corners are sorted as Top right, Top left, Bottom right and Bottom left based on the quadrant in which they lie. It can be expressed mathematically as:

If \( x < \text{Xc} \) and \( y < \text{Yc} \) then corner is: Top Left
If \( x > \text{Xc} \) and \( y < \text{Yc} \) then corner is: Top Right
If \( x < \text{Xc} \) and \( y > \text{Yc} \) then corner is: Bottom Left
If \( x > \text{Xc} \) and \( y > \text{Yc} \) then corner is: Bottom Right

The ultimate goal is to define the region of the image containing the eyes. Thus we further shrink the RoI, as follows:

First we calculate the length of the left and right edges. Mathematically it is the distance between the two corner points. Top Left and Bottom Left corners define the left edge and the other two define the right edge. The lengths are determined using the formula

\[ D = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2} \]

Now we calculate a point on the, say, left edge that is at a distance 0.6*length measured from the Bottom Left corner. It is determined by calculating the slope of the edge and offsetting the co-ordinates. This point is taken as the new Bottom Left corner and is updated. Same is done with the right edge.

It is seen that, now the RoI encloses only the part of the image containing the eyes, with the newly defined regions consisting of one eye each.

VI.3 BLINK DETECTION

The fillPoly() function of OpenCV is used to create two binary image, each consisting of one half of the split RoI. These images are used to create masks for extracting the required regions from the image.
the chosen threshold level is optimum, then merely counting the number of white pixels in the images and comparing them will yield the required information. The corresponding click function is simulated.

VI.4 CURSOR MOVEMENT

To move a cursor to a new point in Cartesian system, the co-ordinates of the destination is sufficient. But calculating the destination co-ordinates is done using polar geometry. The corners and the calculated mass center of the RoI polygon are used here.

The direction and angle of movement is determined by the relative shift in the position of the mass center. The distance moved by the mass center along with the direction of movement is calculated every frame. Non-zero differences indicate that there is a shift from the previous position, and should trigger cursor movement.

If \((X_c,Y_c)\) is the current mass center position and \((X_{c1},Y_{c1})\) is the previous position, then the distance moved is given by:

\[
R = \sqrt{(X_c - X_{c1})^2 + (Y_c - Y_{c1})^2} \quad \cdots \cdots \cdots \cdots \cdots (8)
\]

The angle is calculated as the inverse tangent of the slope of the line joining \((X_c,Y_c)\) and \((X_{c1},Y_{c1})\):

\[
\theta = \tan^{-1} \left( \frac{Y_c - Y_{c1}}{X_c - X_{c1}} \right) \quad \cdots \cdots \cdots \cdots \cdots (9)
\]

The irregularities arising due to negative and zero denominators are taken care of. Now the next cursor position is determined by offsetting the previous position by a suitable function of \(r\) and \(\theta\). The method employed is:

VI.5 DESIGNING THE HEAD GEAR

On the drawing board, the purpose of the headgear was to provide an easily detectable feature whose shift in position and shape could only be brought about by the change in orientation of the user’s head, and remain closely associateable to it too. A basic study of images obtained from a webcam revealed two obvious choices: One was to exploit the fact that skin tones have a narrow hue range. This means a contrasting colour, say green, is easily differentiable from the tones of the face and head on almost all people. The other was that very dark and very light areas can be easily separated from the rest of the image. One way to accomplish that is by a simple thresholding process.

After a few weeks of testing, we realised that the latter was more prominent and showed consistent results. Since it is not practical to induce a controlled dark region, the first and the best thought was to use a source of light as a marker. This made us consider the prospects of using tiny LED bulbs that would produce small clear-cut high-intensity light spots. We required four of them to describe the corners of our region of interest.

Normal LEDs failed to meet our intensity requirement; an indistinguishable faint glow does not fit the description of an ambience-independent track-able feature. Thus we upgraded to high power LEDs. These are bigger and more power packed. This increase in diameter led to a problem: more lens flare. We had to soften it up by introducing a translucent filter, which would evenly distribute the light over the cross section.

Thermocol beads seemed ideally suited for the job. The results achieved are shown below.

![Fig 8 Removing lens flares (before, after)](image)

To power the LEDs, a supply of 3V was required. As they do not draw a high current, four LEDs can be easily connected in parallel and fed by a sine power source. A pair of serially connected AA or AAA sized alkaline batteries is sufficient to power this set up. The circuit diagram and the implemented connection topology are presented.

![Fig 9 Circuit diagram to power the led’s](image)
VII TESTS AND RESULTS
The figure on the right is the input image, and below is its matrix. It is a scaled down version of its matrix representation. The scaling down does not change the positional distribution pattern of the intensity values, the appreciation of which is our primary intention. We see that the high intensity values are grouped into four regions around the LEDs.

The brightness/contrast adjustment. Sliders are visible at the top of the image window, which can be used to adjust the brightness and contrast, depending on the ambient light.

Thresholding. The slider is used to fine tune the thresholding value in real time compensating for various lighting conditions.

VIII CONCLUSION & FUTURE WORKS
The method described here is a potentially viable alternative to the current generation of computer control peripherals. It is not a perfect system yet. Improvements are necessary in many areas to implement it at the OS level as a replacement for the mouse. In such a case more efficient algorithms are absolutely necessary in order to reduce the CPU overhead caused by the image processing functions. Any operating system, like Windows, will also require corresponding Drivers to be written in order for the OS to communicate with the code.

Another area needing improvement is the blink detection, which must be made to adaptively compensate for different lighting conditions.

IX. REFERENCES


