Mouse Operations using Finger Tracking

Saniya Kadam
Information Technology
K.J. Somaiya College of Engineering
Mumbai, India

Nikita Sharma
Information Technology
K.J. Somaiya College of Engineering
Mumbai, India

Thanushree Shetty
Information Technology
K.J. Somaiya College of Engineering
Mumbai, India

Ravindra Divekar
Information Technology
K.J. Somaiya College of Engineering
Mumbai, India

ABSTRACT
Computers are the electronic brains of the era, technology's greatest gift to mankind. The functioning of these electronic brains is controlled by humans. Interaction between the computer and humans is possible with the help of various devices such as keyboard, mouse, joystick, light pen, track ball, barcode reader, etc. Of these devices, the mouse performs various crucial functions in the most user-friendly way. This paper conveys work to implement the same functionalities without the use of external, bulky devices. Hand gestures can be used for natural and intuitive interaction of the user with a computer [1]. This paper intends to replicate different hand gestures as mouse functionalities using image processing tools and techniques.

General Terms
cursor movements, hand gestures, morphological operations.

Keywords
finger tracking, , image processing, virtual mouse, contours, hulls and defects.

1. INTRODUCTION
In today's world various devices such as televisions, play stations, cell phones, and etc. use hand gestures to control different applications and improve ease of interaction with the user. It is often shown in movies that devices such as computers can be easily operated by simply moving fingers in the air freely [2]. Traditional devices are often inadequate in this context.

Mouse operations using finger tracking is an image processing application that recognises the user's finger movements and maps them to cursor movement. All functionalities provided by a physical mouse can be implemented using this application.

Hand gestures provide a naturally immersive and user-friendly way of interaction with the system. Hand gesture of the users are matched to mouse events in the system. This enables the user to operate the system from a distance. It also eliminates the need for a physical connection to perform the same functionalities like that of a mouse. Thus problems such as unavailability of mouse, physical damage to touchpads in laptops, weak connection of mouse to system, etc. can be eliminated.

2. LITERATURE REVIEW
Over the years, many different techniques have been used to accomplish the goals of finger tracking and hand gesture recognition to facilitate Human Computer Interaction.Some of these methods include recognition of gestures with the help of colours [8], [9], shapes [10], [11], and hand features [12], [13].

2.1Viola-Jones Algorithm
One particular technique for the recognition of the hand features provided in one of the literature is based on the Viola Jones algorithm. The first step is to identify a region of interest (ROI). This is done by converting the RGB image of the background containing the hand to HSV. Skin coloured object representing the user's hand can then be isolated by segmenting it using a range of values for skin colour. This segmented sub-image is used for further processing. The Viola-Jones algorithm is applied on the above mentioned sub-image. It identifies a set of Haar-like features at various stages over time. It then uses the concept of integral image to train the application to recognise accurate gestures. To improve real time performance, AdaBoost algorithm is also applied [1].

2.2 Online Training of Fingertip Colours
One more method of fingertip detection using a technique known as on-line training of fingertip colours requires the user to position his finger in a rectangular box that appears on the screen. The system traverses every pixel in the box, counts the number of occurrences of different colours in the frame and creates a colour histogram by normalization. As the RGB colour space is inherently dependent on brightness, image is transformed from RGB colour space to HSV colour space that separates pure colours from brightness and reduces the effect of illumination and similar background for learning the probability distribution of fingertip colours using the hue (H) and saturation (S) values. A 2D colour histogram is computed using these values. Each input pixel is mapped to an object pixel by searching the colour histogram. This process is called back-project [3]. Using back project calculation on the colour histogram the probabilistic distribution image I(x,y) is built. A binary image B(x,y) is obtained from I(x,y) through normalization[3]. The noises and holes present in the binary image are removed using morphological operations. The form is used (1) to eliminate noises and (2) to remove holes.

2.3 Finger tracking in embedded systems

In another paper the finger tracking method includes 6 steps: **RGB to YUV conversion**: The first step is to convert RGB image raw data to YUV data using the required equation which is suitable for 24 bit image. **Colour segmentation**: The pixels of the image are classified into skin colour and non-skin colour according to chrominance values using a skin-colour reference map in YCrCb colour space**. Density regulation**: It deals with removing the noise in the form of small holes such as edges or shadows [2]. To solve this, it performs dilation to fill the holes in the image and erosion to eliminate small objects in background. **Window searching:** Searching the whole frame for finger is very exhaustive and time consuming and also prone to error. To solve this problem the searching area is reduced to 60x60 pixels with the center of the window to be the last known finger position. The next position must not exceed 30 pixels in x,y direction. The minimum time for the movement should be no less than 0.4444s, which is appropriate for the real time case. Any fast movement requiring less time will fail to be detected **[**2**]. Finger tracking**: It assumes that there will be no other object in the background similar to fingertip. Fingertip is detected by searching O2(X,Y)=1 from top to bottom. **Click Recognition**: Different finger motions define various Clicking actions. The algorithm takes last 20 positions for the calculation.
If the 20 values matches the clicking motion, CLICK motion is detected [2].

3. THE PROPOSED SYSTEM

3.1 System Overview

The above proposed system will be executed as shown in fig.1

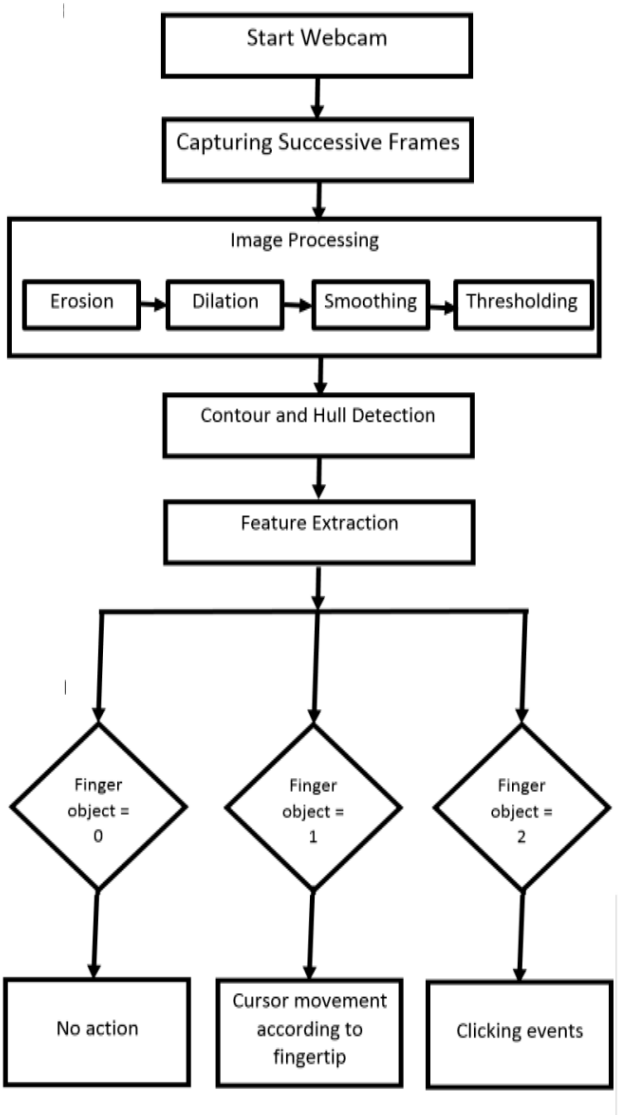

Figure 1: Flow Chart of Mouse Operations Using Finger Tracking

3.2 Explanation

The following is the brief explanations of the working principle of the various major blocks or sections used in the system

- **Start Webcam**

Figure 2 : Webcam Initialization

In the proposed system, the webcam is integrated to the application software using methods provided by C#. As soon as the application is started the integrated camera is initiated. For the application to execute continuously in real time, it is put into the

Idle state. Thus the webcam remains active throughout the process.

- **Capturing Successive Frames**
  As the application executes in idle state, the webcam continuously captures images. The image captured is stored in an object. This image moves on to further processing. The next image captured is overwritten on the same object and this process repeats as long as the application is working. These images captured by the webcam are displayed for the benefit of the user on a window in the application. The successive frame capture gives an illusion of a continuous video because of the speed of the capture. Thus the finger can be tracked continuously.

- **Image Processing**
  This is a crucial phase of the proposed system as precision of the finger tracking and all clicking events is achieved here. There are four steps to be followed
  - Erosion:

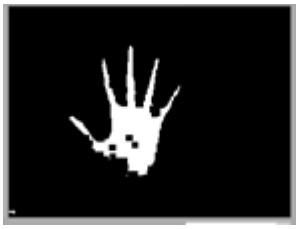

Figure 3 : Erosion

It is possible to remove noise in the captured images by the use of the morphology technique of erosion. Certain objects in the background may fall in the range of target skin colour [14] [15] and may be detected along with the palm of the user. Erosion will reduce the size of these unwanted detected blobs in the background and therefore reduce noise [3]. The erosion of the binary image A by the structuring element B is defined by:

$$A \ominus B = \{z \in E | B_z \subseteq A\} \quad [4]$$

  - Dilation:

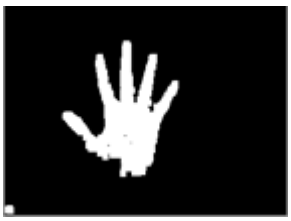

Figure 5 : Dilation

Erosion of the captured image sufficiently removes the noise, but it also reduces the size of the main blob which represents the user's palm. To bring it back to its original size for further manipulation and processing, the complementary morphological operation for erosion-dilation is used. This will sufficiently resize the blob. The dilation of A by the structuring element B is defined by:

$$A \oplus B = \bigcup_{b \in B} A_b \quad [5]$$

  - Smoothing:

Figure 6 : Smoothing

Once the blob has been resized, the next area of concern is its edges. Dilation is a binary morphological operation and the boundary of the resized blob therefore has ridges. It has a staircase like appearance also known as staircase noise. To remove this problem, smoothing operation is performed on the blob. The result is a blob with rounded boundary but the staircase noise is spread out to create the same. Smoothing is given as

$$s_0 = x_0$$
$$s_t = \alpha x_t + (1 - \alpha) s_{t-1}, \; t > 0$$

  - Thresholding:

Figure 4 : Thresholding

To create a precise boundary for the blob, thresholding is used. A threshold value is set and the entire image is divided into two segments either above or below the threshold. The smoothed boundary therefore transforms into a precise boundary around the blob. Thresholding can be best explained as:
If f (x, y) > T then f (x, y) = 0 else f (x, y) = 255 [6]

- **Contour and Hull Detection**

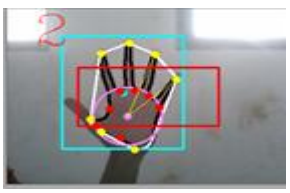

Figure 7: Plotting of contour, hulls and defects of blob

A **blob** is a region of a digital image in which some properties are constant or vary within a prescribed range of values; all the points in a blob can be considered in some sense to be similar to each other [7]. The system traverses through every pixel in the frame. As soon as it detects any pixel of the blob that is the contour pixel it saves the corresponding pixel value in a linked list. The sequential nodes of the linked list are used to store the following contour values of the same blob. Once all the contour details of one blob have been stored in one linked list, it is then connected to another linked list which stores the contour details of the next blob within the frame. By comparison of the linked list values the largest blob is then identified. This blob refers to the palm of the user.

The next step is to identify the hull and defect of the isolated blob. The hulls represent the finger tips and the defects are the lowermost points in the hollow between the two fingers.

- **Feature Extraction**

The defects are joined to create a circle that bounds the palm. The coordinates of the hull points and the coordinates of the center of the palm bounding circle are used to detect the number of fingers. First, the y-coordinate of a point 0.5 times the radius perpendicularly below the center is compared to check if it is more than the y-coordinate of each of the hulls. Then the distance between the hull point and the center of the palm bounding circle is calculated using the Euclidean distance formula. If this distance is more than 1.25 times the radius, that hull point is considered as an indication of an extended finger to perform certain gesture. The values (0.5 times the radius and 1.25 times the radius) are used to provide a margin of error associated with different users. A variable is used to store the number of fingers. This value cannot exceed 2 as the application uses at most 2 fingers to perform any action.

- If the number of fingers is 0, no action is performed and the cursor remains stationary.

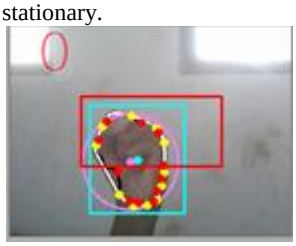

Figure 7 : No finger detected

- If the number of fingers is 1, the cursor moves in the direction of the motion of the user's finger.

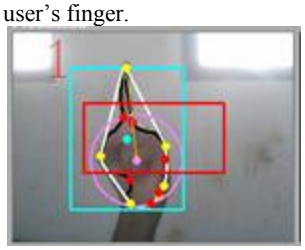

Figure 8 : One finger detected

- If the number of fingers is 2, clicking events are generated. A variable is incremented with each subsequent frame capture by the camera when the same position of the user's fingers is maintained. The value of this variable decides whether that gesture is a left click, right click, double click or drag and drop.

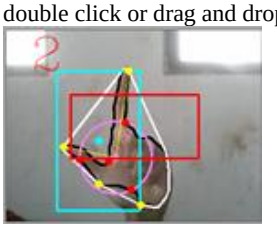

Figure 9 : Two fingers detected

Figure 10 : Left click system message

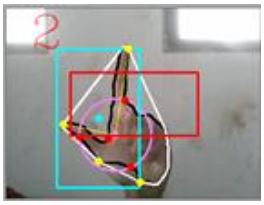

Figure 11 : Two fingers detected

Figure 12 : Double click system message

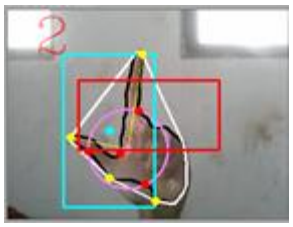

Figure 13 : Two fingers detected

Figure 14 : Drag and drop system message

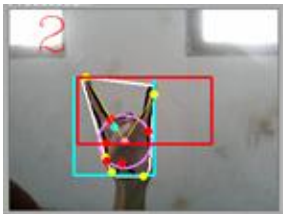

Figure 15 : Two fingers detected

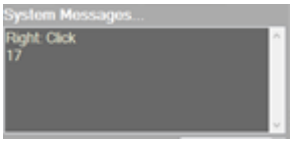

Figure 16 : Right click system message

3.3 Features

The Following are the prominent features of the proposed system

- Accurate fingertip identification
- Convenient to use
- Durability that ensures robustness
- Fast processing speed
- Wireless: Works with any good quality webcam.
- No cables, no additional hardware required
- Easiness and simplicity: Simple installation, intuitive use

4. SCOPE AND APPLICATIONS

The following are some applications defining the scope of the proposed system

- Best alternative for mouse
- Interactive gaming applicationPhoto browsing
- Controls for smart television
- Applications such as Paint and MS Office can be directly accessed using finger-tip tracking.
- Used in teaching environment
- In a public setup, where a system is accessed by many users, this application can reduce the risk of germ spread by avoiding contact with the system.

5. CONCLUSION

The mouse is one of the most commonly used input devices in a computer system and therefore requires frequent maintenance.The proposed system overcomes this drawback by completely eliminating the mouse and its related issues. It tracks the user's fingertip and implements it as cursor movements in the system using image processing concepts.Thus it enables the user to perform mouse functionalities effortlessly and efficiently. This application is in tune with the current technological trends and can also be incorporated in many other modern devices.

6. ACKNOWLEDGEMENT

The authors gratefully thank their guide Mr. Ravindra Divekar for his valuable contributions and support towards this research. They would also like to thank their college for providing the necessary platform for making this paper.

7. REFERENCES

[1] New Hand Gesture Recognition Method for Mouse Operations Ehsan ul haqI, Syed Jahanzeb Hussain Pirzadcl, Mirza Waqar Bailand Hyunchul Shin4

[2] A FINGER-TRACKING VIRTUAL MOUSE REALIZED IN AN EMBEDDED SYSTEM - Wai-Wah Martin Tsang and Kong-Pang Pun

[3] IMPLEMENTATION OF VIRTUAL MOUSE BASED ON MACHINE VISION - LI WENSHENG1, DENG CHUNJIAN2, LV YI3

[4] http://en.wikipedia.org/wiki/Erosion

[5] http://en.wikipedia.org/wiki/Mathematical_morphology

[6] http://www.cse.unr.edu

[7] http://en.wikipedia.org/wiki/Blob_detection

[8] 1. Yao, and 1. R. Cooperstock, "Arm gesture detection in a classroom environment," Proc. IEEE Workshop on Applications of Computer Vision, 2002, pp. 153-157.

[9] Y. Wu and T. S. Huang, "Non-stationary colour tracking for visionbased human computer interaction," IEEE Trans. on Neural Networks, vol. 13, no. 4,2002, pp. 948-960.

[10] A. Ramamoorthy, N. Vaswani, S. Chaudhury, and S. Banerjee, "Recognition of dynamic hand gestures," Pattern Recognition, vol. 36, 2003,pp . 2069-2081.

[11] E. Ong, and R. Bowden, "Detection and segmentation of hand shapes using boosted classifiers," Proc. IEEE 6th International Conference on Automatic Face and Gesture Recognition, 2004, pp. 889-894.

[12] K. Oka, Y. Sato, and H. Koike, "Real-time fingertip tracking and gesture recognition," Proc. IEEE Computer Graphics and Applications, vol. 22, no. 6, 2002, pp. 64-71.

[13] Z. Zhang, Y. Wu, Y. Shan, and S. Shafer, "Visual panel: Virtual mouse keyboard and 3D controller with an ordinary piece of paper," Proc. Workshop on Perceptive User Interfaces, 2001.

[14] Z. Zhang, Y. Wu, Y. Shan, and S. Shafer, "Visual panel: Virtual mouse keyboard and 3D controller with an ordinary piece of paper," Proc. Workshop on Perceptive User Interfaces, 2001.

[15] Zhang, H. Lin, M. Zhao, "A Fast Algorithm for Hand Gesture Recognition Using Relief' Sixth International Conference on Fuzzy Systems and Knowledge Discovery. 2009.