Vision Interfaces and Systems
Laboratory Technical Report
VISLab-94-001
October, 1994

FingerMouse: A Freehand Computer Pointing Interface
Submitted for partial fulfillment of the requirements for the degree of
Masters of Science in
The University of Illinois at Chicago

Thomas A. Mysliwiec
Vision Interfaces and Systems Laboratory (VISLab)
Electrical Engineering and Computer Science Department
The University of Illinois at Chicago
Chicago, IL 60607
ABSTRACT

FingerMouse is a freehand pointing alternative to the ubiquitous mouse. In this system, the user merely performs a pointing gesture above the keyboard. A down-looking camera is trained on the keyboard. A user typing at the keyboard may switch into the 'pointing mode' by simply assuming a hand pointing configuration above the keyboard. A vision system constantly monitors the hand and tracks the fingertip of the pointing hand. As the user gestures in a horizontal plane just above the keyboard, the screen cursor moves accordingly. The user depresses the SHIFT key on the keyboard with the non-pointing hand to register a 'mouse button press'.

This report presents a probabilistic color-based segmentation algorithm which transforms a color transfer function into probability entries in a RGB lookup table. It details a finite-state machine approach for the rapid detection of the pointing hand and describes a principal axis-based approach for fingertip tracking. The system has achieved near real-time performance on a general purpose workstation. Preliminary user testing results are presented which suggest that FingerMouse is a promising candidate as a general purpose mouse-type input device.
INTRODUCTION

The FingerMouse is a vision-based computer pointing device, designed to exploit the advantages of the mouse while improving on its drawbacks. The use of computer vision to detect hand shape and motion removes the constraints of existing glove devices. Additionally, the processing techniques provide a more general solution to the problem of tracking the human hand.

The Mouse and Other Input Devices

There have been numerous comparisons of computer input devices in the literature. Many comparisons include the mouse as one of the devices being tested. Wolf [14] compared stylus, keyboard, and mouse input devices and found the stylus to be preferable for applications which exploit previously learned actions. MacKenzie et.al. [9] compared a mouse, a stylus/tablet device and a trackball for the task of pointing and dragging. Although dragging required more time and produced more errors than pointing across all devices, the mouse performance was superior to the others within each task. The survey of input devices by Milner [10] concluded that no one device is appropriate for all tasks.

Despite this fact, the mouse has shown to be the dominant input device in real-world usage. The design space analysis of input devices in the study by Card et.al. [4] showed the superior effectiveness of the mouse for precision pointing tasks. In addition, the direct mapping between user intention (2D cursor movement) and user input (2D mouse movement) reduces learning time.

However, one deficiency of the mouse is the substantial time required between keyboard and pointing actions. The Keystroke-Level Model of [3] showed that this homing time is responsible for 42% of the time required to move the hand from the keyboard to the mouse, point, and return the hand to the keyboard.

Another drawback of the mouse is the consumption of valuable desk space. In their study, Jellinek et.al. [8] explain that this mouse footprint may be reduced by use of a high-gain mouse. However, they also report that pointing speed may be negatively affected by this gain.

Glove Devices and Hand Tracking

Work has been done in tracking finger and hand motion using specialized glove devices. Examples of these are the Data Glove produced by VPL Research, Inc., the Power Glove produced by Mattel, Inc., and the Dexterous Hand Master by Exos Inc. The common characteristic of such devices is the restraint placed on the user who is forced to wear a cumbersome device. These devices require the user to dedicate one or more hands to a particular input device, prohibiting its use for other necessary actions such as writing, typing or dialing a phone.

Recently, researchers have applied the techniques of computer vision to the detection of hand position and movement, ridding the user of encumbering glove devices. Fukomoto, et.al. [7] report of a glove-free pointing system useful for applications requiring computer control from a distance, such as a slide presentation aid. Wellner [13] discusses the DigitalDesk, a system that allows the user to interact with the computer...
while performing typical 'pen and paper' tasks on a desktop. These approaches, however, are not practical for general purpose gesture recognition.

Darrell and Pentland [5] use correlation methods to track a moving hand within a sequence of frames. They use a set of characteristic images to model the gesture, then correlate these to a set of previously learned motions, finding the candidate training gesture with the highest correlation.

In the VISLab, there has been investigation into other approaches for detecting human gestures in three-dimensional space. Quek [11] investigates how humans perceive gesture in order to construct a gesture vocabulary attainable by methods of machine vision.

**The FingerMouse Approach**

Figure 1 shows the FingerMouse system setup. A down-looking camera is trained on the keyboard. The central idea of the system is for a user operating a keyboard to switch into a 'pointing mode' by simply assuming a pointing configuration above the keyboard, as shown in Figure 2. A vision system constantly monitors the hand and tracks the fingertip of the pointing hand. As the fingertip moves in a horizontal plane just above the keyboard, the screen cursor moves accordingly. The user depresses the SHIFT key on the keyboard with the non-pointing hand to register a 'mouse button press'.

![FingerMouse setup](image)

*Figure 1: The FingerMouse setup.*

This approach incorporates the advantages of the mouse in that the 2D plane above the keyboard in which the user points maps directly into the 2D cursor movement on the screen. Since the keyboard is normally placed squarely in front of the user during typing, the 2D pointing space is also user-centered. Thus the user moves his/her pointing hand away and towards his/her body to move up and down the screen respectively. Right and left pointing movements map directly to right and left cursor movements. **FingerMouse**
alleviates the homing time problem of the traditional mouse in that the user's hand does not have to leave the keyboard to point.

Figure 2: Pointing configuration above the keyboard.

PROCESSING

Figure 3 shows the sequence of processing in the FingerMouse system. Beginning with a 24-bit RGB image, the system employs a three-step process: 1. Color histogram-based image segmentation to separate the hand from the keyboard, 2. Rapid finite-state-machine based detection of the hand configuration, and 3. Tracking of the fingertip.

The process produces a 2-tuple \((\delta x, \delta y)\) designating the distance which the fingertip has moved. If step 2 of the process does not detect the hand in a pointing configuration, no output is generated and the process returns to step 1. If, in step 2, the pointing configuration is detected for the first time, \((\delta x, \delta y)\) is set to \((0, 0)\) and the system proceeds to step 3 to compute the current \((x, y)\) coordinate of the fingertip. The system loops back to step 1. If the hand is still in the pointing configuration, the new fingertip location is computed and the displacement \((\delta x, \delta y)\) is output.

The system uses the resulting displacement value to move the cursor. In effect, the first time the pointing configuration is detected, the cursor position just prior to detection is associated with the fingertip location. As the user moves the pointing finger, the cursor mirrors the movement.
Color-Based Segmentation

A color histogram-based approach is implemented for segmentation of the hand from the keyboard. Such approaches typically assume that homogeneous objects in the image manifest themselves as clusters in some measurement space representing color. Examples of such spaces are RGB (the respective red, green, and blue color components) and HSI (a hue, saturation, intensity model). Other measurement spaces have been proposed by various researchers. The most common color space used in computer displays and images is the RGB color space. Therefore, in the interest of speed, the FingerMouse system works in this color space.

Since the color of different users' hands vary, training images are employed to compute user-specific lookup tables. Sample images of the user's hand are extracted for computation of color statistics. Training images taken against a simple background can be easily segmented, allowing for analysis of only those pixels belonging to the hand. Each cell in the three-dimensional RGB lookup table is populated with a count representing the number of hand pixels having that combination of color components. The result is a volumetric color histogram of the user's hand.

After all hand pixels are considered, the histogram is normalized to one by dividing the value of each cell by the value of the largest pixel-count in the histogram. The normalized values constitute the probability that the corresponding color will be found in pixels of the user's hand. For example, the color with the largest number of pixels in the training image will have a probability of one, implying that the color will definitely be found in any image of the user's hand. On the other hand, those colors with a zero pixel-count will have probability of zero, implying that this color will not be found in an image of the hand.

As the user types or points above the keyboard, a sequence of images are captured and analyzed. The first step is to extract the user's hand, if present, from the keyboard. By applying the lookup table to each pixel, a probability image is produced, where each pixel is represented by the probability that it belongs to the user's hand.

These probabilities can be used to segment the hand from the background. The resulting binary image includes only those pixels with a great likelihood of belonging to the hand. Additionally, it would be possible to perform a region growing algorithm to also include pixels whose neighbors have a large probability. Figure 4 displays the result of the segmentation algorithm.
FingerMouse: A Freehand Computer Pointing Interface

Figure 4: The results of color segmentation.

Mode Detection & Tracking

Once the hand has been segmented from the keyboard background, an analysis of the hand shape determines if the user is pointing. This is implemented with a finite-state machine (FSM) that detects different regions of the hand. Mode detection begins by computing a vertical projection histogram of the segmented image; each value in the histogram is a count of the number of 'hand' pixels in the corresponding row. These values are fed to the FSM, from top row to bottom row, as input. State transitions are performed when the row's pixel count exceeds some threshold, indicating that the width of the hand has increased.

The FSM may be expressed as the set \( \{Q, \Sigma, \delta, S, V, A\} \), where \( Q = \{S, 1, 2, 3, 4, T\} \) is the set of machine states; \( \Sigma = \{p(y)\}_{y=1}^{N} \) is the alphabet of the FSM, \( N \) being the number of rows and \( p(y) \) being the number of 'hand' pixels in row \( y \); \( V = \{count, y_{\min}, y_{\max}\} \) is the set of system variables; \( \delta \) is the set of transitions which determine the next state and the new values of the system variables from the current state, the current system variable values, and the FSM input; \( S \) is the start state; and, \( A = \{T\} \) is the set of accept states (i.e., the final state if the system finds a pointing finger candidate).

Figure 5 illustrates the FSM states and transitions. The FSM has a set of parameters \( \{fWidth, fMin, hWidth, hMin\} \) which determine its behavior. The system variable \( count \) retains the number of sequential rows that have been found with some minimum number of 'hand' pixels (either \( fWidth \) or \( hWidth \)); the variables \( y_{\min} \) and \( y_{\max} \) capture the length of the candidate pointing finger.
The states of the FSM indicate the section of the candidate pointing hand that has been encountered thus far. The machine transitions from the start state $S$ to state $1$ on a null transition, effectively starting the machine in state $1$. State $1$ indicates that no portion of the hand has been detected. State $2$ indicates that a sufficient number of pixels, $fWidth$, have been found in a row to believe that the finger has been detected. If $fMin$ rows are found meeting this criterion, a transition is made to state $3$, indicating that the finger may have been found. When the number of pixels in a row exceeds $hWidth$, a transition is made to state $4$, indicating that the 'body' of the hand has been detected. If $hMin$ rows are processed with $hWidth$ 'hand' pixels, the 'body' of the hand has been reached, hence the pointing finger has been found and the FSM retires to state $T$. Note that in states $2$ and $4$, if a row is encountered with an insufficient number of 'hand' pixels, the FSM forces a transition to the preceding state. In this way, noisy rows do not trigger a false state transition.

If the FSM terminates in states $1$, $2$, $3$, or $4$, the image is discarded since no pointing configuration is found. If the machine terminates at state $T$, the system variables $yMin$ and $yMax$ indicate the length of the candidate finger region. If it does not exceed some threshold, the system surmises that the region is not long enough to be an extended finger in a pointing configuration. If the length exceeds the threshold, the hand is considered to be in the pointing configuration. A similar analysis is then performed in the horizontal direction to limit the search space for further processing.

To locate the fingertip position, the reduced search area is examined for 'hand' pixels. These pixels are used to perform moment computations to determine the principal axis...
for the cluster of pixels. The intersection of this line with the upper boundary of the area is recorded as the fingertip. Figure 6 shows a hand over a keyboard. The fingertip located by the algorithm is marked with cross-hairs.

![FingerMouse example](image)

**Figure 6**: The detected fingertip location.

**EXPERIMENTATION**

Some preliminary user experiments were performed with *FingerMouse*. Since the *FingerMouse* is targeted at repetitive applications with constant keyboard-to-pointing (and vice-versa) transitions, users were asked to enter data into an electronic form (shown in Figure 7). In initial informal testing, users who had no experience with *FingerMouse* were asked to fill in a name and address from five different sets of data. The users were videotaped as they performed the task.

The greatest improvement in performance came between the first and second trials. This is probably because the user became familiar with the methodology of *FingerMouse* and acquainted with the time lags in the system. By the fifth trial, all users seem to have achieved some degree of proficiency with the system.

![Experimentation form](image)

**Figure 7**: The form used for experimentation.
CONCLUSIONS and FUTURE WORK

This report has presented the work done on FingerMouse, a hand gesture-based pointing interface which does not require the user to wear an encumbering device or to manipulate some object away from the computer keyboard. The system detailed in this paper was implemented by applying computer vision techniques to sense the configuration of a user's hand and to track the position of the fingertip. The algorithm utilizes a probabilistically driven color-based segmentation strategy. A finite-state machine was designed to detect the hand configuration and a moment-based computation was performed to determine the fingertip position.

All the algorithms were designed with a real-time hardware implementation in mind. For an interactive system such as the FingerMouse, real-time performance is of utmost importance. The limit of video-based hand tracking is ultimately the frame rate of the camera. The system operates on a Silicon Graphics R4400 Indigo2, which can grab RGB frames to memory at a rate of 15 frames per second (fps). After processing, an effective throughput of 7 fingertip locations per second is achieved. While this is only marginally useful for human input, it should be noted that no special hardware was used.

The pixelwise lookup operation is local, making it readily parallelizable. This is significant since with just the segmentation running, the throughput of the system went from 15 fps down to 9 fps. Furthermore, the finite-state machine mode detection algorithm can be committed readily to a hardware implementation, as can the moment computations to find the fingertip.

Further work on this project will be aimed at investigating different color models and optimizing the algorithms to improve the robustness and efficiency of the system. In the VISLab, there is current research into the use of the HSI color space for improved segmentation. It is believed that this color model is less variant over changes in lighting conditions. If so, this model could be incorporated into FingerMouse without modification to the runtime operation. All that is necessary is to modify the approach to populating the volumetric probability lookup table.

Additional user testing of FingerMouse will also be carried out. The goals in this area are to determine the accuracy and repeatability of the calculated fingertip position, the learning rate of users, the skill transference of users familiar with other pointing devices, and a comparison between 'expert' users of FingerMouse and similar users of a traditional mouse.
ACKNOWLEDGEMENTS

The ideas for FingerMouse were originally conceived and implemented by my advisor and mentor, Francis K.H. Quek. Without his constant guidance and support, the completion of this project would not have been possible.

Many conversations with my colleague, Meide Zhao, have sparked outstanding ideas that have steered this project in the correct direction. Additionally, his research into the HSI color space promises to improve the quality of FingerMouse and future projects.

The work presented here was undertaken in the Vision Interfaces and Systems Laboratory at the University of Illinois at Chicago, under the direction of Ardehshir Goshtasby and Francis Quek.

This research is supported by the National Science Foundation, grant number IRI-9396329.
REFERENCES


