Finger Track - A Robust and Real-Time Gesture Interface

Rochelle O'Hagan
Department of Computer Science
Faculty of Engineering & Information Technology
The Australian National University
CANBERRA ACT 0200
rohagan@syseng.anu.edu.au
Tel. (02) 6216 7092 Fax. (02) 6279 8688

Alexander Zelinsky
Department of Systems Engineering
Research School of Information Sciences & Engineering
The Australian National University
CANBERRA ACT 0200
alex@syseng.anu.edu.au
Tel. (02) 6279 8840 Fax. (02) 6279 8688

Abstract

Real-time computer vision combined with robust gesture recognition provides a natural alternative to traditional computer interfaces. Human users have plenty of experience with actions and the manipulation of objects requiring finger movement. In place of a mouse, users could use their hands to select and manipulate data. This paper presents a first step in this approach using a finger as a pointing and selection device.

A major feature of a successful tracking system is robustness. The system must be able to acquire tracked features upon startup, and reacquire them if lost during tracking. Re-acquisition should be fast and accurate (i.e. it should pick up the correct feature). Intelligent search algorithms are needed for speedy, accurate acquisition of lost features with the frame. The prototype interface presented in this paper is based on finger tracking as a means of input to applications. The focus of the discussion is how the system can be made to perform robustly in real-time. Dynamically distributed search windows are defined for searching within the frame. The location and number of search windows are dependent on the confidence in the tracking of features. Experimental results showing the effectiveness of these techniques are presented.

Introduction

Gesture forms a major part of human communication. In fact one definition of gesture is "body movements which are used to convey some information from one person to another" [Vaananen & Bohm]. The form of gestures may vary, but most cultures use gesture to convey information in addition to speech. Humans are also very familiar with direct manipulation of objects. People use their hands to move and shape objects and to learn about their environment. Direct manipulation gesture-based interfaces to computers should, therefore, be intuitive and familiar to users.

Gesture is either spontaneous or conscious. Spontaneous gestures are usually associated with speech and are the primary form of gesture. Conscious gestures provide other information - they can be used as a means of learning the environment through tactile experience, and as a manipulation device. In order to avoid pre-defined interaction techniques and use gesture as an alternative to the mouse for object positioning and manipulation, conscious gestures must be recognised and understood [Cassell]. To determine an appropriate subset of gestures, it is important to understand different types of gesture and the purpose for which various gestures are used.

Two methods of classifying gestures are by type and by function. Cassell classifies conscious gesture by type into groups such as *emblematic* or *propositional* gestures. Examples of emblematic gestures include the "V-for-victory" gesture and "thumbs up" or "okay" ring. Propositional gestures are also associated with speech, and consist of gestures such as using the hands to indicate size while saying "it was this big". Conscious gestures have three functional roles: *semiotic*, *ergotic* and *epistemic* [Cadoz 94]. Gesture in interaction with computers has traditionally focused on the ergotic function - for example typing on a keyboard, moving a mouse and clicking buttons. Both tie the user to the computer by wired hardware. Neither is a natural interface, merely a means of transferring information to the computer.

Computer vision allows body parts and gestures to be an effective and non-intrusive means of input into the computer. The camera can be located as an unobtrusive sensor of human movement. No wired components or markers need to be introduced into the system allowing greater flexibility of action. Various work has been carried out using computer vision with gesture recognition. Crowley *et al* have shown a system using a finger or other object such as a pencil as an input interface to a simple painting program. The system starts tracking when an object is moved into a trigger region. If tracking of the object is lost, the system waits until another object enters the trigger area and then continues tracking the new object. We believe that automatic recovery is necessary in a tracking system. If the target moves out of the workspace and then reenters, it shouldn't have to move to a specific area for tracking to continue, but should be automatically reacquired by the system.

A system developed by Pavlovic *et al* uses two cameras and special lighting to track the motion of the hand and recognise gestures at approximately 10 frames per second. This is slower than the NTSC video rate (30Hz) and doesn't allow for real-time performance of rapid motion gesture. The development of real-time hardware vision systems [Inoue *et. al.*] allows successful tracking at video rate. These systems have been used for robot vision and head tracking. Zelinsky *et. al.*, using the Fujitsu vision system, have produced a system capable of tracking and recognising facial gestures at frame rate. The head-tracking system uses Kalman filters and a relative feature location network to provide robustness. It relies on located features to help determine the position of other features. This works well with the head and face, but is not appropriate for single feature tracking.

The fundamental requirements of such systems are speed, robustness and adaptability. Tracking of features and processing of gestures must be accomplished in real-time (30 Hz for NTSC video). The system must be able to initially acquire, and when lost automatically reacquire, features quickly and accurately. The system must also not rely on special lighting, marks or stickers on the hands, specific “trigger” locations or any other artificial means of aiding tracking.

Finger Track

As a first step toward full hand tracking in three dimensions, we developed the two-dimensional finger tracking interface. The main purpose of the interface was to support investigation into ways of improving the reacquisition of lost features, and hence the robustness of the tracking system. Exhaustive systematic searching of the entire video frame takes too long to be a viable option. Also, as the system is dynamic, there are no guarantees that the object (hand) will stop moving just because tracking has been lost, so a systematic search could fail to find the object completely. Intelligent search algorithms are needed. The tracking system gives two basic pieces of information - the location where the feature was last detected, and the vector of movement at that time. Our system implements a dynamic search pattern focusing initially on areas where the feature is most likely to be found, and widening the search area over time as the feature remains undetected.

The system used a single video camera and dedicated image processing hardware to track the motion of the user’s finger. Image correlation was used to determine the location of the finger in the video frame. Positional information and point or click status was sent to a server program which generated events similar to mouse events for the application programs. Search algorithms were implemented to produce a robust tracking system.

Advances in computer vision have led to achievable real-time performance in tracking features. Dedicated hardware allows processing of video images at frame-rate giving rise to the possibility of performing real-time tracking and recognition of gestures. In any vision based tracking system, tracked features will be lost at some stage. Indeed the system will be in a “lost” state at startup and will have to find the required features within the whole frame. In order to accomplish this reacquisition as quickly as possible, given a limited amount of processing power available, intelligent search methods are needed. A systematic search of the whole image is time-consuming and, with features that are probably moving, quite likely to fail to find the features. A search process that distributes search windows based on the last known location of the feature and the direction of movement is likely to find the feature more quickly than an exhaustive search. The searched area of the frame is initially constrained to the probable location of the feature, and then gradually increased as the feature remains undetected.

The Vision System Used in this Prototype

The Fujitsu MEP tracking vision system was used to track the finger pointer for the gesture interface. This system is designed to track multiple templates in the frames of a

NTSC video stream in real time. The vision system consists of two VME-bus cards - a video module and a tracking module which can track up to 100 templates simultaneously at video frame rate (30 Hz for NTSC). The vision system is controlled by a MC68040 processor card running VxWorks (see Figure 1).

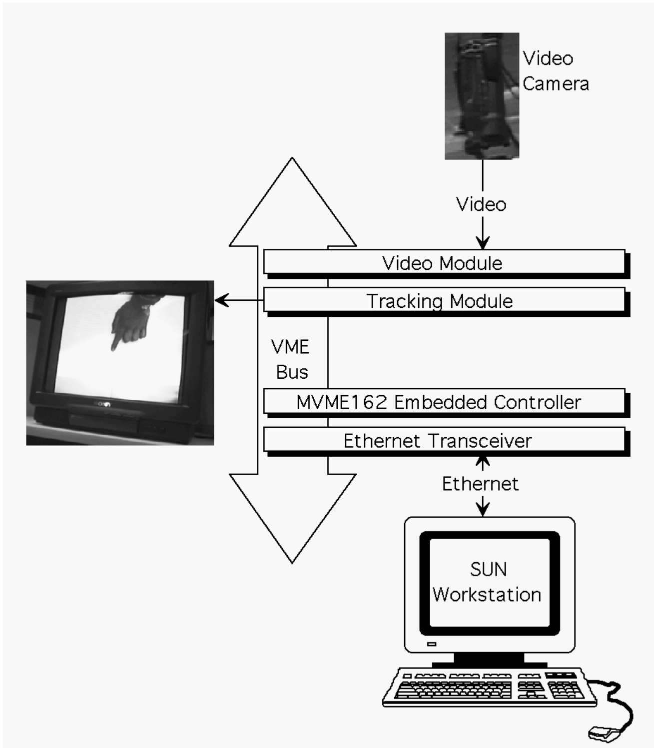


Figure 1. System Configuration

Object tracking is based on comparison of templates within a defined search window. Templates can be either 8x8 or 16x16 pixels in size, with a magnification of up to 4 times a template. The video module digitises the video input stream and stores the digital images in dedicated video memory (VRAM) which is also accessed by the tracking module. The tracking module performs a comparison between the stored templates and the live video at a given location within the frame. The comparison is done using a cross correlation which sums the absolute difference between corresponding pixels in the template and frame. The resulting value is called the distortion, and measures the similarity between the two images. The formula for distortion is shown in Equation 1 where *Size* is the size of the template (8 or 16), $g_t(x,y)$ is the grey value of the template at the specified co-ordinates, $g_f(x,y)$ is the grey value of the pixel in the last frame, m_x, m_y are the magnification of the template in x and y directions and o_x, o_y are the offsets in the frame.

$$D = \sum_{x=0}^{Size} \sum_{y=0}^{Size} \left| g_t\left(x_t : y_t\right) - g_f\left(x_f : y_f\right) \right|$$

where $x_t = x \times m_x$

$$y_t = y \times m_y \qquad \textbf{Equation 1}$$

$$x_f = x \times m_x + o_x$$

$$y_f = y \times m_y + o_y$$

High distortions indicate a poor match while low distortions result when two very similar images are compared. The distortion provides a measure of confidence in the tracking of features. A threshold value of 4000 was determined for the fingertip beyond which tracking seemed to fail. Confidence in tracking was high for distortion values much less than 4000, but dropped off markedly for values above 4000. The linear relationship between distortion and confidence was used in the calculation of search window locations where raw distortion values were too large.

For successful feature tracking it is necessary to calculate the distortion at a number of points within the search window. The tracking system performs up to 256 cross correlations per feature within the search window and finds the position in the image frame where the feature matches with the lowest distortion. Motion is represented by a vector to the origin of the lowest distortion. The search window can be moved along the axis of the motion vector to track an object. For objects that do not change their appearance or shade and are never occluded by other objects, the tracking system works perfectly.

Tracking the Finger

Problems occur when trying to track finger and hand movements as the fingers (in a 16x16 template) do not differ significantly from each other. Also, the shape of the finger is altered by rotation which often occurs when moving the finger over the workspace. Fortunately, the hand's rotation is physically limited in the plane and does not pose difficulty in tracking.

Figure 2. Point Gesture

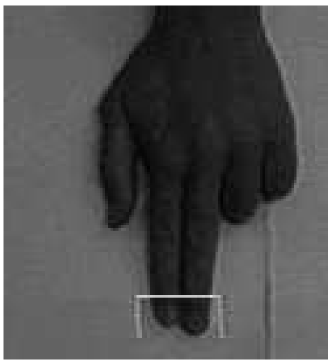


Figure 3. Click Gesture

The vision system was used to track the fore-finger of the hand. Templates were defined of the finger in the "point" position (see Figure 2), and in the "click" position, with the second finger brought alongside the first as shown in Figure 3.

To fully exploit the capability of the vision system, templates were compared not only with the expected position of the finger in the frame, but with surrounding areas as well. The location and number of the surrounding search windows is dependent on the confidence in the current position estimate. When confidence that the finger is being accurately tracked is high, search windows are distributed closely around the expected position. When the feature's position is less certain, the area search windows spread out to cover a wider area of the frame. Fewer search windows are used when the feature is tracking well. If the object is determined to have been lost completely, a search is performed to find it again.

The tracking program allowed the user to define the workspace, move the "mouse" around within the workspace and have the corresponding movement echoed within the application program, and to select screen objects by "clicking" on them. The user's workspace needed to be defined so that accurate relative positions could be mapped to the screen. The user set up the workspace at the beginning of the session by moving the forefinger to the upper-left and lower-right hand corners of the workspace, and performing the clicking action (see Figure 4). This provided the reference coordinates for the workspace area. After setting up the workspace, the user could move the mouse

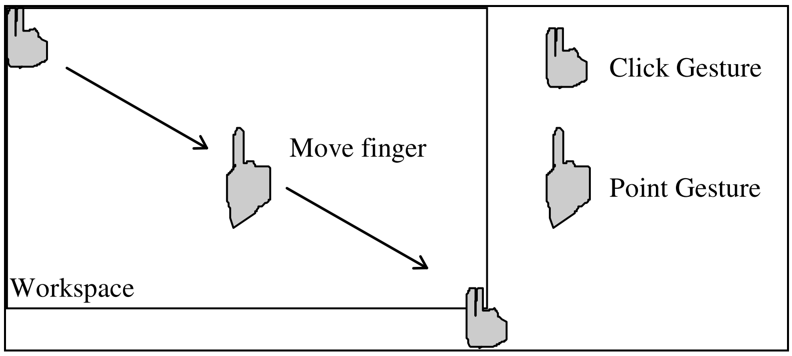


Figure 4. Defining the workspace

cursor around the application program by moving their finger within the defined workspace, and select objects by performing the click action. Double-clicking was achieved by repeating the click action within a short time. The tracking system was set up as a server on VxWorks and passed event calls to application programs as they occurred. This is similar to the way a mouse works. Changes in the position of the finger were sent by the server using Xevents to the application programs. When "clicks" and "double-clicks" were detected, the change in state was also passed to the application. The Finger Tracker interface was trialed with programs such as xv and xfig. This showed that the finger could be tracked in real-time with cursor movement on the screen providing feedback to the user. This was sufficient to test the robustness of the tracking system.

Building Robustness

To initially acquire the finger and start tracking it, it was necessary to find the feature within the frame. The number and location of search windows was determined by the distortion values obtained from the vision system. The program always knows the expected location of the feature and eight surrounding search areas to search in each frame. As the distortion values increased, the certainty that the feature is tracking declines and so more search windows are introduced. It was also important to know that the system was tracking the correct feature. To ensure this, a malus factor was defined, and when the distortion of the new object in relation to the current feature went below the malus factor, tracking switched to the new feature.

Search windows were distributed at a distance, *d*, from the last known position of the feature, and at an angle θ, from the feature's direction of movement. The distribution

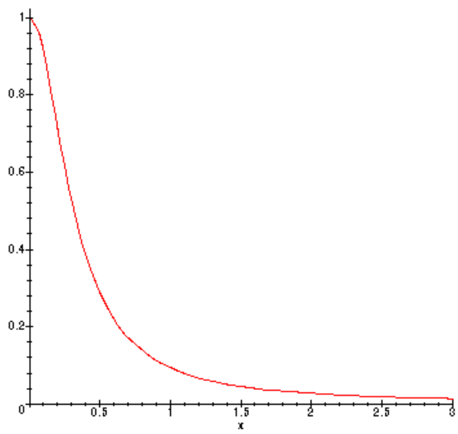


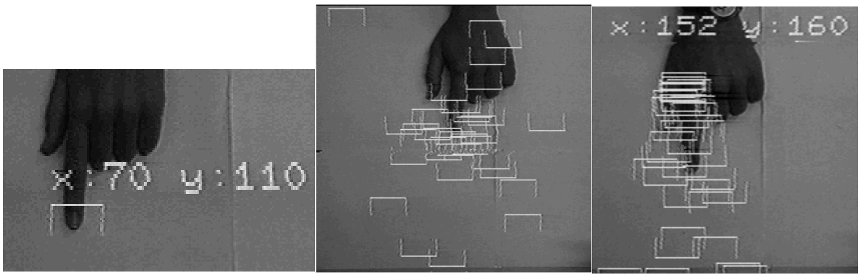
Figure 5. Search window distribution function f(d).

function (Figure 5 & Equation 2) was determined experimentally. The distance *d* was calculated in (Equation 3), where *a, P* are dependent on the confidence of the last known position, and *y* is a random value between 0 - 511 (the maximum size of the frame). This distribution meant that when the confidence in the previous location of the feature was high, the search windows were mostly distributed close to the previous location. When confidence decreased, the windows were more widely spread away from the last location.

$$f(d)=\frac{1}{1+a|d|^{P}}$$ **Equation 2**

$$|d|=\frac{1}{a}\left(\left(\frac{1}{y}\right)^{\frac{1}{P}}-1\right)$$ **Equation 3**

The vision system calculates a movement vector for each template in every frame. This vector gives the direction of motion of the feature when tracking was lost or became uncertain. This movement vector can be used to limit the area of the frame that is initially searched as it is likely that the feature will be somewhere in the direction it was last moving. The confidence in the tracking is used to specify an angle, α (between 15° and 180°), which constrains the search window placement directions. Search windows are placed within $\pm\alpha°$ of the movement vector direction at a randomly generated angle. When the feature was completely lost, for example at startup, α was set to 180, giving a full 360° of search area. As the time that the feature has been lost increased, α was increased to search a wider area of the frame. Figure 6a shows the system when it is tracking the finger. Search windows for a feature that was lost with no reliable movement vector is shown in Figure 6b. The windows are distributed in the full 360° around the centre of the frame. Figure 6c shows the search windows generated for a feature with the movement vector headed straight down the workspace.



a. Tracking the finger. b. 360° Distribution. c. Directed search distribution.

Figure 6. Search Window Distribution

The vision system can track ~100 templates at frame rate. However, there is limited computation time (33ms) between frame grabs. If the system isn't released in time for the next frame grab it waits for the following time slot and so tracking slows down. The computation time is used to calculate the best template match from the distortion values and to calculate the locations of all the search windows. It is important therefore, to only process as many search windows as are necessary to find the feature. The number of search windows was varied according to the confidence of tracking. When the feature was completely lost, the maximum number of search windows were allocated to maximise the chance of finding the feature in the frame.

Conclusions & Further Work

An un-obtrusive, prototype gesture interface allowed investigation of techniques for robust, real-time feature tracking. Fingers of the hand were tracked and events generated and passed through a server to the application program in the same manner as mouse events. When the system lost the finger or confidence in the tracking decreased, more search windows were allocated to recover the feature. A dynamic distribution of the search windows concentrating on the last known location and direction of movement of the object improved the ability of the system to recover the feature and resume tracking.

The long-term goal of the work reported in this paper is to develop a robust, real-time gesture interface for three-dimensional virtual environments. Two-dimensional pointing devices are inadequate for three-dimensional environments such as the Virtual Workbench [Poston *et. al.*]. Currently available interfaces such as data-gloves and the Polhemus FastTrak stylus and switch can be awkward to use in rotation and manipulation of objects in virtual space. These devices are also intrusive, physically connecting the user to the hardware and thus limiting freedom of movement. A vision-based gesture interface could provide an unobtrusive, flexible interface to the virtual world and allow natural manipulation of virtual objects.

The current work forms a good base for tracking the hand and recognising other gestures in three dimensions. The techniques used for feature recovery and acquisition can be transferred to a full hand tracking system, thus providing robustness.

Acknowledgements

This research is supported by The Australian National University and the Co-operative Research Centre for Advanced Computational Systems. The presentation of this report was greatly improved by the assistance of Duncan Stevenson. Technical assistance was gratefully received from Jochen Heinzmann, Gordon Cheng and David Jung.

Bibliography

[Cadoz] C. Cadoz, “Les realites virtuelles”, Dominos, Flammarion, 1994.

[Cassell] J. Cassell. “What You Need to Know about Spontaneous Gesture, and Why You Need to Know It”. 2nd International Conference on Automatic Face and Gesture Recognition, Vermont, October 1996.

[Crowley] J. Crowley and J. Coutaz, “Vision for Man Machine Interaction”, EHCI, Grand Targhee, August 1995.

[Crowley *et. al.*] J. Crowley, F. Berard, and J. Coutaz, “Finger Tracking as an Input Device for Augmented Reality”, International Workshop on Gesture and Face Recognition, Zurich, June 1995.

[Inoue *et. al.*] H. Inoue, T. Tachikawa and M. Inaba, “Robot Vision System with a Correlation Chip for Real-time Tracking, Optical Flow and Depth Map Generation”, Proceedings 1992 IEEE International Conference on Robotics and Automation, pp.1621 - 1626, 1992.

[Pavlovic *et. al.*] V. Pavlovic, R. Sharma and T. Huang, "Gestural Interface to a Visual Computing Environment for Molecular Biologists", Proceedings of the 2d International Conference on Automatic Face and Gesture Recognition, pp. 30-35, 1996.

[Poston *et. al.*] T. Poston and L. Serra, "The Virtual Workbench: Dextrous VR", Proceedings ACM VRST'94 - Virtual Reality Software and Technology, pp. 111-122, 1994.

[Vaananen & Bohm] Gesture-driven interaction as a human factor in virtual environments-an approach with neural networks. In R.A. Earnshaw, M.A. Gigante & H. Jones (Eds.), *Virtual Reality Systems*. London: Academic Press Ltd.

[Zelinsky *et. al.*] A. Zelinsky and J. Heinzmann, "Real-Time Visual Recognition of Facial Gestures for Human-Computer Interaction", Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition, pp. 351-356, 1996.