

Real-time 3D Finger Pointing for an Augmented Desk

Le Song and Masahiro Takatsuka

National ICT Australia*

ViSLAB, School of Information Technologies, University of Sydney
Sydney 2006, New South Wales

lesong@it.usyd.edu.au, masa@it.usyd.edu.au

Abstract

The augmented desk is gaining popularity in recent HCI research. Its layout of a large horizontal screen on the desk enhances immersive and intense collaborative experiences. A responsive and unimpeded input interface is important for an efficient interaction in such an environment. In this paper, we developed a real-time stereo vision-based finger pointing interface for our augmented desk that supports drag-and-drop operation by the tap-and-move of the index finger. The core of our system is a 3D fingertip tracking system, which requires both careful calibration and efficient fingertip localization algorithm. To meet these requirements, we have designed a two-step calibration method that strikes a good balance between accuracy and convenience. Furthermore, based on the chain code representation of the contour, we propose the direction cancellation vector as a tool for fingertip localization. Our algorithm works efficiently with a time complexity of $O(n)$ in term of the length of the chain code. Currently our system allows a user to select and move the displayed contents on the screen directly using his fingertip, and it is applied to the interactive graph drawing paradigm proposed by do Nascimento and Eades (2001). In this application, our real-time pointing interface enables the user to interact with a graph drawing program dynamically, which results in optimal layouts of graphs with maximum symmetry. In the last section, the strength and weakness of our system are discussed, and further suggestions to improve the system are also given.

Keywords: HCI, fingertip tracking, stereo cameras, interactive graph drawing

1 Introduction

In recent years, computers have increasingly influenced the way we communicate with each other, the way we perform actions and the way we interact with environments. Although computers themselves have advanced tremendously in terms of their computing power, the development of interfaces for human to harness this

power conveniently has lagged behind. The most commonly available interfaces between humans and computers are keyboards, mice and LCDs of limited size. These have limited the effectiveness and naturalness of the human-computer interaction (HCI) (Pavlovic et. al. 1997).

To overcome these limitations, many recent HCI research activities have devoted considerable effort to the development of new input interfaces based on natural ways of human communication including gestures, voice and handwriting. On the other hand, various new ideas for computer feedback are also emerging, which not only break the limits of normal screen size and shape but also employ multiple senses for better understanding. Examples of these new generation interfaces include augmented desks that offer a large and open screen for immersive and collaborative experiences (May 1999), volumetric displays that illuminate voxels in 3D space to produce images akin to real world objects (Ebert et. al. 1999), and haptic devices that incorporate the sense of touch to enhance the understanding of visual data (Adcock et. al. 2003).

The augmented desk is a variation of the large screen display that has integrated more elements of collaboration and interaction. The placement of the screen on a desk gives a closer resemblance to real world round-table collaboration and it offers the potential to improve group communication through eye contact, gaze and the ability of each person to experiment with their individual cursor (MacEachren et. al. 2003). Combined with stereoscopic imaging, the augmented desk is able to provide the users a semi-immersive environment for interaction with objects in the virtual world. It is shown that with a certain configuration, real world objects can be reconstructed and inserted into the virtual world for applications in telepresence and computer supported cooperative works (CSCW) (Leibe et. al. 2000; Ullmer et. al 1997).

However, the large horizontal display of the augmented desk amplifies the inconvenience of the traditional mouse. As is similar to the case of Grossman et. al. (2004), the nature of the augmented desks, with images presented on the tabletop, evoke a strong tendency for people to point at them, grab them and move them manually. The existence of the mouse as an intermediary in HCI creates an indirect mapping from the user's interaction space to the display space. To a certain extent, this indirect mapping decreases the degree of resemblance to the object manipulation on a real table. Furthermore, to reduce the necessary arm stretch, a user has to lift his hand to adjust the mappings between the mouse and the cursor, which can be tedious if performed repeatedly.

* National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

Copyright © 2005, Australian Computer Society, Inc. This paper appeared at the 6th Australasian User Interface Conference (AUI2005), Newcastle. Conferences in Research and Practice in Information Technology, Vol. 40. M. Billingham and A. Cockburn, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

An obvious solution is to develop an interface that allows bare-hand selection and manipulation of virtual objects. As one of the most basic tasks, the problem of object selection is addressed in this paper. In our work, finger pointing is used as the interface for selection, which has the following advantages over the traditional mouse:

- It unifies the user's interaction space and information space. For the mouse, a user's intention has to be mapped into the movement of the mouse before it appears as a cursor in the information space. Unlike the mouse, direct finger pointing removes this indirect mapping by coinciding the user's finger movement space with the information space. Furthermore, the user's intention for information manipulation is directly reflected in the change of the displayed contents underneath the fingertip.
- It provides a constant mapping between the movement of the finger and the cursor. As mentioned above, a user has to alter the mapping between hand and cursor frequently in order to use the mouse efficiently. For the finger pointing interface, the mapping is always the same with identical domain and range. This eliminates the need for the user to change coordinate system for comfort.

2 Related Work

However, many finger pointing interfaces are still confined to wired input devices that are tracked through extra hardware by electromechanical or ultrasonic means (Coquillart et al. 1999; Dietz et al. 2001; Seay et al. 1999; Schmalstieg et al. 1999). Commercial products of this type already available in the market include the DiamondTouch (Dietz et al. 2001).

The DiamondTouch embeds a matrix of antennas to the screen surface and requires the user to sit on a special mattress. The system can pick up the signal sent out from the antenna where the finger touches the screen. One obvious shortcoming of this touch screen is that the user is tethered to a specially made mattress that greatly limits their mobility. Furthermore, the interface is blind to the motion of the finger when it is hovering above the screen, since the user is acting as a conductor in the signal transmission loop. Other touch-based interfaces for large screen display are variations of the DiamondTouch and all require the screen to be touched before any tracking is possible.

To address the limitations of tethered devices, researchers have turned to computer vision. Vision-based interfaces for large screen display provide more flexibility for HCI (Hardenberg et al. 2001; Krahnstoever et al. 2002; Leibe et al. 2000; MacEachren et al. 2003; May 1999; Oka et al. 2002; Pavlovic et al. 1997; Rauschert et al. 2002), though certain aspects need further improvement.

An example of exploiting computer vision for interface design is the multimodal interface DAVE_G (Rauschert et al. 2002). DAVE_G employs deictic gestures (pointing using index finger) to locate the region of interests and speech commands such as "zoom" and "center at" are incorporated to allow further operations on the region. The

visual interface is dependent on the speech interface and cannot even perform simple drag-and-drop operations without it. The reason is that only pointing direction of the finger is monitored by the visual interface. This can be improved by incorporating range information into the interface.

One of the most relevant works to this paper is the Perceptive Workbench (Leibe et al. 2000). It uses infrared illuminators to light the scene from above the desk and a camera equipped with infrared filter underneath the desk to extract the contour of a user's arm. Another camera is mounted on one side of the workbench to extract the third dimension of the fingertip as well as its pointing direction. Then a triangulation system calculates the 3D information required to drive an icon and selection ray on the screen of the workbench.

In this paper, we report on our initial attempts for a real-time stereo vision-based finger pointing interface. Compared to other vision-based finger pointing interfaces for augmented desks, our work excels in the following two aspects:

- A drawback of the Perspective Workbench is that the requirement for a side camera reduces the workspace available. This is detrimental to the collaborative environment, especially when several users need to cooperate from different sides of the workbench. Our system can be mounted on the ceiling, thus removing the side camera completely. In return, it provides a larger collaborative environment.
- In order to select an object with some other vision-based interfaces, systems either require the pause of the finger on the object for a certain period (Hardenberg et al. 2001), or require extra gestures to initiate the selection process (Oka et al. 2002). These can be avoided if the motion of the finger is tracked in 3D. Our system reconstructs the temporal sequence of the fingertip position in 3D in real-time. A simple thresholding of the distance between the fingertip and the tabletop suffices for a simulation of an instant virtual button press.

To ensure accuracy of our finger pointing interface, careful calibration is essential. In the following sections, we will first describe the arrangement and calibration of our system. Then the details of our fingertip tracking algorithm is explained; next we will show how the tracked fingertip enable the control of an interactive graph drawing program.

Our current finger pointing prototype supports only primitive operations for coarse interaction, but it can be further improved for finer interaction using the Cross-Key (uses discrete taps on virtual keys integrated with a crosshair cursor) and Precision-Handle (uses a leverage effect to amplify movement precision) techniques proposed by Albinsson et al. (2003). For application in interactive graph drawing, our current implementation can be further developed into full-fledged software similar to the interactive multi-user system of Kobourov et al. (2004). Other limitations and possible improvements are addressed in the last section of this paper.

3 Arrangement and Calibration

3.1 Arrangement of Augmented Desk

The configuration of our augmented desk is similar to the Perceptive Workbench (Leibe et. al. 2000), except that no infrared illuminator is currently used and the frosted glass screen is replaced by two layers of glossy glass with a piece of light absorbent paper between them. We used the glossy glass because it's much less expensive. The mirror underneath the desk is tilted with an angle of approximately 60° and reflects the lights from the projector to the screen above it, as illustrated in Figure 1. A pair of CCD video cameras is mounted on a horizontal rig attached to a tripod, and the two cameras are tilted slightly inward so that their optical axis is not parallel. They are both facing the screen of the desk to capture the hand motions. It should be noted that in our prototype settings the tripod that occupies one side of the desk is an obstacle for collaboration. However, this limitation can be removed easily by mounting the stereo rig on the ceiling.



Figure 1. Arrangement of our augmented desk

3.2 Calibration of Stereo Cameras

For our stereo vision system to work, the two cameras and the screen on the desk have to be calibrated in the same world coordinate system. The world coordinate system could be set in such a way that its x-y plane coincides with the screen plane. Such calibration could be done by displaying a checkerboard pattern on the screen. We have found that this method is not practical since the inexpensive glossy glass reflects the light and tends to blur the corners of the checkerboard. Furthermore, if calibration was done in this way, the relative position between the desk and the stereo cameras has to be fixed, which is not easy to maintain over extended period.

Rather, we perform a two-step calibration process, first for the stereo rig and second for the screen of the augmented desk. For stereo rig calibration, a camera is simplified to a pinhole model, and the perspective transformation matrices of the left and right cameras, C_L and C_R respectively, can be computed by minimizing the following geometric error (Hartley et. al. 2000):

$$e = \sum_i \left(\left\| \mathbf{P}_i^l - C_L \mathbf{P}_i \right\|^2 + \left\| \mathbf{P}_i^r - C_R \mathbf{P}_i \right\|^2 \right)$$

where the summation is over all corner points extracted from a checkerboard pattern. \mathbf{P}_i denotes a corner point in the world coordinates, and \mathbf{P}_i^l and \mathbf{P}_i^r are the projections of \mathbf{P}_i in the left and right image respectively. We present the stereo rig with several different views of the checkerboard pattern and calculate the corresponding parameter matrices for each view using the extracted corners. The view with the least error is selected to set up the world coordinate system and compute the final transformation matrices.

Once the stereo rig is calibrated, less change in the relative position between cameras will occur than that between the desk and the stereo rig. Hence, we can calibrate the stereo rig once at the beginning and use other more convenient method to calibrate the screen of the desk every time there is a change to its position.

3.3 Calibration of the Screen

For a convenient calibration of the screen, it is desirable that the calibration pattern is an intrinsic part of the screen rather than a separate one. This can be achieved by displaying a pattern on the screen. As mentioned in section 3.2, checkerboard pattern is problematic due to the reflection of the screen surface. Instead, we choose a set of 9 (can be more) small circles presented sequentially in 3 rows on the screen, as illustrated in Figure 2.

The centroids of these circles are extracted from the images of both cameras by background subtraction and circularity checking. Then the screen coordinates of the centers of the circles together with the extracted centroids in both images are used to estimate the parameters of the screen plane and to establish the transformation from the world coordinates to the screen coordinates.

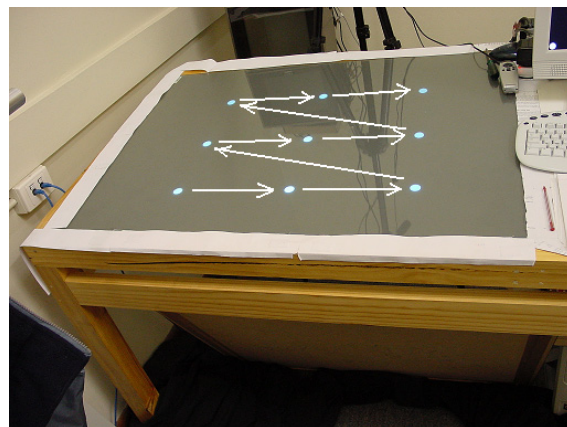


Figure 2. The sequence of circles presented for the calibration of the screen. The arrows indicate the order of the presentation, which starts from the circle in the lower left corner to the one in the upper right corner

The optimal estimation of the parameters for the screen plane can be computed by forming a system of linear equations based on the homography induced by a plane

(Hartley et. al. 2000). However, the method requires the calculation of the fundamental matrix for the stereo rig beforehand, and the positions of the corresponding centroids have to be corrected to satisfy the epipolar constraint. Furthermore, a polynomial equation of degree 6 needs to be solved numerically in order to compute the corrected centroids. Although this method is optimal in its formulation, the programming is relatively difficult. Instead, we use another method that is a good compromise between optimality and complexity.

Suppose the screen coordinates for the center of a circle is $\mathbf{P}_{si} = (x_{si}, y_{si})^T$ where $1 \leq i \leq 9$ and its two projections to the image planes of the stereo cameras are $\mathbf{P}_{li} = (u_{li}, v_{li})^T$ and $\mathbf{P}_{ri} = (u_{ri}, v_{ri})^T$ respectively. The world coordinates $\mathbf{P}_i = (x_i, y_i, z_i)^T$ of \mathbf{P}_{si} can be computed by intersecting two rays and forming triangle system (Takatsuka et. al. 1998) as illustrated in Figure 3.

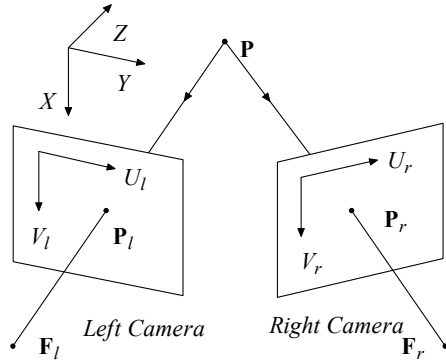


Figure 3. Projective geometric constraints for 3D point \mathbf{P} , whose two images are \mathbf{P}_l and \mathbf{P}_r respectively. \mathbf{F}_l and \mathbf{F}_r are the foci for the left and right cameras respectively

Then the equation of the screen plane $ax + by + cz + d = 0$ in the world coordinate system can be computed using the 9 points that are supposed to lie on the surface. First, we have a system of over-determined linear equation:

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_9 & y_9 & z_9 & 1 \end{bmatrix}_{9 \times 4} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

from which the optimal solution to the equation of the screen plane can be computed using singular value decomposition (SVD). The optimal vector (a, b, c, d) is the column of vector corresponding to the smallest singular value of the decomposition.

3.4 Transformation from World Coordinates to Screen Coordinates

The transformation from the world coordinates to the screen coordinates is useful when we want to determine which position of the screen the fingertip is pointing at. As we will be discussed in the next several sections, the fingertip is tracked in the 3D world coordinate system. We need to establish a mapping from the 3D world coordinate

system to the 2D screen coordinate system. This mapping involves translation, rotation and scaling. Rather than estimate them separately, we take them as a whole, the matrix \mathbf{A} , which satisfies

$$\mathbf{P}_{si} = \mathbf{A} \begin{bmatrix} \mathbf{P}_i \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} \begin{bmatrix} \mathbf{P}_i \\ 1 \end{bmatrix}$$

where $\mathbf{A}_1 = [a_{11} \ a_{12} \ a_{13} \ a_{14}]$, $\mathbf{A}_2 = [a_{21} \ a_{22} \ a_{23} \ a_{24}]$. Reformulate the above transformation and it becomes

$$\begin{bmatrix} \mathbf{P}_i^T & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{P}_i^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1^T \\ \mathbf{A}_2^T \end{bmatrix} = \mathbf{P}_{si}$$

Taking all 9 points into considerations and let

$$\mathbf{Q} = \begin{bmatrix} \mathbf{P}_1^T & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{P}_1^T & 1 \\ \mathbf{P}_2^T & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{P}_2^T & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}_9^T & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{P}_9^T & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} x_{s1} \\ y_{s1} \\ x_{s2} \\ y_{s2} \\ \vdots \\ x_{s9} \\ y_{s9} \end{bmatrix},$$

then the optimal solution to the entries in the transformation matrix \mathbf{A} can be computed in a least square sense by

$$\begin{bmatrix} \mathbf{A}_1^T \\ \mathbf{A}_2^T \end{bmatrix} = (\mathbf{Q}^T \mathbf{Q})^{-1} \mathbf{Q}^T \mathbf{b}$$

Although both the parameter and the transformation estimation above is formulated in the case of 9 circles, it is not difficult to see that more circles can be brought into the calibration and better estimation result can be obtained.

4 Fingertip Tracking

To allow our vision-based interface to have real time performance, we used simple processing of the initial images with algorithms of linear complexity. Our strategy was first to extract the blobs of the hands by thresholding the luminance of the captured images. Then by working on the chain code representations of the contours of the blobs, we were able to compute the fingertip positions using our linear time algorithm. Last, the 3D positions of the fingertips were reconstructed by triangulation and subsequently smoothed using Kalman filters. The focus of our fingertip tracking system is to reduce the processing complexity by limiting sophisticated operations only to geometric primitives. These are detailed in the following three subsections.

4.1 Contour Extraction

The stereo image pairs captured by our cameras are each of size 480×320 , and the majority of time for the fingertip tracking is devoted to the initial localization of the hands. Formerly, we used skin colour segmentation in HSV colour space for this, but it was too slow (10 frames/second in a PENTIUM III 2.4GHz machine with 256Mb RAM) to meet the needs of a real time interface. The reason is that the transformation from the captured

RGB images to HSV colour is time-consuming and it has to run through every pixel of the two image pairs. To avoid this bottleneck, we took advantage of the luminance response of our cameras for blob extraction.

The cameras we used were SONY SSC-DC393P colour video cameras. When we displayed a light-coloured image on the augmented desk and pointed the lens of the cameras towards the screen, the grey level images captured by the stereo cameras were glowing white in the area corresponding to the displayed image, but the rest of the scene were much darker. If we placed our hands on top of the displayed image, we could easily see the shadow of the hands cast on the captured images. This is illustrated in Figure 4(a), and it is this property of the cameras that enables us to extract the blob using simple thresholding. We set the threshold at 16, which made efficient implementation of the threshold possible. We used a binary mask of 11110000 and performed bit-wise “and” operation between the mask and the grey level of a pixel. If the result was none zero, then it was the displayable area of the screen, otherwise it was the shadow of the hand. Note that, the boundary of the displayable area of the screen can be predetermined in the calibration process. We can then limit the luminance thresholding within this boundary (Figure 4(b)).

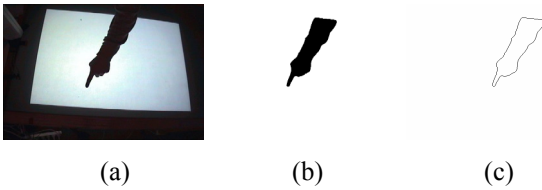


Figure 4. (a) Typical image captured by our cameras. It’s easy to see the shadow cast by the hand. (b) Blob of the hand extracted from the captured image. (c) Contour of the blob

To facilitate our subsequent localization of the fingertip, a standard contour extraction routine ran over the extracted blobs to generate the chain code representations for the edges of the blobs (Figure 4(c)). The information contained in the whole image is then first reduced to a set of pixels (the blobs), and second to a chain of pixels (the contours). These two processing steps are fast and work on a speed of 18 frames/second, quite satisfactory for the real-time purpose. In the following subsection, we will show how our linear algorithm finds the fingertip positions from the chain code.

4.2 Fingertip Detection

The key observation for our algorithm is that the contour of a finger resembles that of a slim test tube and the fingertip lies approximately on the apex of the rounded end of the test tube. It is clear from this heuristic that to judge a position on the contour to be fingertip or not depends not only on the position itself but also on its neighbouring positions. Further, for each position, we had to find a descriptor for its fitness as a fingertip. Instead of computing a single number as the descriptor, we defined a 4-dimensional attribute vector for each position on the chain code, and then 5 criteria were judged against this

attribute vector to determine the fittest position for the fingertip.

Before we further explain the attribute vectors, we will briefly discuss the 8-connected chain code first. As is showed in Figure 5(a), the 8-connected chain code consists of 8 possible values for each position, and these values represent the direction of the next position relative to current position. For example, if the next position is in the southeast (assuming the top of this page is north) of the current position, then the value assigned to the current position is 1. In Figure 5(b), an example contour on a grid is showed, and the positions on the contour are labelled as black, red or green colours. The chain code representation for the whole contour starting from the rightmost red position and running clockwise is (0, 7, 7, 6, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1, 1, 0), and if starting from the topmost green position it is (2, 1, 1, 0, 0, 7, 7, 6, 6, 5, 5, 4, 4, 3, 3, 2).

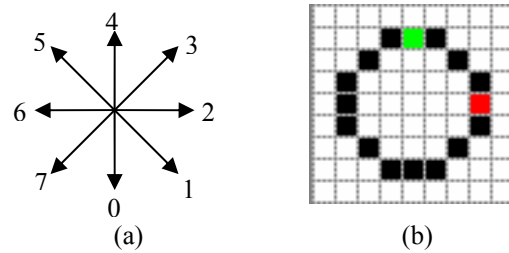


Figure 5. (a) 8 possible value for each code. (b) An example contour

To facilitate our following discussion of the attribute vector, we divide the values of the chain codes into four groups—0 and 4 the first group G_0 , 1 and 5 the second group G_1 , 2 and 6 the third group G_2 , and 3 and 7 the last group G_3 . The two numbers in each group always represent the two opposite directions. For example, in the first group, 0 represents the south, which is exactly the opposite of the north represented by 4. Further, we defined a function $g = g(c)$ on the chain codes which is to compute the group number of a code. For example, if the code for position 20 is 5, then its group number $g(5)$ is equal to 1.

Now, suppose, for each position i , the attribute vector is $\mathbf{v}_i = (v_{i0}, v_{i1}, v_{i2}, v_{i3})$ and its code is c_i . Initially, each dimension of the attribute vectors is set to 0, and then computed by summing over all the information from the $2s$ neighbouring positions in a specific way. The defining formula for the attribute vector $\mathbf{v}_i = (v_{ij})(j=1,2,3,4)$ in position i is:

$$v_{ij} = \sum_{c \in C_{ij}} \left(\frac{c - g(c)}{4} - 1 \right), j=1,2,3,4$$

where $C_{ij} = \{c_k \mid j = g(c_k), i-s \leq k \leq i+s\}$ is the set of codes in the neighbourhood of i whose group numbers are j . Intuitively, each dimension of the attribute vector contains the difference between the number of codes going one direction and the number of codes going the opposite direction, as is illustrated in Figure 6. In this sense, the attribute vector can also be called the direction cancellation vector.

Taking the whole contour in Figure 5(b) as an example, we can compute the attribute vectors for the red-coloured position and the green-coloured position using the above formula. It turns out that both of these two attribute vectors are $(0, 0, 0, 0)$. The result of zero vectors is not an accident, and is due to the symmetry of the contour. The contour showed in Figure 5(b) is symmetric both horizontally and vertically, and the red-coloured position lies exactly on the horizontal symmetric axis and the green-coloured position on the vertical symmetric axis. One can imagine that starting from a position on the symmetric axis, he can travel along the closed contour and back to the starting position. During this tour, every turn of south is cancelled by a subsequent turn of north, and the same thing happens for east and west. So at the end of the tour, the net turning is zero.

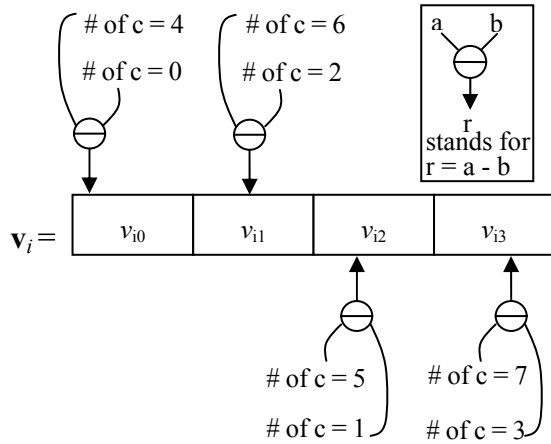


Figure 6. Schematic diagram showing the computation of the direction cancellation vector

Now we can transform our qualitative observation of the symmetry of the finger to a quantitative analysis using our direction cancellation vector. We examine the position of the fingertip together with the s neighbours before and the s neighbours after it on the contour. This span of $2s+1$ positions forms a test tube shape, but it is neither closed nor strictly symmetric. Furthermore the fingertip does not necessarily lie on the symmetric axis. For these reasons, we should not expect the direction cancellation vector for the fingertip position to be zero. However, we can safely assume that this vector is among the smallest of all direction cancellation vectors computed for every position on the contour. To exemplify this claim, we took s as 40 and computed the direction cancellation vectors for each position on a typical contour showed in Figure 4(c). In Figure 7, each dimension of these computed vectors is plotted in a separate curve, and the first order norms of these vectors were also showed in a purple-coloured curve. From Figure 7, it can be seen that the fingertip position obtains or approaches the global minimum in most of these curves. Based on this property, we used these 5 curves to vote for the true fingertip position, by setting the following criteria:

$$|v_{ij}| \leq 12, \text{ for } j = 0, 1, 2, 3 \text{ and } \sum_{j=0}^3 |v_{ij}| \leq 30.$$

In the case of multiple positions satisfying the detection criteria, we simply picked the one with the smallest first order norm to approximate the fingertip position.

In our implementation, the above fingertip detection algorithm required the running through of every chain codes only once. It is because we used the value of the previous attribute vector to find the next, as the spans of contour they covered differed only in the first and last positions. Using techniques similar to box filtering, we were able to obtain an $O(n)$ algorithm in term of the length of the chain codes. In our machine (PENTIUM III 2.4GHz with 256Mb RAM), it took about 1 milliseconds to localize the fingertip, which added very little to the delay and allows for real-time performance.

After the fingertips are localized in the stereo image pairs, the 3D position of the fingertip in the world coordinate system can be calculated by the triangulation method explained in section 3. These recovered fingertip positions are further filtered to smooth out the error.

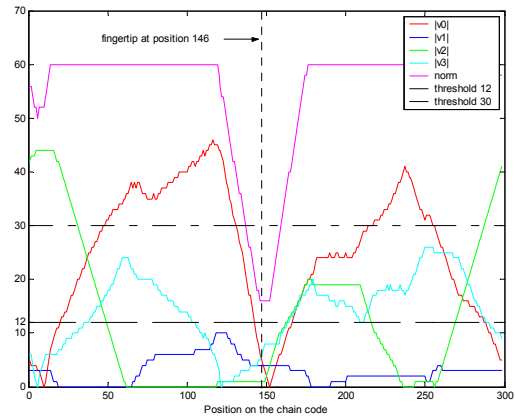


Figure 7. Plot of each dimension of the direction cancellation vectors and the first order norm of the vectors.

4.3 Motion Smoothing

Due to the approximate manner of our fingertip localization algorithm and the errors involved in the 3D position reconstruction, the recovered trajectories for the motion of fingertip contain small jitters and trembles. To provide a stable selection and manipulation of the objects on the screen, we exploited the prediction-and-correction process provided by Kalman filter (Welch et. al. 1995).

We modelled the motion of the fingertip as a first-order kinematics system whose velocity changes smoothly. Hence, the state vector to describe the motion is defined as

$$\mathbf{x}_t = (x(t), y(t), z(t), v_x(t), v_y(t), v_z(t))^T$$

where $x(t)$, $y(t)$ and $z(t)$ are the three coordinates for the position, and $v_x(t)$, $v_y(t)$ and $v_z(t)$ denote the velocities along x , y and z coordinate axes respectively. The reconstructed 3D position in section 4.2 is used to define the observation vector \mathbf{z}_t . As formulated by Welch et. al.

(1995), the state vector \mathbf{x}_t and observation vector \mathbf{z}_t are related by the following system of equations:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_{t-1}$$

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t$$

where \mathbf{w}_t and \mathbf{v}_t represent process and observation noise respectively, which we assume constant Gaussian with zero mean. \mathbf{A} and \mathbf{H} are the state transition matrix and observation matrix respectively. We further assume that the motions along different coordinate axes are independent of each other. Then \mathbf{A} and \mathbf{H} are given as

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta T & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

where ΔT is the interval between frames. Finally, the Kalman filter for the position is formulated as the following prediction

$$\hat{\mathbf{x}}_t^- = \mathbf{A}\hat{\mathbf{x}}_{t-1}$$

$$\mathbf{P}_t^- = \mathbf{A}\mathbf{P}_{t-1}\mathbf{A}^T + \mathbf{Q}$$

and correction process

$$\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_t^- \mathbf{H}^T + \mathbf{R})^{-1}$$

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_t^-)$$

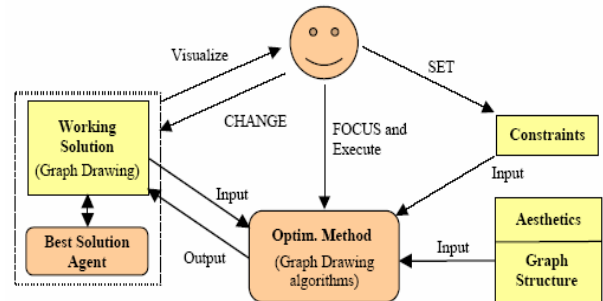
$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}) \mathbf{P}_t^-$$

where $\hat{\mathbf{x}}_t^-$ and $\hat{\mathbf{x}}_t$ are called the *a priori* and the *a posteriori* estimation of \mathbf{x}_t respectively, \mathbf{P}_t^- and \mathbf{P}_t are the *a priori* and the *a posteriori* estimate error covariance matrices respectively, and \mathbf{K}_t is the Kalman gain. \mathbf{R} and \mathbf{Q} are the covariance matrices for the process and observation noise respectively, which are approximated using variance in static gesture. Currently, these two covariance matrices are assumed to be constant for simplicity. However, a more sophisticated modeling of the reconstructed noisy motion can assume variable covariance, which is left for future work.

5 Application: Interactive Graph Drawing

Much of the world's information has a relational structure and can be modelled mathematically as graphs. Example relations include webgraphs, social networks, and biological networks. The aim of many graph drawing algorithms is to produce nice drawings of graphs that satisfy certain aesthetic criteria, such as few edge crossings and few bends on the edges. Unfortunately, to meet these aesthetic criteria, NP-hard problems are usually involved. Consequently, most graph-drawing algorithms are based on heuristics that works relatively fast but may not result in optimal drawings.

One way to deal with this drawback is to manually improve the automatically generated initial layouts. For this purpose, do Nascimento and Eades (2001) has proposed an interactive graph drawing framework where user interaction helps the graph drawing algorithm to escape from local minima. Figure 8 shows the schematic diagram of this framework. Arrows with a capitalized label represent the possible modes of user interaction.



Finger 8. The interactive framework for Graph Drawing (Diagram courtesy of do Nascimento)

According to do Nascimento (2003), the drawing activity is executed as follows: the optimization method automatically creates an initial working solution; then the user starts an iterative process where it is possible to perform manual changes to this drawing, specify layout constraints and re-execute the method on a focused area. Although the framework supports three modes of interaction between the user and the algorithm, we implemented only one mode of user interaction, i.e. the manual changes. As a consequence, the interactive graph drawing framework is simplified to a system of three entities in Figure 9.

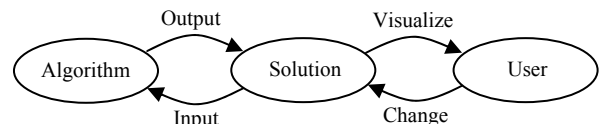


Figure 9. Simplified interactive graph drawing framework

In Figure 10, we show four pictures for a typical interactive optimization process. The first picture shows the initial layout produced by the spring-embedder algorithm (Eades 1984), the second and third picture were taken when the user was changing the position of a node, and the last picture shows the final result after the user finished the arrangement of the nodes so that the drawing reflects maximum symmetry of the graph. It's clear from this simple example that the interaction between the user and the algorithm has improved the layout of the graph to a certain extent.

In our implementation of the framework on the augmented desk, the initial layout of the graph was produced by the spring-embedder algorithm, and then the tracked motion of the fingertip was transformed to a pointing device, which enabled the user to manually change the position of individual nodes. To achieve this, there were two things to do for each tracked position. First, the distance between the fingertip and the screen was computed, and then thresholded to distinguish between the press-down and

lift-up of the fingertip. Second, the projection of the fingertip on the screen was mapped to the screen coordinates, so that the computer knew where on the screen the fingertip was pointing at and displayed a square to represent this position.

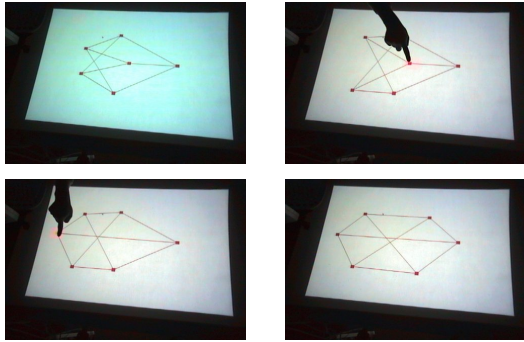


Figure 10. A typical interactive graph drawing process. The upper left image is the initial layout and the lower right image is the final result after manual changes. The other two images show that the user was manipulating a node using his fingertip

Questions concerning the accuracy of our interface arise naturally when one requires the interface to be effective and easy to control. To demonstrate the accuracy of the pointing, we designed an experiment to compare the difference between 5 chosen positions and the tracking results. The 5 chosen positions for the test roughly reside in the centre and the four corners of the screen, with screen coordinates (100, 100), (924, 100), (512, 384), (100, 668) and (924, 668) respectively. In our experiments, a square centred in each of these positions was displayed on the screen and the user was asked to point at the centre of these squares. Since all test positions were on the screen, the error free measure of the distance between the fingertip and screen should be zero. We recorded 100 tracking results for each of these positions, and the mean errors and standard deviations of the tracking results were presented in Table 1.

From the error table, it can be seen that the tracked distance between the fingertip and the screen is quite accurate with mean errors smaller than 1cm. Based on these statistics, we set the press-down threshold as 4 cm, which is fairly resistant to error and requires little effort of the user. A closer look at the error table finds that the errors for screen coordinates can be as large as 25 pixels. Rather than blaming our system for these discrepancies, we attributed them to the user. In our experiment, we displayed 5 squares on the screen and asked the user to point at the centre of the squares. In such situation, the exact locations of the centres are only roughly defined, and the user's subjective judgement plays a large part in where the fingertip should be placed. Furthermore, in the case where the fingertip is allowed as an input device, strict accuracy is less demanded. Indeed, from our experience of using this pointing device for interactive graph drawing, we found that the errors were tolerable, in part because of the size of the displayed objects which were usually larger than 25 pixels. However, a rigorous user study of the effectiveness of our interface still awaits our further investigation.

Table 1. Errors and standard deviations for the tracking results

Items	1	2	3	4	5
m.err.x	1.80	-24.86	8.11	6.39	-1.58
std.x	0.90	1.65	0.55	2.47	0.17
m.err.y	-12.14	13.99	-13.06	4.08	-7.01
std.y	0.62	3.95	2.31	0.43	0.49
m.dis.	-1.51	6.93	-0.13	9.28	7.35
dis.std.	2.06	4.72	3.42	3.92	1.32

* "m." stands for mean; "err." stands for error; "std." stands for standard deviation; and "dis." stands for distance; "x" or "y" stands for x or y screen coordinates.

** Column number 1 to 5 corresponds to the 5 chosen positions (100, 100), (924, 100), (512, 384), (100, 668) and (924, 668) respectively.

*** The unit for the errors of screen coordinates is pixels, while the unit for the errors of distance between fingertip and screen is millimetre.

6 Conclusion

In this study, we developed a real-time bare-hand interface that enables the user to interact with virtual objects in a natural way using their fingertip. Compared to other vision-based interfaces, the drag-and-drop command provided by our interface is more convenient due to its instant response. Unlike other interfaces, which either requires extra gestures to initiate a selection operation or resort to voice for assistance, our interface monitors the distance between the fingertip and the screen instead. When the distance drops below a threshold a mouse press signal is generated automatically, and when the distance increases beyond the threshold the mouse is released.

6.1 Strength

The core of our interface is a stereo tracking system that requires careful calibration for both the cameras and the screen. However, this calibration normally needs to be done only once. The two-step calibration method we designed further lowers the inconvenience caused by possible recalibration. From our experience, changes made to the stereo rig happen much less often than that to the relative position between the stereo rig and the screen. By partitioning the calibration process into two phases and making the calibration pattern part of the screen we are not only able to find a better world coordinate system that minimizes the geometric errors, but also able to cater for the need of a convenient and optimal registration of the screen.

Two parts of our tracking system, the blob extraction and the fingertip localization, consume more than 90 percent of the system's time. It is essential to optimize these two parts in order to meet the real-time requirements. In our system, we used luminance thresholding and direction cancellation vectors to tackle these two problems respectively. Currently both these solutions are efficient and easy to code.

For the blob extraction part, although the luminance thresholding algorithm has to run through every pixel within the displayable area at least once, the operations on each pixel contain only bit-wise “and” and zero-none zero judgement both of which can be implemented efficiently. Further improvement of the blob extraction is possible by combining the information from the prediction process of the Kalman filter. We can use the predicted next fingertip position to define an even smaller area for the luminance thresholding.

For the fingertip localization part, we defined the direction cancellation vector for each position on the contour. This vector provides a compact snapshot of the symmetry of the span of contour under consideration. In our algorithm, the contour of the finger is approximated with a test-tube shape, and the fingertip is taken as the apex of the rounded end of the test-tube. Due to the symmetry of the test-tube shape, we are able to localize the fingertip on the contour by taking votes from the 4 dimensions of the direction cancellation vector and the first order norm of the vector. The computation of the direction cancellation vector is based on the chain code and there already exists efficient implementations of chain code computation. Furthermore, with the help of box filtering technique, one run through the chain code is enough to compute the direction cancellation vectors for each position on the contour. This leads to a fast $O(n)$ algorithm in term of the length of the chain code.

6.2 Weakness and Possible Improvements

Although the preliminary results are quite satisfactory for our real time interface, several aspects of our interface need further improvement.

Currently, the motion tracking mechanism of our interface is a blob-based one, which is robust even to large motion, but the quality of the tracking relies heavily on the successful segmentation of the blob. In the case that parts of the displayed image are of low luminance, the tracking result is not yet satisfactory, since the low luminance parts will be mistaken for the blob of the hand. Therefore, from this point of view, the images that can be displayed on the screen are only of limited range—only images almost bright everywhere are suitable for our current system. In the future, we plan to move the blob segmentation from the natural light domain to infrared domain by adding infrared illuminators underneath the desk and mounting infrared filters to both cameras. This way the segmentation process can be made much easier since the moving arm will block the infrared lights and simple luminance thresholding will be sufficient for the blob segmentation. Furthermore, adding infrared illuminator will virtually allow any kind of images to be displayed on the screen.

Another improvement yet to be made is to allow multiple users to perform selection and manipulation at the same time. Actually, our fingertip extraction algorithm can be scaled up to localize multiple targets in an image, as long as the blobs of the hands are disjoint to each other. However, multiple fingertips create new problem such as correspondence problem between adjacent image frames and occlusion problem. The correspondence problem can

be partially solved by invoking the slow motion assumption, i.e. the movements of the fingertips between two adjacent frames are small. Therefore, the corresponding fingertip in the next frame is the one that is closest to the current fingertip position. The occlusion problem is much harder than the correspondence problem. Once a fingertip is blocked by another hand or fingertip, two hands are joined together in the captured images. Also the information for the occluded hand is missing. Some prior knowledge is needed to resolve these missing data. Due to the intrinsic difficulty of this problem, a lot of research needs to be done before our interface becomes natural and effective for multiple users.

7 References

- Adcock, M., Hutchins, M., and Gunn, C. (2003): Augmented Reality Haptics: Using ARToolkit for Display of Haptic Application. In *2nd International Augmented Reality Toolkit Workshop*, Tokyo, Japan.
- Albinsson, P.A., and Zhai S. (2003): High precision touch screen interaction. In *Conference on Human factors in computing systems*, pp.105-112.
- Coquillart, S., and Wesche, G. (1999): The Virtual Palette and the Virtual Remote Control Panel: A Device and an Interaction Paradigm for the Responsive Workbench. In *IEEE Virtual Reality '99 Conference (VR'99)*, Houston, USA.
- Dietz, P.H., Leigh, D.L. (2001): DiamondTouch: A Multi-User Touch Technology. In *14th ACM Symposium on User Interface Software and Technology (UIST)*, pp. 219-226.
- do Nascimento, H.A.D., and Eades, P. (2002): User Hints for Directed Graph Drawing. in *Proceedings of the 9th Graph Drawing Conference (GD 2001)*, Lectures Notes on Computer Science, 2265:205-219.
- do Nascimento, H.A.D. (2003): User Hints for Optimization Processes. Ph.D. Thesis. School of Information Technology, the University of Sydney, Sydney.
- Eades, P. (1984): A heuristic for graph drawing. *Congressus Numerantium*, 42:149-160.
- Ebert, D., Bedwell, E., Maher, S., Smoliar, L., and Downing, E. (1999): Realizing 3D visualization using crossed-beam volumetric displays. In *Communications of the ACM*, 42(8):101-107.
- Grossman, T., Wigdor, D., and Balakrishnan, R. (2004): Multi-finger gestural interaction with 3D volumetric displays. In *17th ACM Symposium on User Interface Software and Technology (UIST)*, pp.61-70.
- Hardenberg, C.V., and Bérard, F. (2001): Bare-Hand Human-Computer Interaction. In *Proceedings of the ACM Workshop on Perceptive User Interfaces*, Orlando, Florida, USA.
- Hartley, R., and Zisserman, A. (2000): *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK.

- Kobourov, S.G., and Pitta, C. (2004): An interactive multi-user system for simultaneous graph drawing. In *12th Symposium on Graph Drawing (GD04)*.
- Krahnstoeber, N., Kettebekov, S., Yeasin, M. and Sharma, R. (2002): A Real-Time Framework for Natural Multimodal Inter-action with Large Screen Displays. in *Fourth IEEE International Conference on Multimodal Interfaces (ICMI 2002)*, Pittsburgh, USA.
- Leibe, B., Stanner, T., Ribarsky, W., Wartell, Z., Krum, D., Singletary, B., and Hodges, L. (2000): The Perspective Workbench: Towards Spontaneous and Natural Interaction in Semi-Immersive Virtual Environments. In *IEEE Virtual Reality 2000 Conference (VR'2000)*, pp. 13-20, New Brunswick, NJ, USA.
- MacEachren, A.M., Brewer, I., Cai, G., and Chen, J. (2003): Visual-Enabled Geocollaboration to Support Data Exploration and Decision-Making. In *Proceeding of the 21st International Cartography Conference*, Durban, South Africa.
- May, R. (1999): HI-SPACE: A Next Generation Workspace Environment. Master's Thesis. EECS, Washington State University.
- Oka, K., Sato, Y., and Koike, H. (2002): Real-Time Fingertip Tracking and Gesture Recognition. In *IEEE Computer Graphics and Applications*, 22(6):64-71.
- Pavlovic, V.I., Sharma, R., Huang, T.S. (1997): Visual Interpretation of Hand Gestures for Human-Computer Interaction: a Review. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677-695.
- Rauschert, I., Agrawal, P., Fuhrmann, S., Brewer, I., Wang, H., Sharma, R., Cai, G., and MacEachren, A. (2002): Designing a Human-Centered, Multimodal GIS Interface to Support Emergency Management. In *10th ACM Symposium on Advances in Geographic Information Systems (ACM GIS'02)*, Washington, DC, USA.
- Seay, A.F., Krum, D., Ribarsky, W., and Hodges, L. (1999): Multimodal Interaction Techniques for the Virtual Workbench. In *Proceeding of CHI'99*.
- Schmalstieg, P.L., Encarnacao, L.M., and Szalavar, Z. (1999): Using Transparent Props for Interaction with Virtual Table. In *Syposium on Interactive 3D Graphics (I3DG'99)*, Atlanta, USA.
- Takatsuka, M., West, G.A., Venkatesh, V., and Caelli, T.M. (1998): *Low Cost Interactive Binocular Range Finder*. Technical Report No. 5, Curtin University of Technology, School of Computing, Perth, WA, Australia.
- Ullmer, B., and Ishii, H. (1997): The metaDESK: Models and Prototypes for Tangible User Interfaces. In *Proceedings of User Interface Software and Technology (UIST'97)*.
- Welch, G., and Bishop, G. (1995): An Introduction to the Kalman Filter. Technical Report TR95-041, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.