VISUAL PANEL:
From an ordinary paper to a wireless and mobile input device

Ying Wu[*], Ying Shan, Zhengyou Zhang[†], Steven Shafer
[*] Current address: Univ. of Illinois at Urbana-Champaign, Urbana, IL 61801
[†] Contact person. E-mail: zhang@microsoft.com

October 2000

Technical Report
MSR-TR-2000-112

In many intelligent environments, instead of using conventional mice and keyboards, people are looking for an intuitive, immersive and cost-efficient interaction device. This paper presents a vision-based gesture interface system, VISUAL PANEL, which employs an arbitrary quadrangle-shaped panel (e.g., an ordinary paper) and a tip pointer (e.g., fingertip) as an intuitive input device. The system can accurately and reliably track the panel and the tip pointer. The panel tracking continuously determines the projective mapping between the panel at the current position and the display, which in turn maps the tip position to the corresponding position on the display. By detecting the clicking and dragging actions, the system can fulfill many tasks such as controlling a remote large display, and simulating a physical keyboard. Users can naturally use their fingers or other tip pointers to issue commands and type texts. The system, which runs at around 22Hz on PIII 800MHz PC, is scalable and extensible. Further potential applications include multiple-persons interactions.

Keywords: Vision-based user interface, visual panel, new input device, new control device, virtual mouse, virtual keyboard, plane projectivity.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
`http://www.research.microsoft.com`

1 Introduction

Exploring vision-based interfaces is motivated by the unnaturalness of some of the conventional input devices such as mice and joysticks in many intelligent environments where intuitive interactions and tele-operations are required. The bottleneck of such devices comes from the lack of flexibility due to the constraints from the environments, and the lack of the feeling of immersiveness in human computer interaction. Magnetic sensors and transmitters could be a remedy for such conventional interaction devices. However, they are prone to magnetic interferences. At the mean time, many people are reluctant to use them due to the debate that they could be hazard to people's health. On the other hand, combining with human body motion, vision-based interface is of great potential because it provides a non-invasive way to achieve natural and immersive interaction between human and the computer.

There are many examples of the desire of such vision-based interfaces. For an instance, in a smart room, the user wants to control a remote and large display, but he is in a sofa instead of in front of the computer such that the mouse and keyboard may not be accessible. In such situation, what could he do? He may pick up an arbitrary paper at his hand and move his fingers or pens on the paper to drive a cursor or to type some texts. Certainly, such interaction is made possible by having a camera look at the user and analyzing the movement of the paper and the user.

For another example, several people are discussing in a meeting room using a large display. They may need to draw some pictures to show their ideas. However, it is unrealistic to facilitate every user a mouse and a keyboard. What could they do? Again, they may pick up any paper and use their fingers to draw their ideas which will be shown in the large display. By this means, a more immersive discussion can be achieved.

Even more, in a large lecture room, the lecturer sometimes needs to write down something on a small whiteboard. However, the audience far from him or remote audience may not be able to see clearly what he writes. Due to the constraints of the bandwidth, it would not be feasible to broadcast the video of the writing. In this situation, a vision-based system is needed to analysis what the lecturer writes and recover it in remote displays.

In such scenarios as smart rooms, immersive discussions and tele-conferencing, conventional mice and keyboard turn out to be not suitable, which motivates the development of a vision-based gesture interface.

One of the most intuitive and convenient ways to control an intelligent environment is to employ human body motion, especially hand/finger motion[15, 26]. The use of hand gestures has become an important part of HCI in recent years [8, 13, 15]. In order to use human hands as a natural interface device, some alternatives, such as glove-based devices, are used to capture human hand motion by attaching some sensors to measure the joint angles and spatial positions of hands directly. Unfortunately, such devices are expensive and cumbersome. Vision-based technique provides a promising alternative to this goal, since vision-based interface could be very cost-efficient and non-invasive. There have been many implemented application systems in many domains such as virtual environments[2, 6, 9, 30], human-computer interface[1, 4, 10, 16, 18, 22, 25, 27, 28], teleconferencing[12], sign language translation[20, 23], etc.

However, few of such vision-based interfaces are able to achieve accurate display controlling and text input. One of the reasons is that such systems are not facilitated with robust, accurate and fast hand/fingers tracking in live video inputs. 2D tracking could be based on several different cues, such as color[14, 11, 17, 29, 27], motion [24], image features[7], etc. Although color tracking provides a cost-efficient way for tracking, it is prone to lighting changes and it is not suitable for accurate tracking. Although some geometric image features may provide accurate tracking results, the problem is that they are insufficient to represent the object.

We developed a prototype vision-based gesture interface system, VISUAL PANEL, taking advantage of an arbitrary quadrangle-shaped planar object as a panel such that user can use any tip pointer such as his fingertip to interact with the computer. By observing the tip pointer on the panel, we achieve accurate and robust interaction with the computer. Section 2 gives an overview of the design and the main components of the system. Section 3 describes the calculation of the homography, which describes the relationship between the image of the panel and the remote display. Section 4 and Section 5 present the details of the techniques used for tracking the panel and the tip pointer. Section 6 presents the approach we used in the action detector and recognizer. Three applications built on top of the system are presented in Section 7. Section 8 summarizes the paper and gives several possible future extensions.

2 The System

The goal of the VISUAL PANEL system is to use an arbitrary quadrangle-shaped plane object, such as a paper, and a tip pointer, such as a fingertip or a pen, to serve as a natural and convenient input device for accurately controlling remote displays, based on computer vision techniques. Using fingertips, a natural body part, we developed an intuitive and immersive way for accurately interaction with remote and large displays.

The system consists of panel tracking, tip pointer tracking, homography calculation/updating and action detection/recognition. The whole system is shown in Figure 1.

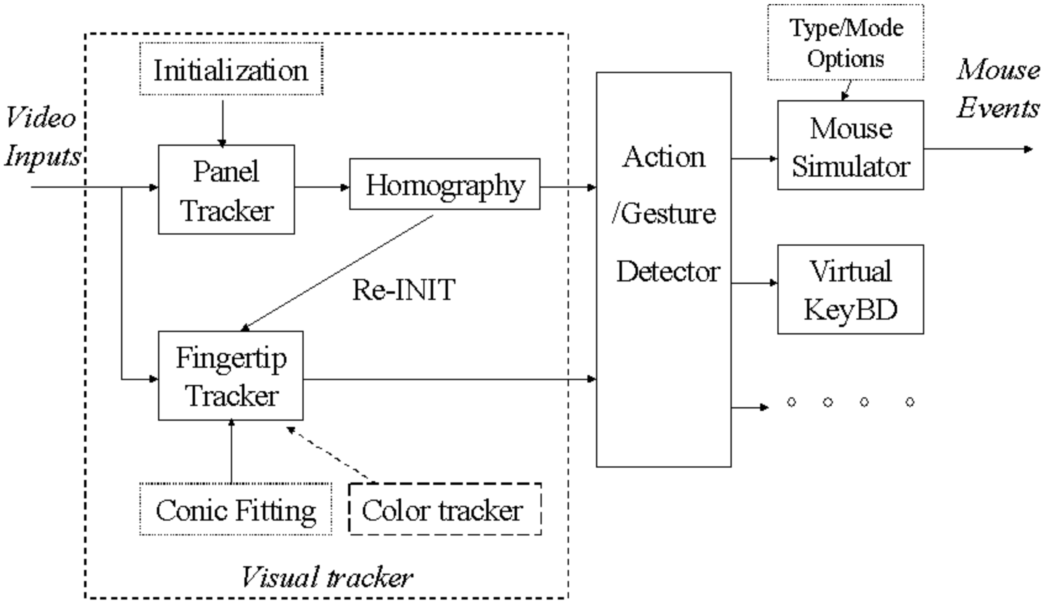


Figure 1: The system of VISUAL PANEL, which consists of panel tracker, pointer tracker, action detector and message generator.

Video sequences are analyzed by a panel tracker and a tip pointer tracker. The panel tracker can accurately track an arbitrary quadrangle-shaped plane object by outputting the positions of the four corners. Since an edge-based dynamical programming technique is employed in the panel tracker, it is quite robust and reliable, even some of the corners of the panel or part of the edges are occluded. At the same time, since the positions of the corners are calculated by intersecting four lines of the quadrangle, the positions are calculated in sub-pixels, which incurs more accurate calculation of the homography, which describes the mapping between such panel and a remote display. Through the homography, any point on the panel is mapped to the corresponding position on the remote display.

In the VISUAL PANEL system, users can use their fingertip as a mouse to simulate a cursor for the remote display. Consequentially, the tracking of the tip pointer should be quite accurate and stable. This is because a small error of tip position will be magnified in the remote large screen. For an instance, we assume the resolution of input video is 320×240 and the remote display is 1024×768. Since generally the panel in the image is roughly half of the size of the image, it is obvious that the tracking error of 1 pixel will incur about 6 pixels error in the large display, which will make the mapped cursor position very shaky. This problem is solved in our system by representing a tip pointer as a conic and fitting a parametric conic to image observations. Therefore, the tip position can also be calculated in sub-pixels such that the error can be reduced.

The current system simulates the clicking/pressing gestures by holding the tip pointer on the position for a while. The message generator in the system gets inputs from the action detector, and issues various mouse and keyboard events, according to the different user input methods.

Building on top of these techniques, our system is capable of performing two types of input: virtual mouse and virtual keyboard, as will be shown in Section 7. The position of the tip pointer can be mapped to the remote display such that a cursor can be simulated. We can also use a paper with a keyboard pattern printed on it as a virtual keyboard, by which users can point the keys on the paper to input texts.

Figure 2 shows the basic idea of visual tracking of the VISUAL PANEL system. Figure 2(a) is one frame of the video input, and Figure 2(b) shows the tracking results of the panel and the fingertip.

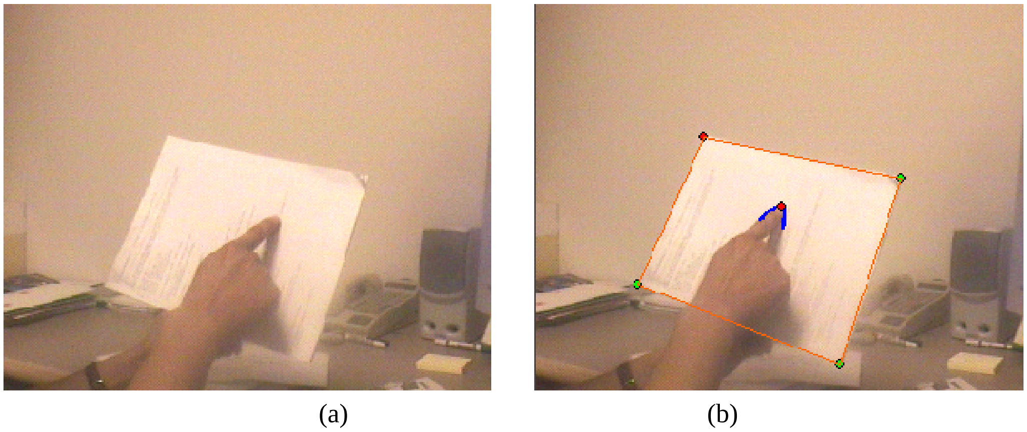

(a) (b)

Figure 2: The tracking. (a) input image (b) tracking outputs: a tracked panel and a tracked fingertip.

- **Camera Setting**: The setting of the camera can be quite flexible. It can be anywhere as long as the panel is not completely occluded. If the camera is fixed, the panel cannot move too far, and it should be in the field of view of the camera. We mount the camera on the ceiling. The user can rotate, translate and tilt the panel to reach a comfortable pose for use. Obviously, users should not let the normal of the panel vertical to the optical axis of the camera. Under some circumstances when the user want to walk around, we can mount a camera on top of his head by wearing a hat, or on his shoulders, such that the user can be anywhere to interact with the computer. This would be quite useful, for example, for a speaker who gives his presentation while walking around.

- **Panel Design**: The panel can be anything as long as it is quadrangle-shaped. For example, we can use a piece of white paper or a cardboard, which is widely available in offices and at homes.

- **Tip Pointers**: The system allows arbitrary tip pointers, such as fingertips and pens, as long as their color is distinguishable from the panel's color. In our usability studies, many users prefer using pens to fingertips in some applications like finger painting, since pens are more intuitive for them. Although using fingertips is of more fun.

- **Clicking**: The current VISUAL PANEL system simulates clicking and pressing by holding the tip pointer for a while. We are exploring the possibility of using some natural gestures to act as clicking.

3 Plane Perspectivity: Mapping Between Panel and Display

Since we use an arbitrarily rectangle-shaped panel to control the cursor position on the remote display, we have to know the mapping between a point on the panel and a point on the display. Furthermore, what is available is an image sequence of the panel which may undergo arbitrary motion (as long as the image of the panel does not degenerate into a line or a point), so we also need to know the mapping between a point in the image plane and a point on the panel. We assume the camera performs a perspective projection (pinhole model) [5]. As the display, the panel, and the image plane are all planes, both above relationships can be described by a *plane perspectivity* [19], as to be explained below.

Given a point $\mathbf{p} = [x, y]^T$ on a plane Π, we use $\tilde{\mathbf{p}} = [x, y, 1]^T$ to denote its homogeneous coordinates. Then, the plane perspectivity between planes Π and Π' is described by a 3×3 matrix $\mathbf{H}$ such that

$$\lambda \tilde{\mathbf{p}}' = \mathbf{H}\tilde{\mathbf{p}} \tag{1}$$

where λ is an arbitrary non-zero scalar. This implies that the homography matrix is only defined up to a scale factor, and therefore has 8 degrees of freedom. If four couples of corresponding points (no three of them are collinear) are given, the homography matrix can be determined [31].

It is not difficult to see that the composition of two plane perspectivities is still a plane perspectivity. Thus, the mapping between the image of the panel and the remote display can be described by a homography matrix. This is very important because what we really need is to use the detected tip position in the image to control the cursor position on the remote display. (If we instead estimate the above-mentioned two homographies, additional calibration is necessary, making the system setup unnecessarily more complicated.)

The composed homography can be easily determined once the four corners of the panel are located in the image (see next section for details). As we know the dimension of the display, we compute the homography by mapping each corner of the panel to a corner of the display.

4 Tracking a Quadrangle

4.1 Quadrangle Representation

The image of the panel can be represented by a quadrangle:

$$\mathcal{Q} = \{l_1, l_2, l_3, l_4\} \tag{2}$$

where l_i is a side line. It can also be represented by four corners $\mathcal{Q} = \{q_1, q_2, q_3, q_4\}$ with $l_k = q_{k-1}q_k$ (we assume $q_0 = q_4$).

Each side of the quadrangle in the image is expected to be a set of edge points due to the difference between the panel and the background. We model the appearance of each side as a random vector $\mathbf{x} = \{G, I\}$, where G is the average gradient and I is the average intensity. The distribution of $\mathcal{X}$ is assumed to be a Gaussian, i.e., $\mathbf{x} \sim N(\mu_x, \Sigma_x)$. More richer modeling of the appearance is under investigation.

4.2 Tracking Through Dynamic Programming

At time frame t, the location of the quadrangle is at $\mathcal{Q}(t) = \{q_1(t), q_2(t), q_3(t), q_4(t)\}$, and the appearance of the quadrangle is $\mathbf{x}(t)$. The tracking can be formulated as a MAP (maximum a posteriori) problem:

$$\mathcal{Q}^*(t+1) = \arg\max_{\mathcal{Q}} p(\mathcal{Q}(t+1)|\mathcal{Q}(t), \mathbf{x}(t), \mathbf{x}(t+1))$$

Because the panel motion between successive image frames is limited, we assume at time $t+1$ these four corner points will be in a range D_i around $p_i(t)$, respectively. The above problem can then be approximated by

$$\begin{aligned}\mathcal{Q}^*(t+1) = \arg\max_{\mathcal{Q}} p(&\mathcal{Q}(t+1), \mathbf{x}(t+1)|\mathcal{Q}(t), \mathbf{x}(t)\\ &: \{D_1, D_2, D_3, D_4\})\end{aligned}$$

Here, ":" means that $\{D_1, D_2, D_3, D_4\}$ are parameters for the probability. Obviously, this is a formidable searching problem. To illustrate this (see Figure 3), we assume the size of each search area of D_i is N. The complexity of the exhausted search for this problem is $O(4^N)$. However, since the four sides of the quadrangle are sequentially connected, this problem can be solved by the *dynamic programming* technique [21].

$$\begin{aligned}\mathcal{Q}^*(t+1) &= \arg\max_{Q} \sum_{i=1}^{4} p(\mathcal{Q}(t+1), \mathbf{x}_i(t+1)|\mathbf{x}_i(t), \mathcal{Q}_i(t)\\ &\qquad\qquad : D_i(q_i(t), q_{i-1}^*(t)))\\ &= \arg\max_{\{q_k\}} \sum_{i=1}^{4} p(\mathbf{x}_i(t+1)|\mathbf{x}_i(t), q_i(t), q_{i-1}^*(t))\end{aligned} \tag{3}$$

That is, we try to estimate each side of the quadrangle sequentially by maximizing a suitable criterion (see below).

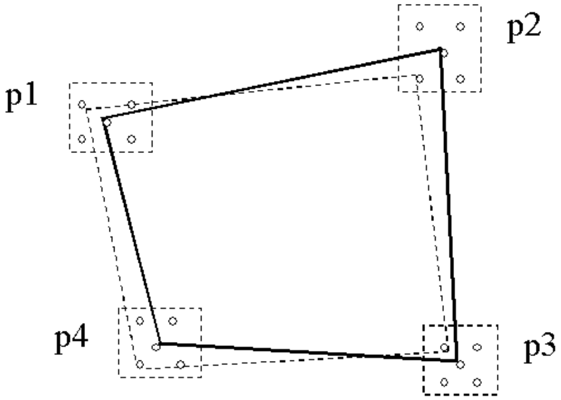


Figure 3: Tracking a quadrangle by dynamic programming technique

In our implementation, the search region for each corner point is approximated by a line segment, instead of a region. This is equivalent to searching for side lines. Corner points are then computed from the intersection of these lines.

Criterion. As mentioned earlier, the appearance of each side line of the quadrangle is modeled by $\mathbf{x}$ that contains both the gradient information and the color information. Maximizing the probability in (3) implies to finding a pair of line segments between t and $t+1$ such that their appearances are closest. This can be done by minimizing the relative entropy between their distributions [3].

Assume Gaussian distribution of X and Y, then the relative entropy:

$$\begin{aligned} D(X||Y) &= E[\lg \frac{p_x(u)}{p_y(u)}] = \int p(u) \lg \frac{p_x(u)}{p_y(u)} du \\ &= \frac{d}{2} \lg \frac{|\Sigma_y|}{|\Sigma_x|} - \frac{1}{2} + \frac{1}{2} E[(x-\mu_y)' \Sigma_y^{-1} (x-\mu_y)] \\ &= \frac{d}{2} \lg \frac{|\Sigma_y|}{|\Sigma_x|} - \frac{1}{2} + \frac{|\Sigma_y|}{2|\Sigma_x|} + \frac{1}{2} (\mu_x-\mu_y)' \Sigma_y^{-1} (\mu_x-\mu_y) \end{aligned}$$

Thus, we have a symmetric distance metric:

$$\begin{aligned} D(X,Y) &= 2(D(X||Y+) + D(Y||X)) \\ &= \frac{|\Sigma_y|}{|\Sigma_x|} + \frac{|\Sigma_x|}{|\Sigma_y|} + (\mu_x-\mu_y)'(\Sigma_x^{-1} + \Sigma_y^{-1})(\mu_x-\mu_y) - 2 \end{aligned} \tag{4}$$

By this means, we can find the best-matched line at time $t+1$ by:

$$l_i^*(t+1) = \arg \min_{\{q_i, q_{i-1}\}} D(\mathbf{x}(t), \mathbf{x}(t+1) : \{q_i, q_{i-1}\}) \tag{5}$$

5 Tracking a Fingertip

The section presents the method of determining the tip location of a tip pointer by conic fitting for tip pointer tracking, and the method of re-initialization of the tracking by a background subtraction technique.

5.1 Fingertip Representation

A tip pointer, such as a fingertip, can be represented, locally around the tip, by a conic

$$a_1x^2 + a_2y^2 + a_3xy + a_4x + a_5y + 1 = 0 \tag{6}$$

Given a set of positions $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ of edge pixels of the tip pointer, we can fit a conic to such data by least-squares estimation. Explicitly, we have,

$$\begin{bmatrix} x_1^2 \; y_1^2 \; x_1y_1 \; x_1 \; y_1 \\ \vdots \\ x_n^2 \; y_n^2 \; x_ny_n \; x_n \; y_n \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_5 \end{bmatrix} = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix} \tag{7}$$

Concisely, this equation can be written as $\mathbf{M}\mathbf{a} = \mathbf{b}$. So, the least-squares solution of $\mathbf{a}$ is given by:

$$\mathbf{a}^* = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{b} \tag{8}$$

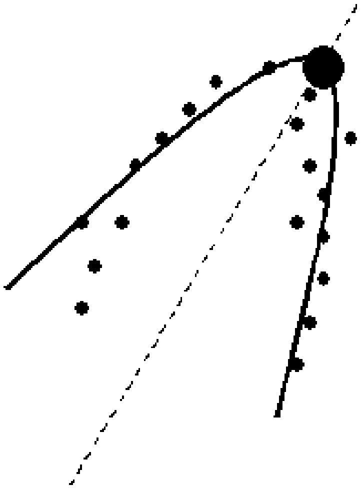


Figure 4: Fingertip detection by conic fitting

Once the conic is known, the tip can be determined, as shown in Figure 4, as one of the intersections of the conic with the major axis.

5.2 Tracking Tip Pointer

The tracking of a tip pointer is quite intuitive and cost-efficient. Assume the position of the tip at time t is $p(t)$. Kalman filtering technique can be employed to predict the tip position $\bar{p}(t+1)$ at time $t = 1$. In a small window, say 30 × 30, we identify as many as possible edge pixels that probably belong to the edge of the tip by thresholding the gradient and taking advantage of color of previous edge of tracked tip. After that, we can fit a conic to these edge pixels and solve the exact tip $p(t+1)$ for time $t+1$.

5.3 Maintaining background and Re-initialization

To make the system more robust and easy-to-use, the scheme of automatic tracking initialization and tracking recovery should be embedded in the system. Here, we developed a technique for re-initialization by a dynamic background subtraction technique.

Assume we have already registered the panel at the beginning of the application, i.e., the position of the panel $\mathcal{Q}(0)$ at time 0 is known. Since at time t, the system can track the panel position $\mathcal{Q}(t)$, the

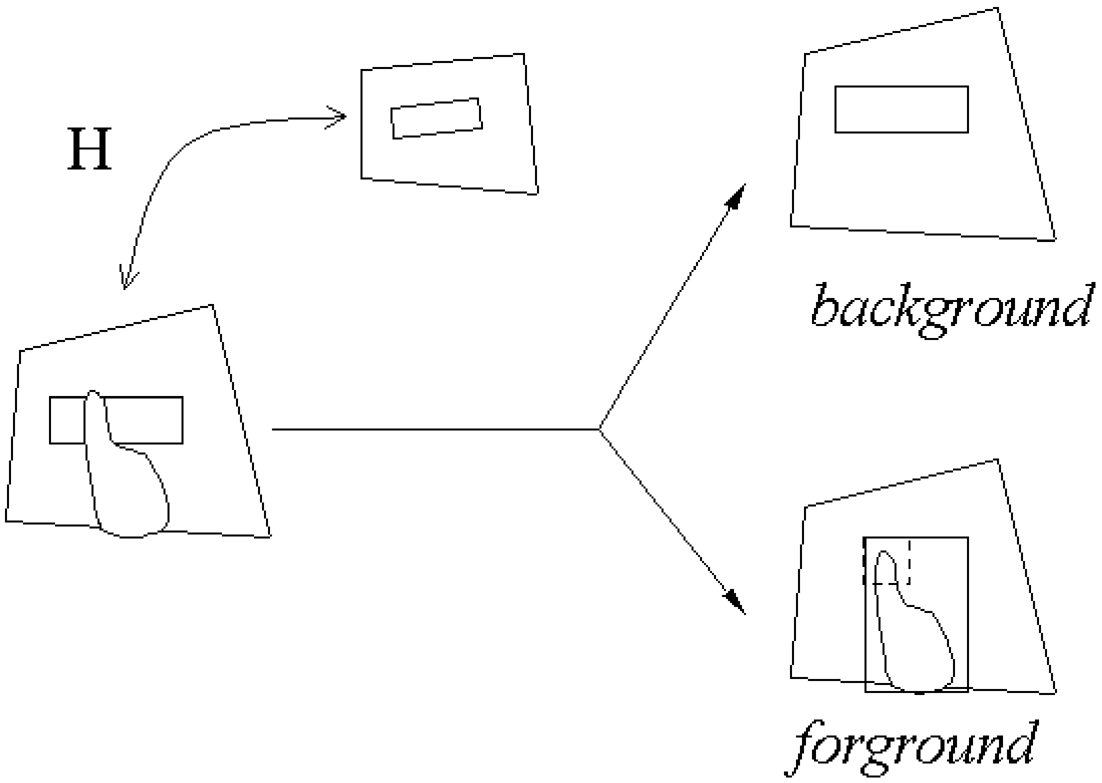


Figure 5: The foreground, i.e. hand, can be segmented out from the background, since the current position of the panel is tracked and a background template is maintained.

homography $\mathbf{H}(t)$ between $\mathcal{Q}(0)$ and $\mathcal{Q}(t)$ can be easily calculated. Through the homography $\mathbf{H}(t)$, the pixels $\mathbf{p}_t(0)$ in the panel at time 0 are mapped to the panel at time t as $\mathbf{p}_b(t)$ by:

$$\tilde{\mathbf{p}}_b(t) = \mathbf{H}(t)\tilde{\mathbf{p}}_t(0)$$

We thus have a virtual image $I_0(\mathbf{p}_b(t))$ that the panel should be observed if there is no tip pointer. Subtracting $I_0(\mathbf{p}_b(t))$ from the current image gives us a difference image. The tip pointer is likely located in areas with large color difference. Figure 5 shows the basic idea of our approach.

6 Action Detection and Recognition

This section presents techniques of action detection and recognition. Our VISUAL PANEL system simulates two mouse button pressing modes (clicking mode and dragging mode), and two mouse motion types (absolute type and relative type).

6.1 Two Mouse Pressing Modes

Our system has two mouse button pressing modes: mode I (clicking mode) which simulates the left button down then up automatically and mode II (dragging mode) which simulates the left button down until released. In our current implementation, clicking/pressing is simulated by holding the tip pointer for a while, say 1 second. This is illustrated in Figure 6, where the horizontal axis indicates the time and the vertical axis of the trajectory indicates the tip location.

A state variable S maintains two states: UP and DN, to simulate the two natural state of a button. The variable S is initialized to be UP. In the clicking mode (mode I), when the system detects that the tip pointer

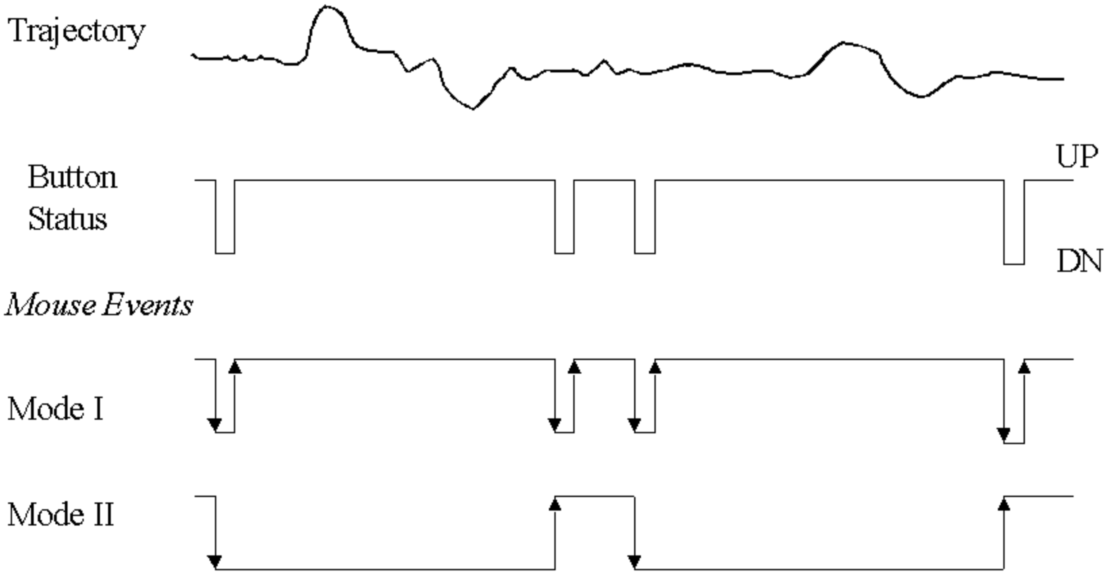


Figure 6: Simulating clicking (mode I) and dragging (mode II)

has been at a fixed place for, say, 1 second (or other amount of pre-specified duration), the state variable S is set to DN. After 0.1 second, the state variable S will be automatically set to UP to simulate button release. Appropriate mouse events are generated, and a clicking action is performed.

Obviously, the clicking mode (mode I) has very limited ability of dragging, since the release is automatic. To simulate dragging, mode II uses another state variable D to memorize the flip of clicking. When the system detects that the tip pointer has been at a fixed place for, say, 1 second (or other amount of pre-specified duration), variable D changes its state (from D_UP to D_DN or from D_DN to D_UP). When the D-state change from D_UP to D_DN is detected, a pressing action is detected; when the D-state change from D_DN to D_UP is detected, a releasing action is detected. Thus, we can pick a window and drag it to a different place.

Note that the clicking event can also be triggered in the dragging mode if the pointer tip stays in the same location twice longer.

In our current implementation, an icon is provided in the menu bar. By clicking on that icon, the system switches between the dragging and clicking modes.

6.2 Two Mouse Motion Types

The VISUAL PANEL system can simulate two mouse motion types: absolute and relative. In the absolute type, the panel will be mapped to the whole remote display, such that each point in the panel will be mapped to the corresponding point in the display. As we discussed before, this type need very accurate tracking, since a small tracking error of the panel and tip pointer will be magnified. However, the absolute type is more intuitive.

We also offer an alternative type based on relative motion, which is much less sensitive to the tracking accuracy, since the cursor is controlled by the relative motion of the tip pointer. Assume the motion direction of tip pointer is $\mathbf{d}_p(t)$ at time t. The moving direction of the cursor will be $\mathbf{d}_d(t) = H(t)\mathbf{d}_p(t)$. The speed of cursor motion is determined by the velocity of the tip pointer, i.e, $\Delta_d = \alpha||\mathbf{v}_p||$, where α controls the scale of the cursor speed on the display. The relative type incurs much smooth movement of the cursor with small α, due to the non-magnification of tracking error.

There could be many other alternatives of relative mouse motion. For an instance, the panel can be just mapped to a window area centered at previous cursor position on the remote display. In this method, the center of the panel corresponds to the previous cursor position. When the tip pointer moves from center to left, the cursor will move left. Obviously, the window area could be smaller than the panel in the image, such that the tracking error can be even minimized.

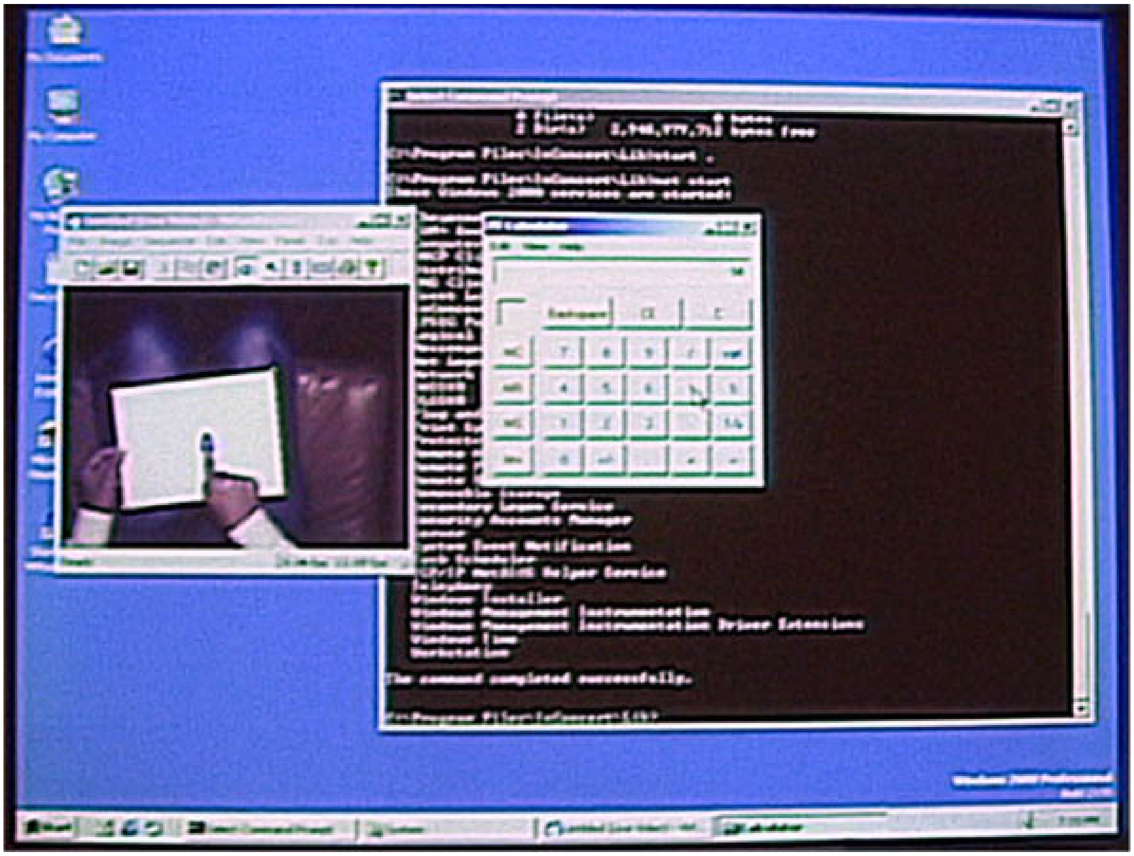
7 Demos

Based on the VISUAL PANEL system, several applications are made to demonstrate the capacity of the system. A video filming a live demonstration is available upon request.

In this section, we explain three applications: control a calculator, draw a picture with a finger, and input text without using any keyboard.

7.1 Controlling a Calculator

This application demonstrates the accuracy and stability of the VISUAL PANEL system. The Calculator, with around 30 buttons, takes a very small part area of the display. In this demo, user can freely use his fingertip to click any buttons or menus of the Calculator, as shown in Figure 7 where the cursor is positioned on the button "×". The tracking error is less than 1 pixel, and the motion of the cursor is very smooth.

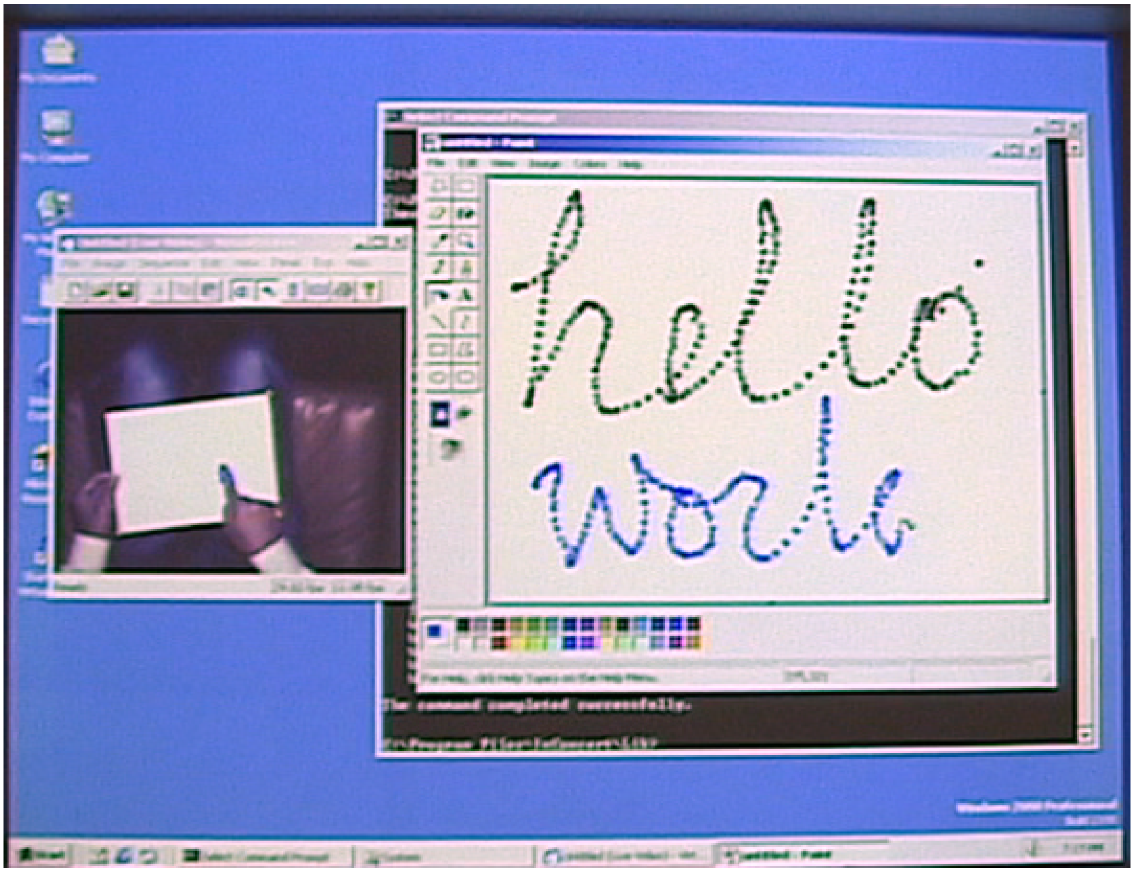


```
These Windows 2000 services are started:

Secondary Logon Service
Security Accounts Manager
System Event Notification
Task Scheduler
TCP/IP NetBIOS Helper Service
Telephony
Windows Installer
Windows Management Instrumentation
Windows Management Instrumentation Driver Extensions
Windows Time
Workstation

The command completed successfully.
```

Figure 7: Controlling a calculator.

7.2 Finger Painting

This application demonstrates different mouse button pressing modes. In `Paint`, a Windows application, a user can use his finger to select any tools and draw anything. Our usability study shows that users learn quickly how to draw a picture and control the remote display with their finger using our VISUAL PANEL system. Figure 8 shows a snapshot of the display while a user was finishing painting "hello world". The left window displays the panel and the hand viewed from the camera.



The command completed successfully.

Figure 8: Finger painting.

7.3 Virtual Keyboard

This application demonstrates that the physical keyboard can be replaced by a printed virtual keyboard in our VISUAL PANEL system. We print a keyboard pattern on the panel, which is shown in Figure 9. When user points to any of the keys on the panel, a key-down message will be sent to the operating system, such that the current active application will receive such key. For example, we can use Windows `Notepad` to receive text inputted by the user. Figure 10 shows a snapshot of the display while a user was inputting “hello world. THANK YOU” with the virtual keyboard.

In our current implementation, users can only use one of their fingers. The typing speed is slow. We are not claiming that we can get rid of the physical keyboard. However, our system could be a very promising alternative when the physical keyboard is not available under some circumstances.

8 Extensions and Future Work

We have developed a prototype vision-based gesture interface system called VISUAL PANEL, which is capable of performing accurate control of remote display and simulating mouse and keyboard input. The VISUAL PANEL system employs an arbitrary quadrangle-shaped plane object as a panel, which can be considered as a display or a keyboard. It can robustly and accurately track the panel and the tip pointer. A user can use their fingers or other tip pointers to simulate a cursor pointer and issue clicking/pressing instructions. After the action is detected, various events can be generated and sent to the computer operating system. Such vision-based gesture interface can achieve more natural and intuitive interaction of human and computer. Three applications have been described: control a calculator, paint with fingers, and input text with a virtual keyboard.

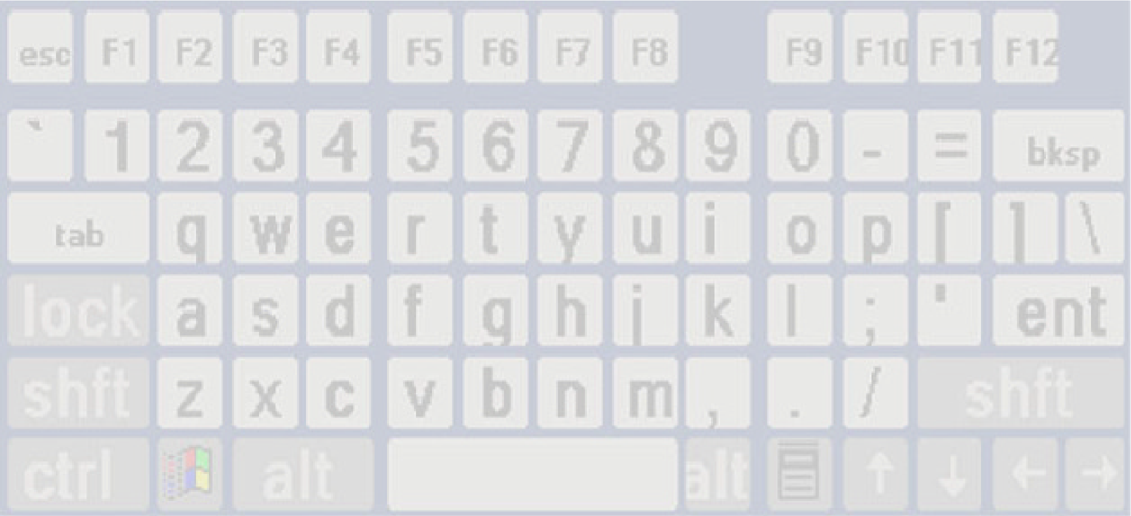

esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
` 1 2 3 4 5 6 7 8 9 0 - = bksp
tab q w e r t y u i o p [ ] \
lock a s d f g h j k l ; ' ent
shft z x c v b n m , . / shft
ctrl alt alt ↑ ↓ ← →

Figure 9: Virtual keyboard

The VISUAL PANEL system leaves a lot of room for extensions and improvements in various aspects, especially in action recognition. In our current implementation, action is triggered when the tip pointer stays immobile for a short duration (say 1 second). We are investigating more natural ways to do that, for example, by combining hand gesture recognition.

References

[1] S. Ahmad. A usable real-time 3D hand tracker. In *Proc. IEEE Asilomar Conf.*, 1994.

[2] G. Berry. Small-wall: A multimodal human computer intelligent interaction test bed with applications. Master's thesis, Dept. of ECE, University of Illinois at Urbana-Champaign, 1998.

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[4] J. Crowley, F. Berard, and J. Coutaz. Finger tracking as an input device for augmented reality. In *Proc. Int'l Workshop on Automatic Face and Gesture Recognition*, pages 195–200, Zurich, 1995.

[5] O. Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, 1993.

[6] K. Imagawa, S. Lu, and S. Igi. Color-Based hands tracking system for sign language recognition. In *Proc. of Int'l Conf. on Face and Gesture Recognition*, pages 462–467, 1998.

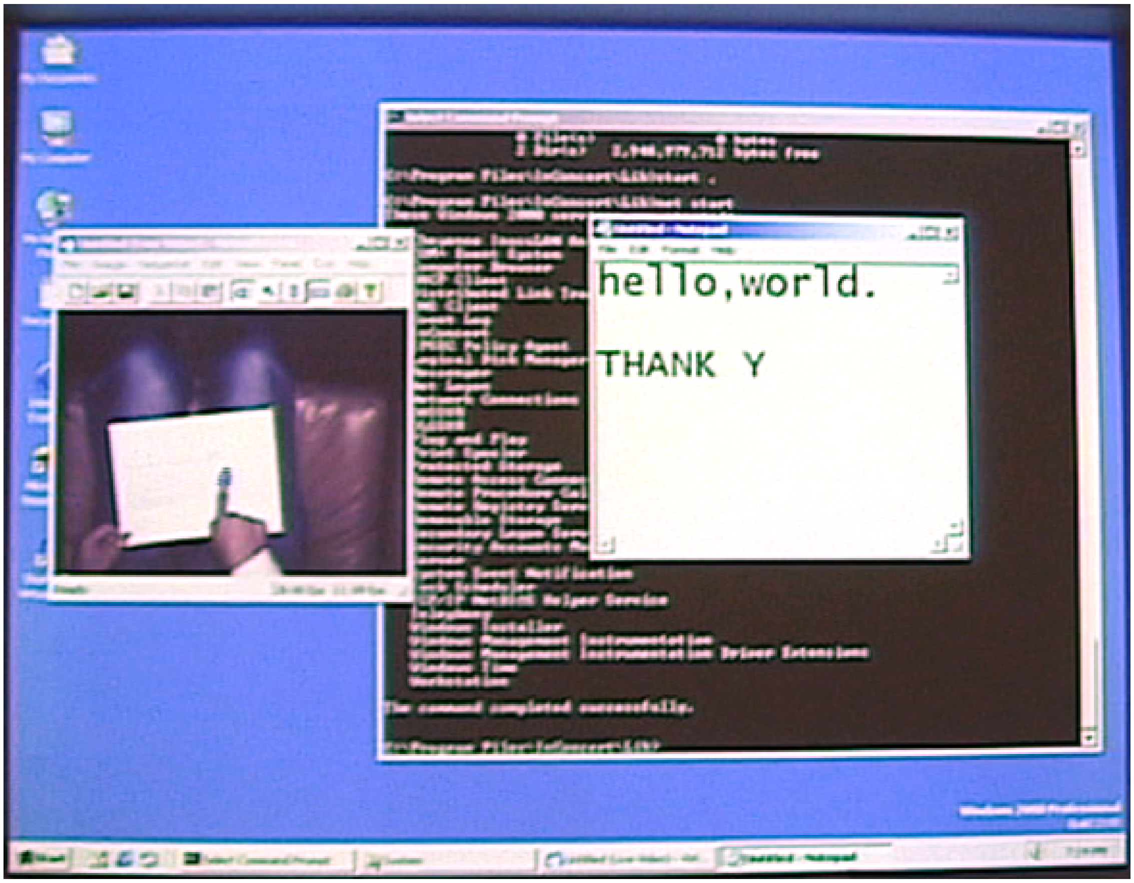
[7] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. of European Conf. on Computer Vision*, pages 343–356, Cambridge, UK, 1996.

[8] J.K.Aggarwal and Q. Cai. Human motion analysis: A review. In *Proc. of IEEE Nonrigid and Articulated Motion Workshop*, pages 90–102, 1997.

[9] K. Jo, Y. Kuno, and Y. Shirai. Manipulative hand gestures recognition using task knowledge for human computer interaction. In *Proc. of IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, Japan, 1998.

[10] N. Jojic, B. Brumitt, B. Meyers, and S. Harris. Detecting and estimating pointing gestures in dense disparity maps. In *Proc. IEEE Int'l Conf. on Face and Gesture Recognition*, Greboble, France, 2000.

[11] M. Jones and J. Rehg. Statistical color models with application to skin detection. Technical Report CRL 98/11, Compaq Cambridge Research Lab., 1998.



```
The command completed successfully.
```

hello,world.

THANK Y

Figure 10: Virtual keyboard

[12] S. Ju, M. Black, S. Minneman, and D. Kimber. Analysis of gesture and action in technical talks for video indexing. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 1997.

[13] R. Kjeldesn and J. Kender. Toward the use of gestures in traditional user interface. In *Proc. of IEEE Automatic Face and Gesture Recognition*, pages 151–156, 1996.

[14] R. Kjeldsen and J. Kender. Finding skin in color images. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 312–317, 1996.

[15] V. Pavlović, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human computer interaction: A review. *IEEE PAMI*, 19:677–695, July 1997.

[16] F. Quek. Unencumbered gesture interaction. *IEEE Multimedia*, 1997.

[17] Y. Raja, S. McKenna, and S. Gong. Colour model selection and adaptation in dynamic scenes. In *Proc. of European Conf. on Computer Vision*, 1998.

[18] J. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Proc. of IEEE Int'l Conf. Computer Vision*, pages 612–617, 1995.

[19] J. Semple and L. Roth. *Introduction to Algebraic Geometry*. Oxford: Clarendon Press, 1949. Reprinted 1987.

[20] T. Starner and et.al. A wearable computer based american sign language recognizer. In *Proc. IEEE Int'l Symposium on Wearable Computing*, Oct. 1997.

[21] G. Strang. *Introduction To Applied Mathematics*. Wellesley-Cambridge Press, 1986.

[22] J. Triesch and C. von de Malsburg. Robust classification of hand postures against complex background. In *Proc. Int'l Conf. On Automatic Face and Gesture Recognition*, 1996.

[23] C. Vogler and D. Metaxas. ASL recognition based on a coupling between HMMs and 3D motion analysis. In *Proc. of IEEE Int'l Conf. on Computer Vision*, pages 363–369, Mumbai, India, Jan. 1998.

[24] C. Wren, A. Azarbayejani, T. Darrel, and A. Pentland. Pfinder: Real-time tracking of the human body. In *Photonics East, SPIE Proceedings*, volume 2615, Bellingham, WA, 1995.

[25] Y. Wu and T. S. Huang. Capturing articulated human hand motion: A divide-and-conquer approach. In *Proc. IEEE Int'l Conf. on Computer Vision (ICCV'99)*, pages 606–611, Corfu, Greece, Sept. 1999.

[26] Y. Wu and T. S. Huang. Human hand modeling, analysis and animation in the context of HCI. In *Proc. IEEE Int'l Conf. on Image Processing (ICIP'99)*, Kobe, Japan, Oct. 1999.

[27] Y. Wu and T. S. Huang. Color tracking by transductive learning. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition (CVPR'2000)*, volume I, pages 133–138, Hilton Head Island, SC, June 2000.

[28] Y. Wu and T. S. Huang. View–independent recognition of hand postures. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition (CVPR'2000)*, volume II, pages 88–94, Hilton Head Island, SC, June 2000.

[29] Y. Wu, Q. Liu, and T. S. Huang. An adaptive self-organizing color segmentation algorithm with application to robust real-time human hand localization. In *Proc. of Asian Conference on Computer Vision(ACCV'2000)*, pages 1106–1111, Taipei, Taiwan, Jan. 2000.

[30] M. Zeller and et al. A visual computing environment for very large scale biomolecular modeling. In *Proc. IEEE Int'l Conf. on Application-Specific Systems, Architectures and Processors*, pages 3–12, Zurich, 1997.

[31] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the 7th International Conference on Computer Vision*, pages 666–673, Corfu, Greece, Sept. 1999. IEEE Computer Society Press. software is available at `http://research.microsoft.com/˜zhang/Calib/`.