使用主動式立體視覺系統發展一個徒手指向器
Development of a Free-Hand Pointer Using an Active Stereo Vision System

洪一平 Yi-Ping Hung　陳永昇 Yong-Sheng Chen
中央研究院 資訊科學研究所
Institute of Information Science, Academia sinica, Taipei, Taiwan

楊曜聰 Yao-Strong Yang　謝英博 Ing-Bor Hsieh　傅楸善 Chiou-Shann Fuh
國立台灣大學 資訊工程研究所
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan

摘要

在人與電腦的互動介面中，雙手扮演著非常重要的角色。我們最終目標是希望使用者不需藉助任何手握的工具即可用雙手與電腦溝通 。在此一研究中，我們利用一個主動式立體視覺系統來追蹤使用者的食指，並發展出一個徒手指向器，可以不須事先假設手指的長度或寬度是已知的。我們首先使用主動式立體視覺系統取得一對立體影像，再由影像中偵測出指尖在影像中的位置及手指在影像中的方向，最後再利用攝影機參數推算出手指在三度空間中的位置與方向。在本系統中，我們提供了兩種操作模式：「手指本身的指向模式」和「眼睛至手指的指向模式」。根據實驗結果，我們傾向於使用較穩定的「眼睛至手指的指向模式」。爲了使手指的活動範圍可以不受到相機視野的限制，我們採用主動式立體視覺的方法，使攝影機能夠隨著手指的運動而轉動。實驗證實主動式立體視覺的技術可以用來發展具有實用價值的徒手指向器。

主要關鍵字：立體視覺，主動式視覺，視訊追蹤，徒手指向器。

Abstract

Human hands play an important role in human-computer interface. It will be most desirable if the user is allowed to use his free hands to communicate with the computer without holding or touching any physical tool. In this work, we have developed a system of free-hand pointer by tracking the index finger with an active binocular vision system. Here, we do not need any assumptions on the length and width of finger. Both the fingertip and the finger orientation in the left and right images obtained with the binocular vision system are detected. Then, the 3D finger position and direction are computed by using the camera parameters of the vision system. Our free hand pointer can be operated in two modes when computing the projection direction: the finger-orientation mode and the eye-to-fingertip mode. The experimental results favor the eye-to-fingertip mode for its robustness. In order to allow the finger to move in a wider 3D space without reducing the pointing resolution, we utilize a well-calibrated active stereo vision system which has a relatively small field of view but can control its stereo cameras to fixate at the moving finger. Our experiments have successfully demonstrated the feasibility of developing a free-hand pointer using an active binocular vision system.

Keywords: Stereo Vision, Active Vision, Visual Tracking, Free-Hand Pointer.

1 Introduction

Human hands play an important role in human-computer interface. They usually function through keyboards, mouses, digitizers, touch panels, and so on. However, it will be most desirable if the user is allowed to use his free hands to communicate with the computer without holding or touching any physical tool. In this work, we have developed a system of free-hand pointer for presentation by tracking the finger of the speaker with an active binocular vision system.

There are many related works of human-computer interaction which utilize human hand/finger tracking. In "Charade" [1], Baudel and Beaudouin-Lafon developed a system for remote control of objects. The user was recommended to wear a data-glove, and a set of gestures were defined to represent some meaningful operations. Their system would recognize the gestures of the user and execute the associated operations. In free-hand tracking, Freeman and Weissman [12] designed a system which allowed the user to control the television by moving the open hand. Rehg and Kanade [8] defined a kinematic model for hand with 27 degrees of freedom (DOFs), and developed a system that could track articulated structures with high degrees of freedom. In [5] and [6], Kuch and Huang proposed another model to describe the human hand, and applied it to vision-based human-computer interface. In [7], Kuniyoshi et al. tracked two fingers of the user with a stereo vision system. By observing the user's operation, their system could learn the task and then command a robot to do some assembly jobs.

Another interesting system is the "Finger-Pointer" [3], which provided multi-modal computer interface. In that system, users communicated with the computer by gestures and voice. It could recognize some simple predefined gestures and the pointing direction of the finger. However, the system assumed that the finger length and width were known *a priori*, which might limit the flexibility and reliability of their system.

Hand/finger tracking has many applications on human-computer interaction. In this paper, we focus on applying finger tracking to free-hand pointer for presentation. In an oral presentation, one usually uses a rod pointer or a laser pointer to point at a site on the blackboard or on the projection screen where he wants the audience to pay their attention to. However, it is more comfortable and convenient if we can design a computer vision system to understand where the presenter's finger is pointing to. Our free-hand pointer system does not need to know the finger length and width in advance. We use stereo vision to compute 3D finger position and direction without building a complicated 3D hand model. The methods used to detect the finger position and to compute the finger direction are simple and efficient. A well-calibrated active binocular vision system – the IIS (Institute of Information Science) head – helps to track and fixate the moving finger in a wider 3D space.

The system used in the experiments consists of a Sparc Station 20, two CCD cameras, a Datacell S2200, and the IIS head. The IIS head is a computer-controllable robot head on which two CCD cameras are mounted. The IIS head and the two cameras form our stereo vision system which is precisely calibrated such that the 3D estimation error is less than 1 mm [9] for an object locating at one meter away from the IIS head. Datacell S2200 is used to grab the stereo images, and the workstation is used for image processing.

In the following, we will describe our free-hand pointer whose block diagram is shown in Figure 1. First, the left and right images taken with our binocular head are binarized with Otsu's method. In the initial stage, i.e., for the first image, we use mathematical morphology to detect and segment the finger, as described in section 2.1. Local search with prediction is used to efficiently track the finger

in the subsequent images, which is described in section 2.4. Time-consuming global search of finger will be invoked again only if the system loses track of the finger. Once the finger is detected and segmented, we use a simple and efficient algorithm to extract the fingertip and the finger orientation, as described in sections 2.2 and 2.3. Then, with the calibrated camera parameters, 3D fingertip position and finger direction can be computed by using the 2D fingertip position and finger direction obtained from the left and right images. We propose two ways for computing the 3D projection site on a given projection plane, which will be described in section 3. Once the 3D projection site is determined, a cursor (or a mark) can be projected onto that site accordingly. If the detected fingertip moves toward the image boundary, the cameras will be controlled to fixate at the fingertip so that the fingertip can remain within the field of view, as describe in section 4.

The paper is organized as follows. Section 2 describes the details of the methods used to extract the fingertip and the finger orientation in 2D images. Section 3 explains how we determine the site on the projection screen where a cursor is to be projected onto. In the above two sections, we tentatively assume that the stereo cameras are fixed. However, this assumption greatly limits the 3D space where the finger can move around. To allow the finger to move in a wider 3D space without reducing the pointing resolution of the 3D free-hand pointer, we utilize a well-calibrated binocular head (i.e., the IIS) to fixate at the moving finger, which is described in section 4. Finally, some experimental results are given in section 5 and conclusions in section 6.

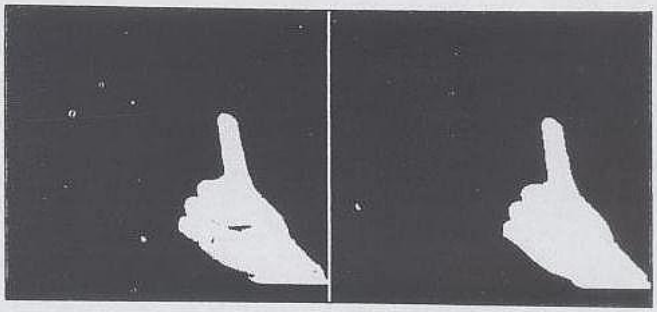

Figure 2: The left image is the original gray level image (512 × 480) and the right is the binarized image with threshold (= 85) determined by Otsu's method.

2 Extraction of Fingertips and Finger Orientation in 2D Images

2.1 Finger Detection

After grabbing a pair of stereo images, we first detect where the finger is in both images. The following is our algorithm for finger detection:

1. Image binarization.
2. Morphological closing for reducing noise.
3. Morphological opening for detecting finger.
4. Labeling and identification of the finger.

Suppose the background is much darker than the hand, the contrast of the image is very sharp and it is relatively easy to select a threshold for binarization. We assume the black and white parts of the image will form two Gaussian distributions, and then apply Otsu's method [4] to determine the threshold automatically. The resulted image is shown in Figure 2.

Sometimes, the image of the hand may have some shadows, which may significantly affect the results of the subsequent processing.

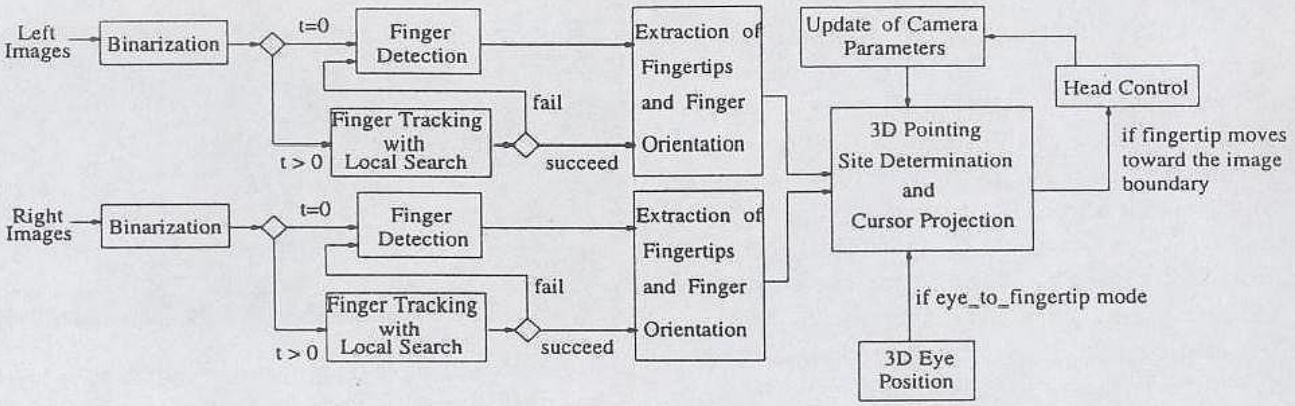

Figure 1: The block diagram of our free-hand pointer.

Hence, we first use a morphological closing to remove the noise.

Next, we want to extract the finger from the image. We know that the finger is the thin part compared with the palm. Hence, the finger can easily be separated from the palm by using a morphological opening. We extract the finger with the operation: $\mathbf{I}' = \mathbf{I} - \mathbf{I} \circ \mathbf{K}$, where $\mathbf{I}'$ is the resulted image, $\mathbf{I}$ is the original image, and $\mathbf{K}$ is the structuring element used for opening.

To speed up the system, we assume that the pointing direction of the finger is approximately vertical in the first pair of stereo images (i.e., an initial finger position). Hence, we can use a 1D structuring element for opening rather then a 2D structuring element. The 1D opening operation is much faster than the 2D opening operation. Besides, the result obtained by using the 1D opening is less noisy. Figure 3 shows the result. The size of the structuring element can be chosen according to the threshold determined by applying Otsu's algorithm to the histogram of the length of horizontal segments. Here, the horizontal segments are obtained by crossing a horizontal scanline through the binary image containing the finger.

Since the background is dark enough in our experiments, the result of the morphological finger detection is not very noisy, and we only have to identify the largest connected component to be the target (i.e., the finger).

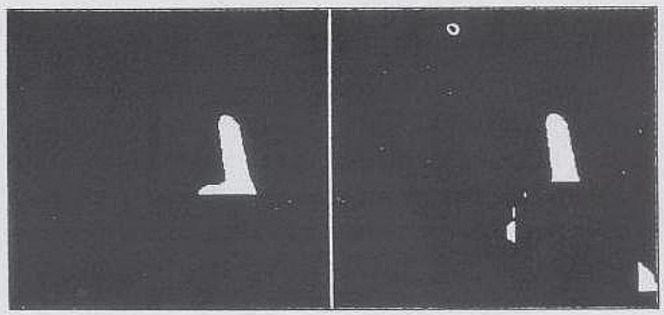

Figure 3: The left image is the result of 1D opening with kernel size of 95. The right image is the result image of 2D opening with kernel size of 47×47.

2.2 Extraction of Fingertip

To locate the fingertip, we first define two types of finger edge points, *Group-A* points and *Group-B* points, which are described below. In the initial mode, *Group-A* points are the edge points of the finger that are connected with the palm part, and *Group-B* points are the edge points of the finger that are not connected with the palm, as described in Figure 4. In the tracking mode, the definition of *Group-A* and *Group-B* points has to be modified because we do not process the entire image but only a partial image contain-

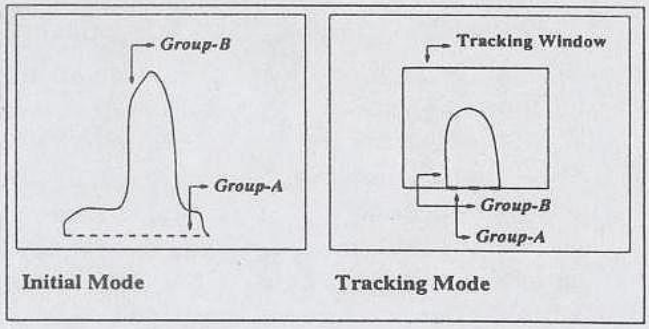

Figure 4: Two types of finger edge points, *Group-A* and *Group-B*, in the initial mode (left) and the tracking mode (right).

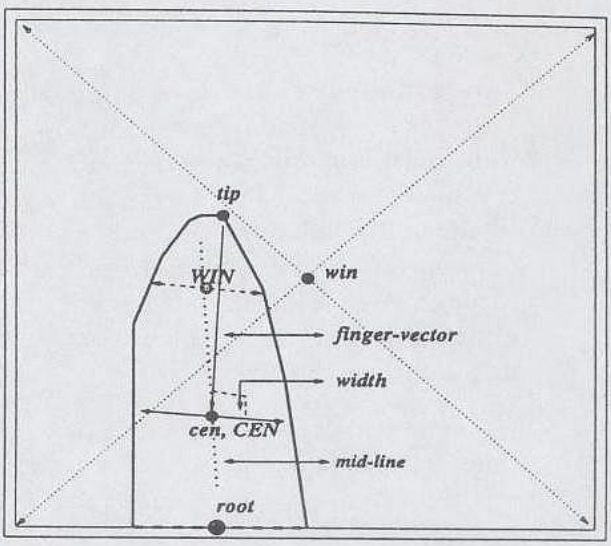

Figure 5: The feature points and the mid-line of a finger.

ing the finger. In the tracking mode, we define *Group-A* points to be the edge points of the finger that are connected with the window boundary, and *Group-B* points to be the edge points that are not in *Group-A*.

Next, we define *root* to be the centroid of *Group-A*, and *tip* to be the farthest edge point from *root* and belonging to *Group-B*. Notice that, once the mid-line of the finger is computed as described in section 2.3, the *tip* will be modified to be *tip'*, the intersection-point of the mid-line and *Group-B* edge points. As shown in Figure 5, we define *cen* to be the finger centroid, *win* to be the center of the current tracking window, *finger-vector* to be the vector from *tip* to *cen*, and *width* to be the length of the segment passing through *cen* of the finger and orthogonal to the finger vector. To define *WIN*, the starting point for tracing the mid-line, we first locate the point which is *width*/2 away from *tip* along the *finger-vector*. Then, the position of this point is modified to be the middle point along the vector orthogonal to the finger-vector, and is then denoted as *WIN*. This point *WIN* will be used for determining the mid-line of the finger, as described in section 2.3, which will be used for determining both the fingertip and the finger orientation.

2.3 Extraction of Finger Orientation

Our method for determining the mid-line (or skeleton) of the finger starts from the image point *WIN*. In order to reduce the computation time, we do not locate all the points on the mid-line. Instead, we locate only a few candidate points on the mid-line and then determine the mid-line of the finger by fitting a line to those candidate points. Our algorithm is as follows:

1. Start from *WIN*, which is obtained in section 2.2. Choose WIN to be the first candidate point on the mid-line.
2. Temporarily choose the next candidate point (only a tentative candidate point) to be the point which is *width*/4 away from *WIN* and is along the direction from *tip* to *WIN*.
3. Next, we find the line segment which is orthogonal to the direction from *tip* to *WIN* and passing through the tentative candidate point found in step 2. Modify the second candidate point to be the midpoint of the above orthogonal line seg-

ment.

4. Locate the next candidate point by following the direction determined by the previous two candidate points for another distance of $width/4$. Then, modify it to be the mid-point along the orthogonal direction. The width of the finger passing through this point along the orthogonal direction is computed, which is to be used in the test in the next step.

5. If the width obtained in step 4 is larger than a threshold determined by Otsu's method (i.e., out of the finger part), go to step 6. Otherwise, repeat step 4.

6. Compute the 2D mid-line of the finger by fitting a straight line to those candidate points found above.

Once the mid-line of the finger is determined, the fingertip (tip') can be obtained by finding the intersection point of the mid-line and the *Group-B* edge points. Another use of the 2D mid-line is to compute the 3D mid-line (or skeleton) using stereo triangulation. This 3D mid-line can then be regarded as the 3D finger orientation and used in the finger-orientation mode, which will be described in section 3.

2.4 Finger Tracking

Because processing the whole image for each frame is very time-consuming, we adopt two search schemes to reduce the processing time.

2.4.1 Scheme I: Tracking by Local Search

This tracking scheme is based on local search. The goal of this scheme is to make sure that the center of the tracking window is always located near the fingertip. One idea is to use the displacement ($Disp$) of the centroid of the detected finger to estimate the movement of the finger (Mov) and then adjust the position of the tracking window accordingly. However, without performing finger detection (in a more global sense), it is not trivial to compute the real $Disp$. In fact, if we set the center of the initial tracking window to be WIN of the previous frame (which is obtained in section 2.2), and compute the centroid by considering only the part of the finger trimmed by this initial tracking window in both the current and the previous images, $Disp$ and Mov may be quite different, especially when the finger motion is along the *finger-vector* (see the right figure in Figure 6 for an example). Notice that the movement of the finger along the *finger-vector* is not equal to the displacement of the finger centroid, which is because the tracking window is not shifted yet and hence the finger area in the new image will be quite different from that in the previous one. There can be many ways to deal with the above problem. The following is just a simple rule for choosing Mov', the movement of the tracking window (see Figure 7).

1. If the $Disp$ is smaller then $width/2$, Mov' is set to be $2 \times Disp$.

2. If the $Disp$ is larger than $width$, Mov' is set to be $Disp$.

3. If the $Disp$ is between $width/2$ and $width$, Mov' is set to be $width$.

In our experiments, the tracking window (or the trimming window) is set to have the width of $3 \times width$ and is initially centered at WIN of the previous frame. Within this initial tracking window, we compute the centroid of the trimmed finger for both the current and previous frames, and accordingly, update the tracking window by Mov', as described above. Once the tracking window is updated to a better position, we can then extract the fingertip and determine the mid-line (skeleton) of the finger within this new tracking window (supposedly better centered) using the algorithms

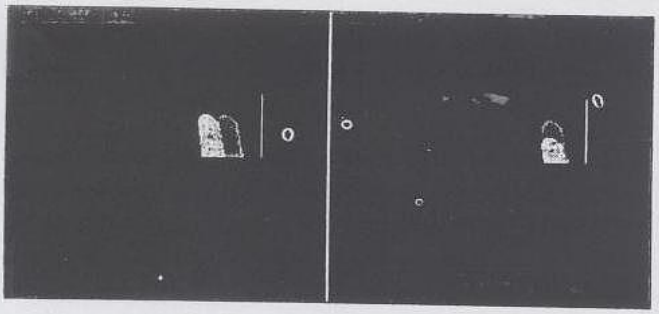

Figure 6: The movement of the finger versus the displacement of the finger centroid. The darker finger is the original finger and the black points are the centroids of the two trimmed fingers. The left shows the finger moves leftward 30 pixels and the centroid displacement is approximately the same. The right shows the finger moves downward 30 pixels, but the centroid displacement is only approximately 15 pixels.

described in sections 2.2 and 2.3. During the process of extracting the fingertip, WIN (i.e., the first candidate point on the mid-line) will be found and will be used as the center of the tracking window for the next frame. This process goes iteratively until it fails to track the finger. If this happens, we switch to scheme II described below.

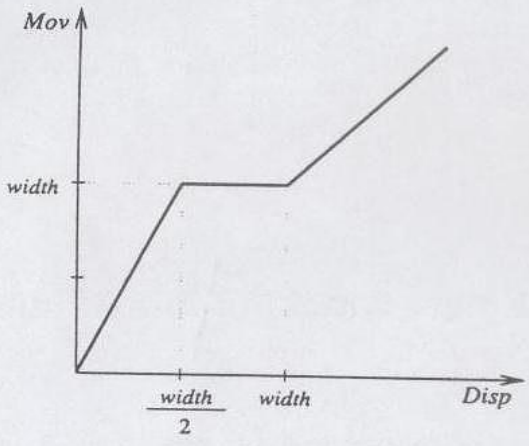

Figure 7: The rule for determining the movement of the tracking window Mov' from the displacement of the finger centroid $Disp$.

2.4.2 Scheme II: Tracking by Hierarchical Global Search

The above local search scheme will work well under the assumption that the movement of the finger between two frames is not too large. It fails if the finger moves too fast. In order to solve this problem, we can use another tracking scheme, tracking scheme II, which is based on the hierarchical (coarse-to-fine) concept. The algorithm is as follows.

1. For each new image pair, we first subsample the image from 512 × 480 to 64 × 60, and compute the approximate positions of the *root* and *tip*, using the algorithms described in sections 2.1 and 2.2.

2. Move the center of tracking window to the approximate *tip* and the finger is surely over there. Then, extract the fingertip and the mid-line in the fine level (the 512 × 480 level) using the algorithms described in sections 2.2 and 2.3.

3 Determination of Projection Site

Once the 2D position of the fingertip and the 2D orientation of the finger in both the left and right images are obtained (with the methods described in section 2), we can compute the 3D position of the fingertip and the 3D orientation of the finger using the calibrated camera parameters of the stereo cameras. In the following, we propose two approaches to determine the pointing direction of the finger, which can be used to compute the projection site of the cursor (or the mark). One is the *finger-orientation* mode which computes the pointing site by using the finger orientation and position of the fingertip. The other is the *eye-to-fingertip* mode which determines the

pointing site by computing the 3D ray determined by the eye position (the pseudo origin) and the fingertip position. Details of these two modes for determining the 3D pointing ray are described below. As soon as the 3D pointing ray is determined, the projection site can be easily computed by intersecting the 3D pointing ray with the projection screen.

3.1 Finger-Orientation Mode

In this section, we introduce the first mode of our free-hand pointer system, the *finger-orientation* mode, which uses the information of the finger orientation. The following is the algorithm used for *finger-orientation* mode:

1. Select several points on the mid-line of the finger in the left image.
2. Compute their corresponding points by finding the intersection points of their epipolar lines in the right image and the mid-line of the finger in the right image.
3. Compute the 3D positions of these selected points, and fit a 3D line to them. Then, we can determine the pointing site by intersecting the 3D line with the projection screen.

3.2 Eye-to-Fingertip Mode

In the *finger-orientation* mode, the resolution of the finger pointer is not very high. Also, its performance is quite sensitive to noise. Hence, we propose another mode, the *eye-to-fingertip* mode, to solve this problem. In the *eye-to-fingertip* mode, we first detect the 3D eye position of the speaker and set this position as the origin. Then, we compute the position of the fingertip, and determine a vector from the eye to the fingertip. Thus, we can determine where the vector is pointing to on the projection screen. Since the distance between the fingertip and the eye is much longer than the length of the finger, its pointing performance is less sensitive to noise. A minor disadvantage is that the user has to raise up his (or her) hand high enough such that the finger is between the eye and the projection screen. The following is the algorithm used for the *eye-to-fingertip* mode:

1. Determine the eye (pseudo origin) position:
   (a) Move the fingertip to any two points between the center of projection screen and the right eye (or the left eye) and compute their positions. Hence, we can get the direction vector between the eye and the center of projection screen.
   (b) With the knowledge of the average distance between the eye and hand, we can have a rough estimate of the 3D eye position. Regard it as the origin.
2. While moving the fingertip, keep on detecting the 3D position of the fingertip, and compute the intersection point of the eye-to-fingertip vector and the projection screen.

4 Finger Tracking in a Wider Space

For visual tracking systems using fixed cameras, the field of view of the camera limits the 3D space where the target can be tracked. This limitation can be removed if the camera can be controlled to fixate at the moving target. This kind of vision system with movable camera is called an active vision system.

Another approach to remove this space limitation is to use a lens (or a camera system) with a wider field of view. However, this approach will make the target look smaller in the images, thus reduce the resolution and accuracy in extracting the 3D position of the target. In the application of free-hand pointer for presentation, the pointing resolution is quite important. Hence, we adopt the approach of using an active vision system.

In our laboratory, we have mounted two cameras on a six DOFs robot head (referred to as the IIS head). The IIS head has four revolute joints and two prismatic joints, as shown in Figure 8. The top two joints of the IIS head are for camera verge (or gazing). The next two joints are for tilting and panning the stereo cameras. All of the above four joints are revolute and are mounted on an X-Y table which is composed of two prismatic joints. The lenses of the binocular head are motorized to focus on objects at different distances. Due to the difficulty in acquiring accurate kinematic and camera parameters, the use of active vision system is not very popular yet. However, our IIS head has been calibrated with a four-stage method [9, 11, 10] and can achieve a very high accuracy which will induce only one pixel prediction error and 0.2 pixel epipolar error. Therefore, it is easy and straightforward for us to use the IIS head to track and fixate at the finger of the speaker in a much wider 3D space.

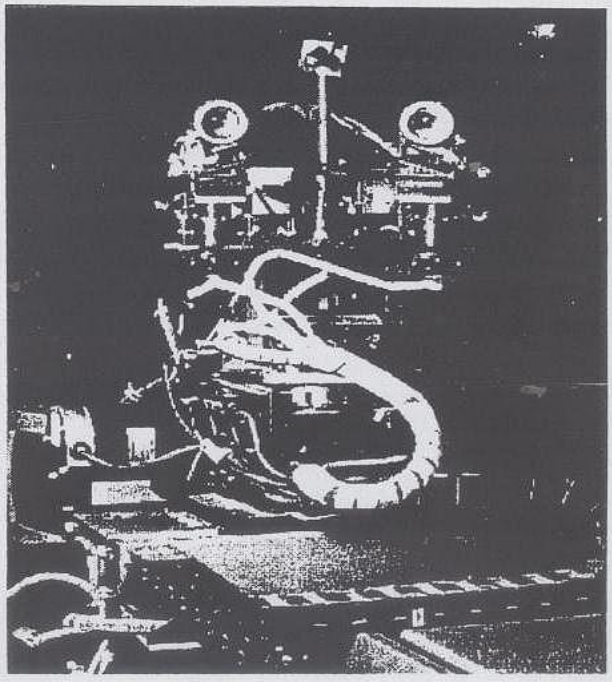

Figure 8: The IIS head used in our experiments.

5 Experiments

First, we compare the computational cost and the flexibility of the two tracking schemes described in section 2.4. As can be seen from Table 1, more than half of the computational time was spent on image transfer. In scheme I, only a small portion of the image is transferred to the computer, while in scheme II, the whole image is transferred. Therefore, scheme I is much faster than scheme II, but it is less flexible because the finger is not allowed to move too much between two consecutive frames. One strategy is to track the finger with scheme I most of the time, and automatically switch to scheme II when the system loses track of the finger. However, the time for image transfer will be greatly reduced after we use a frame grabber with faster image transfer rate.

We will next show some experimental results for the two pointing modes, the finger-orientation mode and the eye-to-finger mode. Figure 9 illustrates the operation of our free-hand pointer in the eye-to-fingertip mode. Stereo images taken with the IIS head are used to compute the 3D finger position and direction based on the known camera parameters. A mark is projected on the projection screen at the site computed from the fingertip position, pre-determined eye position, and the given position and orientation of the projection screen. Figures 10 and 11 show the result of using the *eye-to-fingertip* mode to write a "W".

Figures 12 and 13 show the result of using the *finger-orientation* mode to write a "Y". In this mode, user's finger direction should be as perpendicular as possible with the baseline of the left and right cameras. If the direction of the finger in the image is almost parallel with the epipolar line, it will cause some difficulty in determining 3D finger direction. There is no such limitation for the *eye-to-fingertip* mode because only the fingertip is used to compute the 3D finger direction by use of a pre-determined eye position.

Table 1. Computational time for two different tracking schemes. (unit: second)

Time Required	Image Transfer	Image Processing	Result Displaying
Scheme I	0.10	0.08	0.02
Scheme II	0.30	0.20	0.02

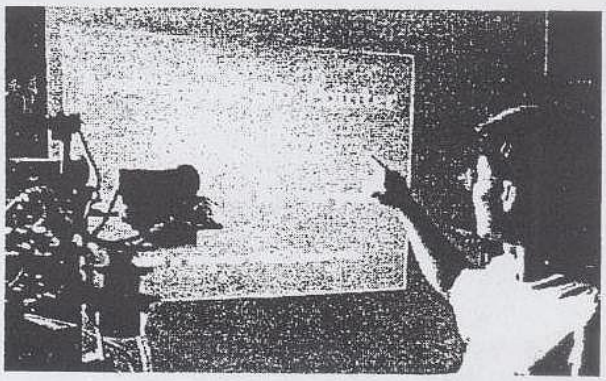

Figure 9: This figure illustrates the operation of our free-hand pointer. The speaker is pointing his index finger to the projection screen (under the word "3D"). The IIS head on the left takes the stereo image pairs to be processed. According to the projection site determined from the stereo image pairs, the computer then controls a PC to project its cursor to the word "3D"

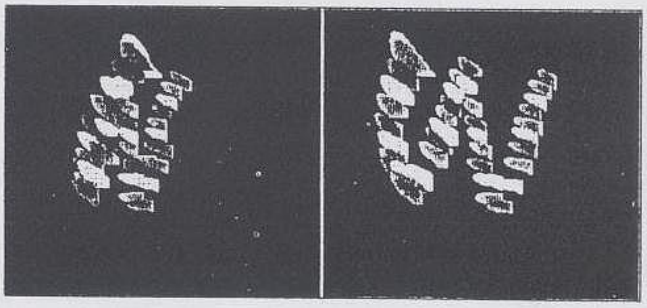

Figure 10: The image trajectory of the finger in the stereo image pairs when using the *eye-to-fingertip* mode to write a "W".

Figure 11: The pointing trajectory on the projection screen when using the *eye-to-fingertip* mode to write a "W".

6 Conclusions

In this work, we have developed a system for human-computer interaction, namely, the free-hand pointer, by using the finger tracking technique based on an active binocular vision system. We have proposed two modes when implementing the free-hand pointer—the *eye-to-fingertip* mode and the *finger-orientation* mode. The system can automatically determine the threshold for image binarization and the kernel size for morphological opening. By defining two types of finger edge points, we can extract the finger features more easily and faster. The space limitation of the finger movement is removed by using a well-calibrated active binocular vision system – the IIS head. Our experiments have successfully demonstrated that the free-hand pointer is feasible, and can be implemented with an active binocular vision system.

In this paper, we assume that the finger is bright and the background is dark. It is a challenging work to extend the system to a

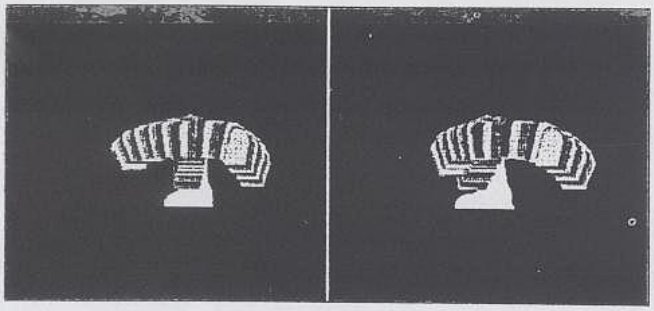

Figure 12: The image trajectory of the finger in the stereo image pairs using the *finger-orientation* mode to write a "Y".

Figure 13: The pointing trajectory on the projection screen when using the *finger-orientation* mode to write a "Y".

more complex background, and we are working on it now. Eventually, for this work to be practical and useful, one should build an active binocular vision system with a much larger stereo baseline and mount it on the ceiling to have a better viewpoint. Each camera should be able to pan and tilt independently. The zooming and focusing of the lens should be able to be controlled by the computer. It will be even better if one can use a fixed camera having a wide-angle lens (or even a omni-directional lens) to cover a wide 3D space where the presenter can move around. This camera can be put at the center of the stereo baseline, and its images can be used to control the motion of the two active cameras. Another application of this work is to use this system as an input device for a 3D signature recognition system, where the 3D signature is the trajectory drawn by the user's finger in the 3D space. We are currently investigating its feasibility.

Acknowledgements

The authors could like to thank Dr. Wen-Liang Hwang and Dr. Sheng-Wen Shih for many useful discussions and suggestions concerning this work. This work was partially supported by the National Science Council under grant NSC85-2213-E-001-016.

References

[1] T. Baudel and M. Beaudouin-Lafon, "Charade: Remote Control of Objects Using Free-Hand Gesture," *Communications of the ACM*, Vol. 36, pp. 28-37, 1993.

[2] M. Fukumoto, K. Mase, and Y. Suenaga, "Real-Time Detection of Pointing Actions for a Glove-Free Interface," *Proceedings of IAPR Workshop on Machine Vision Applications*, Tokyo, 1992.

[3] M. Fukumoto, Y. Suenaga, and K. Mase, "'Finger-Pointer': Pointing Interface by Image Processing," *Computer and Graphics*, Vol. 18, pp. 633-642, 1994.

[4] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Vol. I, Addison-Wesley, Reading, MA, 1992.

[5] J. J. Kuch and T. S. Huang, "Vision Based Hand Modeling and Tracking for Virtual Teleconferencing and Telecollaboration," *Proceedings of the 9-th International Conference on Computer Vision*, 1994, pp. 666-671.

[6] J. J. Kuch and T. S. Huang, "Virtual Gun: a Vision Based Human Computer

Interface Using the Human Hand," *Proceedings of Machine Vision Applications Workshop*, Kawasaki, Japan, 1994.

[7] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance," *IEEE Transactions on Robotics and Automation*, Vol. 10, pp. 799-821, 1994.

[8] J. M. Rehg and T. Kanade, "Visual Tracking of High DOF Articulated Structures: an Application to Human Hand Tracking," *Proceedings of European Conference on Computer Vision*, Heidelberg, pp. 35-46, 1994.

[9] S. W. Shih, Y. P. Hung, and W. S. Lin, "Calibration of an Active Binocular Head," to appear in *IEEE Transactions on Systems, Man and Cybernetics*, Volume 28, Part A, Number 3, May 1998.

[10] S. W. Shih, Y. P. Hung, and W. S. Lin, "Kinematic Parameter Identification of a Binocular Head Using Stereo Measurements of Single Calibration Point," *Proceedings IEEE International Conference on Robotics and Automation*, Nagoya, Japan, pp. 1796–1801, 1995.

[11] S. W. Shih, Y. P. Hung, and W. S. Lin, "Head/Eye Calibration of a Binocular Head by Use of Single Calibration Point," in *IEEE Southwest Symposium on Image Analysis and Interpretation*, Dallas, Texas, pp. 154–159, 1994.

[12] W. T. Freeman and C. D. Weissman, "Television Control by Hand Gestures," *Proceedings of International Workshop on Automatic Face-and-Gesture-Recognition*, Zurich, pp. 179-183, 1995.