A Real-time Monocular Vision-based 3D Mouse System

Sifang Li, Wynne Hsu, Pung H. K.
Department of Information Systems and Computer Science
National University of Singapore
Singapore 119260

Abstract

Speed, robustness, and cost are three important factors that determine the success of a vision-based human-computer interaction system. In our system, we introduce a scheme that uses only one camera (together with a mirror) to derive the 3D coordinates of a target object. To ensure a more natural interaction between the human and the computer, our system allows the user to use his/her bare fingers to point to any position in the 3D space. In addition, robustness and speed is achieved through the use of the chain-code algorithm. The accuracy of the results is improved through the use of suitable post-processing filters. The performance of our system is thoroughly evaluated.

1 Introduction

With the increased popularity of 3D applications, researchers are focusing more and more on efficient yet cost effective 3D interaction solutions. Most existing 3D interaction solutions are electromagnetic-based. One example of an eletromagentic-based 3D input device is the DataGlove. Compared to electromagnetic 3D input devices, the chief advantage of a vision-based solution is that it frees the user from any form of physical attachment that may constraint his/her movement. The vision-based approach has attracted much effort in computer vision fields [1]-[7]. However, current vision-based solutions are too sensitive and computationally too expensive for practical applications, and many of them impose unacceptable constraints on the user. In this paper, we propose a novel 3D vision-based solution using only one camera and a mirror. A number of experiments have been performed to show that our solution is *robust to movement*; has a *fast pick-up* rate so as to allow the user's hand to move in and out of the scene just as s/he tends to pick up a normal 2D mouse now and then; and has a *low computational cost* – thus resulting in fast response time. In addition, it is also *robust to false recognition.* This is important because no algorithm is able to achieve 100% accuracy rate. Thus, recovery from false recognition is critical. Finally, since our solution uses only one camera and one mirror, it satisfies the *low cost* criterion.

2 Related Work

Vision-based 3D mouse was mentioned a number of times in recent publications. Most of them rely on 2D shape models [2,3] (Kalman Snakes [8]) or 3D anatomic models [4]-[6] to guide in deriving the measurements of an image. Movement

models such as smoothness, slowness, affinity or periodic properties have also been used to predict the next model state. However, when it comes to applying these models in practical human computer interaction (HCI) applications, a number of problems surface. The first problem is the difficulty in initiating the model. For practical HCI application, the user's hand should be allowed to re-enter the scene easily. This implies that model initiation needs to be done each time the user re-enters the scene. Such initiation is very expensive without introducing additional constraints such as fixed background [4]-[7] or arranged initialization area [2,3]. The second problem is the inability to restore from false recognition. In the model-based approach, estimation of the image is directly computed from the previous result. If the result is false due to the inaccuracy of the feature measurement, lock-loss is possible. Once the lock is lost, regaining the lock is difficult because the initiation of the model is expensive. In [2], learning is used to prevent lock losing, but this only applies to those movements which are used to train the system. Given the fact that the false recognition in a vision-based system is inevitable and the user's movement cannot be constrained too much, model-based approach is not suitable for our application.

To reduce hardware cost, people try to use only one camera to extract 3D position instead of stereo camera systems [3,7]. Monocular 3D position detection has been addressed in the literature [2,4,5,6]. Their approaches fall into two broad categories:

i. Using depth cues to restore 3D depth information indirectly

The cues can be used are shape size, shading and texture. In [2] the area within the hand contour is used as the depth cue. This approach assumes the hand doesn't turn around before the camera. We can not expect that using size, shading and texture as depth cue can provide accurate outputs.

ii. Using relative 3D structure to restore the depth

[4]-[6] are examples which try to restore the detailed hand or arm gestures using 3D hand or arm models. After the 3D model being successfully restored, one part of the model's 3D position can be determined by introducing some constraints (static should [4]; palm on a plane [6]). The constraints introduced in these works are too strong from the user's point of view. Actually in [4], the author also mentioned that users could not maintain their shoulder static at all and the shoulder's movement caused significant error in the final 3D coordinate outputs.

3 System Setup and Architecture

3.1 System Setup

To accurately determine the 3D position of an object, we need at least two views. In our system, a novel method using one camera and a mirror is introduced. The physical setup is shown in Figure 1. A video camera (VC) is placed at an angle of 45° to the mirror. We build up XYZ and UVW coordinate systems in the 3D space. XY plane is actually VC's image plane and UW plane is parallel to the mirror. The output of our 3D mouse refers to the UVW system, which is more natural for the user, instead of XYZ. A (real) object is presented somewhere in the space, we are going to derive the 3D coordinates of it.

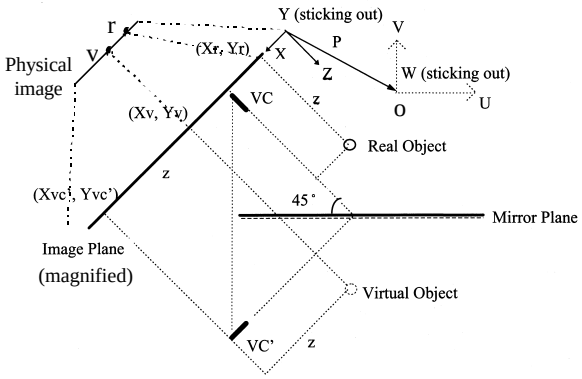

Figure 1. The System Setup.

On the XY plane, let the projection of the real object be (x_r, y_r) and the projection of the mirror image be (x_v, y_v). We call this mirror image the corresponding *virtual object* of the real object. Let's assume that we also know the projection $(x_{vc'}, y_{vc'})$ of the virtual camera VC'. We can then decide the 3D position of the real object by using these three projections - (x_r, y_r), (x_v, y_v) and $(x_{vc'}, y_{vc'})$.

It's straightforward that the X and Y positions of the real object are x_r and y_r respectively. From the symmetric property provided by the mirror, we have

$z = x_{vc'} - x_v$.

By a simple geometry transform, we transform the real object's XYZ co-ordinates into the UVW coordinate system (see Figure 1) where P is the translation vector that maps the origin of the XYZ coordinate system to the origin of the UVW coordinate system.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} -\sin\frac{p}{4} & 0 & \cos\frac{p}{4} \\ -\cos\frac{p}{4} & 0 & -\sin\frac{p}{4} \\ 0 & 1 & 0 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \end{bmatrix} + P = \begin{bmatrix} -\sin\frac{p}{4} & 0 & \cos\frac{p}{4} \\ -\cos\frac{p}{4} & 0 & -\sin\frac{p}{4} \\ 0 & 1 & 0 \end{bmatrix} \bullet \begin{bmatrix} x_r \\ y_r \\ x_{vc'} - x_v \end{bmatrix} + P \quad (1)$$

Consider the starting point O(0,0,0) of UVW system. We denote the XY coordinates of the projection of O and the projection of its corresponding virtual object as (x_{r0}, y_{r0}) and (x_{v0}, y_{v0}), respectively.

From (1), we have (2).

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\sin\frac{p}{4} & 0 & \cos\frac{p}{4} \\ -\cos\frac{p}{4} & 0 & -\sin\frac{p}{4} \\ 0 & 1 & 0 \end{bmatrix} \bullet \begin{bmatrix} x_{r0} \\ y_{r0} \\ x_{vc'} - x_{v0} \end{bmatrix} + P \quad (2)$$

Subtract (2) from (1), we get (3).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} -\sin\frac{p}{4} & 0 & \cos\frac{p}{4} \\ -\cos\frac{p}{4} & 0 & -\sin\frac{p}{4} \\ 0 & 1 & 0 \end{bmatrix} \bullet \begin{bmatrix} x_r - x_{r0} \\ y_r - y_{r0} \\ -(x_v - x_{v0}) \end{bmatrix} \tag{3}$$

Now we can restore the 3D position of the object using (3) provided we know the XY projections of the following points: UVW origin O and its mirror image, the object and the virtual object. Because the XY plane is actually the video camera image plane, we can consider a physical video image frame is miniature of the projections of all things before the camera (A orthographic camera model is assumed here) . So any point's projection can be found in the video image, if the camera can "see" the point.

After we set up the system, (x_{r0}, y_{r0}) and (x_{v0}, y_{v0}) are fixed. As long as the camera can "see" the object (x_r, y_r) and its virtual object (x_w, y_v), we know the object's 3D co-ordinates in UVW system using (3). With this method, we merely need one camera to obtain the accurate 3D position. This results in substantial cost saving.

3.2 Overview of the System Architecture

Figure 2 shows an overview of the system architecture. The initialization module is responsible for building up the background models describing the intensities properties of each background pixel. Next, a target object is introduced into the scene and an object model is built for this target. When all models have been built, the chain-code algorithm is invoked to detect the contour of the target object. This contour is then analyzed and interpreted. Finally, post-processing is performed to obtain a more accurate 3D position of the target object. In our implementation, the target object is a human hand. The index finger is used to specify the desired 3D position while the thumb is used to specify the desired operation. For example, an erected thumb signifies the releasing of the mouse button while a bent thumb signifies the pressing of the mouse button (see Figure 3).

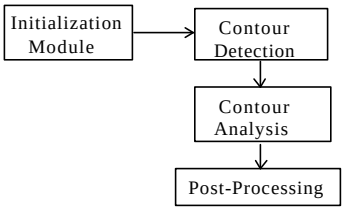

Figure 2. Overview of System Architecture

Figure 3. The Two Thumb States

3.3 Initialization Module

Background knowledge is probably the most powerful knowledge in video segmentation. In our system, we assume a fixed but complex background, and instead

of using the hand shape knowledge, we use prior background and hand intensity knowledge to perform feature measurement, which guarantees the robustness for a practical 3D mouse. The background and hand intensity knowledge are modeled using simple Gaussian models. To learn the background models, an initial startup time of n continuous frames is required to build the Gaussian models:

$$m(x,y)=\frac{1}{n}\sum_{k=0}^{n-1} i_k(x,y), \quad s^2(x,y)=\frac{1}{n}\sum_{k=0}^{n-1} i_k(x,y)^T i_k(x,y)-m(x,y)^T m(x,y) \tag{4}$$

where *i* is the intensity vector expressed in RGB space.

Based on the background Gaussian models, a single Gaussian model describing the hand intensity is built up by placing the hand before the learned background. Hand pixel segmentation is achieved by:

$$f(x,y)=\begin{cases} i(x,y), & if \quad |m(x,y)-i(x,y)|>ts(x,y) \\ 0, & if \quad |m(x,y)-i(x,y)|\le ts(x,y) \end{cases} \tag{5}$$

where τ is the threshold value.

After the segmentation, the Gaussian model for all hand pixels can be built in a similar way:

$$m_h=\frac{1}{n_h}\sum_{f(x,y)\neq 0} f(x,y), \quad s_h^2=\frac{1}{n_h}\sum_{f(x,y)\neq 0} f(x,y)^T f(x,y)-m_h^T m_h \tag{6}$$

where n_h is the total number of non-zero pixels in segmented image f.

Using these models, we can classify all the pixels in an image by thresholding (7).

$$\frac{P(I|H)}{P(I|B)}=\frac{s}{s_h}\exp\left(\frac{|i-m|^2}{2s^2}-\frac{|i-m_h|^2}{2s_h^2}\right) \tag{7}$$

where H, B and I denote the following events respectively: the pixel belongs to the hand (H); the pixel belongs to the background (B) and the pixel intensity is I (I).

Figure 4 shows the results when we compute (7) over all image pixels. The pixel brightness reflects the magnitude of the results (value of the right side of (7)). The bright area in the figure is the hand and its mirror image with some noise caused by the shade and illumination. Some pixels in the hand is not very bright because they are too similar to the local background.

3.4 Detection of Hand Contour

The chain code algorithm [10] does not require any prior shape knowledge is thus robust to fast and drastic changes in shape. Note that the chain code generation are usually carried out on a grid that is imposed on the image. In [11], Scholten et. al proved theoretically that a hexagonal grid can describe a curve more accurately as compared to the triangle or square grid. The chain coding algorithm on hexagonal grid is given below.

Algorithm chain-code

Let N and V be two functions where

$N_j(C)$ returns the j^{th} neighbor of vertex C, and

$V(C_i,C_j)=t$, if C_i is the t^{th} neighbor of C_j;

$V(C_i,C_j)=-1$, otherwise;

Given the initial contour queue "C_0C_1," the algorithm is:

1. k=1;
2. t=V(C_k, C_{k-1});
3. j=(t+3) mod 12 ;
4. C_{k+1}= N_j(C_k);
   if (C_{k+1} in the boundary) goto 6;
   else j=(j-1) mod 12;
5. goto 4
6. add C_{k+1} to the contour queue, k=k+1;
   if (C_k=C_0) then halt;
7. goto 2

In Figure 4, we make all pixels above the dotted line hand pixels, thus we apply the above algorithm on the image and constrain the algorithm to search the whole image from left to right instead of terminating too early. The detected contour is shown in black solid line in Figure 4.

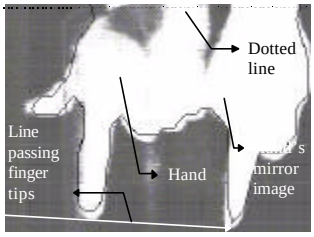

Figure 4. The Hand Contour.

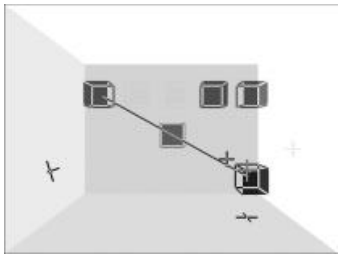

Figure 5. A Simulated 3D Room

4 Contour Analysis

To achieve real-time performance, the detected contour is not analyzed in an elaborated manner. To obtain the position of the index finger tip, we constrain the user's index finger tip to be farthest point from the dotted line. After the rough position of the two index finger tips (the real finger and its mirror image) are detected from Figure 4, we still use the similar chain coding algorithm on a denser square grid to search for more accurate finger tip positions in the neighborhood based on the rough finger tip positions. Because only a small neighborhood needs to be searched, the search is very fast.

After that, the thumb state detection is done by measure finger tip features. Using the discovered index finger tip position as reference, we "guess" a range within the contour whereby the thumb must lie. If the contour bends significantly within the selected range, we say that the thumb has been found.

5 Post-processing

After the finger tips are fixed, (3) is used to restore the 3D coordinates of the finger tip thus a stream of 3D coordinate signals is generated from the video stream. Due to the incorrect detection or noise, the signals generated are not stable thus need to be post-processed.

5.1 Screening the False Signals

From the geometry constraint provided by our setup, we find all straight lines, that pass through the real finger (R') and its mirror image (V') (the white solid line in Figure 4), converge to single point in the image plane [12]. This properties is used to filter out most false signals. In our experiment, less than 5% of the raw signals are filtered by this constraint. When a signal has been filtered, there is a "hole" in the signal stream. Such holes are filled by the predictions of Kalman filters (see below).

5.2 Kalman Filtering and Adaptive Filtering

A set of linear Kalman filters are used both to suppress the Gaussian noise in the signal and to make predictions for the holes in the signal. To ensure a more stable signal with short delay, a set of adaptive filters are also used to smooth the signal.

6 System performance

A 3D mouse application (Figure 5) is built to demonstrate the final performance of our system on a SGI Indy workstation. A simulated 3D room with several cubes inside the room is displayed on the screen. User is asked to manipulate the cubes in 3D space and arrange the cubes in some pre-defined patterns.

- *Sampling rate* -- Because only the boundary pixels are analyzed, the computation cost is low and we are able to achieve a rate of 25 frames per second (each frame is 640X480).
- *Robustness* -- We tested the system with different backgrounds, hand sizes, hand colors, and even with hands holding objects. The test results indicate that our system is robust to dramatic shape changes, fast movements, and frequent re-entry. A experiment in which a hand is doing fast (1-2 Hz) circular movement before a clustered background is shown in Figure 6. The results indicate that our system are able to follow the finger tip in real-time and the output are relatively stable and smooth. Since false recognition has little effect on the later frames recognition, it doesn't have lock loss problem which is quite normal in strong model approach (see results in [2]).
- *Resolution* -- We map the 3D coordinates into a 320x240x200 space. Because of finger thickness, the hand shaking and noise, we achieved an 80x60x100 effective resolution, that is to say the user can effectively specify 80x60x100 positions in 3D space. This result is good enough for most entertainment and education applications.
- *Lag* -- The lag in our system is mainly caused by the Kalman filters and adaptive filters which are used to stabilize the signal. This lag in our system is 0.08 - 0.16 sec. The lag is acceptable from the user's point of view.

7 Conclusions

We have proposed a novel low cost system setup to achieve accurate 3D position and address the constraint provided by the setup. Using the setup, a practical 3D mouse is designed and implemented. Successful post-processing utilizing the constraint provided by the setup is applied to stabilize the outputs. Experimental results show that our system satisfies the requirements for a robust and user friendly 3D input device.

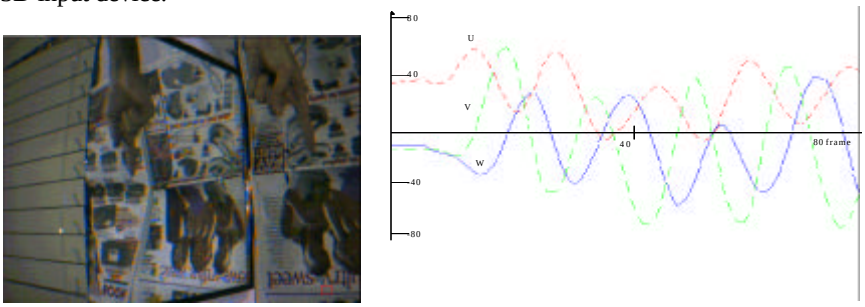

Figure 6. An Example of Clustered Background.

REFERENCES

1. F. K. H. Quek, "Eyes in the Interface," *Image and Vision Computing*, Vol. 13, No. 6, Aug. 1995. pp. 511-525.
2. A. Blake, M. Isard and D. Reynard, Learning to Track the Visual Motion of Contours, *Artificial Intelligence*, Vol 78, 1995, pp. 101-133.
3. R. Cipolla and N.J. Hollinghurst. Human--robot interface by pointing with uncalibrated stereo vision. *Image and Vision Computing*, 14(3):171--178, 1996.
4. L. Goncalves and E. D. Bernardo, Monocular tracking of the human arm in 3D, *Proc. Of 15th International Conference on Computer Vision*, 1995, pp 764-770.
5. J. Kuch and T. Huang, Virtual Gun: A Vision Based Human Computer Interface Using the Human Hand, *Proc. IAPR Workshop on Machine Vision Application*, Tokyo, 1994, pp. 196-199.
6. J. Rehg, DigitEyes: Vision-Based Human Hand Tracking for Human-Computer Interaction, *Proc. of the 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*. 1994, pp. 16-22.
7. A. Utsumi Real-time Hand Gesture Recognition System, *Proc. of ACCV'95*, 1995, pp. 249-253.
8. M. Kass, A. Witkin, and D. Terzopoulos Snakes: Active Contour Models, *Proc. Ist Int. Conf. On Computer Vision*, 1987, pp. 259-268.
9. C. R Wren., A. Azarbayejani, and A. Pentland, "Pfinder: Real-time Tracking of the Human Body," *Proc. of the 2nd International Conference on Automatic Face and Gesture Recognition*, 1996.

10. Freeman, H., "Computer Processing of Line-drawing Data," *Comput. Surveys*, v. 6, Mar. 1974, pp. 57-59.
11. D. K. Scholten and S. G. Wilson Chain Coding with Hexagonal Lattice, *IEEE Trans. PAMI*, No. 5, Sept. 1983, pp. 526-533.
12. Sifang, Li, A Vision-based 3D Mouse, MSc. Thesis, National University of Singapore, 1997.