Combining Pointing Gestures with Video Avatars for Remote Collaboration

Seon-Min Rhee[1] and Myoung-Hee Kim[1,2,*]

[1] Department of Computer Science and Engineering, Ewha Womans University, 11-1 daehyun-dong Seodaemun-gu, Seoul, 120-750, Korea
blue@ewhain.net

[2] Center for Computer Graphics and Engineering(CCGVR), Ewha Womans University, 11-1 daehyun-dong Seodaemun-gu, Seoul, 120-750, Korea
mhkim@ewha.ac.kr

Abstract. We present a simple and intuitive method of user interaction, based on pointing gestures, which can be used with video avatars in a remote collaboration. By connecting the head and fingertip of a user in 3D space we can identify the direction in which they are pointing**.** Stereo infrared cameras in front of the user, together with an overhead camera, are used to find the user's head and fingertip in a CAVE^{TM}-like system. The position of the head is taken to be the top of the user's silhouette, while the location of the user's fingertip is found directly in 3D space by searching the images from the stereo cameras for a match with its location in the overhead camera image in real time. The user can interact with the first object which collides with the pointing ray. In an experimental result, the result of the interaction is shown together with the video avatar which is visible to a remote collaborator.

Keywords: gesture interaction, immersive display, human-computer interaction.

1 Introduction

A spatially immersive display can be used to promote efficient remote collaboration by making a remote collaborator apparently visible in a local environment, and permitting them to manipulate shared virtual objects. In previous work [1] we proposed a stereo image based video avatar of a collaborator which can be merged into a virtual world. This allows users in remote VR environments to see each other. In addition to this feature, to create an efficient collaborative virtual workspace a simple and intuitive way of supporting interactions between the collaborator's and virtual objects needs to be provided. And the results of interactions need to be made visible to both collaborators. We achieve this with a tracking interface that can interpret pointing gestures.

To support vision-based user interaction in a virtual world, two categories have been studied. The first approach uses multiple images for pose estimation, in which an

[*] Corresponding author.

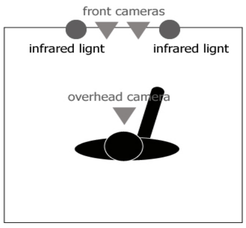
articulated body model is fitted to 3D data [2, 3, 4]. However, this approach does not permit real-time tracking of articulated structures. Second approach is gestural interaction using feature points. Kahn and Swain [5] were responsible for an early method of detecting pointing gestures, and Nickel and Stiefelhagen [6] have presented a method of pointing detection for human-machine interactions, using hidden Markov models. Kolesnik and Kulessa [7] used an overhead camera and segmented the silhouettes of users into two parts, corresponding to the body and an arm. But these three techniques have only been applied to interaction in a local environment.

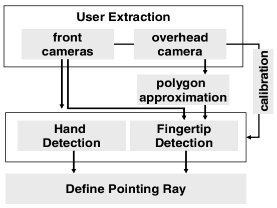
We have implemented our method within a CAVE^{TM}-like system to support remote collaboration. Interpreting a user's gesture in such an environment is made more complicated because the background to the user is a changing image on the VR display, and not a static background. We therefore rely on infrared imaging to detect the user's movements robustly. We employ the concept of a pointing ray, which connects the head and the fingertip of a user in 3D. Images from two frontal and one overhead camera are used to detect the end-points of this ray. This gestural interaction is implemented in the context of our video avatars [1], so that a user can see their collaborator and their actions with respect to a virtual object.

This paper is organized as follows. In Section 2 we provide an overview of our approach to gestural interaction. In Section 3 we explain how we detect the position of the user's head and fingertip, which define a pointing gesture. We describe our implementation and present the results in Section 4, and conclusions follow in Section 5.

2 Overview

In a spatially immersive display environment such as a CAVE^{TM}-like system, uniform illumination can not be guaranteed because a changing virtual world is always being displayed. In our previous work [1] we used infrared images, which ignore the visible light from the screen, so as to achieve the static background which is essential for robust feature extraction, and we use the same environment in this current work. Fig. 1 shows our hardware setup and software organization. Two cameras and infrared light sources are installed on the upper front part of the screen and an overhead camera is attached to the ceiling.





(a) (b)

Fig. 1. (a) Our hardware setup in a CAVE^{TM}-like system and (b) the software components that support gestural interaction

Gestures are analyzed using a pointing ray. While making simple pointing gestures, humans tend to look at the target object. The pointing ray is this line of sight, and it can be constructed by connecting the head and fingertip of a user. The use of an overhead camera makes it easier to find the fingertip, which is usually the body part that is farthest from the center of the body in the overhead camera image. We can construct the equation of a pointing ray *P* in terms of the 3D position of the head *H*, and the fingertip position *F*, as follows:

$$P = H + \alpha \frac{H - F}{|H - F|} \tag{1}$$

where α determines the length of the ray.

3 Analyzing Pointing Gestures

Prior to attempting detection of the user's head and fingertip the cameras must be calibrated. We use Tsai's method [8] to obtain internal and external camera parameters.

3.1 Head Detection

We use an efficient heuristic for head detection. If the silhouette of a user can be extracted robustly from an image, then the position of their head can simply be taken as the highest point on the silhouette. We can eliminate small irrelevant areas which are not related to the user's silhouette by region growing, as shown in Fig. 2(b). The seed point for region growing is the centroid of the region, which is defined as follows:

$$C = (\bar{x}, \bar{y}), \quad \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \quad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{2}$$

where n is the number of points inside the silhouette, and their coordinates are (x_i, y_i).

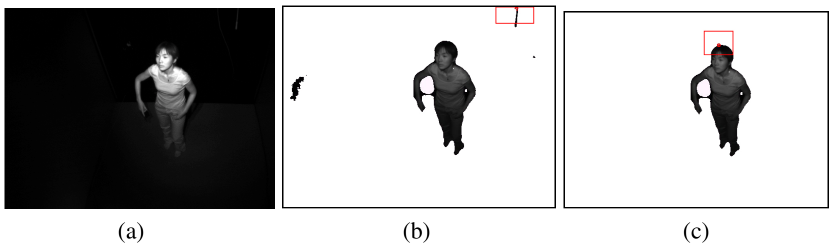


Fig. 2. Head detection: (a) an original infrared image, (b) the initial attempt at silhouette extraction and (c) the result of improved extraction result by region growing. In (b) and (c) the detected head position is located in the red box.

Using the two head points in the images from the frontal cameras, the 3D position of the head can be calculated by triangulation [9].

3.2 Fingertip Detection

We locate a fingertip in 3D in two steps. First, the fingertip is found in the overhead camera image. We then search for its 3D position with the help of the two frontal camera images.

The process of fingertip detection using the overhead camera is summarized in Fig. 3. We assume that the fingertip will be the farthest point from the centroid of the silhouette of the user, as shown in Fig. 4(a). The centroid can be calculated using Equation (2), which was given in the previous section.

To find the farthest point from the centroid, we could check every point on the boundary of silhouette. However, to reduce the search time, we approximate the boundary by a polygon.

A distance function D_j is defined as $Dj=\sqrt{(x_j-\bar{x})^2+(y_j-\bar{y})^2}$, $Vj=(x_j,y_j)$, where V_j is a vertex of the approximated boundary. Fig. 4(b) shows the value of this function for each vertex of an example polygon. If there is more than one most distant point, the brightest in the overhead camera image is chosen to represent the fingertip.

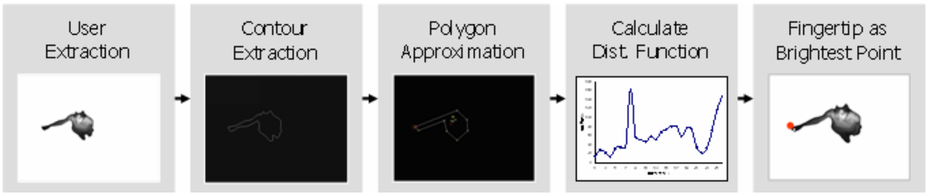
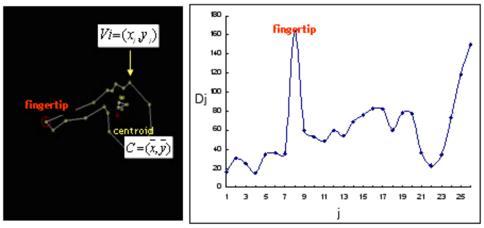


Fig. 3. Fingertip detection from the overhead camera image

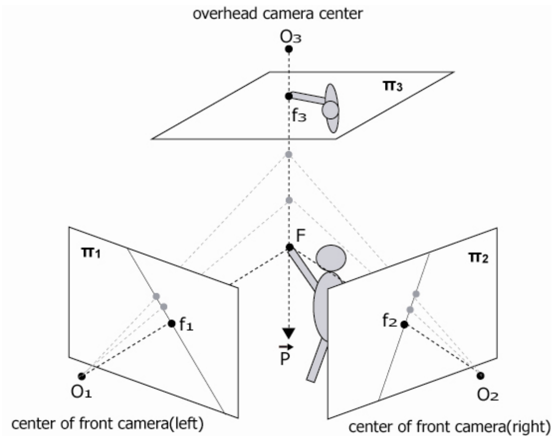


(a) (b)

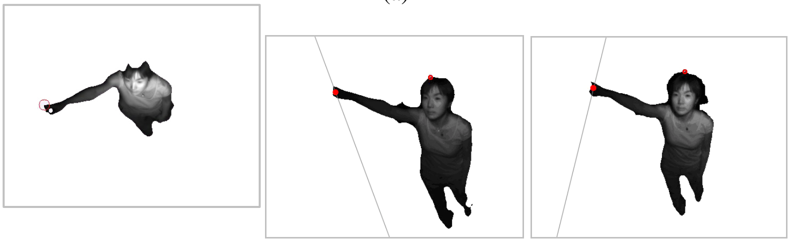
Fig. 4. (a) A polygon approximating the user's silhouette and (b) a plot of the distance function from the centroid of the silhouette to each vertex of the polygon

In the second step, we search for the 3D position of the fingertip with the help of the two frontal camera images (Π_1 and Π_2), as shown in Fig. 5(a). Because the infrared images only have a single channel of color information we cannot use a traditional stereo matching algorithm to find the 3D position of the point. We have therefore formulated a method of finding its 3D position directly, without using stereo matching.

The fingertip f3 on the image plane of the overhead camera image Π_3 can be reprojected into a 3D space along with the vector $\vec{P}$. The actual fingertip will be a point on this vector. Each point on $\vec{P}$ can be projected on to the image planes Π_1 and Π_2. If this point is the actual fingertip then the two projected points are also fingertips in their respective images and therefore both points will be within the silhouette of the user. Fig. 5(b) shows the result of fingertip detection in the overhead camera image and Fig. 5(c) and 5(d) shows projected fingertip candidates, which form an epipolar line.



(a)



(b) (c) (d)

Fig. 5. (a) Conceptual view of fingertip detection: (b) shows the fingertip found in the overhead camera image, and (c) and (d) show the epipolar line of fingertip candidates. The final fingertip, which is within the user's silhouette in both images, is circled in red.

4 Implementation and Results

We used separate computers for detecting the pointing ray and for rendering the virtual world. To detect the head and fingertip we used an Intel Pentium 4 Dual CPU3.0GHz with 2GB RAM. The Flycapture [11] and OpenCV [12] libraries were used for image acquisition and processing. As a rendering server we used an Intel Pentium Xeon 2.4GHz with 2GB RAM, supporting the Blue-c API [10].

The 3D positions of the head and fingertip are transmitted to the rendering server. At this juncture their positions are expressed in a world coordinate system which is defined during camera calibration. Both head and fingertip points must be transformed into scene coordinates to support interaction with virtual objects in the scene.

The user is able to interact with the first object which collides with the pointing ray. An example of this form of interaction is shown in Fig. 6, in which the user selects a virtual object by pointing, and that object is then shown as a highlighted wireframe.

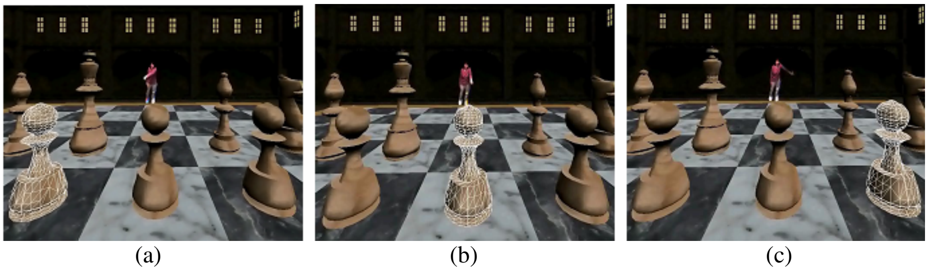


Fig. 6. Object selection using a pointing gesture. A user in the virtual environment points at the chess pieces in the order (a), (b) and (c). In each case the selected object is rendered as a highlighted wireframe.

5 Conclusion

We have proposed a simple but efficient method of interaction based on pointing gestures, which can be used with video avatars for collaboration in a spatially immersive virtual environment. We detect gestures from the 3D positions of the head and fingertip of the users, using two cameras installed in front of the screen and a third overhead camera. A pointing gesture can be used to select a virtual object to be manipulated. This results in a more powerful collaborative VR workspace in which our video avatars can operate. Users can see each other and collaborate within the same virtual world.

Acknowledgments. This work was supported by the Korean Ministry of Information and Communication (MIC) under the Information Technology Research Center (ITRC) Program. We would like to thank to Remo Ziegler (ETH Zurich, Switzerland) for the fruitful comments for this work.

References

1. Rhee, S.-M., Ziegler, R., Park, J., Naef, M., Gross, M., Kim, M.-H.: Low-Cost Telepresence in Collaborative Virtual Environment. IEEE Transaction on Visualization and Computer Graphics 13(1), 156–166 (2007)
2. Cheung, G.K.M., Baker, S., Kanade, T.: Shape-From-Silhouette of Articulated Objects and Its Use for Human Body Kinematics Estimation and Motion Capture. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 77–84 (2003)

3. Mikic, I., Trivedi, M., Hunter, E., Cosman, P.: Human Body Model Acquisition and Tracking Using Voxel Data, International Journal of Computer Vision, vol. 53(3), 199–223 (2003)
4. Tollmar, K., Demirdijan, D., Darrell, T.: Gesture and Play - Exploring Full-Body Navigation for Virtual Environments. In: Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop, vol. 5(47) (2003)
5. Kahn, R.E., Swain, M.J.: Understanding People Pointing: The Perseus System, International Symposium on Computer Vision. A Motion III, 11 (1995)
6. Nickel, K., Stiefelhagen, R.: Real-time Recognition of 3D-Pointing Gestures for Human-Machine-Interaction. In: Michaelis, B., Krell, G. (eds.) Pattern Recognition. LNCS, vol. 2781, pp. 557–565. Springer, Heidelberg (2003)
7. Kolesnik, M., Kulessa, T.: Detecting, Tracking and Interpretation of a Pointing Gesture by an Overhead View Camera. In: Proceedings of the Annual Symposium of the German Association for Pattern Recognition, pp. 429–436 (2001)
8. Tsai, R.Y.: An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 364–374 (1986)
9. Taubin, G.: Camera Model and Triangulation, Note for EE-148: 3D Photography, CalTech (2001)
10. Naef, M., Staadt, O., Gross, M.: Blue-C API: A Multimedia and 3D Video Enhanced Toolkit for Collaborative VR and Telepresence. In: Proceedings of ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry, pp. 11–18 (2004)
11. FlyCapture® Software Development Kit, http://sourceforge.net/projects/opencvlibrary/
12. Open Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/