Arm-Pointer: 3D Pointing Interface for Real-World Interaction

Eiichi Hosoya, Hidenori Sato, Miki Kitabata
Ikuo Harada, Hisao Nojima, and Akira Onozawa

NTT Microsystem Integration Laboratories
3-1, Morinosato Wakamiya, Atsugi-shi, Kanagawa, 243-0198 Japan
{hosoya,hide,kitabata,harada,nojima,onoz}@aecl.ntt.co.jp

Abstract. We propose a new real-world pointing interface, called Arm-Pointer, for user interaction with real objects. Pointing at objects for which a computer is to perform some operation is a fundamental, yet important, process in human-computer interaction (HCI). Arm-Pointer enables a user to point the computer to a real object directly by extending his arm towards the object. In conventional pointing methods, HCI studies have concentrated on pointing at virtual objects existing in computers. However, there are the vast number of real objects that requires user operation. Arm-Pointer enables users to point at objects in the real world to inform a computer to operate them without the user having to wear any special devices or making direct contacts with the objects. In order to identify the object the user specifies, the position and direction of the arm pointing are recognized by extracting the user's shoulders and arms. Therefore, an easy-to-use real-world oriented interaction system is realized using the proposed method. We developed an algorithm which uses weighted voting for robust recognition. A prototype system using a stereo vision camera was developed and the real-time recognition was confirmed by experiment.

1 Introduction

This paper presents a new method for a real-world pointing system, called Arm-Pointer. Among computer-human interaction techniques, identifying a target object by pointing to the object is one of the most fundamental, yet critical, processes. A typical example of such a pointing process is when we use a computer mouse to point at icons and menus on a computer display.

In a real world environment, there are many objects, such as electronic devices, that are operated by human users. Each real-world object has its own user interface, such as a mouse, keyboard, or remote controller; however, because of the number of such objects, those operations become more complicated. To make the usage of these real-world objects easier, a more intuitively learnable and unified interface is required. From the viewpoint of usability, nothing should be attached to the user's body, the pointing should be detected in real time, and the recognition accuracy should be reliable enough.

Some research have been devoted to achieving object pointing at some distance from the computer display using sensors [1-4]. Put-That-There [1] uses magnetic

N. Sebe et al. (Eds.): HCI/ECCV 2004, LNCS 3058, pp. 72-82, 2004.

sensors and can extract the position of the cursor on the screen. Ubi-Finger [2] is a gesture-input device that uses bending, touch, IR, and acceleration sensors to detect finger gestures for interaction with real objects. These methods can point to a real or virtual object with high accuracy and high speed, but they require the attachment of some equipment to the user [2, 3] or the construction of an environment with sensors [1, 4], which constraints user's position and motion during use.

Other pointing methods based on image processing have been proposed [5-13]. HyperMirror [9] and the Mirror Metaphor Interaction System [10] detect pointing position on a two-dimensional (2D) screen overlapped by self-image. Finger-Pointer [11] and GWINDOWS [12] uses two cameras and extract the three-dimensional (3D) position of the finger to control a cursor on the 2D screen. With these methods, the user does not have to wear any equipment, so they reduce operation-related stress, but they need a display to show the objects to be pointed.

Our proposed method, Arm-Pointer, is a method based on image processing. It can perform 3D recognition of arm pointing direction, which is advantageous for usability and applicability. With Arm-Pointer, the user does not have to wear any equipment and can point to a target real object directly just by stretching out his arm without using a display. Arm-Pointer restricts user position and motion much less than sensor-based methods, and also restricts the shape of user's body much less than other image processing methods that need difficult and robust recognition of the position of small parts or shapes like the finger. Therefore, Arm-Pointer has various applications, such as the control of electric appliances. This paper describes the prototype and discusses its operation and performance based on experimental results.

2 Arm-Pointer

Our goal is to build a real-world pointing system that a user can operate without having to wear equipment and that allows the user to point at object directly in the real world. It is also important for the system to be robust to background images and the user's dress. In what follows, the Arm-Pointer algorithm is explained.

2.1 System Configuration

The configuration of Arm-Pointer is shown in Fig. 1. The system consists of a stereo vision camera, an infrared (IR) remote controller, and a PC. A user stands in front of the stereo vision camera facing it. The user points to a target object by extending his arm, and the system detects that object using the position pointed to. And if the object has a function for interactive operation, the function will be activated. For example, when the user just points at the TV, the TV will turn on.

2.2 Processing Flow

Figure 2 shows the processing/data flow of the whole system. A depth image and a color input image are generated from an output image of a stereo vision camera. A mirror image is also generated and displayed, but this is an additional function for the

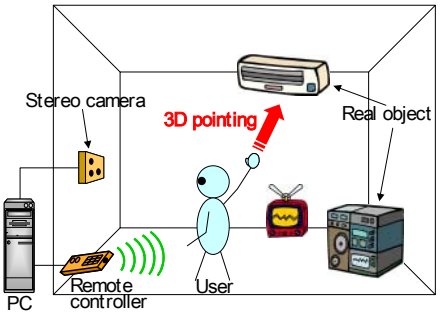


Fig. 1. Arm Pointer: Real-world pointing interface

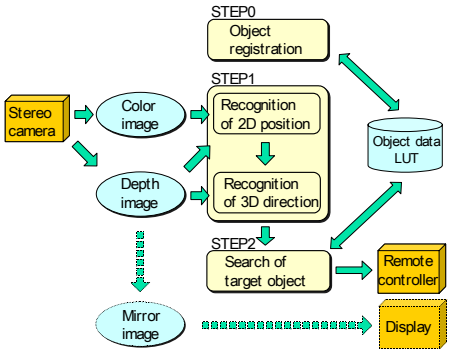


Fig. 2. Processing/data flow

evaluation of user feedback. The system can perform all operations without the display. In the case of interaction with an object, there are roughly two recognition steps. In STEP 1, 3D coordinates of shoulders and arms are recognized and the 3D coordinates of the pointed position are calculated. In STEP 2, the target object is found using the detected position information. All objects are registered beforehand for initialization, which is in STEP 0. STEP 0 to STEP 2 are outlined as follows:

STEP 0 Initialization
: Subdivide voxels and register target objects.

Repeat following steps for each video frame.

STEP 1 3D recognition
: Extract skin color and recognize 3D position of arms and shoulders.

STEP 2 Object search
: Calculate 3D pointing line, vote, and activate command.

3 Initialization

Information about all target objects in the room has to be registered beforehand. This information is used for object search and detection from the pointing direction. In this section, voxel space is defined and the registration of objects described.

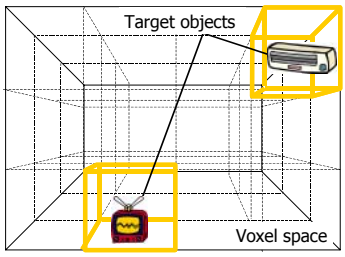


Fig. 3. Object registration in the voxel space

3.1 Voxel Subdivision

Real space is divided into voxel space (discrete space) as shown in Fig. 3. The resolution of voxel space changes with application or the number of objects. Our system can be used in rooms of various sizes by changing the size of voxel space.

3.2 Target Object Registration

We create a look-up table (LUT) corresponding to all voxels. Every record of the LUT contains information about a voxel, such as 3D coordinates, name, and functions. Using this LUT, the system can retrieve the information about a target object, such as televisions, air-conditioners, or walls, by using the 3D coordinates as a search key.

4 3D Recognition and Object Search

The 3D coordinates of the position of the user's shoulders and arms, that is, the 3D pointing direction, are computed by using a depth image and color input image obtained from the stereo vision camera. The object pointed to is detected by determining the pointing direction in the 3D voxel space. First, in this section, the image processing for 3D recognition of 3D pointing direction and the simple object search are described for easy explanation. Next, the improved method is described.

4.1 3D Recognition Flow

The 3D pointing direction is obtained by extracting the 3D position of shoulders and arms, and the 3D position of shoulders are calculated from the 3D position of the face. Fig. 4 shows the coordinate corresponding to each position to be extracted. The steps in the 3D recognition process are outlined as follows:

STEP 1 3D recognition
  STEP 1-1 Extract skin color from input image
    $h_1<h<h_2$, $s_1<s<s_2$, $v_1<v<v_2$
  STEP 1-2 Recognize face and arms
    Reduce image noise by 2D image processing
    Exclude background noise by depth image
    Exclude error extraction by 2D position restriction of face and arms
    Calculate 2D coordinates of center of gravity of face and arms;
      (x_f, y_f), (x_{ar}, y_{ar}), (x_{al}, y_{al})
  STEP1-3 Extract shoulders
    Calculate 2D coordinates of shoulders from face position;(x_{sr}, y_{sr}), (x_{sl}, y_{sl})
  STEP1-4 Extract distance
    Calculate 3D coordinates of shoulders and arms from depth image;
      (x_{sr}, y_{sr}, z_{sr}), (x_{sl}, y_{sl}, z_{sl}), (x_{ar}, y_{ar}, z_{ar}), (x_{al}, y_{al}, z_{al})

In STEP 1-1, the skin color region in the color input image is extracted. The range on the HSV color space of the input image is specified for skin color extraction. The calibration for skin color restrictions is performed beforehand. Each range ($h_1<h<h_2$, $s_1<s<s_2$, $v_1<v<v_2$) is decided by the calibration for the current light condition.

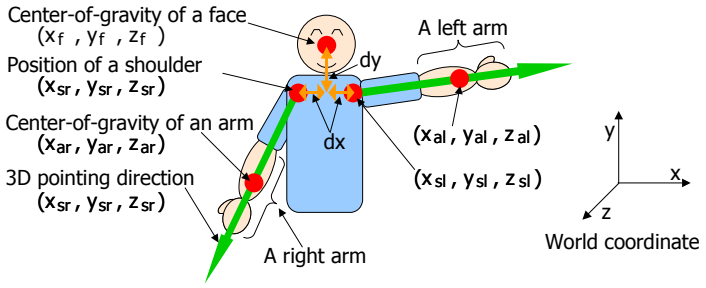


Fig. 4. Calculation for 3D recognition of arms

In STEP 1-2, face and arms recognition is performed. A pre-process reduces noise by avoiding holes and too small areas. The face and arm regions are detected within the given permissible range of user's position, as far as the user faces the camera. With these restrictions, incorrect recognition because of background colors is reduced. Here, (x_f, y_f) are the center-of-gravity coordinates of the face, (x_{ar}, y_{ar}) are the center-of-gravity coordinates of the right arm, and (x_{al}, y_{al}) those of the left arm.

In STEP 1-3, the positions of both shoulders are calculated from (x_f, y_f). Here, for ease of computation, we assume that the user is looking to the front (in the direction of the camera), and that the shoulder is fixed at a certain distance and direction. The coordinates of shoulders, (x_{sr}, y_{sr}) and (x_{sl}, y_{sl}), are determined as those separated certain amount of length from (x_f, y_f).

$$\left.\begin{array}{l} x_{sr} = x_f - dx, \quad y_{sr} = y_f + dy, \quad x_{sl} = x_f + dx, \quad y_{sl} = y_f + dy, \\ dx, dy: \text{difference lengths of between face and shoulders.} \end{array}\right\} \quad (1)$$

Thus, 2D coordinates of the shoulders and arms in a pointing direction can be acquired from a color input image.

In STEP 1-4, the z coordinates of shoulders, z_{sr}, z_{sl}, and arms, z_{ar}, z_{al}, are obtained as depths at their x and y coordinates in the depth image.

The user's 3D pointing direction is indicated by a 3D pointing line that extends from the shoulder along the arm. With this method, the arm has to be fully extended for pointing, but shape recognition of a finger or hand is not necessary. Therefore, there are no constraints imposed by the user's clothes or finger shape.

4.2 Object Search

The target object is searched for in the pointing direction obtained by 3D recognition of the shoulders and arms. The steps in object search are outlined as follows.

STEP 2 Object search
  STEP 2-1 Calculate 3D pointing line
  STEP 2-2 Vote
    Calculate crossing voxel of the line
    If an object is registered in a voxel, increase voting value for the voxel
    If a voting value is over the threshold, go to STEP 2-3, else go to STEP 2-4
  STEP 2-3 Activate command

If the registered object has an interactive function, a command is activated.

STEP 2-4 Decrease voting value after the certain number of frames

In STEP 2-1, the equation of the 3D pointing line is as follows:

$$\frac{x - x_{sr}}{x_{ar} - x_{sr}} = \frac{y - y_{sr}}{y_{ar} - y_{sr}} = \frac{z - z_{sr}}{z_{ar} - z_{sr}} = t_r, \quad (2)$$

$$\frac{x - x_{sl}}{x_{al} - x_{sl}} = \frac{y - y_{sl}}{y_{al} - y_{sl}} = \frac{z - z_{sl}}{z_{al} - z_{sl}} = t_l. \quad (3)$$

In STEP 2-2, voting is performed for detection of the voxel with the target object. First, each voxel crossing the pointing line is detected from the nearest to the farthest in order. If the registered object exists in the voxel, voting value is incremented. If the voting value reaches the threshold after several repetitions for frames, the system goes to STEP 2-3. STEP 2 is repeated until the object is found or room wall is reached. In STEP 2-3, if the detected object is registered as an object with an interactive function, the command of the function is activated. For example, a TV can be turned on or off. In STEP 2-4, the voting value is decreased after a certain time (number of frames).

4.3 Improvements

The method explained in the previous section, referred to as the simple method here, often can not detect a target object quickly. In this section, we propose some improvements using a weighted voting algorithm for better extraction performance. In the improved method, the system votes not only for the crossing voxel through which the pointing line passes, but also voxels neighboring the crossing voxel with a weighted voting value. The area for voting comprises 26 neighboring voxels as shown in Fig. 5. Voting values are decreasing with the distance from the arm to crossing voxel. Using this improved method, it is possible to detect the target object faster, even if the detected position of face and arms are unstable. In this work, we tested Arm-Pointer using only one set of voting weights, which we decided voluntarily. In future work, we will test and evaluate the system using other sets of voting weights to further improve the performance of the voting method. The processing flow in the new voting step (STEP2-2') is as follows. This step replaces STEP2-2 in the simple method.

STEP2-2' Vote

Calculate the crossing voxel V_c (X_c, Y_c, Z_c) that includes an object.
Vote V_c and its 26 neighboring voxels according to the weight,
if they include any object.

5 Application Prototype

We developed a prototype system using our proposed method. The system consists of a PC (Pentium IV, 2 GHz), stereo vision camera (Digiclops), infrared multi-remote controller (CROSSAM2+USB), and image processing software. In this system,

a display can be used additionally for confirming operations by overlaying CG information, although our method can be implemented without a display.

The user has to face the camera and extend his arm to the target. If the object included in the voxel is interactive, its function will be activated. This system makes it possible to operate many apparatuses in the room with only one remote controller.

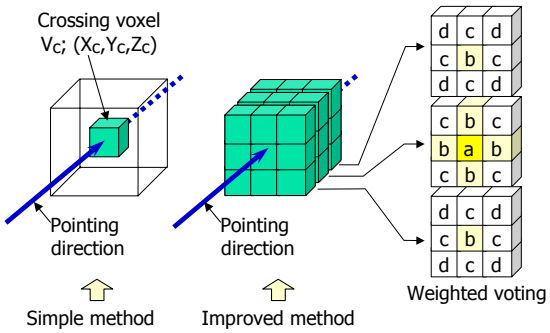
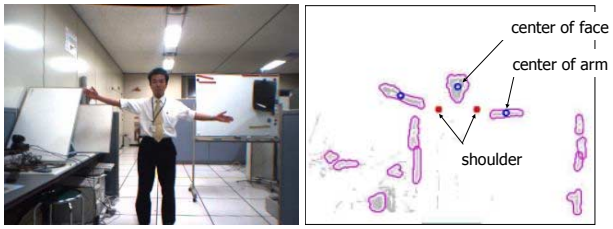


Fig. 5. Weighted voting

(a) Input image (b) Depth image (c) Position extraction

Fig. 6. Examples of 2D extraction



(a) Input image (b) Position extraction

Fig. 7. In the case of short-sleeved shirt

6 Evaluation

We tested the system for evaluation. In this section, examples of user operation and results of a comparison evaluation are shown.

6.1 Experiment

An experiment using the prototype system confirmed basic operations. The processing speed of this system reached about 150 msec/frame, including indication on the display.

An example of the extraction results for the shoulders and centers of the arms are shown in Fig. 6. Figure 6(a) and (b) are an input color image and a depth image, respectively. Figure 6(c) shows the result of skin color extraction, and the 2D position of the center of a face, shoulders, and arms. Although many candidates are extracted at first, finally only the face and arms are detected. Figure 7 shows an example of the extraction results when the user is wearing a short-sleeved shirt. In the case of either a long-sleeved shirt (Fig. 6) or short-sleeved shirt, the 3D direction of the arm can be extracted similarly, without changing any system parameters.

Figure 8 shows an example of the results of the 3D pointing experiment. The red line segment that shows the pointing direction and the voxel frame of the 3D pointing position is displayed and changes. Figure 9 shows an example where interaction with an object was performed. In the case of this figure, since a TV is selected by pointing, the frame of the voxel with the TV is emphasized on the screen. The TV is thus turned on and the message indicated on the screen.

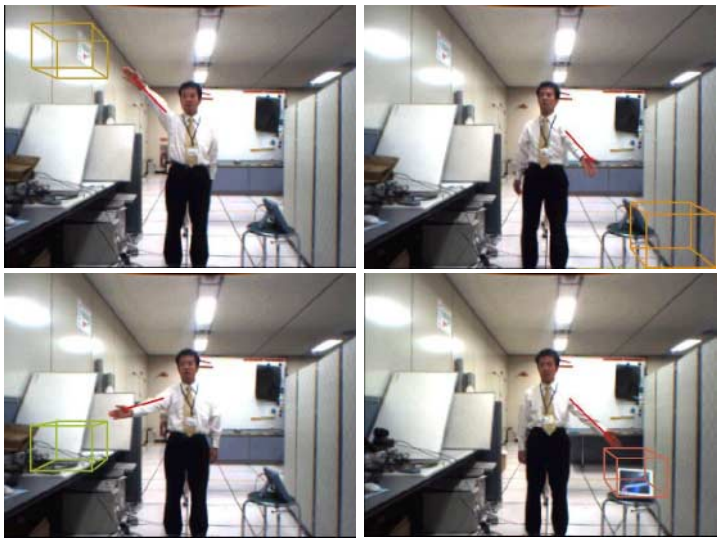


Fig. 8. Examples of 3D pointing

6.2 Comparison Evaluation

For the evaluation, the simple method Arm-Pointer, improved method Arm-Pointer, and an ideal case were compared. The working time for detecting an object pointed at is one of the most important indices for user interface evaluation. In the experiment, users pointed at objects indicated randomly and the detecting time was measured. We prepared a laser pointer as the ideal device for pointing, because user can point at precise positions in real time perfectly with a laser pointer, though he can't interact with objects. So we used the measured working time of the laser pointer as the ideal working time value. Working time was evaluated for three types of method: the laser pointer, the simple method, and the improved method.

The number of showing objects for the user was 18 (6 objects x 3 times). Each 3D position of the 6 numbered objects was random on a wall. The sequence of showing objects was random. Four subjects were participated. The object sequence was the same for each. Voxel space resolution was 5 x 5 x 5 for size 2.5 x 2.5 x 2.5 m^3. Voting weights in the improved method are (a, b, c, d) = (10, 5, 2, 1). The object number was indicated automatically on the display in front of the user. The working time from object number indication until pointing finished was measured.

Table 1 shows the results of experiment. Averages of working time were 1.44 [s] for the laser pointer, 3.66 [s] for the simple method, 2.74 [s] for the improved method, respectively. The improved method is faster than the simple method, and the speed is not so slow compared with the laser pointer as an ideal device for pointing. A laser pointer can point at precise positions, but no interactions are available with it and the user needs to keep holding. However, our method has advantages in that it allows interaction with real objects and the user does not have to hold anything. In this experiment, laser pointer has a feedback effect because of projected laser beam. We plan to perform an experiment with feedback for our simple and improved methods and evaluate the methods under more similar conditions in future.

Table 1. Experiment results

Method	Ave.[s]	SD
Laser pointer	1.44	0.33
Simple method	3.66	2.06
Improved method	2.74	1.34

SD: Standard Deviation

7 Conclusion

We proposed a 3D pointing interface called Arm-Pointer that can perform 3D recognition of arm pointing direction. The user does not have to wear any equipment and can point to a target object directly by extending his arm. Since Arm-Pointer can point to real objects in a real space, it realizes an intuitive interface. In the case of either a long-sleeved or short-sleeved shirt, Arm-Pointer can extract the arm pointing direction correctly. Thus, Arm-Pointer will have various applications in the real world. Future work includes considering more efficient voting algorithms especially for the case of a large number of objects, and further improvements on the accuracy of the system.

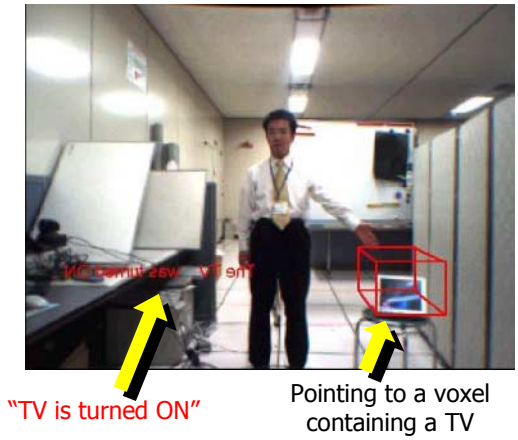


Fig. 9. Interaction with a real object

Acknowledgement

The authors thank Dr. H. Ichino for supporting this research. They also thank the members of the Home Communication Research Group at NTT for helpful discussions.

References

[1] Bolt, R.A. "Put-That-There": Voice and Gesture at the Graphics Interface. Proc. SIGGRAPH 1980, Vol14, No.3, (1980), pp. 262-270.
[2] Tsukada, K. and Yasumura, M. Ubi-Finger: Gesture Input Device for Mobile Use. Proc. APCHI 2002, Vol. 1, (2002), pp. 388-400.
[3] Sugimoto, A., Nakayama, A. and Matsuyama, T. Detecting a Gazing Region by Visual Direction and Stereo Cameras. Proc. ICPR 2002, (2002), pp. 278-282.
[4] Rekimoto, J. SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces. Proc. CHI 2002, (2002), pp. 113-120.
[5] Koike, H., Sato, Y., Kobayashi, Y., Tobita, H. and Kobayashi, M. Interactive Textbook and Interactive Venn Diagram: Natural and Intuitive Interface on Augmented Desk System. Proc. CHI 2000, (2000), pp. 121-128.
[6] Freeman, W.T. and Weissman, C.D. Television control by hand gestures. Proc. AFGR 1995, (1995), pp. 179-183.
[7] Krueger, M.W., Gionfriddo, T. and Hinrichsen, K. VIDEOPLACE An Artificial Reality. Proc. CHI'85, (1985), pp. 35-40.
[8] Shibuya, Y. and Tamura, H. Interface Using Video Captured Images. Human-Computer Interaction: Ergonomics and User Interfaces, Vol.1, (1999), pp. 247-250.
[9] Morikawa, O., Yamashita, J., Fukui, Y. and Sato, S. Soft initiation in HyperMirror-III. Human-Computer Interaction INTERACT'01, (2001), pp. 415-422.

[10] Hosoya, E., Kitabata, M., Sato, H., Harada, I., Nojima, H., Morisawa, F., Mutoh, S. and Onozawa, A. A Mirror Metaphor Interaction System: Touching Remote Real Objects in an Augmented Reality Environment. Proc. ISMAR'03, (2003), pp. 350-351.
[11] Fukumoto, M., Suenaga, Y. and Mase, K. "FINGER-POINTER": Pointing Interface by Image Processing. Computer & Graphics, Vol. 18, (1994), pp. 633-642.
[12] Wilson, A. and Oliver, N. GWINDOWS: Towards Robust Perception-Based UI. Proc. CVPR 2003, (2003).
[13] Jojic, N., Brumitt, B., Meyers, B., Harris, S. and Huang, T. Detection and Estimation of Pointing Gestures in Dense Disparity Maps. Proc. AFGR 2000, (2000), pp. 468-475.