# Immersive 3D Human-Computer Interaction System

Po-Hsien Wang, Ting-Ying Wang, Ya-Chu Chang, and Ching-Chun Huang
National Chung Cheng University, Min-Hsiung. Chia-Yi, Taiwan

***Abstract*— In this paper, we proposed a system that allows users interact with the virtual world. We have applied the stereoscopic imaging technology to create 3D virtual reality so that users are able to visualize 3D virtual objects. On the other hand, based on a somatosensory camera (Kinect), we build up a 3D human-computer interface to allow users interact with the virtual objects by fingertip touching. Our experiments also show users can interact with 3D virtual objects seamlessly through our system.**

## I. INTRODUCTION

Nowadays, virtual reality (VR) and augmented reality (AR) are more and more prevalent. How to connect the real world with the virtual environment is the hot topic recently. For instance, if an online teaching system can provide 3D-based teaching materials, it will bring students a more tangible learning experience. Thus, we hope to design a user-friendly virtual interactive system that users can see virtual objects and interact with the objects through actions such as touching [1].

To achieve that, we use image synthesis functions provided by NVIDIA 3D Vision and Unity for our system. To create a realistic user experience, the virtual object is expected to stay at the same position, even if the user changes his viewing angle and position. However, the off-the-shelf functions from Unity do not consider user positions while synthesizing the virtual objects. Hence, from a user's perception, a visualized static 3D object is moving while the viewing position changes. In order to deal with this problem, we design an algorithm named as coordinate calibration in this work.

When a user interacts with virtual objects, fingertips are the most frequently utilized body parts. Therefore, in our system, we treat fingertips as the major interactive medium. To extract fingertips, the fingertip joints, HANDTIP, provided by the Kinect system can be used. However, we found the Kinect-extracted fingertip might drift slightly; also, the instability of fingertip detection influence the performance of our system. To provide a better user experience, more accurate fingertip detection is necessary. As a result, we refer to a fingertip detection algorithm [2] for improvement.

## II. PROPOSED METHOD

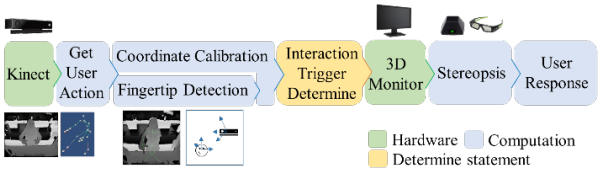



Fig. 1. Flowchart of the whole system

In order to enhance user experience, reality and stability are two keys in a 3D interaction system. To meet the requirements, we introduce coordinate calibration and robust fingertip detection. Fig. 1 illustrates the flowchart of our whole system.

### A. Coordinate calibration

In order to build a consistent visual perception of an object in a virtual scene and make the perception invariant to a user's location, we proposed to adjust virtual world in 3D Unity by dynamic scaling and geometric correction (Fig.2). The process also helps to connect the human-visual coordinate and virtual-world coordinate.

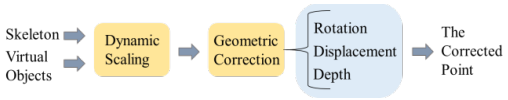



Fig. 2. Flowchart of coordinate calibration

#### 1) Dynamic scaling

To match the depth perception between real and virtual worlds, the size of a virtual object is modified dynamically according to the distance of a user to the screen. First, we detect the head position of a user and calculate the distance $D_H$ between the user to the monitor by the helps from Kinect. Then, we changed the scale of virtual–world coordinate by Equation (1)

$$Scale = Scale_{base}/D_H \ . \qquad (1)$$

Here, $\text{Scale}_{\text{base}}$ is a reference scale and is determined when the head distance $D_H$ is equal to 1 meter.

#### 2) Geometric calibration

As shown in Fig.3, when a user faces to a 3D monitor, user can visualize a virtual 3D object through a stereoscopic image pair. Note that the left and right images pair is synthesized by projecting the 3D Unity objects onto the monitor screen plane. However, when viewing the screen from sides, we still see the front view of the virtual 3D object instead of the side view. The 3D object position is also shift from the correct position.

To address the side view problem, we estimate the viewing angle θ and let all Unity objects in the virtual world rotate the same angle. On the other hand, to keep the visualized 3D object at the correct position, we translate each Unity object in the virtual world according to a user's viewing location. Later on, we project the Unity scene onto the screen plane and generate stereoscopic images so that the 3D visual perception is invariant to a user's location and viewing angle.

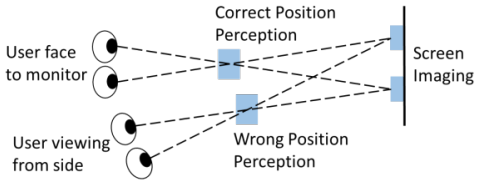



Fig. 3. Position perception

### B. Fingertip detection

If simply using the Kinect skeleton for fingertip detection, we find the extracted fingertip position would drift frequently. It affects the accuracy and stability of our interaction system.

In this work, another goal is to precisely locate the fingertip and improve the accuracy and stability while human and computer interact. To achieve that, we realized and modified

the algorithm in [2]-[3] based on the combination of the depth image and the Kinect skeleton. The detection flow as shown in Fig. 4 is composed of (a) hand ROI (region of interest) finding, (b) ROI binarization, and (c) fingertip extraction.

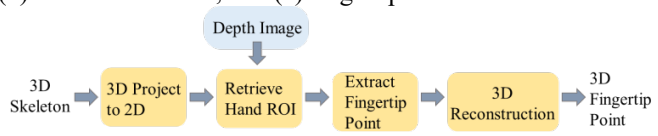


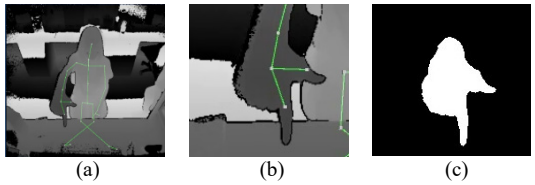

Fig. 4. Flowchart of fingertip detection

*1) Hand region extraction*

We project the 3D Kinect skeleton to its 2D depth image as shown in Fig. 5(a), and use the depth of skeleton values to extract the region of interest (ROI) within a depth image. The center of the ROI is set to be the Kinect hand joint $(H_x,H_y)$. The rectangle size of the ROI is denoted as $(S_x,S_y)$. During our calibration step, a user places his hand 1 meter away from the Kinect sensor and the optimal ROI reference size $(R_x,R_y)$ is experimentally determined. In the testing phase, the ROI size $(S_x,S_y)$ can then be dynamically estimated by

$$(S_x,S_y)= (R_x,R_y)/D. \quad (2)$$

In (2), $D$ is the depth of the Kinect hand joint. An example of the extracted hand ROI is shown in Fig. 5(b).



(a) (b) (c)

Fig. 5. An example of fingertip detection. (a) Depth image with skeleton. (b) Hand ROI within a depth image. (c) Binary hand ROI.

*2) Depth image binarization*

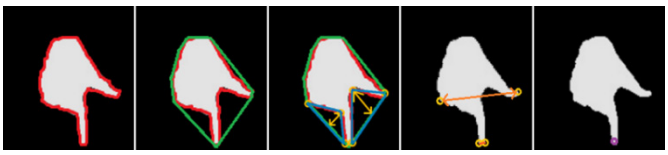
Next, we binarize the hand ROI by thresholding the depth. A binarized result $ROI_B$ is then extracted by

$$ROI_B(x,y) = \begin{cases} 1, (D-TH) \le Pixel_D(x,y) \le (D+TH) \\ 0, \quad \text{Otherwise} \end{cases}. \quad (3)$$

$Pixel_D(x,y)$ is the depth at pixel (x,y) in the ROI. $D$ is the depth of the Kinect hand joint. $TH$ is a threshold value defined experimentally. A binarized image result is shown in Fig. 5(c).

*3) Extract fingertip point*

In our system, the Kinect camera is set up on the top of a 3D monitor. Hence, a regular view captured by the Kinect camera would look like Fig. 5(a). Also, a specially defined hand gesture like Fig. 6 is required to activate the interaction function when the fingertip touches a 3D virtual object. In order to realize the interaction function, we apply the OpenCV function to detect convexity defects. The function uses a hand contour like Fig. 6(a) and the convexity hull like Fig. 6(b) as inputs. The outputs are convexity defects. The $i$th defect is modeled by a start point $(S_p^i)$, an end point $(E_p^i)$, a farthest point $(F_p^i)$, and the distance from the outer edge to the farthest point $(Dis_i)$. Next, via the detected defects, we recognize the defined hand gesture and locate the fingertip. Since the gesture is form by two large and closed defects, we verify a binarized ROI as a correct gesture if only two larger defects are detected by thresholding the value $(Dis/D)$ and the minimum distance of the set $\{\overline{S_p^1S_p^2}, \overline{S_p^1E_p^2}, \overline{E_p^1S_p^2}, \overline{E_p^1E_p^2}\}$ is smaller than a threshold. Here, number 1 and 2 mean the 1st and 2nd large detects. Fig. 6(c) and 6(d) show examples of two largest defects (blue triangles) and two point distances. After finding the minimum point distance, the fingertip location can be extracted as shown in Fig. 6(e).



(a) (b) (c) (d) (e)

Fig. 6. Schematic diagram of fingertip detection steps

## III. Experimental Result

In order to evaluate our system, we evaluate the accuracy of fingertip detection and whole system respectively. First, we calculate the mean error between the true fingertip point and the estimated fingertip point. The result is shown in TABLE I

TABLE I
FINGERTIP DETECTION ERRORS (distance unit by m)

| | X axis | Y axis | Z axis |
|---|---|---|---|
| **Average error** | 0.01098 | 0.00129 | 0.00243 |
| **Positive direction error** | 0.01744 | 0.02265 | 0.02541 |
| **Negative direction error** | 0.00262 | 0.00795 | 0.01559 |

To evaluate the perceptive performance of the whole system, we change the viewing directions and locations and calculate the location difference between a visualized 3D object and its expected object position. The result is presented in TABLE II.

TABLE II
COORDINATE CALIBRATION (distance unit by cm)

| | Face to the monitor | By the right side of the monitor | By the left side of monitor |
|---|---|---|---|
| **Average error** | 0.165 | 0.225 | 0.640 |
| **Maximum error** | 0.400 | 0.500 | 1.000 |

## IV. Conclusion

The purpose of this paper is to realize a practical interaction system between the real and virtual worlds in terms of (1) reality (2) stability. In order to enhance user visual perception and improve interactive reality, we calibrate the human-visual coordinate system and the virtual-world coordinate system. This step is critical to make the visualized 3D object position consistent and invariant to human eye parallax, locations, and viewing angles. On the other hand, we improve the accuracy of fingertip detection and create a stable interaction system by considering the skeleton information and the depth image.

## Reference


[1] Jiahui Wang, Yuman Xu, Haowen Liang, Kunyang Li, Haiyu Chen, and Jianying Zhou, "Virtual Reality and Motion Sensing Conjoint Applications Based on Stereoscopic Display," in *Progress In Electromagnetic Research Symposium (PIERS)*, 2016, pp. 3382-3386.

[2] Valentino Frati and Domenico Prattichizzo, "Using Kinect for hand tracking and rendering in wearable haptics," in *World Haptics Conference (WHC)*, 2011, pp. 317-321.

[3] Bian Junxia, Yin Jianqin and Wei Jun, "Hand detection based on depth information and color information of the Kinect," in *Proceedings of the IEEE Chinese Control and Decision Conference*, 2015, pp. 4205–4210.