HUMAN COMPUTER INTERACTION BASED ON HAND GESTURE USING RGB-D CAMERA

Rajesh T , Gowri Saranya S

AB STRACT-In the mass-market products several new interaction paradigms are exists, but they are still lagging in effective interactions such as button, touch screen etc. In this project develop a novel human computer interaction (HCI) interaction involving no human touch technology by using gesture tracking. In order to track gesture use RGB-D sensor and finally integrate with windows real-time kernel such that based on our gesture human can scroll, zoom, select, etc., and perform much more interactions. Finally our interaction provides cheaper way of interface in many public spaces: lobbies, shops, coffee houses, museums and so on. So, interaction with public displays will be made easier and don't need an additional devices to control screen content.

Index terms – RGB-D Sensor, Human Computer Interaction.

I. INTRODUCTION

Virtual interaction technology is a new inexpensive means to resolve problems in respect of product design, interaction training, etc. Virtual Interaction System (VIS) principally adopts technologies such as virtual reality, augmented reality and mix of virtual and actual reality. In terms of human- computer interaction, it requires expensive hardware support including data glove, position tracker, synchronizer, 3D helmet-mounted display. In despite of high efficiency and strong sense of immersion, they are expensive and mostly customized for particular purpose, and their software platforms are not universal, so such virtual interaction can hardly be adopted in colleges and universities.

Manuscript received April, 2015.
***Rajesh T**, P.G student, Department of CSE, Dhanalakshmi srinivasan Engg college,Perambalur, Tamil Nadu, India.*

***Gowri Saranya S**, Assistant Professor, Department of CSE, Dhanalakshmi srinivasan Engg college,Perambalur, Tamil Nadu, India.*

They are not suitable for popularization and the application in experiment and practice education.The VIS based on PC desktop system adopts WIMP interaction in which Windows, Icon, Menu and Point-Click are main interactive elements, and mouse and keyboard are main interactive devices [1]. The advantage of VIS using WIMP lies in its low price, and the "point-click" input is accurate and possible to relize shortcut operation in 2D interaction.

However, the disadvantage is that the operation becomes complicated when interacting with 3D object and the user experience is poor, and it is impossible to make best use of the imagination and immersion of virtual reality system. This paper study on Microsoft Kinect in combination with virtual reality engine Unity3D to establish inexpensive virtual interaction experiment system with strong sense of immersion based on desktop. It collects user's space motion information including body motion, gesture, voice, facial expressions as interactive channel to virtual interaction system. This makes it possible to accomplish virtual interaction tasks by natural habitual behavior.

Current researches on Kinect as a 3D virtual interaction device mainly focus on gesture recognition. The content of gesture recognition consist of two aspects: preprocessing which including hand tracing, gesture segmentation, feature extraction, etc. [2] and gesture motion trajectory tracking and classification. Reference [3] presented a method adopting 3d coordination system to get center point of gravity in hand depth image, and the recognition accuracy rate is up to 90%. Reference [4] presented a method for fingertips detection and center of palms detection distinctly for both hands using MS Kinect in 3D from the input image.

II. VIS FRAMEWORK BASED ON SOMATOSENSORY INTERACTION

A. Kinect Interaction

Originated in the game industry, somatosensory interaction refers to human-computer interaction realized by user's body motion, body sensation, voice, etc [7]. In general, somatosensory interaction cannot be achieved without data glove, 3D helmet and other multiple hardware. However, methods have been caught to capture body motion data to achieve human-computer interaction with the development of computer image recognition, computer. vision and other technologies. For example, Yoshihiro have proposed the concept of Virtual Input Devices based on motion capture and collision test [8]. Kinect developed by Microsoft is the realization of Virtual Input Devices concept. It can implement somatosensory interaction without using any handheld device. By this way human-computer interaction tasks can be completed through somatosensory interaction without support of expensive hardware.The appearance of Kinect is shown in Fig.1, Main components include the infrared transmitter, RGB color camera, 3D depth sensor composed of infrared CMOS camera and microphone array. Kinect has the function of AF tracking and the base motor can drive Kinect to turn 27 degree left and right.

Fig 1.Kinect appearance

The difference between Kinect and common camera lens is the CMOS infrared sensor of Kinect, which can be used to perceive the environment. Such sensor is designed to perceive environment by means of black and white spectrum: absolute black represents infinitely distant and absolute white represents infinitely close. The grey zone between black and white corresponds to the physical distance between object and sensor. It collects every node within range of view and form a depth image representing surrounding environment. The sensor generates depth image flow at a speed of 30 frames per second to reproduce 3D surrounding environment in real time, and then the software layer performs scene recognition, hand recognition, bone node tracking, and even facial expression recognition on 3D image via advanced algorithm in order to complete 3D input and output [9].

B. Design of VIS Based on Somatosensory Interaction

The VIS framework based on Kinect somatosensory interaction includes three layers: user input layer, Kinect interaction layer and virtual interaction application layer. Compared to traditional desktop-based virtual interaction system, there is a front device of Kinect between user and virtual system to collect user somatic information instead of mouse and keyboard.

The development of Kinect application is principally based on Microsoft's official SDK [10] or open drivers supported by open source community, such as OpenNI framework of OpenNI. Owing to the development interface for Unity 3D engine provided by OpenNI SDK, this paper employs OpenNI framework to develop virtual interaction system for the purpose of improved efficiency in development.

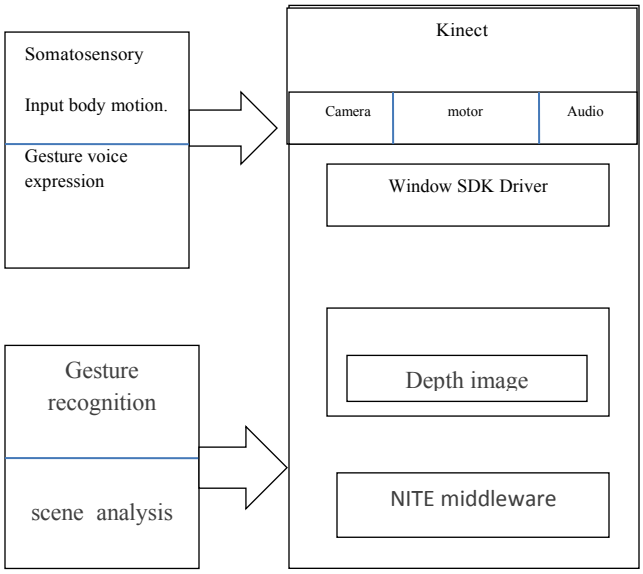


Fig 2. Kinect somatosensory interaction VIS framework using

The NITE middleware of OpenNI shall be installed before OpenNI Framework is put into use. NITE defined basic gesture analysis, gesture recognition, whole body analysis, scene analysis and other core module functions. This research adopts OpenNI+NITE+OpenCV+Unity3D development platform based on Windows 7 operating system.

Virtual interaction scene and logical task management are implemented by means of

Unity3D engine. Unity3D is a popular virtual reality production engine with powerful editor function, physical system and rendering capacity for perfect support 3Ds max, Maya, XSI, etc. Additionally, Unity3D support finite-state machine (FSM) programming perfectly and can manage the virtual interaction state and logical relations easily.

Establishing interaction model data, hand gesture recognition and changing virtual interaction scene view by body motion are three key issues in developing somatosensory interaction virtual interaction system. And the "treelike hierarchy modeling", "gesture semantics extension" and "multidimensional head tracking" methods discussed as follow.

III. INTERACTION MODEL DATA ESTABLISHMENT

Every part must be removed from or installed into interaction following a definite sequence and spatial path in virtual interaction process. As per interaction, the entire virtual interaction environment is composed of moving objects and fixed objects, of which the moving objects mainly include virtual interaction targets, 3D menus and interaction tools. The virtual interaction target is quite complicated and can be further divided into three categories.

- Geometric model of part: Geometric model of part is created by CAD and 3DsMax software and output in form of dts as required by Unity3D.
- Interaction constraint model: Interaction constraint model is to define the constraint relations between parts and determines the spatial position matrix and assembling features of parts in interaction.
- Interaction hierarchy model: A hierarchy tree is employed to describe interaction hierarchical relations of virtual part model.

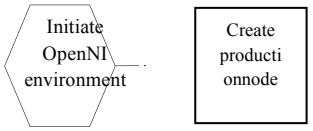
IV. KINECT GESTURE RECOGNITION AND SEMANTICS EXTENSION

The gesture is intuitive and natural, which is deemed as a significant human-computer interaction way for virtual interaction. The gesture interaction denotes the more intuitive interaction achieved via acquiring motion feature of hand, such as click, direction and motion trail of finger.

A. OpenNI for Kinect Development Process

OpenNI employs the definition of Production Nodes to describe its work procedure [11]. As a tool set, Production Nodes are available to generate data necessary for natural interactive program. Every Production Node can call data from bottom nodes and pass the data to upper nodes. The Production Nodes that generate data are called Data Generators. If an application is about to track human motion in a 3D environment, a Production Node will be available to generate body data (such node is called User Generator), and User Generator needs to read the date from Depth Generator. There are 9 Data Generators defined in OpenNI, such as commonly used UserGenerator, DepthGenerator, GestureGenerator, HandPointGenerator.

The development of Kinect based on OpenNI is principally reliant on Data Generators and related API functions to read and process raw image data in order produce meaningful 3D data for comprehensible and interpretable scene. Therefore OpenNI development process is adopted as shown in Fig.3.



B. OpenNI Gesture Recognition Process

Among OpenNI nodes, HandsGenerator nodes can track hand position to achieve gesture detection. Such nodes only need to track hand rather than to grasp whole body. OpenNI middleware defined four types of basic gestures: Wave, Click, RaiseHand and MovingHand. Gesture recognition process is shown in Fig.4.

Fig 4. Gesture recognition process

A Complete gesture recognition process covers six steps as follow: Initiate OpenNI and start read data from Kinect sensor. Create OpenNI node and get related data by DepthGenerator, GestureGenerator and HandsGenerator. Add four

types of pre-defined gesture to GestureGenerator, and define detection area if necessary. Register callback function of gesture processing. Start to process data flow and loop gesture detection. The registered callback function is invoked to execute corresponding events if a gesture is detected. For example, menu sliding function would be executed after a Wave gesture is detected.

In Fig.5, hand point position and four basic gestures can be detected following the process in Fig.4.

Fig 5. Hand point position detection and basic gesture recognition

The center data of hand point detected in previous frame will be saved and compared to center data in current frame. If deviation falls outside range of threshold, the center in previous frame will continue to be shown. Otherwise, the position of hand point center will be drawn again so that the gesture can be prevented from slight jitter.

C. Behavior Metaphor Design and Gesture Semantics Extension

1) **Gesture Extension Definition**:The purpose of somatosensory interaction is to enable user to complete interaction tasks through natural behavior in life. And those four basic gestures defined in OpenNI are far from enough for virtual interaction operation.

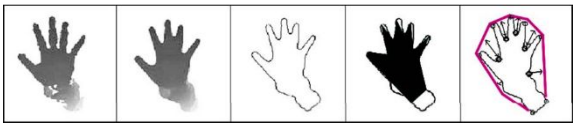
The extended gesture interaction and semantics is shown in Table 1.

TABLE I. GESTURE INTERACTION SEMANTICS DEFINITION

flag	Gest	Semantics
101	Move	Move parts
102	MoveRight	Move parts
103	Move	Move parts up
104	MoveDown	Move parts
105	WaveHand	Slide Menu
106	HandPull	Click
201	HoldFist	Select/Pick up
202	OpenFist	Unselect/Relea
301	2PalmsPullOp	Zoom
302	2PalmsPullClo	Narrow
303	2PalmsRotate	Rotate Parts

2)**Algorithm And Steps:**

Gestures of flag 101 to flag 106 could be recognized by comprehensive use of those four basic gestures predefined in OpenNI according to the values of position, angle, distance, and etc. Gestures of flag 201 to flag 303 are more complex, it needs to make a more complicated processing in palm and fingertips on the basis of OpenNI hand tracing. So this paper proposed the convexity detection algorithm for gesture recognition based on Sklansky's 3-coins algorithm [13]. This approach attempts to locate the convex points and concave points of the hand shape. Then according to the number of convex and concave points detected, a gesture and its semantics is recognized. This approach requires less number of parameters, and it proves to be more robust in our application.Firstly the hand node and hand image should be obtained by HandsGenerator as shows in Fig.4, then the hand depth image can be separated by threshold processing according to depth value of palm center, and the result is shown infig.6(a).



(a) (b) (c) (b) (c)

The remaining key steps are as follows.

a) Median filtering: Hand gesture depth image extracted from Kinect contains noise and a median filter with a window size of 10 x 10 pixels was used to remove the noise while preserving image edges better. This filtering operation can be written mathematically as:

$$F(i,j)=\text{median}(S((i+k),j+1)h(k,l)) \quad (1)$$

Where S is the source image as shown in Fig.7 (a) and h is the uniformly-weighted median-filtering kernel. This pre-processing result is shown in Fig.7 (b). Obviously the pre- filtered image got smooth edges and reduced the number of unwanted convexity points. The corresponding OpenCV function is <medianBlur>.

b) Hand contour tracing: The hand contour was traced using the OpenCV function <findContours> in order to find the curvature points of the hand shape. The contour values were saved in a vector of Point type. Only acquire the hand contour can reduce the computational complexity because of restricting the computations only to the hand contour but not the entire image. The drawing of hand contour is shown in Fig.8.

b) *Approximating the hand contour with a polygon:* This step aims to reduce the unwanted convexity points number by approximating the hand contour with a polygon that has fewer vertices function <approxPolyDP> which is based on the Douglas-Peucker graph algorithm. The polygon calculated could be drawn by function <drawContours> as shown in Fig6 (d).

d) Calculate the convex and concave points of the approximation polygon: This work was done by OpenCV function <cvConvexityDefects> which is based on the Sklansky's graph algorithm. Using this function we got the convex hull and points set as shown in Fig.6 (e).

e) Extended gesture recognition: Extended gesture recognition could be finished by numbers of concave and convex points, and their geometrical relationship with hand center point. For example, HoldFist gesture is recognized if there is no concave point and all the convex points need to be within a threshold radius from the approximated center of the palm. On the contrary, the OpenFist recognition condition is that there are at least four concave points. A related callback function would be executed after a gesture recognized successfully, such as picking up an interaction tool in virtual scene by function <VISPickupTools>.

In addition, a middleware named FAAST was employed to map extended gesture into keyboard and Unity3D input signal. FAAST is a gesture recognition library complied by use of OpenNI in combination with Kinect [14]. So by this way we can use gestures to complete some of the shortcut key combination operation.

V. SOMATOSENSORYCONTROL OF VIRTUAL INTERACTION SCENE VIEW

Rapid changing the scene view to right angle for interaction operation in 3D system is a high-frequency behavior, and it is fairly difficult in WIMP interaction because of the need to perform a complex series of steps at the same time, such as zoom, rotation, camera track and other key combination operation. Now Kinect can reduce the complexity easily. It can collect the data of distance and angle offset of user's head, and the data would be utilized to control the movement of camera in the scene so that the view of scene will be changed along with the move of user's head. When user approaches the screen, the view will be magnified automatically and vice versa. The 3D scene view will be changed along with the head moving for a good effect of hologram.

Firstly, DepthGenerator shall be used to acquire depth image. The pixel value of projective coordinate system of depth image directly denotes depth value (unit: mm), representing vertical distance between a certain point and plane of Kinect, thus the distance of head node will be acquired. The angle offset data of head node will be acquired via function <GetSkeletonJoint> of OpenNI SkeletonCapability, and so it could read information regarding joint position and angle. Using parameter <XN_SKEL_HEAD> in such function can specify only head node is read rather than whole body.

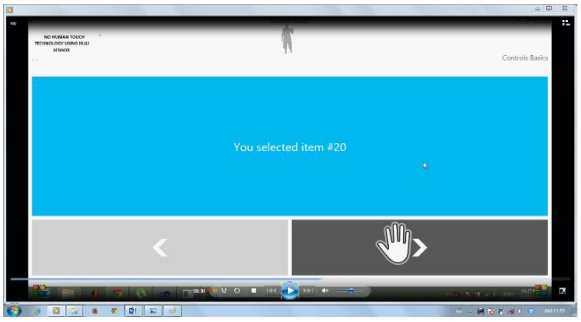
Secondly, a target camera is added to Unity3D scene and a script file is added to control it. The citation of OpenNI DLL shall be added to script file. The information regarding head position and angle can be acquired by real-time tracking of user's head node, and the position coordinates of camera are updated immediately by these data. The Y value in Kinect and the Y value of Unity3D target is reverse, So it is necessary to carry out spatial coordinate transposition from Kinect to U3D.

Fig.8 shows visual effect of scene view changing by user's head motion.

Fig 7. Scene view change controlled by head move

VI. VIRTUAL INTERACTION EXPERIMENT SYSTEM

The speed reducer is a typical teaching example for machinery manufacture major student in college. This research employed Microsoft Kinect somatosensory technology to realize 3D windows Virtual Interaction Experiment System. The systematic hardware consists mainly of Kinect, personal computer, common projector and projection screen. The interaction model was created in windows8 system. The secondary developing tool of ObjectARX was employed to acquire the parts, features, restraints and other information from model file. The Unity3D was employed to create, manage and render virtual interaction scene. The somatosensory interaction logic management was realized by means of Unity3D script files, and hand gesture recognition is realized by means of OpenNI2.1 and the convexity detection algorithm. The virtual interaction system operation is shown in Fig.8.



VII. CONCLUSIONS

This paper discussed the key issues of establishing interaction model data, gesture recognition and changing virtual interaction scene view by body motion in virtual interaction experiment system. The classification and feature description of interaction model established the basic logical relations and laid significant foundation for system realization. As the kernel of this system, the gesture recognition and its semantics extension reduced reliance on traditional mouse and keyboard, so the interaction was more natural and efficient than before The interaction scene view control by body motion enhances the user immersion. A Windows based Virtual Interaction System was designed and realized by these methods, and practical results show that the method of gesture recognition has a good accuracy and robustness, fast, correctness, finite detection and Kinect somatosensory interaction can improve interaction efficiency and enhance user immersion with low–cost in 3D virtual system.

REFERENCES

[1] R. J.K Jacob, A. Girouard, L. M. Hirshfield, et al., "Reality-based interaction:a framework for Post-WIMP interfaces," Proc. Proceeding of the twenty-sixth annual SIGCHI conference on Human factors incomputingsystems,ACM,Sep.2012,pp.201210, doi:10.1145/1357054.1357089.

[2] J. Suarez, R. Murphy, "Hand gesture recognition with depth images: A review," RO-MAN, IEEE Press, Sept. 2012, pp. 411- 417, doi:10.1109/ROMAN.2012.6343787.

[3] T. Wan, Y.J. Wang, J.K. Li, "Hand gesture recognition system using depth data," Proc. Consumer Electronics, Communications and Networks (CECNet), IEEE, Apr. 2012, pp. 1063-1066, doi:10.1109/CECNet.2012.6201837.

[4] J.L. Raheja, A. Chaudhary, K. Singal, "Tracking of fingertips and centers of palm using Kinect," Proc. Computational Intelligence, Modelling and Simulation (CIMSiM), IEEE, Sept. 2011, pp. 248–252, dio:10.1109/CIMSim.2011.51.

[5] H. Haggag, M. Hossny, S. Nahavandi, D. Creighton, "Real Time Ergonomic Assessment for

Interaction Operations Using Kinect," Proc (Computer Modelling and Simulation UKsim) , Apri. 2013, pp. 495500, doi: 10.1109/UKSim.2013.105.

[6] Y.Y. Chen, "Gesture recognition based on Kinect and application in the virtual interaction technology," Electronic Design Engineering, vol.21, May. 2013, pp. 4-7.

[7] J. E. Brough, M. Schwartz, S. K.Gupta et al., "Towards Development of a Virtual Environment-based Training System for Mechanical Interaction Operations," Virtual Reality, vol.11, n.4, Sep. 2007 pp.189-206, doi:10.1007/s10055-007-0076-4.

[8] Y. Okada, K. Shinpo, Y. Tanaka, D. Thalmann, "Virtual Input Devices based on Motion Capture and Collision Detection," Proc. Computer Animation, IEEE Press, May 1999, pp. 201-209.

[9] I. Tashev, "Kinect Development Kit: A Toolkit for Gesture and Speech-Based Human-Machine Interaction," Signal Processing Magazine, vol. 30, May 2013, pp. 129-131.

[10] J. Webb, J. Ashley, "Beginning Kinect Programming With the Microsoft Kinect SDK," California: Apress, 2012, pp.4-6.

[11] OpenNI. OpenNI User Guide, 2011, pp.3-6, unpublished.

[12] D. Saffer, "Designing For Interaction Creating Innovative Application and Devices," 2rd ed., New Riders Press, 2009, pp23-25.

[13] G. Klette, "A recursive algorithm for calculating the relative convex hull," Proc. Image and Vision Computing New Zealand (IVCNZ), IEEE, May. 2012, pp. 1-7.

