Gesture control system for Unity

Andrey Kolchanov
Skolkovo Institute of Science and Technology
ul. Novaya, d.100, Karakorum Building, 4th floor,
Skolkovo, 143025 Russian Federation
+7 (495) 280 1481
andrey.kolchanov@skolkovotech.ru

Vladimir Eremin
Skolkovo Institute of Science and Technology
ul. Novaya, d.100, Karakorum Building, 4th floor,
Skolkovo, 143025 Russian Federation
+7 (495) 280 1481
vladimir.eremin@skolkovotech.ru

ABSTRACT
The purpose of this research is developing a mechanism of creation custom continuous gestures for Senz3D camera. The project allows manipulating (move, zoom, rotate, select parts, cut) 3D model of human body in Unity game engine using gestures; and will use the Oculus Rift helmet to create a virtual reality of surgeon operations. This application will be used by surgeons for education purposes.

Keywords
Time-of-flight (ToF) camera, 3D model, game engine, gesture recognition, HMM, Senz3D, Intel Perceptual Computing SDK, Oculus Rift.

1. INTRODUCTION
Modern medicine uses a lot of technologies that help doctors make decisions and perform operations in a more precise or efficient way.

One of the current trends is a patient-specific treatment [1], which assumes that doctor has a patient's 3D model that represents all of the specific details about him and doctor is able to customize procedures for him.

Such systems are also very important because they allow doing basic operations in a virtual environment, without risking of real patients. This is especially a great way for teaching and training - simulator allows any number of operations at any time.

2. PROBLEM
Time is money, especially when it comes to peoples' lives. That is why it is crucially important to have a fast access to the 3D model of a patient.

Secondly, it is very important that the system allows manipulating it without touching any unsterilized surface and not putting the surgical gloves off. Otherwise it is again a waste of time and an additional risk of infection.

These requirements put a lot of complications into the process of development of such system.

3. BACKGROUND
Currently there are several ways to solve the problem of touch free interaction with computer-based systems.

One of them is a voice control. It has an advantage that it can be used while maintaining any other physical activity but it is not very accurate and rather slow.

The other option is a gestural input which has other problems but is more reliable and smaller input lag can be achieved.

One of the first gesture recognition systems was presented by Maggioni and Rottger [2] at Siemens. This development uses a video projector which displays a user interface onto any surface and a video camera to capture the hand of a user. By moving the hand, the user can move objects on the projected desktop. This interface uses dynamic hand movements to control the system. Another possibility are lexical gesture based systems. Such systems use static hand gestures to control the system. A static gesture based system was presented by BMW and is described in paper [3]. The system is used to control the infotainment inside a vehicle. For image capturing, a standard webcam is employed which makes segmentation very complex compared to a TOF camera.

The University of Bielefeld developed a camera based system for static gestures based on neural networks. It is called Gesture Recognition based on Finger Tips (GREFIT) as described in Noelker and Ritter[4]. The system works in two steps: first, the position of the finger tips is calculated in global coordinates by using neural networks, and in the second step the gesture is reconstructed by a model of the hand. Another work is presented by Athitsos and Sclaroff[5]. They use a model based approach to classify 26 static gestures. For each gesture a set of 4128 different views is stored. Hence, the complete database consists of 107328 pictures. The captured image is compared to the stored images from the database.

Chang et al. [6] presents a feature extraction based approach based on Curvature Scale Space (CSS) for translation, scale, and rotation invariant recognition of hand gestures. The CSS image is used to represent the shapes of boundary contours of hand gestures. Nearest neighbor techniques are used to perform CSS matching between the multiple sets of input CSS features and the stored CSS features for hand gesture recognition. For six gestures and 300 sets of data, the recognition rate is 98.3 %. All systems presented so far use standard cameras for image recording. These cameras do not provide depth information about the scene.

4. PROPOSED SOLUTION
Our solution utilizes a system for gestural control using ToF Senz3D camera as an input device and a Unity Game Engine to visualize the model and operations (see fig.1).

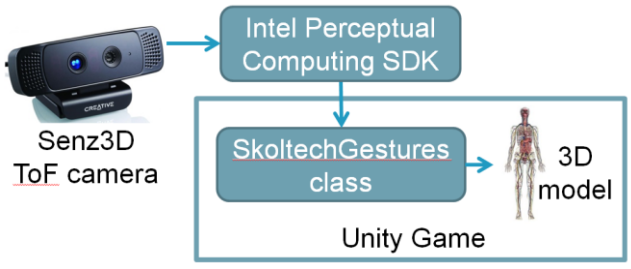

Figure 1. System architecture

Senz3D camera provides us with a depth map and a set of standard points (GeoNodes) on each hand. This information is used directly in our custom continuous gestures to determine the amount of change that has to be imposed on the controlled object. Some standard gestures from camera SDK are also used to trigger some of the continuous gestures.

5. SYSTEM COMPONENTS

The system consists of hardware and software parts.

5.1 HARDWARE

Creative Senz3D camera was used for acquiring distance information. The camera is plugged to a computer with USB 2.0. The drivers are included into Intel Perceptual Computing SDK and are available for Windows only.

5.2 SOFTWARE

The software part includes several components.

5.2.1 Unity 4.3.1

This environment provides us with a set of tools to manipulate the model which has to be imported.

5.2.2 Intel Perceptual Computing SDK 2013 R6[7]

This package is vital for using Senz3D camera. It includes drivers, example projects, one of which was used for debugging.

6. ALGORITHMS

The application that was created can recognize three gestures:

- Swipe. We are getting coordinates of thumb and index fingers and calculate distance between them;
- Rotate – rotate the 3D object along the axes. This gesture consist of two parts: firstly we recognize a standard gesture "moving hand", and secondly we measure the distance between hand and camera;
- Move – we just recognize the standard gestures of movement to up, down, right and left.

Every 3D object in Unity can has connected C# class with two function by default: *void Start()* and *void Update()*. The function *void Start()* called when game started and should allocate memory for variables C# class has and initialize them. The function *void Update()* called every tact, and this function is used to manipulate a 3D object. The class for 3D object should be inherited from *MonoBehaviour* class.

Using the access to the instance of the *PXCUPipeline* class in void Update() function we can get Geo Nodes and gestures that were recognized. There are two queries in PXCUPipeline class: QueryGeoNode and PXCMGesture. Intel Perception SDK places the recognized gestures and geo nodes to these queries.

Next step is the processing the standard gestures and coordinates. A special class called *SkoltechGestures* was created for it. The *SkoltechGestures* class consist of:

1. Variable describe the current state of application: zooming and rotating;
2. Function void newZooming(double depth) – it called when we calculate new distance between fingers;
3. Fucntion void newDistance(double distance) - it called when we calculate new distance between camera and hand.

SkoltechGestures class uses Boolean logic for making a predication about current gesture. A code that uses *SkoltechGestures* class can call *int State()* function to get information about current gesture.

After calls of methods of SkoltechGesture class we can make some actions on the 3D object. Almost all actions on the objects are performed using the *transform* variable. There a lot of variables like a transform.position, transform.rotation for manipulate an object. In the application is implemented as follows:

```
switch (skoltechGestures.State) {
        case (1)://zoom in
                speed = new Vector3 (0, 0, -25.0F);
                transform.Translate(speed * Time.deltaTime);
                break;
        case (-1)://zoom out
                speed = new Vector3 (0, 0, 25.0F);
                transform.Translate(speed * Time.deltaTime);
                break;
        case (3)://rotate from up to down
                transform.Rotate(Time.deltaTime * (-200));
                break;
        case (4)://rotate from down to up
                transform.Rotate(Time.deltaTime * 200);
                break;
}
```

It is not enough to simply apply these methods to operations on 3D objects. When you rotate object 180 degrees around an axis, gesture "move left" worked properly early will work in unusual way: it can move object to the right. The solution is to realize the absolute position of the object in *SkoltechGestures* class and make calculations about transform methods when some gesture had recognized.

7. RESULTS

We created the Unity application with gesture control 3D model of human body with Senz3D camera and SkoltechGesture class to recognize custom continuous gestures. It recognizes default Intel Perception SDK gestures and custom continuous gestures as well, and allows to make the next operations on the 3D objects:

1. **move object** using standard gesture *move left*, *move right*, *move up* or *move down*;

Figure 2. Move gestures

2. **zoom object** using custom continuous gesture swipe;

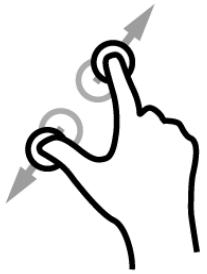

Figure 3. Zoom gesture

3. **rotate object** using standard *circle* and continuous custom gesture *distance* afterwards.

Figure 4. Rotate gesture

Gesture recognition accuracy depends on a gesture type. It works perfectly for ‘move’ and ‘rotate’ gestures, because the application waits for ‘move’ and ‘round’ standard gestures first; standard ‘route’ gesture enable ‘rotate’ mode. Precision of recognizing ‘swipe’ gesture is about 60%. The reason for this variation is described in the limitations section.

8. LIMITATIONS

Intel Perception SDK couldn’t guarantee that position of finger is real position of exactly finger that we ask. Intel Perceptual Computing SDK can confuse the fingers and say that coordinates of the middle finger is coordinates of the index finger. The consequence of this is inability to analyze 100% coordinates finger gestures for gestures analysis. The way to solve it is analyzing big jumps of distance between fingers, and if it happened - stop recognizing the current gesture.

9. NEXT STEPS

Current prototype system cannot provide user with an interface for convenient working. But there are several ways to improve it.

First of all it is obviously necessary to add more custom gestures to enable extra types of operations with the model: select parts of human body, cut, delete, and make other virtual surgery operations.

Then we are going to use Hidden Markov Models or some other methods that can help to differentiate gestures in a more reliable way.

Finally we are going to try using it with Oculus Rift to build a complete system.

10. ACKNOWLEDGMENTS

Our thanks to Victor Lempitsky for suggesting to use Senz3D camera in our project and Vasiliy Kramarenko for providing us with a 3D model of a human blood system and inspiring with some ideas.

11. REFERENCES

[1] Amit Gefen(Ed.) Patient-Specific Modeling in Tomorrow's Medicine. Studies in Mechanobiology, Tissue Engineering and Biomaterials, Vol. 09, 2012, 532p.

[2] Maggioni, C. and RÄottger, H. (1999) `Virtual Touchscreen - a novel User Interface made of Light - Principles, metaphors and experiences', Proceedings of HCI International on Human-Computer Interaction: Ergonomics and User Interfaces-Volume I, Vol. 1, pp.301{305, 22-27 August, Munich, Germany

[3] Nolker, C. and Ritter, H. (2002) `Visual Recognition of Continuous Hand Postures',IEEE Transactions on Neural Networks, Vol. 13, No. 4, pp.983{994, July 2002

[4] Akyol, S. and Canzler, U. and Bengler, K. and Hahn, W. (1999) `Gestensteuerung fur Fahrzeugbordsysteme', Informatik aktuell. Mustererkennung 2000. 22. DAGM Symposium, pp.139{146, 13-15 September, Kiel, Germany, G. Sommer, N. KrÄuger and Ch. Perwass (Eds.), Springer Verlag

[5] Athitsos, V. and Sclaro®, S. (2003) `Estimating 3D Hand Pose from a Cluttered Image', Proceedings of the 2003 International Conference on Computer Vision and Pattern Recognition, Vol. 2, pp.432{439, 16-22 June, Madison, USA

[6] Chang, C.-C. and Chen, I.-Y. and Huang, Y.-S. (2002) `Hand Pose Recognition Using Curvature Scale Space', Proceedings of the 16th International Conference on Pattern Recognition, Vol. 2, pp.386{389, 11-15 August, Quebec City, Canada

[7] Intel Perceptual Computing SDK http://software.intel.com/en-us/vcsource/tools/perceptual-computing-sdk.

[8] I Oikonomidis, N Kyriazis, A Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. *01/2011; In proceeding of: BMVC 2011*