Manipulation of virtual objects through a LeapMotion optical sensor

Stoyan Kerefeyn[1], Stoyan Maleshkov[2]

[1] Virtual Reality Laboratory, Technical University Sofia, Sofia 1797, Bulgaria

[2] Virtual Reality Laboratory, Technical University Sofia, Sofia 1797, Bulgaria

Abstract

The purpose of this paper is to present a gesture-based approach for controlling and manipulating remotely virtual objects in a virtual reality environment through a LeapMotion optical sensor. The interaction is performed by a separate software module, which is installed on the same system, where the virtual reality application is put into operation. The software architecture, implementation and user experience details are discussed and evaluated. In comparison to similar techniques our solution has the advantage to be intuitive, to provide means for creation of user interfaces without physical contact to the sensor, having minimal delay while performing the control actions.

Keywords: *LeapMotion, Gesture Recognition, Virtual Reality, Multimodal Presentation, Engineering Analysis Results Validation.*

1. Introduction

Virtual Reality (VR) and Augmented Reality (AR) are rapidly becoming powerful tools for design and development of new products, providing almost unlimited opportunities for improvements and minimizing time to market. They also provide an environment to simulate dangerous tasks, study hazardous materials or reactions without any risk to the involved individuals. Recognizing the benefits a virtual reality system can offer, some potential obstacles remain, which have to be resolved – mainly the manner the interaction with the system is designed and implemented. Traditional WIMP methods can be applied, which use windows, mouse, icons and pointers, but they are not always intuitive or responsive enough for the freedom a VR system provides. Other solutions like optical or magnetic trackers can be expensive [1]. With the advancement of technology in the last years even more affordable sensors are entering the market, which can be used with great success as interaction devices in VR systems. The solution for manipulation of virtual objects described in this paper relies on and utilizes such device – the LeapMotion sensor. LeapMotion [2] is a small hardware component, which can be connected via USB to a desktop computer running Windows or MacOS X operating system. It uses two monochrome infrared cameras and three infrared LEDs to track an approximated cone-shaped area with height of approximately 1 meter above the sensor. The LEDs generate a 3D model of infrared light dots and the cameras generate almost 300 frames per second of the reflected data, which are sent via the USB cable to the computer. Subsequently these data are analyzed with a specialized algorithm [3] that gives the developer exact and anatomically correct data for the position and movement of hands, wrists and fingers above the sensor, providing the opportunity for creation of a non-physical gesture-oriented input interface. The LeapMotion device has been selected for our study based on the results described in [4], where the LeapMotion has proved to provide better interaction functionality in comparison to the traditional mouse concerning single-target three-dimensional selection and manipulation tasks. In [5] the authors evaluated this controller as possible replacement of traditional interface devices for fast high-precision optical motion capture system operating in limited space and with limited number of objects. The method allows performing successfully complex tasks in 3D without constraints to the operator. In [6] the authors have developed a human-machine communication interface linking the Leap Motion controller to robotic arm, which principles can be also utilized for manipulation of objects in virtual reality environments.

2. The gesture-based approach

When we developed the concept of the approach described in this paper we structured the functionality and architecture following a top-down design principle. It was decided that the designed software solution will not expose an explicit visual user interface but will fulfill the function of a driver that connects the hardware sensor with a specialized application for manipulation and data sampling

of 3D engineering models in virtual reality environments. This allows the module to be applied as "stand-alone" input interface device in other similar VR applications with identical functionality and shortcuts. At future development stage it will be enhanced with additional specific features and will be integrated as part of a low cost VR system used for presentation of engineering analysis results.

2.1 Functionality through gestures

A main phase in the proposed approach is the functionality mapping between the VR application interaction features and hand gestures. The created input interface allows five types of interaction commands with objects in the virtual scene, defined as follows:

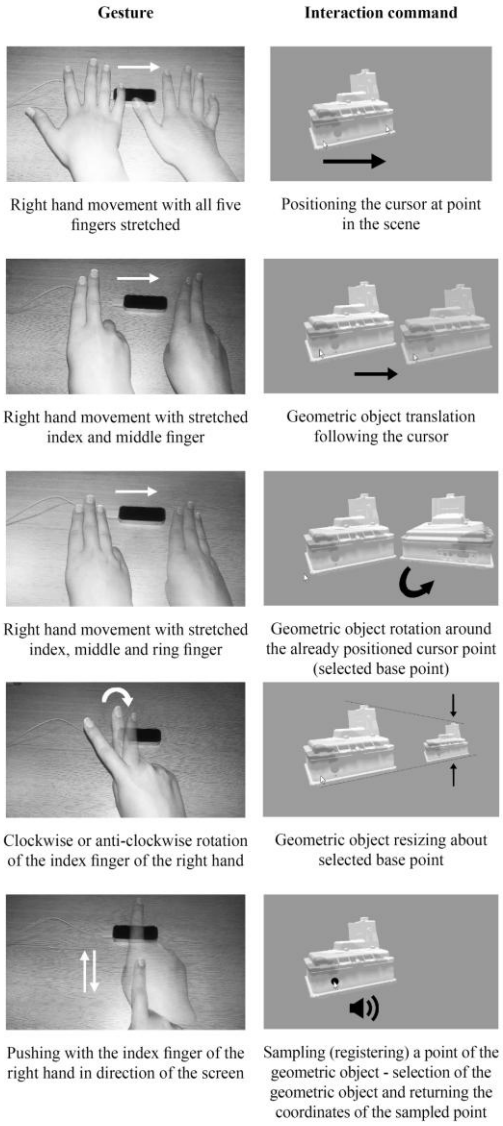

Fig. 1. Gestures defined for interaction with the specialized virtual reality software.

- Positioning the cursor at point in the scene;
- Geometric object translation following the cursor;
- Geometric object rotation around selected base point (the already positioned cursor point);
- Geometric object resizing about selected base point (the already positioned cursor point);
- Sampling (registering) a point of the geometric object – selection of the geometric object and returning the coordinates of the sampled point.

Each one of these interaction commands is typically performed using a combination of keyboard input, mouse movement and buttons pressing. These combinations can be attached to intuitive gestures defined by movements of the hands and fingers as described on Fig. 1

2.2 Sensor communication

The designer of the LeapMotion sensor have provided a well documented application programming interface (API) [7] that helps developers interact with the device. We have utilized and incorporated various programming objects and methods from this API in our business logic. The flowchart of the business logic is presented on Fig. 2.

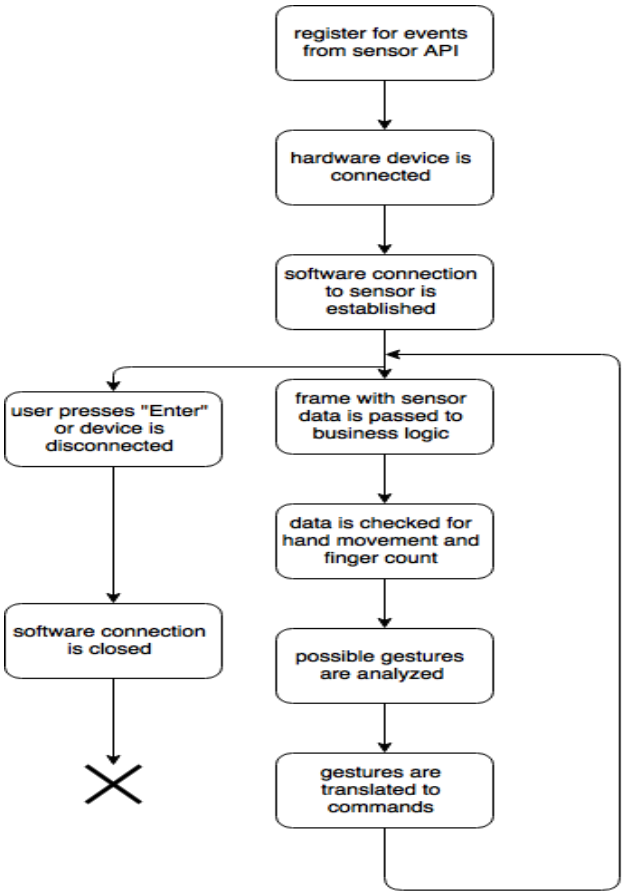

Fig. 2. Application business logic flowchart.

When the tracking module is started, a **com.leapmotion.leap.Controller** object is initiated as shown on Code 1. This programming object controls the initial starting parameters of the sensor and the sensor itself. The tracking module launches the sensor in background mode – this allows our application to receive and analyze sensor data even if it's not the active foreground application. An event listener is added, which includes the gesture recognition logic. The application enters an endless loop for recognition and execution until the keyboard Enter button is pressed. This action terminates the execution of the module and the listener is released. This basic approach is valid for all applications that use the LeapMotion sensor and can be reused.

```
public class Main {
    public static void main(String[] args) {
        // Create a sample listener and controller
        LeapTrackingManager listener = new LeapTrackingManager();
        Controller controller = new Controller();
        controller.setPolicyFlags(Controller.PolicyFlag.POLICY_BACKGROUND_FRAMES);

        // Have the sample listener receive events from the controller
        controller.addListener(listener);

        // Keep this process running until Enter is pressed
        System.out.println("Press Enter to quit...");
        try {
            System.in.read();
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Remove the sample listener when done
        controller.removeListener(listener);
    }
}
```

Code 1. Standard sensor initialization

The **LeapTrackingManager** event listener implements the abstract interface **com.leapmotion.leap.Listener**, which notifies the business logic for sensor events. When the listener is created another object **RemoteEventsListenerImpl** is initiated, which implements the interface **RemoteEventsListener** described in Code 2. The implementation of this interface translates the recognized gestures in computer commands.

```
public interface RemoteEventsListener {

    public void onZoomOut();

    public void onZoomIn();

    public void onMouseMove(float x, float y);

    public void onLeftMouseButtonClick();

    public void onItemTranslationStart();

    public void onItemTranslationEnd();

    public void onItemRotationStart();

    public void onItemRotationEnd();
}
```

Code 2. Functionality interface

The **LeapTrackingManager** event listener is registered to the LeapMotion controller, which is provided by the LeapMotion creators and pre-installed on the computer. This controller monitors the hardware connection to the device and notifies the event listener for changes in the connection state.

2.3 Gesture recognition

When a sensor is connected to the computer the **LeapTrackingManager** is notified from the **com.leapmotion.leap.Controller** and a software connection to the hardware device is established. The API gives us the opportunity to register for potential more complex gestures like finger rotation and finger push. In this case the gesture recognition for such pre-defined gestures is left to the API. Through the implementation of the **com.leapmotion.leap.Listener** the onFrame() callback is called continuously. This method (Code 3.) gives access to the current frame captured by the device.

```
@Override
public void onFrame(Controller controller) {
    // Get the most recent frame and report some basic information
    Frame frame = controller.frame();

    HandList hands = frame.hands();
    if (!hands.isEmpty()) {
        // Get the first hand
        Hand hand = hands.get(0);

        // Check if the hand has any fingers
        if(hands.count() == 1 && hand.isRight()) {
            int fingerCount = extendedFingerCount(hand);
            System.out.println("FingerCount: " + fingerCount);
            if(fingerCount > 3 ) {
                remote.onItemTranslationEnd();
                remote.onItemRotationEnd();
                calculateMouseMovement(frame, hand);
            } else if(fingerCount == 3) {
                remote.onItemTranslationEnd();
                remote.onItemRotationStart();
                calculateMouseMovement(frame, hand);
            } else if(fingerCount == 2){
                remote.onItemRotationEnd();
                remote.onItemTranslationStart();
                calculateMouseMovement(frame, hand);
            } else {
                remote.onItemTranslationEnd();
                remote.onItemRotationEnd();
                calculateGesture(frame);
            }
        }
    }

}
```

Code 3. onFrame() method

Hand movement and finger gestures are gathered from the frame-callback provided by the controller. The frame includes all needed information like hand and finger count and position, as well as information for eventually recognized gestures, that we are listening to. If both hands are detected in the infrared cone of the LeapMotion sensor, only the movement of the right hand will be tracked. At this stage of the development a limited array of gestures and functions are implemented, which can be easily represented with one-hand movements. The sensor offers tracking of both hands and in the future a setting for left-

handed users or more complex gestures involving both hands can be added. Every frame is processed with different algorithms based on the extended finger count as follows:

- If the fingers are more than three all candidates for rotation or translation gesture are neglected and the hand movement is interpreted as pure mouse cursor movement. The movement of the hand in the 3D space is translated to a 2D cursor movement, which is translated as command to the computer;
- If the extended fingers are exactly three we recognize intent of the user to perform geometric object rotation. The shortcut combination, that matches the command for rotation of the specialized virtual reality software is simulated and the further hand movement is processed to determine the direction of the rotation;
- If the extended fingers are exactly two we recognize intent for geometric object translation. The shortcut combination for translation is simulated and the hand movement is processed to determine the direction of the translation;
- If only one finger is extended we recognize intent for a specific finger gesture – either finger rotation or finger push. A specialized method **calculateGesture()** that deals with this event is called. This method describes an algorithm for interpretation of the recognized gesture as shown in Code 4.

```
private void calculateGesture(Frame frame) {
    GestureList gestures = frame.gestures();
    for (int i = 0; i < gestures.count(); i++) {
        Gesture gesture = gestures.get(i);

        switch (gesture.type()) {
        case TYPE_CIRCLE:
            CircleGesture circle = new CircleGesture(gesture);
            if (circle.isValid()) {
                // Calculate clock direction using the angle between
                // circle normal and pointable
                if (circle.pointable().direction().angleTo(circle.normal()) <= Math.PI / 4) {
                    // Clockwise if angle is less than 90 degrees
                    remote.onZoomOut();
                } else {
                    remote.onZoomIn();
                }
            }
            break;
        case TYPE_SCREEN_TAP:
            ScreenTapGesture tapGesture = new ScreenTapGesture(gesture);
            if (tapGesture.isValid()) {
                remote.onLeftMouseButtonClick();
            }
            break;
        default:
            System.out.println("[gestures] Unknown gesture type.");
            break;
        }
    }
}
```

Code 4. calculateGesture() method

Each LeapMotion frame offers a list of potentially recognized gestures. This list is cycled and if a valid, defined gesture is found it is linked to a matching command. If a finger rotation event is captured the movement direction of the finger is calculated – it is determined if the rotation is in clockwise or anti-clockwise direction. The recognized direction is matched to the user's intent to perform zoom-in or zoom-out operation. The actual zoom in/out command is executed with a pre-defined step – one finger rotation always zooms in or out applying one and the same angle of rotation. The push with a finger towards the screen is interpreted as sampling (registering) a point of the geometric object, what is equivalent to the command to perform a single left mouse button click at the current cursor position returning the corresponding 2D coordinates.

2.4 Translating commands to the computer

After a gesture is being recognized the command mapped to this gesture needs to be executed on the computer. With a standard input interface each command is represented as a combination of left and right mouse button clicks combined with mouse movement and keyboard button presses. These exact button combinations are mapped to the gestures and are virtually executed on the computer through the simulation interface **LeapTrackingManager**. The implementation of this interface uses the standard java class java.awt.Robot [8] for mouse and keyboard command simulation. The 3D hand movement is translated to 2D mouse movement using the following algorithm:

- Two sensor frames containing hand data are needed for processing – the current frame and the last frame – this is previous frame before the current frame;
- A 3D Vector is built by averaging the positions of all fingertips of both frames. This gives more precise coordinates than simple hand position;
- The X and Y coordinates of the vector of are separated and used in further calculations;
- To calculate the movement of the hand the position of the hand on the last frame is subtracted from the position of the hand on the current frame. This is performed by each of the X and Y axis separately;
- Jitter and faulty sensor data are taken into consideration and minor irregular movements, smaller than some predefined amount are discarded;
- The calculated hand movement is multiplied with an acceleration factor which is determined by taking into account the monitor's size and aspect ratio (for the regular case X factor is greater than those of Y);
- The resulted X and Y movement is passed to the java.awt.Robot mouseMove() method;

- The current frame hand data is set as last frame hand data and the calculations are repeated in loop from the beginning.

The processing of the sensor data described above is terminated when the sensor is physically disconnected or the Enter keyboard button is pressed and the software connection to the controller is deactivated.

3. Experimental study

As a test case we have extended the functionality of our affordable VR system, developed in the framework of a previous project [9] and used to present and evaluate engineering analysis results [10] with LeapMotion sensor implementing the described approach. The specialized application for manipulation and data sampling of a 3D model of an engineering object in a virtual reality system, as well as our LeapMotion module have been pre-installed on the system and an engineering model – in this case of a boiler – has been loaded. The interaction with this object was performed using on two different VR system setups: with standard LCD monitor (2D) and with 3D monitor. Although the two VR systems delivered different level of immersion, the LeapMotion module provided in both cases an intuitive input alternative to the mouse and keyboard input. As interaction task the user was requested to select the object (Fig. 3) and to translate it to a new position in the scene (Fig. 4). After that he/she had to rotate the model and to sample a specific point from the surface of the model. The sampling operation executed an audio feedback signal produced through the audio system of the computer. This feedback delivers additional information to the user about the displacement at the specific object point. We have carried out a usability test, which has indicated clearly that the presented approach allows easy and intuitive navigation in the VR system. The application can be successfully used in cases where physical access to I/O interfaces is not possible. Consider as an example the following scenario: the user is manipulating a real physical object (e.g. performing welding operation) and is enhancing his/her work experience with the simultaneous usage of the specialized VR application. The application is visualizing the same physical object in a VR environment but is also providing additional information about it, thus engaging the user in an AR environment. The additional information can present thermal data collected in real-time by additional sensors and superimposed to the image. Due to safety regulation the user has to wear protection gloves. In this situation the usage of traditional I/O interface devices will hinder or even make the operation impossible and simultaneous work with both the physical object and the virtual reality presentation could not be done. The LeapMotion sensor overcomes these difficulties – even with gloves the gestures and hand movements remain the same and that allows convenient manipulation of the objects in the scene. Just by pointing at a point on the virtual object the user receives visual and audible feedback providing additional information about the real physical object features.

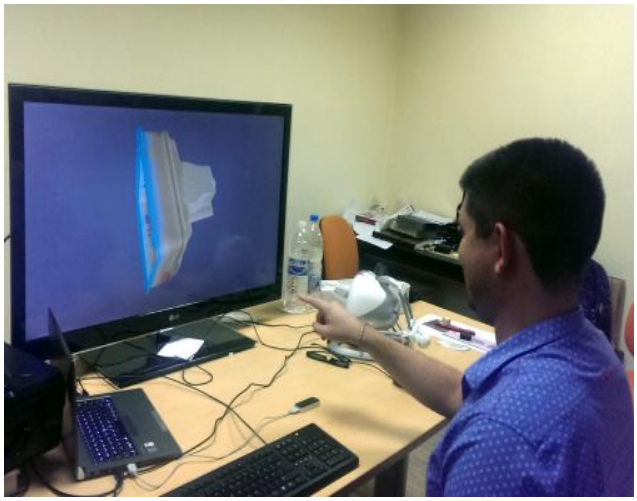

Fig. 3. Selection of a 3D object in an affordable virtual reality system.

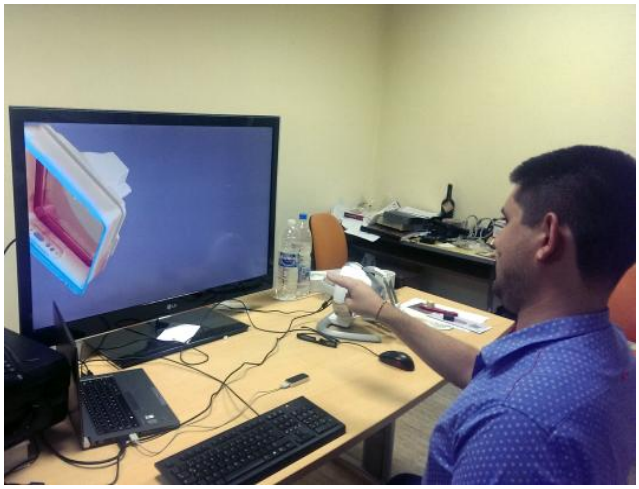

Fig. 4. Translation of the selected 3D object.

4. Conclusions

Our solution, which connects the LeapMotion sensor with the specialized virtual reality software, replaces a set of functions that are normally implemented with traditional input devices – mouse and keyboard using natural hand movements and gestures. This allows the communication with the application to be continued even in cases where physical contact with I/O interface devices is not possible providing means for seamless VR and AR experience.

Acknowledgments

The results presented in this paper are obtained in the framework of research activities funded by the Internal research grant of the Technical University of Sofia – 2015 and partly supported by LaSciSo (Large Scale Industrial Structural Optimization) project, funded under the FP7 PEOPLE Work Programme, Action IAPP (Grant 285782).

References

[1] D. Chotrov, S. Maleshkov, A. Bachvarov, " Interaction in virtual reality systems through gestures" (in German: "Interaktion in Systemen der virtuellen Realität durch Gesten"), Wissenschaftliche Konferenz Technik und Wirtschaft in der globalen Krise, ISSN 1310-3946, 26-27 November 2009, Technische Universität – Sofia, pp. 113-120.

[2] Official LeapMotion website https://www.leapmotion.com/

[3] F. Weichert, D. Bachmann , B. Rudak, D. Fisseler, "Analysis of the Accuracy and Robustness of the Leap Motion Controller", Sensors 2013, 13(5), 6380-6393; doi:10.3390/s130506380 http://www.mdpi.com/1424-8220/13/5/6380/htm

[4] J. C. Coelho, F. J. Verbeek, "Pointing Task Evaluation of Leap Motion Controller in 3D Virtual Environment", Chi Sparks 2014 Conference, ISBN: 978-90-73077-55-3 http://chi-sparks.nl/2014/wp-content/uploads/2014/Chi%20Sparks%202014%20proceedings-web.pdf#page=78

[5] J. Guna, G. Jakus , M. Pogačnik , S. Tomažič, J. Sodnik, "An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking, Sensors" 2014, 14(2), 3702-3720; doi:10.3390/s140203702 http://www.mdpi.com/1424-8220/14/2/3702/htm#b8-sensors-14-03702

[6] D. Bassily, C. Georgoulas, J. Guettler, T. Linner, T. Bock "Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller", Conference: the 45th International Symposium on Robotics (ISR 2014) and the 8th German Conference on Robotics (ROBOTIK 2014), 978-3-8007-3601-0

[7] Official LeapMotion development website https://developer.leapmotion.com/

[8] Sun, Oracle Java Robot API http://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html

[9] S. Maleshkov, D. Chotrov, "Affordable Virtual Reality System Architecture for Representation of Implicit Object Properties", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 2, July 2012, pp. 23-29.

[10] S. Maleshkov, D. Chotrov, "Post-processing of Engineering Analysis Results for Visualization in VR System", International Journal of Computer Science Issues, ISSN 1694-0784 Vol. 10, Issue 2, No 2, March 2013, pp. 258-263.

Stoyan Kerefeyn has received BSc. (2010) and MSc. (2012) degrees in computer systems and technologies from the Technical University of Sofia, Bulgaria. Currently he is PhD student, acting at the Virtual reality lab.

Stoyan Maleshkov has Eng. degree in system and control engineering (1975), master in applied mathematics (1977) and PhD in computer aided system design (1981), all received from the Technical University (TU) of Sofia, Bulgaria. Fulbright scholar (1989–1990) at the Interactive Modeling Research Lab, Louisiana State University, Baton Rouge, USA. Professor of computer aided engineering and computer graphics at the TU of Sofia. Department chair (2000-2004) and vice dean (2004-2008), both at the TU Sofia. Since 2008: Head of the Virtual reality lab, TU Sofia. Professor of computer graphics at the New Bulgarian University, Sofia, as a second job. IEEE member.