Gesture-based Navigation

Stefani Suchanek [1], Ivan Minárik [2], Gregor Rozinaj [2]
[1] Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovakia
[2] Institute of Multimedia ICT, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovakia
[1] *steffany995@gmail.com*, [2] *ivan.minarik@stuba.sk*, [2] *gregor.rozinaj@stuba.sk*

Abstract - **A housekeeper? It's a word that will soon become obsolete and go to archaism. In the modern world, where people come tired from the work where they spent all day, the best way to make life easier is to "order" to light up, to drop shutters, to turn on TV or boost music ... and yes, all this is possible without a home assistant! (*Abstract*)**

Keywords **- Gestures; Navigation; Kinect; Smart room**

I. Introduction (Heading 1)

Controlling a computer using gestures is a topic that is increasingly being discussed these days. It is one of the methods that helps a person to control the work of a computer and a general machine more naturally and quickly. We live in an era of great achievements, and this project should be done every day to be as efficient as possible. The most characteristic feature of the human species is constantly inventing the possibilities of facilitating life. Modern technological advances include, for example, computers with which people commonly communicate via conventional I / O devices such as keyboard, mouse, monitor and speakers. Methods of communication using these input devices are already obsolete for a person who communicates most often with speech and gestures. Therefore, new technologies and methods are being sought to identify natural human orders as accurately as possible.

The primary goal of the article is to identify definable gestures using the Kinect sensor and show the usage in real life. Implement the implementation in C#, through which the real-time percentage of the variable output can be controlled.

II. Ease of Use

Kinect is an input device, motion sensor for the Xbox 360 console, made by Microsoft. The device allows the user to communicate with the console without using the controller. It communicates with gestures and body as well as voice command.

Microsoft has provided Kinect with the basic software support, Software Development Kit 2, which it innovates and develops for Windows 8 and above. Various algorithms for skeleton detection are applied to the Kinect SDK v2. Under the skeleton is meant a simplified human skeleton. Kinect v2 can recognize six people and assign them six skeletons with the position of hinged joints in the space. In addition to skeletal detection, there are also possibilities to recognize three special hand states that can be incorporated in basic gestures [1].

III. Skeletal Tracking

Kinect for Windows SDK provides us with a set of APIs that allow easy access to the skeleton. The SDK supports up to 20 joint points. Each position is identified by its name (head, shoulders, elbows, wrists, arms, spines, hips, knees, ankles, etc.) and tracking status is determined by Tracked, Not Tracked, or Position Only.

Kinect v2 with SDK 2 can recognize the position of the following parts of the body: right and left hand end, right and left hand, left and right elbows, left and right wrists, right and left thumb, right and left shoulder, right and left feet, right and left ankles, right and left hips, left and right knees, head, neck, spine at shoulders, middle point of the spine, and spine at the loins [2].

A. Gesture Recognition

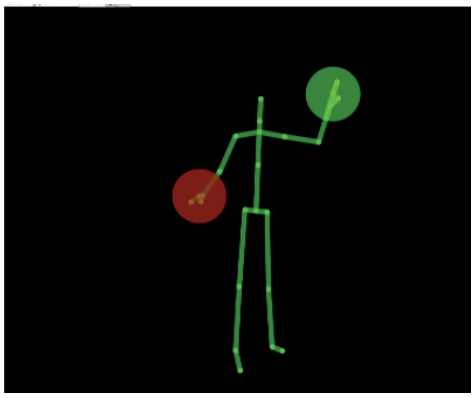

Figure 1. Example of Kinect gesture detection

Gesture is the movement of a human body or action that is intended to communicate with the application. In fact, there is no physical connection between the user and the device, using gestures of technology. Therefore, this technology forms the backbone of the NUI for the Kinect device. Gesture recognition is one of the most exciting processes and involves various calculations, algorithms, approaches and methods.

Procedures may vary depending on the gesture, so gestures can be divided into simple to complex. We classify gesture recognition procedures in the following ways:

- Basic gesture recognition
- Algorithmic procedure
- Weighted-network procedure

- Template-based procedure

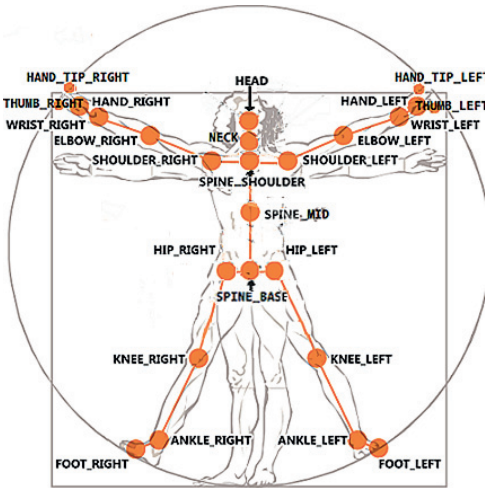

Figure 2. Skeleton points

Kinect Interaction uses a combination of depth data, skeleton data, sophisticated hand tracking algorithms, gesture recognition, and other features. Hand detection should be 1.5 to 2 meters away from the sensor and oriented directly towards the sensor. Based on known issues from Microsoft, the accuracy of hand detection is worse for the left than for the right hand [3]. Kinect has his own hand states already:

- Open
- Closed
- Lasso
- Unknown
- NotTracked

We used some of those gestures in this project, but some of them we must to create manually:

1) Pointing by finger (linear equation)

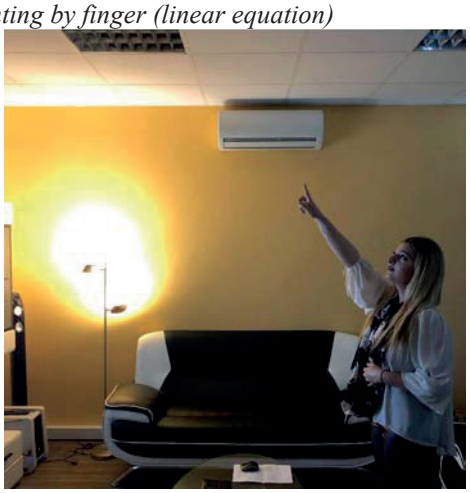

Figure 3. Linear equation by gesture usage

The line vector can be obtained by the difference of two points lying on it. We created a function where the input parameters are the x, y, the elbow point and the right hand point. Straight line is limited to the relative dimensions of the room, so a line is created.

In this manually created gesture, we use two points. Hand tip and elbow. We use linear equation to accomplish this gesture. A common form of a linear equation in the two variables x and y is

$$y = mx + b, \quad (1)$$

where m and b designate constants (parameters).

2) Joind hands (tips)

Second manually created gesture is joint hands tips. We will use two points, right and left hand finger tips.

Figure 4. Jointed hands gesture usage

IV. SMART ROOM SETUP

At the Faculty of Electrical Engineering and Informatics in Bratislava, at the Telecommunications department, there is one room equipped with new technology, and they have been used for the purpose of this project.

This smart room contains TV, air conditioning, lamp, LED, light, ventilator, curtains and other things that we can manage. Also, there are Kinects, which we got to use and which helped us to make this project real.

Figure 5. Current smart room at Faculty of Electrical Engineering and Information

Determining the specific points on which the devices we want to control are located can be written manually in the code, which is the harder way to calculate the distance X, Y, Z from the Kinect sensor. Therefore, the ideal solution was to use Kinect for this purpose. To determine the points on which the device is located, we chose a simple gesture - a closed hand. This gesture is already generate by SDK2.

Figure 6. Closed hand gesture usage

Firstly, it's necessary to be sure that our Kinect can catch all these devices that we want to use for our smart room. So, we need to choose the best angle (the best option is one of corners in room). Only then, we can start with room setting.

We need to come as closer as we can to the device we want to setting. Then, we will close our hand and say word which we are generated in advice in our XML file.

If we want to set lamp, we need to close our hand and say "Set lamp".

If we want to set light, we need to close our hand and say "Set light".

If we want to set LED, we need to close our hand and say "Set LED".

Application will generate an XML file with all those coordinates, for each device we located and save it. The example shows code for generate XML file for Smart room setting up.

```
using System.Xml.Serialization;
using System.Collections.Generic;
using System.IO;

namespace
Microsoft.Samples.Kinect.BodyBasics
{
    internal class Pozicia
    {
        public string name;

        public static List<Pozicia> list =
new List<Pozicia>();

        public double X;
        public double Y;
        public double Z;


        public static void
Serialization(string fileName)
        {
            fileName =
"C:\\Users\\User\\Desktop\\Stefany\\GestVo
ice\\BodyBasics-WPFDP2\\Database.xml";
            XmlSerializer serializer = new
XmlSerializer(typeof(List<Pozicia>));

            using (var stream =
File.OpenWrite(fileName))
            {

serializer.Serialize(stream, list);
            }
        }
    }
}
```

For lamp, LED and light control we use one gesture combined with voice (pointing by finger). We are sending command by our point finger. Linear equation helps us to find right straight line (using two points).

For setting darkness (which means to switch off lights), when you want to leave smart room, we will use joint hands gesture.

A. Gesture and speech recognition

For simpler and more reliable usage of gestures in smart room, I work on this project together with my colleague Jakub, where his task was to connect gestures with sound.

Kinect can recognize voice commands too. When we use gestures with voice commands, it's more precisely.

When installing the Microsoft Kinect SDK, the components needed for speech recognition are automatically installed. These include DirectX Media Object (DMO) voice capture and Speech Recognition API (SAPI).

We have an XML file with basics voice commands which we use in our application, for example: set light, set lamp, set darkness…

If it seems that the voice commands are quite similar, we can adjust voice sensitivity in our application.

V. CONCLUSION

In this project, we have proved that it is very easy to use Kinect for the need of managing Smart room. Kinect is already widely used in other spheres of life, it facilitates everyday life, helps us in medicine, informatics and in other spheres of life. I would like to see the use of Kinect expanded and for the purpose of learning at other faculties of our faculty.

Sending a command through gestures is very simple, but it's even simpler when using the voice commands with gestures. We

tried to make the gestures and voice commands more natural and to imitate the normal behavior of a man. This way of managing the Smart room could be managed with other Smart Room devices.

ACKNOWLEDGMENT

The research described in the paper was financially supported by the H2020 project NEWTON, No. 688503 and VEGA project INOMET, No. 1/0800/16.

REFERENCES

[1] ABHIJIT, JANA. 2012. Kinect for Windows SDK. In : Birmingham, UK : Packt Publishing Ltd., 2012. ISBN 978-1-84969-238-0.

[2] JointType Enumeration. [Online] [Dátum: 20. Apríl 2016.] <https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx

[3] MSDN.MICROSOFT. 2015 Skeletal Tracking. Microsoft developer network. [Online] [Cit. 2015-01-30] Online: <https://msdn.microsoft.com/en- us/library/hh973074.aspx>.