Proceedings of the 12th INDIACom; INDIACom-2018
5th 2018 International Conference on "Computing for Sustainable Global Development", 16th – 18th March, 2018
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA)

Recognizing Complex Gestures via Natural Interfaces

Houda Ben Abdelmoumen
ETSI Telecomunicación, Universidad de Málaga
Málaga, España
Email Id: houda_benab@hotmail.com

Dr. Javier Poncela
ETSI Telecomunicación, Universidad de Málaga
Málaga, España
Email Id: jponcela@uma.es

Abstract – Natural interfaces have revolutionized the way we interact with computers. They have provided in many fields a comfortable and efficient mechanism that requires no computer knowledge, nor artificial controlling devices, but allow as to interoperate via natural gestures. Diverse fields such as entertainment, remote control, medicine, fitness exercise are finding improvements with the introduction of this technology. However, most of these sensorial interfaces only provide support for basic gestures. In this work we show how it is possible to construct your own complex gestures using the underlying capabilities of these sensor devices.

Keywords – Gesture recognition, complex gestures, natural interface, Kinect sensor.

I. INTRODUCTION

The popularity of natural interfaces has experienced an overwhelming growth due to the fact that the best human-machine interaction is becoming more and more imperative through an adequate interface that provides both comfort and efficiency.

A natural user interface provides a way to interact with a system or application much more instinctive, in which the user uses body movements and voice commands to interact with the system, instead of peripherals such as the mouse or the keyboard. Thanks to these characteristics, people without computer knowledge are able to handle all kinds of applications.

The Kinect sensor is a device that allows you to track the user and the position of its body while providing other functionalities. Kinect is able to recognize the position and movement of certain points of reference in the body, but the set of gestures that recognizes is limited.

The objective of this work is to describe how it is possible to recognize complex gestures made by the user making use at all times of a natural user interface. For the development we have used the Kinect Development Kit for Windows [1] provided by Microsoft on .NET technology, using Windows Presentation Foundation (WPF) [2], and Visual Studio.

This paper is structured as follows. Section II describes the Kinect sensor. Section III shows how complex gestures can be designed. Section IV describes the code implementation for the recognition of gestures.

II. NATURAL INTERFACES

Several companies, among them Sony and Nintendo, have developed devices that provide natural interfaces, such as PlayStation Move motion controller [3] or Nintendo Wii controller [4]. They are commercial and closed devices that do not legally allow the development of applications or tools that make use of these devices so that they can be of some type of utility for the scientific community.

The Kinect sensor [5], developed by Microsoft, was born as a new controller for the Xbox 360 [6] video game console under the slogan "You are the controller". It was created by Alex Kipman and its innovation lies in that it allows you to operate the video game console without the need for a traditional controller or remote control. You can control it only with your body movements and voice commands.

The technology used in Kinect is based on software created by Rare [7], a company belonging to Microsoft, and on a camera capable of measuring distances created by PrimeSense [8], a company specialized in the creation of natural user interfaces. Kinect consists of three different sensors that work together to create the experience of a natural user interface. The Kinect sensor today has become a very useful tool in multiple fields such as advertising, entertainment and also in health. Its creators and teams of developers have created applications that are of great value when it comes to treating patients. Thus, the sensor has positioned itself as an ideal device to join rehabilitation and play in research and specialized centers.

A. Kinect Sensor

The Kinect sensor is a horizontal bar about 23 cm (9 inches) connected to a small circular base with a ball-and-socket joint shaft, and it is designed to be placed longitudinally above or below the display screen.

The Microsoft Kinect sensor allows you to capture audio data, depth (depth) and color RGB images. Processing these last

two data flows, depth and color, is able to generate more complex data about the users that are located in front of the sensor:

- Skeleton: The skeleton represents a user identifying the position of a set of joints and joints between them (bones). See Fig. 1. Its coordinates are stored as 3D.
- FaceTracking: The facial mask, identifying and positioning several important points of the face, allowing to follow tilt and rotation of the head.
- Interactions: information about user interaction with the hands, including events that notify if a user opens or closes the hands.

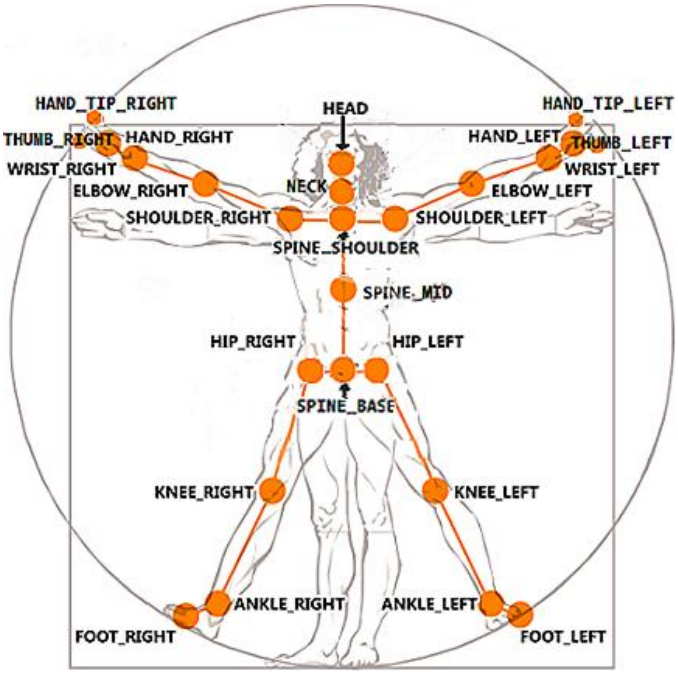

Fig. 1. Joint types. [9]

III. DESIGN OF GESTURES

A gesture is made up of partial gestures executed in a certain order. Each partial gesture is a specific movement that, when combined with other partial gestures, constitutes the whole gesture.

A partial gesture or also called segment is a concrete position that the body has at a given moment. It is defined as a set of conditions based on relative positions of the body parts, which are the euclidean coordinates (x, y, z) of the joints that the Kinect sensor provides us with.

For this work, we have opted for the implementation of stretching gestures that must be done before proceeding to exercise:

1. Move both hands up and down (Hands_UpDown).
2. Move both hands from right to left (Hands_RightLeft).
3. Throw a ball with the right foot and with raised hands (Kick_Ball).

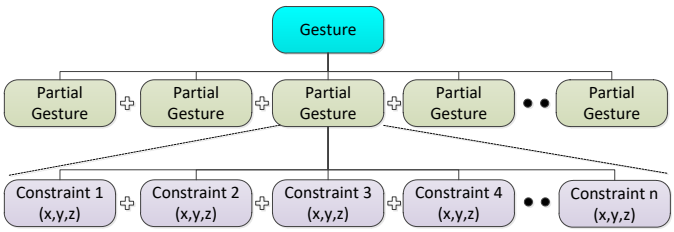

Fig. 2. Definition of gestures.

A. Gesture 'Move both hands up and down' (Hands_UpDown)

This movement consists of a vertical displacement of the two hands in the hip head range, the initial position of the two hands is above the head, and the final position below the waist.

This gesture will be composed of three partial gestures. The mathematical definition is shown in Table I:

G1a. The two hands are raised vertically upwards above the head.
Position criteria: The vertical position of both hands must be above the head.

G1b. The two hands extended forward and with slight opening towards the sides.
Position criteria: The vertical position of the two hands must be between the shoulder and the hip. The depth of the two hands has to be smaller (closer) than the depth of the elbow. The depth of the elbow must be less (closer) than that of the shoulder. The horizontal position of the right hand must be greater than that of the elbow to make sure it is stretched. The horizontal position of the left hand must be less than that of the elbow to ensure it is stretched.

G1c. Both hands are below the hip.
Position criteria: The vertical position of both hands must be below the hip.

TABLE I. MATHEMATICAL CONSTRAINTS FOR GESTURE HANDS_UPDOWN.

Partial Gesture	Constraints

G1a	`HandRight.Y > Head.Y`
	`HandLeft.Y > Head.Y`
	`HandRight.X > Head.X`
	`HandLeft.X < Head.X`

G1b	`HandRight.Y > HipRight.Y`
	`HandLeft.Y > HipLeft.Y`
	`HandRight.Y < ShoulderRight.Y`
	`HandLeft.Y < ShoulderLeft.Y`
	`HandRight.Z < ElbowRight.Z`
	`HandLeft.Z < ElbowLeft.Z`
	`ElbowRight.Z < ShoulderRight.Z`
	`ElbowLeft.Z < ShoulderLeft.Z`
	`HandRight.X > ElbowRight.X`
	`HandLeft.X < ElbowLeft.X`

G1c	`HandRight.Y < HipRight.Y` `HandLeft.Y < HipLeft.Y`

B. Gesture 'Move both hands from right to left' (Hands_RightLeft)

This movement consists of a horizontal movement of the two hands, the initial position of the two hands is above and to the right of the head and the end position is above and to the left of the head. This gesture will be composed of two partial gestures. The mathematical definition is shown in Table II:

G2a. The two hands raised up above the head and tilted to the right.
Position criteria: The vertical position of the hands must be above the head. The horizontal position of both hands has to be to the right of our head.

G2b. The two hands raised up above the head and tilted to the left.
Position criteria: The vertical position of the hands is above the head. The horizontal position of both hands has to be to the left of our head.

TABLE II. MATHEMATICAL CONSTRAINTS FOR GESTURE HANDS_RIGHTLEFT.

Partial Gesture	Constraints
G2a	HandRight.Y > Head.Y HandLeft.Y > Head.Y HandRight.X >Head.X HandLeft.X > Head.X
G2b	HandRight.Y > Head.Y HandLeft.Y > Head.Y HandRight.X < Head.X HandLeft.X < Head.X

C. Gesture 'Throw a ball with the right foot and with raised hands' (Kick_Ball)

This movement consists in having both hands raised upwards at all times, tilting the right foot backwards and then forward.
This gesture is composed of two partial gestures. The mathematical definition is shown in Table III:

G3a. The two hands are lifted vertically upwards and the right foot lifted off the ground and tilted backwards.
Position criteria: The vertical position of the two hands is above the head. The vertical position of the right foot is above that of the left. The depth of the right foot is greater (farther) than that of the left foot.

G3b. The two hands are lifted upwards so that the vertical position of the two is above the head, the right foot is raised above the left foot and inclined forward.
Position criteria: The vertical position of the two hands is above the head. The vertical position of the right foot is above the left one. The depth of the right foot is smaller (closer) than that of the left foot.

TABLE III. MATHEMATICAL CONSTRAINTS FOR GESTURE KICK_BALL.

Partial Gesture	Constraints
G3a	HandRight.Y > Head.Y HandLeft.Y > Head.Y HandRight.X > Head.X HandLeft.X < Head.X FootRight.Y > FootLeft.Y
	FootRight.Z > FootLeft.Z
G3b	HandRight.Y > Head.Y HandLeft.Y > Head.Y HandRight.X > Head.X HandLeft.X < Head.X FootRight.Y > FootLeft.Y FootRight.Z < FootLeft.Z

IV. GESTURE RECOGNITION

The data provided by the sensor are those related to the skeleton, and to each of the joints present in it. This information is organized as sets of values in the Skeleton and Joint classes, which provide the list of joints and their positions.
To access this information, we must register in the corresponding communication channel:

```
// Enable skeleton data flow
this.sensor.SkeletonStream.Enable ();

// Choose the tracking mode (standing or sitting)
this.sensor.SkeletonStream.TrackingMode =
   SkeletonTrackingMode.Default;

// Indicate if we want the detection in near or far
   mode
this.sensor.SkeletonStream.EnableTrackingInNearRange
   = true;

// Subscribe to the data channel
this.sensor.SkeletonFrameReady +=
   this.SensorSkeletonFrameReady;
```

By initializing all the components and subscribing to the data channels that are needed for recognition, the sensor provides the data of the skeletons of the users that are in its field of vision. Once the skeleton data is available, the first step is to check if the position of the joints corresponds to the defined gestures. This checkup is done using the `checkGestures` method of the `GestureManager.cs` class.
In this method, the list of gestures is scrolled to see which of them is being performed. This is implemented by creating an instance of the `checkGesture` method of the `Gesture.cs` class for each of the gestures. This method checks the compliance of the partial gestures that are part of the gesture by calling the `checkGesturePart` method for each of them. When one of these instances detects the beginning of a gesture, it will launch an event, and, when the gesture ends, it will launch an event of completion.
To check the first segment of the gesture, we first check the status of the joints that are part of that segment, whether they are being followed or not. If they are not, this method returns false, since we can say directly that the segment detection has failed. If the joints are well detected, the compliance of the conditions of the partial gesture is checked by ascertaining the position of the joints. The execution time is limited by a timer.
The diagram in Fig. 3 explains the full process of detecting a gesture. The sequence diagram corresponding to the complete

detection process is shown in Fig. 4. The class diagram for the recognition module is shown in Fig. 5.

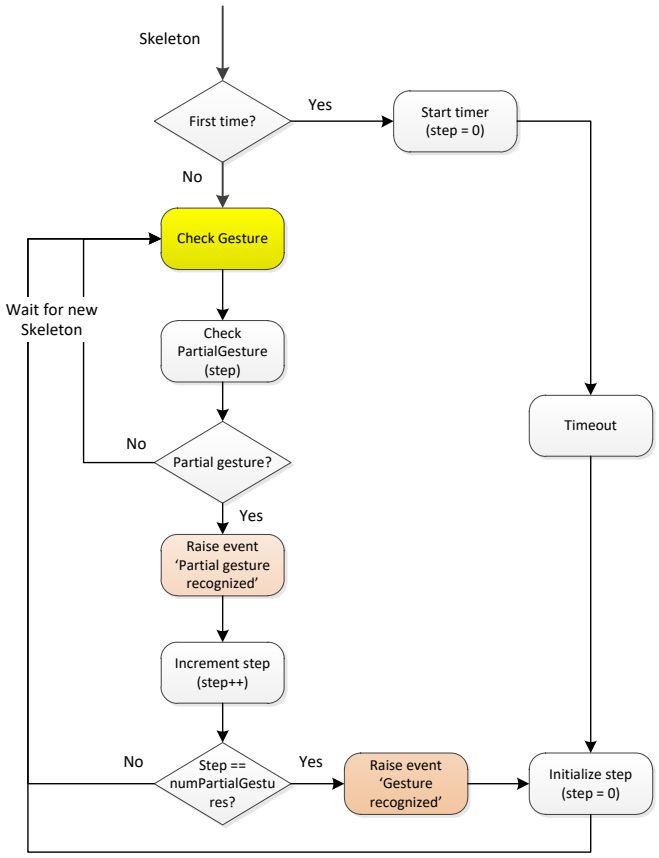

Fig. 3. Process of gesture detection.

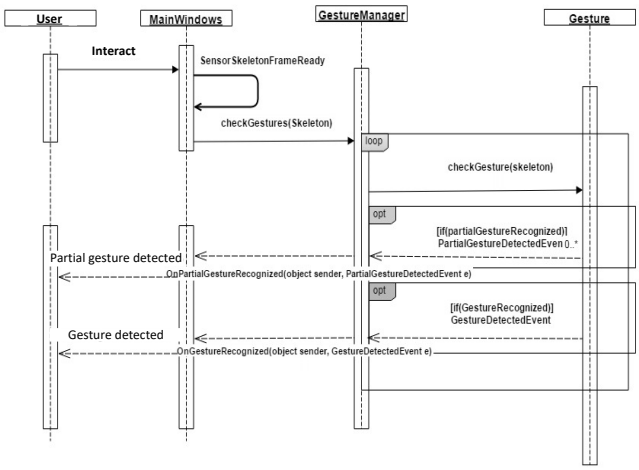

Fig. 4. Process of gesture detection.

V. CONCLUSION AND FUTURE SCOPE

With the advent of natural interfaces, the applications of this technology have multiplied. However, these devices usually only offer recognition of basic gestures. We have shown in this paper how it is possible to create personalized complex gestures based on the recognizing capabilities of these sensors. The paper has described the mathematical conditions that define three complex gestures (move hands up and down, move hands left and right, and kick a ball), and has shown how they can be implemented.

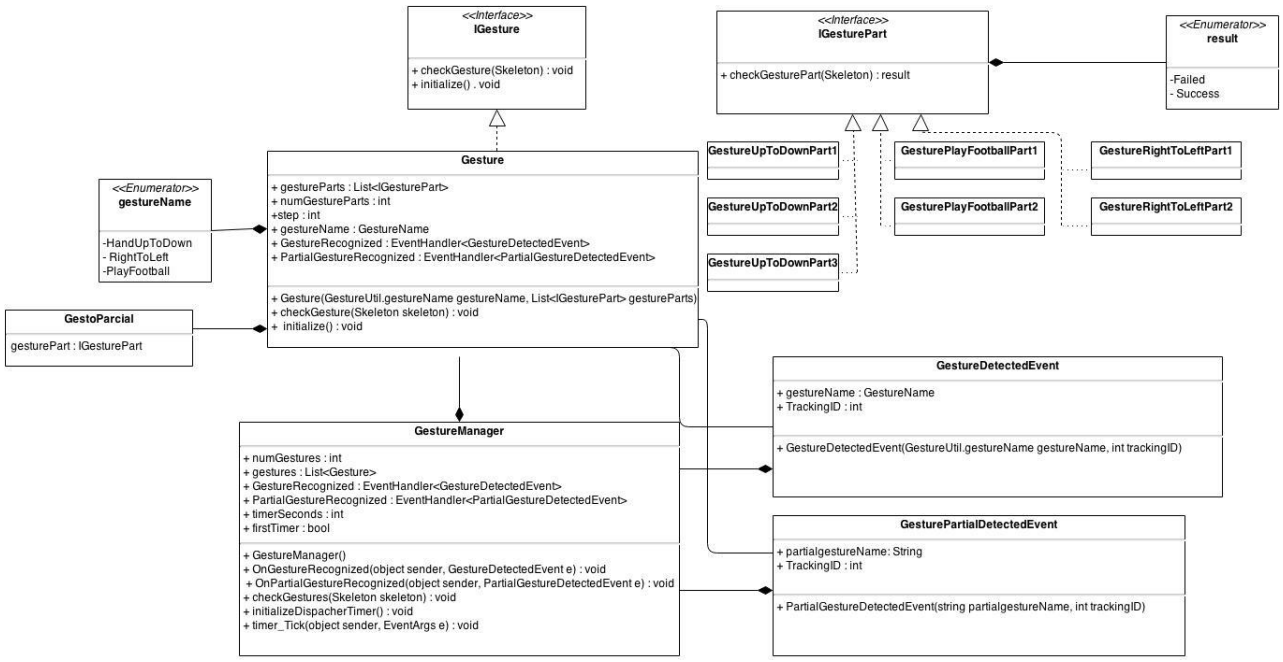

Fig. 5. Class diagram.

ACKNOWLEDGEMENT

This work has been partially funded by the 'Plan Propio de Investigación y Transferencia' of the Universidad de Málaga.

REFERENCES

[1] Kinect Development Kit for Windows, https://developer.microsoft.com/es-es/windows/kinect/develop, Microsoft, 2014.
[2] Windows Presentation Foundation, https://msdn.microsoft.com/es-es/library/ms754130(v=vs.110).aspx, Microsoft, 2015.
[3] PlayStation Move, https://www.playstation.com/en-us/explore/accessories/vr-accessories/playstation-move/, Sony, 2015.
[4] Nintendo Wii controller, https://www.nintendo.com/wii/, Nintendo, 2013.
[5] Kinect sensor, https://www.xbox.com/es-ES/xbox-one/accessories/kinect, Microsoft, 2013.
[6] Xbox 360, https://www.xbox.com/, Microsoft, 2013.
[7] Rare, https://www.rare.co.uk/, 2014.
[8] PrimeSense, https://www.primesense.com/, 2013.
[9] JointType Enumeration, https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx, Microsoft, 2015.