MULTITOUCH KINECT FOR WINDOWS7, BASED ON TUIO – OSC PROTOCOL

Dr. Rosario Salvatore Barbagallo

Computer Engineering Department, University of Catania
Catania, Italy

barbagallosaro@virgilio.it

Abstract

This paper describes the implementation of an input provider based on the use of the SDK Beta2 Microsoft Kinect and its use hand motion detection. The input provider sends the coordinates of the hands to a windows driver using a clustering hand detection algorithm, which analyzes data in depth at a distance between 800 mm and 1100 mm. The driver sends the data to a virtual Human Interface Driver which translates the "touch messages" into suitable commands for an application.

I. INTRODUCTION

Systems for gesture recognition are useful in all areas where interaction with an application without physical contact with the system is required. This paper describes an approach based on the use of the Kinect XBOX 360 sensor, developed by Microsoft for game applications and examines how to improve its capabilities to gesture recognition applications for general purpose.

Development of the application has required the following components:

- Sensor Kinect XBOX 360;
- Dual-core 2.66-GHz;
- S.O. Windows 7;
- Microsoft DirectX® SDK - June 2010;
- NET Framework 4.0;
- Runtime for Microsoft DirectX® 9;
- Microsoft Speech Platform Runtime, v. 10.2;
- Microsoft Speech Platform – Software Development kit, v. 10.2;
- Kinect for Windows Language Pack, v. 0.9.

Kinect features

Kinect has an RGB camera and a dual infrared depth sensors: a projector and an infrared sensitive camera on the same band. The RGB camera has a resolution of 640 × 480 pixels, while the infrared camera uses a matrix of 320 × 240 pixels.

Fig. 1: Kinect Sensor

Moreover Kinect has a microphones array used by the system for calibration of the environment where the user is located. Through the analysis of sound and the reflections on the walls of the room background noise and sounds are cancelled and the system can correctly recognize voice commands. Kinect is also motorized along the vertical axis.

Outline of deployed modules

Kinect SDK Beta 2 allows the interaction with Kinect, it receives RGB, audio and depth data.

Hand Detection can detect hands and fingers using depth data with a k-means clustering algorithm. The library Candescent has been used. [1]

The *input provider* is a server that is based on C # Tangible User Interface Objects (TUIO) and sending Open Sound Control (OSC) message.

The *driver*, the *Human interface Driver (HID) and Touch Message* are parts of the module Multi-Touch Vista.

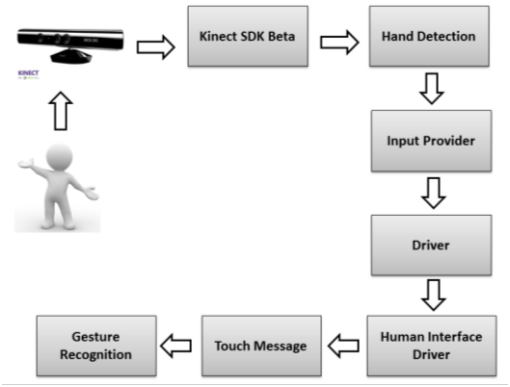

Fig. 2: Schema of modules deployed

II. MULTITOUCH KINECT

The input provider has been implemented using C# language. After the elaboration of the SDK Beta2 Microsoft Kinect [2] and the algorithm of HandDetection, the input provider sends the coordinates to the windows driver (the algorithm uses clustering HandDetection / K-means, that analyzes data depth at a distance of 800 mm and 1100 mm).

The driver sends the data to a virtual Human Interface Driver (HID) that translates "touch messages" for WindowsA TUIO input provider has been implemented and the processed data are "OSC" sent on UDP. The driver, the service and the TUIO client are parts of *Multi Touch Vista.* [3]

The TUIO Server is necessary for communication with the driver that uses the API TUIO protocol to send the translation-touch. The user can remotely control a PC without having to physically connect the Kinect.

OSC fits the data for Kinect, Touch Monitors, etc.

The structure of OSC "message" is made of an: id and a cursor (that contains the coordinates x, y).

The server has two endpoints: "localhost" and port "3333". The main methods of the server are: *AddTuioCursor (id, point), UpdateTuioCursor (id, point), DeleteTuioCursor (id).* The id is an integer that is connected to the device sending the input, the id 0 indicates a finger of right hand and the id 1 identifies a finger of left hand.

A raised finger sends an input.

Fig. 3: Send touch input

With the hand closed, the cursor is displayed on the screen without transmitting touch input

Fig. 4: No touch input

The method "AddTuio" has been used to send the coordinates when the finger is detected if the movement continues, proceed with the method "UpdateTuio", when the hand is closed has been used the method "DeleteTuio". To improve the user experience, a cursor has been implemented (a red dot) that displays the position of the hand. It is necessary to keep the closed hand (fist) because when the user lifts a finger, the system sends an input signal (touch). The program displays one or two points, according to the number of hands "detected" by the algorithm.

Win32Interop.dll and *Win32.dll* has been used to display the cursors on the desktop. The input provider detects the second hand and it sends the data without restarting the communication. The driver and the client are part of the framework *Multi-Touch Vista*, that handles input from various devices (touchlib, TUIO etc.). The user must start: first the ServiceConsole, second the DriverConsole and finally the input provider. This system interacts with WPF applications that have TouchPanel, TouchPoint – Stroke and manage multi-touch events. The TouchDevice.Id allows us to distinguish the various sources.

Important Note: Windows 7 natively supports touch events (so the user can interact with many features including, for example, move up or down a web page, manage windows media player etc.) but in some cases we need to manually enable this feature from the control panel. The touchpad must be disabled because the operating system gives the highest priority ignoring the touch input.

III. VOICE CONTROL

It has been integrated a system of events linked to voice commands with multiple features outlined below.

To implement voice commands it is necessary:

– to determine which recognizer is installed in the machine, in this case, the only supported by the SDK Kinect is *SR_MS_en-US_Kinect_10.0*; [4]

– to istantiate an SpeechRecognitionEngine, the creation of a grammar containing the words to recognize;

– the implementation of an event handler that handles SpeechRecognized object containing the word / command that has the highest confidence.

The following commands have been implemented:

1. *Kinect one:* the default configuration, the left and right hand send an input, must lift a single finger of the hand.
2. *Kinect two:* the system detects one hand if the user raise two fingers, the system sends two inputs.
3. *Kinect stop:* disables the tracking of the hands.
4. *No voice:* disables the voice commands, to reactivate it must keep both hands visible and the left hand with three fingers raised to more than 5 frames.
5. *Kinect play:* enables the tracking of the hands.
6. *Open Paint \ Close paint:* opens \ closes the Paint program Windows 7
7. *Open photo \ Close photo:* opens \ closes a sample application for image management.
8. *Open paint special\ Close paint special:* opens \ closes a sample application for the paint.
9. *Web Night \ Night Close:* opens \ closes a web page
10. *Republic web \ Close republic:* opens\ closes a web page

To avoid the activation of unwanted commands, the user must keep both hands closed while he says the command. The camera's control does not require this condition, but only the voice command. All controls to be tested must have a confidence level greater than 0.93

IV. DEPTH DATA ANALYSIS

For each event the system gets a PlanarImage that contains a byte [5] array that contains the distance of each pixel (distance values mm.). Because there are 2 bytes pixel (16 bits) that represents the distance, the application uses the operators BitShift to get the distance of a particular pixel.

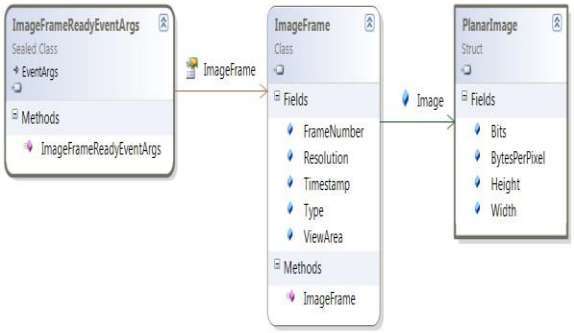

Fig. 5: Depth ImageFrame

The array contains the distance (mm) for each image point, and this varies from a minimum of 800/850 mm to a maximum of about 4000 mm. If the Kinect fails to recover the depth, the distance returned is 0. The array size is equal to the product image size (eg 320x240) dot 2, and it is organized by rows starting from the upper left corner.

V. SKELETON DATA ANALYSIS

For each SkeletonFrameReady, the system uses a LINQ query to get the desired skeleton.

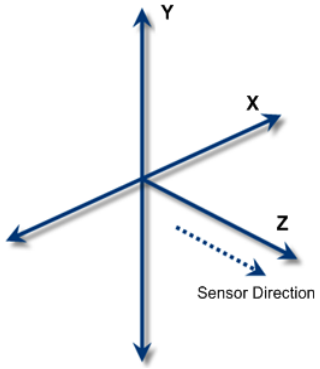

Fig. 6: Skeleton reference system

In fig. 6: X – horizontal position measured in meters along the x axis from Kinect; Y –vertical position measured in meters along the Y axis from Kinect; Z – distance in meters measured from Kinect.

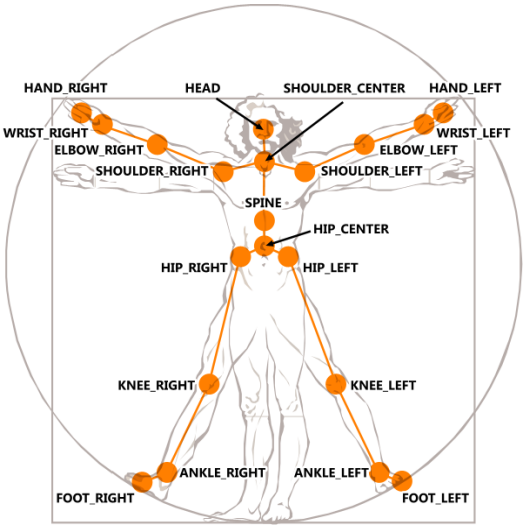

Fig. 7: Kinect Vitruvian Man

Each skeleton is a collection of 20 Joints that represent 20 points of the human body with coordinates x, y, z. Each joint has a state "traced" "not traced" "hypothesized". The system detects only 2 skeletons "active" and 4 passive. MultiTouchKinect works with the skeleton, the only difference being that instead of using the fingers of hands, the user uses the hand directly.

VI. TUIO PROTOCOL

The TUIO protocol is encoded using the format Open Sound Control, which provides an effective method of encoding binary data for transmission. The default transport method for the TUIO protocol is the encapsulation of binary data in OSC message within UDP packets sent to port 3333. [6, 7]

Implementation Details

The TUIO protocol defines two main classes of messages: SET messages and ALIVE messages. SET messages are used to communicate information about an object's state such as position, orientation, and other recognized states. ALIVE messages indicate the current set of objects present on the surface using a list of unique Session IDs.

To avoid possible errors evolving out of packet loss, no explicit ADD or REMOVE messages are included in the TUIO protocol. The receiver deduces object lifetimes by examining the difference between sequential ALIVE messages. In addition to SET and ALIVE messages, FSEQ messages are defined to uniquely tag each update step with a unique frame sequence ID. An optional SOURCE message identifies the TUIO source in order to allow source multiplexing on the client side.

Efficiency & Reliability

In order to provide low latency communication the implementation of the TUIO protocol uses UDP transport. When using UDP the possibility exists that some packets will be lost. For efficiency reasons set messages are packed into a bundle to completely use the space provided by a UDP packet. Each bundle also includes a redundant alive. Each packet is marked with a frame sequence ID (fseq) message: an incrementing number that is the same for all packets containing data acquired at the same time. This allows the client to maintain consistency by identifying and dropping out-of-order packets.

VII. APPLICATION EXAMPLES

Some applications have been developed through the approach described.

Touch Paint Application

This is an application like Windows Paint. It handles some specific events i.e. TouchDown, TouchUp, TouchMove, and TouchLeave TouchEnter and allows the user to draw a colored line on the screen.

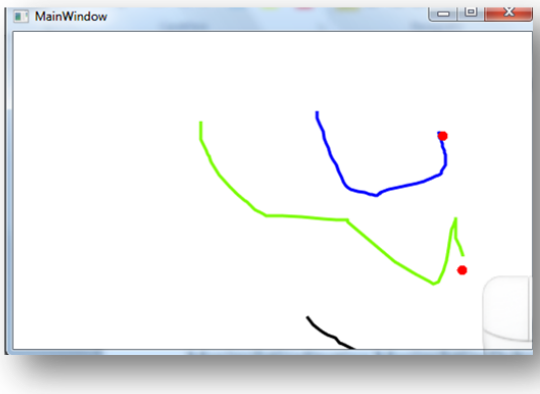

Fig. 8: Touch Paint

Photo Touch Application

Application that interacts on two images. It handles events ManipulationStarting ManipulationDelta e ManipulationInertia. The user can rotate and zoom the photo.

Fig.9:Photo touch

VIII. CONCLUSIONS

This input providers allows us to use any application (past and future) that works on touch devices.This paper has described the system and two start touch applications that demonstrates how to use the system implemented. However, the developer can use this type of software to do also different applications like for example home automation applications.

The use of the TUIO protocol guarantees not only low latency but also a certain robustness because it includes redundant information to correct any lost data packets.

The possibility of transmitting data over the network allows us to use the kinect without a direct connection to the PC where the final application is installed. This way the control can be distributed over multiple machines. The Kinect has made faster processing, and it has combined the use of gestures and voice control that extends the usage scenario.

A demo of the work done is available at http://www.youtube.com/watch?v=TgNj06Tsjbs

REFERENCES

[1] http://candescentnui.codeplex.com/
[2] http://www.microsoft.com/en-s/kinectforwindows/
[3] http://multitouchvista.codeplex.com/
[4] http://msdn.microsoft.com/
[5] http://it.wikipedia.org/
[6] http://www.tuio.org/
[7] http://opensoundcontrol.org/