International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Interactive 3D Touch and Voice Control System

Prashasti Sar[1], Mariya Ali[2], Vaidehi Pawar[3]

[1, 2, 3]B.E. Electronics (Mumbai University)

Abstract: ***In this paper, we design an interactive 3D touch and voice control system. The system can use any large surface as a touch/gesture and voice controlled pad. People investing in large screen devices can use this technology, making their lives more convenient. Thus, we aim to achieve the following goals: To create a library of gestures which will be recognized by the system. To train the system to learn new gestures at runtime upon activation of the recording function. To identify different layouts of applications at run time. To be able to access functions at OS level using our code.***

Keywords: Kinect, interactive UI, touch/gesture, voice control

1. Introduction

An Interactive screen can be obtained by projecting a mobile/ laptop screen onto any flat surface and interacting with that surface. This implies that said surface will behave as a new screen on which operations can be performed. This has been done by using depth cameras and Kinect sensors in the past. This paper is also achieving the UI with the help of Kinect sensor. It is achieved by recognizing gestures and patterns that a user makes to interact with the screen and giving them a definite meaning.

For example, when we see the gesture of a hand waving in our direction, due to previously acquired learning, our brain deciphers it as a hand wave and so as a response we wave back. This learning of gestures by the software enables us to use any surface as our interactive screen.

Section 2 consists of a brief description of related work and literature survey done previously. Section 3 talks about the hardware and software components used. Section 4 throws light on the results and a few case studies undertaken for this paper. Section 5 draws the conclusion.

1.1 Design Considerations

Gesture recognition comes with its own challenges; the most important one being how efficiently the gesture is recognized by the system. Since the entire system is real time based, suitable algorithms have to be used to obtain high degree of precision, accuracy and efficiency. Presently, gaming softwares which use gesture recognition offer an efficiency of about 97%. Obtaining these values in a system show the merit of the system. The system is meant to be portable and easy to use. Hence the number of hardware components must be kept in check. For the prototype design, only three devices are used and if actually implemented by fulfilling the scope of the project, then the number of hardware devices can be reduced to one.

2. Related Work

2.1 Hand Gesture Recognition Using Kinect

Hand gesturerecognition (HGR) is an important research topic because some situations require silent communication with sign languages. Computational HGR systems assist silent communication, and help people learn a sign language. In this article, the system was able to detect the presence of gestures, identify fingers, and recognize the meanings of nine gestures in a pre-defined Popular Gesture scenario. The accuracy of the HGR system was from 84 to 99% with single-hand gestures, and from 90% to 100% if both hands performed the same gesture at the same time.

2.2 Gesture recognition based on arm tracking for human-robot interaction

In this paper, the proposed system utilizes upper body part tracking in a 9-dimensional configuration space and two Multi-Layer Perceptron/Radial Basis Function (MLP/RBF) neural network classifiers, one for each arm. Classification is achieved by buffering the trajectory of each arm and feeding it to the MLP Neural Network which is trained to recognize between five gesturing states.It was observed that the successful recognition ratio did not drop below 86% while the false negative percentage remained in low levels as well.

2.3 A Real-Time Hand Gesture Recognition Method

In this paper, the authors propose a robust real-time hand gesture recognition method. In their method, firstly, a specific gesture is required to trigger the hand detection followed by tracking; then hand is segmented using motion and color cues; finally, in order to break the limitation of aspect ratio encountered in most of learning based hand gesture methods, the scale-space feature detection is integrated into gesture recognition. Applying this method to navigation of image browsing, experimental results showed that the method achieved satisfactory performance.

2.4 A Hand Gesture Recognition System Based on Difference Image Entropy

In this paper, the authors propose a real-time hand gesture recognition system based on the difference image entropy obtained using a stereo camera. In existing systems, hand detection has been primarily conducted in a constrained environment. However, in this system, the authors implement a recognition system for incoming hand images in real-time. In the detection step, we implement a depth map using the SAD method based on right-left images acquired using a stereo camera. This system perceives the foreground

object and performs hand detection. The experiment results show that the proposed method has an average recognition rate of 85%.

2.5 Hand Gesture Recognition with Depth Images: A Review

This paper presents a literature review on the use of depth for hand tracking and gesture recognition. The survey examines 37 papers describing depthbased gesture recognition systems in terms of:

(1) The hand localization and gesture classification methods developed and used, (2) The applications where gesture recognition has been tested (3) The effects of the low-cost Kinect and OpenNI software libraries on gesture recognition research. The papers that use the Kinect and the OpenNI libraries for hand tracking tend to focus more on applications than on localization and classification methods, and show that the OpenNI hand tracking method is good enough for the applications tested thus far. Though five different types of depth sensors were used, the Kinect was by far the most popular. The Kinect also has available hand-tracking software libraries that were used by 8 of the papers, and the papers that used the Kinect tended to focus more on applications than on localization and classification techniques.

3. Components of the Project

3.1Block Diagram

Following are the main components of this project:
1) Kinect Sensor
2) Laptop/PC
3) Projector

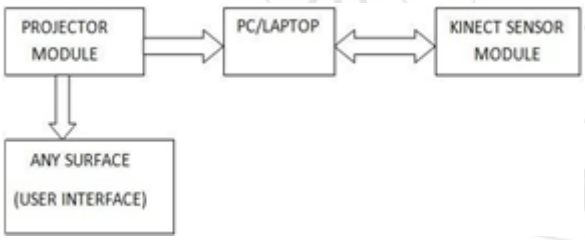

Figure 1: Block diagram of the system components

3.2 Component Description

Kinect Sensor

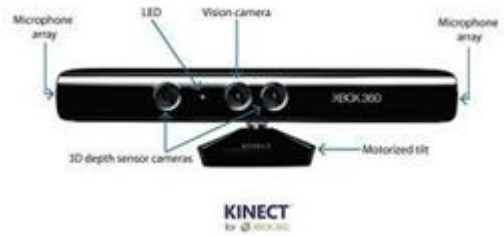

Figure 2: Kinect Sensor

Kinect is a line of motion sensing input devices by Microsoft for Xbox 360 and Xbox One video game consoles and Windows PCs. The sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an "RGB camera, depth sensor and multi-array microphone running proprietary software", which provide full-body 3D motion capture, facial recognition and voice recognition capabilities.

Laptop/PC:

The laptop has basically two functions:
- To process the image captured by the Kinect and run the application to perform the gesture recognition.
- Since a word processor is being controlled by these gestures, which is a laptop application, functions on the laptop OS level have to be accessed through the code.

Projector:
A projector simply projects images onto a wall or projection screen. Projectors are typically used to increase the image size for viewing by audiences.

3.3Software Development Kits Used

In order to be able to recognize a person and its movements the Software development kit(SDK) needs to support tracking of the user and the user movements.

Freenect:
Libfreenect is a user space driver for the Microsoft Kinect. It runs on Linux, OSX, and Windows and supports
- RGB and Depth Images
- Motors
- Accelerometer
- LED
- Audio

OpenCV:
OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 20 thousand people of user community and estimated number of download exceeding million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

Visual Studio:
Microsoft Visual Studio is an integrated development environment(IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web applications and services. Visual Studio uses Microsoft software development platforms such as Windows API Windows Forms, and Windows Presentation Foundation. It can produce both native code and managed code.

4. Design and Implementation

This section presents different ideas for the design of cursor control, gestures and how the gestures are recognized. The different ideas are discussed, and an idea for the cursor control, the gestures and the recognizing of gestures is chosen. It must be noted that the interface is developed with the right hand as the user's dominant hand. The reason for omitting the choice of the left hand as the dominant hand is, because it is a trivial implementation and not relevant in this study.

4.1 Cursor

In order to interact with a browser a similar interaction form as a cursor is required. The basic idea is to use the most commonly used input device, which is a mouse, as inspiration, where the movement of users' hand is mapped directly to the position of the cursor. If the user moves his hand to the left, the cursor should move to the left side of the screen. Xlib library functions were used for implementation of the same.

The libraries below are used for mouse manipulation
from Xlib import X,
display import Xlib.XK
import Xlib.error
import Xlib.ext.xtest

Event types:

d = display.Display() #Display reference for Xlib manipulation
def move_mouse(x,y): #Moves the mouse to (x,y). x and y are ints
s = d.screen()

def click_down(button): #Simulates a down click. Button is an int
Xlib.ext.xtest.fake_input(d,X.ButtonPress, button)
d.sync()

def click_up(button): #Simulates a up click. Button is an int
Xlib.ext.xtest.fake_input(d,X.ButtonRelease, button)
d.sync()

4.2 Gestures

The gestures required to perform various functions on a word processor can be done in two ways, using the fingers or the arm. The different ideas to make these gestures have therefore been divided into two categories: finger gestures and arm gestures.

Finger gestures
The idea of finger gestures is to use the fingers to make the gestures. An example is by showing one finger the functionality of a mouse click is triggered, and by showing two fingers the functionality of a double click on a mouse is triggered. In order to perform finger gestures the functionality of counting the number of fingers that are shown is required.

To be able to count the fingers, the available output from the Kinect and the OpenCV SDK has been studied. The available output from the OpenCV SDK is the depth image and the RGB-image, and based on these, two possible ideas have been developed and tested. One idea is based on the depth image, and the other idea is based on the use of skin-detection on the RGB-image from the Kinect.

Depth image
The depth image from the Kinect provides the data needed to count the number of fingers that are shown. The depth image from the Kinect is shown in Figure 3.1, and as seen the fingers can be seen as outgoing tops from the palm.

Figure 3: Depth image from Libfreenect

The number of fingers can then be counted using the Convex Hull algorithm. The Convex Hull algorithm can be illustrated by having a board with nails in, where an elastic band is stretched around the nails to form an outer border. The Convex hull algorithm is capable of calculating the number of edges that are needed to form the border. This number can then be converted to the number of fingers shown, because each shown finger requires a new edge as seen in Figure 3.2.

The advantage of this idea is that the hand can be separated from background because of functionality provided by the Kinect and the OpenCV SDK. The disadvantage of this approach is the quality of the depth image that the OpenCV SDK delivers. The quality of the depth image is only sufficient in our implementation, when the user holds his hand within 80 cm from the Kinect. After the limit of 80 cm the depth image becomes blurry, and this increases the difficulty of detecting and separating the fingers from each other. Figure 3.3 shows a screenshot of when holding the hand at the limit of 80 cm.

The OpenCV SDK can also only create a sufficient depth image, when the user is farther than 50cm away from the Kinect. If the user is closer than 50 cm, the quality of the depth image is reduced significantly and can therefore not be used. To sum up, this is only functional within the limit of 50 - 80 cm away from the Kinect, meaning a 30cm spread.

Figure 4: Depth image used with Convex Hull algorithm

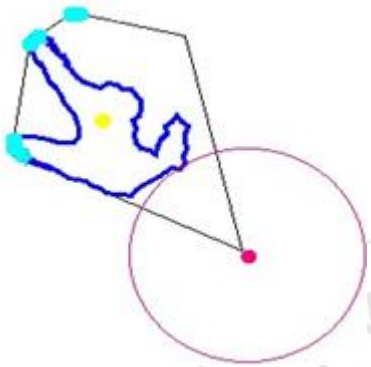

Figure 5: When holding the hand at the limit at 80cm

Skin Detection

The other idea is to use the RGB-image from the Kinect, because the quality of the RGB-image is not reduced as significant as the depth image after 80 cm. The problem of this approach is to separate the hand from the background, and to handle this, separation skin detection is applied. Skin detection used the fact that the human skin has a particular color spectrum that can be used to separate the skin and thereby the hand from the background.

Using skin detection, the shape of the hand can be detected, and this shape can be applied as data in the Convex Hull algorithm. This will, as with the depth image return the number of fingers that are shown.

As mentioned, the advantage of this approach is that the quality of the RGB-image from Kinect is sufficient at a larger distance than the depth image. The disadvantage with this approach is the fact that Kinect's perception of the color of the skin changes according to the light settings. The Kinect tries to adjust to the light settings in the room. By switching off the light, the color of objects can appear differently to the Kinect. Other objects than the actual skin can therefore be detected to be within the predefined color spectrum that the skin detection algorithm is searching for. The approach with skin detection is discarded because of the sensitivity concerning the light settings. If the system is calibrated, and the light is switched on, this can break the system.

5. Results and Case Studies

The gestures and voice control feature are implemented on a word processor in order to completely control its functioning. The case studies performed were to individually verify following features:

- The Gesture Recognition
- The Voice Control
- Slideshow Control

5.1 Static Gesture Recognition

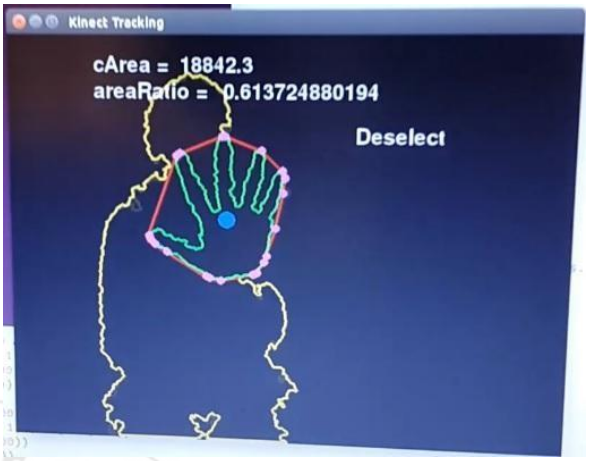

Figure 6: Gesture Recognition operation

The static gesture recognition is done using the Convex Hull algorithm. To identify a gesture efficiently, two parameters are calculated; *cArea* and *areaRatio*. *cArea* is the circumference of the hand of the user while a gesture is being made. Using these two parameters the gesture is recognized by the sensor up to desired accuracy. A library of 10 gestures has been pre-stored and is available at the users' disposal.

5.2 Voice Control

Voice Control is implemented to introduce a feature of voice to text conversion in order to write on to a notepad. As demonstrated in figure 4.1 and 4.2, the voice control is initialized and words spoken by the user are recognized by the sensor.

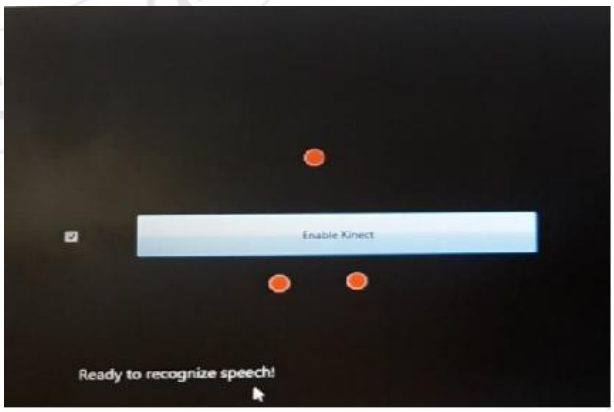

Figure 7: Initializing voice control

Figure 8: Voice Recognition implemented

These words are stored as text onto a notepad at runtime. User can access this notepad by performing a gesture which triggers the notepad to open up. The case study result is demonstrated in figure 4.3

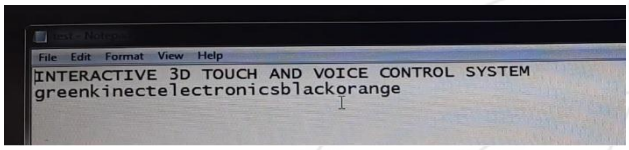

Figure 9: Recognized words updated on notepad

5.3 Slideshow Control

The slideshow function implements swipe gestures. As the user swipes right or left, the image on the screen or the slide displayed also moves to the next or previous one. This function can be implemented on a power point or an image viewer application.

Figure 10: Actual operation of slideshow

6. Conclusion

Use of Kinect sensor to supplant large screen devices is an innovation in itself. A training mode is added to the system for runtime recording and learning of gestures. This algorithm aims to be the most efficient method of learning by using artificial intelligence concepts. The accuracy of the system is at 89%. This system can be further improved by bettering the algorithm and generalizing it for other applications. However, we leave that as part of its future scope on which anyone can work on.

7. Acknowledgment

We take this opportunity to express our sincere gratitude to **Prof. SapnaPrabhu**, Department of Electronics Engineering for providing us with technical guidance and suggestions regarding the paper.

References

[1] A Real-Time Hand Gesture Recognition Method*yikai Fang*1, *Kongqiao Wang*2, *Jian* Cheng1 And Hanqing Lu1
[2] Real-Time Gesture Recognition Using 3d Depth Camera by Guan-Feng He, Sun-Kyung Kang, Won- Chang Song, Sung-Tae Jung
[3] Learning To Be A Depth Camera For Close Range Human Capture And Interaction By Sean Ryan Fanello, CemKeskin, Shahram Izadi, PushmeetKohli, David Kim, David Sweeney, Antonio Criminisi, Jamie Shotton, Sing Bing Kang, Tim Paek
[4] Hand Gesture Recognition With Depth Images: A Review By Jesus Suarez* And Robin R. Murphy, Member, Ieee
[5] Hand Gesture Recognition Using Kinect by Yi Li
[6] https://github.com/GestureDetection
[7] https://msdn.microsoft.com/enus/library/hh855360.aspx
[8] http://www.microsoft.com/enus/download/details .aspx?id=44561
[9] http://www.codeproject.com/Articles/716741/Implementing-Kinect-gestures
https://code.msdn.microsoft.com/windowsdesktop/Simple-Gesture-Processing-097c5527