Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 6, No. 2, May 2015), Pages: 99-107
www.jacr.iausari.ac.ir

Natural Interaction with Projected Screen on Surfaces using Infrared Depth Camera

Faraein Aeini✉
Dept. of Computer Eng., Sari Branch, Islamic Azad University, Mazandaran, Iran
f.aeini@iausari.ac.ir

Received: 2014/07/04; Accepted: 2014/08/03

Abstract

Technology advancement improves the way we interact with computer systems. Keyboard, mouse, and touchpad provide an easier and more natural way of interaction with a computer system. Nowadays, by introducing low cost and high performance depth cameras like Kinect and Xtion, new opportunities are available for creating even more natural interfaces. In this paper, a new model was demonstrated for natural interaction with a projected screen on a surface by detecting hand gestures and touch points using depth camera. Because lighting conditions, color, and background have little impact on the performance of systems based on depth camera, these systems can be integrated in a variety of applications in daily life. In this paper, a model was proposed for Microsoft Office as goal presentation. The main steps of the proposed model can be categorized as follows: hand detection, gesture detection, touch point detection, and control software. The proposed model had reasonably low cost with minimum configuration and calibration prerequisite and can replace the expensive and not-convenient-for-moving tablets.

Keywords: *Natural User Interfaces, Depth Camera, interactive surface.*

1. Introduction

Computer systems are already the integral part of almost every human activities such as robotics[1], tracking facial expressions[2], gesture recognition [3], and body monitoring in assistive environments[4, 5]. Recent works have shown that use of Kinect-style cameras can help people with cognitive impairments [4].

Over time, we have utilized different interface technologies to interact with a computer system like keyboard, mouse, and more recently touchpad and touch screen. The general idea is to create an easy to use and more natural way of communication with a computer system with minimum physical interaction. Most of the current efforts for developing new interfaces focus on vision- and speech-based interfaces.

The ability to use gestures is extremely important for selecting commands in natural interaction. In previous works, a number of gestural vocabularies have been proposed which are generally restricted to one or two fingers or can be difficult to learn. In addition, in some applications, click simulation is important. In this paper, the point of natural user interface was added to the proposed model.

Another limitation which has been observed in most of the previous methods is that they are based on application. Given that Microsoft Office has many common options such as minimize, maximize, close, save, and save as and an almost similar toolbar, a new model that was applicable for Microsoft Office software was proposed here.

Transform projected screen to interactive surface leads to interaction on a much larger screen and any surface of objects. Furthermore, it can replace expensive and not-convenient-for-moving tablets that are produced by famous companies like Samsung and Microsoft[6]. The proposed model simulates the interactive surface based on video projector, Kinect, and White screens.

Another innovation of the proposed model was projecting white screen into an interactive surface by detecting hand gestures and touch points using depth camera.

The present model was based on the movement of the human arm and hand or hand gestures in front of the depth camera.

Microsoft Kinect provides an inexpensive and easy way for real-time user interaction. The software driver released by Microsoft called Kinect software development kit (SDK) with application programming interfaces (API) provides access to raw sensor data streams as well as skeletal tracking. However, there is no hand specific data available for gesture recognition, although it includes information of the joints between hands and arms.

Recently, researchers are more interested in exploring computer vision-based analysis and interpretation of hand gestures after the release of Microsoft Kinect depth sensor. Aims of these studies have been to eliminate the limitation of optical sensors, sensitive to lighting conditions and cluttered backgrounds.

Yang et al. [7]proposed a gesture recognition system using depth information provided by Kinect and implemented it in a media player application. The hand trajectory was examined by a 3D feature vector and the gesture was recognized by a hidden Markov model (HMM). This system demonstrated the applicability of using Kinect for gesture recognition in a contactless user interface (UI). The system was able to recognize motion gestures like wave and move up/down, left/right, and forward/backward.

A method for tracking fingertips and centers of palms using Kinect was presented by Raheja et al.[8]. When fingers were extended, the accuracy of detecting fingertips was nearly 100% and that of palm centers was around 90%. However, this method did not attempt gesture recognition. He et al. [9] proposed an approach using depth data provided by Kinect to detect fingertips. After fingertips were found, the mouse clicking motion was recognized and tested on the popular game Angry Bird, which recognized only one gesture.

In [6], a novel method for contact-less HGR using Microsoft Kinect for Xbox was described and a real-time HGR system was implemented. The system was able to detect the presence of gestures, to identify fingers, and to recognize the meanings of 18 gestures in two pre-defined gesture scenarios: Popular Gesture and the Numbers. The system of the present work consisted of three main components: hand detection, finger identification, and gesture recognition. This system was built on the Candescent NUI [5] project, which is freely available online.

Datcu et al.[10]proposed a computer vision-driven model for natural free-hands interaction in augmented reality. The novelty of their research was in the use of robust hand modeling by combining Viola and Jones and active appearance models (AAM). A usability study evaluated free interaction model of hands with a focus on the accuracy of hand-based pointing for menu navigation and menu item selection. In this approach, detection of dominant and non-dominant hands was done using robust hand detection by adapted Viola and Jones object detectors [11].

The aim of this work was to develop a novel application which could control interactive media via a gesture recognition system. This application enabled working with many common options between Microsoft Offices: Word, PowerPoint, Excel, and Access.

In this paper, simple and intuitive gestures of hands and fingertips were used for fast, applicable, and expressive simulation of natural interaction. Innovations of the proposed model can be divided into three groups: 1. Compatible with most Microsoft Office, 2. Based on menu techniques, and 3. simulation of mouse actions.

Interaction by hand gestures, pointing, and showing count of fingers to select an option, even for a new user, is natural and can be inherently understood. Consistent assignment of signs and gestures for each action is the case that should be considered. Learning a large number of figures is difficult. Thus, in the proposed mode, simple gestures in proportion to the corresponding visual cues were used.

The main steps of the proposed model can be categorized as follows: hand detection, gesture detection, touch point detection, and control software.

The structure of this paper is organized as follows. After the introduction, the steps of proposed model are explained in detail. In Section 3, application of the proposed model in classroom presentation is illustrated. In the final section, conclusion and future works are presented.

2. Architecture of the proposed model

Converting white screen into interactive surface was the goal of the proposed model. The proposed model was tested on programs of Microsoft Office, such as PowerPoint that is menu-based. Menu-based techniques for organizing and selecting commands are used in many applications and menu-based systems have a low cognitive load. Linear menus (menu bar, pull-down menu, etc.) are traditional methods of selecting commands in personal computers.

In the proposed model, traditional menus were extended to finger count based menus. So, the user could refer to menu options with either common ways such as keyboard, mouse, and touch screen or using finger count gestures.

In addition, mouse was simulated by fingertips to facilitate actions like zoom and point.

In the following sections, each step of the proposed model will be completely described.

2.1. Initializing the depth camera

The inputs of the proposed method were depth images (640 pixels wide by 480 pixels high) which were provided by Kinect sensor.

2.2. Background detection and calibration

To distinguish operators' hands from static scenes, a frame of depth image should be captured as the background in advance. The background of the present model was produced by a video projector on the white screen.

At the beginning of starting Kinect, the captured data were unstable. Thus, a frame of depth image as background had to be captured after running Kinect several seconds later.

Here, fifteenth frame of depth image were selected and saved as the background. Depth image of the background was in fact the matrix of depth data. Each number meant the distance between Kinect and background in each pixel. The value "0" meant that the sensor could not get the accurate distance value at this pixel point. The object which should be detected, such as hand, is a blob of points. As a result, the undetected points could be ignored even though they had no value.

2.3. Detecting hand gesture and in case of touching the screen detect touch point

After the background was confirmed, the next phase was hands recognition and tracking. There are different ways for hand gesture detection problem; but, because in the present task, background was static and hands were dynamic, background subtraction methods could be used to separate potential hand pixels from non-hand pixels. First, the depth image was acquired in real-time. Then, it was compared with the saved background. If some values of pixels were different from the background in the corresponding area in a specified range, these areas can be definitely supposed to include the objects located on the background. After that, the areas standing for hands and should be tracked were distinguished.

1) Comparison with the background: In order to detect whether there was a hand above the background, the value of each pixel in the newest image was subtracted from its corresponding value in the background image. Then, a matrix of differences was generated.

2.4. Detecting touched points

Hand gesture recognition was an important step in the proposed method. A simple and effective method that was proposed in [12] was applied.

Let N(a) be the eight-point neighborhood of a pixel a, P denote the current contour pixel, q denote the starting pixel of the current neighborhood checking, and C be the set of detected contour points, which is initialized to be the empty set. From top to bottom and left to right, all pixels on the screen are scanned until a pixel s as a hand point is found, which is set as the starting point. Set the current contour pixel P as s. Set the starting pixel of neighborhood checking q as the point to the immediately north of s. Insert p in C and calculate the neighborhood N(p).

After the hand contours are detected, the centers of the palms are used to calculate the directions of fingers. Centers of the inscribing circles are much more stable than the centroids of hand clusters, because the latter largely varies while opening/closing hands or bending fingers.

Fingertips are detected by checking the three-point alignment relationship [12]. Candidate points are the points that are on both the convex hull and hand contour.

Once the fingers are identified with their names, gesture recognition can be started. Gestures are passed through three layers of classifiers: finger counting, finger name collecting, and vector matching.

1) Finger counting classifier: Gestures are first classified by the number of extended fingers and then sent to the corresponding second layer of classifiers.

2) Finger name collecting: Gestures are further classified by the names of the extended fingers. If the combination of the extended fingers in one gesture is unique among all the gestures, the recognition process terminates and the meaning of the gesture is displayed; otherwise, the gesture is sent to the corresponding third layer of classifiers.

3) Vector matching: The direction vectors of all the extended fingers are measured and all pairwise angles between the extended fingers are calculated. Meaning of the gesture is now classified according to these angles. Then, the meaning of the gesture is assigned and displayed on the screen.

In this work, finger count and fingertip were used for selecting a command. For detecting finger count, the number of fingertips detected in the screen area was counted. If the distance between the detected fingertips exceeded the threshold value, the fingertips were considered to be related to both hands. Finger count gestures were simple, robust, expressive, and fast to perform. Then, each number and path of fingertips had a specific action fingertip.

2.5. Control software

After detection hand gesture or fingertip, they should be used for predefined purposes.

To easily detect options and their related actions, white screen was assumed to be blocking. Each range of blocks was assigned to a particular option. Since toolbars of various Microsoft Office software have collaborative options, the proposed model can be conveniently used for all of them. To review the details of the proposed model, the application of PowerPoint was considered.

Finger count was used to select options of the menu. Users just needed to put a given number N of fingertips in front of the screen. The number of fingers assigned to each option was written on its side. The corresponding command was activated when the user lifted all his/her fingers. After selecting each option, if any, the following options would appear. In the proposed model, the left hand acted to open the menu, while the right hand acted for selecting the item.

Since the maximum options of finger count menu can be more than 25 options, in the proposed model, to move left, right, down, or up between the options of each menu, moving 4-connected fingers in the desire direction was considered.

Based on the path of fingertips on the specific blocks, the mouse action could be simulated.

In the proposed model, if one fingertip was left motionless for a few seconds on the areas pre-defined for the specific Option, click would be performed and that option should be run. If the one fingertip on the areas that were not previously considered for the specific Option was remained motionless for a few seconds, zoom operation would be performed. Zoom action was implemented by up-sampling.

In the proposed model, only finger count and simple, static hand gestures were used. Thus, an incorrect diagnosis or improper interference was easily prevented.

In the proposed model for switching between the finger count menu and mouse modes, fist gesture of the right hand was used. By default, the proposed model used finger count to select menu options. By placing the right hand grip in front of the depth camera, a movement occurred between menu-b and mouse-based states.

3. Application in Classroom Presentation

This section thoroughly examines the natural user interface for presentation by PowerPoint. PowerPoint has a menu like other Windows software. As mentioned in the previous section, for selecting menu options, finger count and hand movements in front of the depth camera were used.

Some user actions will not affect the proposed model (e.g. walking users while presenting). Therefore, in changing the physical to logical states, it is reasonable to consider only the user actions that control physical constraints.

This mapping depends on system operation. In cases such as movement between PowerPoint slides, the mapping between the state of physical device and its corresponding logical position is one by one. Changing conditions in the PowerPoint presentation is limit; the left and right buttons are scrolled to view the slides.

Semantic feedback of pressing the arrow keys is movement between PowerPoint slides. To change the status of a natural user interface, referring with four connected fingers to the left and right is considered.

One of the basic requirements in a PowerPoint presentation is pointing to parts of the slide. Traditionally, this work has been done with a tool called projector. In the proposed model, the use of additional tools was eliminated. Simulating mouse and pointing with fingertip on the slides can act like a projector.

3.1. Implementation

To implement the proposed model, a video projector was needed to produce the background and a depth camera detected the user actions.

Kinect, which is a depth camera, captured a live movie of its front scene and sent it to the computer. So, as explained in the previous section, by comparing the sequence of depth images, movements of the objects and people in front of it could be matched in "real time". It should be noted that the color of each pixel in depth camera indicated how far that part of the image was from the camera and can be used to locate objects. In other words, brighter objects were closer and darker ones were farther away.

The aim of the proposed model was to enable users to control something in the model by using hand gestures. So, finding the pixels of hands in the current frame of series images was always required. In practice, what we needed was to read the images from both depth camera and color camera and then determine the required action according to hand gesture movement in depth image on the pre-defined area of color image for a specific Option. In order to detect users and their actions from raw depth image, "OpenNI" was applied. At all times, while the user was using the application, OpenNI provided accurate information on the position of each user's joints.

Details of the proposed procedure were explained in the previous section. As an example of openNI programming, in Figure1, the source code of hand's contour extraction is illustrated. Figure2 shows a sample of the detected hand.

```
import SimpleOpenNI.*;
SimpleOpenNI kinect;

int[] diffdepth;
int diff;
intclosestValue;
void setup()
{
size(640, 480);
kinect = new SimpleOpenNI(this);
kinect.enableDepth();
}
void draw()
{
closestValue = 8000;
kinect.update();
int[] depthValue1   = kinect.depthMap();

kinect.update();
int[] depthValue2  = kinect.depthMap();

for( int y=0; y<480; y++){
      for( int x=0;x<640;x++){

intreversedX = 640-x-1;
                  inti = reversedX + y * 640;

intcurrentDepthValue = depthValue2[i];

if(currentDepthValue> 610 &&currentDepthValue< 1525
&&currentDepthValue<closestValue){

                  closestValue = currentDepthValue;
                  diff= depthValue1[i]- currentDepthValue;
                  if ( diff >0)
                              diffdepth[i] = 1;
                  else
                              diffdepth[i] = 0;
      }
}
}
image(diff,0,0);

}
```

Figure 1. Source code of hand contour extraction.

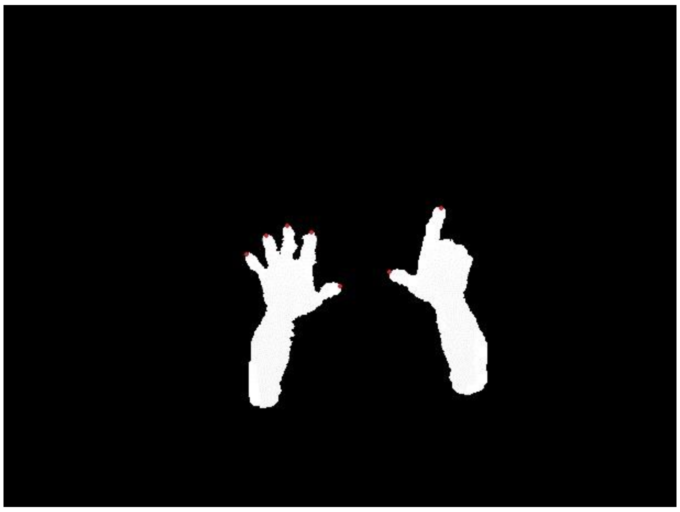

Figure **2. The depth image to capture hands.**

As illustrated in the previous section, the fingertips could be detected. Red points in Figure2 show fingertips. Using the number of fingertips, the finger count can be determined and options of finger-count menu can be selected.

If one fingertip was found, as explained before, it was important to determine its position. If a predefined specific option was considered for this area, "click" had to be simulated. Thus, the image was blocked and a specific option was considered for some blocks In order to more simply identify the areas.

Since the fingertips were identified, the direction of fingertip movements can be determined to highlight parts of a slide. Parts of this code are shown in Figure3. By moving the slide before or after, the highlighted direction would be removed.

```
floatinterpolatedX = lerp(lastX, closestX, 0.3f);
floatinterpolatedY = lerp(lastY, closestY, 0.3f);
line(lastX, lastY, interpolatedX, interpolatedY);
lastX = interpolatedX;
lastY = interpolatedY;
```

Figure **3. Parts of code to shown hand tracking.**

As mentioned in the previous section, the ability to zoom during a presentation was another function in the proposed model. After the region of interest was determined and saved in Cimage, with the instructions shown in Figure 4, this task can be easily performed. For each image, their scale was multiplied by their original width and height.

```
image(Cimage, CimageX, CimageY, Cimagewidth * imageScale, Cimageheight * imageScale);
```

Figure **4. Parts of code for zoom operation.**

4. Conclusion and future works

In this paper, a new model was proposed to transform white screen into interactive surface. Interaction on a much larger screen and any surface of objects were the main benefits of the proposed model. Furthermore, it can replace expensive and not-convenient-for-moving tablets and smart boards. For this purpose, hand gesture and fingertips were used.

The proposed model can run with all software in Microsoft Office package. Because most of Windows applications have similar toolbars, in future works, a general model that can work with many Windows applications can be generated.

5. References

[1] Castaneda, V., D. Mateus and N. Navab, *SLAM combining ToF and High-Resolution cameras*, in *IEEE Workshop Appl. Comp. Vision*January 2011: Kona, Hawaii, U.S. p. 672–678.

[2] Q, C., et al., *3D Deformable Face Tracking with a Commodity Depth Camera*, in *Eur. Conf. Comp. Vis., Lecture Notes in Computer Science.* September 2010: Crete, Greece. p. 229–242.

[3] Ramey, A., V. Gonzalez-Pacheco, and M.A. Salichs, *(Integration of a low-cost RGB-D sensor in a social robot for gesture recognition*, in *6th Int. Conf.Human-robot Inter*March 2011: Lausanne, Switzerland. p. 229–230.

[4] Chang, Y., S. Chen, and J. Huang, *A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities.* Devel,Disab, 2011. **32**(6): p. 2566 – 2570.

[5] Bo A. P. L., M.H. and P. Poignet, *Joint angle estimation in rehabilitation with inertial sensors and its integration with kinect*, in *IEEE Intl. Conf. Eng. in Med. and Bio. Society*September 2011. p. 3479 -3483.

[6] CNET: Gates demos TouchWall computer, A.h.n.c.c.-.-.-h.p.r.s.n.t.-.-.-.

[7] Yang, c., et al., *Gesture recognition using depth-based hand tracking for contactless controller application*, in *Consumer Electronics (ICCE), 2012 IEEE International Conference*2012. p. 297 -298.

[8] Raheja, l.L., A. Chaudhary, and K. Singal, *Tracking of fingertips and centers of palm using KINECT*, in *Third International Conference on Computational Intelligence Modelling Simulation*2011 p. 248-252.

[9] He, G.-F., et al., *Real-time gesture recognition using 3D depth camera*, in *in Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference* 2011. p. 187 -190.

[10] Datcu, D., *Free-hands interaction in augmented reality*, in *SUI '13 Proceedings of the 1st symposium on Spatial user interaction* 2013: ACM New York, NY, USA. p. 33-40

[11] Viola, P. and M. Jones, *Robust real-time object detection.* International Journal of Computer Vision, 2002.

[12] Li, Y., *Hand Gesture Recognition Using Kinect.* IEEE, 2012.