USING KINECT FOR TOUCHLESS INTERACTION WITH EXISTING APPLICATIONS

Andrej Černivec, Ciril Bohak
Faculty of Computer and Information Science
Univerity of Ljubljana
Tržaška 25, 1000 Ljubljana, Slovenia
Tel: +386 1 47 68 483; fax: +386 1 426 46 47
e-mail: ciril.bohak@fri.uni-lj.si

ABSTRACT

In this paper we present a system for touchless interface using Kinect sensor. The system can be used for general purpose interaction with existing applications. We show a case study implementation for use with web-browser where user can interact with the content using gestures. We also present implementation of the game that uses presented system for handling interaction with the user. We also present advantages and disadvantages of such framework and give possible solutions for some of the problems.

1 INTRODUCTION

In the recent times there is increasing demand for natural user interaction and consequently there is a need for such systems as well. While there were many implementations of touchless interfaces in the past [1, 4, 5], most of them were implemented for use with dedicated applications in limited environment. In the past there were many different devices used in such setups, in recent time more interfaces are available at consumer affordable prices such as Microsoft Kinect[1] and Leap Motion[2]. While the intended primary use of Kinect was for gaming purposes, many researchers and developers have successfully used it in more general purpose scenarios. Both interaction devices were used as an input in different systems. Use of Kinect is presented in [3], where authors present a web based kiosk system. Leap motion was used as a doodling interface presented in [8].

In our case we present a system for general purpose touchless interface for use with existing applications on Microsoft Windows platform. The rest of the paper is organised as follows: in the following section we present the related work, in Section 3 we present our implementation. In Section 4 we discuss the advantages and disadvantages of our method. In the end in Section 5 we give the conclusions and possible future extensions of presented approach.

[1]http://www.microsoft.com/en-us/kinectforwindows/
[2]https://www.leapmotion.com/

2 RELATED WORK

The idea of controlling Windows applications using the Kinect sensor is not new and there have been many attempts to achieve this functionality. Most solutions are based on Kinect for Windows Software Development Kit, while some of them use open source libraries like OpenNI[3] or OpenKinect[4]. Basically all of the existing solutions were open source projects, developed by individual programmers in their free time. Most common practice is controlling Windows applications by transforming hand movements into movements of the mouse cursor as described in [7]. Limited functionalities of the left and right click are achieved through simple gestures, like raising one hand or pressing towards the screen with the palm of the hand.

Some of the solutions extend the functionality to support a couple of other gestures to control actions like zoom in and zoom out or even drag and drop. Limited gesture recognition is also presented in [6]. While some work quite well, we could not find any of them that are dealing with the issues of controlling the application in harder circumstances, for example in a public place with a lot of people (e.g. expos, conventions and conferences). This means that there can be high level of background noise (in form of people passing by behind the user) as well as occlusions (in form of people passing between the user and the system). That is why we decided to create our own system that implements all of the functions of existing solutions and takes a step further in terms of usability.

3 OUR APPROACH

Our system consists of four main elements:

- PC running Microsoft Windows 7 or above,
- Microsoft Kinect sensor,
- Large computer display or TV screen,
- Software framework.

[3]https://github.com/OpenNI/OpenNI
[4]http://openkinect.org/

The idea is that a person can stand in front of a large screen and control any existing Windows application through touchless interface using Kinect sensor. As it is displayed in the Figure 1, large TV screen is used to display an image from the computer. Kinect sensor is connected directly to the computer and positioned in front of the TV screen. The application was developed using Kinect for Windows SDK, .NET framework and two open source libraries, Coding4fun Kinect Toolkit[5] and InputSimulator[6]. Coding4fun Kinect Toolkit is intended to speed up the process of developing applications for Kinect by replacing repetitive parts of code with short commands. The other used library, InputSimulator, helps with the simulation of mouse and keyboard commands using Win32 SendInput method.

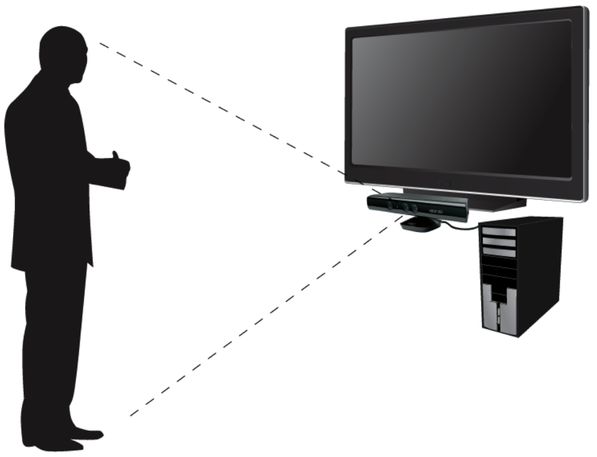

Figure 1: The scheme of our system.

3.1 System outline

When a person steps in front of the Kinect sensor, he or she immediately starts to interact with the computer using the predefined commands. For normal interaction, user has to have both of his palms wide open and facing the sensor. If the user moves his right hand, the mouse cursor on the screen moves accordingly. If he closes and opens the palm of his right hand while keeping the palm of his left hand open, he clicks the object under the mouse cursor. By keeping the right hand palm closed for a longer period of time, the user can perform drag and drop action. If the user closes the palm of his left hand and keeps it closed, he can scroll in any direction just by moving his right hand in desired direction. The distance of the right hand from the starting point translates into the speed of scrolling. If the user wants to zoom in, he needs to close both of his palms simultaneously and, while keeping them closed, pull the hands away from each other. Zooming out can be achieved by doing the opposite action, pulling the hands together. Using this simple commands, user can control and browse through any content on the screen. This is ideal for several cases, like promoting products on conventions, info points, public web browsing and so on.

[5]http://c4fkinect.codeplex.com//

[6]http://inputsimulator.codeplex.com/

3.2 Interaction adaptation

To make the experience of using touchless interface easy for an ordinary user, we had to use different approaches to solve some problems. The commands needed to be simple, logical and very limited, so the user could master them very quickly. Scaling of the moves had to be integrated, because making a lot of repetitive big moves in front of the screen can be tiresome. The hardest part was to find the line where user moves are small enough to be easy and still accurate enough so that he can select the content on the screen without a problem. We got the best ratio of scaling moves through testing different settings on a number of users, which turned out to be about 1:3. Smoothing of users moves also had to be used to improve accuracy and to avoid unwanted random movement of the mouse cursor.

Next step was to make sure that all of the gestures were working as intended and did not collide with each other. The problem appeared when user tried to use zoom in or zoom out gestures. These gestures require that the user closes the palms of both hands at the same time. This turned out to be hard, because if the user closed one palm slightly before the other, different gesture would be triggered, for example mouse click. To avoid this problem, a slight delay was introduced to give the user more time to complete the gesture. Through testing on a number of users we established that a delay of 0.7 seconds is enough to avoid gesture mistakes and still keep the system responsive enough. This was obtained from an user experience evaluation presented in [2].

Because our main goal was to use the system in public places, we had to solve the problem of user recognition and control locking. First generation of Kinect sensor was intended to simultaneously track the skeletons of two persons, while it can recognize up to six persons in his field of view. Our system was designed to allow only one user at a time. We solved this problem by assigning a unique ID to each user in a field of view. Only the first person in the field of view was granted control over the system. As long as the user stayed in the field of view, other users could not take over the control or interfere with it. After the first user was done, he would step aside and the system would grant the control to the next user and so on.

3.3 System structure

Our system consists of several components as presented in Figure 2. Kinect is used as an input that sends the data through the Kinect for Windows SDK. Our application connects to the Kinect Windows SDK and transforms retrieved data into form appropriate for InputSimulator. Input simulator then sends data to the Windows operating system through native Windows input method. Coding4Fun is used for easier implementation of Kinect communication code.

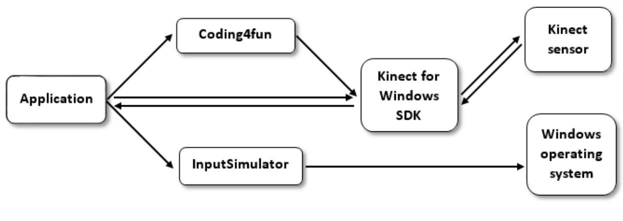

Figure 2: Scheme of system implementation.

3.4 System goals

The main thing that we wanted to achieve was the ability to control any Windows application without communicating with the application. We wanted to develop a system in such a way that the application we control believes it is controlled by a normal mouse and keyboard. This way, there would be no compatibility issues with old or new applications. To get the desired functionality we used existing Windows APIs to inject mouse and keyboard commands into the operating system.

After we implemented user tracking, we used this information to implement another function, automatic program starter. Every time a system detects a new user, it can automatically start any predefined program or open a new website. The idea behind this function is that every time a new user steps in front of a sensor, a so called "welcome" sequence can be started as shown in Figure 3. This way, a new user can be greeted and introduced to all of the functions our system enables him to use.

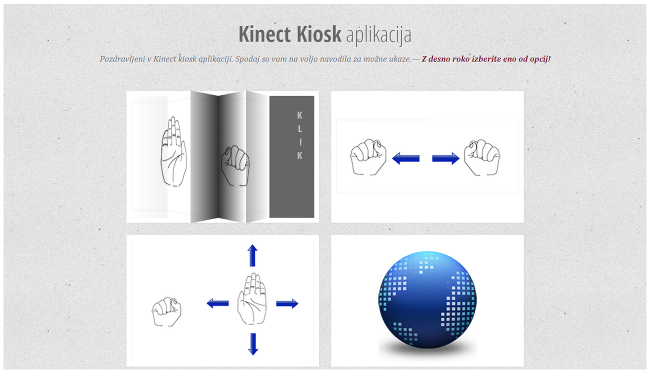
Kinect Kiosk aplikacija
Pozdravljeni v Kinect kiosk aplikaciji. Spodaj so vam na voljo navodila za možne ukaze.— Z desno roko izberite eno od opcij!
KLIK

Figure 3: An example of welcome screen which can be used with our system.

Each of the gestures can be separately turned on or off, so they can be used only if needed. More gestures always means chances for a user to make a mistake.

3.5 System integration

We tested our implementation of touchless interface in a real life situation. An IT company from Slovenia wanted to present itself on an IT conference using touchless interface. The goal was to stand out and attract people passing by. We developed a simple browser based game that was controllable using our touchless interface. The application was developed for general use and can use conventional means of interactions as well (such as mouse and keyboard). The goal was to follow a given path with a mouse cursor from start to finish without touching the borders of the path. To avoid mistakes, all unnecessary gestures were turned off. The game was interesting to play when using our touchless interface because every user first had to learn how to guide the mouse cursor using only his hand and familiarize himself with the responsiveness and accuracy of our system. It turned out that some people had trouble replacing the mouse with a touchless interface while other mastered the game in seconds. It was very obvious that with a bit of training, usefulness and accuracy of our system was good. A screenshot of the application is presented in Figure 4.

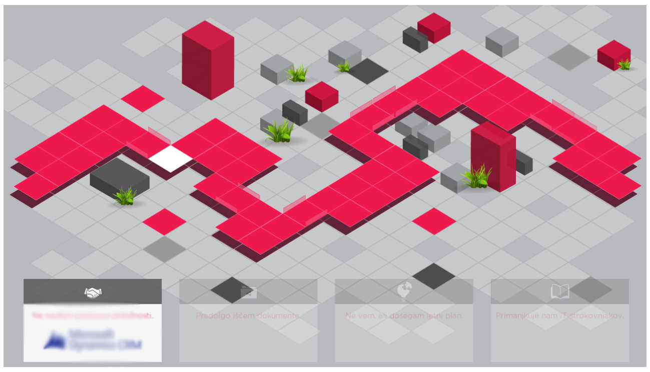

Figure 4: A screenshot of example application that uses our system.

4 DISCUSSION

Even though there are already numerous existing solutions that enable users to control Windows applications using Kinect sensor, our system stands out in a number of ways. It does not only focus on enabling the users to give basic commands using touchless interface but also focuses on dealing with problems of using such an interface in public places. It implements functionalities such as user control locking, gesture recognition delay, user movement scaling, selective gesture enabling/disabling and automatic program starter. It was developed with a specific purpose in mind and therefore concentrates on problems that other applications were not meant to encounter. That is why it deals with the problems better than other similar applications and is usable not just in a lab environment but also in real world situations.

On the other hand, our application is limited to only a few gestures and could be expanded to support other commands. This way, user could control more complicated applications or get desired results faster. By expanding our feature set and improving the quality of already implemented features we could develop a more all-rounded application that would be useful in different situations and for different purposes.

The great advantages of presented system is possibility to use such system with existing windows applications. That can come in handy in many cases where we want to adapt a kiosk system for use with such applications. Using separate hand for pointing and clicking gives user a better accuracy in selecting the correct user interface (UI) elements.

On the other hand use with existing applications is limited to some extent due to the limitation in UI. Usually UI elements are just too small for interaction with such system. There is a possibility of adapting applications for use with presented system. In the conclusion we give possible adaptation of web-based application for easier use with presented system.

Our system was also tested in the real-life situation at a conference, for purposes of obtaining relevant user feedback. Users were participants of IT conference, which means that majority of them had experiences with different ways of interaction. Majority of users gave us positive feedback. During the testing we have realized that one of the main limitations of our system is that it fails in interaction with smaller UI elements, where one has to be very observant to pinpoint the element exactly.

5 CONCLUSION AND FUTURE WORK

In this paper we have presented a system for touchless interaction using Kinect sensor, that allows robust user interaction. The system is developed for Microsoft Windows and allows interaction with native applications. While the idea of such implementation is great, it shows that there are rare applications that allow such interaction due to UI limitations. We have shown that presented system can be used for kiosk-based applications, in our case an interactive game, where user is robustly tracked and the system is not distracted by the actions in the background.

There are several possible extensions of the presented system that would improve its usability. One such extension is implementation of intermediate application for web browsers that would adapt the displayed page for easier interaction with the system. Such application would "snap" cursor to the defined links in the website and emphasize them for easier recognition and interaction. Another extensions of the system is implementation of more gestures, such as right-clicking, onscreen keyboard with auto-completion etc.

The presented system was tested in real-life environment with positive user feedback. We will further develop the system and also perform user experience study to get a relevant feedback from the users.

References

[1] R. Barré, P. Chojecki, U. Leiner, L. Mühlbach, and D. Ruschin. Touchless interaction - novel chances and challenges. In J. Jacko, editor, *Human-Computer Interaction. Novel Interaction Methods and Techniques*, volume 5611 of *Lecture Notes in Computer Science*, pages 161–169. Springer Berlin Heidelberg, 2009.

[2] C. Bohak and M. Marolt. Kinect kiosk user experience evaluation. In *Proceedings of the 16th International Multiconference Information Society - IS 2013*, pages 201–204, Ljubljana, 2013.

[3] C. Bohak and M. Marolt. Kinect web kiosk framework. In A. Holzinger, M. Ziefle, M. Hitz, and M. Debevc, editors, *Human Factors in Computing and Informatics*, volume 7946 of *Lecture Notes in Computer Science*, pages 785–790. Springer Berlin Heidelberg, 2013.

[4] R. Held, A. Gupta, B. Curless, and M. Agrawala. 3d puppetry: a kinect-based interface for 3d animation. In R. Miller, H. Benko, and C. Latulipe, editors, *UIST*, pages 423–434. ACM, 2012.

[5] R. Johnson, K. O'Hara, A. Sellen, C. Cousins, and A. Criminisi. Exploring the potential for touchless interaction in image-guided interventional radiology. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3323–3332, New York, NY, USA, 2011. ACM.

[6] G. Ruppert, L. Reis, P. Amorim, T. Moraes, and J. Silva. Touchless gesture user interface for interactive image visualization in urological surgery. *World Journal of Urology*, 30(5):687–691, 2012.

[7] Y. S. Ryu, D. H. Koh, D. Ryu, and D. Um. Usability evaluation of touchless mouse based on infrared proximity sensing. *Journal Of Usability Studies*, 7(1):31–39, 2011.

[8] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: An interface for sketching character motion. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.