IBIMA Publishing
Journal of Software & Systems Development
http://www.ibimapublishing.com/journals/JSSD/jssd.html
Vol. 2016 (2016), Article ID 557104, 9 pages
DOI: 10.5171/2016.557104

Research Article

Modelling Gesture Recognition Systems

Zenon Chaczko[1] and Wael Alenazy[2]

[1]The Centre for Innovation in IT Services and Applications (iNEXT), Faculty of Engineering and Information Technology University of Technology, Sydney

[2]The Centre for Innovation in IT Services and Applications (iNEXT), Faculty of Engineering and Information Technology University of Technology, Sydney and King Saud University, Riyadh, KSA

Correspondence should be addressed to: Wael Alenazy; walenazy@ksu.edu.sa

Received date: 31 August 2015; Accepted date: 9 December 2015; Published date: 7 June 2016

Academic Editor: Janet Renwick

Abstract

Gesture recognition technologies have recently become available to a general consumer market. The most notable of these technologies rely on camera to support gesture recognition using multiple sensors that allow for standard color pictures, infrared pictures and depth sensing. These capabilities allow for a variety of applications, including a body's skeleton or hand recognition. The recognition features allow for a multiple type input control. They allow an actor, who is in front of the camera, to apply different hand gestures, or body movements as the controls for an application or game that is running on the system the camera is plugged into. This paper discusses modelling and simulation of gesture recognition technologies.

Keywords: modelling and simulation, gesture recognition, multiple sensors systems.

Introduction

Gesture recognition (Erol et al. 2007, Nickel & Stiefelhagen 2007, Fengjun and Shijie 2010) represents a relatively new field of study. In recent years, gesture recognition technologies have become readily available for educational use (Montero 2013) and a general public at an affordable price. The most prominent of these technologies is the Kinect for Xbox and Windows. The Kinect camera units are equipped with multiple sensors that are integrated (Wong 2011) to allow for captures of standard color pictures, infrared pictures and depth sensing. These capabilities allow the system to be used in various augmented reality mobile applications where skeleton and hand recognition functions do not require extensive computation capabilities (Prasad 2013). The recognition features allow for a different type of input control in various applications. Often these recognition features are used in immersive and interactive applications (Dede 2009, Dunleavy et al. 2009). One of the most prominent of these applications is the Augmented Reality (AR) which is commonly used in educational, training, entertainment or even military contexts (Chaczko and Alenazy, 2014, Sholtz, 2014, Chen et al. 2014, RemoteLab 2015). The recognitions features allow the user or actor, who is in front of the camera, to apply different hand gestures, signals or body movements as the controls for an

Cite this Article as: Zenon Chaczko and Wael Alenazy (2016), "Modelling Gesture Recognition Systems ", Journal of Software & Systems Development, Vol. 2016 (2016), Article ID 557104, DOI: 10.5171/2016. 557104

application or game that is running on the system the camera(s) is plugged into. Another gesture recognition technology, that is available for general consumer market, is DepthSense from SoftKinetic (2014).

The DepthSense cameras provide functions designed to recognise hand gestures at a close range, as well as, various functions that require a support for longer range interactions suited for the full skeleton recognition applications. At present, there are multiple ways of developing software available for these cameras. Microsoft has developed its own freely available Software Development Kit (SDK), and there are other third party options. One such option is SoftKinetic's IISU SDK. IISU is the "Most advanced real-time 3D gesture recognition software platform on the market" (SoftKinetic 2014). The IISU SDK does not currently have direct support for the Kinect camera but that feature will be available in an up-coming update of the SDK.

This paper elaborates on the investigation of the IISU infrastructure software as a platform for modeling, simulation and development tool of gesture recognition user applications. These user applications would allow decision makers to monitor the actors who are in front of the camera, as well as, to enable controls that can be hooked into a piece of software in order to facilitate remote controls.

Modeling of Gesture Recognition

Initial Requirements

For simplicity, there was a rather small set of initial requirements (Table 1) being defined for the development of the software system prototype to monitor the camera(s), actors and the third party applications.

Table 1: General Requirements for the Prototype

ID	Description	Priority
GRS-001	Users shall be provided with an application that can monitor the camera and actors	Mandatory
GRS-002	Users shall be able to control third party applications	Mandatory
GRS-003	The application shall run on smart devices	Mandatory
GRS-004	The Gesture Recognition System shall use the IISU SDK toolkit	Mandatory
GRS-005	The application will work with gesture recognition camera(s) such as Kinect and/or similar	Non-mandatory
GRS-006	The Gesture Recognition System will allow for high usability of the selected third party middleware	Non-mandatory

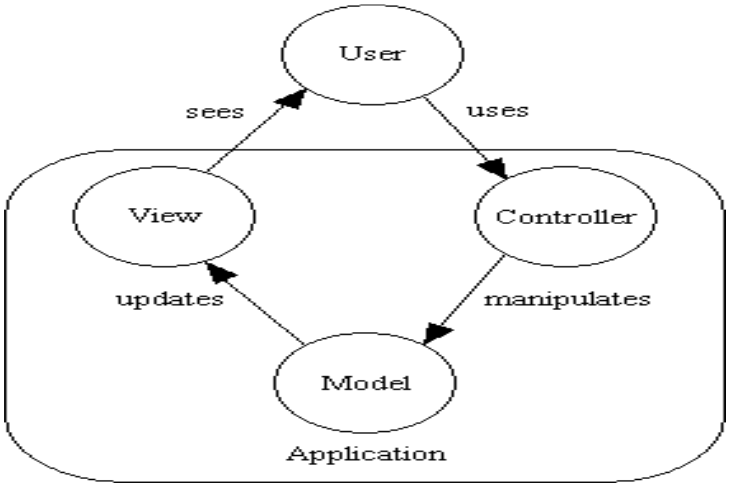

Figure 1: Model-View-Controller (MVC)

Programming Environment

A prototype of Gesture Recognition application was implemented using ASP.Net and HTML5. The rationale was to ensure that the application will be able to run on various smart devices using Android, iOS or Windows Phone. For portability, the application was developed using HTML5 scripting instead of using a dedicated high level programming language such as Java, Objective C or a .NET C# language. This decision was made due to all three major mobile operating systems supporting HTML5 as an application language. Using HTML5 also allows the application to be run in various Internet browsers on desktop or laptop computers. The IISU SDK was another reason to use ASP.NET and HTML5. The IISU SDK includes .NET language libraries for both C++ and C# to allow for a direct communication with the DepthSense cameras. Since IISU's SDK does not support Java or Objective C, this ruled out the development of an Android or iOS application. Windows Phone applications can be developed with .Net languages such as C#; however, there was no readily available testing and integration platform for this option to be viable. Thus, for these reasons alone, a language that allows for the use of Internet browsers that run on all three operating systems was chosen.

Architectural Model

Choosing the language to be ASP.NET, along with previous experience, limited the design architecture to the Model-View-Controller (MVC) architecture (Fig. 1). The MVC architectural model consists of three separate types of information representation. The model also includes application data and business rules. The MVC view consists of anything that the user is able to see on the screen, which can consist of a graph or an image, or the whole page that is shown in the Internet Browser. The controller is what converts the user's input into logic for the model and displaying different views.

Hardware

The initial hardware decision was to use an Xbox Kinect as the camera hardware for the system. This was due to the fact that they are readily available to consumers. Also, Microsoft provides its own SDK for the Kinect; however, this was not compatible with the IISU SDK and could not be installed, if the IISU was to be used. This decision was changed in favour of using a DepthSense 325 camera a half way through the project, due to compatibility issues with the current version of the IISU middleware and the Kinect camera.

Application Communication

There is a number of different ways for communication between separate applications. These include: Windows Communication Foundation (WCF), named pipes, and Microsoft remoting. Each option was investigated for usability in the application. Microsoft remoting has become a legacy technology that has been essentially replaced by WCF. Hence, WCF was chosen as the method for communication between two different applications due to better application integration. WCF provides the ability for applications to communicate while running on the same machine, as well as, for applications that are running in distributed computing environment.

Development Technologies and Tools

For the development and testing of the application, the Samsung Galaxy S2 smartphone and Nexus 7 (Google) tablet running Jellybean Android, as well as a desktop PC with Windows 8, SoftKinetic DepthSense 325 and the Xbox Kinect were used. For developing the software application, the following software tools were used:

- Visual Studio 2012. This was used as the main program to develop the application in HTML5 and ASP.NET. It also allowed for the integration of the IISU SDK.

- IISU middleware. The IISU SDK and middleware were used to create scripts that can be integrated into a .NET application; these help determine the actions that the actor is taking in front of the camera.

- Internet Information Services (IIS). This was used to host the website which allowed for testing on mobile devices.

- Chrome for Android. This was the web browser that was used as a testing platform.

Application Development

Streaming

Originally the plan was to stream video directly to the website being developed. In researching the camera and reviewing the documents of the IISU SDK, this was not easily achievable with the samples provided. The camera works by taking individual frames of the scene and then processing them to give the relevant information, such as where the actor or player is. As such the SDK provides a connection to be able to pull the image information frame by frame. To use this for streaming, each frame would have to be individually added to a video file, which would then stream to the website. If the camera is then left running while building a video, storage issues could occur. Instead of streaming video, individual images which would refresh at the time the user chooses were implemented. Using individual images allowed for the image to be displayed quickly on the website due to their small size. There are still issues with this, as when the user chooses a higher refresh rate (100ms), the image flickers on and off while the image is updated. This is less noticeable at higher refresh rates such as one second.

Controlling Separate Applications

Integrating WCF into the application was not very difficult due to previous experience and online resources. The only issue that arose was that a service in the control panel needed to be running. In the administrator tools in the control panel, in the services application, the Net. Tcp Port Sharing Service needed to be turned on. This was to allow WCF communication using the Net.tcp implementation.

The IISU Middleware

The original plan for this project was to use a Kinect camera for the IISU SDK to communicate with. This plan was changed to use the DepthSense 325 camera from SoftKinetic due to compatibility issues between Kinect and the IISU middleware. In order to use the Kinect with IISU, the OpenNi framework is required to be installed. This was unable to be achieved due to version changes of both IISU and OpenNi. OpenNi recently updated to OpenNi version 2 which has more compatibility with Kinect. This version was not compatible with the IISU SDK as it did not recognise the framework as being installed. Unfortunately, downgrading the OpenNi to version 1.5 and installing Nite, another piece of software supplied by OpenNi, and a specific Kinect driver provided by OpenNi also did not work as recommended by their previous installation notes. This was because the latest IISU SDK version had been upgraded and was not able to communicate with the camera. It was also not possible to download a previous version of the IISU SDK to communicate with the previous version of OpenNi, due to the fact that the IISU site does not provide this option. Using the DepthSense 325 was easier to use and install as it is a camera provided by SoftKinetic and the drivers have direct compatibility with the IISU middleware. Once the drivers were installed, the users are able to choose the camera as the source and instantly start using it with the IISU middleware.

Help and Support

The help and support systems for the IISU SDK that were found over the course of the project were not entirely sufficient. IISU middleware itself provides a number of ways for help and support such as a helpdesk ticketing system, and a forum where users and IISU employees can post problems and solutions that they have run into.

The user forums are a moderate source of information for problems and solutions that SoftKinetic provides. Posting a problem on the forums does not guarantee an immediate response, or any response at

all from someone, which can be frustrating if time is a factor. The existing problems and solutions on the forums do not cover a wide variety of issues and a lot of the forum posts have not been responded to with a solution. The user forums are a moderate source of information for problems and solutions that SoftKinetic provides.

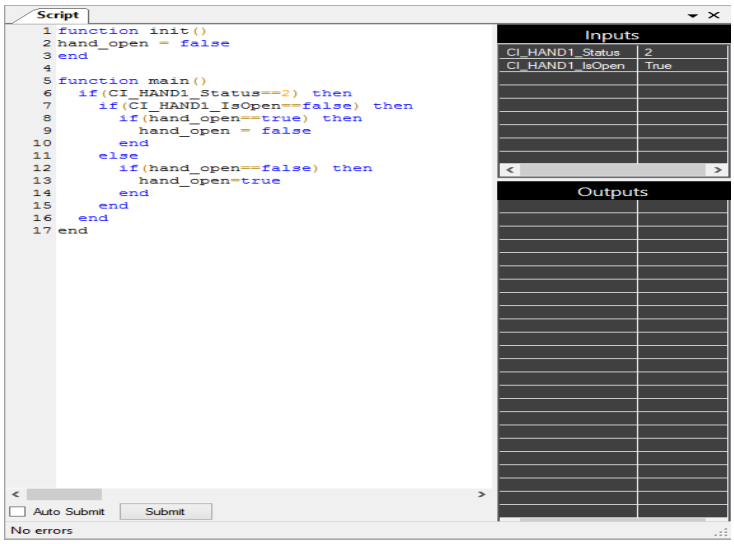

Figure 2: The IISU's Lua script

Posting a problem on the forums does not guarantee an immediate response, or any response at all from someone, which can be frustrating if time is a factor. The existing problems and solutions on the forums do not cover a wide variety of issues and a lot of the forum posts have not been responded to with a solution. This resource was tolerable for help and support but could benefit from more active users. Getting in direct communication with SoftKinetic was not easy either. The helpdesk page did not have an obvious create new ticket option, and the styling of their website on the helpdesk page was not consistent and hard to navigate. Once a ticket has been created, however, the questions were answered quickly, thoroughly and they were very helpful. The only email option for SoftKinetic is to get in contact with the sales department. The sales department was quick to reply and gave as much help as they could with the issues that arose.

IISU Scripts

The Interaction Designer is part of the IISU software that allows developers to create scripts that can be used to recognise people, body parts and gestures. The presented example of the Lua script (Fig. 2) checks whether the user's hand is shown on the screen, and then, if it is shown will output whether or not the users hand is open or closed. The language the scripts are written in is Lua. The Interaction Designer offers a fairly easy to use interface that offers the developers pre-defined inputs that the camera can provide, such as left arm angle, right hand angle, and torso position. These inputs return specific information that can be utilised by the developers to create scripts they wish to implement in the application. For example, using the angle of the torso and arms, and the position of the head of the user, the script is able to recognise someone bowing to the camera. Also, if the camera is the close interaction DepthSense 325, it offers a number of different inputs such as whether the user's hands are open or individual information about the user's fingers. The scripts are able to compute these inputs and outputs into specific variables of the developers' choice. For example, in the previous example of the user bowing to the camera, instead of outputting the individual angles of the user's torso and head, the script can output a Boolean that just shows if the user is bowing or not.

The Interaction Designer offers a number of tools that allow for easier creation of the scripts by developers. These include the ability to record a video and use it as the input to the script instead of the camera. This allows for easier development as it will mean the developers do not have to keep getting up and performing an action in front of the camera to test the script. The Interaction Designer also allows for real time debugging which enables the developers to create the script and see the inputs and outputs instantaneously without having to run a separate application and restart if the code has changed. The .NET libraries that IISU provides also provide an easy way for these scripts to be utilised. They allow for the script to be imported to the application being developed in .NET and give access to the outputs the developers have created in the script itself.

Net Support

The IISU SDK provides libraries that can be implemented with the .Net Languages. The specific languages that they support directly are C# and C++. The documentation that IISU provides is far more thorough for C++, which provides a lot more sample code and information about the specific functions that can be used. The documentation shows examples of container classes, such as Vector3 and Image, which seem to be completely different in the C# documentation and implementation. Through research and experimentation, it was found that the classes are named only slightly differently in C#.

Communication between the DepthSense camera and .NET languages is handled well with the IISU libraries. To start using the camera, a few steps need to be taken before turning the camera on. First an IISUHandle must be declared. The IISUHandle is the integration point of the IISU Dynamic-link Library (DLL), which allows for the creation of the camera device, and gives access to two services, the events and commands. The handler can also load specific external configuration files if the user wishes; otherwise the system will use the configuration that has been set up in the IISU Advanced Configuration. Once the handler is created, the camera device can be created and started. When the device has been started, the code can start pulling and updating frames of the scene the camera is viewing. If the developers choose, they are also able to load the script files that have been created within the IISU middleware. This will allow the developers to have access to the outputs that have been specified in the scripts. To start pulling information from the camera, after the device has been set up, querying of the information that the camera can be provided is possible as well.

Figure 3: A sample of the depth map image

This is done by registering a data handle from the device, and specifying what information is wanted from the device. There are specific strings values that are sent to the device to pull the information, such as "SOURCE.CAMERA.COLOR.Image" which will return an ImageData object that the camera can provide (Fig 3).

Retrieving an Image from the Camera

The main requirement of this project was to be able to monitor what the camera was seeing. Being able to pull an image from the camera was not the easiest part of the project. The task might look deceptively trivial, due to the fact that pulling the ImageData from the camera is fairly easy. Pulling the ImageData is done by registering a data handle, just like all the other information the camera can provide. The ImageData provides information about the image, such as width and height. The image itself is stored in memory and the ImageData contains a pointer to this location. Converting the image to something that is viewable is not a trivial task. First, the pointer must be converted to an array of sixteen bit integers. This should be the raw image data property, which still needs to be normalised. This Integer array is then normalised into a byte array, which can then be converted into a bitmap and then saved. The ability to save the depth image was found in a forum post on IISU's website, and the code needed to be heavily experimented with and altered before a clear depth map image was visible. The prototype is currently only able to retrieve a visible depth map image (Fig. 3) which shows the person in the scene in different shades of black, white and grey. Also, it is able to show how close or far away the user is. The colour image that the camera can produce is not as easy to retrieve. There is a piece of sample code that shows how to pull a colour image, but instead of saving it, it is rendered using the OpenGL libraries. These libraries can render an image to be viewed on a Win Forms application, but when registering it to a website, which specifically needed a source picture, it was not usable.

Results

The result of the presented work was a development of an internet website application that can successfully monitor the users who are acting in front of the camera(s), and that can also provide controls for other possible applications. The application runs in chrome on a PC desktop. Fig. 4 shows the website, which has the following features:

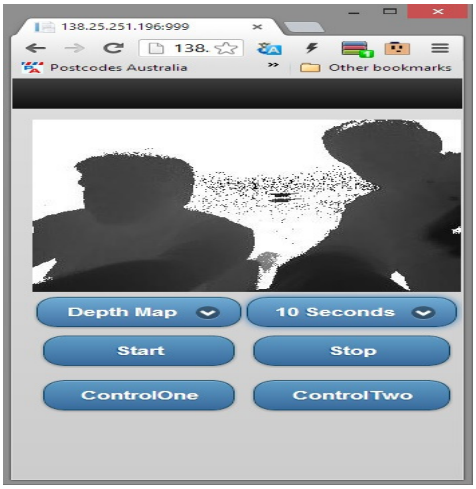

Figure 4: The web-enabled gesture recognition application

• Image of the camera output. At the top of the screen, an image shows the scene that the camera is currently seeing. It is a depth map image which will show a gradient of how close or far away the user is.

• Controls for the Image. There are a number of controls that have been set up for the image. The image selector allows the choice of the colour image or depth map image; however the colour image does not currently work. Controls for the refresh rate of the image. This can vary from 100 milliseconds, up to 10 seconds.

• Buttons to control a separate application. These buttons consist of start, stop, control1 and control2. These are customisable buttons that can be hooked into another application to offer a modicum of control.

To allow the website to use these features, it has some supporting infrastructure:

• Custom WCF Library. This library is implemented in the website, as well as the target application that the website will be controlling. It contains the methods and logic to use WCF in both applications.

• Camera Communication Console program. This is the program that talks to the camera and saves the image it is viewing. When the console program is not running, the image will not refresh properly. And when the program is not already running, navigation to the website will initiate start-up.

By reviewing these features alongside the requirements, this project has been deemed a success.

Conclusion

To conclude, the IISU SDK is very useful software to model and develop gesture recognition applications. The IISU SDK supports a few different camera models and the supported cameras are being updated and added to.

At present, the Kinect has still some version issues with the third party drivers and middleware such as the IISU to connect with the Kinect. However, in the IISU bulletin news it has been announced that the next version of IISU being released will have direct compatibility with the Xbox Kinect.

Developing applications using the Interactive Designer is easy to use and provides many inputs that the camera computes such as arm angles and whether hands are open or closed. If there are any issues with development with IISU, there are help and support forums and a ticketing system that can provide help. Unfortunately as it is not a widely used piece of software, the existing help and support is limited, however response from employees in the ticketing system or emails, gives a fairly quick response.

References

1. Chaczko, Z. & Alenazy, W. (2014), 'The extended technology acceptance model and the design of the 21 st century classroom', Computer Aided System Engineering (APCASE), 2014 Asia-Pacific Conference on, IEEE, pp. 117-21.

2. Chen, D.R., Chen, M.Y., Huang, T.C. & Hsu, W.P. (2013), 'Developing a mobile learning system in augmented reality context', International Journal of Distributed Sensor Networks, vol. 2013.

3. Dede, C. (2009), 'Immersive interfaces for engagement and learning', science, vol. 323, no. 5910, pp.66-9.

4. Dunleavy, M., Dede, C. & Mitchell, R. (2009), 'Affordances and limitations of immersive participatory augmented reality simulations for teaching and learning', Journal of Science Education and Technology, vol. 18, no. 1, pp. 7-22.

5. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D. & Twombly, X. (2007), 'Vision-based hand pose estimation: A review', Computer Vision and Image Understanding, vol. 108, no. 1, pp. 52-73.

6. Fengjun, G. & Shijie, C. (2010), 'Gesture Recognition Techniques in Handwriting Recognition Application', Frontiers in Handwriting Recognition (ICFHR), 2010 International Conf. on, pp.142-7.

7. Montero, A., Zarraonandia, T., Aedo, I. & Díaz, P. (2013), 'Uses of Augmented Reality for Supporting Educational Presentations', Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on, IEEE, pp. 426-8.

8. Nickel, K. & Stiefelhagen, R. (2007), 'Visual recognition of pointing gestures for human–robot interaction', Image and Vision Computing, vol. 25, no. 12, pp. 1875-84.

9. Prasad, S., Peddoju, S.K. & Ghosh, D. (2013), 'Mobile augmented reality based interactive teaching & learning system with low computation approach', Computational Intelligence in Control and Automation (CICA), 2013 IEEE Symposium on, IEEE, pp. 97-103.

10. Remote Lab at UTS, (2015), <http://www.uts.edu.au/about/faculty-engineering-and-information-technology/what-we-do/facilities-and-services/remote>.

11. Sholtz, P. (2014), How To Make An Augmented Reality Target Shooter Game With OpenCV: Part 3/4, RAYWENDERLICH TUTORIALS FOR DEVELOPERS & GAMERS.SoftKinetic (2014), <http://www.softkinetic.com>.

12. Wong, W. September 8, (2011), 'Electronic Design', vol. 2011, Natural User Interface Employs Sensor Integration, <http://electronicdesign.com/embedded/natural-user-interface-employs-sensor-integration>.