Kinect Wizard Game for Upper Limbs Rehabilitation: Interim Report

X. Li
Newcastle University
Newcastle upon Tyne
United Kingdom
X.Li42@newcastle.ac.uk

ABSTRACT

In this paper, gesture recognition in rehabilitation will be introduced as well as the Kinect Sensor. Although gesture recognition is simplified by Kinect, that could still be a complex problem when the candidates are complicated gestures, and it also depends on the algorithms that to be implemented. This project aims to create a system to record and recognize gestures and to build a game which can potentially help the rehabilitating process.

General Terms

Experimentation

Keywords

Human-Computer Interaction, Kinect Sensor, Upper-limb Rehabilitation, Gesture Recognition

1. INTRODUCTION

With the development of computational devices, human-computer interaction is not just limited to mice and keyboards. Vision-based human-computer interaction is one of the most effective ways to get information (like body movement, gestures, postures, etc.) from people. Because the technology of motion sensor devices (like Kinect) becomes mature and easily accessible, applications using this technology are used in broad areas such as smart home, simulation, rehabilitation and games. Actually, motion capture and recognition has been researched for decades, and the early work has been done with the help of wearable equipment (i.e. instrumented gloves, body suit, etc.) to gather data. The appearance of motion-sensing input devices allows people to be freed from the equipment, and the research interests have transferred to the area of markerless motion recognition. Sensor devices simplify the process of capturing motion data and provide the track the human body movement; however, that is basic of human gesture recognition. Analyzing the raw data from the sensor to recognize the gestures is the main challenge.

Most rehabilitation takes place at hospitals or rehabilitation centers, and that is not always convenience for patients to attend the rehabilitation treatments. In recent years, motion capture technics are used widely in the field of medicine, and it makes home-based rehabilitation possible with the help of proper hardware and well developed software [1]. Traditional rehabilitation is usually supervised by therapists. Therapists teach patients the exercise they need to do and observe the patients' condition to adjust the progress and methods. However, with the help of machines, all these work could be handled automatically. Machines will store and process all the data and choose the suitable approaches for patients accordingly. It provides a cheaper alternative for people who need rehabilitation treatment.

Kinect (figure. 1) is used as input device in this project. It is a popular motion input device in the market. A Kinect has four major components:

- A RGB camera stores three channels of color data in the resolution of 1280 * 960.
- An infrared (depth sensor) emitter emits infrared light beams, and an IR depth sensor reads the beams that reflected back by objects. They are used for measuring the distance of objects.
- A multi-array microphone which consists of four microphones records the audio and locates the direction of the source.
- A 3-axis accelerometer configured for 2 times gravity to determine the orientation of Kinect.

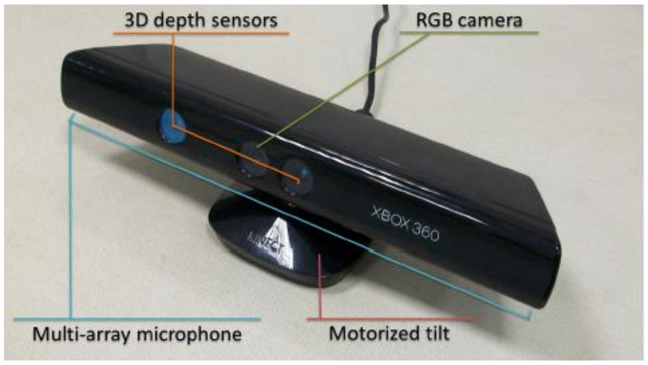

Figure 1. The Microsoft Kinect for Xbox 360 [15]

Except for the data of RGB image and depth image, Microsoft's Kinect Software Development Kits (SDK) also provides Application Programming Interfaces (API) to track the positions of skeleton (figure. 2). The Kinect SDK uses the information from RGB camera and IR sensor to track the 3-D coordinates of 20 joints (30 frames per second) [2]. The data of skeleton is not very accurate especially when the joints overlap, but it is enough for the area of rehabilitation which does not have high requirement for precise. Besides, although Kinect is able to track the position of joints, the task of using the data to recognize gestures is still left for the developers.

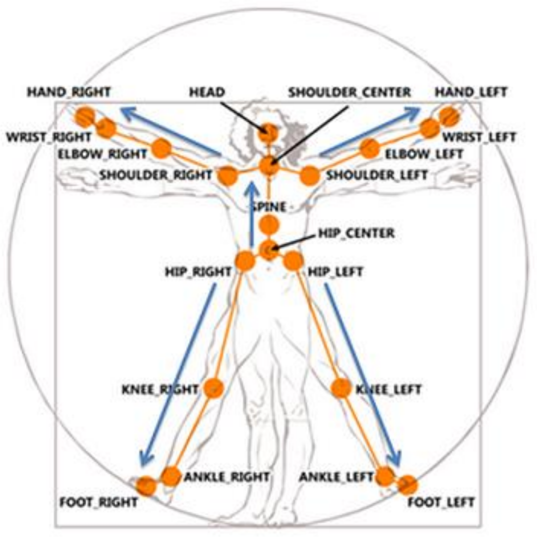

Figure 2. The skeleton joints from the Kinect SDK (Source: https://msdn.microsoft.com/en-us/library/jj131025.aspx)

2. RELATED WORK

2.1 Gesture Recognition in General

In the perspective of machines, a gesture is a sequence of movement generated by a human and recorded as a series of data. The process of resolve the data can be divided into three steps: 1) extracting the human body information from images; 2) tracking the related data from each frame; 3) recognizing the gesture [3]. Kinect accelerates the process of developing a recognition system because the first two steps have already been done for the developers.

The representation of gestures can be roughly divided into two categories: the features in 3-D space and images in 2-D planes. No matter the perspective, 3-D features remain the same, but 3-D models usually require a lot of training and computation. Furthermore, there is a difficult problem of matching the gestures with corresponding models. In comparison, the representation of 2-D images is relatively simple. However, it is only suitable for the situation which the viewpoint is fixed, so that the sensor can capture the gestures from a proper angle to provide a correct 2-D image [4].

To recognize the movement, there are two types of approaches: rule-based approaches and approaches which use template-matching [5]. The first approach is to add constraints to the data regarding to different gestures and to use those constraints to decide if the movements satisfy all the criteria. This approach relies on the parameters and thresholds on joints locations, and users can link gestures to different actions by adjusting the value of thresholds. It is an intuitive way to make the machines recognize human gestures, but that will need lots of assistants from humans by coding those gestures, and the process is error-prone especially when define those complicated gestures. The template-matching approaches will be able to recognize gesture more generally, which means humans will not be focused on making rules for specific gestures. It sees the gesture recognition as a classification problem, and uses the technics of machine learning. In this approach, machines will be able to record and to save the gestures into data sets, and each gesture is labeled to a classifier. When the gestures are presented in front of machines, they evaluate the similarity between the real-time date and each of the gesture patterns in the data set and find the gesture with the highest similarity. Although this approach offers a better solution for gesture recognition, developers have to implement those machine learning algorithms and train the machine with a lot of data [6].

2.2 Recognition with Kinect

There is fairly large amount of work has been done in the area of human body movement recognition. Kinect does not have much support for hands tracking. The positions of hands cannot always be tracked accurately when the hands overlap with the other parts of human body especially in the situation like folding hands or both hands come across each other. Because of that, some works are focused on tracking the movements of hands and detection of hands gestures [7] [8].

Besides, some works are about detecting postures [9]. These works only recognize static gesture instead of dynamic movement of human body. Although postures are static, they can be used as key frames of a gesture and be stored in decision trees or decision forest to achieve the functionality of recognizing movement [10]. Some works propose rule-based approach like the Flexible Action and Articulated Skeleton Toolkit (FAAST) [11] and Full Body Interaction (FUBI) Framework [12]. As mentioned above, a lot of effort is required for defining and testing the rules that are set to gestures. Additionally, they are not suitable for the gestures which need to be recognized in this project (figure. 3) since many of them are barely or very difficult to be distinguished by only constraints.

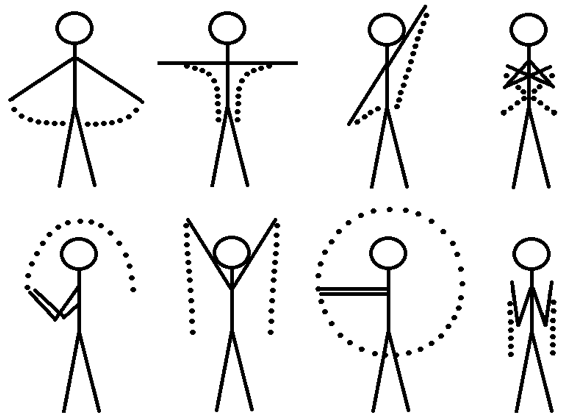

Figure 3. Part of the gestures need to be recognized

Template-matching approach is also applied in many works [9] proposed a comparison of human gesture recognition using data mining classification method in video streaming. They uses positions of specific joints in human body to detect the gesture patterns of sit down, lay down and stand. Various methods of data mining classifier (e.g. BPNN, SVM, decision tree, and naive Bayes) are used to find out the most optimal one.

Dynamic time warping (DTW) is a technique to find an optimal alignment between two given sequences which are different in speed and time, and it is widely used in lots of Template-matching approaches. EasyGR (Easy Gesture Recognition) is a tool to help reduce the human effort involved in gesture recognition, and it is based on machine learning algorithm [6]. It is similar to Kinect Space [19] which also uses DTW to make machines automatically record and recognize gestures.

Another work that uses DTW is done by Su et al [1]. Except for implementing the DTW algorithm, they use fuzzy logic to store the result in a range of 0 to 1 instead of using traditional logic to set the value (true of false) by comparing trajectory and speed.

2.3 $1 Unistroke Recognizer

The $1 Unistroke Recognizer [13] is a 2-D single-stroke recognizer which can recognize mouse-based gestures using Golden Section Search (originally) and Protractor (a faster solution). In the term of machine learning, $1 is a geometric template matcher with a Euclidean scoring function. It only needs very few templates comparing with huge datasets in typical machine learning method to perform relative accurate recognition. Besides, the core part of this approach is not complicated; on the contrary, it is easy to implement and deploy.

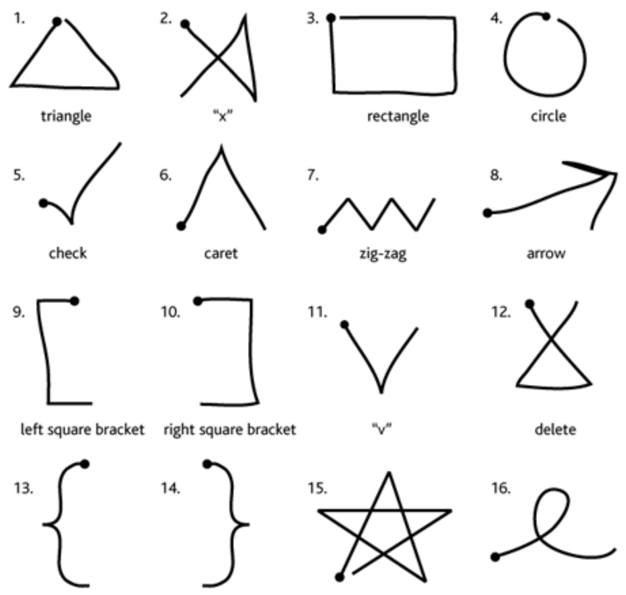

Figure 4. The tested gestures in $1 [13]

$1 uses a simple four-step algorithm:

1. Resample the data of points so that a gesture with a different size of M points is replaced by a gesture in the same shape but with a fixed number of points N.
2. Rotate the gesture to make sure the first point is on the line which the value of y of the centroid point is 0.
3. Scale the gesture to a reference square and translate the gesture so that it is centroid point is on (0, 0).
4. Find the optimal angle and get the final score.

These four steps are applied to the raw points which are the candidates and to the templates which are previously recorded. Then it uses Golden Section Search on the processed data to find out the optimal angular alignment of each point to get a final score of matching.

The performance of gesture recognition for this approach is fairly well according to the test cases in the original paper. Comparing with DTW and Rubine classifier, it has the similar accuracy in recognizing gestures with DTW and the fastest speed among the three algorithms. However, there are some limitations existing. First, the results can only show the closest match in one 2-D Euclidean space. It is not enough for recognize complex 3-D human gesture. For example, waving the hands for 90 degrees with different directions will all generate a track of quarter-circle; however, to observe the movement from the same perspective will result different tracks from a line to a quarter-circle. Besides, the original algorithm cannot distinguish gestures which are different in orientations, locations and ratios in order to provide tolerance to gesture variation. In human gesture recognition, orientation, location and ratio all need to be taken into consideration. Especially in rehabilitation treatments, different gestures even with slight difference in those attributes are designed to exercise different part of the body. At last, there is no time involved in $1 algorithm; thus, gestures with different speed will be treated as the same.

3. PROJECT APPROACH

In this article, a relative easy approach which is inspired by $1 Unistroke Recognizer is proposed. It combines both rule-based approach and template-matching approach and uses tracked position of hands and other relative joints as data source. The purpose of building this system is to recognize a series of gestures for upper-limb rehabilitation provided by the medical school. It also considers the possibility of the future expansion of the gesture set. Therefore, it will be aimed to develop a system which can generally recognize arbitrary gestures without any particular rules.

After the completion of the gesture recognition system, a wizard game will be built using Unity to help patients who need upper-limb rehabilitation. Recently, using commercial video game as rehabilitation tools has gained a lot interest, and some study shows games are successful for rehabilitation following stroke and spinal cord injury [14]. By playing a game while doing the rehabilitation exercise, the treatment process could potentially be more effective since patients are also entertaining during the process, and it could lead to better recovery.

3.1 Gesture Recognition System

The movement of hands needs to be tracked instead of the movement of the mouse in a mouse-based recognition system. The shapes consist of all the points that generated by hands will be captured by Kinect and be used for recognition. The challenge of using hands to generate the shapes is that there is no button that can be clicked or released like a mouse to inform the machine to start or to stop recording data. This is one main difficulty in the design of an effective recognition system while another difficulty is that same gestures may be varying between individuals [14]. Kinect V2 has added two joints (one for the thumb and another one for the rest of the finger) on the hands so that it is possible to detect grip and release, and that could be a potential replacement for the mouse button. However, the hardware that is used in this project is Kinect V1, and it does not have the extra joints on the hands. As a result, an alternative approach is used in this project.

The solution that is used in the system (at least for now) is to store the hands' positions from the latest N (100 is found suitable) frames, and when the hands stop moving for M frames, a gesture will be considered finished. There are some drawbacks using this approach. Depending on the lengths of gestures, users need to adjust the speed of performing a gesture. For example, a template with more than N-M frames needs to be performed faster than normal; otherwise, some positions at the beginning of the position list will be replaced by the points recorded later. That will lead to a different final shape, and the gesture may not be recognized correctly. Another drawback is that users have to hold their moves after finishing the gestures for a little while. If extra move occurs before the machine considers the move is finished, the gestures may need to be performed again.

The movements of hands for many gestures are in similar shapes but different rotations and positions. The original algorithm rotates all the gestures to make sure the first point of the data is on the line y = 0, and also it translates the gestures to the scale space in which the point (0, 0) is the centroid point. Because of that, those gestures with different rotation cannot be distinguished. In order to recognize those gestures differ in rotations and positions, the step of the rotating the points is abandoned so that the rotation of each gesture is left as-is when the machine records the data. Consequently, rotation will be distinguished, but the gestures need to be performed more restrict in order to be recognized. The step of translating the points is preserved. Because the same gestures cannot be guaranteed to be performed at the same positions in the Kinect space, removing the translation cannot solve the problem. Instead, a step of checking constraints of gestures is added before the final step of matching templates and calculating the scores. The constraints which are added to gestures are relative directions between hands and head, hands and elbow, etc. Moreover, the efficiency of the algorithm will be improved by adding constraints to gestures because only those gestures which pass the constraints checking will enter the final phase of calculating score and matching templates.

Primarily, x and y value are used for the gestures recognition, which means the shapes project on the x-y plane will be used to match the templates. However, human gestures are performed in a 3-D space, and not all of them can be recognized only by checking the shapes on the x-y plane. In order to match 3-D human gestures, the position data (i.e. vector 3s with the values of x, y and z) is stored in three copies, one for the x-y plane, another for the x-z plane and the last for the z-y plane. These three copies of data are basically the shapes of gestures that are projected to those three planes. When a gesture is recorded, a value which indicates the primary plane used for the gesture will be input by users. When the final score is calculated for each gesture, the relative data for the specific plane will be used. The limitation of this approach is the accuracy of Kinect.

The original system struggles to get an effective score as it mentions in the paper. Therefore, for the gestures which will form lines, an extra step is added to decide if the distribution of the position data is closed to lines or not, and if a gesture is decided to be a line, it will be treated differently. Correlation coefficient is calculated to decide the distribution of points, and it is a float value ranged from -1 to 1. The negative sign or positive sign represents for the direction of the distribution (values increase or decrease). The more the coefficient value is close to -1 or 1, the more the distribution is close to a line [17]. The following formula is used to calculate the coefficient:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\ \sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

When a line is generated by the gesture, the system will not run through the process of calculating scores and matching templates. Instead, it will first check if the templates are lines as well. For those templates which are lines, the angles between them and y = 0 are used to compare with the angle of the gesture. The template with the closet angle of the gesture will be the one which matches the best. Other gestures will go through the Golden Section Search algorithm [18] to compare with the templates to get a score. First, to get the average distance between corresponding points of candidate C and stored template T using the formula:

$$d_i = \frac{\sum_{k=1}^{N} \sqrt{\left(C[k]_x - T_i[k]_x\right)^2 + \left(C[k]_y - T_i[k]_y\right)^2}}{N}$$

Then the distance is converted to range of 0 to 1 using the length of a side of the reference square to which all the gestures were scaled to, and the equation is:

$$score = 1 - \frac{d_i^*}{\frac{1}{2}\sqrt{size^2 + size^2}}$$

To sum up, the workflow of the system is listed below:

- Recording the gestures that need to be recognized which contains basic information of the gestures (e.g. name, time, length, etc.), the positions of both hands in time manner, the constraints of the gestures.
- Capturing and storing the latest hands' positions into a list with fixed size. When the finished sign of a gesture is recognized, the current data in the list is going to be used to recognize the gesture.
- Calculating the correlation coefficient to decide if it is a shape of a line. If it is, filtering the gestures which are not lines, verse vice.
- Checking the constraints of the data, and only comparing it with those templates which have the same constraints.
- If the data is a line, calculating the angle of the line with the line y = 0 to decide which gesture it matches. Otherwise, calculating the score using Golden Section Search and looking for the highest score among all the templates.
- At the end, either the system finds a template which satisfies the minimum requirement of matching, or the gesture cannot be recognized. No matter the result, the list will be cleared to prepare for a new gesture.

3.2 Using Unity

In this project, Unity personal version is used for the purpose of making a rehabilitation PC game prototype. Unity is a cross-platform engine which supports development for PC, consoles, mobile devices and websites. C# is the programming language that is used for writing the scripts with the help of Unity APIs and some Windows system libraries. Unity does not support .NET framework 4.0, so some of the APIs in Kinect SDK does not work. Therefore, the recognition system is also built in Unity to avoid problems that could possibly occur when integrating the system into the game engine. It is built on the Microsoft official Unity package [20] since the package provides some examples to show how to access the data from the Kinect sensor in unity.

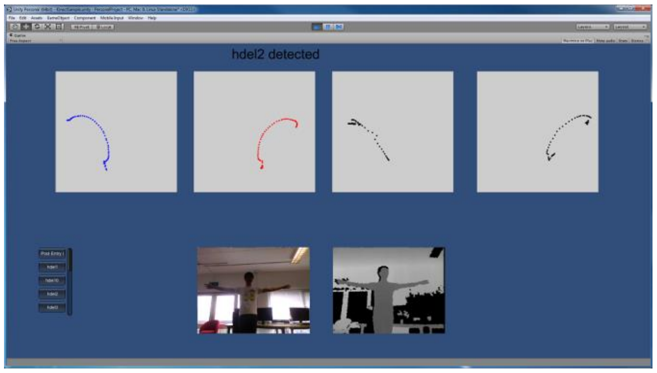

Figure 5. UI of the recognition system in unity

4. PROJECT PLAN

So far, the gesture recognition system is running smoothly, and it can recognize 12 out of 18 required gestures (mirror gestures are not taken into account) with acceptable accuracy.

Three major components of the system have been implemented: a recorder which records the gesture data into XML files; a detector which polls the real-time skeleton data from Kinect sensor every frame and constantly calls the recognizing function using the stored data; and the algorithm which loads the data from XML files and compares that with the data the detector stored to recognize the gestures.

The recognition system has simple UI (figure 5), which consists of two squares showing the track of recorded templates, two squares showing the latest track of both hands, and two squares showing the color image and depth image respectively. The templates to be displayed on the screen can be selected on the list on the bottom left corner, and when a gesture is detected, the name will be printed on the top of the screen. The purpose of visualizing the data on the screen is for the convenience of testing and debugging.

The aim of the gesture recognition system is to correctly recognize all the gestures in the provided videos. However, some of the gestures cannot to be recognized by using the current system. For example, gestures that flip or rotate the wrists cannot be tracked only using the data of skeleton joints because that kind of movement is too slight for the sensor to track. Besides, gestures like crossing two hands before the chest cannot be tracked using the skeleton data as well due to too much overlaps occur during the movement. The problem of how to track the hands accurately using Kinect itself could be a project [7] [8], so those gestures will be ignored at least before finishing building the game. If there is enough time, more research will be done in the field of hands gesture recognition to find a suitable way to track the hands correctly.

The next goal of the system is to add support for the gestures that cannot be recognized just using the x and y value of the points for example moving hands back and forward. Recognizing these gestures will also require the usage of the z value. An approach needs to be found to use the z value effectively to deal with those gestures. After that, the accuracy of the system will be tested.

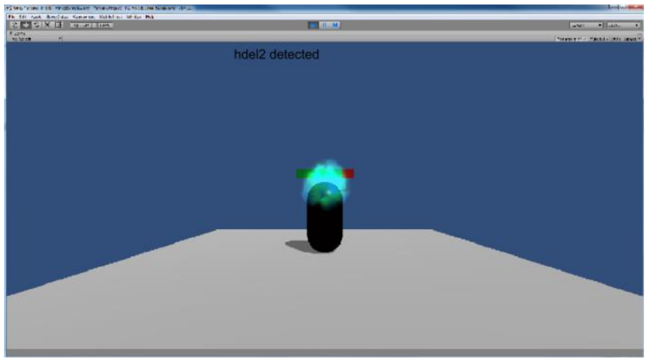

Figure 6. A blue fireball is shooting to the target from the camera's position when a gesture is detected

The wizard game is being built, and some classes have been added to the project. Functions like shooting projectiles, destroying and respawning monster have been implemented. The idea of the game is inspired by a game called Clicker Heroes [16]. The game will be presented in a first person view, and it only has a single scene with the player and monsters (bosses). When the player performs a gesture as requested, the character will cast a spell to the monsters with corresponding levels and types. To defeat monsters and to advance to higher levels, players will need to follow the instruction to do the exercise. With the monsters getting stronger, players will need to "ascend" the game world to get bonus and to improve the damage so that they could beat strong monster. This design encourages players to practice more basic excise repeatedly. The interesting part of the game is to spend the gold that is earned through beating monsters and find the most cost-efficient way to increase the damage.

5. REFERENCES

[1] C. Su, C. Chiang & J.Huang, "Kinect-enabled home-based rehabilitation system using Dynamic Time Warping and fuzzy logic," *Applied Soft Computing Journal.*, vol. 22, pp. 652-666, Sep. 2014.

[2] C. Kam Lai, P. Konrad & P.Ishwar, "A gesture-driven computer interface using Kinect," *2012 Ieee Southwest Symp. Image Analysis Interpretation*, pp. 185 – 188, Apr. 2012.

[3] P. Turaga, R. Chellappa, Subrahmanian. V.S & O.Udrea, "Machine Recognition of Human Activities: A Survey," *Ieee Trans. Circuits Syst. for Video Technology*, vol. 18, no. 11, pp. 1473 – 1488, Nov. 2008.

[4] G. Xu, Y. Cao &, "Action Recogn ition and Activ ity Understanding: A Review," *J. Image Graphics*, vol. 14, no. 2, pp. 189 – 195, Feb. 2009.

[5] J. Aggarwal & Q. Cai, "Human motion analysis: A review,"*Comput. Vision Image Understanding*, vol. 73, no. 3, pp. 428 – 440, Mar. 1999.

[6] R. Ibanez, A. Soria, A. Teyseyre & Campo. M "Easy gesture recognition for Kinect," *Advances Eng. Software*, vol. 76, pp. 171 – 180, Oct. 2014.

[7] Z. Ren, J. Yuan, J. Meng & Z. Zhang, "Robust Part-Based Hand Gesture Recognition Using Kinect Sensor," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1110 – 1120, Aug. 2013.

[8] Y. Li, "Hand gesture recognition using Kinect," *2012 Ieee Int. Conf. Comput. Science Automation Eng.*, pp. 196 – 199, June. 2012.

[9] O. Patsadu, C. Nukoolkit & B.Watanapa, "Human gesture recognition using Kinect camera," *2012 Ninth Int. Conf. Comput. Science Software Eng.*, pp. 28 – 32, May. 2012.

[10] L. Miranda, T. Vieira, D.Martinez, T. Lewiner, A.W. Vieira & M. F. M. Campos "Real-Time Gesture Recognition from Depth Data through Key Poses Learning and Decision Forests," *2012 25th Sibgrapi Conf. Graphics, Patterns Images*, pp. 268 – 275, Aug. 2012.

[11] E. Suma, D. Krum, B.Lange, S. Koenig, A. Rizzo & M. Bolas, "FAAST: The Flexible Action and Articulated Skeleton Toolkit," *2011 IEEE Virtual Reality Conf.*, pp. 247 – 248, Mar. 2012.

[12] F. Kistler, B. Endrass, I.Damian, C. Dang & E. Andre, "Natural interaction with culturally adaptive virtual characters," *J. Multimodal User Interfaces*, vol. 6, no. 1-2, pp. 39 – 47, Jul. 2012.

[13] J.O. Wobbrock, A.D. Wilson & Y.Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," *User Interface Software Technology: Proc. 20th Annual Acm Symp.*, pp. 159 – 168, 2007.

[14] B. Lange, Change. C, Suma. E, Newman. B, Rizzo. A, Bolas. M, "Development and evaluation of low cost game-based balance rehabilitation tool using the Microsoft Kinect sensor," *Annual Int. Conf. IEEE Eng. Medicine Biology Society. Conference, 2011*, vol. 2011, pp. 1831 – 1834, 2011.

[15] L. Gallo, A.P. Placitelli & M.Ciampi, "Controller-free exploration of medical image data: Experiencing the Kinect,"*2011 24th Int. Symp. Comput.-based Medical Syst.*, vol. 2011, pp. 1 – 6, June. 2011.

[16] Playsaurus, 2014. *Clicker Heroes*. Available at: <https://www.clickerheroes.com/> [Accessed: June 24, 2015].

[17] Mathsisfun, 2014. *Correlation*. Available at: <https://www.mathsisfun.com/data/correlation.html> [Accessed: June 24, 2015].

[18] Gerald, C.F. & Wheatley, P.O., 2005. *The Golden Section Search Method* . Available at: <http://homepages.math.uic.edu/~jan/mcs471/Lec9/gss.pdf> [Accessed: June 24, 2015].

[19] Wölfel, M., 2012. *Kinectspace - Training, Analyzing and Recognizing 3D Gestures*. Available at: <https://code.google.com/p/kineticspace/> [Accessed: June 24, 2015].

[20] Microsoft, 2013. *Microsoft Kinect - Microsoft SDK - Unity 3D*. Available at: <http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Microsoft_SDK> [Accessed: June 24, 2015].