Kinect Wizard Game for Upper Limbs Rehabilitation: Interim Report

Xuzhe Li
Newcastle University
Newcastle upon Tyne
United Kingdom
07413575723
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, Unity

1. INTRODUCTION
The development of computational devices expands the ways of human-computer interaction which is not just limited to mice and keyboards in nowadays. 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 the basic of human gesture recognition. The main challenge is analyzing the raw data from the sensor to recognize the gestures.

Most rehabilitation takes place at hospitals or rehabilitation centers, and that is not always convenient 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.

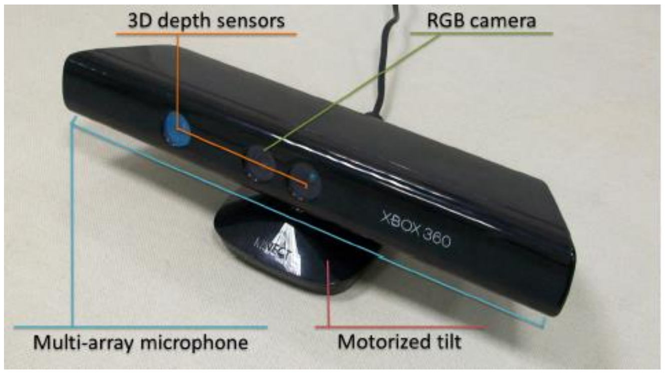

Figure 1. The Microsoft Kinect for Xbox 360 [15]

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.

The rest of paper is organized as follow: in the related work section, general research on gesture recognition will be reviewed, and some specific cases of gesture recognition system will be discussed in detail; an adaptive gesture recognition system is proposed in the project approach section; at last, the progress and the plan of the project are described in the project plan section.

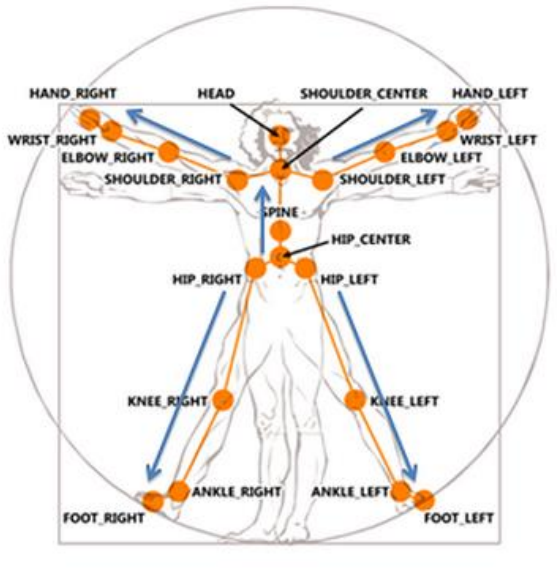

Figure 2. The skeleton joints from the Kinect SDK (Source: https://msdn.microsoft.com/en-us/library/jj131025.aspx)

2. RELATED WORK

2.1 Gesture Recognition in General

For the perspective of machines, a gesture is a sequence of movement generated by a human and recorded as a series of data. The challenge of gesture recognition is if a system can be built to recognize the data automatically. As Turaga et al [3] states, the process of resolving 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. As they point out, most of the raw information from images is irrelevant to identify gestures, such as the background, the color, and etc. How to extract the useful information from images and track it correctly itself is a challenge of recognizing gestures. However, this job has already been done by Kinect as well as tracking the data every frame, so the focus of this project will be how to use that data to recognize gestures.

The representation of gestures can be roughly divided into two categories: the features in 3-D space and images in 2-D planes. The advantage of using 3-D features to represent gestures is that no matter the point of view, 3-D features remain the same. However, 3-D models with massive amount of parameters usually require a lot of training and computation. Furthermore, matching the gestures with corresponding models is also a difficult problem, and the implementation may require a lot of knowledge not only in the field of computing but also in math. In comparison, the representation of 2-D images is relatively simple, but the drawback is only suitable for the situations in which the viewpoints are fixed, so that the sensor can capture the gestures from proper angles to provide a correct 2-D image [4]. Because of the complex situations in real life, applications using 2-D planes to represent gestures will be limited in real life due to the complex environment. The positions of humans may change constantly during the process of observation, and that leads to the images which are captured from the same view point become unusable.

To recognize the movement from the gesture data, there are mainly two types of approaches which are used commonly: rule-based approaches and template-matching approaches [5]. The former approach is to use constraints (relationships among joints, absolute positions of joints, etc.) to describe specific gestures. Constraints are stored as data regarding to different gestures, and the data is used to check if the movements satisfy the requirement. 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 in exchange of simplicity, it needs a large amount of assistants from humans by define those gestures, and the process is error-prone especially when define those complicated gestures. Besides, people who define the gestures may need a quit amount of knowledge and experience from the field. The template-matching approaches are capable of recognizing gestures more generally, which means humans will not be focused on making rules for specific gestures, but on the algorithms of matching gestures. It sees the gesture recognition as a classification problem, and uses the technics of machine learning, data mining, and etc. By using these approaches, machines will be able to record and to save the gestures into data sets, and each gesture could be labeled to a classifier to match the movement. 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 out the classifier 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, and some works are discussed below.

2.2.1 Hands Detection

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 moving across each other. Because of that, some works are focused on tracking the movements of hands and detection of hands gestures.

Zhou et al [7] use the Kinect SDK to locate the rough position of the hand and add a pair of depth values as thresholds to get the region of the hand. They also use a black wrist belt to help them to get more accurate results by using RANSAC (Random sample consensus) to filter the image data after detecting the black color pixels of the belt. After removing irrelevant information but the shape of hands on the frame images, they represent the data as a curve in time manner which shows the distance between each point on the outline of the hand and the centroid point.

Li [8] uses a similar way to estimate the rough region of the hands by manually setting depth thresholds, but his system can track two hands simultaneously. He uses 2-D images on which the hands are projected to do advanced analysis. To be more detailed, he applies the K-means clustering algorithm in order to partition the points into two groups. The distance of centroid points of the two clusters is used to decide if there are two hands or one. If the distance is smaller than a pre-defined minimum value, then there

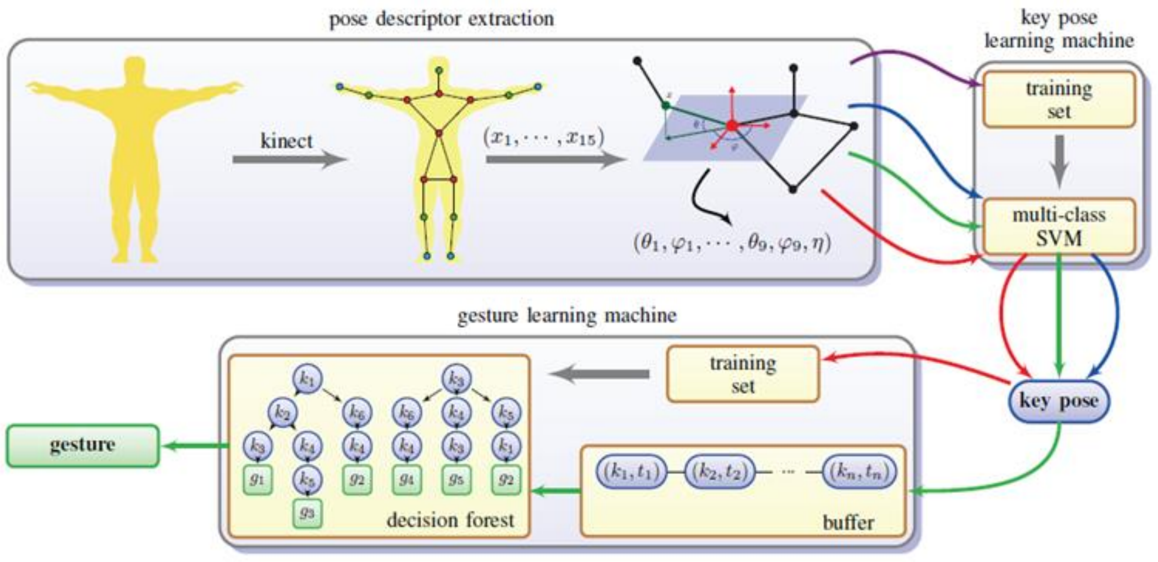

Figure 3.Work flow of the recognition system proposed by Miranda et al

is only one hand, and the two clusters will be merged to represent that hand; otherwise, each cluster will represent one hand. After partitioning the clusters, he uses the Graham Scan Algorithm and a modified Moore-Neighbor Tracing Algorithm to compute and detect the outline of the hands. Above approaches can track the hands effectively; however, because of the depth thresholds they set in order to detect the hands, the movement users will be limited. If the hands move out of the depth range (like punching, swing hands forward and backward), they cannot be detected.

2.2.2 Machine Learning and Data Mining

Machine learning and Data mining are proved to be suitable for gesture recognition, and works below using these technics to detect static postures instead of dynamic body movement, and those postures are used as key frames to recognize completed gestures. Patsadu et al [9] proposed an approach using data mining classification method in video stream. They get the data of 3-D vectors for the 20 body joints from Kinect sensor and use various method of data mining classification (BPNN, SVN, decision tree, naive Bayes) to compare the performance among them. An open-source data mining tool named KNIME is used to achieve those classification methods. They tested sit down, stand, and lie down three gestures and found all the methods are effective.

Miranda et al [10] uses a pose descriptor to represent human body poses. The descriptor uses joint angles representation for the nodes of the skeleton. This way of representing the joints is more robust than the original frame skeleton data provided by the Kinect sensor. It provides invariance to sensor orientation and removes the redundant data but the information for the classification of key poses. Training sets are used to build multiple SVM learning machines to identify key poses in real-time. After the learning machine is trained to recognize the key poses, users can define gestures as sequences of key poses, and a decision forest will be built so that the poses are connected to improve the efficiency of searching gestures. It can even take the time/speed constraints into consideration when recognizing gestures.

These approaches are robust and perform very well on recognizing specific gestures, but knowledge of machine learning and data mining is required in order to design and implement a robust system. Besides, the training process may take a long period of time, and some assistants may be required from others.

2.2.3 Rule-based Approach

Flexible Action and Articulated Skeleton Toolkit (FAAST) [11] and Full Body Interaction (FUBI) Framework [12] recognize gestures through detecting sequences of postures as well, but they use simple rule-based approach to recognize postures. They describe gestures using the relations among specific joints as well as the time limit and directions, and then they pre-store them in configuration files. In run-time, the joints' positions are used to compare with the configuration every frame. The speed and directions of joints are also calculated constantly to combine with the constraints of postures to recognize specific gestures. When the joints data meets the requirement of both time constraints and specific postures, the gesture will transit to next phase until a full gesture sequence is completed. As mentioned in the previous section, a lot of effort, experience, and knowledge are required for defining and testing the rules that are set to gestures. Additionally, they can effectively recognize gestures like waving hands, but they are not suitable for the complex gestures which need to be recognized in this project (figure 4) since many of them are very difficult to be distinguished by only constraints.

2.2.4 DTW and HMM

Dynamic time warping (DTW) is a technique of finding an optimal alignment between two given sequences which are different in speed and time. Hidden Markov model (HMM) models data in a statistical Markov model by assuming that is a Markov process with hidden state. They are widely used in many Template-matching approaches.

EasyGR (Easy Gesture Recognition) [6] is a tool using DTW to help to reduce the human effort involved in gesture recognition. In terms of matching templates, it uses a different algorithm but works similarly with the system in this project. Template gestures are first performed by humans and recorded by the machine, and

they are sequences of joints data. Then the data will be moved to the center of the detection field and to be normalized. Thresholds then will be generated by comparing the two most different samples of same gestures. Thus, when users perform the same gestures in front of the camera, the system will store the joints' positions in buffer data and use DTW algorithm (or HMM alternatively) to get values for each template, and then the best match will be finally found out.
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. They designed a two-stage fuzzy logic system, and it contains two ANFIS (adaptive neuro-fuzzy inference system) based evaluators, which are implemented by using Matlab, for trajectory and speed, and a Mamdani fuzzy evaluator for the overall performance of the gestures. Calculation of the overall performance takes consideration of the results from all three evaluators. The fuzzy logic system not only recognizes the gesture more precisely but also helps therapists to observe the patients' performance with more detailed information.

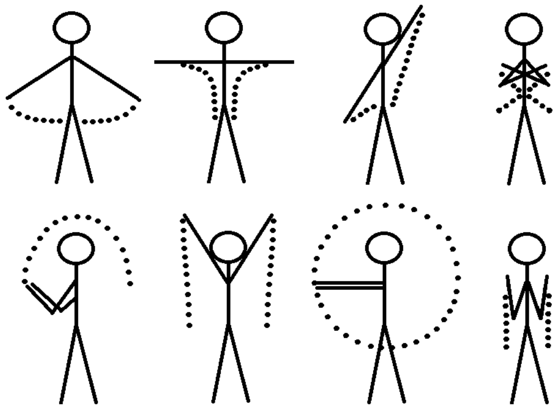

Figure 4. Part of the gestures need to be recognized

2.3 $1 Unistroke Recognizer

$1 Unistroke Recognizer is separated in one sub-section and is discussed in more detail because it is the core algorithm used in this project in terms of template-matching. 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.

$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.

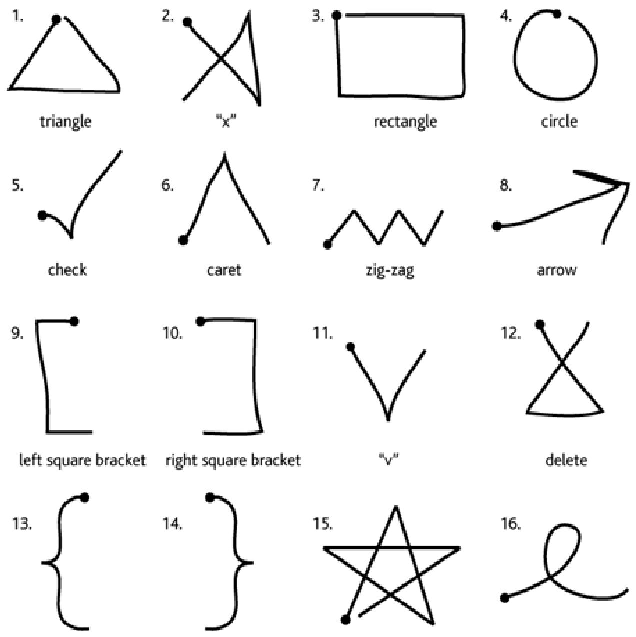

Figure 5. The tested gestures in $1 [13]

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 algorithm 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

In order to adapt the mouse-based recognition system to a body movement recognition system, the first step is to change the way of input. The movement of hands needs to be tracked instead of the movement of the. The points that generated by hands movement will be captured by Kinect sensor and be used for recognition (other joints are also stored for adding constraints). 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]. The second difficulty will not be a challenge in this project since the algorithm will normalize the gesture data, so as long as the gestures performed by different people are the same ones, the data will be similar enough to be recognized as the same gestures. However, the first difficulty still needs to be addressed by finding alternative for buttons or using a different way to interact with the machine.

Kinect V2 has added two joints (one for the thumb and another one for the rest of the fingers) on the hands so that it is possible to detect grip and release, and that could be a potential replacement for the mouse buttons. 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 proposed. 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 (the distance between latest two points is smaller than a predefine value) 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 restrictedly 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, and removing the translation cannot solve the problem. Instead, a step of checking constraints of the 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 of 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, or maybe the scores from multiple planes will be combined together. This is just an idea of how to use the data from all three planes, and experiment will be carried out to find out a suitable solution.

As mentioned by Wobbrock et al [13], the original system struggles to get an effective score when 1-D gestures are performed or stored as templates. Therefore, for dealing with gestures which will form lines, an extra step is added. First, the distribution of the positions will be checked to decide if it is closed to a line or not, and if a gesture will be processed differently if it is decided to be a line. To decide the distribution of points, correlation coefficient which is widely used to measure the degree of linear dependence between two variables is calculated. The following formula is used to calculate the coefficient:

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

As the formula shows, if the distribution of the point is on the line Y = 0 or X = 0, the denominator will be 0, and that will make the formula invalid. Hence, before the coefficient is calculated, all the points will be rotated until the angle between the line formed by the first point and the last point and the line Y = 0 is 45 degree. This step makes sure the coefficient will be a float value ranged from 0 to 1. The more the coefficient value is closed to1, the more the distribution is close to a line [17], and a minimum valued is set for deciding lines. When a line is detected, the system will not run through the process of calculating scores. Instead, it will check if the templates are lines as well. Templates which are not lines will be filtered, and the templates left will be compared with the gesture. The angles between templates and Y = 0 and the angle from the gesture are used for comparison. The template which has the closest angle to gesture's angle will be the one which matches the best.

Other gestures which are not lines will go through the Golden Section Search algorithm [18] to compare the hands positions with the points in templates to get a score. To calculate the score, first, the average distance between corresponding points of candidate C and stored template T will be get 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 (d) and the length of a side of the reference square to which all the gestures were scaled to (size) will be used in the equation below to get a final score which is range from 0 to 1.

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

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.
- Loading the templates from files, packaging them and storing in a list, so that they are prepared to compare the hands' positions data.
- 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 will be packaged in the same way as the templates, then it is going to be applied to the algorithm.
- 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 meet the same requirement.
- 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.
- In the end, either the system finds a template which satisfies the minimum requirement for recognition, or the gesture cannot be recognized. No matter what the result is, the list that stores the real-time hands' positions 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 [19] since the package provides some examples to show how to access the data from the Kinect sensor in Unity.

For the recognition system, Unity is mainly used as a tool to visualize the data. That is achieved by using the functions in the Texture2D class. Normalized 3-D vectors which store the positions of hands are passed to the draw functions, and then the functions call the SetPixel function for each pixel to draw the data. So far, only the points projected on X-Y plane are displayed.

4. PROJECT PLAN

So far, the gesture recognition system runs 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 6), which consists of two rectangles showing the track of recorded templates, two rectangles showing the latest track of both hands, and two rectangles 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 effectively 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 detection to find out a suitable way to track the hands correctly.

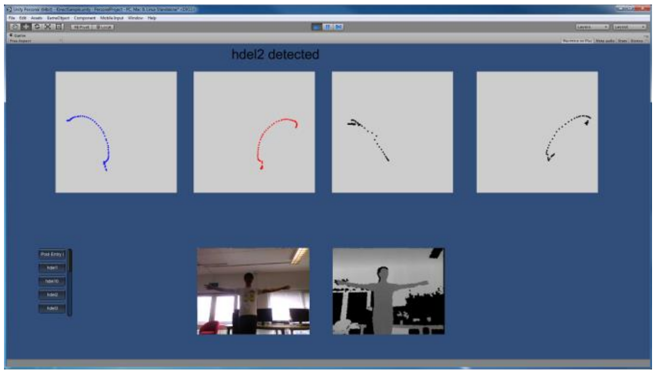

Figure 6. UI of the recognition system in unity

The next goal of the system is to add support for the gestures that cannot be recognized by just using x and y value of the points, such as moving hands back and forward. Recognizing these gestures will also require the proper usage of 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.

The wizard game is being built, and some classes have been created and integrated 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.

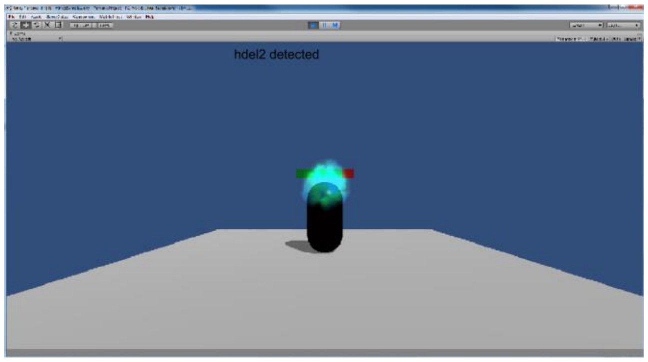

Figure 7. A blue fireball is shooting to the target from the camera's position when a gesture is detected

Overall, there are four major jobs remaining to be done in this project: integrating the unused z value into the recognition system; testing the efficiency of the recognition system; building the game; tracking the hands' movement accurately. With 8 weeks remaining, the jobs are scheduled as follow: first three weeks will be spent on completing the recognition system by adding the usage of depth of the joints and finishing some trivial tasks. After completion, the system will be tested to see the performance, and the process may last for one week. If the progress sticks to the plan, the last four weeks will be spent on researching and implementing how to track hands properly, preparing for demonstration and writing the dissertation. Lastly, the game will be building simultaneously while doing other jobs.

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] 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].