Kinect Wizard Game for Upper Limbs Rehabilitation

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 for rehabilitation 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 more effective ways to get information (body movement, gestures, postures, etc.) from people. Because the technology of motion sensor devices such as Kinect becomes mature and easily accessible, applications using this technology are developed in broad areas such as smart home, simulation, rehabilitation and games. Actually, the research of motion capture and recognition has been doing 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. Sensors simplify the process of capturing the motion data and provide the function of tracking the human body movement; however, that is just the basic of human gesture recognition. The main challenge is analyzing the raw data captured by the sensor, and using proper methods to extract the correct 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. Traditionally, therapists will show the patients the exercise they need to do and observe the patients' condition to adjust the progress and methods, but 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 could potentially provide 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.

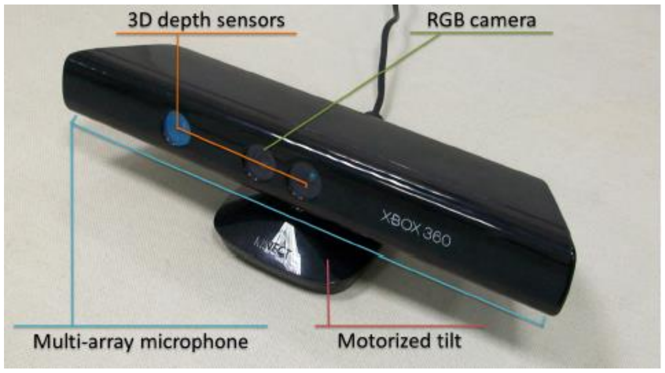


Figure 1. The Microsoft Kinect for Xbox 360 [15]

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 precision. Although Kinect is able to track the position of joints and output the data, the task of using the data to recognize gestures is still left for the developers.

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 the recent 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.

The rest of paper is organized as follow: in the section of background, general research on gesture recognition will be reviewed as well as the concept of serious game, and some specific cases of gesture recognition system will be discussed in detail, especially the $1 recognizer and DTW algorithm; the design and implementation of the gesture recognition system and the rehabilitation game is introduced and explained in section 4; then, the test results of the recognition system is shown in section 5 along with the explanation of why, and how the test is executed; at last, the dissertation is summarized in conclusion, and it is discussed how the project can be taken forward.

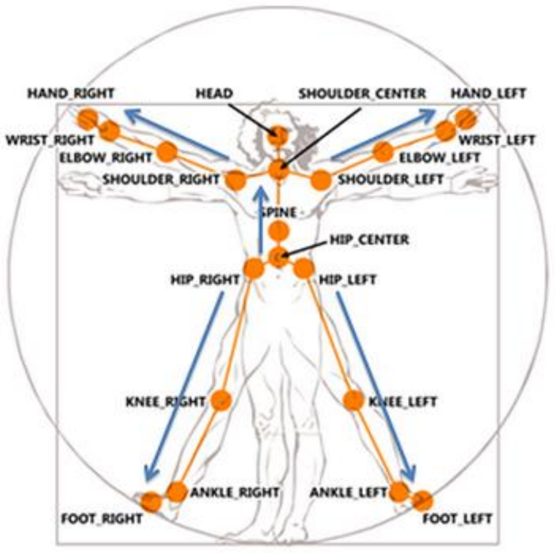


Figure 2. The skeleton joints from the Kinect SDK (Source: https://msdn.microsoft.com/en-us/library/jj131025.aspx)

2. BACKGROUND

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 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 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 6) since many of them are very difficult to be distinguished by only constraints.

2.2.3 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.

2.3 Dynamic Time Warping Algorithm

Dynamic time warping (DTW) is a dynamic programming technique, and it was originally applied to the field in speech recognition, and it is one of the most popular algorithms in gesture

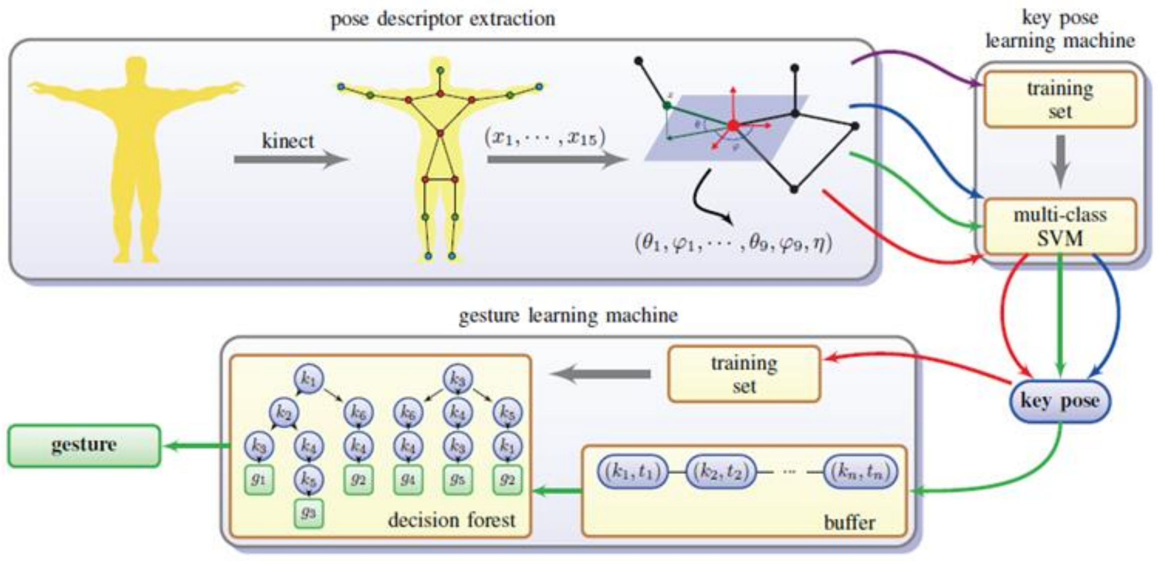

pose descriptor extraction
kinect
$(x_1, \cdots, x_{15})$
$(\theta_1, \varphi_1, \cdots, \theta_9, \varphi_9, \eta)$
key pose learning machine
training set
multi-class SVM
gesture learning machine
training set
key pose
gesture
k_1 k_2 k_3 k_4 k_5 k_6
g_1 g_2 g_3 g_4 g_5
decision forest
(k_1, t_1) (k_2, t_2) ... (k_n, t_n)
buffer

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

recognition. The objective of DTW is to compare two data series, $X := (x_1, x_2, \dots, x_N)$ of length $N \in \mathbb{R}$ and $Y := (y_1, y_2, \dots, y_M)$ of length $M \in \mathbb{R}$ which are not only differ in amplitudes, but also in time progression, and to calculate the warping distances between the two.
When the algorithm is applied to gesture recognition, it calculates warping paths and distances between a test sequence and all the template sequences. Assuming X is the sequence of the gesture data, and Y is one of the template sequences. In order to get the warping path, a local distance matrix D $\in R_{N\times M}$ which stores the distances between each point in two sequences will be needed. To be more specific, each element of D is the value of distance between X(n) andY(m). Euclidean distance is typically used (this project also chooses to use Euclidean distance), and Manhattan distance will be used as an alternative in some occasions.

$$\mathrm{D(n,m)} = \sqrt[2]{(X(n)_x - Y(m)_x)^2 + (X(n)_y - Y(m)_y)^2}$$

A warping path matrix W $\in R_{N\times M}$ then can be calculated based on the matrix$D$. The initial condition for constructing the warping matrix is to assign the element $W(0,0)$ with the value of $D(0,0)$. First, the first row and the first column which start from the first element $W(0,0)$ need to be calculated. To calculate the values in the first row of W, only the horizontal path needs to be considered (using the equation $W(n-1,m) + D(n,m)$). Similarly for the first column, only the vertical extensions will be considered (using the equation $W(n,m-1) + D(n,m)$), and values of the rest elements will be calculated using the following equations:

$$W(n,m) = \min \begin{cases} W(n,m-1) \\ W(n,m) \\ W(n-1,m) \end{cases} + D(n,m)$$

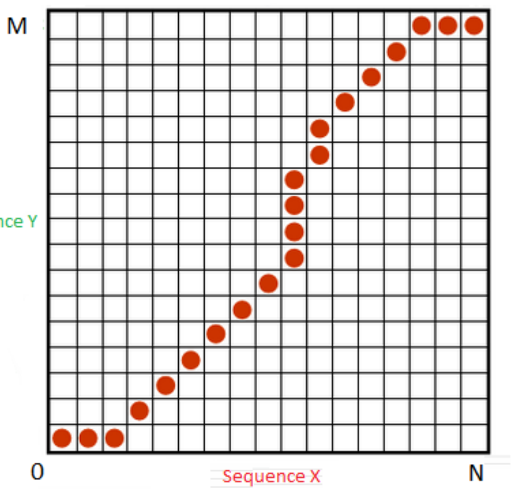


Figure 4. The red points consist of warping path

There are four constraints regarding to calculate the warping path:

- Monotonicity: the alignment path has increasing indices which means both n and m are either increased or stay the same.
- Continuity: the warping path does not jump in time series, meaning m and n can only be increased by 1 at a time, so that important features will all be captured.
- Boundaries: the path starts from (0,0) and ends at (N, M), and that could guarantee the whole sequences will be considered when path is calculated.
- Warping Window: A good warping path should be close to the diagonal of the matrix.

The length of the optimal warping path equals to the value of $W(N, M)$, and the path can be get by tracking reversely from $W(N, M)$ to $W(0,0)$ by choosing elements in $W(n-1, m-1)$, $W(n, m-1)$ and $W(n-1, m)$ with the smallest value. To get the final result, the warping path length will need to be normalized using a Normalized Factor (NF), and the template has the smallest normalized path length will be the best alignment of the gesture.
To use NF to normalize warping path lengths is necessary because each template will have a different path length, and that would affect the lengths of warping paths and lead to inaccurate results. Normalizing the sequences has been disscussed in detail by [21]. They use two differet appraoches to compare the approaches: reinterpolation all the sequences to have the same length; normalising the returned warping distance. As the result of their experiment shows, the two approaches have similar accuracy. The second approach is used in this project and the NF is applied as the following four conditions:

- No normalization on the distance.
- Normalize by the length of the optimal warping path.
- Normalize by the length of the shorter time series (for each pair of the time series during each DTW computation).
- Normalized by the length of the longer time series.

2.3.1 Approaches using DTW

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.
Celebi et al [22] proposed an approach called Weighted DTW, which computes a weight for every joint in each gesture templates. It boosts the efficiency of conventional DTW algorithm since different gestures have different weights on different joints, and weights will be adjusted according to different templates when recognizing the gestures. They use a parametric model which based on the body joints movement (the placement during the performance of gestures) to compute the weights. After filtering the noise in the gesture data (e.g. shaking, trembling), they apply some equations to the data to calculate the result with

the effect of calculated weights. Their work helps to improve the discrimination capability of DTW, and it shows a better performance comparing to the original DTW.

2.4 $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.

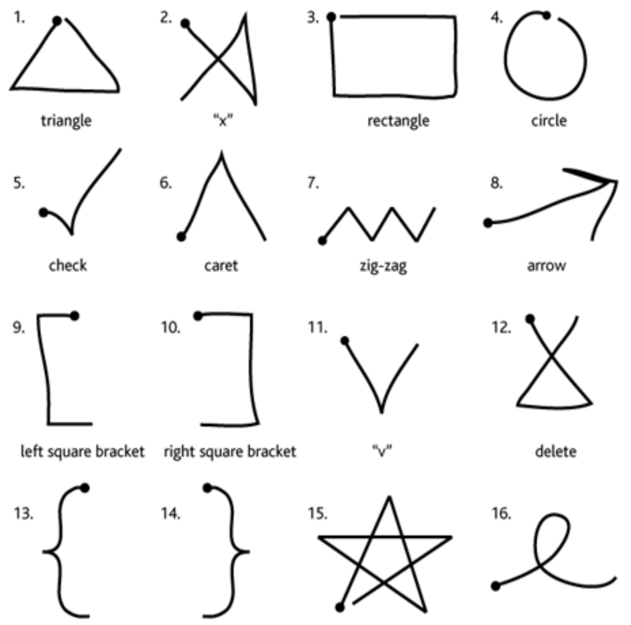


Figure 5. The tested gestures in $1 [13]

$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 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.

2.5 Serious Games

Using commercial video games as a rehabilitation tool has gained a lot of interest in recent years, and some studies show that games are successful for rehabilitation following stroke and some severe injury [14]. Serious games are the digital games that are used for purposes not only for entertainment, but also for many other purposes such as E-learning, training, rehabilitation, etc. Comparing to the traditional methods in rehabilitation, games have some advantages. For example, games are usually designed highly interactive, engaging and maybe even addictive for normal players, this factor could also be applied to patients and help them to involve in the rehabilitation procedure more willingly. Games can create a rich, graphical virtual environment, and that could encourage users to become more motivated, involved and immersed. Virtual environments also provide an approach of safe and cheap training, which can be tailored to particular persons' interests and capabilities. Moreover, the performance and progress can be monitored and analyzed automatically by machines, and the data could be saved and sent back to therapists or be uploaded to remote servers to help therapists observe the treatment and give feedback more accurately.

VR and video-capture systems which could make the process even more immersed; however, the high cost of the equipment and difficulties of setting up the system prohibit that from being popular for home rehabilitation. On the other hand, game consoles and user movement capture devices are widely used in serious games. Game consoles and motion capture devices are relatively cheaper and easier to be accessed. Also, the technics of game consoles are relatively mature, which lower the technical requirements for the system to be used. There are fair amount of researchers reported on the usage of different consoles (e.g. Kinect to XBOX, EyeToy to PS2, and Wii). With the proper usage of user movement capture devices, players can interact with the game effectively using body movement [20].

There are some limitations on developing rehabilitation games on consoles. Commercial console games are usually developed by professional game studios whose target users are general players since the most profit of the game industry comes from them. Individual developer may be more willing to develop rehabilitation games; however, they are usually not authorized to do so, and they may lack of knowledge and experiences on developing console games. Fortunately, companies gradually lower the requirements of developing console games for developers. Microsoft authorizes members of Xbox Creator's Club with only $99 per year, while Nintendo has a similar platform called WillWare. Besides, the consoles are getting more powerful, and the technical difficulties of developing console games reduce, that allows more people to take part in developing console games. Moreover, devices like Kinect also works on PCs, and all of these positive conditions open up possibilities for rehabilitation games to be developed and deployed on consoles and PCs [20].

There are some commercial games which are designed for healthy life, such as Wii sports, Tai-Chi and Yoga-based games for Nintendo and Sony consoles. They may not be specially designed for rehabilitation, but in fact could be suitable. However, as Burk et al [20] pointed out, those games looks suitable for rehabilitation, but the pace is still too fast for patients who have just suffered diseases like stroke. Special designed games just for rehabilitation are still necessary. To design games which are suitable for rehabilitation will require a lot of related knowledge as well as experience. They divided the design into two principles: meaningful play and challenges.

Meaningful play is a design pattern about how players interact with the game. The players should be easily noticed on how their actions affect the game play, and it makes feedback a core part of maintaining meaningful play in games. Feedback helps users to understand how their choices and behaviors affect the game and to measure their progress in achieving goals. The feedback is usually presented to players by visual effect and sound, but it can also exist with other different forms such as numerical scores, health bars, camera movement, and etc. The feedback should occur in all the occasions when human computer interaction happens during the game play, like monsters getting damaged and causing damage, players acquiring new skills and leveling up.

Players usually expect less challenge at the start of a game and desire the challenge increases with the development of the game as they are getting more familiarized with the game and more skilled. It is common for games to have increased difficulties, but achieving that in rehabilitation games would be more difficult. Patients who play the rehabilitation games may vary significantly in experience of playing video games. Some of them maybe play games a lot, while the other may have not played video games before. A possible solution for this problem could be adjusting the difficulty of the game according to the collected data from medical organizations and to the progress in the game. The pace of rehabilitation games is also an important feature related to the difficulty. Due to the inconvenience of movement for patients, the game speed should be slower, while the movement of player-controlled characters should be amplified.

3. DESIGN AND IMPLEMENTATION

In this article, an easy to implemented approach which is inspired by $1 Unistroke Recognizer as well as the DTW algorithm 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 patterns.

A wizard game which integrates the gesture recognition system is built using Unity3D to help patients who need upper-limb rehabilitation to recover. 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 a better recovery.

3.1 Recognition System

The original idea was using $1 recognizer as the recognition algorithm since it can recognize many different kinds of shapes which might be generated by the hands movement, see Figure 5. In order to adapt the mouse-based recognition system to a body movement recognition system, the first step is to change the way of data input. The movement of the mouse was replaced by the movement of the hands, and the points that generated by hands movement will be captured by Kinect sensor and be used for recognition (other joints are also stored for checking constraints and future use). The joints data that Kinect output is 3D vector3 in which the X and Y are ranged from -1 to 1 and Z is ranged from 0 to 8 (the distances between joints and the sensor in meters).

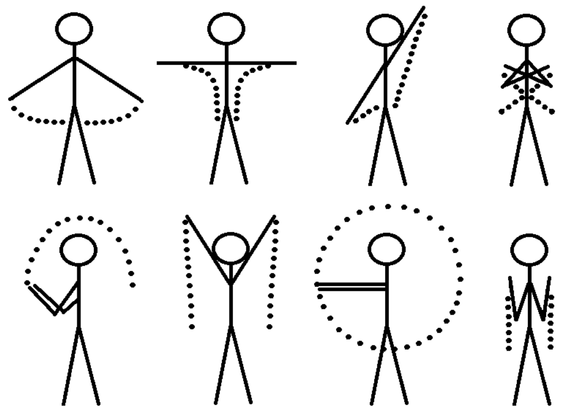


Figure 6. Part of the gestures need to be recognized

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, the data will be similar enough to be recognized as the same gestures. 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 (click and release). 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 was used, and it was to store the hands' positions from the latest M frames in a list with a fixed size in which the oldest position will be removed to free space for the newest position if the list is full. The elements in the list will be used to as the gesture data after the hands holding the position (within an acceptable range of movement) for a certain period of time T. The latest N elements in this list are constantly being checked if the size of the list is larger than N to decide if the movement is stopped (the distances between the last point and the rest N–1 points are smaller than a predefine value). However, it was found not very efficient because to fill the list with the only data for the candidate gestures is hardly to be achieved and a lot of noise will be in the list.

The solution has been modified later to avoid the problem of noisy as the project goes on. Instead of only giving the system a signal when the gesture finished, the system will also be waiting for a signal to start recording. After the system decides to start recording, the list will be cleared, and the gesture data will be pushed into the list without the existence of noise. The rest of the procedure is the same as it was before with a slight change. The difference is that when the list is full, the recording will be finished instead of replacing old positions with new positions, and the latest N positions will be removed after finishing recording because they are used for checking the end signal and should be treated as noise.

There are some drawbacks using this approach. Depending on the lengths of gestures, users may need to adjust the speed of performing a gesture. For example, a gesture which will generate more than $M - N$ positions with the regular speed needs to be performed faster; otherwise, there will not be enough space in the list to store the data. It might be acceptable for the system, but that could lead to a different final trajectory, and the gesture may not be recognized correctly depending on how many positions are missing. Another drawback is that users have to hold their moves both before start to record and after finish performing the gestures for a short while. If extra moves occur before the machine is informed by the signal, that one gesture will be failed to be recognized and needs to be performed again.

The data in the list then will be passed to the recognition system, and the system will process the data so that it can be applied to compare with the templates. The original $1 algorithm also has this process, and it can also be used in gesture recognition with some changes. First, the position list will be resampled to a fixed size N while keeping the trajectory unchanged. To resample, the total length of the path needs to be calculated first, then the length will be divided by N – 1 to get the average length I of each increment. To calculate the new list for resampled date, it will start from the first point of the old list and add the distance between the current point and the next point to a variable D. A new position will be interpolated after the value of D exceeds I, and the interpolated point will be pushed to the new list as well as be inserted to the old list to be the start of calculating the next new position. The second step of processing the date in$1 is to rotate the data to make sure the first point of the data is on the line Y = 0. However, it is not suitable for gesture recognition because the movement of hands for many gestures well generate similar tracks but in different rotation, and those gestures with different rotation cannot be distinguished if the data is rotated. This step of the rotating the points is abandoned in this system so that the rotation of each gesture will be 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 (i.e. the starting point and end point should be close to where they desire to be). The original algorithm then scales the gesture to a referenced rectangle, and translates the gestures to the centroid point (0, 0). This step is useful for normalizing the gestures data because the same gestures cannot be guaranteed to be performed at the same positions in the Kinect space, and the same gesture will be performed by people with different heights.
Some additional steps are needed to adapt $1 to gesture recognition. To solve the problem of different gestures have the similar trajectories but different start positions and end positions, a step of checking constraints of the gestures is added before matching templates and calculating the scores. The constraints which are added to gestures are relative directions between hands and head, hands and elbow, and etc. the constraints will be checked according to the gesture data, and only the templates which have the same constraints with the gestures will be considered necessary to be processed to the next stage. Moreover, the efficiency of the algorithm will be improved by adding constraints to gestures because only those gestures which don't pass the constraints checking will be filtered.

As mentioned by Wobbrock et al [13], the original system struggles to get an effective score for 1-D trajectory such as a line. Therefore, for dealing with gestures which will form lines, an extra step is added. The distribution of the positions will be checked to decide if it is closed to a line or not, and the step of template matching will be skipped if a gesture is decided to be a line. The results will be based on the angles between templates and Y = 0 and the angle of the templates. To decide the distribution of points, correlation coefficient which is commonly used to measure the degree of linear dependence between two variables is calculated using the gesture data. 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 (0.98 is found suitable) is set for deciding lines.

Gestures which are not lines will go through the Golden Section Search algorithm [18] to compare the list of hands points with the points in each template to get scores. To calculate the score, first, the average distance between corresponding points of candidate C and a stored template T will be get using the formula:

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

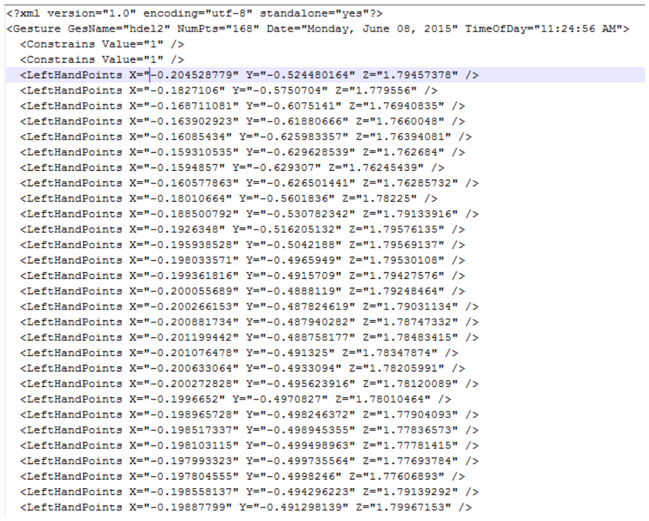
Then the distance d_i and the length of a side of the reference square (size) will be used in the equation below to get a final score which is range from 0 to 1 using the equation:

$$Score = 1 - \frac{d_i^*}{{}^1\!/_2\sqrt{size^2 + size^2}}$$

To sum up, the workflow of the $1 recognizer in gesture recognition is listed below:

- Recording the gestures that need to be recognized as templates and saving them as in XML files which contain the positions of both hands in time manner, basic information of the gestures (e.g. name, time, length, etc.), and the constraints of the gestures.
- Loading the templates from files, packaging them and storing in a list, so that they are prepared to be compared with the real-time data of hands' positions.
- Capturing and storing the latest hands' positions into a list with fixed size. Packaging the data in the same way as the templates, and passing the processed data to the algorithm.

- Comparing the constraints of the gesture data and the templates, filtering the templates which don't meet the same requirement of the constraints.
- Calculating the correlation coefficient to decide if it is a path is a line.
- 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 fine a template which matches enough. After completion of the algorithm, the list that stores the real-time hands' positions will be cleared to prepare for a new gesture.



```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Gesture GesName="hdel2" NumPts="168" Date="Monday, June 08, 2015" TimeOfDay="11:24:56 AM">
  <Constrains Value="1" />
  <Constrains Value="1" />
  <LeftHandPoints X="-0.204528779" Y="-0.524480164" Z="1.79457378" />
  <LeftHandPoints X="-0.1827106" Y="-0.5750704" Z="1.779556" />
  <LeftHandPoints X="-0.168711081" Y="-0.6075141" Z="1.76940835" />
  <LeftHandPoints X="-0.163902923" Y="-0.61880666" Z="1.7660048" />
  <LeftHandPoints X="-0.16085434" Y="-0.625983357" Z="1.76394081" />
  <LeftHandPoints X="-0.159310535" Y="-0.629628539" Z="1.762684" />
  <LeftHandPoints X="-0.1594857" Y="-0.629307" Z="1.76245439" />
  <LeftHandPoints X="-0.160577863" Y="-0.626501441" Z="1.76285732" />
  <LeftHandPoints X="-0.18010664" Y="-0.5601836" Z="1.78225" />
  <LeftHandPoints X="-0.188500792" Y="-0.530782342" Z="1.79133916" />
  <LeftHandPoints X="-0.1926348" Y="-0.516205132" Z="1.79576135" />
  <LeftHandPoints X="-0.195938528" Y="-0.5042188" Z="1.79569137" />
  <LeftHandPoints X="-0.198033571" Y="-0.4965949" Z="1.79530108" />
  <LeftHandPoints X="-0.199361816" Y="-0.4915709" Z="1.79427576" />
  <LeftHandPoints X="-0.200055689" Y="-0.4888119" Z="1.79248464" />
  <LeftHandPoints X="-0.200266153" Y="-0.487824619" Z="1.79031134" />
  <LeftHandPoints X="-0.200881734" Y="-0.487940282" Z="1.78747332" />
  <LeftHandPoints X="-0.201199442" Y="-0.488758177" Z="1.78483415" />
  <LeftHandPoints X="-0.201076478" Y="-0.491325" Z="1.78347874" />
  <LeftHandPoints X="-0.200633064" Y="-0.4933094" Z="1.78205991" />
  <LeftHandPoints X="-0.200272828" Y="-0.495623916" Z="1.78120089" />
  <LeftHandPoints X="-0.1996652" Y="-0.4970827" Z="1.78010464" />
  <LeftHandPoints X="-0.198965728" Y="-0.498246372" Z="1.77904093" />
  <LeftHandPoints X="-0.198517337" Y="-0.498945355" Z="1.77836573" />
  <LeftHandPoints X="-0.198103115" Y="-0.499498963" Z="1.77781415" />
  <LeftHandPoints X="-0.197993323" Y="-0.499735564" Z="1.77693784" />
  <LeftHandPoints X="-0.197804555" Y="-0.4998246" Z="1.77606893" />
  <LeftHandPoints X="-0.198558137" Y="-0.494296223" Z="1.79139292" />
  <LeftHandPoints X="-0.19887799" Y="-0.491298139" Z="1.79967153" />
```

Figure 7. Part of the XML file of template data

Since the $1 is a 2-D algorithm, only X and Y of the hands' positions are used for recognition which means the paths projected on the x-y plane will be used to match the templates. However, human gestures are performed in the 3-D space, and not all of them can be recognized only by checking the trajectories on the x-y plane. Because how $1 does to match the templates involves the usage of angles to find the best alignment, it needs a proper understanding and effort to adapt the algorithm to support 3-D positions. It might be achieved by using matrices to rotate in 3-D space and to use quaternions to replace the angles. The thought of using the 2-D algorithm to match 3-D human gestures alternatively was to store the position data in three copies, one for the x-y plane, one for the x-z plane and the one for the z-y plane. These three copies are three paths of the same gestures that are projected to those three planes, and they will all be applied to the algorithm, and the final results will be affected by the scores from all three planes according to the weights which are set by the users. This is idea is not successful after being experimented, and there are a couple of reasons why it is not a suitable solution. First, gestures will form meaningless trajectories in one or more plane which are useless for recognition. Second, there is no a proper way to assign the weights. Therefore, the idea was abandoned, and that leads to the implementation of DTW.

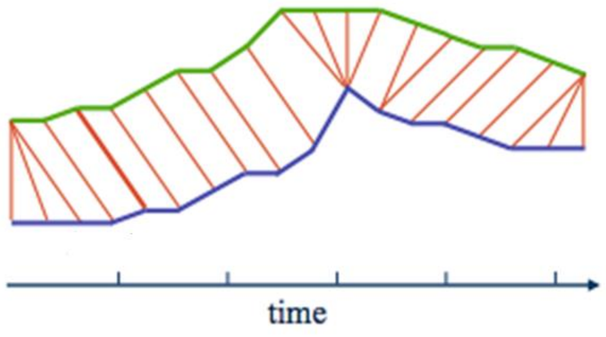


Figure 8. Comparing two sequences using DTW

The progress of implementing DTW is much simpler than implementing $1 because of the gained knowledge of gesture recognition and the completion of the recognition system framework. For implementing the DTW, what need to do are to understand the algorithm as well as implement it, and to implement a procedure of normalizing the data for DTW matching. The process of normalizing the data for $1 is found also suitable for DTW, but DTW doesn't need the gesture and the templates have the sample amount of positions, so resample is not necessary for the gesture data, and resample is only applied to the templates when they are loaded from the files to make sure there will be a fair comparison. The classic DTW algorithm is implemented in this project, and its working flow is the same as what is described in the previous section. The processed gesture data and each of the templates are used to calculate the distance matrices first, and then the distance matrices will be used to get the warping path matrices. Because one dimensional path can be handled by DTW as long as the data is normalized correctly, the algorithm will be applied after the comparison of constraints, and the step calculating the correlation is not necessary when this method is used to recognize gestures. As mentioned above, the way of normalizing data for $1 cannot also be used in DTW, but this is not perfect, and there is no a better way yet been found in this project. Although it works fine for most cases, it has problems in some tricky situations which will be discussed in the result section. Different methods of normalizing the data have been tried, but none of them could solve the problem.

After getting the optimal distances for all the remaining templates, they will be normalized using the equation which is mentioned in the last section, and all the normalized factors will be applied to the equation. The smallest value among all the templates' results will be used as the final result, and it will be converted to the range of 0 to 1 using a similar equation that is used in $1.

Because of the nature of the DTW algorithm, it could be easily adapted to support 3-D gesture recognition. To calculate the warping path in 3-D space is almost the same as it is in 2-D space except for the equations that are used to calculate distances. By adding functions to support the calculation in 3-D space, a 3-D version of DTW is also be achieved, and it can solve the problem of recognizing 3-D recognition and be used to compare the efficiency with 2-D gesture recognition.

3.2 Rehabilitation game

The idea of the rehabilitation game is inspired by a game named Clicker Heroes [16]. The rehabilitation game will be presented in the first person view, and it only has a single scene which has the player, the monsters (bosses) and the terrain. The basic concept of the game is that players cast spell by performing gestures to beat monsters and use the diamonds dropped by the monster to upgrade spells or unlock new spells to defeat stronger enemies. The values in the game can be designed accordingly to the rehabilitation treatment of the patients. For example, only on spell can be used in one level, and the monsters' HP and the damage of the spell could lead to the effect that a certain amount of times of performing the designated gestures is required to win the level. When players advance to a more difficult level, a more difficult gesture will be unlock to cast more powerful spells. Through this mechanism, the game could lead players to finish a treatment procedure with gradually added difficulties. Considering that to do the same excises repeatedly will be necessary in rehabilitation, the diamonds can also be used to upgrade low level spells to make them more powerful than higher level spells in later game, so that players will still choose to perform the easy gestures at some points of the game.

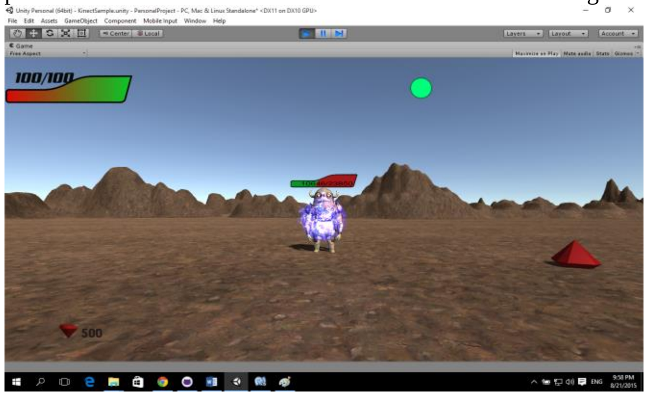


Figure 9. The game UI

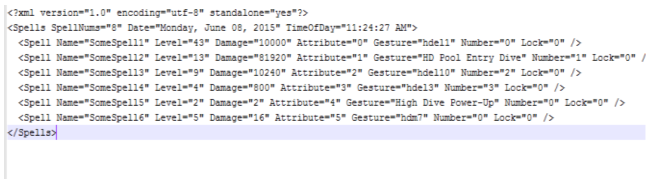
In order to make the game more challenging, monsters are set to attack players once in a certain period of time if they are not killed. The health and attack of the monsters will be increased as the game goes on. The player will gain experience after killing monsters, and the health of the player will be increased and restored by leveling up. The health can also be restored by spending diamonds to buy portions. If the player is killed by the monster, the penalty will be minimum (e.g. spending some diamonds to revive) in order to avoid upsetting the players.

After defeating a certain amount of monster in one level, the player and the camera will move around the map to reach the position for the next level and to fight different monsters. Diamonds that dropped after the death of monsters can be collected by the player using the hand cursor on the screen, or they can be left on the ground and be collected automatically as the game moves to the next. The game is designed to be endless, but an ultimate boss could be set in the game to be the flag of the end of the treatment process.

The movement of hand cursor is controlled by the position of user's right hand. The positions collected by the Kinect will be converted into relative positions in screen space, and using the Lerp function to move it smoothly. Diamonds that are dropped by monsters will be collected when the screen positions of hand cursor and diamonds overlap. The diamond counter which consists of Unity GUITexture and GUIText will reflect the change of the amount of diamonds when they are collected or consumed.

The spell in this game is made using the Particle System in Unity3D. Each spell prefab has a Particle System component to take care of the rendering part of the game object, a rigid body to simulate the physical movement and a collider for collision detection. The spells are connected to the gestures in the configuration file which stores all the spells, their information, and related gesture names.

Monsters in this game have only three behaviors: idling, attacking, and being hit, the animation is handled by the legacy Animation and Animation Clip components for simplicity. The behaviors and attributes of the monster are written in the Monster class, and the GameLogic class is in charge of spawning different monsters. The health bar of the monster is achieved by placing one texture (green) on top of another (red) which overlap each other completely at the beginning when the health is full. When the monster is damaged by a spell, the Lerp function of Unity engine is called to gradually reduce the value of health as well as to move the green texture along the negative direction of X axis accordingly. Because the a mask was added to the red texture, the part of green texture which is not overlapped with the red texture cannot be seen, and the red texture will gradually appear as the green texture moves away until the health bar turns to be totally red when the monsters die.



```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Spells SpellNums="8" Date="Monday, June 08, 2015" TimeOfDay="11:24:27 AM">
  <Spell Name="SomeSpell1" Level="43" Damage="10000" Attribute="0" Gesture="hdel1" Number="0" Lock="0" />
  <Spell Name="SomeSpell2" Level="13" Damage="81920" Attribute="1" Gesture="HD Pool Entry Dive" Number="1" Lock="0" /
  <Spell Name="SomeSpell3" Level="9" Damage="10240" Attribute="2" Gesture="hdel10" Number="2" Lock="0" />
  <Spell Name="SomeSpell4" Level="4" Damage="800" Attribute="3" Gesture="hdel3" Number="3" Lock="0" />
  <Spell Name="SomeSpell5" Level="2" Damage="2" Attribute="4" Gesture="High Dive Power-Up" Number="0" Lock="0" />
  <Spell Name="SomeSpell6" Level="5" Damage="16" Attribute="5" Gesture="hdm7" Number="0" Lock="0" />
</Spells>
```

Figure 10. The XML file for spells

The Unity Terrain component is used for the creation of the ground on which the monsters live. Since the game is design to be endless, the terrain should also be endless so that the player will always have the next destination. In this prototype, only one terrain is created manually, and it will be looped around with its copy to achieve the endless terrain. The terrain is divided into four section (just like four quadrants if looks the terrain as a plane), and each section will be a battle field. The player will move to the next section after the boss in each section is defeated, and after finish all four section in one terrain, the player will move to the next terrain. To make the game more attracting, more terrains could be created or the terrains could be generated by some algorithm on run-time to make it not be repeated, which will give players more motivation to explore the game.

UI is also an important part of the game. There are two planes to draw the movement of hands on 2D space as well as two planes for displaying the trajectory of selected templates by using the Unity Texture2D class and SetPixel function. They make the job of testing the game much easier, and the players could understand more how they should perform the gestures. However, the disadvantage of using plane to display the hand track is that they will distract the players from the game by make them focus on generating desired 2-D tracks to get better results. Therefore, the planes will be disabled during the gameplay.

A menu for consuming the diamond is built based on the Unity Scroll View component. The menu can be opened using two hands to swipe from right to left and closed using one hand swipe from left to right. All the available spells will be displayed in the list as UIButtons, and the distribution and positions of the spells are handled by components including Vertical Layout Group, Content Size Filter, and Layout Element. Gestures swipe down and swipe up are used to move the focus of the spell, and swiping right to left using right hand will simulate the behavior of clicking the button to achieve upgrading the spell. On the left side of the menu, an animation which indicates the related gesture to the spell is displayed for users to check how to cast spells. Lastly, the game data will be saved to record the progress that users made, so that they will be able to continue the process accordingly.

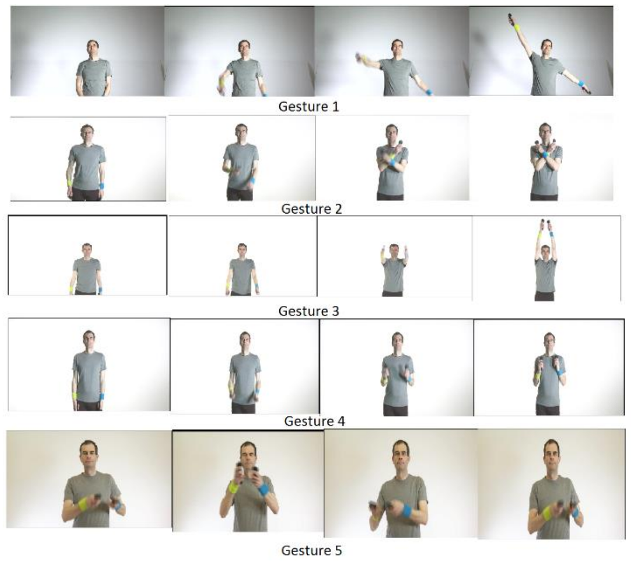


Figure 11. Some of the test gestures

4. RESULT AND EVALUATION

The $1 and DTW both have been tested using the same gesture set that is in the videos provided by the medicine school. Since the gesture data will be normalized, the distance between the user and the Kinect sensor is flexible as long as there is enough for Kinect to track the upper-limbs of the user entirely, which means the upper-limbs should be contained in the camera space no matter how they move. The test gestures were performed 20 times each, and some results show in Table 1 to demonstrate the strength and the weakness.

Algorithms / Gestures	$1	DTW(2D)	DTW(3D)
Gesture1	100%	100%	100%
Gesture2	0%	10%	20%
Gesture3	100%	100%	100%
Gesture4	100%	10%	10%
Gesture5	100%	100%	100%

Table 1. Test results

The recognition results are accurate overall for most of the selected gestures (some of them are not listed above), but the system performs poorly in recognizing some gestures.

Gesture 1, 2 and 3 are all recognized by the system with the perfect accuracy, because the data is suitable for the algorithms to be applied.

The system fails to recognize Gesture 2. The reason why $1 failed is that the skeleton cannot be tracked correctly when the hands overlap. For DTW, except for the above reason, the gesture will be confused with Gesture 4 and 5 due to the normalization was just an approximation, and that leads to the results for those gesture quit close, and for some unknown reason Gesture5 usually gets the highest score. This is the same reason why Gesture4 cannot be recognized correctly by DTW.

Gesture 3 and 4 will generate similar tracks, but Gesture 3 can be recognized correctly while Gesture 4 cannot, because the constraints of Gesture3 is unique in this set and the constraints of Gesture4 (the hands start from positions below the head and also end at positions below the head) are the same with Gesture 2 and 5.

5. CONCLUSION

In this section, the project will be summarized including describing how it went and highlighting some key features. Beside, how this project could be expanded will also be discussed.

5.1 Project summary

After doing research on the subject of gesture recognition and reading related articles, the knowledge of how gesture recognition works and how could it be achieved has been gained, and the recognition system has been created. The system performs well in recognition without training, and that meets the expectation of creating a simple system with acceptable efficiency.

Limitations are found for this project after running experiment. For the $1 algorithm, it is an easy employed algorithm, and performs identically with the popular DTW algorithm. However, since it is originally developed for recognizing 2-D shapes and hand writing, the performance in 3-D gestures is quite limited. When distinguishing two gestures which will form similar track on the main plane (x-y plane), the accuracy is unsatisfied. It especially fails to recognize some gestures which do not generate meaningful tracks on the main plane, and the exploration of finding a proper solution was failed. Besides, it cannot deal with 1-D tracks correctly without making changes to the Golden Section Search algorithm. The solution that is used in this project works solved this problem, but it restricts the gestures that users perform.

When using DTW algorithm to recognize gestures in 2-D space, it gives more flexibility for users to perform the gestures since it can handle 1-D trajectory. The problem of DTW is a method of normalizing the data ideally for recognition is not yet been found, and the algorithms fails to recognize gestures with similar trajectories but different directions. Even after expanding DTW to support to recognize 3-D trajectories, the problem still exists. On the other hand, using DTW to recognize 3-D gestures directly instead of their projections improved the usability and accuracy of the system.

The rehabilitation game prototype has a completed game cycle and game logic, and shows the basic concept of the game design. There are still some features have not been added to the game due to time limit, and a lot of new elements can be added. For the accuracy of recognizing gestures, the start and the end of each gesture that users perform are set to holding for a short period of time, and that limits the use experience since users can only perform gestures according to the signals.

5.2 Further work

Although the recognition system works fine in the test environment, there is still a lot of optimization can be done. The currently system only tracks the movement of hands which are the major joints for upper limbs gestures. However, to track more joints like shoulders and elbows could help the system to recognize gestures more precisely and to distinguish gestures with similar hands movement more accurately. Machine learning algorithm for classification could be implemented, and with proper training to store more templates for each gesture to improve the accuracy. $1 is adapted to recognized body gestures without modifying the 2-D algorithm. With a proper understanding and modification for this algorithm, it could be using the raw 3-D positions instead of using the 2-D projections.

Moreover, the template matching method for gesture recognition is not very suitable (at least if it is used as the only approach) for applications like games from my personal experience. A separated rule-based system can be implemented, and combining two types of approaches together to take advantages of both of them can make the application more practical.

Lastly, the game is only a prototype which shows the basic concept of the design and how the game will be played, and a lot of works can be done. For example, the game system could be organized more robust, and the economy, difficulties and battles could be detailed designed to fit for rehabilitation. A database can be established to store the information about the performance of different users to play this game, and the data can be used to analyze the patients' performances and to track the rehabilitation process.

6. 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].

[20] J.W. Burke, M.D.J. McNeil, D.K. Charles, P.J. Morrow, J.H. Crosbie & S.M. McDonough "Optimising engagement for stroke rehabilitation using serious games," *Visual Comput., vol. 25, no. 12, pp. 1085 –1099, Dec. 2009.*

[21] *C.A. Ratanamahatana* & Keogh, "Everything you know about dynamic time warping is wrong," *Third Workshop On Mining Temporal and Sequential Data*, Aug. 2004.

[22] S. Celebi, A. Aydin, T. Arici & T.Temiz, "Gesture Recognition using Skeleton Data with Weighted Dynamic Time Warping," *Third Workshop On Mining Temporal and Sequential Data*, vol. VISAPP, no. 1, pp. 620 – 625, Feb. 2013.