An Automated Physiotherapy Exercise Generator

Ross Yeager

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-91
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-91.html

May 17, 2013

1 Introduction

Natural user interfaces (NUI) are the new standard in human computer interaction; they involve communication based on human senses and include (among others) voice recognition, motion detection, and touch interaction [1]. The NUI's allow users to interact with their electronic devices in a manner that mimics user interaction with the physical world, thus making device use easier for all users. However, the applications for such technology have only begun to be tapped. One particular technology that has become a successful NUI device is the Microsoft Kinect. This small device has a camera that generates accurate depth images from the field of view; user skeleton data can then be extracted from these depth images [2]. Applications that utilize the Kinect process this depth information in order to interpret various controlling movements. In our work, the main objective was to create a fully functional physiotherapy application based on this NUI concept. This application utilizes Kinect's depth information and human skeletal readings and allows patients to properly perform their exercises within the comfort of their own home or available environment. In order to create an application that tracks patients' progress during exercises while offering specific feedback, a substantial exercise set must be developed.

This paper focuses on the automatic generation of these exercises in the context of a Kinect based application.

We make two contributions: First, an XML-based exercise definition language that describes exercises as a sequence of distinct poses; second, a method to create exercise definitions from user demonstrations.

Both components are instrumental in creating a robust and functional therapy application.

To generate motion trajectories between key poses, we use quaternion interpolation of skeletal limbs similar to methods developed in graphics to create animated movement [3]. This paper explores the definition of exercise movement using key poses and rotational interpolations as well as automated methods for identifying and calculating relevant exercise information.

2 Previous Work

In regard to previous work, there are several previous studies that attempt to take on the challenge of defining motion.

European researchers have developed a prototype video-game application, PlayMancer, that uses multiple sensors along with computer vision software to track and monitor patients [4]. PlayMancer is a video game platform developed to treat mental disorders; it measures galvanic skin response, oxygen saturation, heart rate, skin temperature, breathing frequency, as well as facial gestures using the MobiHealth Mobile biosensor system [4]. They defined key components in their rehabilitation by periodically measuring the MobiHealth Mobile sensors at key moments within the video game in order to gauge the user's mental and physical state.

Yeong et al [5] analyzed rehabilitation movement in robotic devices. Their study used object marker velocities to differentiate outward and inward movements of human users performing routine daily tasks. Motion was defined as threshold components of the specified marker peak velocity. This method of movement definition was highly accurate; however, it is limited in its application to Kinect-based exercise definition because it requires a digital optical motion system along with passive markers to classify motion [5]. Yeong et al divided up movements (subsets of the overall motion) into 60-120 clusters and created normalized deviations of each of their motions.

Another previous study on motion definition came from Hondori et al [6]. This study differs in that it used sensor fusion to examine the user's motion and also covered a very specific set of motions that only involve food and drink consumption for post-stroke patients. They used both the Kinect information as well as inertial sensors to track movement by measuring change in joint combination positions and angles. The different movements/tasks were then defined as a combination of positions and angles over a determined period of time [6]. A limitation to this method is that it was not designed for real-time feedback; it required the gathering data beforehand [6]. Because of this, it is not sufficient to use their method to construct the exercise structure. Hondori et al provided inspiration for our work to take on a similar approach to provide rapid input to our pre-defined XML structure, thus allowing for fast and efficient exercise definition modification.

In the area of developmental disabilities, researchers created a Kinerehab application that targeted patients with motor disabilities [7]. In this study, Chang et al created a system that only used the Kinect to process patient exercises and motor movements. Patients were shown the proper exercise, and the Kinerehab software determined whether or not the patient successfully completed the exercise [7]. More specifically, exercises were determined based on ending points calculated by analyzing joint angles and position in 3D space [7]. The rehabilitation exercises in this particular study resembled static position holds rather than the dynamic movements in a fully functional physiotherapy application. Because of this, the finishing point exercise definition structure provided little value to the more advanced exercise application developed in our study.

Previous studies have also pointed out additional issues and parameters involved in motion detection such as Kinect user calibration [8], noise involved in user movement [9] [10], as well as occlusion [10]. Although not direct components of exercise definition, these types of limitations need to be accounted for within the exercise definition structure in order to provide users with a smooth and accurate product.

3 Methodology and Approach

3.1 From Exercise Definition to User Tracking: An Overview

With the value of the mKinect physiotherapy application relying heavily on the presence of a substantial database of exercises, both the speed of input and overall structure of the exercise definition are particularly important. Exercise definition must capture key poses (denoted interchangeably as steps throughout this paper) that are defined by the therapist as well as capture the transitional movements between these steps. All exercise definitions are generated in an XML format that is read by the front end application during patient use. With a concrete definition of any arbitrary exercise, a recording application can then be used to calculate the defined measurements in order to generate exercise recordings in XML format. The following section describes the data needed to fulfill the exercise definition requirements and how that data is used to structure the output file format.

3.2 Exercise Description File Format

As mentioned above, exercise definition must include both key frame recording as well as transitional movements between these key frames. The Exercise Generation Interface achieves this by calculating 3D points, joint angles, and quaternions at each step. Each of these parameters is automatically calculated using the exercise generation application and recorded in the definition XML file. The XML structure is defined by the following hierarchy:

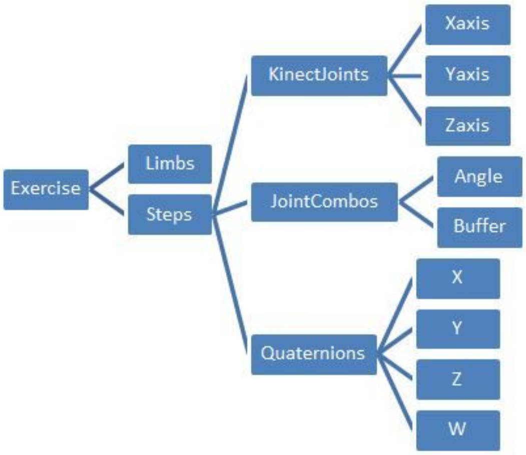


Figure 1: XML Description Layout

3.3 XML Layout: 3D Points

The first category within the XML step layout is the KinectJoints. KinectJoints are found for each of the 20 joints (see Figure 2) identified using the Kinect SDK API and are centered at the trainer's HipCenter joint (root joint).

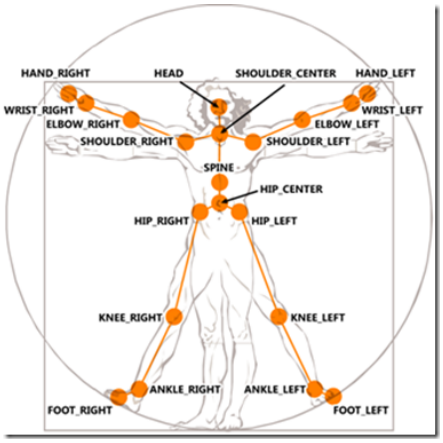


Figure 2: Kinect Skeleton Joints [11]

We measure these points in order to have the 3D joint coordinates at each key pose. These measured quantities are compared to the realtime 3D joint coordinates of the patient using the exercise recognition application.

The HipCenter is anatomically considered to be the "root" joint in that any rotation of this joint follows with rotations of all other joints; it is the anatomical origin [12]. This point was chosen as the origin based on observations of various physiotherapy exercises, and from the joint hierarchy seen in the Kinect SDK 1.7; most exercises do not involve a rotation at the HipCenter joint and are, in general, focused mainly on lower or upper body extremities. We considered any hip-based exercises as a special case and leave them for future work. The centering of the 3D points provides a coordinate normalization that can be used to center each individual patient's coordinate system to provide more accurate results during testing. The application records for 6 seconds while 72 such 3D measurements are taken for all joints. These measurements are then averaged at each joint and output to the XML file.

3.4 XML Layout: Limbs

The Limb tag is another item recorded in the XML file. This parameter holds the length, in meters, of each of the bones connecting each of the 20 joints produced by the Kinect skeletal tracking data described above. See Appendix A for a detailed description of these limbs. We measure this quantity in order to provide a coordinate normalization that will allow the application to work with users of varying body types. This is referred to as retargetting. Although we measure the data needed to implement retargetting, we reserve the full development for future work.

3.5 XML Layout: Joint Angle Combinations

Another XML parameter within each step is the angle in between the skeletal bones (See Appendix A). This parameter is used in conjunction with the 3D joints to establish the key pose steps. The exercise recognition application will use these angles and the 3D joint coordinates to compare with the realtime measurements of the user in order to ensure the exercise is being performed correctly. Tagged as JointCombos within the XML file, these angles are calculated by using 3 joints interpreted as 2 vectors (bones) with a common origin (the middle joint). The angle in between these normalized vectors is calculated using the dot product:

$$\theta = \arccos(\frac{\mathbf{a} \cdot \mathbf{b}}{|a||b|}) \tag{1}$$

The angle at each of these joints is calculated at each key pose. Only the joint combinations relevant to the exercise are kept; this includes joint combinations that had a significant change in magnitude between steps as well as all of the respective children joints. Children combinations are included because they are implicitly moved with their parent combination. The hierarchy for the joint angles can be seen below in Figure 3.

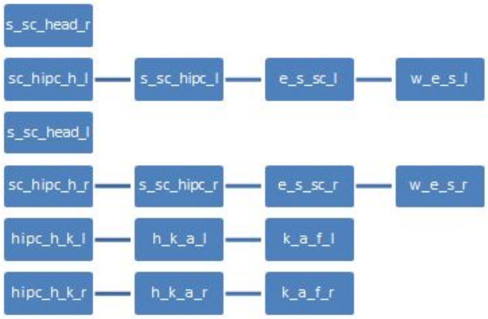


Figure 3: Joint angle hierarchy

3.6 XML Layout: Quaternions

Another quantity measured and stored in the XML exercise definition is the quaternion point at each joint. Because of the simplicity in representing rotations and the ability to easily interpolate points between two quaternions, paths between each step in the exercise are defined using quaternion representation. Due to the anatomical human structure, joint movements consist of rotations of child bones around corresponding parent bones [12]. Bones are described in a hierarchical fashion outlined in Figure 4. Quaternions can capture such rotational movement. Before further examining why we use quaternions in our application, a brief overview of quaternions is necessary. A quaternion is a vector method for representing the $\mathbb{R}^3$ rotation that traditionally uses Euler angles $\{\psi, \phi, \theta\}$. The sequence of Euler angle rotations about an arbitrary axis can be generalized using rotaion matrix multiplication as follows:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2}$$

Using quaternions, the same rotation can be represented using the quaternion product [13]:

$$\mathbf{w} = \mathbf{q}\mathbf{v}\mathbf{q}^* \tag{3}$$

where $\mathbf{w} = [0, x', y', z']$', $\mathbf{v} = [0, x, y, z]$ $\mathbf{q} = [\cos(\frac{\theta}{2}), l\sin(\frac{\theta}{2}), m\sin(\frac{\theta}{2}), n\sin(\frac{\theta}{2})]$', $\mathbf{q}^*$ = the conjugate of $\mathbf{q}$, and $[l, m, n]$' as the $\mathbb{R}^3$ vector axis of rotation.

In vector notation, a quaternion, q, is the vector representation of the angle of rotation around an arbitrary axis, where

$$\mathbf{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}; q \in \mathbb{R}^4 \tag{4}$$

Furthermore, quaternions have a unique property in that given two key frames in a motion (i.e. the steps in the application), a Spherical Linear Interpolation (SLERP) can realistically calculate a rounded interpolation between two points, allowing for the path between two steps to be easily calculated [13][14]. SLERP interpolation is already heavily used among game programmers and animators to accurately show human movement, making it a logical approach to calculate motion between the exercise steps within the mKinect study [15].

The SLERP interpolation equation can be seen below:

$$q_t = \frac{\sin((1-t)\phi)}{\phi} q_1 + \frac{\sin(t\phi)}{\sin(\phi)} q_2 \tag{5}$$

with q_t as the interpolated quaternion, q_1 as the starting quaternion point, q_2 as the ending quaternion point, ϕ as the inverse cosine of the dot product between q_1 and q_2, and t as the percentage of the path between q_1 and q_2.

We use quaternions in our exercise definition because of this interpolation property; the application can capture N points in between two given quaternions. With the knowledge of two quaternions at the key poses, the transition paths can be calculated, capturing the whole exercise movement just using key pose information. The exercise generator can effectively capture N exercise path points by simply storing 2 quaternions at each step, maximizing efficiecy and minimizing storage size.

In our application, the automated exercise generator records these averaged quaternions for every joint in the hierarchy at each step; these are output to the XML file. The final frontend application traverses the joint hierarchy tree and calculates the interpolated quaternions in between these given quaternions (Figure 4).

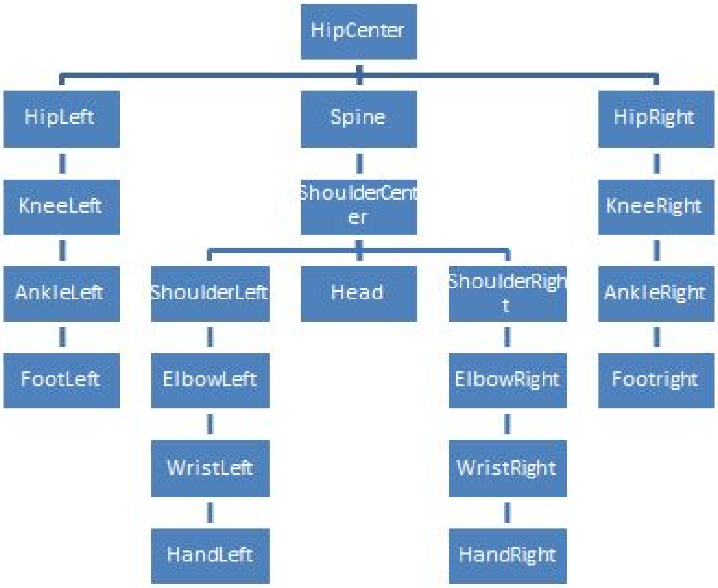


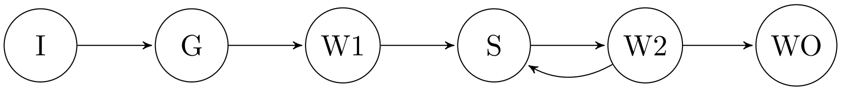
Figure 4: Joint hierarchy

The hierarchical quaternion points are detected at each step for all of the joints in regard to their corresponding bones. Defined in many graphics methods, a bone is defined as a vector between two joints[16]. The quaternion defines the angle of rotation between the given joint bone and its corresponding parent bone (defined in the hierarchy in Figure —4); the vector normal to the parent and child bones is used as the axis of rotation [17].

The exercise recognition application then uses the quaternion distance between the live user quaternion and the calculated quaternion from interpolation to provide feedback based on this distance. The most recent release of the Microsoft Kinect SDK (1.7) provides a hierarchy structure that allows access to calculated rotation quaternions on a joint-by-joint basis and is used in our application to computethe joint quaternions [17]. This API allows for the quaternions at a specified bone to be computed in terms of the reference Cartesian axis in the parent-bone space; the y-axis in this object space is equivalent to the parent bone [17]. The quaternions are measured over a 6 second period (72 readings at the Kinect's 12 frame/sec speed) and are averaged by continually taking the measured quaternion and the previous averaged quaternion as inputs into the SLERP equation with a time step of $\frac{1}{72}$. The result is an averaged quaternion for every joint at each step.

3.7 Recording State Machine

The application uses a simple state machine to record the exercises. It cycles through 5 states: IDLE, GLOBAL, WAIT1, STEPS, WAIT2, and WRITEOUT. The steps are continued until all steps are recorded. Voice recognition is used to increment between steps to allow the user to maintain their spacial positioning. The simple FSM is seen below. Each STEPS state also takes a screen shot of the user defining the exercise and saves it for debugging and for displaying the steps to the patient. The XMLHelper class then uses the collected data and outputs it to the defined XML structure (Figure 1).



4 Developed System

The following subsections describe the automated exercise generator interface and present the results of the exercise application using the generated exercises.

4.1 Early Pilots: Manual Point-to-Point

Our first prototype structure initially mimicked the definition approach seen in the work by Fernandez-Aranda et al [4], with exercises based on periodic measurements of Kinect sensor readings. The setup was similar to that seen in the Kinect rehab trials seen in Huang et. al [18], with exercises made up of a starting point and a finish point. Kinect data would be analyzed at regular time intervals to compare the user to the programmed exercise. A basic 90^0 Right Arm External Rotation exercise (Figure 5) was chosen with start and stop points manually entered into the exercise definition XML file. This method was only satisfactory for basic learning and interaction with the Kinect API and sensor readings. Once the key poses in an exercise were calculated, the perspective projection coordinates were analyzed and specified joint angles were calculated. This method did not account for the movement transitions between these steps, allowing the user to take any route from start point to stop point: clearly an invalid completion of the exercise.

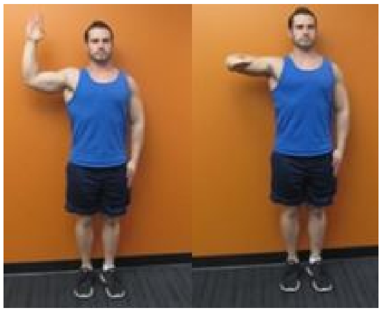


Figure 5: An example of exercise points 1 and 2

We observed from initial prototypes that the actual angles calculated by the Kinect were substantially different from the intuitive angles. This made it practically impossible to manually scribe exercises. For example, holding one's arms out in a T position would intuitively seem to have 180^0 angle between the Shoulder Center, Shoulder, and Elbow; however, the actual angle is approximately 158^o (which would be very difficult and tedious to estimate by inspection). This heavily constrained any type of manual input of the exercise definition based on visual inspection and led to the automated exercise process seen in the current application.

4.2 Recording Exercises with the Exercise Generation Interface

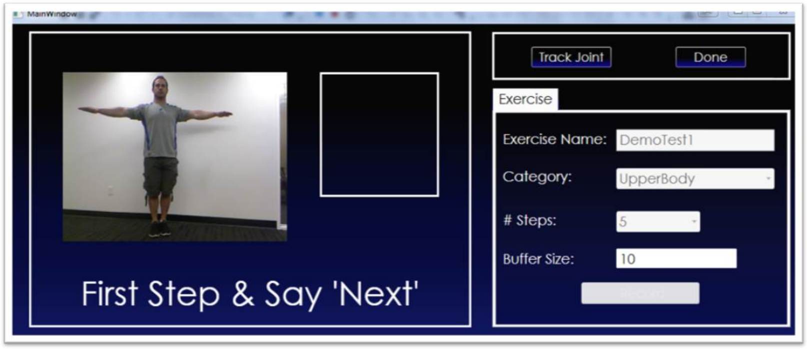
The Exercise Generator Interface is a separate backend user interface designed for application designers and physical therapist. This interface allows designers and therapist to directly perform the exercise in a step by step manner and outputs the corresponding XML file and step pictures (as in Figure 5).

The application first takes "global" measurements in order to get a reading on the overall dimensions of the user. This was done so that exercises can be proportionally scaled based on the patient's hip center when the exercise is being monitored on the actual application. The exercise name, relevant extremity, number of steps and general buffer size are entered into the interface; the user then performs each step until the exercise is completed (see Figure 6).

When recording an exercise, the user/trainer must know the general steps to this exercise (usually described in any physical therapy textbook such as in [19]). An exercise must be broken down into these steps. Based on testing, a generalized rule to follow in regard to step breakdown was to have each step resolution restricted to having all joint rotations less than 90^0 around any of the standard axis. This distinguishes various possible rotations between points by keeping each step to be, at most, within one quadrant of its starting point and to limit the motion to 1 degree of freedom, allowing the exercise XML generator to properly document the exercise. For example, in order to properly document the external arm raise motion, one could not go from hands at his side position at step 1 to having his arms raised above his head in step 2 as there would be too much ambiguity between steps; finer resolution in the exercise steps would be required.

Once the exercise is correctly broken down into corresponding steps, the user can then input the information into the interface and begin the exercise. When the user is positioned at a key pose step, built-in voice recognition recognizes when the user says 'Next' and the application then records the corresponding exercise step. Steps are continued until the exercise is complete.

The following screenshots show the recording of an exercise using the automated exercise generator:



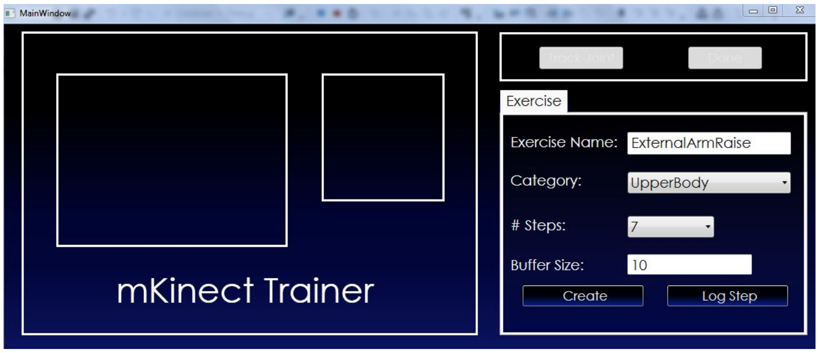
Exercise Name: DemoTest1

Category: UpperBody

Steps: 5

Buffer Size: 10

Figure 6: Taking global measurements before recording exercise



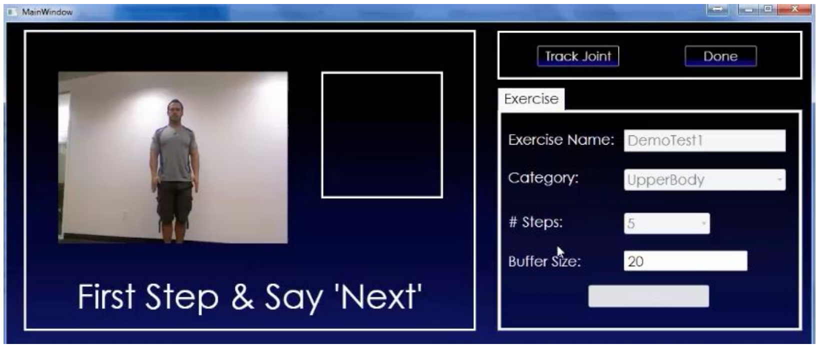
Exercise Name: ExternalArmRaise

Category: UpperBody

Steps: 7

Buffer Size: 10

Figure 7: Enter exercise name, number of steps, extremity, and buffer size



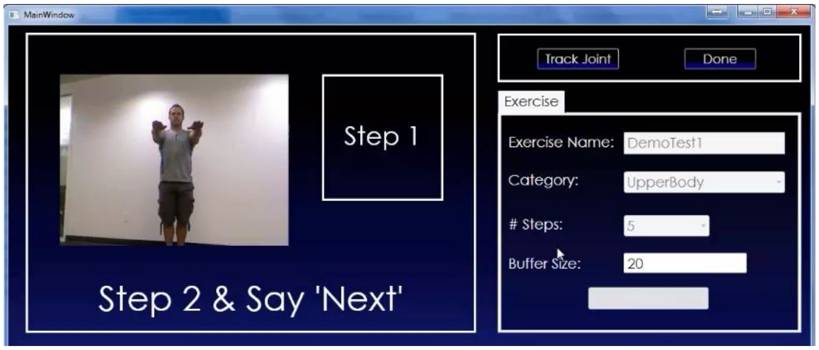
Exercise Name: DemoTest1

Category: UpperBody

Steps: 5

Buffer Size: 20

Figure 8: Get into the position for the first step, say ’Next’ and hold position until prompted



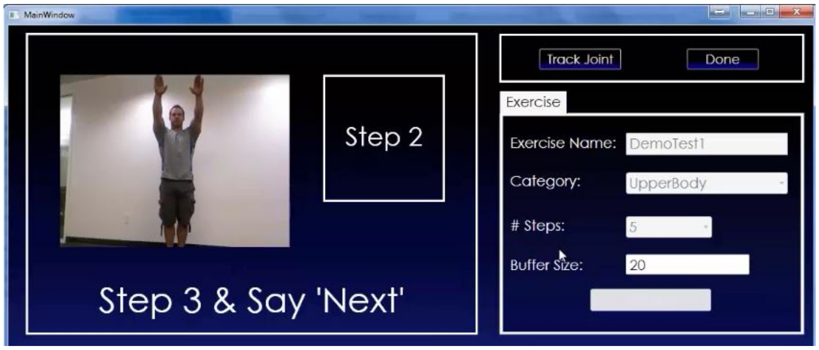
Exercise Name: DemoTest1

Category: UpperBody

Steps: 5

Buffer Size: 20

Figure 9: Get into the position for the second step, say ’Next’ and hold position until prompted



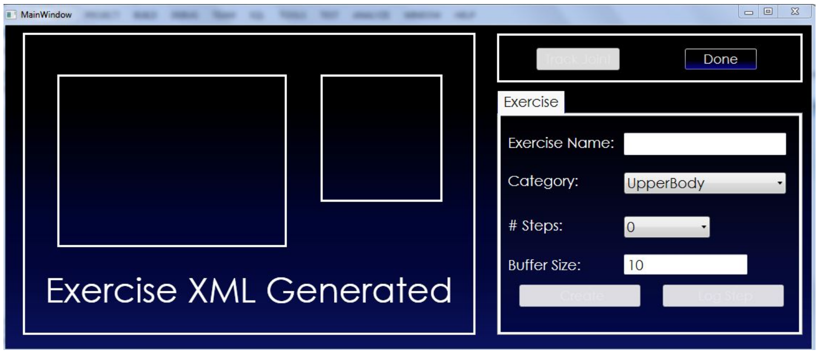
Exercise Name: DemoTest1

Category: UpperBody

Steps: 5

Buffer Size: 20

Figure 10: Get into the position for the final step, say ’Mext’ and hold position until prompted



Exercise XML Generated

Exercise Name:

Category: UpperBody

Steps: 0

Buffer Size: 10

Figure 11: The corresponding exercise is generated

The resulting XML file can be directly input into the frontend exercise recognition application for patient use.

5 Results

During testing, 5 key exercises were recorded using the exercise generator. These exercises were then tested on the frontend application. The exercises were chosen in that they represented a wide variety of skeletal movements in order to provide intuition as to how the full application (recording exercises and running the frontend application) would perform on exercises not specifically tested. Testing included the following exercises (see Appendix B for an example of the first exercise, the Military Press) [19]:

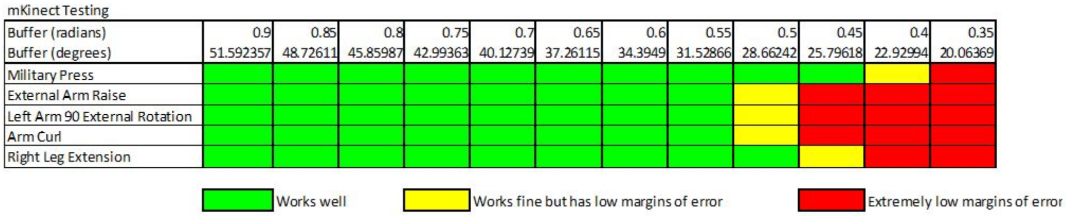
- Military Press
- External Arm Raise
- Left Arm 90^0 External Rotation
- Arm Curl
- Right Leg Side Extension

Each exercise was recorded and tested on the frontend application using the same user for recording and testing to avoid any retargetting issues (retargetting was not addressed in the current version of this application). The distance between the Kinect and the user was varied as well in order to test robustness.

Observations during testing revealed that defining more steps in an exercise structure increased the application accuracy. This was due to the fact that an increased number of steps reduced the amount of point-to-point interpolation, and was most evident if the interploated quaternions were spacially separated by larger distances. The number of interpolations between quaternions was also used as a parameter and tested. Increasing the number of interpolation points increased accuracy (as intuitively one would suspect) but also introduced lag. Through testing, it was found that using 1000 interpolation points maximized the number of interpolations without introducing lag in the application. Furthermore, it was observed that the correct choice of thresholds and buffer sizes plays a crucial role in the accuracy and functionality of the application.

Feedback during interpolation was calculated using quaternion distance between the correct interpolated path and the patients. Each exercise was tested with the same distance threshold on the quaternion distance measurement to see if the algorithm still recognized the correct exercise.

Testing results showed that each of the tested exercises were correctly tracked with an interpolation threshold of 28^0-31^0. This global quaternion threshold was found to allow enough of a buffer to not overconstrain the user as he performs the exercise while still holding the user accountable to the exercise path. The threshold results can be seen in the table below:



mKinect Testing

Buffer (radians) Buffer (degrees)	0.9 51.592357	0.85 48.72611	0.8 45.85987	0.75 42.99363	0.7 40.12739	0.65 37.26115	0.6 34.3949	0.55 31.52866	0.5 28.66242	0.45 25.79618	0.4 22.92994	0.35 20.06369
Military Press	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥
External Arm Raise	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥
Left Arm 90 External Rotation	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥
Arm Curl	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥
Right Leg Extension	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥

🟩 Works well 🟨 Works fine but has low margins of error 🟥 Extremely low margins of error

Figure 12: Testing of 5 key exercises and global quaternion distance threshold

The quaternion path was verified by attempting incorrect movements during the exercise and verifying that the corresponding bone orientation was flagged as incorrect. In addition to the interpolated path, the recognition application also required the user to go through the positional "checkpoints" at each step. Each exercise (interpolation threshold independent) correctly identified the positional and angle combinations at each step as defined in the XML (with a buffer on the joint angles of 10^0). All 5 exercises completed this testing and performed proper recognition and feedback of the corresponding exercise. These results verified the generated data from the exercise generator application.

6 Conclusion

Within the limited development time, an automated exercise recording application was developed. The ability to rapidly record exercises for patients to use is a key component that is unique to this application, allowing therapist to continually grow the exercise database.

However, despite having a functional application, there are several limitations to the application, one of which includes the fact that some exercises require the rotation of an axis. These types of exercises would not be recorded accurately by this application. For example, picture a patient starting in a T position with his thumb facing upwards. If the exercise requires him to roll the thumb downwards without moving the rest of the body, the application would not be able to record this exercise since the rotation is rotating an axis (the outstretched arm). The quaternion changes would not be detected. Another limitation of this application is that it does not account for exercises that utilize the HipCenter joint. Although there are few physical therapy exercises that involve this joint (as most physical therapy exercises are isolated to specific body parts rather than full body [19][20]), it must be noted that this application does not currently account for these types of exercises; future work would include adding such detection. Additional future work would also involve implementing into the algorithm a method to re-record an exercise a specified number of times and average all of the data points taken. The algorithm currently only averages one demonstration of the exercise and may, in a sense, be overtraining to that specific instance of the exercise. With the exercise repeated several times, the program could incorporate machine learning to "learn" desired measurements such as the quaternion points, 3D data points, and joint combinations. The exercise definition also included threshold parameters that have been specified to allow for margins of error in the exercise steps. Currently, there is only a global threshold, but the future exercise structure would include error margins for each JointCombos sub-angle and movement thresholds for the rotational interpolation. Threshold parameters for each exercise could be determined using similar clustering techniques to those seen in Yeong et al. [5]. Steps would be divided into specified clusters and normalized deviations calculated for each exercise definition. The use of custom structured exercise thresholds would allow for variation among patients and greater overall accuracy on the actual physiotherapy application.

Through the design of the automated exercise generator, new insights were gained into the concepts of exercise definition and how to efficiently represent and record movement. The combination of 3D points, joint angles, and step quaternions allowed for the desired exercise to be easily recorded in XML format for use by the recognition application. Moreover, the use of the application extends beyond the realm of physiotherapy due to its ability to record desired movements rather than simply relying on a pre-compiled database. The generic qualities of this application combined with the quaternion interpolation extend beyond patient therapy and into the broad range of motion based learning.

References

[1] Norman, D. A. 2010.Interactions 3 (17): 6-10.

[2] "Natal Recognizes 31 Body Parts, Uses Tenth of Xbox 360 Computing Resources.", last modified September 6, 2012, accessed 12/9, 2012, http://kotaku.com/5442775/natal-recognizes-31-body-parts-uses-tenth-of-xbox-360-computing-resources.

[3] Shoemake, K. Quaternions. Philadelphia, PA: Department of Computer and Information Science, University of Pennsylvania.

[4] Fernandez-Aranda, F., S. Jimenez-Murcia, J. J. Santamara, K. Gunnard, A. Soto, E. Kalapanidas, R. G. A. Bults, C. Davarakis, T. Ganchev, and R. Granero. 2012. "Video Games as a Complementary Therapy Tool in Mental Disorders: PlayMancer, a European Multicentre Study." Journal of Mental Health 21 (4): 364-374.

[5] Yeong, C.R., Melendez-Calderon, A. and Burdet, E.. 2009. "Analysis of Pick-and-Place, Eating and Drinking Movements for the Workspace Definition of Simple Robotic Devices.".

[6] Hondori, H. M., M. Khademi, and C. V. Lopes. 2012. Monitoring Intake Gestures using Sensor Fusion (Microsoft Kinect and Inertial Sensors) for Smart Home Tele-Rehab Setting. Irvine, CA: University of California, Irvine.

[7] Chang, Yao-Jen, Shu-Fang Chen, and Jun-Da Huang. 2011. "A Kinect-Based System for Physical Rehabilitation: A Pilot Study for Young Adults with Motor Disabilities." Research in Developmental Disabilities 32 (6): 2566-2570.

[8] Lange, B., Chien-Yen Chang, E. Suma, B. Newman, A. S. Rizzo, and M. Bolas. 2011. "Development and Evaluation of Low Cost Game-Based Balance Rehabilitation Tool using the Microsoft Kinect Sensor.".

[9] Burba, N., M. Bolas, D. M. Krum, and E. A. Suma. 2012. ”Unobtrusive Measurement of Subtle Nonverbal Behaviors with the Microsoft Kinect.”.

[10] Matyunin, S., D. Vatolin, Y. Berdnikov, and M. Smirnov. 2011. ”Temporal Filtering for Depth Maps Generated by Kinect Depth Camera.”.

[11] Microsoft, ”Joint Type Enumeration.” Last modified October 01, 2012. Accessed May 17, 2013. msdn.microsoft.com.

[12] Soderkvist, Inge and Per-Ãke Wedin. 1993. ”Determining the Movements of the Skeleton using Well-Configured Markers.” Journal of Biomechanics 26 (12): 1473-1477.

[13] Mukundan, R. 2002. ”Quaternions: From Classical Mechanics to Computer Graphics, and Beyond.”2002.

[14] Johnson, M. 2003. ”Exploiting Quaternions to Support Expressive Interactive Character Motion.” Doctor of Philosophy, Massachusetts Institute of Technology.

[15] Lengyel, E. 2001. Mathematics for 3D Programming and Computer Graphics: Charles River Media.

[16] Cappozzo, A., F. Catani, U. Della Croce, and A. Leardini. 1995. ”Position and Orientation in Space of Bones during Movement: Anatomical Frame Definition and Determination.” Clinical Biomechanics 10 (4): 171-178.

[17] ”Joint Orientation.” Natural User Interface, msdn. msdn, accessed April, 1, 2013, http://msdn.microsoft.com/en-us/library/hh973073.aspx.

[18] Huang, J-D. 2011. ”Kinerehab: A Kinect-Based System for Physical Rehabilitation: A Pilot Study for Young Adults with Motor Disabilities.” The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility: 319-319-320.

[19] Bandy, W. and B. Sanders. 2008. Therapeutic Exercise for Physical Therapist Assistants, edited by P. Sabatini, A. Klingler. 2nd ed. Vol. 1. Baltimore, MD 21201: Lippincott Williams and Wilkins.

[20] Fox, E. L., Bowers, R. and Foss, M. 1993. The Physiological Basis for Exercise and Sport. Brown and Benchmark.

Appendix A

Joint Combinations

16 joint combinations were defined based on the joints detected by the Kinect. Each joint combination is made up of 3 joints which can be interpreted as 2 vectors with a common origin. The angle in between these vectors is the angle used in the JointCombo section of the exercise definition. The class that defines these 16 combinations can be seen below:

- Wrist Elbow Shoulder Right
- Wrist Elbow Shoulder Left
- Elbow Shoulder ShoulderCenter Right
- Elbow Shoulder ShoulderCenter Left
- Shoulder ShoulderCenter Head Right
- Shoulder ShoulderCenter Head Left
- Shoulder ShoulderCenter HipCenter Right
- Shoulder ShoulderCenter HipCenter Left
- ShoulderCenter HipCenter Hip Right
- ShoulderCenter HipCenter Hip Left
- HipCenter Hip Knee Right
- HipCenter Hip Knee Left
- Hip Knee Ankle Right

- Hip Knee Ankle Left
- Knee Ankle Foot Right
- Knee Ankle Foot Left

Limbs/Bones

11 limbs were defined based on the joints detected by the Kinect. Each limb consists of 2 joints and the Euclidean distance between these two joints. They are defined as follows:

- Forearmleft = ElbowLeft+WristLeft
- Forearmright = ElbowRight+WristRight
- Armright = ShoulderRight + ElbowRight
- Armleft = ShoulderLeft+ElbowLeft
- Shoulders = ShoulderLeft+ShoulderRight
- Torso = ShoulderCenter+HipCenter
- Hip = HipLeft+HipRight
- QuadLeft = HipLeft+KneeLeft
- QuadRight = HipRight+KneeRight
- ShinLeft = KneeLeft+AnkleLeft
- ShinRight = KneeRight+AnkleRight

Appendix B

Military Press

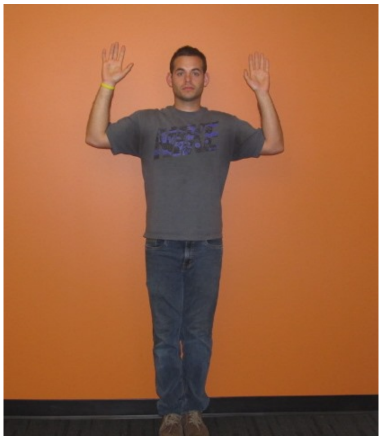


Figure 13: Step 1

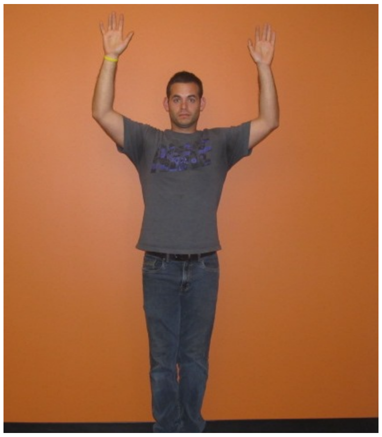


Figure 14: Step 2

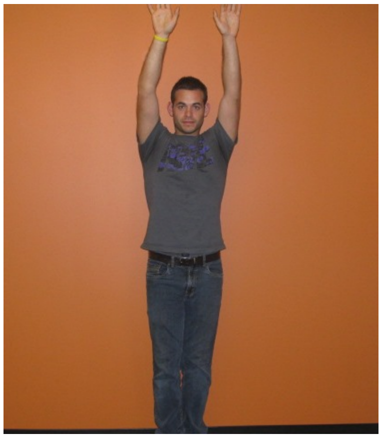


Figure 15: Step 3

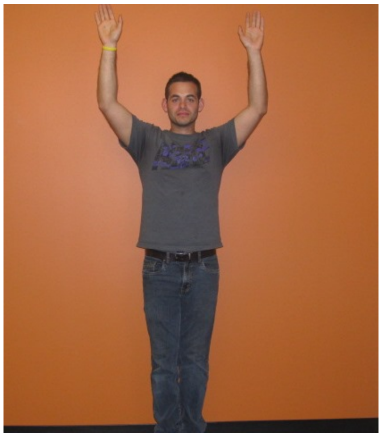


Figure 16: Step 4

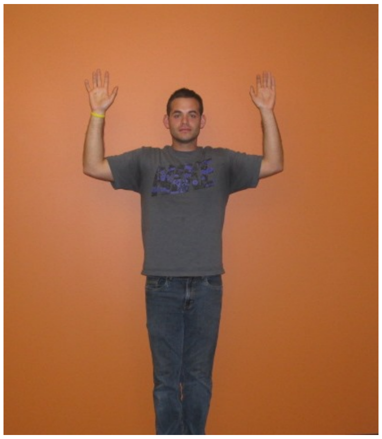


Figure 17: Step 5