

Spring 2012

Virtual Kung fu Sifu with Kinect

Naveen Kumar Keerthy
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Keerthy, Naveen Kumar, "Virtual Kung fu Sifu with Kinect" (2012). *Master's Projects*. 252.
DOI: <https://doi.org/10.31979/etd.d86z-hzmm>
https://scholarworks.sjsu.edu/etd_projects/252

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Virtual Kung fu Sifu with Kinect

A Writing Project
Presented to
The Faculty of the Department of Computer Science
San José State University

In Partial Fulfillment of the
Requirements for the
Degree Master of Computer Science

By
Keerthy, Naveen Kumar
May 2012

© 2012
Naveen Kumar Keerthy
ALL RIGHTS RESERVED

SAN JOSÉ STATE UNIVERSITY

Undersigned Writing Project Committee Approves the Writing Project Titled
Virtual Kung fu sifu with a Kinect

By
Keerthy, Naveen Kumar

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Chris Tseng, Department of Computer Science 05/21/2012

Dr. John Pearce, Department of Computer Science 05/21/2012

Dr. Soon Tee Teoh, Department of Computer Science 05/21/2012

Acknowledgement

I take it as a great honor to thank Dr. Chris Tseng for his great guidance and belief in me, and also Dr. John Pearce for the idea and sponsoring the equipment. I would like to acknowledge Dr. Soon Tee Teoh for his guidance. I would also like to thank Mr. Steven Macramalla for taking his time and helping me out with this project.

Abstract

In this computer world almost everything is made available online. All possible systems are making use of the Internet and so does the learning systems. People willing to learn any kind of arts, but have no time would look for a tutor who would be available at his/her ease. An attempt to satisfy the same is the project that I've implemented. My focus is mainly on Kung fu an ancient form of Chinese Martial arts. Learning Kung fu needs a lot of practice and therefore demands having a professional kung fu master monitoring all the time, which is very expensive. Therefore, I have developed a new way of learning experience by creating a virtual Kung fu master normally called as Sifu. In this application, players can learn and perfect different styles and techniques with the help of virtual Sifu. All these styles have been pre recorded by a professional Sifu using a motion sensing input device called Kinect. This Virtual Trainer is designed to help users learn and improve their martial arts by comparing the user's movements against those of a pre-recorded motion profile of Sifu by giving an immediate feedback to the user. Although many comparison algorithms exist I have chosen Dynamic Time Warping algorithm, which uses the Euclidean Distance formula. One of the main advantages of Dynamic Time Warping algorithm is, it overcomes the problems of motion analysis in speed and time.

Table of Contents

1. INTRODUCTION	1
2. THEORY AND CONCEPTS	3
3. TOOLS AND TECHNOLOGIES USED	8
4. DESIGNS AND IMPLEMENTATIONS	9
4.1 Overview.....	9
4.2 Capturing Master’s motions with two Kinects	10
4.2.1 Kinect.....	10
4.3 Recording BVH files with Miku Miku Capture	10
4.3.1 What is Miku Miku?	10
4.3.2 Connectivity between Miku Miku and animation tool Blender.....	17
4.4 Working with Blender to import the Miku Miku Dance recorded files.....	18
4.4.1 What is Blender Animation tool and why are we using it?.....	18
4.4.2 BVH file format	18
4.4.3 Rigging a character onto the skeleton	22
4.4.4 Output of the Blender.....	24
4.5 Creating the Virtual Training System using Kinect SDK for Windows	25
4.5.1 Visual Studios.	25
4.5.2 Kinect for WPF and C# in developing the Virtual Trainer System	25
4.6 Software Installation and un-installation	39
4.6.1 Software installation	40
4.6.2 Software un-installation	42
5. CONCLUSION	46
6. REFERENCES	47

Table of Figures

Figure 1: Comparisons between two patterns	5
Figure 2: Representations of comparison in matrix version	6
Figure 3: Values close to the output	7
Figure 4: Screenshot of Miku Miku after installation.....	11
Figure 5: Miku Miku recording with the help of Kinect	12
Figure 6: Miku Miku Recording default screen after setting.....	14
Figure 7: Mr. Steven looking at his recorded motions using MMDR	14
Figure 8: Mr. Steven performing for recording the master motions	15
Figure 9: Miku Miku while saving the BVH files	16
Figure 10: Miku Miku replaying the recorded motions.....	17
Figure 11: Representation of BVH file	19
Figure 12: Reading the BVH file using BVH viewer	22
Figure 13: Reading the BVH file using BVH viewer	23
Figure 14: Character rigging on the BVH skeleton file.....	23
Figure 15: Blender animation in all direction simultaneously.....	24
Figure 16: Another view of the Blender animation, with master and user characters next to Mr. Steven	25
Figure 17: Visual Studio output for Virtual Trainer system version 1	26
Figure 18: Blender Character motion with Kinect.....	27
Figure 19: Virtual Kung Fu Sifu Trainer System.....	28
Figure 20: Original Package	40
Figure 21: Uncompressed Package	40
Figure 22: Opened tar Package	41
Figure 23: Select the executable	41
Figure 24: Security Warning	42
Figure 25: Add / Remove Programs.....	43
Figure 26: Virtual Trainer entry	43
Figure 27: Uninstall / Change Button	43
Figure 28: Uninstall Virtual Trainer.....	44

1. INTRODUCTION

The advancements in technology increase the demand for time. If an individual wishes to learn martial arts, the major constraints would be time, money and availability of both learner and Kung Fu master. In order to overcome these obstacles and provide a better learning to the user a system is required, where the user doesn't require a physical presence of the Kung Fu master called as sifu. Instead he can teach himself from a pre-recorded video of a sifu with immediate feedback being provided. For this to happen, there needs to be a medium that can read the human motions and compare it to the pre recorded video. Kinect is a device that will sense the motion and also allows users to interact without any intermediary device. Kinect's camera has a face and gesture recognition sensor. Kinect was initially developed for gaming industry although after having its own advantages it made a move into virtual shopping, education and tele-health service.

The main focus of this project is on the immediate feedback of the virtual training system given to the user. The algorithm needs to compare both the motions of Kung fu Sifu and the learner. As Kinect recognizes the human body in the form of "Skeleton joints" All the skeleton data obtained from the Kinect can be stored in the form of (x, y, z) tuple. So I have considered many approaches to compare the data between the master motions and the user recorded motions. One approach will be considering Shadow Overlay, as Kinect captures the depth information of the user and generates a user's shadow. This shadow will be calibrated beforehand in such a way that it is scaled, so that it will be the same each time even if a new user plays. This way the virtual trainer system will be able to compensate for different users having different body styles. Users will then be judged based on the percentage of their shadow that currently encompasses the area. Real-time feedback can

allow the user to make corrections on the fly. But the approach that I have followed is by comparing the skeletal joints data using Dynamic Time Warping Algorithm (DTW). [1]

In the below sections of the paper, I will explain about the DTW algorithm, the various tools and the programming languages used with more details. I will describe how I have achieved my goals of creating an efficient way of learning. The entire process of creating the animations and the tools used will be explained in detail.

In the later sections of the paper “Theory and Concepts” section is used to explain about the Dynamic Time Warping Algorithm and its mathematical explanation. In “Tools and Technologies” section I will explain the animation tools and technology used in order to achieve my goal. The next section will be about the implementation of the project and the design ideas involved in the Virtual Training System.

2. THEORY AND CONCEPTS

To measure the similarities between any two sequences that vary in time and space, Dynamic Time Warping (DTW) algorithm is very effective. Even if there is some amount of delay in motion this algorithm can be used to identify the similarities. For example, in one video a person walks fast and in another video the person walks slow with the help of DTW algorithm we can identify the similarity in the pattern. The algorithm can be used on any type of data if the data is represented in the linear form. This algorithm can be applied in audio, graphics and also video.

Consider two signals that are encoded as evenly spaced sequential values, so to compare the signals normally, we can add differences in the frequency. If the two signals are aligned in a correct way there won't be any problem but the problem arises if there is a variation in the alignment of the signal. If one signal is compressed or stretched when it is compared with other signal we need to know which points to compare. The DTW algorithm to overcome the above problem uses dynamic programming logic.

The DTW algorithm uses a dynamic programming technique to solve this problem. The first step is to compare each point in one signal with every point in the second signal, generating a matrix. The second step is to work through this matrix, starting at the bottom-left corner (corresponding to the beginning of both signals), and ending at the top-right (the end of both signals): for each cell, the cumulative distance is calculated by picking the neighboring cell in the matrix to the left or beneath with the lowest cumulative distance, and adding this value to the distance of the focal cell. When this process is complete, the value in the top-right hand cell represents the distance between the two signals according to the most efficient pathway through the matrix.

In general, DTW is a method that allows a computer to find an optimal match between two

given sequences with certain restrictions. The sequences are "warped" non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension. This example illustrates the implementation of dynamic time warping when the two sequences are strings of discrete symbols. $d(x, y)$ is a distance between symbols, i.e. $d(x, y) = |x - y|$. [3]

Algorithmic representation of DTW:

```

int DTWDistance(char s[1..n], char t[1..m]) {
  declare int DTW[0..n, 0..m]
  declare int i, j, cost

  for i := 1 to m
    DTW[0, i] := infinity
  for i := 1 to n
    DTW[i, 0] := infinity
  DTW[0, 0] := 0

  for i := 1 to n
    for j := 1 to m
      cost := d(s[i], t[j])
      DTW[i, j] := cost + minimum(DTW[i-1, j ], // insertion of data
      DTW[i , j-1], // deletion of data
      DTW[i-1, j-1]) // match of data
  return DTW[n, m]
}

```

Example:

Suppose we wish to compare and evaluate the difference between the following two signals:

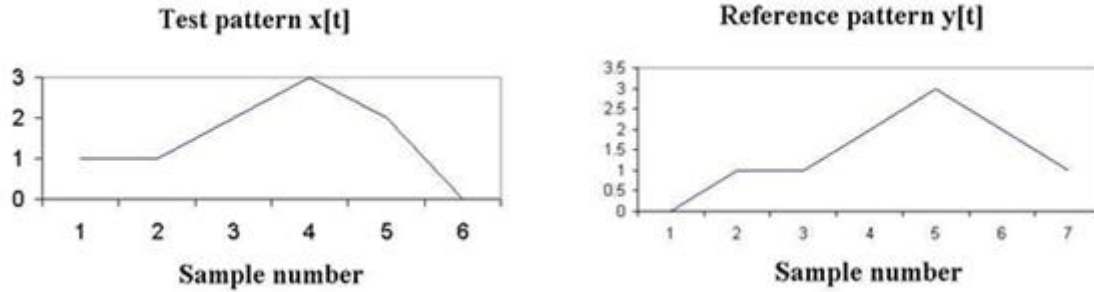


Figure 1: Comparisons between two patterns

a) (Input) test signal, x[t]	1	1	2	3	2	0	
b) (Stored) reference signal, y[t]	0	1	1	2	3	2	1
Sample-by-Sample difference, x[t]-y[t]	1	0	1	1	-1	-2	undefined

Both signals are similar in a way as both are single-peaked. However, the stored reference signal is longer than the test signal, and the peak is later. In other words, the two signals are not synchronized in time. To calculate the difference between them, consider a matrix of distance between every sample of x[t] and each sample of y[t]. The distance matrix D is:

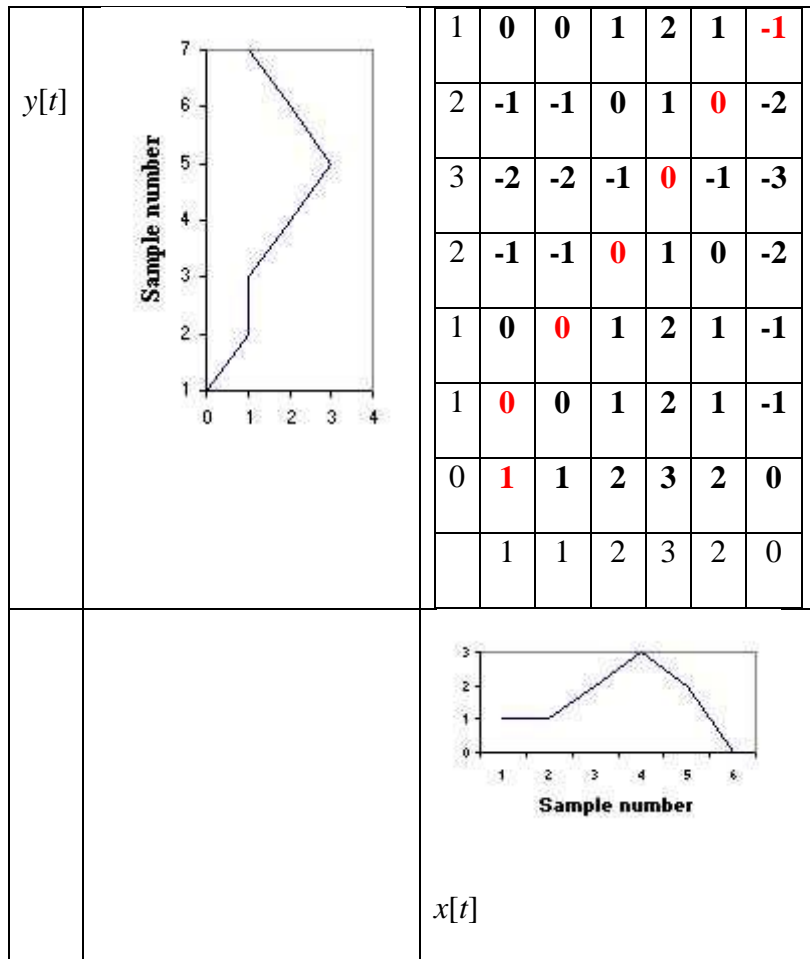


Figure 2: Representations of comparison in matrix version

There is a sequence of low numbers, close to the diagonal, indicating which samples of $x[t]$ are closest in value to those of $y[t]$. These are marked in red. Instead of a simple subtraction, it is customary to use a symmetrical distance measure, such as $(x[t] - y[t])^2$:

$y[t]$	<p>1 0 0 1 4 1 1</p> <p>2 1 1 0 1 0 4</p> <p>3 4 4 1 0 1 9</p> <p>2 1 1 0 1 0 4</p> <p>1 0 0 1 4 1 1</p> <p>1 0 0 1 4 1 1</p> <p>0 1 1 4 9 4 0</p> <p>1 1 2 3 2 0</p>
	$x[t]$

Figure 3: Values close to the output

3. TOOLS AND TECHNOLOGIES USED

Under this section, I will be briefing about the technologies and tools that I have used. This project demands a feel good interface. Microsoft has provided Kinect Software Development Kit for Windows, with which developers can create applications, which support gesture and audio recognition. To develop a Kinect application in Microsoft visual studios I have used C# as the back-end and Windows Presentations Forms (WPF) as the front-end. Kinect will display the recorded video of the Kung Fu master, but in order to get the gaming feel, a special 3D animation tool called Blender has been used. By using the animations created in the Blender we can display the video of the captured Sifu in the form of 3D animated video. For displaying the 3-D version of Sifu pre-recorder motions I have used “Miku Miku Dance Recorder”. The recorded data is in the form of a special file format called as Bio vision Hierarchical data (.BVH file)

4. DESIGNS AND IMPLEMENTATIONS

The flow of the project goes as follows. Firstly, record the Kung fu masters motions using two Kinects for capturing the data and also record the same motions using a digital camera for further reference. The second step involves capturing the motions using Miku Miku Dance recorder to get the BVH files so we can create the animations in Blender. Third step involves creating animation using the Blender animation tool. Fourth step involves comparing the Master recorded motion and capture the fresh recorded Kinect video of the user. The fifth and the final step is to compare both the motions and giving the feedback on the fly.

4.1 Overview

The following steps are followed to create a virtual training system after installing the software.

1. Capturing the Masters motion with two Kinects.
2. Recording BVH files with Miku Miku Capture.
3. Working with Blender to import the Miku Miku Dance recorded files.
4. Creating animation using the .BVH files and an animated character rig.
5. Creating the Virtual Training System using Kinect SDK for Windows and Microsoft Visual Studios.
6. Motion Analysis using Dynamic Time Warping Algorithm.

After following all these steps, the user will be able to get the live feedback of where exactly he is going wrong. This actually works on the fly when the user tries to perform the Kung fu. Based on the DTW algorithm the user can see RED colored joints on the screen respective to the body joints. It suggests that the user is doing the motion incorrectly. The GREEN colored joints suggest that the motion of the user is in sync with Sifu's motion.

4.2 Capturing Master's motions with two Kinects

4.2.1 Kinect

Kinect is the motion control device developed by Microsoft to be used with their Xbox 360 console. There are no gadgets to hold, swing, push or pull. The User will be the sole controller. Kinect lets user to interact with games and entertainment in a natural way using his/her body along with voice commands to control. Kinect comprises of a powerful set of sensors, a RGB camera and a depth sensor in order to capture the motion, a multi-array microphone to sense the audio and also a motorized pivot, which helps in full body 3D motion, capture enabling facial and voice recognition.[5]

Recording with Kinect

The recording of data is done with the help of two cameras. The recording from one camera would be used to create a virtual training system where as the recording from the second camera would be of the joint skeleton data. The master profile created from the former would be useful for comparing the master and user's motion.

4.3 Recording BVH files with Miku Miku Capture

4.3.1 What is Miku Miku?

Miku Miku Capture is Japanese animation software used to create motion for animated characters. This software has the option of connecting to Kinect and record the motions. Using this Kung fu master's skeleton data has been recorded. Miku Miku Capture has the capability of exporting the recorded video to a BVH format (Bio vision Hierarchical) for reuse. Few features that make this software robust include the ability to display characters in 3D space, simultaneous recording, and the ability to view the model from different

perspectives. Unlike Miku Miku Dance, this allows for a easily usable BVH files instead of the need to convert Vocaloid Motion.[7]

Data (VMD).

4.3.2 Installation, setup and running

1. Go to http://www.geocities.jp/higuchuu4/index_e.htm and download both the DxOpenNI driver and the DirectX9 Ver. of Miku Miku Dance.
2. Once downloaded, Miku Miku dance requires no additional installation. Extract the archive to a location for use.
3. Before starting usage of MMD, we should find the “data” folder contained with the MMD package (Should be found in a path similar to C://.../MikuMikuDance/Data) and paste DxOpenNI to for Kinect compatibility
4. Once the above steps are done, we need to open Miku Miku dance and make sure that the Kinect is already plugged into the system. Now we should be able to see Figure 4 on the screen.

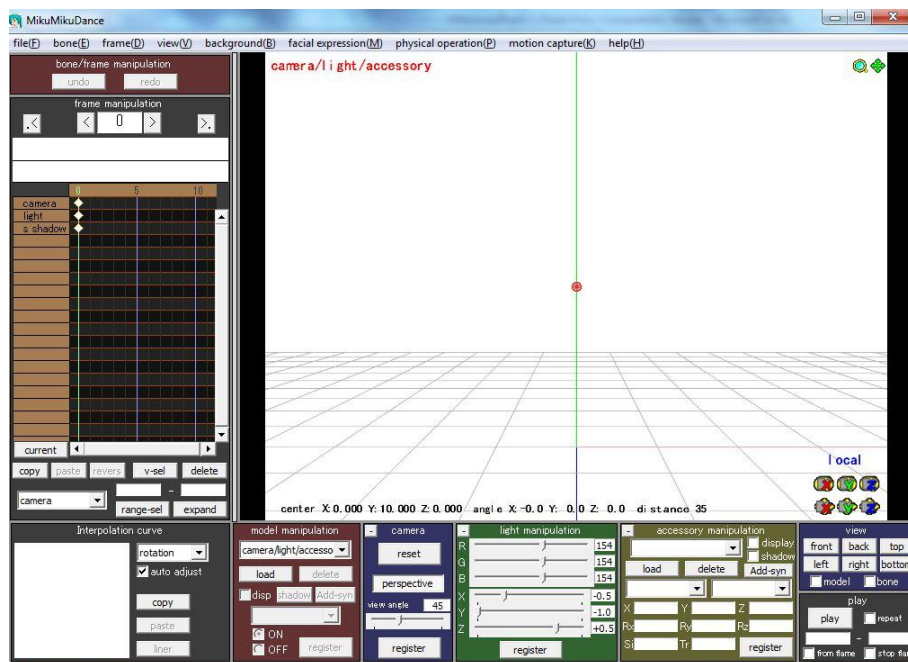


Figure 4: Screenshot of Miku Miku after installation

5. Now we need to click the box named “load” under “model manipulation”, and load any of the displayed models into the window, *.PMD. This will be the skeleton-tracking avatar.
6. Need to check the “Kinect” option from the toolbar and click “motion capture”
7. Next after enabling the “only playtime” option, the avatar should display in the upper right hand corner a player and a mimicking model. This fulfills active viewing and tracking of motion.
8. To record motion, go to the toolbar “motion capture” and choose “capture”. A countdown begins and whatever motion is activated, and it actively records the motion profile until “capture” is once again clicked. The current motion profile is now recorded.

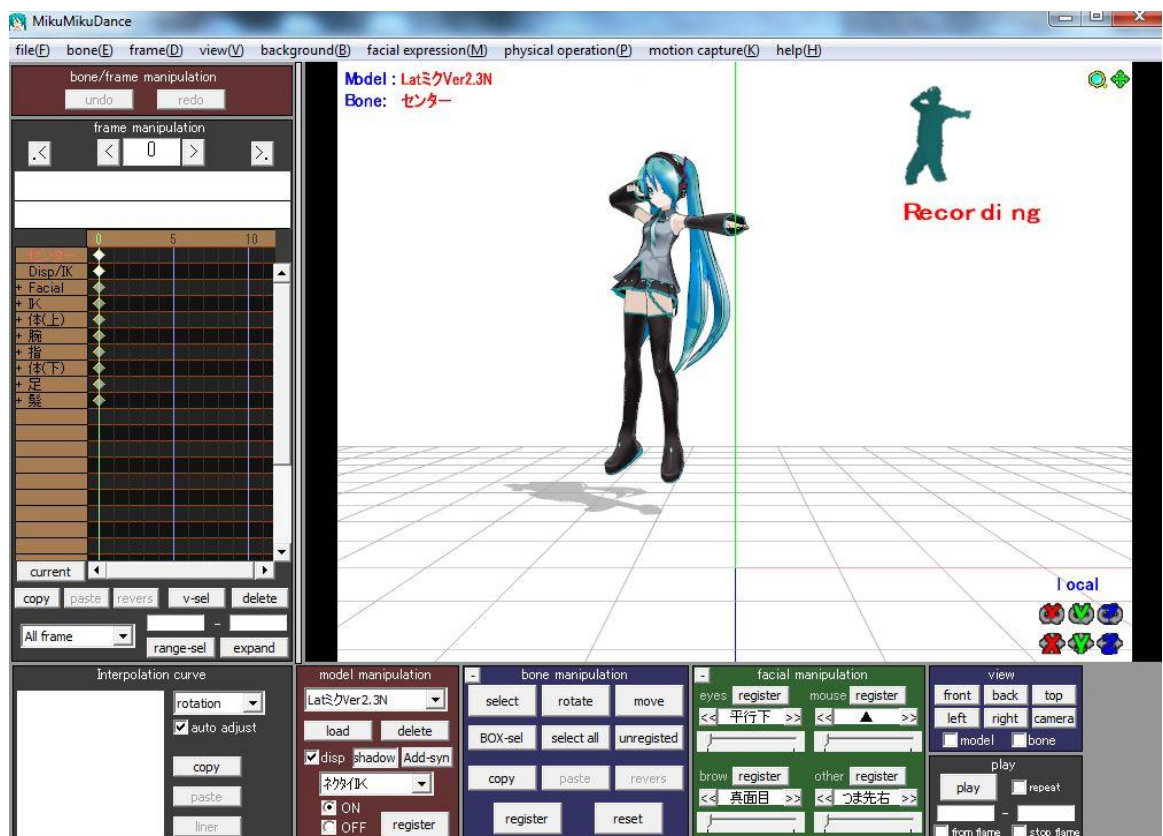


Figure 5: Miku Miku recording with the help of Kinect

9. We need to save our motion profile as a *.VMD file. This can be used with Blender after usage of 2 plug-in components and conversion of the data. We can replay the actions recorded by pressing the “Play” button and the character avatar (Miku) will replay the motions and key points.

4.3.3 Recording BVH files

Step 1: Setting up the rig

We need to first open Miku Miku capture and turn on the Kinect capture ability. Then we load the model and change its attributes so that it can capture the motion. Please make sure that you have all the necessary components to utilize the NITE architecture and the non-Microsoft SDK.

Step 2: Capturing

Within MMC (Miku Miku Capture), after the things are configured correctly load a new avatar model from the folder of Miku Miku dance, by clicking the load button in the control panel the appropriate model is loaded. If everything is setup correctly, then we need to go to Capture. Once the Kung Fu master is in front of the Kinect calibrate the virtual master with the real one. At any point of time we can click Begin Recording and recording will start.

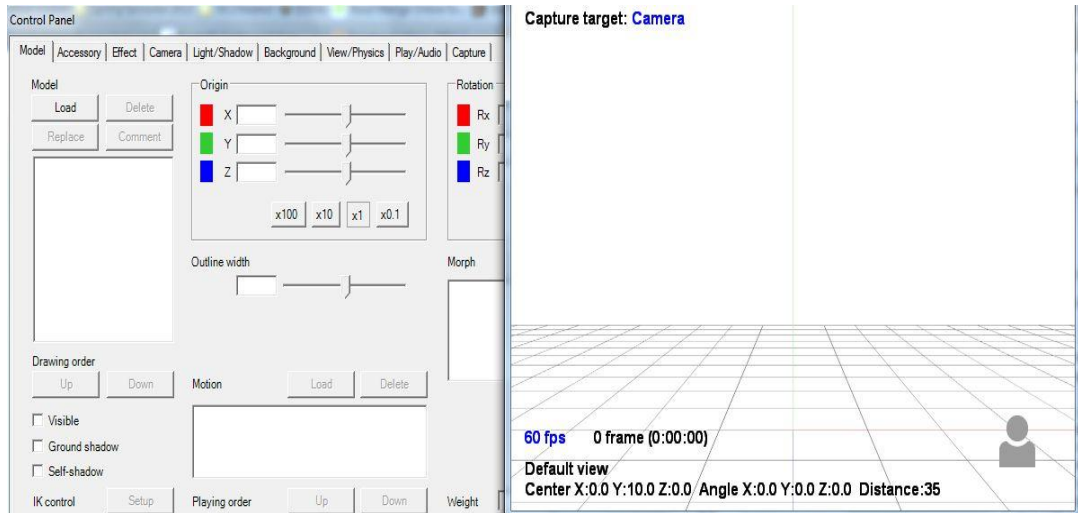


Figure 6: Miku Miku Recording default screen after setting

Step 3: Export Data

Save the motion to get desired format either BVH or VMD data for Blender and MMD respectively. Change the settings and the origin of recording by manipulation of the X/Y/Z planes. Motion capture of the main profile is successful.

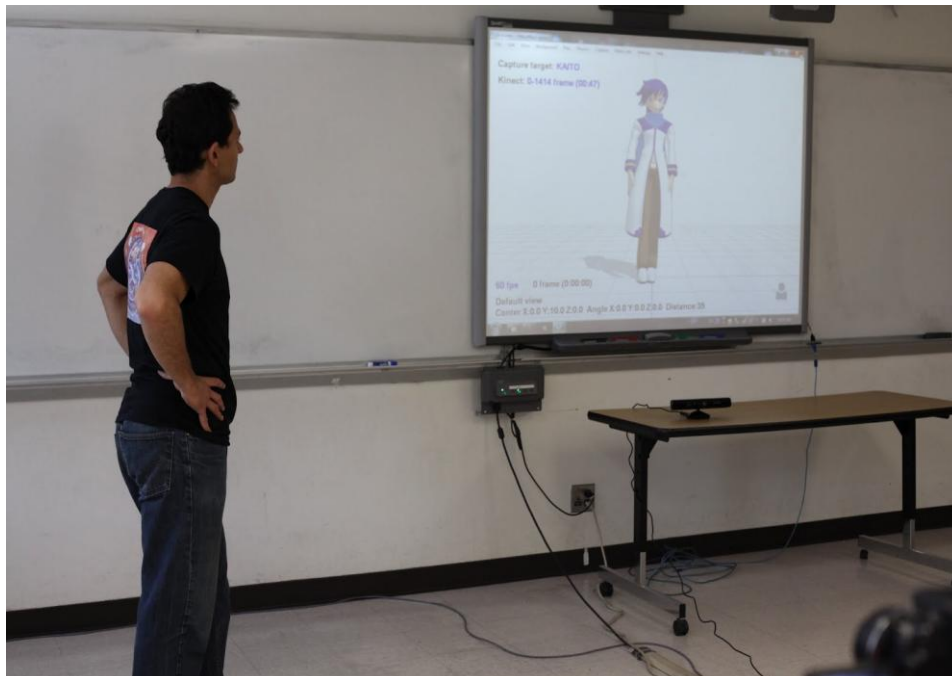


Figure 7: Mr. Steven looking at his recorded motions using MMDR

Step 4: Capturing Sifu's motion.

Take motion capture data from the “master”. Using MMC we capture the masters motion and save it as a VMD file. This is a great success with the aid of Miku Miku dance.

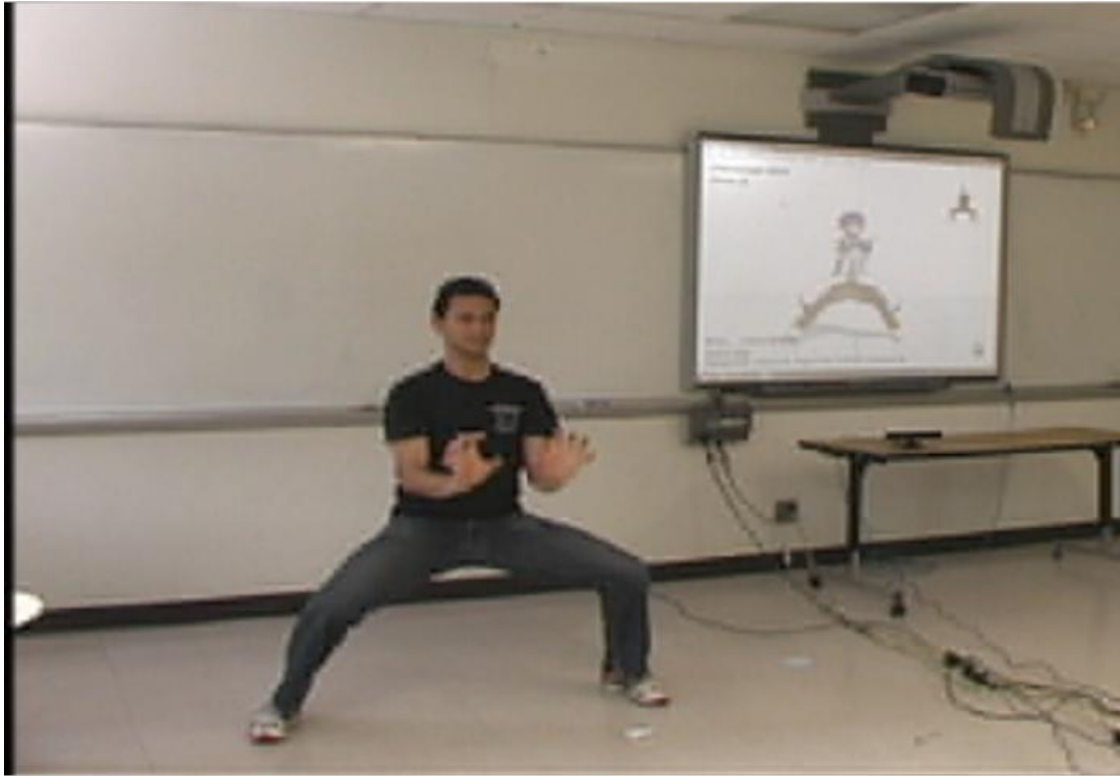


Figure 8: Mr. Steven performing for recording the master motions

Step 5: Importing file to Miku Miku Dance

We import the file into MMD and rig up the captured motion data to one of our rigs.

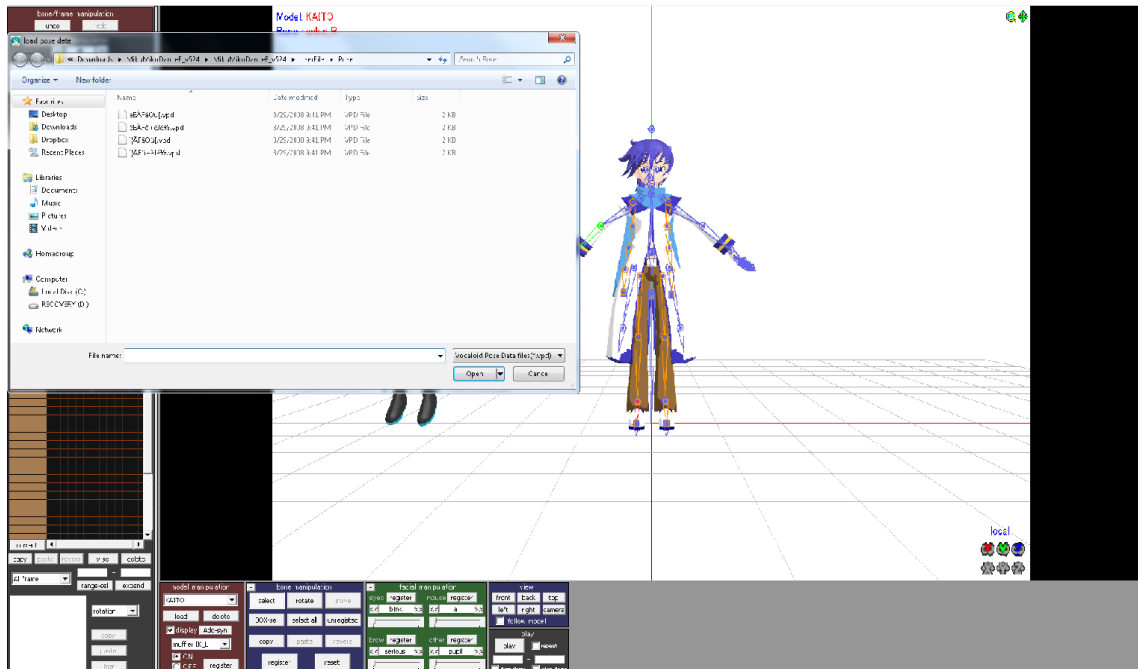


Figure 9: Miku Miku while saving the BVH files

Step 6: Setup a “user” rig so that players can have an avatar on the same screen.

1. Load one of the Kung Fu master motion profiles.
2. Attach the motion data to the master model
3. Make the user model take input from the Kinect camera.



Figure 10: Miku Miku replaying the recorded motions

4.3.4 Connectivity between Miku Miku and animation tool Blender

Miku Miku dance connectivity with Kinect allows for extremely straightforward recording and visual editing, as well as playback of previous *.VMD models, which can be finalized locally, then utilized in Blender for a final product. It is also a free Japanese program that has been translated into English for usage with motion captures, and currently developing a capture-only client called “Miku Miku Capture”. The software originally built to create digital music video graphics with character avatars around popular Japanese pop-culture character Yamaha Vocaloid Voice-Synthesizing Program Hatsune Miku has been further developed as a tool in which to capture motion data and fine-tune any errors in the physics or motion of an avatar build. Playback, bone viewing, custom model importing, as well as keyframe-by-keyframe analysis and changes are possible with extremely low processing power and resource usage. The real time motion

profile is displayed in the upper right hand corner, the model that follows along is easily recordable for use. The only constraint that Miku Miku has is its compatibility with other technologies. Miku Miku has an excellent working functionality with blender whereas, it isn't much compatible with rest of the technologies available online. Although it's a free tool available because of this drawback it is not widely used. The possible areas for improvements in this software are one being compatibility as discussed before and the other being installation process and Language support. English translation of their native language is not completely provided. There are also methods of converting models created for MMD into Blender for usage in motion avatar profiling.

4.4 Working with Blender to import the Miku Miku Dance recorded files

4.4.1 What is Blender Animation tool and why are we using it?

Blender is free open source software used to create 3D image. It is compatible with most of the operating system with general public license. The main features of blender includes Particle Simulation, UV unwrapping, texturing, Skinning, Rigging, Compositing, 3D modeling, Animating, Rendering. Blender is a package of lot of features and is an open source hence used by most of the animators.[1]

4.4.2 BVH file format

Blender takes several forms of data as input. The format, which we are interested for this project, is Bio vision Hierarchical motion capture data (.bvh)

A BVH file is nothing but an ASCII file that contains motion capture data for three-dimensional characters; used by 3ds Max's Character Studio and other 3D animation programs to import rotational joint data.

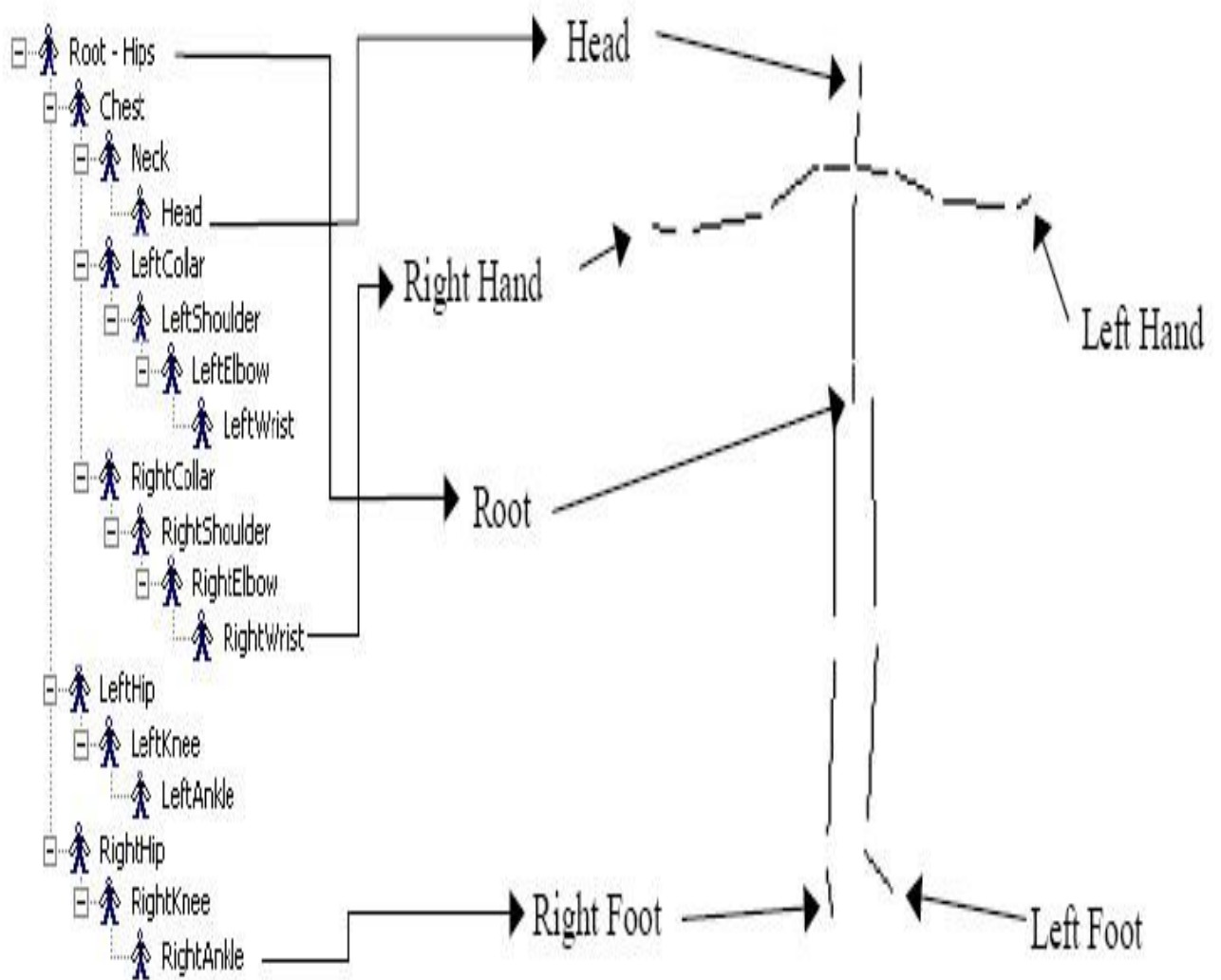


Figure 11: Representation of BVH file

BVH file structure

The BVH file is divided into two major sections: HIERARCHY and MOTION.

The HIERARCHY section describes the joint-to-joint connections and offsets for the sampled motion data. The MOTION section describes the movement of these individual joints on a per-sample basis.

HIERARCHY

ROOT Hips

```
{  
  OFFSET [x_float] [y_float] [z_float]  
  CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation  
  JOINT LeftHip  
  {  
    OFFSET [x_float] [y_float] [z_float]  
    CHANNELS 3 Zrotation Xrotation Yrotation  
    JOINT LeftKnee  
    {  
      OFFSET [x_float] [y_float] [z_float]  
      CHANNELS 3 Zrotation Xrotation Yrotation  
      JOINT LeftAnkle  
      {  
        OFFSET [x_float] [y_float] [z_float]  
        CHANNELS 3 Zrotation Xrotation Yrotation  
        End Site  
        {  
          OFFSET [x_float] [y_float] [z_float]  
        }  
      }  
    }  
  }  
}
```

Below Hips there is Right Hip, Chest, Right Collar and Neck. They have similar structure.

MOTION

Frames: [number_of_time_samples_to_follow]

Frame Time: [0.033333]

[samp1_chan1_float] [samp1_chan2_float] ... [samp1_chanN_float]

[samp2_chan1_float] [samp2_chan2_float] ... [samp2_chanN_float]

...

[sampN_chan1_float] [sampN_chan2_float] ... [sampN_chanN_float]

There will be 63 points consisting of x,y,z position and x,y and z rotation

An example of BVH file skeleton containing frame points

```
-8.489557 4.285263 -0.621559 -8.244940 -1.784412 90.041962 8.849357 5.557910 -  
1.926571 -5.487280 4.119726 -4.714622 -5.790586 -15.218462 -3.167648 -15.823254  
3.871795 -4.378940 22.399654 2.244878 -29.421873 -6.918557 6.131992 4.521327 -  
18.013180 3.059388 -3.768287 8.079588 10.124812 5.808083 -22.417845 -15.736264  
18.827469 -8.070700 9.689109 2.417364 -7.600582 2.505005 -1.625679 2.430162 -  
27.579708 -3.852241 -1.830524 12.520144 -1.653632 -2.688550 4.545600 0.296320  
8.031574 13.837914 -28.922058 2.077955 -9.176716 7.166249 -5.170825 -13.814465  
4.309433
```

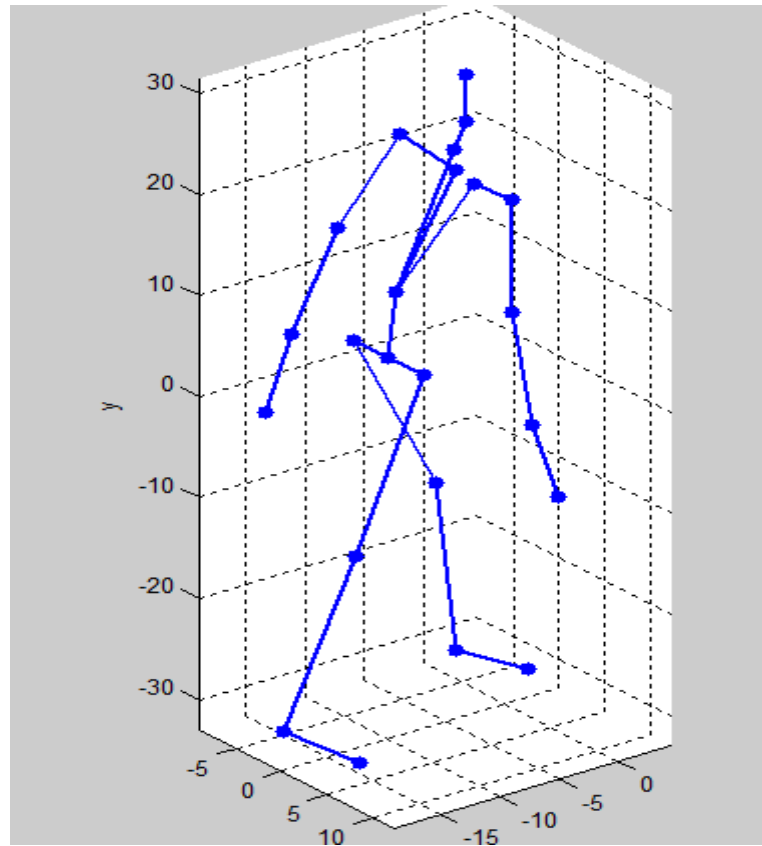


Figure 12: Reading the BVH file using BVH viewer

Based on this information we can understand the BVH file and using this we can create animated character in Blender.

4.3.3 Rigging a character onto the skeleton

From the BVH file we will be able to get the animation but it is limited only to skeletal motion. To give an animated look to the Sifu, a character has been added onto the BVH file data. This process is known as Rigging. Before rigging the BVH file information can be viewed as shown in the figure 13

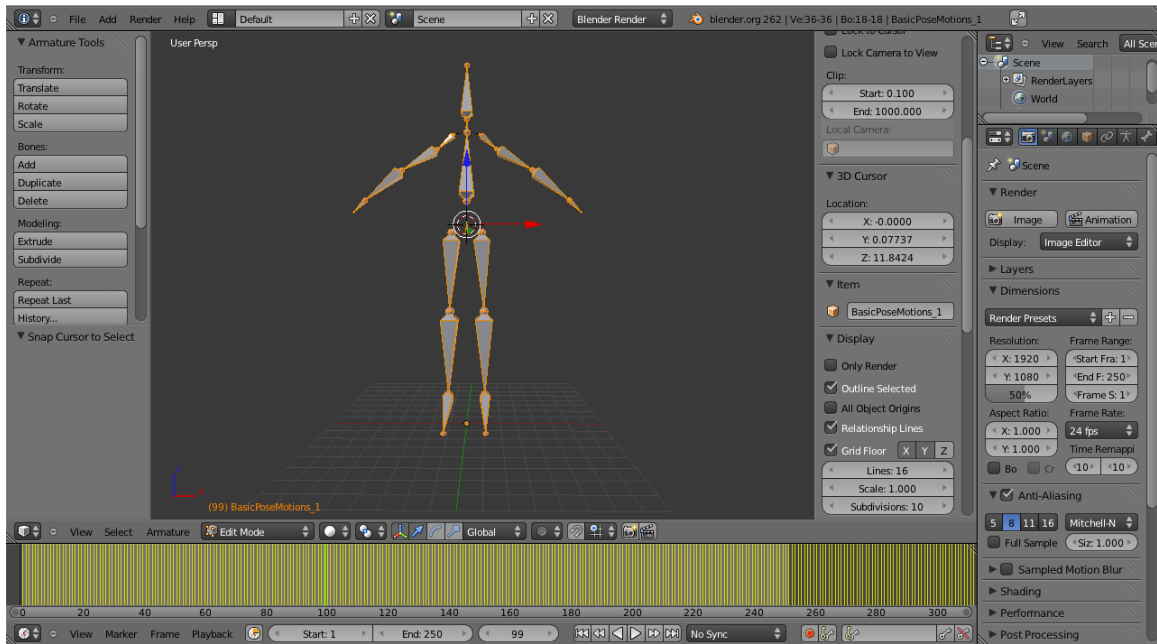


Figure 13: Reading the BVH file using BVH viewer

We can create our own avatar of any character-using Blender and put it on the skeleton.

Figure 14 shows how it looks when the skeleton is merged with character.

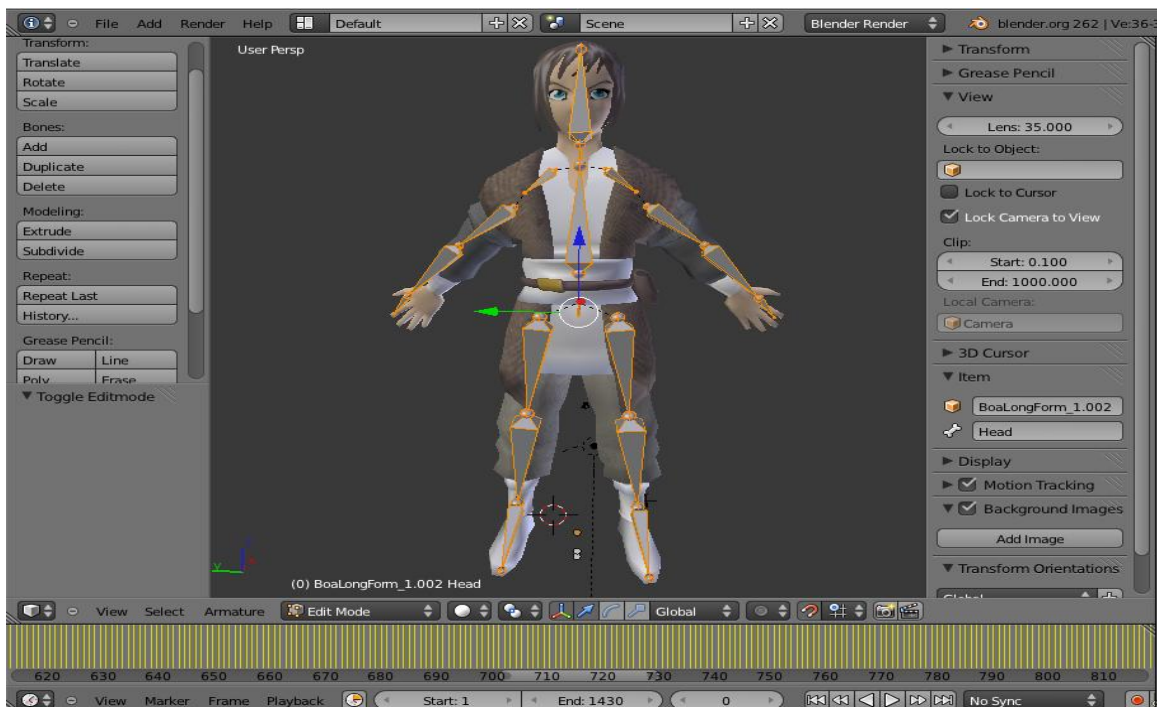


Figure 14: Character rigging on the BVH skeleton file

Once rigging process is completed, to create animation based on it the BVH file is saved in the form of .blend file and the animation action is later loaded onto the main window. The virtual training system demands output in several directions and hence the same rigged character is used for all the animations.

4.4.4 Output of the Blender

Once the animated character is ready, each kung fu move is animated each video is exported to the virtual training system. The main advantage of this virtual training system is that the master's moves are displayed in several directions. (side view, top view, front and rear views).



Figure 15: Blender animation in all direction simultaneously

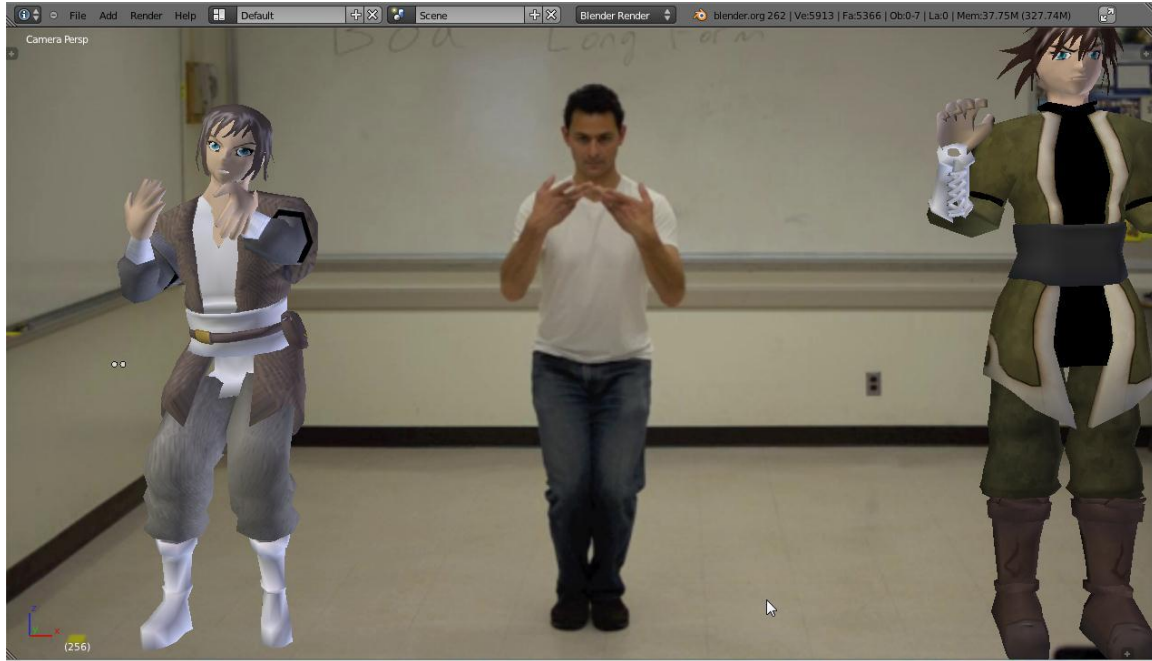


Figure 16: Another view of the Blender animation, with master and user characters next to Mr. Steven

4.5 Creating the Virtual Training System using Kinect SDK for Windows

4.5.1 Visual Studios.

SDK of Kinect sensor technology for Windows allows programmers to create several applications that support gesture and voice recognition. Kinect for Windows SDK is supported by Windows 7 and Windows 8 only. [8]

4.5.2 Kinect for WPF and C# in developing the Virtual Trainer System

The DTW functionality has been added into the code and made the gesture recognition possible by combing it with the Kinect Toolbox. The record gestures can be saved with a time period of 3 seconds. While replaying the recorded gestures we can try to check what gesture it is exactly and try to print it on the screen. This will help in future works.

Here from the figure17 It is clear that the recorded gestures and the live gestures are exactly the same.

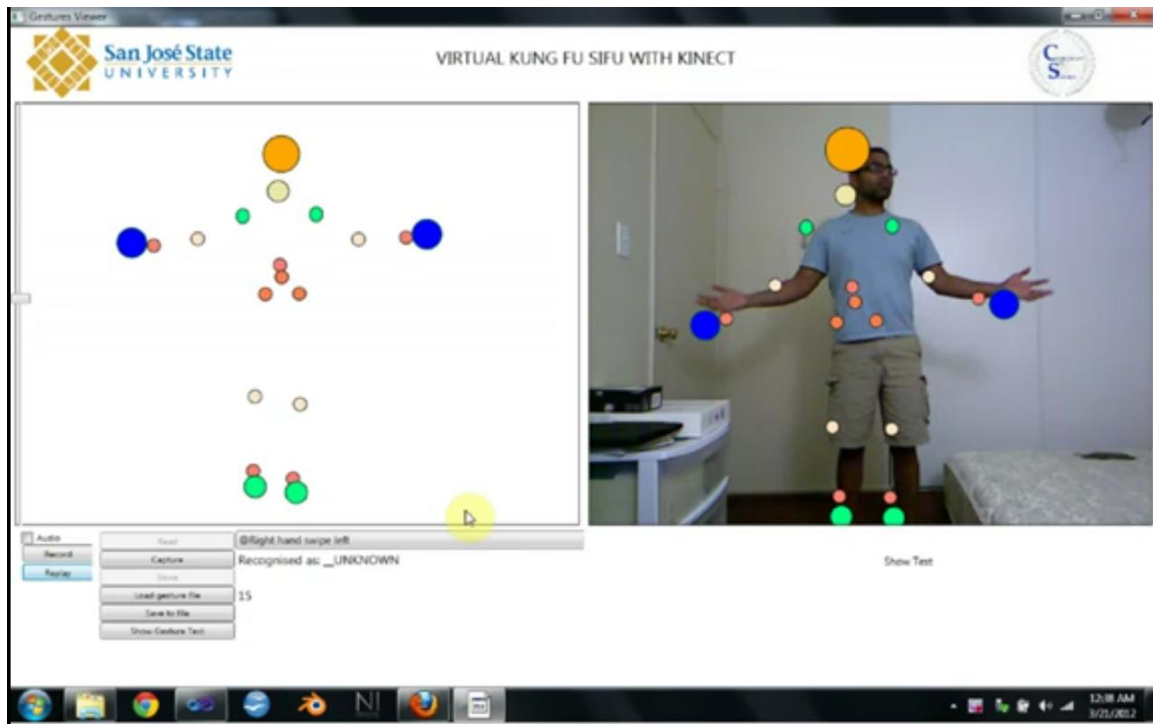


Figure 17: Visual Studio output for Virtual Trainer system version 1

Future work on Visual Studios for animation:

A Blender character has been added to the Microsoft Visual Studios in multiple Steps from figure18 we can see that the response of the blender character to the gestures of the user.

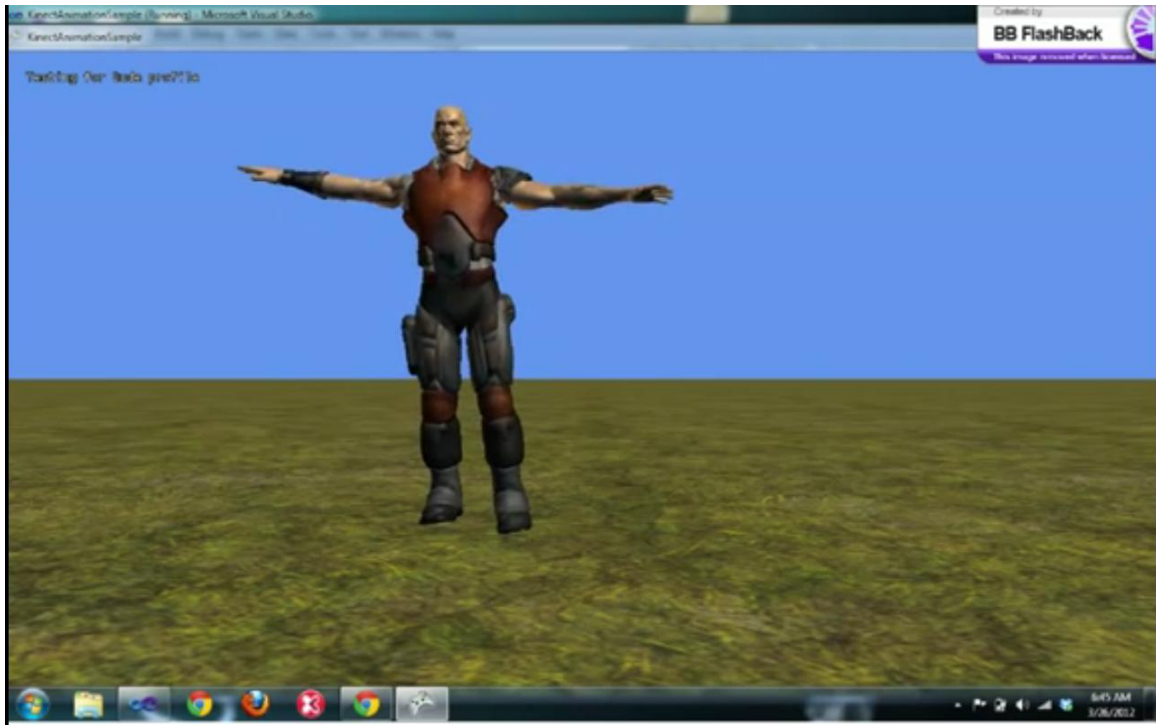


Figure 18: Blender Character motion with Kinect

To make the blender character work along with Kinect and Microsoft visual studios, few steps have to be followed.

1. Firstly, a Blender character is needed which is available for download at Blendswap.org and to import into Visual Studio
2. Secondly, get the XNA code compatible to work with Microsoft Visual Studios.

Copy the Blender character data (which will be in .fbx format), we can **get** import as many characters as possible. The video can be copied from blender and can be split or merge the videos according to the requirement.

In order to connect XNA code to the Kinect we make use of DigitRune. It is used in integrating blender character with XNA and Kinect.

As a future implementation I was planning to make the Game motion move onto WPF. As the previous project (Kinect Toolbox and DTW algorithm was completely base on the WPF). The WPF can have controls on the screen, assumed that this will be ideal for the project purpose. But the complete movement from Blender to XNA to Kinect SDK to WPF can be replaced with the Blender itself by adding coding style in Python.

Virtual Training System after implementing the animation

After release of multiple version and through testing we decided that the version depicted in Figure 19 is more user friendly.



Figure 19: Virtual Kung Fu Sifu Trainer System

This Virtual Training System allows the learners to see themselves on the system and follow the Kung Fu master. The Master screen on the left side has 4 variations in its camera angles and learner can select any of them. Once the Master video starts, user needs to follow the master and user can simultaneously look at himself on the other end of the screen with feedback.

Pseudo code for front-end

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog ofd;
    ofd = new OpenFileDialog();
    ofd.AddExtension = true;
    ofd.DefaultExt = ".*.*";
    ofd.Filter = "Media (*.*)|*. *.*";
    ofd.ShowDialog();
    try
    {
        master_window.Source = new Uri(ofd.FileName);

        camera_path = ofd.FileName;
        front_path = ofd.FileName.Substring(0,(ofd.FileName.Length)
            - 4) + "_front.mov";
        side_path = ofd.FileName.Substring(0 ,(ofd.FileName.Length)
            - 4) + "_side.mov";
        round = ofd.FileName.Substring(0,(ofd.FileName.Length) - 4)
            + "_round.mov";

        master_window.LoadedBehavior = MediaState.Manual;
        master_window.UnloadedBehavior = MediaState.Manual;
    }
}
```

```

    }
    catch
    {
        new NullReferecneException("Error");
    }
}

```

The code is designed in such a way that the front end operates on the click of a button and fetches the video that the user wants. An option to load various forms of Kung fu recordings from the master and corresponding directions is also provided. A comparison between the actual video and the user performing can be done at any time of the video. All the videos are animated and a rig character will be performing. Based on this the feedback is given to the users.

```

using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
{
    if (skeletonFrame != null)
    {
        Skeleton[] data = new Skeleton[skeletonFrame.SkeletonArrayLength];
        skeletonFrame.CopySkeletonDataTo(data);

        foreach (Skeleton skeleton in data)
        {
            if (skeleton.TrackingState == SkeletonTrackingState.Tracked)

```

```

{
SkeletonPoint point = skeleton.Joints[JointType.Head].Position;

writer.Write("Head: " + point.X + " " + point.X + " " + point.Y

point = skeleton.Joints[JointType.ShoulderCenter].Position;

writer.Write("ShoulderCenter: " + point.X + " " + point.X + " " +
point.Y + "\r\n");

point = skeleton.Joints[JointType.ShoulderRight].Position;

writer.Write("ShoulderRight: " +
point.X + " " + point.X + " " + point.Y + "\r\n");

point = skeleton.Joints[JointType.ElbowRight].Position;

writer.Write("ElbowRight: " + point.X
+ " " + point.X + " " + point.Y + "\r\n");

point =
skeleton.Joints[JointType.WristRight].Position;

writer.Write("WristRight: " + point.X
+ " " + point.X + " " + point.Y + "\r\n");

point =
skeleton.Joints[JointType.HandRight].Position;

writer.Write("HandRight: " + point.X +
" " + point.X + " " + point.Y + "\r\n");

point =
skeleton.Joints[JointType.ShoulderLeft].Position;

writer.Write("ShoulderLeft: " +

```

```

point.X + " " + point.X + " " + point.Y + "\r\n");

        point =
skeleton.Joints[JointType.ElbowLeft].Position;

        writer.Write("ElbowLeft: " + point.X +
" " + point.X + " " + point.Y + "\r\n");

        point =
skeleton.Joints[JointType.WristLeft].Position;

        writer.Write("WristLeft: " + point.X +
" " + point.X + " " + point.Y + "\r\n");

        point =
skeleton.Joints[JointType.HandLeft].Position;

        writer.Write("HandLeft: " + point.X +
" " + point.X + " " + point.Y + "\r\n");

        point =
skeleton.Joints[JointType.Spine].Position;

        writer.Write("Spine: " + point.X + " "
+ point.X + " " + point.Y + "\r\n");

        point =
skeleton.Joints[JointType.HipCenter].Position;

        writer.Write("HipCenter: " + point.X +
" " + point.X + " " + point.Y + "\r\n");

        point =
skeleton.Joints[JointType.HipRight].Position;

        writer.Write("HipRight: " + point.X +

```



```

" " + point.X + " " + point.Y + "\r\n");
        point =
skeleton.Joints[JointType.KneeRight].Position;
        writer.Write("KneeRight: " + point.X +
" " + point.X + " " + point.Y + "\r\n");
        point =
skeleton.Joints[JointType.AnkleRight].Position;
        writer.Write("AnkleRight: " + point.X
+ " " + point.X + " " + point.Y + "\r\n");
        point =
skeleton.Joints[JointType.FootRight].Position;
        writer.Write("FootRight: " + point.X +
" " + point.X + " " + point.Y + "\r\n");
        point =
skeleton.Joints[JointType.HipLeft].Position;
        writer.Write("HipLeft: " + point.X +
" + point.X + " " + point.Y + "\r\n");

        point =
skeleton.Joints[JointType.KneeLeft].Position;
        writer.Write("KneeLeft: " + point.X +
" " + point.X + " " + point.Y + "\r\n");
        point = skeleton.Joints[JointType.AnkleLeft].Position;
        writer.Write("AnkleLeft: " + point.X +

```



```

int N = StudentArray.Length;

var DTW = new double[M + 1,N + 1];

//Initial matrix

DTW[0, 0] = 0;

for (int j = 1; j <= M; j++)

{

    DTW[j, 0] = double.PositiveInfinity;

}

for (int i = 1; i <= N; i++)

{

    DTW[0, i] = double.PositiveInfinity;

}

//End of Init

for (int i = 1; i <= M; i++)

{

    for (int j = 1; j <= N; j++)

    {

        double cost = EuclideanDistance(MasterArray[i - 1], StudentArray[j - 1]);

        DTW[i, j] = cost + Math.Min(DTW[i - 1, j],    //insertion

                                   DTW[i, j - 1],    //deletion

                                   DTW[i - 1, j - 1]); //match

    }

}

return DTW[M, N];

```

```
}
```

Walkthrough of the pseudocode


```
public static double DTWdistance(MasterArray[], StudentArray[])
```

MasterArray[] contains the (pre-recorded) locations of the Master's left elbow joint (for example) over a certain amount of time or a certain number of frames.

MasterArray[] =

(X1, Y1, Z1)	(X2, Y2, Z2)	(X3, Y3, Z3)	(X4, Y4, Z4)	(X5, Y5, Z5)
--------------	--------------	--------------	--------------	--------------

Frame number / time




StudentArray[] contains the locations of the Student's left elbow joint.

StudentArray[] =

(x1, y1, z1)	(x2, y2, z2)	(x3, y3, z3)	(x4, y4, z4)
--------------	--------------	--------------	--------------

Frame number / time



Note: the 2 arrays don't have to have the same length.

```
int M = MasterArray.Length;  
int N = StudentArray.Length;  
var DTW = new double[M + 1, N + 1];  
  
//Initial matrix
```

```

DTW[0, 0] = 0;
for (int j = 1; j <= M; j++)
{
    DTW[j, 0] = double.PositiveInfinity;
}
for (int i = 1; i <= N; i++)
{
    DTW[0, i] = double.PositiveInfinity;
}

```

create a matrix DTW to store the DTW distance values.

DTW =

0	infty	infty	infty	infty
infty				
infty				
infty				
infty				
infty				

```

for (int i = 1; i <= M; i++)
{
    for (int j = 1; j <= N; j++)
    {
        double cost = EuclideanDistance(MasterArray[i - 1], StudentArray[j - 1]);
    }
}

```

```

DTW[i, j] = cost + Math.Min(DTW[i - 1, j],    //insertion
                            DTW[i, j - 1],    //deletion
                            DTW[i - 1, j - 1]); //match
}
}

```

Populating the DTW matrix. The diagonal entries (and those close to it) will be smaller values since the joint positions will match more closely here (e.g., Frames 2 and 3, Frames 4 and 4, etc.). Entries farthest from the diagonal will be larger values (since the Student’s joint at Frame 5 will most likely not match the Master’s joint at Frame 1).

Recall MasterArray and StudentArray entries look like:

$(X_i, Y_i, Z_i) \leftarrow \text{master}$

$(x_i, y_i, z_i) \leftarrow \text{student}$

Then the pseudocode for the Euclidean Distance b/w the points MasterArray[i] and StudentArray[i] is $\text{EuclideanDistance} = \text{SQRT}((X_i - x_i)^2 + (Y_i - y_i)^2 + (Z_i - z_i)^2)$

The Math.Min arguments are the diagonal entry and its adjacent neighbors. This comparison is the main point of DTW – it deals with the case where the student lags a bit behind the master or is not completely in sync with him. Basically, those that match closest to the Master (shortest distance) is added to Cost (the EuclideanDistance) and entered in/near the diagonal.

DTW =

0	infty	infty	infty	infty
Infty				
infty				
Infty				
Infty				
Infty				

The important DTW values (where the Student matches closest to the Master) are gradually added to entries along the diagonal.

=====

```
return DTW[M, N];
```

We return the last diagonal entry (DTW[M, N]) which we use to compare against a threshold to see how close the Student's particular joint is to the Master's.

$DTW[M, N] < \text{threshold}$ or whatever.

4.6 Software Installation and un-installation

4.6.1 Software installation

1. Unpack the Group1Milestone3PartB_Virtual Trainer.tar.gz package in the temporary directory of your choice.

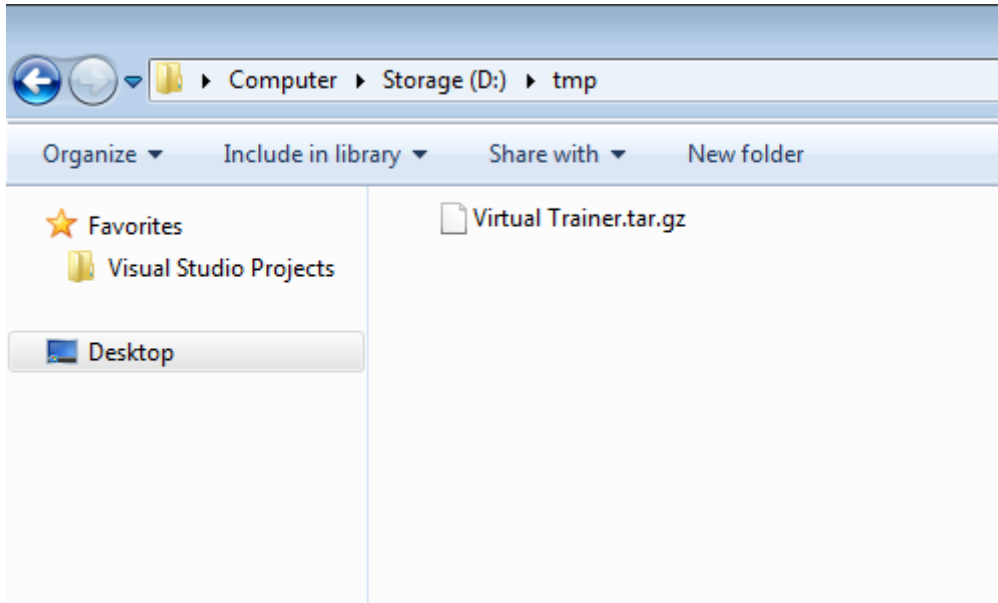


Figure 20: Original Package

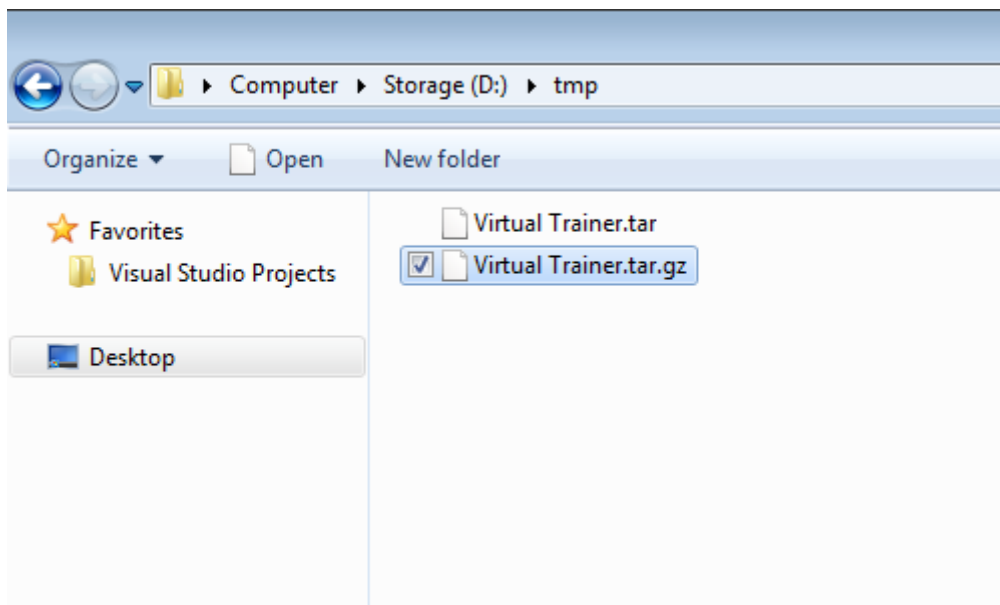


Figure 21: Uncompressed Package

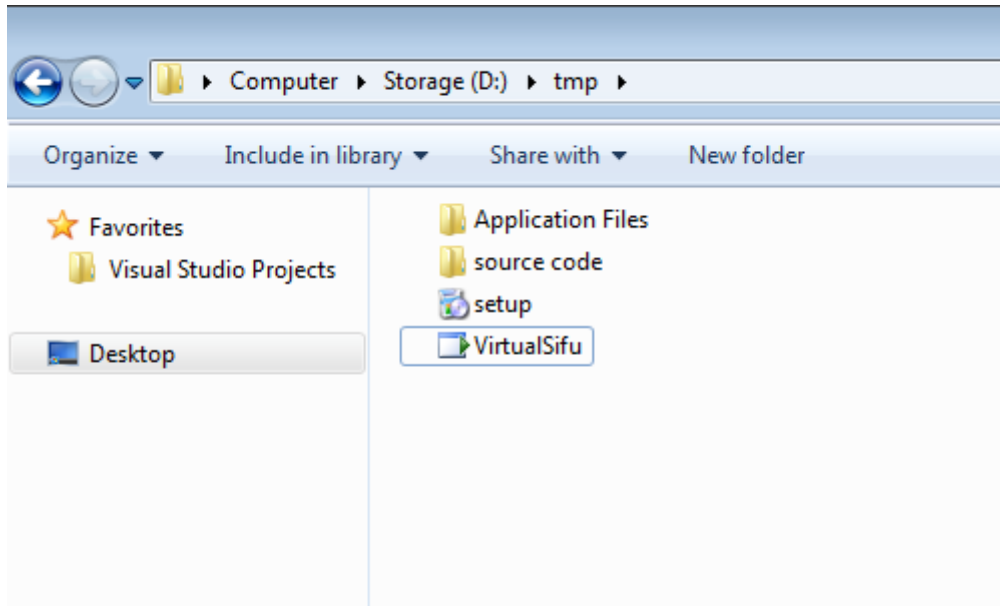


Figure 22: Opened tar Package

2. Run the setup executable.

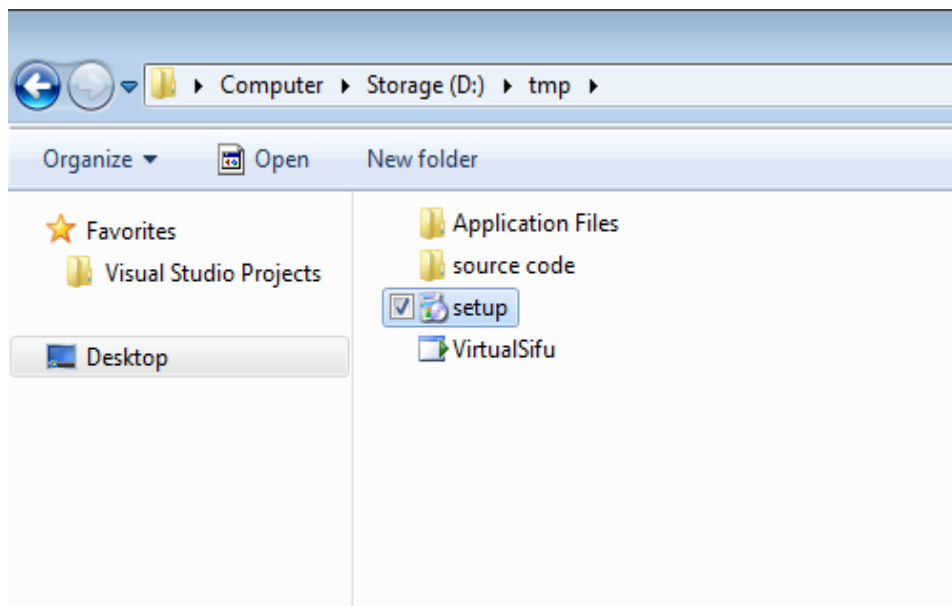


Figure 23: Select the executable

3. Select Install when the Application install - Security Warning appears.

This appears because Virtual Trainer is an in-house developed application that has been published to give users an executable rather than require them to compile and run the source code. However, this also means that the executable has no publisher information that Windows will recognize.

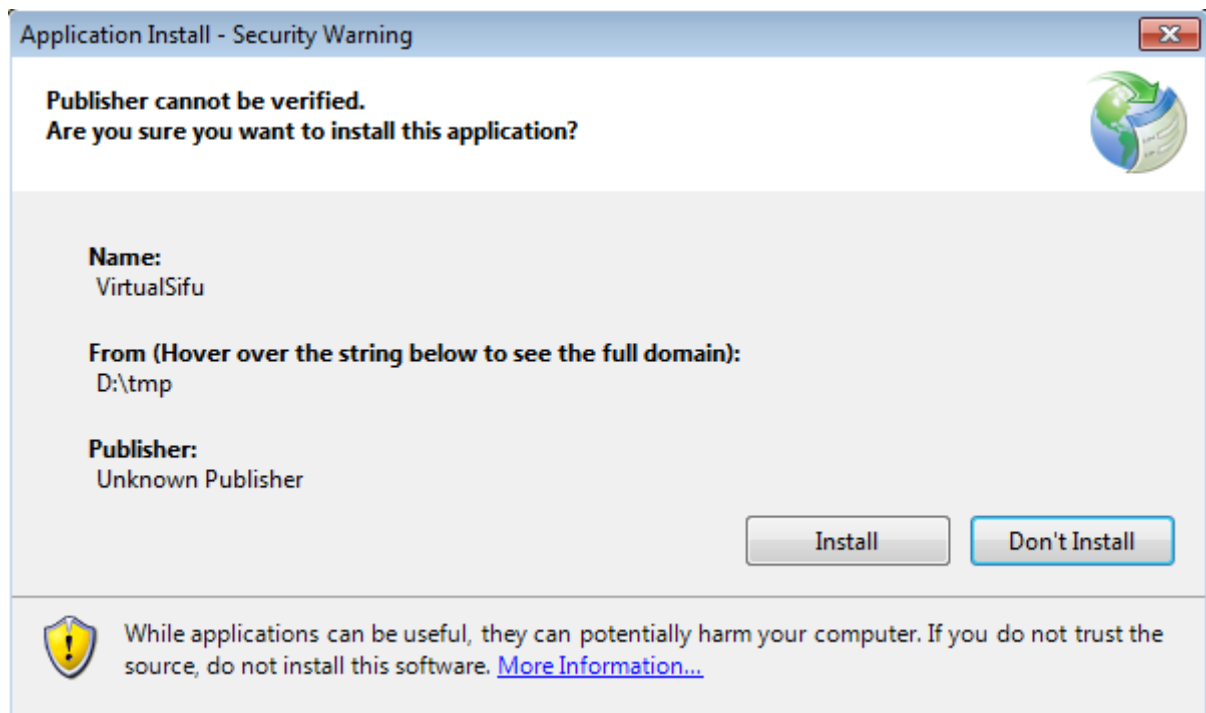


Figure 24: Security Warning

Virtual Trainer will now run.

4.6.2 Software un-installation

1. Open the Add / Remove Programs menu.

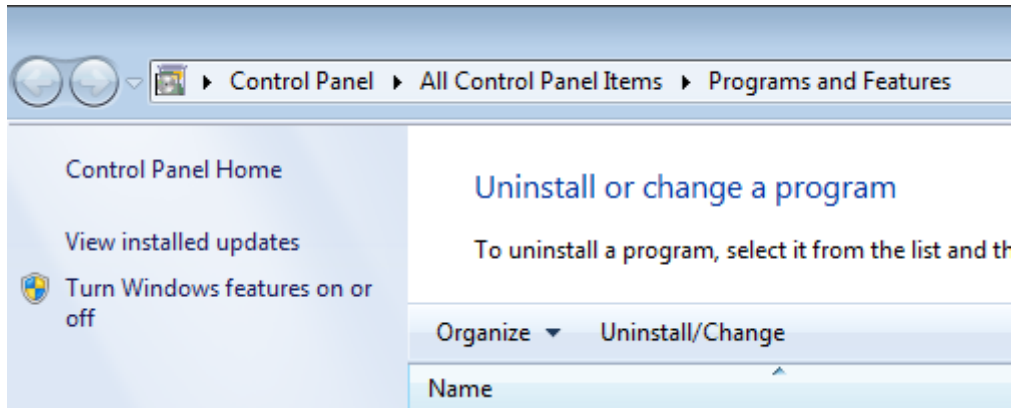


Figure 25: Add / Remove Programs

2. Select *Virtual Sifu*.

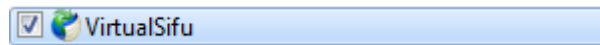


Figure 26: Virtual Trainer entry

3. Click Uninstall / Change.

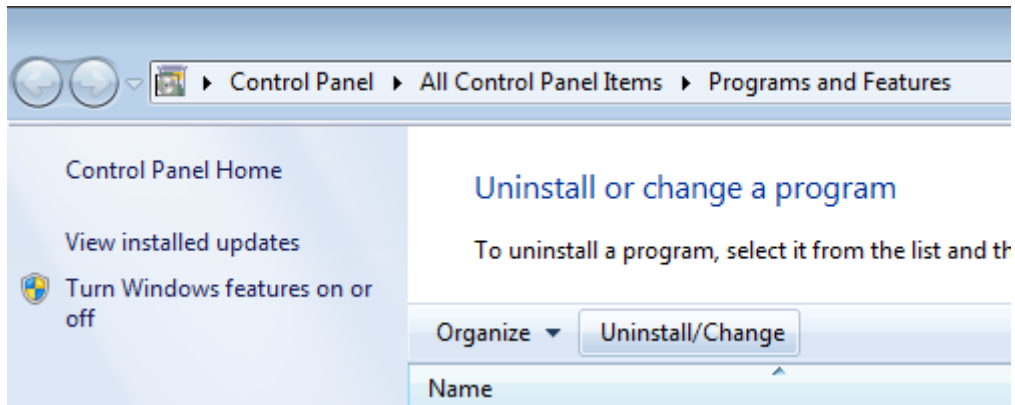


Figure 27: Uninstall / Change Button

4. Select to remove the application.

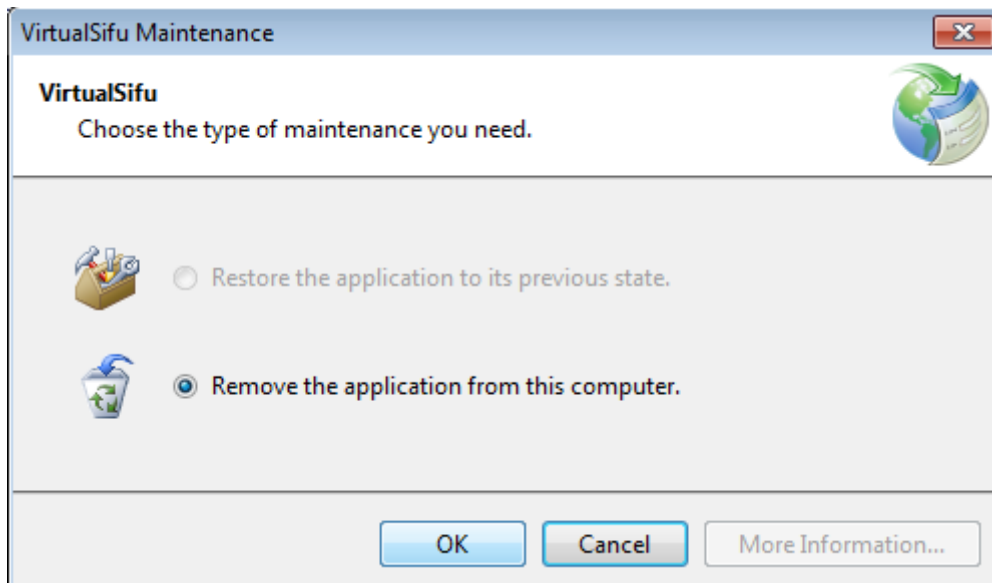


Figure 28: Uninstall Virtual Trainer

Virtual Trainer is now uninstalled.

5. CONCLUSION

In this project, I have developed a new technique of learning martial arts. I have explored the number of ways in which we can use Kinect by integrating it with various other tools. By implementing Dynamic Time Warping algorithm two data sequences irrespective of time and speed are calculated for similarities. The captured moves of the Sifu are in 2D, but in order to give user a better feel and make it livelier the moves were displayed in several different angles. So the user can select any particular angle in order to learn better. With the help of Blender I was able to create the animations using the data obtained from MMDR ^[7]. Blender is capable of creating multi directional videos and can export videos in all desired formats. By using Visual studios we are able use the Kinect SDK 1.0 released by Microsoft with C# as backend. The advantage of using C# is that we can integrate XNA 4.0 to develop games using Kinect. Based on the results, this virtual training system gives an approximate feedback of the gestures made. But the main disadvantage of this system is the prediction of the physical force applied by the master and the user. This project still needs an improvement on accuracy of the motion analysis algorithm in order to provide better feedback to the users. To make the learning experience more realistic and fun, the execution of a punch or kick has to be made appropriate and simulate opponent who is attacking them and then lot of improvements could be made in the visuals, like people applauding for the winner or a judge showing the signs.

6. REFERENCES

- [1] Blender(N.D.). Retrieved May 15 2012 from <http://www.blender.org/>
- [2] BLEND SWAP(N.D.). Retrieved May 15 2012 from <http://www.blendswap.com/>
- [3] Dynamic time Warpping. *In wikipedia*. Retrieved May 15 2012 from http://en.wikipedia.org/wiki/Dynamic_time_warping
- [4] Rath, T.M., & Manmatha,R.(20 june 2003).Word image matching using dynamic time warping.2003 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)*. vol.2, pp. (II-521- II-527). doi: 10.1109/CVPR.2003.1211511
- [5] Kinect.*In wikipedia*.Retrieved May 15 2012 from <http://en.wikipedia.org/wiki/Kinect>
- [6] Meredith, M. & Maddock, S.(N.D.). Motion Capture File Formats Explained. Retrieved from <http://www.dcs.shef.ac.uk/intranet/research/resmes/CS0111.pdf>
- [7] Vocaloid Promotion Video Project(N.D.). Retrieved May 15 2012 from http://www.geocities.jp/higuchuu4/index_e.htm
- [8] KINECT for Windows(N.D.). Retrieved May 15 2012 from <http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx>
- [9] NI-Mate. Retrieved May 15, 2012 from <http://www.ni-mate.com/>
- [10] Dynamic time warping (DTW) A comparative study of several dynamic time-warping algorithms for connected word recognition.
C. S. Myers and L. R. Rabiner. The Bell System Technical Journal, 60(7):1389-1409, September 1981.

[11] Ray Chambers Kinect Tutorials

<http://raychambers.wordpress.com/tag/ray-chambers-kinect/>

[12] Steven Macramalla's YouTube Channel

<http://www.youtube.com/user/smacram>

[13] Steven Macramalla's Website

<http://www.sixanimalskungfu.com/>