One-Shot Learning of Gestures using Kinect data

Author
Shashank Sonkar
IIT Kanpur
shashank@cse.iitk.ac.in

Advisor
Prof. H. C. Karnick
IIT Kanpur
hk@cse.iitk.ac.in

Abstract

In this report, we address the problem of recognizing simple and repetitive gestures. With the help of kinect, and the Microsoft SDK, we can now track the human skeleton in real time. Given the information of coordinates of body joints, we aim to distinguish different gestures.

The approach we take is to extract the trajectories made by the hand joints while performing the gesture and then use Fréchet distance[1] to measure the similarity between the trajectories. Fréchet distance has similar applications in hand-writing recognition and comparision of two line strings[2].

Our final aim is to come up with a method that can distinguish between gestures using only a single instance of the gesture. We have been given a dataset of 12 gestures, which have around 6000 gesture instances.

Previous work involved representing a gesture by a unique signature. The signature was represented as a string and compared using the Needleman-Wunsch [3] algorithm. However, accuracy did not exceed 75-80%. The accuracy using Fréchet distance is far better.

The method based on Fréchet distance is later integrated into the skeletal tracking library of Microsoft Kinect SDK and is used to control the VLC music player. It has six gestures to control the player - volume up, volume down, next song, previous song, pause and mute. The method also has applications in sign language recognition where repetitive gestures are used.

1 Introduction

Humans are capable of recognizing patterns like hand gestures after seeing just one example. We want computers to be able to do the same. Take for instance, a hand gesture. Humans basically know the trajectory through which the hand has moved. Can machines have a way to compare two trajectories?

The first step for a machine would be to identify the person in the frame and figure out where his/her hands are. The next step would be to calculate the space coordinates of the hands with reference to an origin. We use Microsoft Kinect and its built-in skeletal tracking library. It can identify upto to 6 people in the frame and also track their body joints. It collects frames at the rate of 30 frames/second. Space coordinates for a total of 20 body joints can be retrieved with the sensor assumed to be at the origin.

The next step is to compare two trajectories in space. Researchers faced a similar problem when they had to to find closeness between hand-writings of two people. One way was to use Fréchet distance.

Now that we have the gesture represented as a curve, this becomes similar to hand-writing recognition. So, we explore the use Fréchet distance in recognizing gestures.

1.1 Dataset

Microsoft Research Cambridge-12 Kinect gesture data set[1] has been used. There are a total of 12 different gestures. The gestures are of two kinds, iconic and metamorphic. Iconic gestures are those that have a correspondence between the gesture and the reference. Metaphoric gestures are those that represent abstract content. For the former, there are six gestures from a first person shooter game and for the latter, there are six gestures for a music player. The dataset comprises of 6244 gesture instances, collected from 30 people performing 12 gestures.

For each gesture instance, the space coordinates (x,y,z) of 20 body joints has been given for each frame. Frames are collected at the rate of 30 per second. The joints include head, shoulder center, shoulder left, shoulder right, elbow left, elbow right, wrist left, wrist right, hand left, hand right, hip center, hip left, hip right,

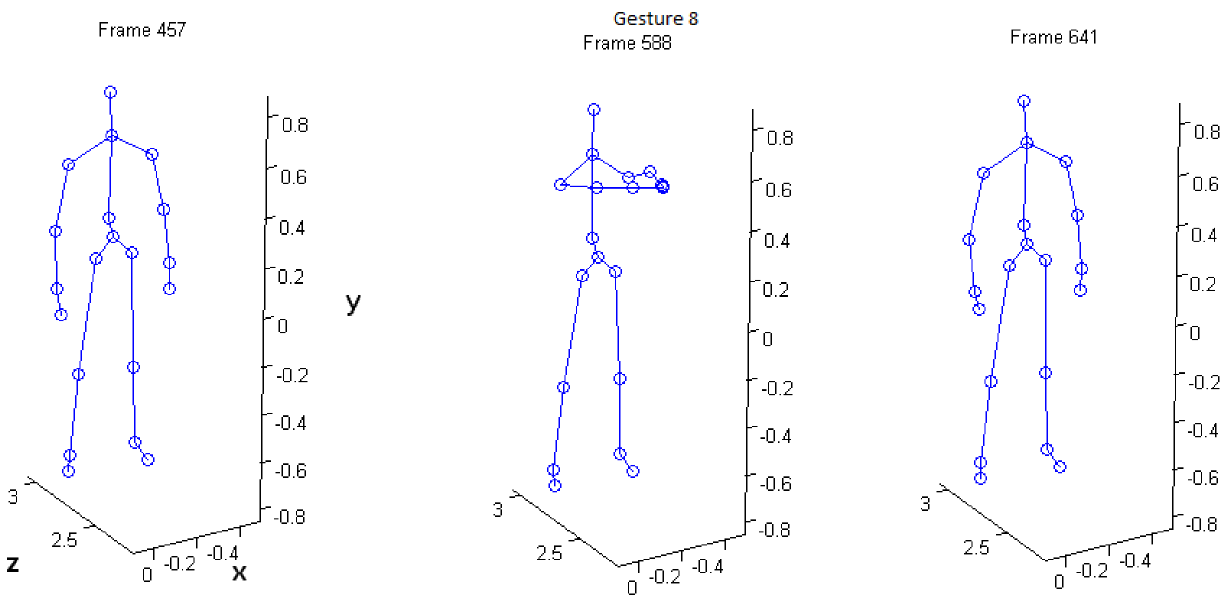

Figure 1: Visualisation of gesture from a dataset

spine, knee left, knee right, ankle left, ankle right, foot left and foot right. The origin is the Kinect sensor. ‘Z’ coordinate represents the depth of the person depicting the action. ’Y’ coordinate represents the axis along the height of a person.

For one instance of the "shoot" gesture (gesture 8), the start frame, the mid frame where the person brings both hand together as if holding a gun, and the last frame have been visualised in figure 1.

1.2 Kinect Skeletal Tracking using MS SDK

We first tested whether the library could track the human skeleton in real time. When we stood infront of the kinect, it tracked the skeleton after a delay of 2-3 seconds and continued tracking it until we moved out of the range of the sensor. There was very little time lag (1-2 seconds) between body joints moving and kinect tracking that movement. Moreover, there is an API for accessing the space coordinates of 20 body joints.

1.3 Previous Work

Previously, for each gesture, we aimed to build a MTechunique signature that could demarcate it from other gestures. The approach that we took was to discretise the space in cubes. A signature was the ordered sequence of cubes through which the joint moved. Then, we compared two gestures using a string matching algorithm(Needleman-Wunsch algorithm).

The gesture completion time varied across different instances of the same gesture as the person could perform a gesture with varying speeds. Also, different people performed the same gesture with different speeds. So, the gesture signature had to be time invariant.

The factor that controlled the accuracy was the extent of discretization of space. However, the accuracy did not exceed 75-80%.

1.4 New Methodology

The current technique is to use the Fréchet distance for one-shot gesture recognition.

Moreover it can also deal with more complex gestures like "waving goodbye to a friend". One person may wave his/her hand five times, whereas another may wave only thrice. In such repetitive gestures it is necessary to extract the base gesture that is repeated. We tackle this by extracting the number of times the gesture is repeated and then creating a grammar rule like representation for the gesture.

Such gestures have obvious applications in real life, for example in sign language or controlling devices or software using gestures. For example N repetitions may mean N slides forward. But one will only make a single gesture to train the machine and the machine has to generalise it to n.

2 Fréchet distance

Fréchet distance is a measure of similarity between curves that takes into account the location and ordering of the points along the curves.[5]

The formal definition is as follows[1]. Let V denote in the following an arbitrary Euclidean vector space.

Definition 1: A curve is a continous mapping $f : [a, b] \rightarrow V$ with $a, b \in R$ and $a < b$. A polygonal curve is a curve $P : [0, n] \rightarrow V$ with $n \in N$, such that for all $i \in \{0, 1, ..., n-1\}$ each $P_{[i,i+1]}$ is affine, i.e. $P(i + \lambda) = (1 - \lambda)P(i) + \lambda P(i + 1)$ for all $\lambda \in [0, 1]$. n is called the length of P.

Definition 2: Let $f : [a, a'] \rightarrow V$ and $g : [b, b'] \rightarrow V$ be curves. Then $\delta_F(f, g)$ denotes their fréchet distance, defined as

$$\delta_F(f, g) = \inf_{\alpha[0,1]\rightarrow[a,a'],\beta[0,1]\rightarrow[b,b']} \max_{t\in[0,1]} || f(\alpha(t)), g(\beta(t) ||$$

where α, β range over continuous and increasing functions with $\alpha(0) = a, \alpha(1) = a', \beta(0) = b$ and $\beta(1) = b$.

An intuitive interpretation is: imagine a person and his dog have been restrained to walk on two different continuous curves. They both have to reach from one endpoint to another of their corresponding curves without backtracking. Then, the Fréchet distance is the shortest leash that is required by the owner to walk with his dog.

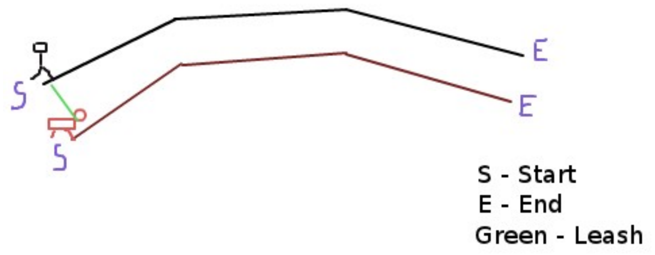

Figure 2: Illustration of Fréchet distance

The owner and his dog can change their velocities as they walk along their respective curves, however they cannot backtrack. For each such pair of velocity and acceleration profiles of the owner and his dog, there is a minimum length leash that is required by the owner such that the dog does not run away.

Fréchet distance is the shortest of all leashes that would suffice for the walk. Without this length of the leash the walk is not possible.

The walk that requires only Fréchet-distance length of leash can be understood with an interesting example. Suppose that the leash has a positive cost per unit length and both the owner and the dog want to cut down on the cost of leash required. So, they collaborate and adjust their velocities such that they remain as close to each other as possible while they walk on their respective curves, thus cutting down the cost of the minimum leash required. This walk can be interpreted to require only Fréchet-distance length of leash. Note that such a walk in not unique. Many such walks exist.

2.1 Algorithm

For our purpose, we only need the discrete Fréchet distance as the curves are not continuous. The curve/trajectory in our case is an ordered sequence of coordinates of the body joint as it went through space.

In mathematics, such a curve is called a polygonal chain. A polygonal chain is denoted by a sequence of points known as vertices. Line segments connect two consecutive vertices[6].

We can compute Fréchet distance between two polygonal chains using a simple dynamic programming algorithm. Consider two polygonal chains having m and n vertices respectively. Let the vertices of first chain be denoted by $\{a_1, a_2, .., a_m\}$ and vertices of second by $\{b_1, b_2, .., b_m\}$. Let *dist* be a function that given two vertices returns the Euclidean distance between them. F_d(i,j) stores the Fréchet distance between the subchains denoted by $\{a_1, a_2, .., a_i\}$ and $\{b_1, b_2, .., b_j\}$. The algorithm is given below.

```
$F_d(1, 1) \leftarrow dist(a_1, b_1)$
**for** $i = 2$ to m **do**
  $F_d(i, 1) \leftarrow max\{\ F_d(i-1, 1), dist(a_i, b_1)\}$
**end for**
**for** $j = 2$ to n **do**
  $F_d(1, j) \leftarrow max\{\ F_d(1, j-1), dist(a_1, b_j)\}$
**end for**
**for** $i = 2$ to m **do**
  **for** $j = 2$ to n **do**
    $c \leftarrow min\{\ F_d(i-1, j), F_d(i, j-1), F_d(i-1, j-1)\}$
    $F_d(i, j) \leftarrow max\{c, dist(a_i, b_j)\}$
  **end for**
**end for**
```

2.2 Time Invariance

Gesture completion time varies across different instances of the same gesture. However, computation of Fréchet distance only takes into account the space through which the hand joints have travelled. Therefore, whether one does the gesture quickly or slowly has no effect. Vertices of the polygonal chain will be more separated when gesture is done fast as the rate at which frames are collected remains the same. This does not influence the result because the frame collection rate is sufficiently high.

3 Grammar of gestures

Some gestures symbolise a single action but they have one or more repetitions of a base gesture or curve. So, different instances of a gesture will vary but the gesture class is the same. For example a gesture such as 'waving good bye' can repeat the base waving gesture multiple times.

3.1 Motivation to count repetitions

We focus on one-shot gesture recognition. A person may train the machine using a gesture in which (s)he waves twice. However, (s)he expects the machine to generalise it to any number of waves. In addition, we wish to extract the number of times the base gesture is repeated. The exact number can sometimes be meaningful - for example it may code for some kind of intensity or number.

3.2 Logic

Figure (3) shows the curve plotted by the right hand joint when a player punches 3 times. As one can observe, the gesture can be broken down into three parts.

1. Onset: The ascend of the hand till it reaches to the point from where the punch starts.

2. Base curve: The subcurve that is getting repeated. In figure(3), the subcurve joining one green dot to the next red dot and finally to the next green dot is the base. By definition and also one can see in figure (3), in case of three punches, base is repeated three times.

3. Offset: The descend of the hand to the point from where it started.

Thus the gesture can be written as:
$Gesture_n \rightarrow (Onset).(Base)^n.(Offset)$
So our aim is:

- Given a training gesture of type $Gesture_n$, extract $Gesture_1$.

- Given test gesture of type $Gesture_n$, find the value of n and $Gesture_1$. Thereafter, use $Gesture_1$ for the Fréchet-distance comparision with training gestures of type $Gesture_1$.

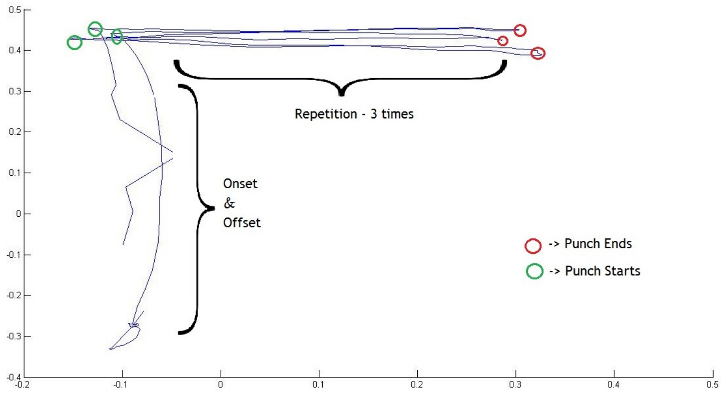

Figure 3: Punch - 3 times

3.3 Algorithm

Lets define the notion of "visitors" for a coordinate c_i in a curve $\{c_1, c_2, .., c_i, c_{i+1}, .., c_n\}$. In a plot with
$y_{c_i}(x) = distance_{euclidean}(c_x, c_i) \forall x \in \{i+1, i+2, \ldots n\}$
let $\{x_1, x_2, ..x_m\}$ be the coordinates where local minima exists. Minima at the extremes are excluded.

Then we define visitors as:
$visitors = \{z \in \{x_1, x_2, ..x_m\}$ s.t. $dist(c_i, z) < T\}$
where T is some threshold.

Figure(4) represents a curve formed by the hand joint in which the base- curve gets repeated three times. Consider the red dot lying at the end of the onset and beginning of the first base-curve (sub-curve in red). The black dots denote the visitors for this red dot.

However, for the green dot, the black dot that lies outside the black circle(threshold) is not a visitor, though in the graph, local minima will exist at that coordinate.

The point to note is that
$no.\ of\ visitors_{redpoint} = no.\ of\ repetitions.$

This happens at the red dot because one starts a new punch close to the coordinate from where the previous

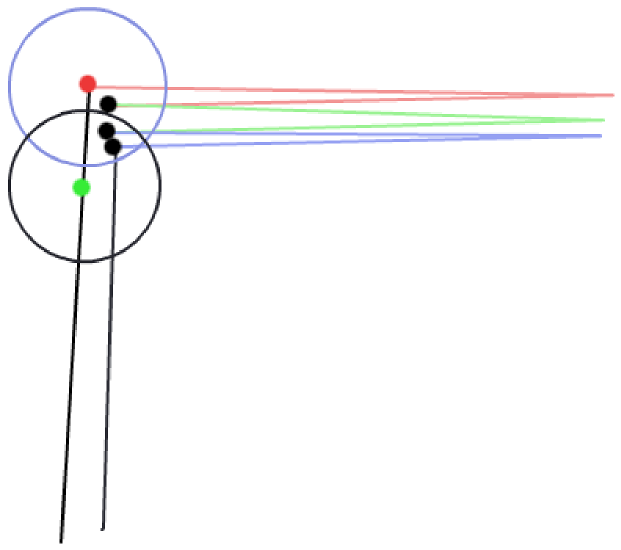

Figure 4: Local Minima

punch started. Had we not considered the threshold ($T \to \infty$), the equality would have been true even for the green dot.

But, we preferred to go for the red dot because later this will be used to extract the base-curve as well.

The next step is to locate the red point. The first point c_i in the sequence $\{c_1, c_2, .., c_n\}$ that has $visitors >= 2$ is the red point.

Sometimes confusion arises as coordinates on the onset (like the green dot in figure(4)) near the start of first base-curve also become possible candidates. So when we encounter a coordinate c_{pr} that is a possible red point, we calculate:

$M \leftarrow Mode\ of\ \{V_{pr+1}, V_{pr+2}, .., V_{pr+10}\}$
$where\ V_i = no.\ of\ visitors\ of\ c_i.$
Any coordinate $c_j \in \{c_{pr+1}, c_{pr_2}, .., c_{pr+10}\}$
that has $V_j = M$ can be choosen as the red point.
This M denotes the number of times base-curve is repeated. It is the value of n in $Gesture_n$. While calculating mode, we took V_i values of the next ten coordinates because of three main factors:

- False candidates are found near the beginning of the first base-curve (figure (5)). Choosing from the next ten coordinates makes us go closer to the red dot of figure(4) and farther from the green dot.

- Density of points increases at the start of the base curve because speed of the hand decreases. So, we do not run the risk of going too far beyond the red point.

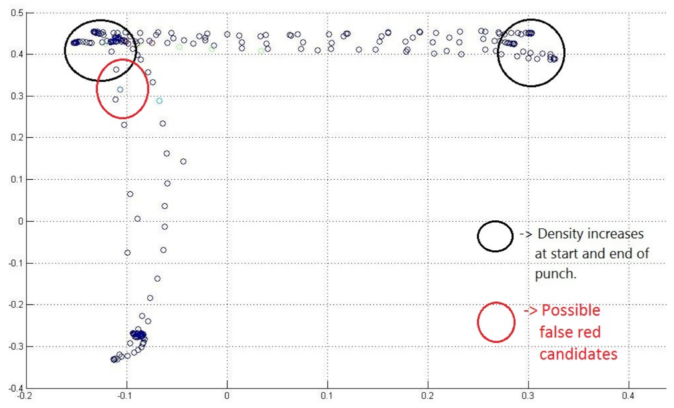

Figure 5: Mode and density

- We come closer to the red dot and thus have more coordinates with $V_i = V_{reddot}$. See figure(5). The high density of points plays a role here. Now taking mode on V_i values of such coordinates also removes the chance of having $M = V_{greendot}$.

.

3.4 Extracting $Gesture_1$

Let the red point be c_r in the curve $\{c_1, c_2, .., c_n\}$. Its visitors be $\{v_1, v_2, .., v_k\}$. Then,
$Onset \leftarrow \{c_1, c_2, .., c_r\}$
$Base \leftarrow \{c_{r+1}, c_{r+2}, .., v_1\}$
$Offset \leftarrow \{v_k, v_k + 1, .., c_n\}$

4 Experiments

4.1 Experiment 1

For a person let all gesture instances of the i^{th} gesture belong to the set G_i where $i \in 1..12$.

We take a test gesture $test_i \in G_i$. Then randomly select one instance from all G_k, $k \in 1..12$. Call them $train = \{g_1, g_2, ..g_n\}$ such that $test_i \neq g_i$.

We compute
$F_r = Fréchet_{righthand}(test_i, g_k)\ \forall g_k \in train$
where $Fréchet_{righthand}(g_i, g_j)$ represents the Fréchet distance between the trajectories of right hand joint of gestures g_i and g_j. Similarly,
$F_l = Fréchet_{lefthand}(test_i, g_k)\ \forall g_k \in train.$
If the quantity $(F_r + F_l)$ for $test_i$ the test gesture is minimum at $g_i \in G_k$, then the gesture has been correctly recognised.

Gest. → Pid ↓	1	2	3	4	5	6	7	8	9	10	11	12
5	100,9	94.4,18	100,9	100,18	100,9	77.8,18	100,9	94.4,18	100,9	100,18	61.1,18	100,18
6	100,17	77.8,18	100,17	88.9,9	100,10	89.5,19	88.9,18	100,18	72.2,18	100,9	100,9	82.4,17
7	100,19	83.3,18	100,18	100,9	64.2,53	100,9	100,18	100,18	95,20	100,9	75,20	90,10
11	55.6,18	100,9	100,10	100,18	100,19	100,19	70.6,17	100,9	100,9	100,18	100,18	100,18
12	100,19	100,9	100,20	100,9	97.3,37	45.5,11	100,18	94.4,18	100,18	100,9	47.4,19	100,9
14	100,18	94.7,19	100,19	94.4,18	100,9	100,9	88.9,18	100,18	88.9,18	70,20	81.8,11	88.9,9
16	100,9	100,9	100,18	100,18	100,18	68.4,19	100,9	100,9	100,18	100,10	90.5,21	75,16
22	100,18	100,19	100,20	100,9	100,7	94.4,18	89.5,19	100,16	72.2,18	100,10	87.5,8	95,20
23	100,17	100,18	55.6,18	100,18	100,9	100,9	72.2,18	100,18	94.4,18	100,18	100,9	100,9
28	100,21	100,18	100,19	88.9,9	100,9	83.3,18	33.3,18	100,18	100,18	77.8,9	66.7,9	88.9,18
29	77.8,18	100,9	100,9	100,18	100,19	95.2,21	88.9,18	100,9	100,9	100,19	100,19	77.8,18
30	100,9	40,10	100,10	100,18	100,29	85,20	90,10	100,18	90,10	84.2,19	57.9,19	95,20

Table 1: Results Exp. 1

Table(1) shows the results of this experiment. The first row shows the gesture that is tested. The first column has the person id. Twelve persons were selected at random for the experiment. Each element in the table has two values separated by a comma.

First shows the % of gestures correctly classified and second is the number of gestures on which testing was done.

4.2 Experiment 2

In this experiment, we check whether our algorithm can correctly predict the value of n in $gesture_n$. Recall that, n is the number of times a base-curve gets repeated. Due to scarce examples of such gestures in the dataset, we had to make our own dataset.

4.2.1 Integration in Microsoft SDK

To test the code live, we had integrated the Fréchet-distance code into the skeletal tracking library provided in Microsoft Kinect SDK. We also implemented the algorithm to find out the number of repetitions. We used this to build our dataset and test the algorithm. We include two gestures. Gesture *hello* and gesture *swing right* as shown in figure(6) and (7) respectively. Stick figures have been made using screenshots from the Kinect API.

4.2.2 Results

We recorded a total of 27 gestures. The algorithm gave correct value of n in 24 cases. The results are shown in Table(2).

Gesture Id	Repetition	Instances	Accuracy
Hello	2	2	100.0
Hello	3	4	100.0
Hello	4	3	100.0
Hello	5	3	100.0
Hello	6	3	66.6.0
Hello	7	1	100.0
Swing	2	2	50.0
Swing	3	2	50.0
Swing	4	3	100.0
Swing	6	1	100.0
Swing	5	2	100.0
Swing	7	1	100.0

Table 2: Results $Gesture_n$

5 Controlling Music Player

The motivation behind one-shot gesture recognition was easy control of applications on the machine using gestures. The Microsoft Cambridge gesture dataset had six gestures for music player control and six for controlling a shooting game.

For testing our ideas, we had to use this dataset. However, the final aim of the exercise was to integrate the algorithm, if successful, into the skeletal tracking library of Microsoft Kinect SDK. Fréchet distance measure for gesture recogniton produced good results when tested on this dataset.

So, we integrated the algorithm into the API and used it to control the VLC music player. We used six gestures for volume up, volume down, next song, previous song,

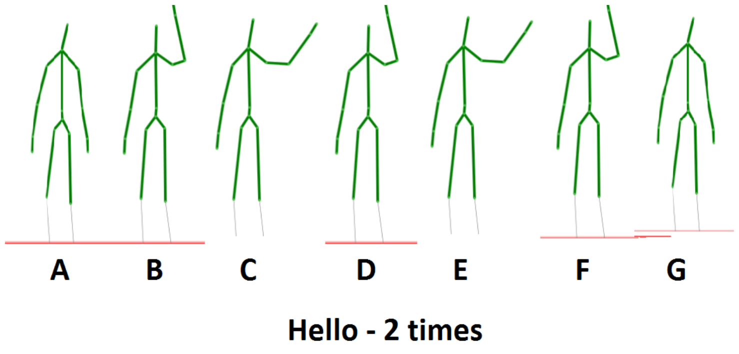

Figure 6: Hello Gesture

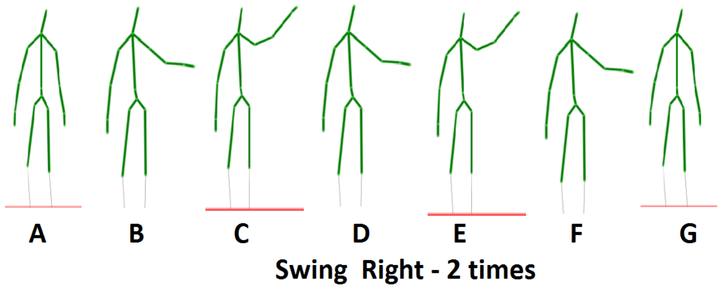

Figure 7: Swing Gesture

mute and pause.

We also have an additional feature - counting the number of repetitions. One could train the machine using a gesture which means 33% increase in volume (like swing right gesture). Repeating the gesture twice produced 66% increase in volume.

Similarily other applications can be controlled.

6 Future Work

As of now, we use Microsoft's SDK which tracks the skeleton and provides information on joint motion. However, it cannot be used to control applications on the Linux platform or any other OS. In figure (8), (i) shows the depth image and (ii) shows the skeleton/stick figure imposed on it. Depth data from Kinect can be accessed in Linux, however the stick figure and joint coordinates are calculated by MS SDK. The source code or algorithm that does this is not available.

However we can assume that the person is the first object in front of the camera and we can get the depth image from which we can extract the human figure. As can be seen in the depth image of figure (8.i), the human has a different shade of grey because of his nearness to the camera.

One can build a large dataset of such images and corresponding stick figures. Using a machine learning algorithm, given just the depth image, one should be able to predict the stick figure.

This will make the API system independent as well as the learning algorithm that creates the stick figure should be valuable in its own right.

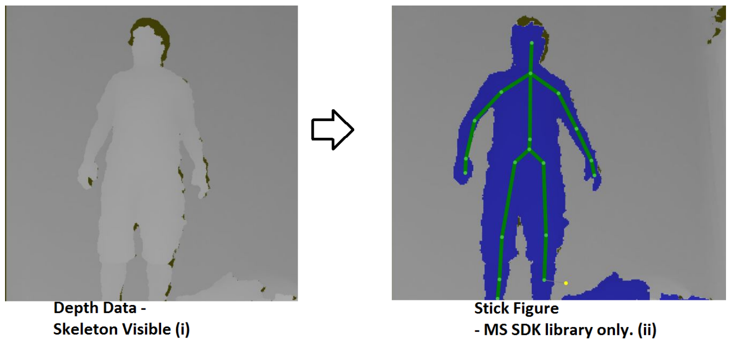
Depth Data - Skeleton Visible (i)
Stick Figure - MS SDK library only. (ii)

Figure 8: Depth Data and Imposed Skeleton

7 References

1. H. Alt and M. Godeau. Computing the Fréchet distance between two polygonal curves. International Journal of Computational Geometn & Applications, pages 75-9 I , 1995.

2. http://msdn.microsoft.com/en-us/library/bb-895372.aspx

3. S.B.Needleman and C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, J.Mol.Biol.48(1970),443-453

4. Fothergill, S., Mentis, H.M., Kohli, P., Nowozin, S. (2012). Instructing people for training gestural interactive systems. In Proceedings of the ACM conference on human factors in computing systems (pp. 1737–1746).

5. http://en.wikipedia.org/wiki/Fréchet_distance

6. http://en.wikipedia.org/wiki/Polygonal_chain