Motion Capture To MIDI

Interim Report

by
Joe Starling
0920088

Supervisor: Prof. David Marshall
Moderator: Dr. Lai

1. Introduction

The aim of this project is to successfully interface with a computer using only physical gestures. Once a gesture is learned and recognised it will be associated with a specific MIDI sound or instrument which will be output from the speakers. In this report I will discuss the theory behind implementing such a recognition system, and the progress thus far.

The concept of having virtual synthesized instruments has been around for some time, but a hands free method of creating sound or sound-effects simply with body movement is relatively new and I believe it could be accepted in the mainstream in the future. Countless new instruments and sounds could be added, creating many possibilities for musicians looking for adaptability and once the concept is fully working more complex gestures and associated sounds could be implemented, which would be an interesting prospect for many performing musical artists.

Traditional synthesisers are extremely popular with modern performers, yet they still consist of a person pressing buttons and playing keys that are invisible to the crowd. Virtual synthesisers like the one I will be creating would make the entire concept more interactive, giving the performer more freedom of movement and making performances more aesthetically pleasing. Playing a virtual instrument has the potential to be just as intuitive as a traditional instrument, with minor intricacies and instrument-specific movements being interpreted correctly such as a guitar fret slide or hammer-on, but with the added benefit of mobility, the possibility of recording without additional hardware, and possible switching to other instrument sounds instantly. Of course, the user must also be within certain constraints – movement, positioning, or otherwise, dependant on the methods employed.

2. Background

The traditional view of Motion Capture technology, especially when used in film making and animation, is that of an actor wearing an all in one costume dotted with several optical markers. Usually several cameras are used to calibrate the position and movement of these reflective markers, but recently even these methods are starting to look dated due to the rapid advancement of motion capture technology. Many suits worn today do not require the use of cameras, allowing for more mobility and ease of use. A great deal of the major advancements have been due to the film industry demanding consistent improvements in the technology available, but there are other applications where motion capture can still make major advancements, and music is one of them.

Today there is a demand for new technology to be integrated with every day life more than ever before, and motion capture is the obvious answer. Input methods such as the keyboard and mouse are being challenged, and it is likely that in the near future, visual gesture recognition will become commonplace. For example, the Xbox Kinect [1] uses the entire body in place of a traditional controller operated by hand. The Kinect sees in 3D and allows for remarkably accurate control of game-play, with even small movements and subtle flicks being translated to interactive, on-screen actions. The device broke the world record for the fastest selling electronic device in history upon release [2], proving that there is a great demand for such interactivity.

Possibly the most common form of pattern recognition that appears in everyday life is the touch pad, or touch screen phone. The technology has passed the need for a stylus and users can simply use their hands and fingertips, as well as use more than one finger at a time in order to give greater control to manipulate the screen, such as zooming in and out, called multi-touch. The screens use a method of measuring the changes in capacitance as the user makes contact with the screen, utilising the fact that humans are electrical conductors, as well as other similar methods. In order for a device to recognise handwriting and transform it into digital text for example, it must have already learned the appropriate gestures (written letters/words) of a written language, and subsequently recognise when one is used with a high enough error threshold so as to allow for each individual's style of writing. This form of writing has not become popular as the technology is still not accurate enough, however the concept of pattern/gesture recognition is an

important one, especially for this project.

2.1 Applications

In order to give some context I will discuss other products and projects that are related to my own. Most are music-related, but the same concept can be used in totally different scenarios, leaving many possibilities, so I believe reviewing projects with different goals could be beneficial. By evaluating other applications it is possible to determine whether my own method is reasonable and achievable, and also the success of the other projects can be a good indicator for my own. For example, if a project has failed, it would be easy to assess why, and then be able to avoid the same mistakes.

Gesture Therapy

First I will look at a non-musical application, Gesture Therapy [3]. The product uses some similar principles to those I hope to implement, but the end goal is something totally different, so it is interesting to see how people are using the same technology for different purposes. Gesture Therapy is a form of distance learning where debilitated stroke victims perform movements to aid with their recovery. The movements are very specific to the physiotherapy aspect of stroke recovery, so it is important the victims can perform these movements correctly, but without the need for a carer or other health professional to be present at all times. Using the same basic principles as many musical applications, it recognises performed gestures. Using two inexpensive cameras (fair quality webcams), the software recognises the hand using pre-learned skin colour data. This data was 'learned' through training a Bayesian Classifier with thousands of different coloured pixels that could possibly represent skin colour. Given the set of data (the likely pixel colours of a hand) the Bayesian classifier calculates the probability that a collection of pixels seen by the cameras is indeed a hand.

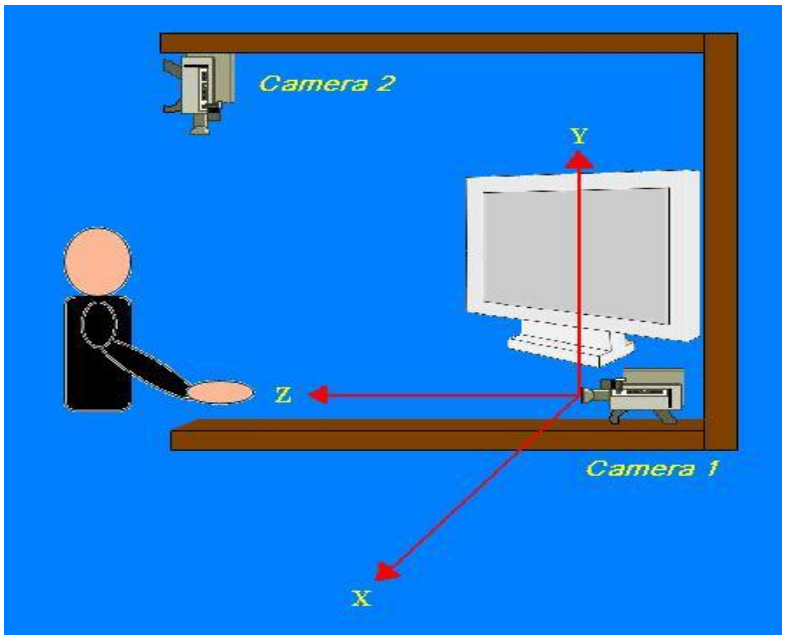

Figure 1. Gesture Therapy basic setup

If the probability is above a certain threshold, then it assumes a hand has successfully been recognised and the program can continue to the next stage. A similar method will be employed during my own project to recognise the gestures themselves, where successful recognition equates to a sound being played (more on this later). It is recommended that the user wears long sleeves and performs in front of a plain, uniform background to avoid confusion and ensure the hands are recognised when using Gesture Therapy. Once it has recognised the hand, it estimates the 3D coordinates and uses them to interact with the software. Basic real-world situations and movements are mimicked on screen and the program judges whether the correct gesture is being performed. This product was developed in South America and used at a hospital in Mexico City and was deemed a success, however it is yet to see much commercial deployment. Other organisations are still exploring the concept [4], meaning it is still currently a credible technique to use for rehabilitation, and will likely progress in the future.

Ability Music (Virtual harp)

Now a more relevant application created for Ability Music, an American charity that helps children and young people with disabilities to create music [5]. They recently worked with Alex Shubert, a student at Algonquin College, NY, to implement a virtual harp for people with little or no mobility in their fingers; this research project in particular is very interesting.

The harp has no strings, just empty space, meaning a simple wave of the arm can create music. It doesn't recognise gestures as such, but each area corresponds to a musical note, so the idea is very similar to this project. It is encouraging to see an example of this technology being used charitably and not solely for the purpose of entertainment.

Gestrument

A recently released application for the iPad called Gestrument [6] creates MIDI sounds based on the movement across the touch screen, allowing the control of up to 8 instruments at a time. This is a similar idea to the project but the physical movements are minimal in comparison, with fingers rather than full body or limb gestures, yet each small movement can create various interesting sounds simultaneously. The creator of Gestrument, Jesper Nordin, has also utilised the previously mentioned Xbox Kinect and created something very close to what I am trying to achieve [7]. The latest device tracks the movement of the hands in 3D, and plays different and complex sounds based on the position and speed of movement. There are plans in place to take this project further in the future, Nordin even mentions the use of dancers, a promising sign that people are finally realising the potential of gesture recognition in the field of music.

Figure 2. Live iPad Gestrument demonstration

Figure 2 shows Gestrument in action, with multiple areas of the screen being utilised

simultaneously, giving even greater control of the output. Different areas create different notes or even simulate different instruments altogether. This product is sold as a composition tool for musicians looking to compose music interactively and on the move.

Virtual Air Guitar

Similar to the Gestrument Kinect venture mentioned earlier is the Virtual Air Guitar [8], a company founded by a group of researchers in Finland. It uses a basic webcam rather than the sophisticated Kinect, and specifically attempts to mimic the sound and playing style of a guitar. Like Gesture Therapy, the Virtual Air Guitar must recognise the hands of the user, but to do this the user must be wearing bright orange gloves – an alternative method to training thousands of shades of skin colour. This method is simple, inexpensive, and effective. Each frame of the video input stream is examined, and the centre of mass of an orange 'blob' is treated as a hand. Based on the relative positioning and the speed of movement of these two 'blobs', corresponding guitar sounds are produced.

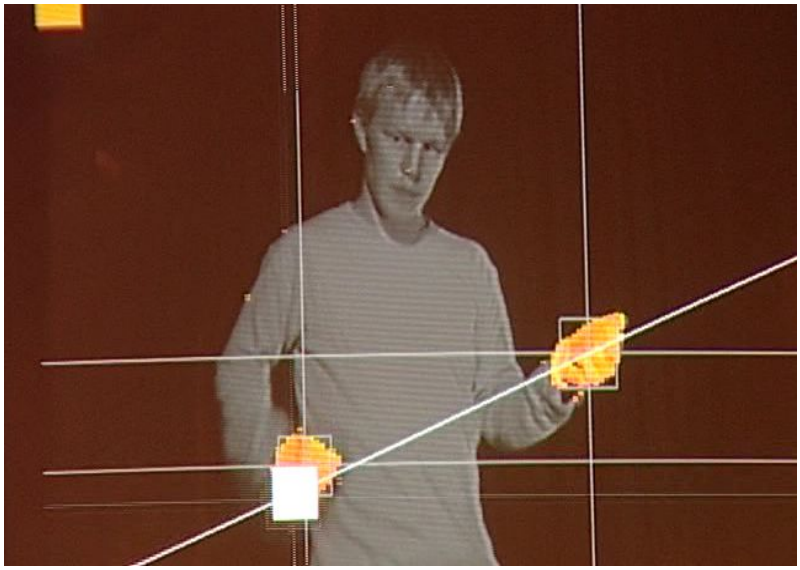

Figure 3. Demonstrating the Virtual Air Guitar

There is a virtual centre-line which represents the guitar, and as the user's right hand passes through it above a certain velocity (to avoid accidental sounds) a guitar sound is activated. Interestingly the creators of Virtual Air Guitar explicitly state they are not intending to replace a real guitar, but are instead simply bringing the art of playing the 'air guitar' to life, as an entertainment device. They do however acknowledge that the theory could be applied to more serious applications, such as genuine instrument imitation. A

great deal of inspiration can be gained from this product; it achieves a more specific goal, and does so in a different manner, but it is proof that a project such as mine is achievable. Also, encouragingly, they claim 'exceptional popularity' and 'media success'.

Mudit

Another application which is in the same category as what I am trying to achieve is Mudit [9], a brand new hand held music controller. It is a low cost, open source project which consists of two hand held devices that allow the user to control sound loops. The devices also light up, and it is being described as a live performance instrument, meaning it is intended to create credible music as well as to visually impress a live audience. The development is still in its early stages, but the makers have been very keen to ensure the product is open-source and free to use, and actively encourage positive contributions from others around the world. The hope is that by sharing the knowledge, more exciting advancements will be made within the field, and there is a real excitement currently about what will be next. This product also looks as though it is capable of applying filters to a live stream of music, changing the output drastically without applying any actual sounds, something I am interested in experimenting with, if possible.

Mouthesizer

The Mouthesizer uses a head-mounted camera pointed directly toward the mouth of the wearer, meaning the wearer is free to walk around to an extent. Using a basic image processing technique called Thresholding on the mouth, it tracks the shadowed area inside and as a result, shape parameters can be extracted and they are mapped to MIDI control changes [10]. For example, the height of the mouth and the width of the mouth can be assigned certain sound effects, which means mouthing particular sounds can have a direct effect on the output of the music/instrument being played. Facial gestures account for a large portion of the way in which we communicate with one another, from grimaces to raised eyebrows, and the facial muscles are capable of very intricate and specific movements (some individuals more so than others), which makes facial recognition an excellent tool in a wide range of applications.

There are many ways currently that a system can implement motion capture, as we have seen, but for pattern recognition there is one outstanding mathematical model used almost universally; the hidden Markov model (HMM). In order to achieve the gesture recognition required in this project, this is what I will be using.

2.2 Hidden Markov Models

The Hidden Markov Model has been used for many years, especially in the field of speech recognition, but its applications are wide and varied and it remains a popular choice for many forms of recognition, even today. A defining paper and reference point for many is a tutorial written by Rabiner in 1989 [11].

A *regular* Markov chain or process defines a set of states where the probability of moving from one state to the next depends solely on the current state (i.e. it is memoryless), where the states are directly visible to the observer. The states for this project could be considered the position of the hand.

For example, consider the three states (number of states, $N = 3$) in a simple model to be *Left, Centre*, and *Right* for a basic left-to-right swipe of the arm. The observation sequence for a successful gesture can be defined formally as $O = \{S1, S2, S3\}$, where $S1 = Left$, $S2 = Centre$, and $S3 = Right$. A possible matrix of state transition probabilities, $A = a\{i,j\} =$

	Left	Centre	Right
Left	0.4	0.3	0.3
Centre	0.2	0.6	0.2
Right	0.1	0.1	0.8

If the first state at time $t = 1$ (where state at time t is denoted q_t) is *Left*, the probability of the gesture occurring (i.e. $S2 = Centre$ and $S3 = Right$) given the model can be calculated as:

$$P(O|Model) = P(S1, S2, S3 \mid Model)$$
$$= P(S1) \cdot P(S2|S1) \cdot P(S3|S2)$$
$$= \pi_1 \cdot a_{12} \cdot a_{23}$$

$= 1 \,.\, (0.3)(0.2)$

$= 0.06$

Where the initial state probability $\pi i = P(q1 = Si),\ 1 \leq i \geq N$

From this model we can also ask other interesting questions, such as: given the model is in a known state, what is the probability of it staying in that state for exactly *d* days?

This crudely demonstrates the basic idea behind the Markov model, however the basic model alone is not adequate to be of practical use. We must extend the model to include the case where the observation is a probabilistic function of the state, i.e. the resulting model (a HMM) is a doubly embedded stochastic process with an underlying stochastic process that is not observable, and can only be observed through another set of stochastic processes that produce the sequence of events [11], and as such, it is an ideal candidate for coping with the stochastic properties of gesture recognition. A HMM is a collection of finite states connected by transitions, much like the Bayesian classifier employed in the Gesture Therapy app [3], where each state is characterised by two sets of probabilities: the transition probability and a discrete output probability distribution [12]. In a particular state an outcome or observation can be generated according to the associated probability distribution. Put simply, only the outcome is visible to the observer, and not the states, which are hidden.

The HMMs are used to represent the actual gestures, and the parameters are learned from the set of training data, an important feature. The trained models represent the most likely way that a human gesture will be performed, and are used to classify new incoming gestures [13]. By training the models with repeated gestures and iterations of the Baum-Welch algorithm, they are adjusted each time to maximise accuracy. The algorithm computes maximum likelihood estimates for the parameters of a HMM (i.e. the transition and emission probabilities) from the provided training data. HMMs scale well, meaning that adding additional gestures will not effect the already learned HMMs, however they require a large amount of data to train to an acceptable level [14]. The fact that many models can be used alongside one another is beneficial to the project, as it means I will not be limited to a maximum number of gestures.

In order to get the best results, the number of states to include in a HMM is an

important factor to consider. Naturally, a higher number of states equates to greater accuracy of recognition, yet a higher number of states greatly reduces the rate at which gestures are evaluated [13], which is not ideal for an application such as the real-time gesture recognition required for this project.

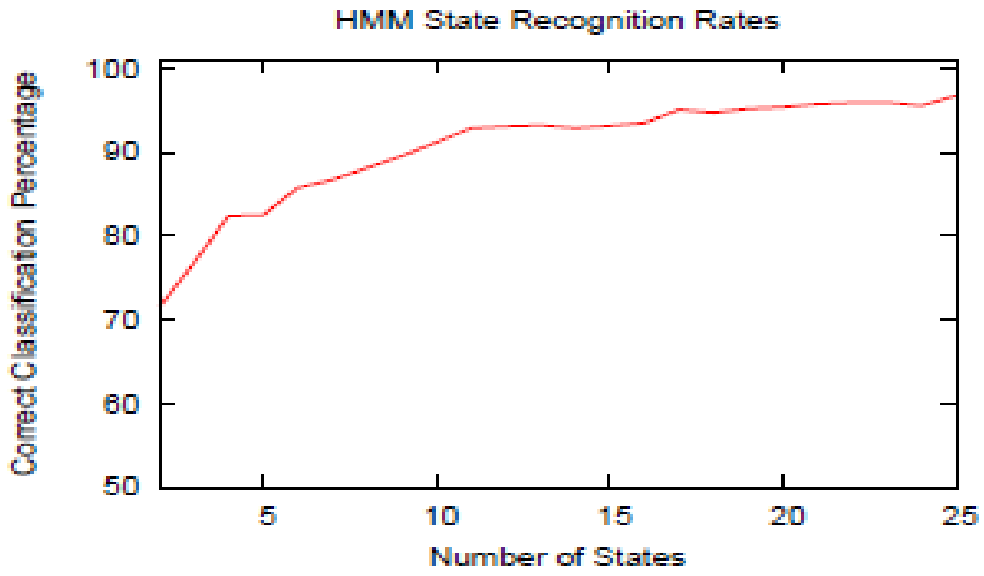

Figure 4. Chart displaying the positive effect the number of states has on recognition accuracy [13]

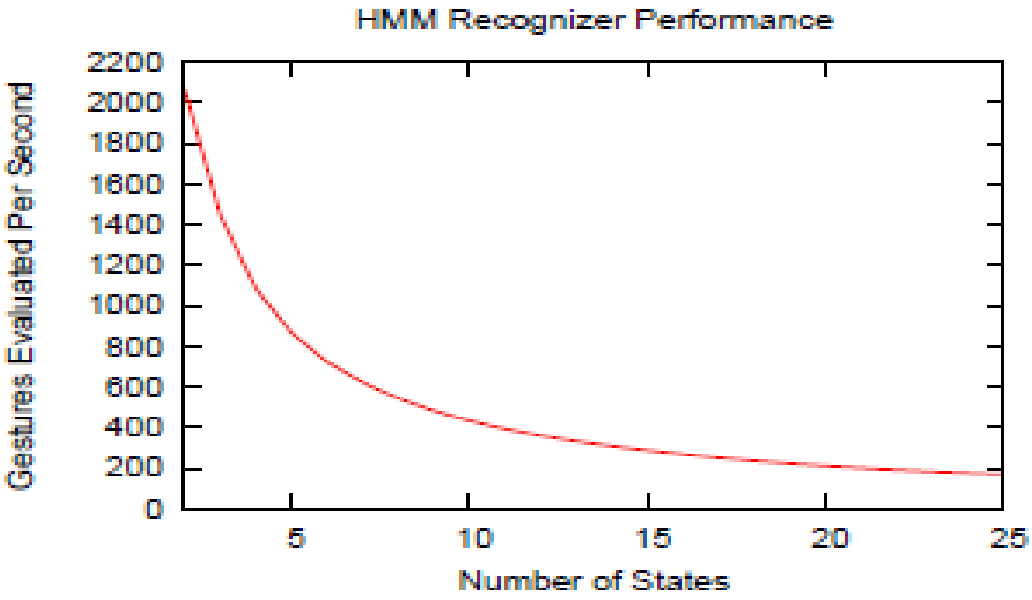

Figure 5 Chart displaying the effect the number of states has on Gesture Evaluations Per Second [13]

The sensors used in this project can capture data at a relatively high rate, which is useful, but consequentially will provide too much data to be used for a HMM. To overcome this problem, I will be reducing the number of states using the k-means algorithm to form

clusters.

2.3 Clustering

K-means clustering is a very useful method of collating dozens of elements, or coordinate readings, into *k* number of clusters, a more manageable number. Given a set of observations of length *n*, the algorithm aims to partition them into k clusters *(*where $k \leq n$*)*. For example, if three 'mean' points are chosen at random from a set of elements, the nearest elements to each mean are assigned to that cluster. Then, within these clusters, the centroid becomes the new 'mean', and the cluster is optimized. This process is repeated until the ideal cluster sizes are reached [15]. This process is illustrated below.

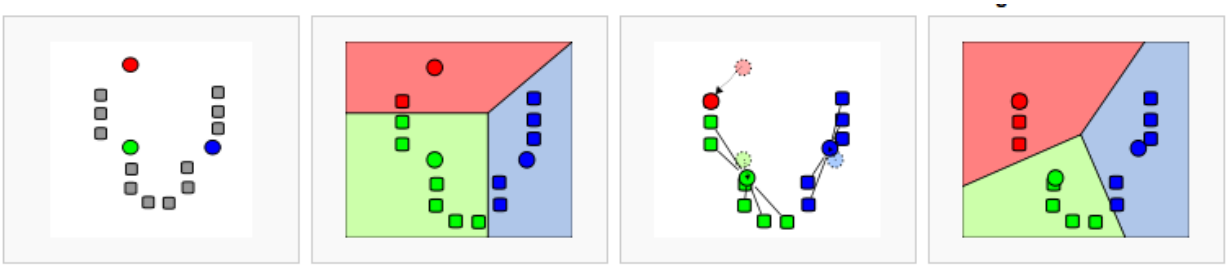

Figure 6. Diagram demonstrating the assignment of 3 clusters to a set of data

Clustering is vital to the recognition of gestures in this project, they generate a larger area for the user to perform in, meaning the gesture doesn't need to be performed 100% accurately, and they make creating usable, practical HMMs a reality. Once the points are clustered, the Baum-Welch algorithm can be applied to generate the best HMMs.

Before beginning the project, it was necessary to choose an appropriate programming language. As I am already familiar with the Java programming language I have decided that it is the language I will use to implement the project. However, due to the complexity of implementing HMMs using Java, it was necessary to search for an existing package. This lead me to discover JAHMM, a free-to-use Java implementation of various Hidden Markov Model algorithms including K-means, forward-backward, Baum-Welch, and Viterbi [16].

2.4 JAHMM

The documentation and examples are limited, which has meant a considerable amount of time thus far has been spent on learning how to use the classes effectively and to create working HMMs which implement clustering using random data, rather than using the motion sensors immediately. So far, the aim has been to understand how JAHMM represents HMMs and how best to use it for the purpose of the project. Using arbitrary input data it has been possible to achieve this. Contained within the package are various classes which directly implement some of the main features that are necessary.

A good gesture recognition example was found which tracks the movement of a computer mouse [17], and it has been a useful reference point for learning the intricacies and behaviour of JAHMM. The example is a tutorial from the University of Lille, meaning that many key points have been left unfinished as tasks to complete, which was an ideal situation for learning. It successfully clusters and trains a HMM using the relevant classes from JAHMM.

2.5 MIDI

As the end result of the system will be a MIDI event, it was important to understand what MIDI is and how it works. MIDI stands for Musical Instrument Digital Interface, and is a standard that allows digital musical instruments and computers to communicate, including hardware specifications to ensure a physical connection is always possible. A MIDI file is very small in size and contains all of the information required to create sound, and on a computer the sound card uses a built-in synthesizer with that information to create the sound. Multiple MIDI sounds can be played simultaneously, allowing for richer sounding digital instruments, and one instrument can remotely control another, leading to many possibilities. These features revolutionised the music industry in the early 80's.

Within this project I will be using the pre-existing Java package that supports MIDI and already contains a soundbank of various different instruments and sounds. By setting the various parameters and loading an instrument beforehand, a sound can be triggered by sending a *MidiChannel* object a *noteOn* message within a *Synthesizer* interface. When training a gesture, it should be possible for the user to assign a MIDI instrument and a

name to that gesture. Such an event will be triggered when a gesture is recognised, and the associated instrument will be played.

A more advanced method would involve outputting the MIDI data to external synthesizers to allow for greater scope and much more interesting sounds. This will possibly be done within this project, but the priority will be using the Java synthesiser.

3. Approach

For this project, I will be using a MotionStar Sensor device which creates a magnetic field, and reads input data in *x, y,* and *z* directions through the use of a physical sensor, attached via a cable, which disrupts the magnetic field. A final commercial version of this project would ideally be wireless, or use cameras similar to the Virtual Air Guitar [7], but as a proof of concept and considering the scope of the project, this wired application is suitable. The program should run continuously and recognise gestures instantly, otherwise it will not be of much use. Successful communication with the device for retrieval of information was achieved during another project [18] using the same hardware. The program will enable positioning data to be read from the MotionStar device by sniffing appropriate network packets and then interpreting them correctly. The entire project will be created using Java and the JAHMM library.

The basic flow of events for training one gesture will be:

- Capture multiple sets of training data (repeating the same gesture)
- Cluster the data using K-Means algorithm
- Create a HMM which is optimised, using the Baum-Welch algorithm
- Save the HMM

Once the HMM has been successfully saved and the program is running in normal mode (as opposed to 'training' mode), the system will run as follows:

- Capture live, continuous data
- Attempt to map the data to pre-existing clusters

- Run the Viterbi algorithm to calculate the probability that a gesture was being performed
- If the probability is above a certain threshold, proceed with the associated sound/event
- If it is not, ignore the data and continue evaluating.

The Viterbi algorithm finds the most likely sequence of hidden events that results in a sequence of observed events. It finds the most likely gesture given the most recently captured coordinates, and is fully supported in JAHMM.

As stated earlier, I will be using the Java MIDI synthesiser to generate a sound through my computer, which will be triggered by a high enough gesture probability produced by the Viterbi algorithm.

The number of clusters to use with the k-means step is still undecided, and I will be conducting an experiment to decide on the best number, balancing *correctly* identified gestures with the overall number of gestures recognised.

4. Conclusions

Up until this point, most of the work completed has consisted of researching appropriate methods of doing the project, and how they will help me achieve the end goal. I have decided the best way to complete the project is using Hidden Markov Models along with the related algorithms (K-Means, Viterbi, Baum-Welch) to enable gesture recognition, which are all implemented in a purpose-built, free to use Java library called JAHMM. HMMs have proven to be a reliable method of gesture recognition in numerous real-world applications for many years, and I believe they are a good choice for my own version. The priority and main requirement for the output is to have working gesture-recognised sounds using the Java synthesiser, however an optional requirement would involve the use of external synthesisers. The idea would then be to have greater, more advanced control of the sounds, as well applying effects to a constant stream of music.

The programming work completed has been experimental thus far; simply establishing the behaviour of HMMs as well as their JAHMM implementation, allowing for a solid foundation to build upon for the next section of real implementation. Ideally, work on

the final program would have begun already but delving into the theory behind HMMs and learning how to create them using JAHMM has taken longer than expected. However I am now confident that the project will be completed successfully, and am hopeful that an interesting finished product can be achieved.

Next term I will be working with the hardware as often as possible and experimenting with the program to get the best results. By the end of week 2 I aim to be able to retrieve MotionStar data, and record and recognise a basic gesture. From then I will work on the output (MIDI sounds). By week 4 at the latest I aim to have a solid foundation of a program to build upon in order to work on more advanced features and sounds, as well as optimise the performance of the system. This will hopefully give me plenty of time to go beyond the basic requirements and build a more interesting product that is capable of more than recognising just simple gestures.

References

[1] Microsoft Xbox Kinect, Online, Available at: http://www.xbox.com/en-US/kinect

[2] Microsoft Kinect 'fastest-selling device on record', 2011, BBC News [Online, accessed 9/12/12], available at: http://www.bbc.co.uk/news/business-12697975

[3] L. Sucar et al, 2008, Gesture Therapy, International Conference on Health Informatics

[4] Software package aids motor recovery (at Ulster University), 2011, by Andrew Czyzewski, The Engineer [online, accessed 11/12/12], available at: http://www.theengineer.co.uk/sectors/medical-and-healthcare/news/software-package-aids-motor-recovery-in-stroke-patients/1008143.article

[5] Virtual Harp, Algonquin College, 2012 [online, accessed 29/11/12], available at: http://www.youtube.com/watch?v=kEJ3_2H11tw&list=PL69C1948A382B76D1&index=2

[6] Gestrument App Translates Gestures Into MIDI, 2012, Synthtopia [online, accessed 28/11/12], available at: http://www.synthtopia.com/content/2012/11/14/gestrument-app-translates-gestures-into-midi/

[7] Future Development of Gestrument, 2011 [online, accessed 3/12/12], available at: http://www.gestrument.com/, video demonstration at http://player.vimeo.com/video/24004279

[8] Karjalainen et al, 2006, Virtual Air Guitar, J. Audio Eng Soc 54(10), pp 964 – 980

[9] Mudit – A New Gestural Controller For Music, 2012, Synthtopia [online, accessed 8/12/12], available at: http://www.synthtopia.com/content/2012/12/07/mudit-a-new-gestural-controller-for-music/

[10] Michael J. Lyons & Nobuji Tetsutani, 2001, “Facing the Music: A Facial Action Controlled Musical Interface” Conference on Human Factors in Computing Systems
March 31 - April 5, Seattle, pp. 309-310

[11] Rabiner, L. R, 1989, "A tutorial on hidden Markov models and selected applications in speech recognition.", Proceedings of the IEEE 77(2): 257-286.

[12] Tie Yang, Yangsheng Xu , 1994, “Hidden Markov Models For Gesture Recognition”, Carnegie Mellon University

[13] Wood. D, Methods For Multi-touch Gesture Recognition For Games, University of Cape Town

[14] Kadous, W, 2002, Computer Based Machine Learning Unit, PhD Thesis, University of New South Wales

[15] Laure. P et al, 2011, Modified K-Means Clustering Method Of HMM States For Initialisation of Baum-Welch Training Algorithm, 19th European Signal Processing Conference (EUSIPCO 2011)

[16] François, J.-M. (2006). "Jahmm - Hidden Markov Model (HMM).", [online, accessed 10/12/12], available at: http://code.google.com/p/jahmm/

[17] Hidden Markov Models, Mouse Gesture Recognition, 2012, University of Lille, [online,

accessed 10/12/12], available at [In French]: http://www.lifl.fr/~casiez/VisA/TP/TPMarkov/.

[18] Bowen. D, 2010, Motion Capture To MIDI, Final Year Project, Cardiff School Of Computer Science.