Computer Science Honours Project Short Paper

Interactive Synthesis of Avatar Motion from Preprocessed Motion Data

by Andrew Peirson.
Department of Computer Science, Rhodes University
29 September 2003

Supervisors: Shaun Bangay, Adele Lobb and George Wells.

Abstract

A recent dilemma being faced by animators, especially within the context of three-dimensional games and virtual environments, is the ability to produce natural avatar motion in real-time. Motion capture is one of the most effective methods of animating realistic human motion. However, one drawback of the motion capture process is the fact that the range of avatar movement is limited to the set of avatar behaviours that have already been produced. The aim of this project is to research and implement an approach, based on a paper by Lee et al (2002), to synthesize new motion for a controllable avatar from a set of preprocessed motion data. This paper discusses the issues surrounding the implementation of the chosen approach and how it differs from the approach used by Lee et al (2002). The results achieved to date are also discussed and an outline of the work still to be done is given.

1. Background

According to Arikan and Forsyth (2002), motion is one of the key elements of all computer graphics endeavours. Thus, there are a number of applications that require large collections of realistic looking motion. One of the most important forms of motion, especially for the three-dimensional computer gaming and movie industry, is that of human or avatar (animated character) movement. However, Lee et al (2002) state that the ability to produce natural avatar motion is a major dilemma being faced by animators worldwide.

Arikan and Forsyth (2002) describe three techniques used to obtain realistic motion, namely key framing, physically based modelling and motion capture. Using traditional key framing, it is relatively straightforward to construct motion for rigid objects through translational and rotational trajectory curves. However, manipulating and coordinating the limbs of a human figure via key frames is a complex task that requires lots of work and highly developed skills. Otte (2000) describes physically based modelling as the process of modelling the laws of physics and how they affect the motions of bodies. Once again, this method works well when applied to simple systems, but is difficult to use with more complex arrangements such as the human body. The last technique, motion capture, is the most effective method for producing realistic avatar motion. Real-time 3D motion capture equipment is used to record motion data for an approximate skeletal hierarchy of a live subject. This motion capture data can be used to facilitate conventional animation, or to create libraries of reusable clip-motion which serve as databases for on-the-fly assembly of animation in interactive systems. (Kim, 2000)

Unfortunately, motion capture has its own drawbacks which have affected its adoption and widespread use within the computer animation industry. Arikan et al (2002) identify one of the disadvantages as being the prohibitive costs of purchasing and using a motion capture system. Actors and technicians need to be hired and the process is very time consuming. This dilemma is becoming less and less of a problem, however, due to the increased availability of free motion capture files from the Internet. Another difficulty is that the range of avatar movement is limited to the set of avatar behaviours that have already been captured. This means it is necessary to design a large repertoire of motion sequences to cope with a variety of human actions that will satisfy the animator's requirements. This requires extensive manual processing. Thus, a number of techniques to overcome this problem have been suggested by various authors, most based on

the use of a motion database (preprocessed graph of sample avatar motions) to generate further motions of the same style.

2. Project Overview and Objectives

It is the aim of this project to research and implement a solution to the above mentioned drawbacks of using motion capture data to produce avatar animation. The solution adopted is largely based on the technique described by Lee et al (2002) in their paper entitled "*Interactive control of avatars animated with human motion data*". While many video games today make use of a large motion database that consists of many short, carefully planned, labelled motion clips to provide natural avatar motion, Lee et al (2002) describe an alternative method which makes use of a pre-processed motion database of longer, unstructured sequences of motion capture data that allows the animator to synthesize new avatar motion based on simple input. In broad terms, their approach describes an algorithm that compares different motion capture files and identifies possible ways of generating new human motion by cutting and pasting existing motion capture data. An in-depth discussion of how the motion files are compared is given in the next section of the report, but the main project objectives are identified below:

Objective 1: Create *an archive of motion capture data in a common format*. At present, there are many different types of motion capture files available, each slightly different in terms of the way the skeleton hierarchy and the actual motion data relating to the various joints of the skeleton is stored. To compare different motion capture files, it is necessary to build up a large database of motion capture data of the same format. This implies that a single file format must be chosen and used throughout the project.

Objective 2: Develop *a tool to animate the motion data using OpenGL*. The tool must be able to read motion capture files of the chosen format and then animate an avatar based on the motion data stored in the file.

Objective 3: *Investigate and implement an approach to synthesize new motion sequences from existing motion data given some simple user input*. In addition to displaying the avatar motion in a 3D environment, the tool should allow the animator to generate and store new motion sequences based on some simple input.

Objective 4: *Integrate this technique for use with the open source, 3D modelling and animation environment, Blender*. In order to make the tool more useful, it should allow for easy integration with an external 3D environment such as Blender.

3. The Approach Followed

The following section of the report describes the procedures taken during the initial phases of implementation. It also discusses the theory behind the approach to motion synthesis adopted by Lee et al (2002). The implementation procedures have been divided into three main sections as explained below:

3.1 Building the Archive

According to Lee et al (2002), the **size** and **quality** of the underlying archive of motion capture data is key to the success of their approach. While their raw data was captured using a Vicon optical motion capture system, such equipment was not available for me to use, so it was necessary to turn to other possible sources of motion capture data. Fortunately, there are several websites that offer free motion capture data for anyone to download. References to these websites are provided on my "Resources" page, accessible from my project webpage[1].
Due to the success of motion capture, a number of companies have been established that can record and provide motion data. Unfortunately, most of these production houses[2] have developed their own file format which has led to a lack of standardization. Maddock and Meredith (2001) have performed extensive research on the various motion capture file formats in use today and have identified a total of eight different formats. After researching the various advantages and disadvantages of these formats, I decided to use the **BVH**[3] file format. The motive behind my choice is three fold. Firstly, **BVH** tends to be one of the more common formats used. Thus, there are a number of sources on the Internet that together provide a wide variety of free **BVH** motion sequences. Secondly, good format specification documentation is available for the **BVH** file format, which made it easier to parse and analyse the data stored by each **BVH** file. Lastly, each **BVH** file is divided into two sections; the first describing the skeleton hierarchy for the avatar and

[1] http://www.cs.ru.ac.za/research/students/g00p1523/
[2] Examples of production houses include BioVision, http://www.biovision.com, Motion Analysis, http://www.motionanalysis.com, House of Moves, http://www.moves.com and e-Motek, http://www.e-motek.com
[3] BVH – BioVision Hierarchical Motion File Format.

its initial pose and the second describing the frame by frame motion of the avatar. Using this structure, it is easy to distinguish between the skeleton object and its associated motion data.

Having chosen a specific motion capture file format, I began building up my motion archive. To help with the process of updating the online version of the archive, I developed a simple shell script that automatically adds all new motion files to the archive's web interface.

After collecting enough raw data to proceed to the next phase of implementation, I realised it would be necessary to analyse the different skeletal structures of the **BVH** files to ensure that it would be possible to compare them later on when implementing the motion synthesis algorithm described by Lee et al (2002). The figure below illustrates the results obtained when comparing **BVH** files from eight different sources based on the number of joints (bones) each describe.

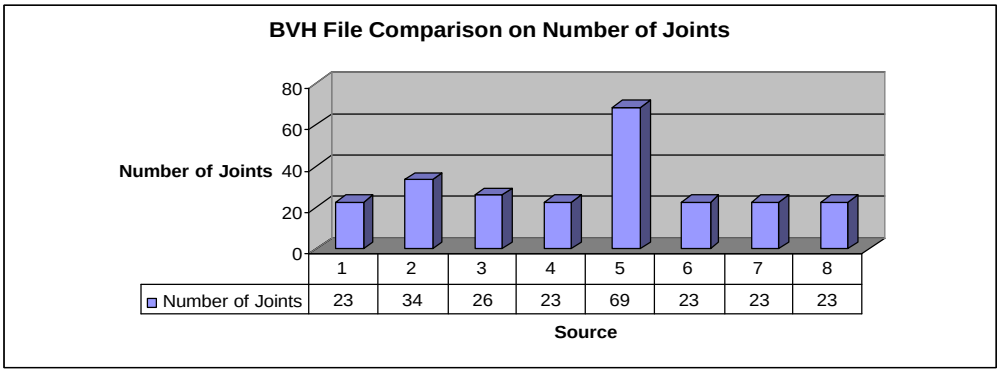

Figure 1: BVH File Comparison Based on the Number of Joints Defined in the Skeleton Hierarchy.

Figure 1 shows that although a number of the sources investigated use twenty three joints to define their skeletons, there are still some other sources that use varying numbers of joints. Also, after further analysis, it was discovered that the names used to describe each joint did not match from one source to another. This meant that it would not be possible to compare motions **across sources** using the existing **BVH** files.

Therefore, it is necessary to define each BVH file as having a common skeleton hierarchical structure. Hence, the next step in implementation was to develop a tool that could convert any **BVH** file to a **BVH** file describing the same motion sequence, but using a standard maximal skeleton hierarchy. Part of the chosen maximal skeleton hierarchy is illustrated over the page:

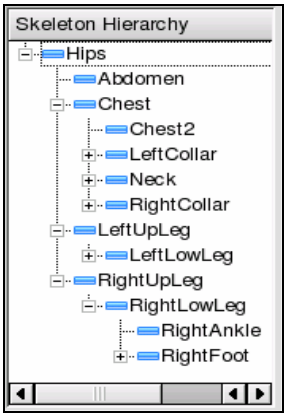

Figure 2: Maximal Skeleton Hierarchy

The conversion process consists of two main phases:

Phase 1: For each joint name used by the existing **BVH** file, a list of valid synonyms is searched and if the name is found, it is changed to the corresponding maximal skeleton joint name. This eliminates the problem caused by different naming conventions between sources of **BVH** files.

Phase 2: The skeleton hierarchy section of a **BVH** file identifies the structure of joints defining the skeleton as well as the initial pose of the skeleton. The joint angles describing the motion of the skeleton are all calculated relative to the skeleton's initial pose. This means that when converting **BVH** files with different initial poses to that of the maximal skeleton, it is necessary to slightly offset each joint angle so that the new initial pose is taken into account and the motion sequence is not corrupted.

3.2 Building the Motion Display Tool

Having completed the first project objective, building a suitable archive of raw motion capture data in a common format, the next step was to develop a simple tool to animate the motion data. Implementation of the tool was carried out in C++, making use of the OpenGL graphics library and Trolltech's Qt windowing toolkit. In order to create tools that could be easily integrated with existing tools being used in the department, all development was done under Linux RedHat 8.0. Animating the motion data found in the **BVH** files is done in two steps. Firstly, the **BVH** file is parsed and all the important information is extracted and stored in appropriate data structures. Secondly, the data is processed to produce the avatar's motion. These steps are described in more detail below:

Step 1: Parsing the BVH File

The skeleton hierarchy of a **BVH** file is essentially recursive in nature, which means that a simple recursive decent parser can be used to extract the skeleton data. The motion data, on the other hand, can be parsed line by line and stored in a matrix of channels, where each channel describes three separate joint angles, representing rotation of translation about the X, Y and Z axes.

Step 2: Processing the Extracted Data

The following algorithm is used to obtain the position of a bone segment for a particular frame in the motion sequence:

- Obtain local translation and rotation information.
    - For each joint (Skeleton Node), translation information comes from the offset defined in the hierarchy section of the **BVH** file.
    - For each joint, rotation information comes from the motion data and the quaternions for each rotation channel.
    - For the root joint, translation information = offset data + root's Translation channel data.
- Once the local transformation is created then concentrate it with the local transformation of its parent, then its grand parent etc.

OpenGL's matrix and stack operation functions are used to keep track of parent and child transformations using a recursive method. Once the position of each bone segment is known, all that is left to do is to render a 3D representation of the bone. AC3D, a 3D object editor, was used to create individual 3D **OFF**[4] objects for each bone by dividing up a large skeleton object. Parsing and rendering of the **OFF** object files was done using code developed by Bangay (2003).

3.3 Incorporating the Process of Motion Synthesis

Phase three of the project implementation involved researching and incorporating the algorithms described by Lee et al (2002) so that a preprocessed motion database could be constructed from some of the raw data in the underlying motion archive. This preprocessed motion database could then be used to synthesize new avatar motion based on some simple rules and user input.

[4] OFF – Object File Format

Theory behind the Motion Synthesis Process

In their paper[5], Lee et al (2002) discuss how raw motion data from the motion archive can be "preprocessed for flexibility and efficient search and exploited for real-time avatar control". This is done by identifying likely transitions between motion segments based on good matches in poses and velocities of the character. They provide a method of efficiently searching the resulting graph structure, known as "clustering". Thus the data representation of the human motion suggested by Lee et al (2002) consists of a two-layer structure. "The higher layer is a statistical model that provides support for the user interfaces by clustering the data to capture similarities among character states. The lower layer is a Markov process that creates new motion sequences by selecting transitions between motion frames based on the high-level directions of the user." (Lee et al, 2002) The process of clustering will not be discussed further as it has not been implemented in the chosen approach. However, implementation of the Markov process is at the heart of my project so will be examined in more detail.

The Markov Process

Based on similar work done by Schodl et al (2000), the motion data is modelled using a first-order Markov process. The result of the process is represented as a matrix of probabilities $\mathbf{P_{ij}}$ describing the likelihood of transitioning from frame **i** to frame **j**. $\mathbf{P_{ij}}$ is estimated using an exponential function that maps a measure of similarity between frames ($\mathbf{D_{ij}}$) to a value between 0 and 1:

Equation 1: Used to calculate the Probability of a good transition from frame **i** to frame **j**.

The variable **s** in the above equation is used to control the steepness of the slope described by the exponential function. The following figure illustrates three different graphs obtained by using different values for **s** .

[5] "Interactive control of avatars animated with human motion data."

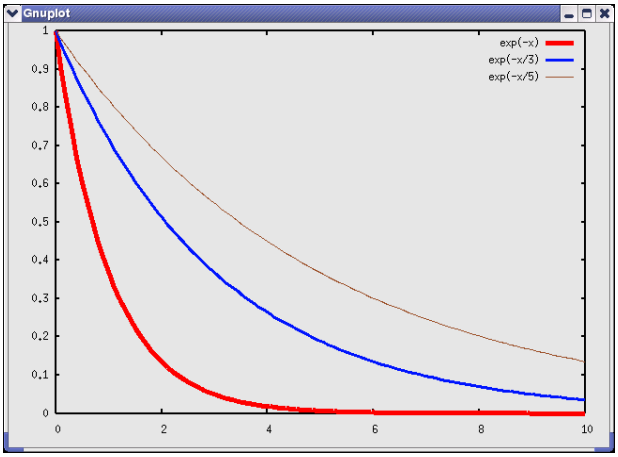

Figure 3: Graphs of Equation 1 on the previous page, using 3 different values for **s**: 1, 3 and 5.

The distance measure $\mathbf{D_{ij}}$ takes into account the weighted differences of joint angles as well as the weighted differences of joint velocities. Analysing the joint angles ensures that the difference in character pose from one frame to another is not significant, while analysing the joint velocities helps to preserve the dynamics of the motion.

The resulting matrix of probabilities is extensive and requires a large amount of storage space. Since most of the frame to frame permutations result in low probabilities, Lee et al (2002) suggest pruning the database to reduce the storage space required and improve the quality of the resulting motion by avoiding too frequent transitions. Pruning is done based on a number of criteria, one of which is ignoring all transitions with probabilities below a user-specified threshold.

Implementation Issues

As mentioned earlier, no attempt has been made to summarize the motion data using clustering. Instead, implementation of the motion synthesis process has focused on the Markov process alone. Since the motion archive I am using is built from raw motion data collected from several different sources, it is very difficult to find good frame transition points between two different motion sequences based on the original formulas for the Markov process used by Lee et al (2002). Thus, it was necessary to experiment with different versions of the formulas that weren't so strict in eliminating the possibility of transitioning to slightly disparate frames. One of the major changes made was to the term of $\mathbf{D_{ij}}$ that estimates the weighted differences of joint angles. Originally, this term took into account the difference in position of the root skeleton node. This meant that very few possible transitions were found because most motion data from different

sources did not occur in the same 3D space. To solve this problem, this part of the equation was removed so that frames were only compared based on joint angles. Then to produce a realistic looking translation motion from one frame to the next, a relative translation value is calculated by comparing the changes in the root position between the two motions.

Also, instead of creating a matrix of all possible transition probabilities and then pruning it at a later stage, I decided to make use of a list to keep track of all transitions with a probability above the threshold value mentioned earlier. The user has the option of creating a number of different **motion databases** and then saving them to file so that the frame matching process does not have to be repeated each time the program is restarted. Each database consists of a list of the motion files and the results of applying the Markov process to these motion files. At present, new motion is generated by randomly selecting a possible transition frame from those options stored in the motion database. In this way, the new motion sequence begins with frame 0 from the **base motion** and then consists of a randomly selected combination of motion segments from the underlying raw motion data.

4. Current Status and Progress to Date

Objective 1: *Create an archive of motion capture data in a common format*.
At present, I have built up an online archive of raw motion data consisting of approximately 300 **BVH** files in 11 different categories ranging from dancing to death. A shell script has been written that automatically generates the **HTML** code to update this archive. Also, to ensure that all the **BVH** files can be compared using the Markov process, a **BVH** converter has been implemented to standardize the motion files to use a maximal skeleton structure. The figure below illustrates the results of converting a sample **BVH** file. The larger, darker skeleton structure represents the maximal skeleton, while the smaller, lighter skeleton represents the original **BVH** skeleton structure.

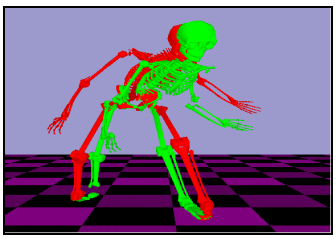

Figure 4: Results of using the **BVH** converter to create a maximal skeleton hierarchical structure.

Unfortunately, there are still some errors in writing the converted joint angles to file. The equations that are being used to convert the quaternions[6] back to Euler angles seem to be introducing a few errors with the joint angles. Possible solutions to this will be discussed in the next section, under work still to be done.

Objective 2: *Develop a tool to animate the motion data using OpenGL.*

A simple motion display tool has been developed. This will help the animator control the motion synthesis process and view the results of combining different motion files. Additional functionality such as zooming, rotating and translating the 3D view using the mouse has been added to the interface to help the animator view the motion from the best angle and position. A playback control panel has also been created which allows the user to playback, loop and stop the current motion. A camera tracking option has been implemented which ensures that the motion only occurs in the centre of the display screen (see Figure 5).

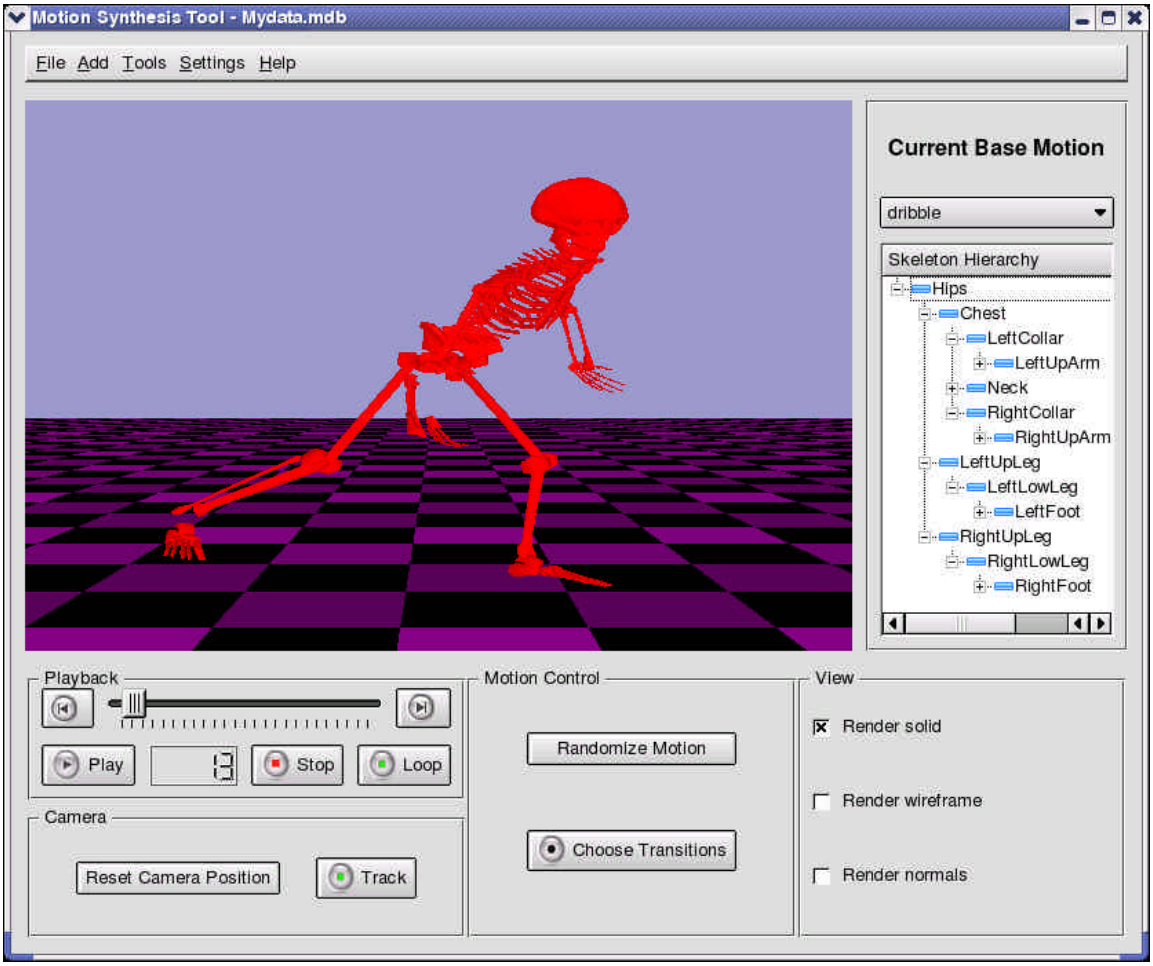

Figure 5: User Interface of the Motion Display Tool.

[6] A quaternion represents a rotation about a vector.

Objective 3: *Investigate and implement an approach to synthesize new motion sequences from existing motion data given some simple user input*. Based on the Markov process explained by Lee et al (2002), the motion display tool has been extended to allow the user to create new avatar motion. The main features provided by this enhancement include:

- Ability to create a new Motion Database File using the standard settings for the threshold. When a new motion database is created, the user selects an initial base motion. This essentially sets up the skeleton structure that all additional motion files added to this database must define.
- Ability to load an existing Motion Database that has already been processed.
- Allows the user to save a newly generated Motion Database. This means that the Markov process does not have to be repeated the next time the database is needed.
- Allows the user to save the current **BVH** motion file.
- The user can add new motion files to the current motion database and thus update the associated list of possible transitions.
- A button has been provided that allows the user to randomize the motion being displayed. In this case, a new motion sequence is generated by randomly selecting new transition points from the current base motion.
- The tool also shows the skeleton hierarchy of the motion database's base motion. This helps the user select new motion files that match in terms of skeleton hierarchy.
- Allows the user to change the base motion thereby generating a new starting frame for the current motion.

The screenshots below (Figure 6) illustrate a transition point discovered by applying the Markov process to a motion database consisting of a drunk walk motion and a basketball dunk motion using a low threshold value of 0.6. The screenshot on the left shows the frame from the drunk walk and the screenshot on the right shows the frame from the dunk motion.

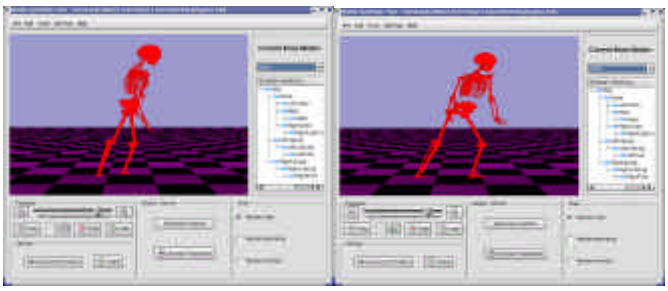

Figure 6: Two consecutive frames from a newly generated motion.

5. Work Still to be Done

One of the main project objectives still to be implemented involves linking the motion synthesis tool to Blender. Other enhancements and work still to be done include the following:

- The quaternion mathematical functions used by the **BVH** converter could be improved so that the converted **BVH** files can be stored accurately and used by the motion synthesis tool to create larger motion databases. If the maths used cannot be updated, then an alternative solution is to store the new rotations as quaternions in intermediate files.
- Incorporate the **BVH** converter with the Motion Synthesis Tool by making it available from the menu bar.
- Give the user the option to change the Markov process settings used when generating the list of possible transitions.
- Allow the user to select between the most likely transitions using a simple interface technique based on showing the paths taken by the skeleton for each possible motion sequence.
- Carry out experiments to evaluate the results of the Markov process using different constants and threshold values.
- Carry out experiments to see the differences in motion databases created when the root joint's position is taken into account and when it is left out of the Markov process comparison.
- Carry out experiments to apply motions to animals. By changing the structure of the maximal skeleton and the objects used, it may be possible to apply the generated motions to animal skeletons.
- Add extra functionality in terms of camera positioning and motion control options.
- Eliminate the problem of dead ends. At present, no algorithms have been implemented to avoid dead ends that are reached when generating the new motion sequence. This means that the new motion sequence sometimes gets stuck on a single frame and thus no movement occurs.

7. References

Arikan O and Forsyth, D. A. (2002)	*Interactive motion generation from examples: SIGGRAPH 2002, Computer Graphics Proceedings*, ACM Press New York, NY, USA pp. 483--490
Bangay S (2003)	Code for an OFF object reader.
Kim J (2000)	Motion Editing Techniques for Realistic Human Motion Generation. [On-line]. Available: http://www.seas.gwu.edu/~graphics/cs367/survey-motion.pdf
Lee J, Chai J, Reitsma P, Hodgins J and Pollard N (2002)	*Interactive control of avatars animated with human motion data: SIGGRAPH 2002, Computer Graphics Proceedings*, ACM Press New York, NY, USA pp. 491--500
Meredith M and Maddock S (2001)	Motion capture file formats explained, Department of Computer Science Technical Report CS-01-11, University of Sheffield.
Otte R (2000)	Physically Based Modelling and Animation of Rigid Body Systems. [On-line]. Available: http://www.cs.auckland.ac.nz/GG/Otte/RobinOtte_Thesis.pdf
Schodl A, Szeliski R, Salesin D.H. and Essa I. 2000	*Video textures: SIGGRAPH 2000, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, Kurt Akeley pp. 489--498.