INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

Character Animation from Motion Capture Data

Technical Report

Adriana Schulz

Instructor: Luiz Velho

Rio de Janeiro, March 4, 2010

Contents

Chapter 1

Introduction

Realistic animation of human motion is a challenging task, firstly, because human movements can be quite complex since joints have many degrees of freedom and, secondly, because people are very sensitive to inaccuracies in rendered motion.

In this context, motion capture comes out as a very interesting technique for character animation. Some of the advantages of motion capture over traditional computer animation of a 3D model are:

- More rapid, even real time results can be obtained. In entertainment applications, this can reduce the costs of keyframe-based animation.
- The amount of work does not vary with the complexity or length of the performance to the same degree as when using traditional techniques. This allows many tests to be done with different styles or deliveries.
- Complex movement and realistic physical interactions, such as secondary motions, weight and exchange of forces, can be easily recreated in a physically accurate manner.
- The amount of animation data that can be produced within a given time is extremely large when compared to traditional animation techniques. This contributes to both cost effectiveness and meeting production deadlines.

Nevertheless, MoCap by itself only allows for the reproduction of an acquired motion and, therefore, much effort has been, and is currently being, put into extending the applications of MoCap data.

This technical report describes a course done at IMPA in January/February 2010 under the supervision of Professor Luiz Velho. The objective of this course is to study character animation based on motion capture data.

In Chapter 2, we describe the first part of the course which aims at understanding the MoCap technique and building a database which will be used in future projects. Motion capture was done in the Visgraf Laboratory using OPTITRACK cameras and the ARENA software to process the acquired data and export a bvh file.

In Chapter 3, we describe the second part of the course which involves filtering and editing motion data. We studied signal processing transformations applied to motion and implemented a cartoon animation filter [1], which aims at making the motion more "animated" by adding effects, such as anticipation, follow-through, exaggeration and squash-and-stretch.

In Chapter 4, we discuss the problem of synthesizing new streams of motion from previously acquired data. We implemented a motion graph [2] that cuts pieces from a motion database and reassembles them in order to form a new motion.

Finally, in Chapter 5, we present some conclusions of our study, interesting application and future directions.

Chapter 2

Motion Data Acquisition

The fist part of this course involves acquiring motion capture data. In this chapter we discuss the most common motion capture techniques and describe the system we used to create a motion database.

Altough there are many kinds of motion capture systems (such as mechanical and magnetic systems), the most common methods make use of Optical systems.

Optical systems utilize data captured from image sensors to triangulate the 3D position of a subject between one or more cameras calibrated to provide overlapping projections. Data acquisition is traditionally implemented using special markers attached to an actor[1]. These systems produce data with 3 degrees of freedom for each marker, and rotational information must be inferred from the relative orientation of the markers.

As with traditional animation and many other arts, MoCap is actually composed of a number of phases:

- studio set-up,
- calibration of capture area,
- performance and capture of movement,
- clean-up of data, and
- post-processing of data.

[1]More recent systems are able to generate accurate data by tracking surface features identified dynamically for each particular subject. However, the markerless approach to motion capture will not be studied in this course.

Motion capture was done in the Visgraf Laboratory using six OPTITRACK cameras, as shown in Figure 2.1. Tracking a large number of performers or expanding the capture area is accomplished by the addition of more cameras. Since each marker must be "seen" by at least two cameras, a greater number of cameras diminishes the possibility of occlusions.

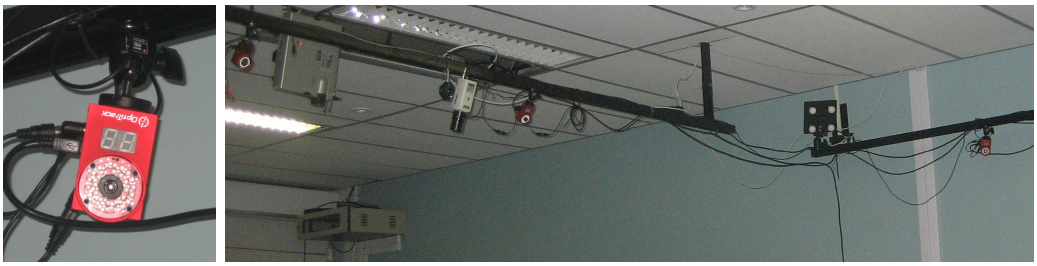


Figure 2.1: OPTITRACK cameras set up at the Visgraf Laboratory.

Performers wear a special set of cloths with retroreflexive markers that are captured by the infrared cameras (see Figure 2.2). These markers are placed in a way that the position of the rigid body parts are easily acquired. In most cases, we use three markers placed in a way to form a triangle shape, as shown in Figure 2.2 .

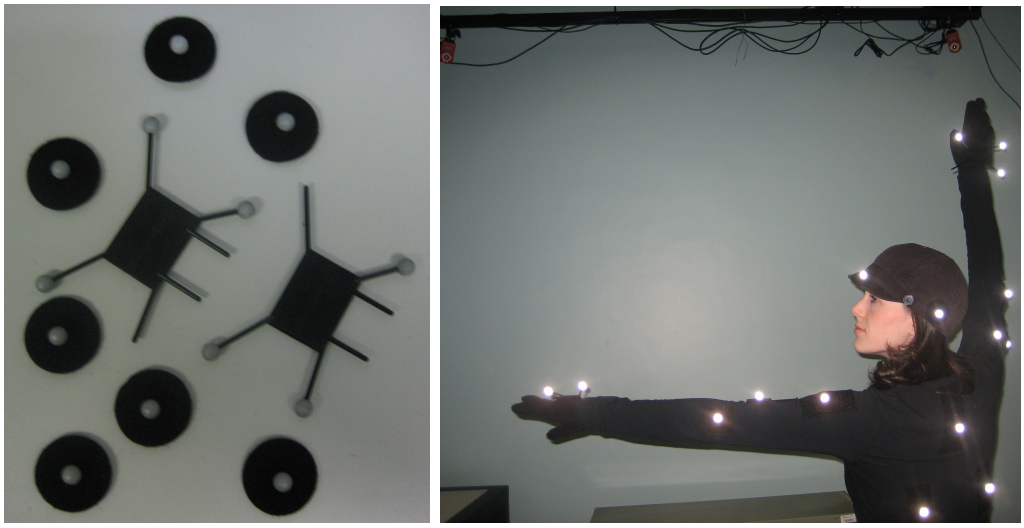


Figure 2.2: Retroreflexive markers.

The ARENA software (see Figure 2.3)was used to process the acquired

data. The first processing step is to trajectorize the data. This procedure takes the 2D data from each individual camera and changes it to a fully 3D path of each individual marker. After this procedure the software allows for post capture editing (such as filling gaps, fixing swaps and smoothing tracks) and exporting a file in a BVH or C3D format.

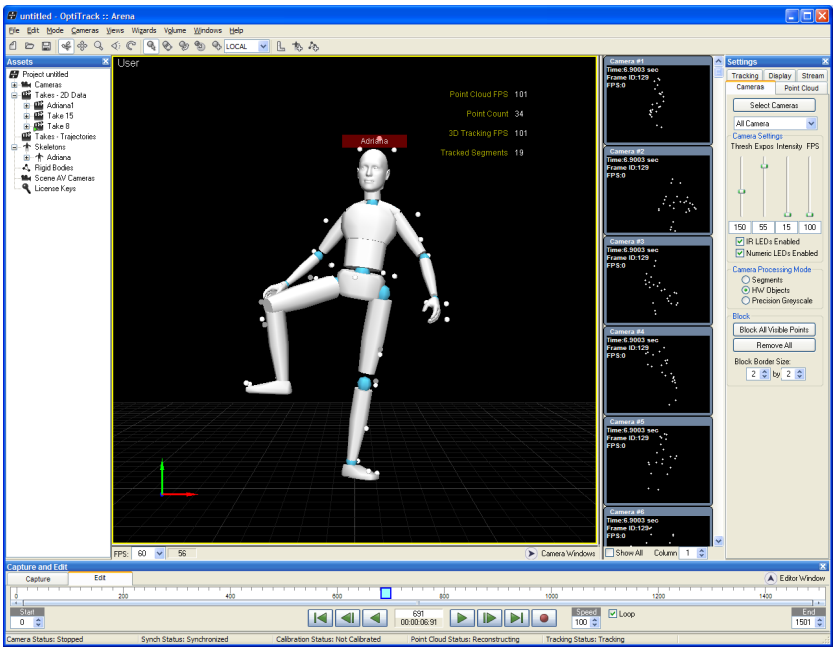
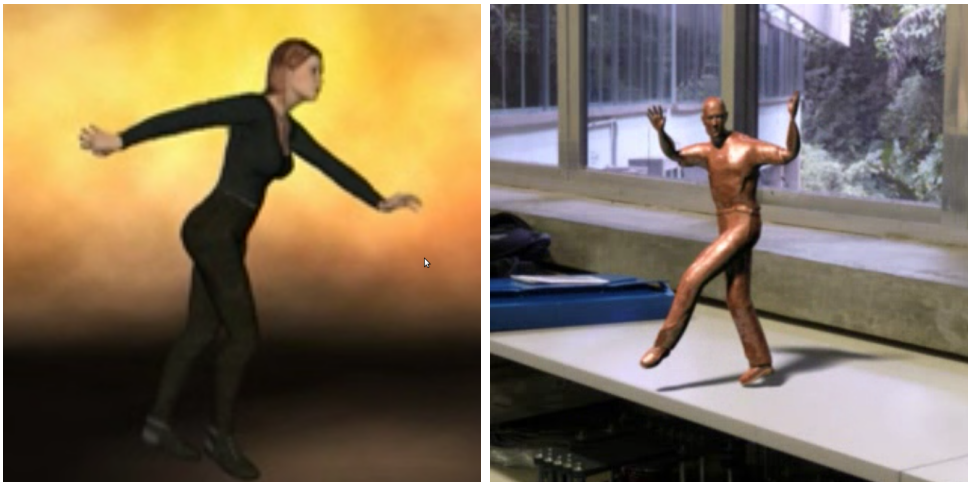


Figure 2.3: The ARENA software.

The BVH file format was originally developed by Biovision, a motion capture services company, as a way to provide motion capture data to their customers. Nowadays, it is widely used, and many applications support importing and exporting files in this format. It consists of two parts, a header section which describes the hierarchy and initial pose of the skeleton, and a data section which contains the motion data.

We built a motion database using this file format and we used DAZ studio and Maya software to read the BVH files and create character animation based on the acquired motions. Figure 2.4 illustrates the rendering of a dance motion.

We captured movements of dance and fencing as well as some simple movements such as walking, playing golf, bending and moving arms and



(a) (b)

Figure 2.4: In (a) is shown the rendering done in DAZ studio and (b) is shown the rendering done in Maya.

legs. A complete motion database will soon be available at the Visgraf web site.

Chapter 3

Motion Editing and Filtering

There are many reasons that make editing captured motion extremely important. Firstly, it is usually necessary to eliminate artifacts generated during acquisition. Secondly, it is important to match time and space of computer generated environments, overcome spatial constraints of capture studios and allow for the existence of motions that would be extremely hard for an actor to perform, such as the ones used in special effects. Finally, it is interesting to be able to reuse motion data in different occasions. For example, given a walking scene, it should be possible to generate a walk on an uneven terrain or steeping over an obstacle.

Therefore, the second part of this course involved editing the acquired MoCap data.

Because there are many degrees of freedom animation and movements can be quite complex, when developing such editing tools it is important to bear in mind not only efficiency but also simplicity. In this context, methods that make use of signal processing techniques prove to be quite interesting for motion editing [3, 4]

In the next section we discuss one of the many existent methods for motion filtering and editing.

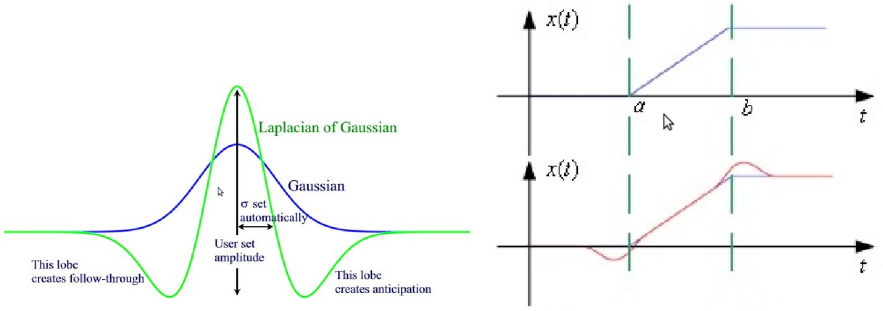
3.1 Cartoon Animation Filter

We implemented the Cartoon Animation Filter [1], which aims at making the motion more "animated" by adding effects of traditional animation. Methods for manipulating motion in order to emphasize primary actions and bring the

scene to life include anticipation, follow-through, exaggeration and squash-and-stretch[5].

Exaggeration can be used to focus attention and stress a particular action. Anticipation effects introduces an action before the character actually performs it. Follow through refers to the way the character behaves while completing an action. For example, when throwing a ball, the action continues after releasing the ball modifying the way the characters arm moves. Squash and stretch effects are the exaggeration to what happens for example if we bounce a rubber ball: it squashes up as it hits the ground, and then it stretches out. These effects define the rigidity of an object by distoring its shape during an action.

The method described in [1] implements the four motion manipulations described above by using the filter illustrated in Figure 3.1. The two external lobes generate the anticipation and follow-through effects and the middle lobe produces the exaggeration effect. Though we did not implement it during this work, [1] shows how time-shifted versions of this filter can be used to generate and squash-and-stretch movements.



(a) (b)

Figure 3.1: In (a) is shown the cartoon animation filter and in (b) the result of its aplication to a ramp singnal.

Each individual degree-of-freedom (DOF) motion curve from the MoCap session is filtered individually and independently. The simplicity of this approach is quite interesting because, as we have previously mentioned, processing MoCap data can be very challenging, since it yields a quite large and an unstructured representation (a sequence of joint angle values).

We implemented this filter in MATLAB and applied it to the motions

we had previously captured. Some of the results we obtained are shown in Figures 3.2 and 3.3.

Figure 3.2: Results of our implementation of the cartoon animation filter. The First row is the input motion of a golf swing and the second row is the filtered result

It is noteworthy that when signal processing methods are used to transform the original acquired movements, physically impossible motion can be produced. Because the editing tools do not consider physical constraints, the resulting movement usually presents many undesirable artifacts. One particularly distracting artifact is when the characters feet move when they ought to remain planted, a condition known as footskate.

To solve these problems, post-processing techniques should be developed in order to adjust constraint violations.

Though the results we obtained do present some of these undesirable artifacts, it is not within the scope of this course to study or implement such techniques. Nevertheless, we did implement a transform that translates the root node in order to keep the character's feet on the ground, so the character does not "fly" during the scene.

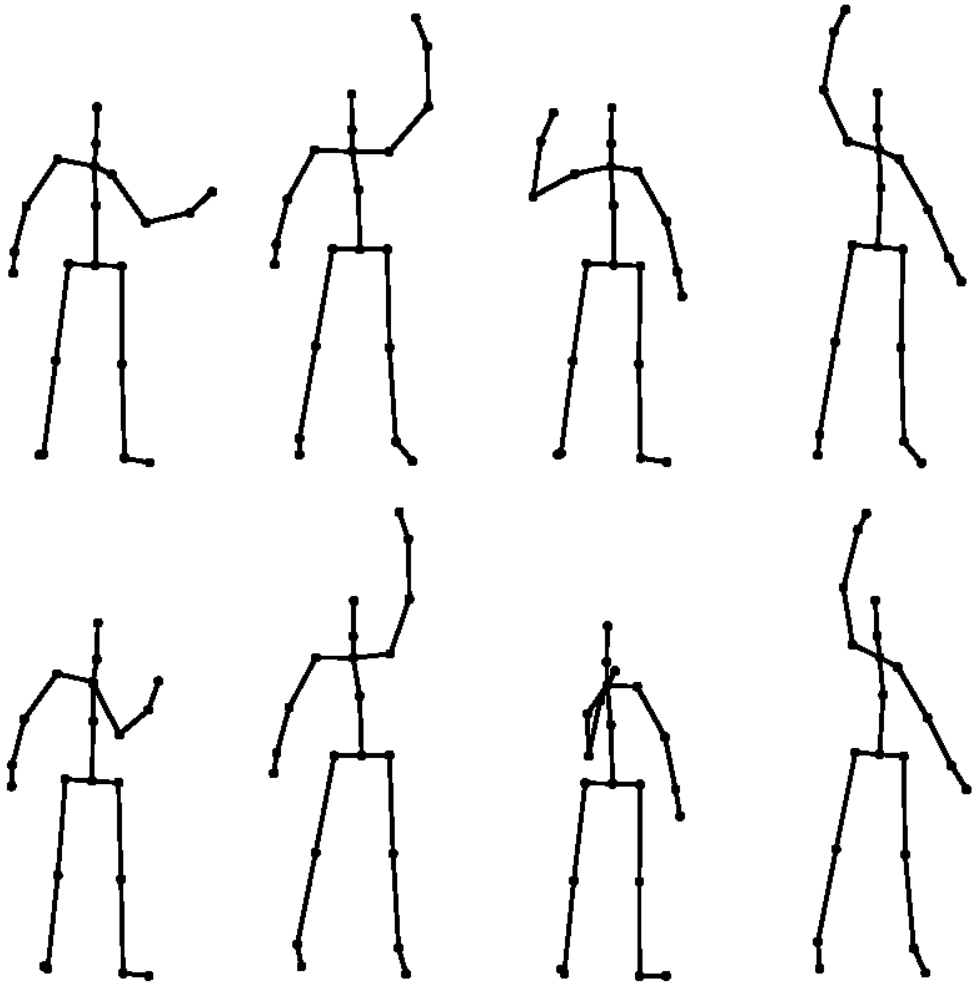


Figure 3.3: Results of our implementation of the cartoon animation filter. The First row is the input motion of arms stretching and the second row is the filtered result. Notice how the stretching of the arms is exaggerated and how the movement is anticipated as the hands go further down before starting to go up.

Chapter 4

Motion Synthesis

In the previous chapter, we discussed editing techniques that are able to modify previously acquired motion clips. However there is more to motion editing than simply altering individual clips.

It is the interest of many researches in the field to synthesize new streams of motion from previously acquired data and, therefore, be able to create new and more complex motions from previously acquired data.

Motion synthesis strategies include constructing models of human motion [6, 7], interpolating motion to create new sequences [8] and reordering motion clips employing a motion graph.

In this course, we studied motion synthesis from previously acquired data based on motion graphs. In the next section, we discuss the basic concepts and our implementation of the work of Kovar et al. [2].

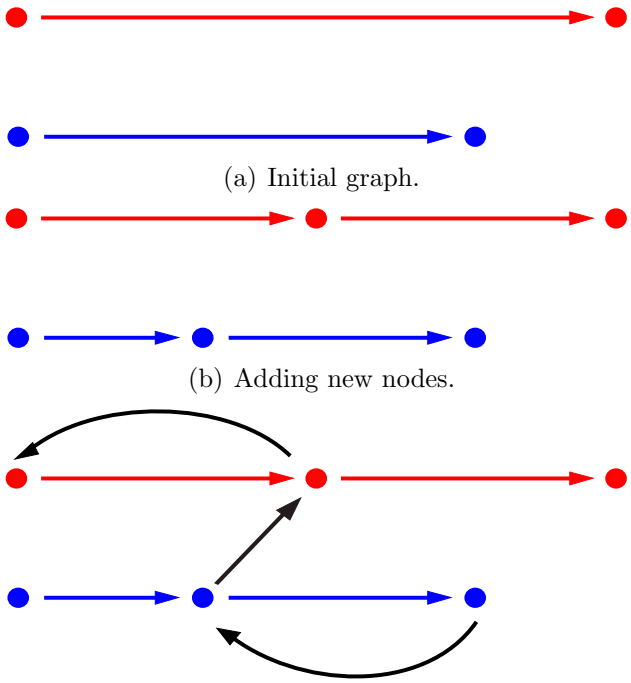
4.1 Motion Graphs

In [2], a motion graph is created in order to encapsulate connections among a database. In this graph, edges correspond to motion clips (sequence of frames) and nodes to choice points (specific frames) connecting these clips. After selecting the similar frames, or windows of frames, and creating the graph connections, a walk along the graph allows us to re-assemble the captured clips, creating new motion.

Algorithms from graph theory and AI planning can be used to synthesize a desired motion by extracting a graph walk that satisfies certain properties. It is not within the scope of this course to study such methods. Instead, we

implemented the framework responsible for constructing the motion graph and synthesized new motion by performing random graph walks. Future directions of this work will definitely include extracting motion from the motion graph in a fast and efficient way, meeting user specifications.

Figure 4.1 illustrates the construction of a motion graph. Initially we have two clips of motion from a MoCap database. The resulting graph is the one shown in Figure 4.1(a), which is disconnected and quite simple. Notice that it is possible to add nodes in the graph, simply by breaking the initial motion clips into two or more small clips, as shown in Figure 4.1(b). Nevertheless, this graph is still disconnected.



(a) Initial graph.

(b) Adding new nodes.

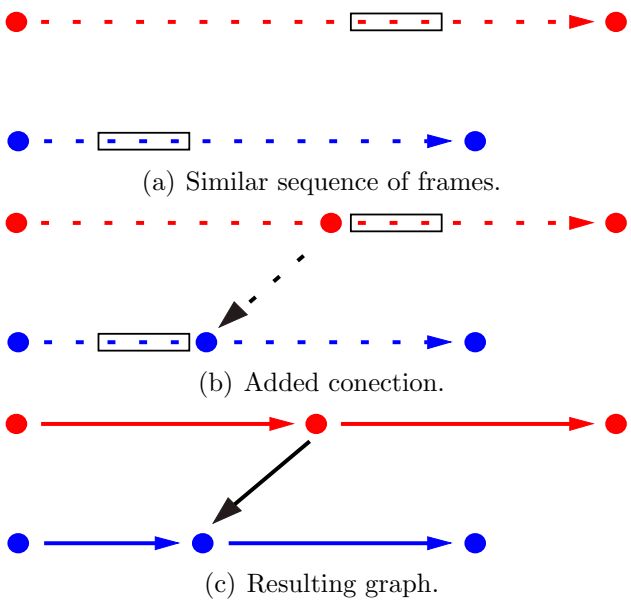
(c) Create greater connectivity.

Figure 4.1: Construction of a motion graph.

To create greater connectivity, we introduce transition clips, as shown in Figure 4.1(c). Notice that these new edges also represent clips of motion which have to be synthesized by the technique. Therefore in order to create these new edges we first have to select the nodes which can be efficiently connected and then create the motion clips to which the transition edges will correspond.

Following the ideas proposed in [2], we select the nodes which are sufficiently similar so that straightforward blending is almost certain to provide valid transitions and use a similarity norm that takes into account a sequence of frames and a point cloud driven by the skeleton .

As in the BVH file, a motion frame is fully described by the position and orientation of the root node and the rotation of the other joints. However, an attempt to use this information to calculate similarity between frames will be highly unsuccessful because some of these parameters have a much greater overall effect on the characters position than others (for example, a knee rotation is usually much more significant than a wrist rotation). In addition, it is quite hard to set fixed weights that work well in all cases, since the effect of a joint rotation on the shape of the body depends on the current configuration of the body.



(a) Similar sequence of frames.

(b) Added conection.

(c) Resulting graph.

Figure 4.2: Addition of a transition edge.

To solve this problem, we use a point cloud, which describe the skeleton pose (ideally this point cloud is a downsampling of the mesh defining the character) and vector distance norm. It is important to notice that a motion is fundamentally unchanged if we translate it along the floor plane or rotate it about the vertical axis. Therefore, before comparing the distance between two point clouds we apply a rigid 2D transform to the second point cloud

that sets it at the optimal position so that weighted sum of squared distances is minimal.

In addition, it is important to consider not only the current position but also derivative information, such as velocity and acceleration. We incorporate this information to the similarity metric by using a window of frames. This also helps the blending procedure. An example is shown in Figure 4.2. The similarity metric selects the two sequences of frames as shown in Figure 4.2(a). Therefore we can add an edge that connects the beginning of first sequence of frames to the end of the second one, shown in Figure 4.2(b). The resulting motion associated to this transition edge is created by blending the two sequences of frames.

Finally, after the edges are created, we prune the graph by computing the strongly connected components (SCCs) . We use only the largest SCC so as to guarantee that there are no dead ends, and therefore we can synthesize the motion indefinitely.

The user interface was developed in C++ using QT toolkit. We also imprevented a BVH file parser and a player that allows us to compare captured to rendered motion. As shown in Figure 4.3, the interface shows where each part of the rendered motion comes from. The graph transitions (blended motions) are represented in black and the original motion clips in an increasing saturation scale of color.

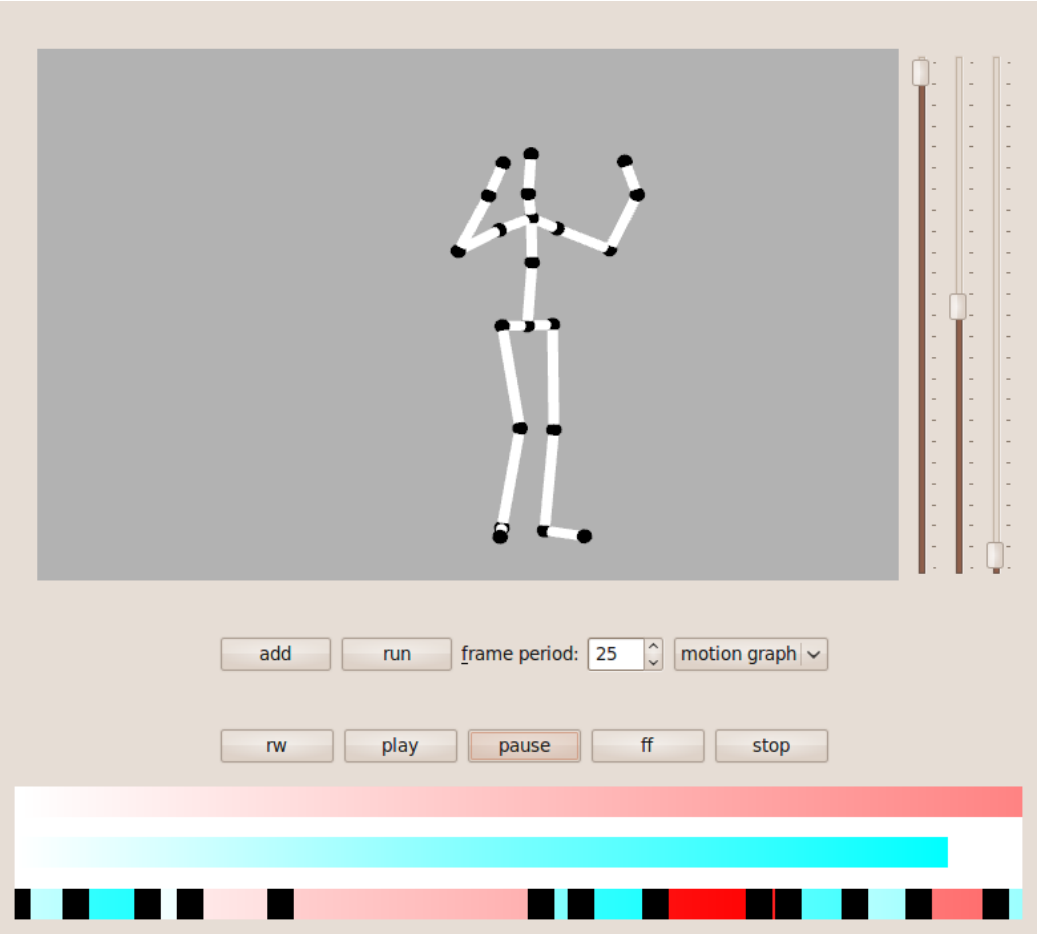


Figure 4.3: User interface that was implemented.

Chapter 5

Discussions and Future Work

During this course we studied capture, representation, editing, and synthesis of articulated body motion. We started by understating the acquisition procedure and building a motion database and then went on to study and implement algorithms that enable motion capture data to be edited and synthesized. We also developed an application for parsing a BVH file and playing a motion.

With this, we have set up the basic principles with which to work with character animation from MoCap data. It is our interest to further investigate the area and eventually contribute to a couple of problems in this field. In the next section we will present the future direction of our study. We will first discuss the problems which we consider to be intrinsic to the application of motion captured data to character animation and, finally, present a couple of ideas for future research in this field.

5.1 Future Directions

5.1.1 Intrinsic MoCap Problems

There were a couple of problems that are intrinsic to the application of motion capture data to character animation. Though it was not withing the scope of this course to study such problems, in the future it will be interesting to take these points into consideration.

First, there is the concern of mapping a motion to a 3D character. A character has limbs, muscles and other body features that must be altered

during the motion in order to allow a realistic rendering of the animation. Therefore, in order to create an animated character it is necessary to specify its internal skeletal structure (a process referred to as rigging) and to associate each bone in the skeleton with some portion of the character's visual representation (skinning).

In addition, there is the problem of retargetting [9]. In many applications, it may be important to apply a motion created by one articulated figure to another figure with identical structure (connectivity of the limbs, types of joints, number of degrees of freedom) but different segment lengths. For example, it an animator may want to map the movements of a tall slim woman to a short fat woman, of an adult to a child, of a man to a woman, or of a teenager to an old lady. In this scenario, it is important to guarantee that the adaptation preserves important aspects of the motion and avoids adding additional artifacts.

A second step would be to adapt the motion to environment. In most cases, MoCap is done in a small studios and on a flat surface. However, the animated 3D character should be able to interact with highly complex scenes. Therefore, MoCap data has to be edited in order to overcome spatial constraints and permit mapping to computer generated environments with may have uneven terrain and obstacles.

Finally, there is the problem of removing artifacts introduced during motion processing. As we have previously mentioned, the raw motion data itself is rarely used. Rather, it is first fit onto a skeleton and then edited to satisfy the particular demands of the animation. This process often introduce artifacts into the motion, which should be eliminated in order for the motion to appear realistic and to fit the animator's specifications. There are many methods to post process the motion data in order to guarantee that it preserves a certain number of constraints, most of which solve an inverse kinematics problems.

5.1.2 Brainstorm of Ideas

An important aspect of MoCap is that it requires a very expensive set up: the studio must be have several calibrated cameras and actors must where a special outfit with carefully positioned retro-reflexive markers. To make this technology useful in everyday applications (for example to control players in computer games) it is important to construct a low cost system which only requires a small number of cameras and markers.

This is done in [10]. Their approach is to make use of a large database of pre-recorded human motion and use it to complete the information given by a small set of markers (6-9 markers captured by 2 cameras). The system uses a small number control points (given by the few markers) and the motion database to construct a series of local models which are used to reconstruct the character's pose.

A different direction is to study the combination of locomotion and action. It is usually possible split and combine motion of the upper and lower part of the human body. For example, a character can carry a box or wave independent of whether it is walking, running or standing still. Nevertheless motions of the upper and lower body can be highly correlated and therefore, splicing techniques should consider not only physical needs such as balance preservation but also properties associated with stylistic form of human motion. An interesting work that addresses this problem is [11].

Finally, it would be interesting to consider the rhythmic patterns of motions. An intriguing work in this area is [12], which presents a novel scheme for synthesizing a new motion from unlabeled exampled motions while preserving their rhythmic pattern.

Bibliography

[1] Jue Wang, Steven M. Drucker, Maneesh Agrawala, and Michael F. Cohen. The cartoon animation filter. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1169–1173, New York, NY, USA, 2006. ACM.

[2] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002. ACM.

[3] Andrew Witkin and Zoran Popovic. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108, New York, NY, USA, 1995. ACM.

[4] Armin Bruderlin and Lance Williams. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104, New York, NY, USA, 1995. ACM.

[5] John Lasseter. Principles of traditional animation applied to 3d computer animation. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 35–44, New York, NY, USA, 1987. ACM.

[6] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 465–472, New York, NY, USA, 2002. ACM.

[7] Matthew Brand and Aaron Hertzmann. Style machines. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[8] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.*, 23(3):559–568, 2004.

[9] Michael Gleicher. Retargetting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, New York, NY, USA, 1998. ACM.

[10] Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696, 2005.

[11] Rachel Heck, Lucas Kovar, and Michael Gleicher. Splicing upper-body actions with locomotion.

[12] Tae-hoon Kim, Sang Il Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 392–401, New York, NY, USA, 2003. ACM.