NOTE TO USERS

LEARNING HAND-MOTION TO MUSIC BY EXAMPLE

By

Yasushi Akiyama

A Thesis Submitted to

Saint Mary's University, Halifax, Nova Scotia

in Partial Fulfillment of the Requirements for

the Degree of Master of Science

SEPTEMBER 2005, Halifax, Nova Scotia

Approved: Dr Sageev Oore, Senior Supervisor

Approved: Dr Karan Singh, External Examiner

Approved: Dr Joseph MacInnes, Supervisory Committee

Approved: Dr Norma Linney, Supervisory Committee

Approved: Dr Dirk Arnold, Supervisory Committee

Date: SEPTEMBER 2005

Your file Votre référence
ISBN: 0-494-09945-3
Our file Notre référence
ISBN: 0-494-09945-3

Abstract

"Learning Hand-motion to Music By Example"
By Yasushi Akiyama

The process of creating character animation requires hours of work and years of experience by skilled animators. Assisting this process by providing tools to automate some of the procedures during the animation creation is, therefore, of interest to many computer graphics researchers.

This thesis presents a system that generates arm dancing motion to new music tracks, based on sample motion captured data of dancing to other pieces of music. Rather than a common approach of creating character animation, which is to synthesize motions from an existing motion database, our system is novel in that it analyzes both the supplied motion and music data for certain characteristics such as distance-from-body, amplitude, and centres-of-motion (for motion data), and melodic contour, loudness, and note-density (for music data). It then learns relationships between the music and motion. When new music is provided, the characteristics are analyzed as before, and used to predict characteristics of the motion. A generative process then creates motion according to these constraints.

The system is evaluated with the music data that are used to trained the model, as well as new musical data including existing music tracks that are not specifically created for the purpose of animation creation.

September, 2005

To Sharon

Acknowledgements

First of all, I would like to thank my supervisor, Sageev Oore. Without his knowledge and experience in both computer science and music, this thesis could not have been completed. His ideas and guidance through the last two years were essential to this research, and his exceptional ability to overcome chaotic situations at the very last minute has helped me get through many critical circumstances.

I would also like to thank everyone who helped me to complete this work; the members of the examining committee, Joe MacInnes, Norma Linney, Dirk Arnold, and the external examiner, Karan Singh, the department's technician, Owen Smith, the department secretary, Rose Daurie, for her support and witty jokes, and my family for dragging me back and forth between sanity and insanity. Balance is the key here, and Sharon, Tanaka, and Kagan know it well. Finally, I would like to thank my parents, Masami and Sumie for their support and love that never fails to reach me even from overseas.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

There are limitless possibilities for creating motion to music. Certain characters may have particular styles of dancing, based both on their ways of moving, and also on how they perceive the music, or based on the directions of a choreographer. There are various examples of dance motions in which many salient movements are created in such a way that they correspond to some musical characteristics such as pitch, loudness, and note density. Although there are no constant relationships between motions and musical ideas, there may likely be a particular connection or association in certain contexts. This thesis presents an initial approach towards exploring this issue by analyzing the characteristics of music and motion data, and by then providing a prototype tool for animators to specify some of these relationships implicitly by example.

Various work [6, 7, 9, 10, 13, 17, 19, 24, 31] has focused on the problem of adapting existing motions to music. In many of these systems, the primary mappings between the features of music and motion are specified explicitly by the user. In others,

motions are synthesized from an existing database. This research proposes a system for learning the characteristics of desired motions associated with musical phrases by user-provided examples, in order to produce an animated figure that responds not only to the musical phrases that are used to train it, but will also be able to handle new musical ideas. Such a tool would be intended for use by animators to choreograph characteristics of dance for a musical performance, so the final motions can be created automatically. For example, ultimately the dance style in a certain nightclub or a set of backup singers/dancers for a musical act, or recurring dancers in a video game, could be done in this way.

Since the motion-music relationships are inferred from the data, it follows that different data sets will lead to different styles of interpreting the music. For example, if all of the supplied training data were to depend exclusively on the pitch, and not be affected in any way by changes in loudness (e.g. the 'dynamics of the phrasing', in musical terms), then this should be visible in the resulting animation. The system is not intended to learn a single definitive relationship between motion and music based on a large comprehensive motion database, but rather to allow certain relationships– as needed for a particular set of characters or situations– to be shown by example.

1.2 Related Work

The issue of synchronizing animation to music has been addressed by a number of researchers. Kim et al. [13] created a system using motion graphs [14] and beat analysis to synthesize motion from existing sample motions synchronized to background music. Alankus et al. [1] use a very similar approach, although they use a genetic algorithm

to search the solution space, and use a different algorithm for motion analysis. The system by Cardle et al. [6] synchronizes motion curves by locally modifying the initial motions using perceptual cues obtained from the music, both in the format of MIDI and analog signal. The mappings between music and motion features are determined by the user. Lee and Lee [17] extended Cardle's idea by including the modification of the music in addition to the modification of the motion curve. Their approach is to synthesize background music and motion by locally changing the timing of the music and using time-warping of the original motion. Lytle's [19] system creates animation of musical instruments from orchestrated MIDI music. This approach also requires input from the animator to ensure that the salient features of music will correspond to the output animation. Another system that creates animation from MIDI music is presented by Goto and Muraoka [9]. Their approach is to have multiple musicians controlling a single animated character. Each musician is assigned to control specific features of the animation. Greuel et al. [10] also used explicit mappings between music and animation. The analog sound signal was used in the system by Penasse and Nakamura [24]. Their approach is to adjust key frames so that they fall on the beats of music, although how they get their keyframes in the first place is not clear. ElKoura and Singh [7] use input music data in a slightly different form, an augmented tablature notation for guitar. This format specifies target locations in space (which fret on which string) and time (when to play note(s)). Their system *Handrix* tackles the complexity of multiple concurrent reaching tasks by first mapping arbitrarily specified hand configurations to realistic hand configuration that are obtained by a k-nearest neighbours search in example space, and then by performing a procedural

algorithm to minimize a cost function in order to control fretting hand motions. Wang et al [31] predict the joint angle trajectories to create conducting motions by using kernel-based Hidden Markov Models. This model produces animation to new music, but only by learning the music with the same time signature, or a time signature that has the same musical accents. Also, it relies on having beat information provided in the MIDI file. These factors may limit its flexibility of creating motions responding to music that is played more freely. Using explicit mappings of features between music and animation seems like a standard approach.

The study by Lipscomb and Kim [18] suggests that particular mappings between music and motion characteristics make sense while the others can have more than one sensible combination. As their experiment used only very simple sequences of notes such as triads and systematic sequential patterns, it is unclear how their results extend to more complicated musical structures. Furthermore, different styles of music may lead to different kinds of mappings, and people whose musical experience is derived from various cultural backgrounds are quite likely to have different preferences of the mappings.

Chapter 2

System Overview and Outline of the Thesis

The goal of our system is to accept examples of synchronized motion and music, in order to learn a relationship between the two, and subsequently use this information to generate animation for new music input. For demonstrative purposes, the present system focuses on arm motions recorded to melodic lines in MIDI format.

Our system operates in two phases, training and generative phases. During the training phase, we train the model so that it learns relationships between certain characteristics in example motions and their corresponding music. In the generative phase, we construct motions based on the output data obtained from the model, and then use an inverse kinematics algorithm (Appendix A.2) to produce animation. The model can produce motions to the music data used during the training phase as well as new music data.

We now give an outline of this thesis, along with an overview of the two phases.

2.1 Training Phase

A set of examples is provided consisting of hand-motion recorded in synchrony with music in MIDI format. The data collection procedures are described in Section 3.1. Based on our motion model, the motion data is analyzed for certain characteristics: distance-from-body, amplitude, and centres-of-motion (described in Section 3.2). Likewise, the music data is analyzed for characteristics such as melodic contour, loudness, and note density (Section 3.3). A relationship between the input and output characteristics is learned so as to be able to generate a distribution over likely output motion characteristics given music input characteristics (Section 4.1). Figure 2.1 shows the conceptual model of the training phase.

2.2 Generative Phase

New music input is provided. The system analyzes it, and uses the learned model to stochastically generate a set of motion characteristics. A generative process is applied to create motions for the final animation that satisfies the motion characteristics. This phase starts with predicting both temporal and spatial positions of *motion peaks*. The animation frames between these peaks are interpolated using quartic polynomial interpolation (Section 4.2), and then the final configuration for each frame is computed by inverse kinematics algorithm (Section 4.3). The model of the generative phase is shown in Figure 2.2.

Finally, in Chapter 5, we evaluate the system. This is done first by using the music data that are used to train the system. In this way, we have example motions

that can be compared with the resulting motions created by the system. It is also evaluated with new musical data that are not seen during the training process.

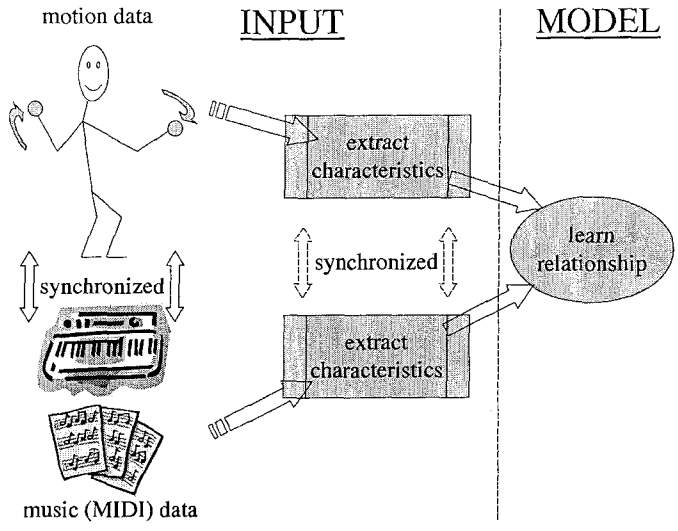


Figure 2.1: Training Phase: Synchronized examples of motion and music are provided to the system.

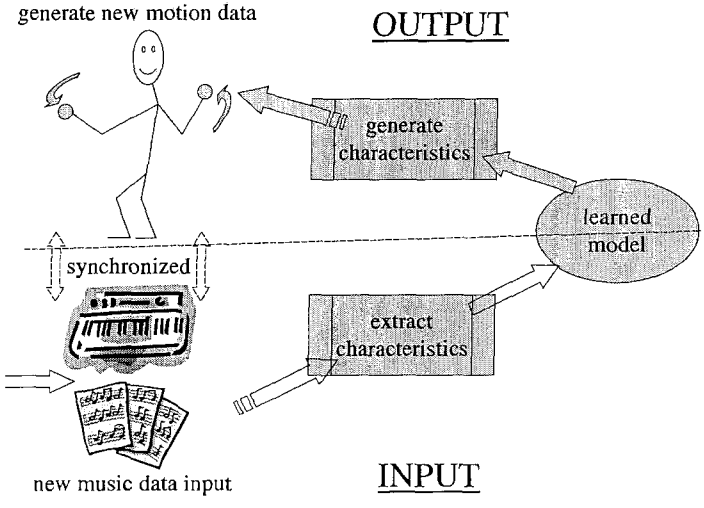


Figure 2.2: Generative Phase: New music is now provided, and using the learned model, motion is now the output rather than input (as indicated by the reversed arrows).

Chapter 3

Data Collection and Analysis

3.1 Data Collection

3.1.1 Music Data

Music is input in the Type 1 MIDI file format (see Appendix A.1), which allows a file to have multiple tracks. This is convenient as it lets us focus directly on processing individual lines (e.g. melody, bass) and avoid the task of separating a given score into parts ([27, 28, 30]).

A variety of music data are collected to evaluate the system's flexibility. These include, but are not limited to, one-line melodies, melodies with occasional harmonies, music with musical embellishments such as trills and tremolos and/or a few small mistakes, piano music (both hands), music arranged for a small jazz combo setting (e.g. a few horns and a rhythm section) each part of which to be assigned an individual animated character, and so forth. Any of these can be played in a steady tempo (slow, medium, or fast), or free of tempo (*Rubato*).

In addition to the newly collected music data for the purpose of the creation of dance movements, we also use existing songs created not specifically for producing animations.

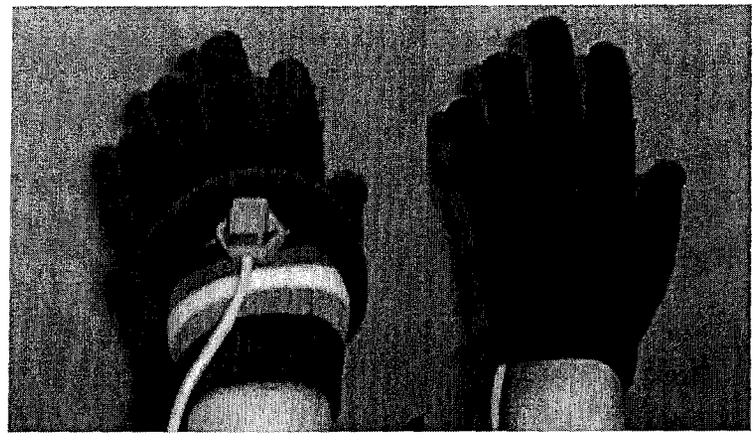


Figure 3.1: Polhemus 3D motion capture sensors are attached to the gloves. (Only the left hand is shown)

3.1.2 Motion Capture

A 3D motion capture device (Polhemus Isotrak) is used to collect the sample motion data (Figures 3.1 and 3.2). The dancer familiarizes herself with the music prior to the motion recording session. It is important for dancers to know the music well because in real life, choreographers generally listen to and analyze the music before they come up with movements.

Two sensors are attached to the gloves that the dancers wear. We acquire three degrees of freedom (DOF) from each sensor, measuring hand position. As the dancer is aware that the motion will be used for the arm motion of a character who is standing in one spot during recording, the dancer also remains in one spot to ensure that the

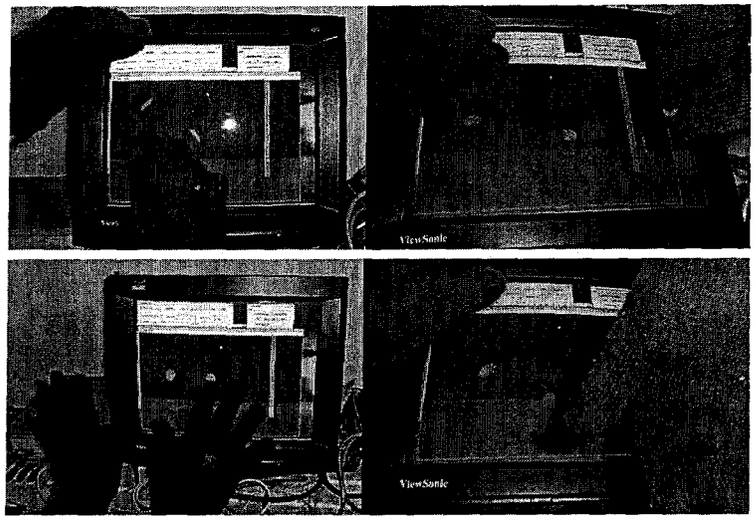


Figure 3.2: Motion Capturing

motion examples are representative. A number of different motions are recorded to the same music to have a sufficient sample size for the statistical analysis described in Section 3.2.

3.2 Dance Motion Analysis

Our motion model is developed based on a few key properties observed in the captured sequences. In the following discussion, $B_1, B_2, \ldots, B_N$ represent N different recorded motions, each of which was danced in sync to the same music track M.

3.2.1 Spatial Characteristics

An important visual feature of the arm motion is the distance of the hands from the body, corresponding to *extent* as described by Laban [15] and Neff and Fiume [21]. We

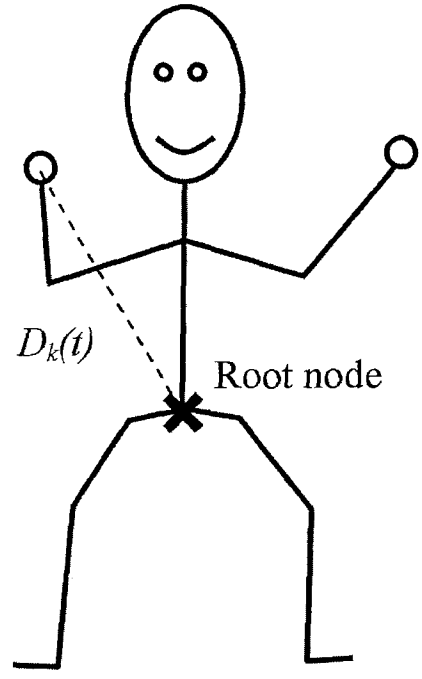


Figure 3.3: Root position (marked with an X) and $D_k(t)$ (dotted line)

first compute $D_k(t)$, the Euclidian distance of the hands from the root node at time t, for motion B_k (shown in Figure 3.3). We then estimate the *global motion centre* by computing the mean position of the entire hand motion. Finally, we compute $dist_k(t)$, the Euclidian distance of the hands from the global motion centre. This is shown in Figure 3.4. Figure 3.5 shows $dist(t)$ of four sample captured dance sequences to the same piece of music M.

The motions tend to be comprised of oscillating segments, corresponding to back-and-forth (or side-to-side, up-and-down etc) movement of the hands. We model such oscillations by estimating, for each time frame, an approximate centre about which the current oscillation is taking place, as well as its approximate amplitude. To do this we define a window $V(t_i)$ around time t_i, corresponding to a set of $S+1$ frames of recorded dance motion from $(t_i - S/2)$ to $(t_i + S/2)$. This is shown in Figure 3.6. The

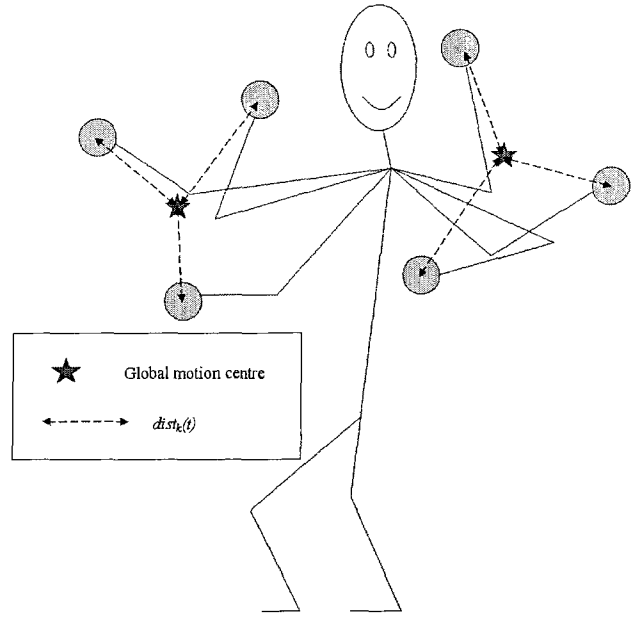


Figure 3.4: global motion centre and $dist_k(t)$

size of $V(t_i)$ should be adjusted so it can capture the characteristics of the motion; when it is too big, only the condensed information of multiple oscillating segments is obtained, while if it is too small, it is unable to capture the ups and downs of the curve. The size of V is determined so that it equals the average duration of the oscillations.

For each window $V(t_i)$ we compute $m_k(t_i)$, the mean hand distance:

$$m_k(t_i) = \frac{1}{S+1} \sum_{t_j=t_i-\frac{S}{2}}^{t_i+\frac{S}{2}} (dist_k(t_j))$$

Then, using $m_k(t_i)$, we compute the variance $v_k(t_i)$ for window $V(t_i)$:

$$v_k(t_i) = \frac{1}{S+1} \sum_{t_j=t_i-\frac{S}{2}}^{t_i+\frac{S}{2}} (dist_k(t_j) - m_k(t_i))^2$$

The square-root of this value ($\sqrt{v_k(t_i)}$) relates to the amplitude of the oscillations;

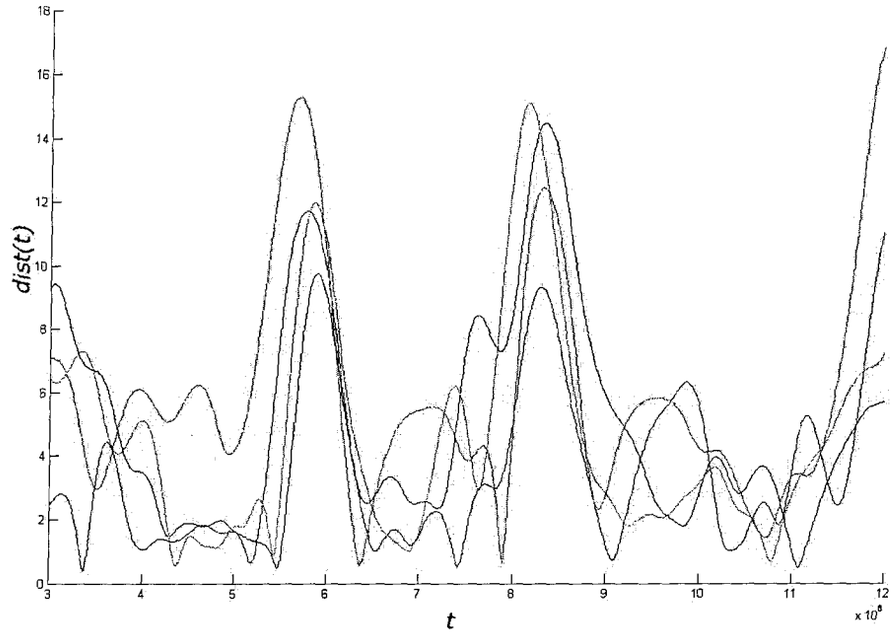


Figure 3.5: This illustrates four different sets of $dist(t)$ for a single music track. Each colour corresponds to one recorded motion. Note that while each curve is different, they share certain characteristics, and those characteristics are what we would like to learn. For example, the two largest peaks happen around the same time in all the motion samples, and the ranges of the oscillation amplitudes are very similar within a certain period of time. Also, they share similar oscillatory nature.

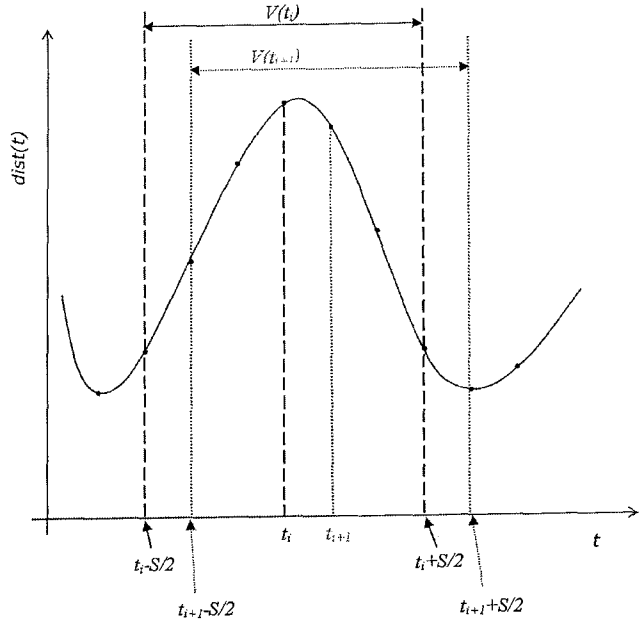


Figure 3.6: Moving window $V(t_i)$

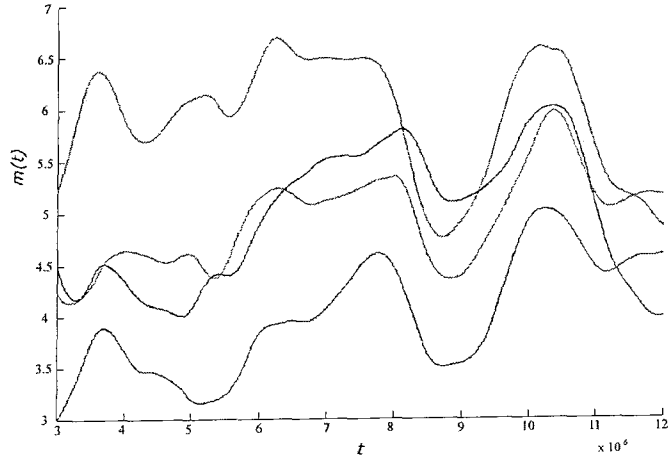


Figure 3.7: The mean distance $m_k(t)$. The same colors correspond to the same motions in Figures 3.5 and 3.8.

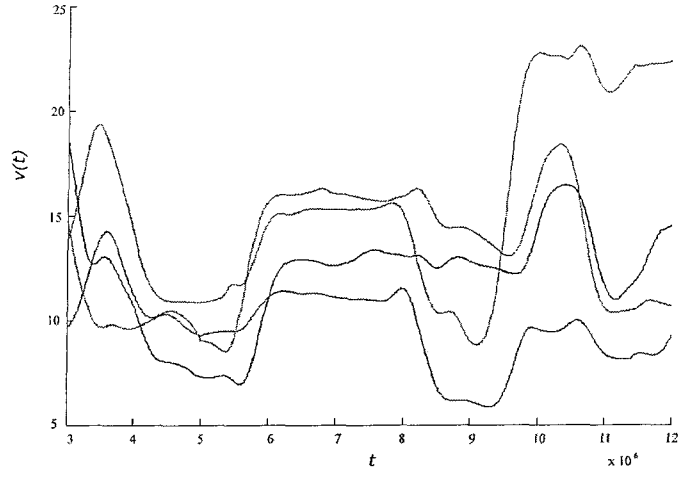


Figure 3.8: The variance $v_k(t)$

the larger $\sqrt{v_k(t_i)}$ is, the larger movement of the hand is around t_i. Figures 3.7 and 3.8 show $m_k(t)$ and $v_k(t)$, respectively, for clips from a set of motions all corresponding to the music track M. Looking at these graphs, we see that, for a given piece of music, there are sections where there is considerable variance in the m_k, and other sections wherein the m_k are relatively tight. Similarly, looking at the oscillation amplitudes over the same motion set, more coherence is evident in some areas than in others. This suggests that, if we are to generate $m(t)$ and $v(t)$ for some new piece of music, and if we want to be able to create multiple animations sharing "characteristics", we would like to generate a *distribution* which we will specify by a mean $\mu_m(t)$ and variance $\sigma_m(t)$, from which we will later sample values (Section 4.2) for $m(t)$. A small value of $\sigma_m(t)$ will lead to values predicted for $m(t)$ that will be close to $\mu_m(t)$. Similarly we will generate a distribution specified by a mean $\mu_v(t)$ and variance $\sigma_v(t)$

of $v(t)$. In order to learn to generate these values for a new piece of music, we first estimate them for the given data:

$$\mu_m(t_i) = \frac{1}{N}\sum_{k=1}^{N}(m_k(t_i))$$

$$\sigma_m(t_i) = \frac{1}{N-1}\sum_{k=1}^{N}(m_k(t_i) - \mu_m(t_i))^2$$

and

$$\mu_v(t_i) = \frac{1}{N}\sum_{k=1}^{N}(v_k(t_i))$$

$$\sigma_v(t_i) = \frac{1}{N-1}\sum_{k=1}^{N}(v_k(t_i) - \mu_v(t_i))^2$$

where we have used an unbiased estimator for the variance.

Both the amplitude of the oscillations, and their centre, change throughout the sequence. In Section 3.3 we will describe some of the music characteristics that may drive $m(t)$ and $v(t)$, but first we consider some temporal characteristics of the motion.

3.2.2 Temporal Characteristics

While the frequency is of course not constant, there is likely a relationship between the motion and the rhythm of the music, and once again, we will be trying to capture certain characteristics of this relationship. One type of visually salient event is the *peak* of an oscillation— a hand stopping and going in another direction. This event corresponds to zero-crossings of derivative of the distance function $dist(t)$, and we would like to model these *peak points* both in time and in space. The amplitude and motion centre predictors (μ_v, σ_v, μ_m, and σ_m) will give us, based on the music characteristics, a distribution over where the peak point should be in space; another

predictor will be used to stochastically choose the frame at which that peak point should occur so that it is in synchrony with musical characteristics. To provide training data for the learning algorithm, we create, for each motion B_k, a binary variable

$$peak_k(t) = \begin{cases} 1 & \text{if a local maximum occurs in } t \pm \Delta t, \\ 0 & \text{otherwise} \end{cases}$$

where Δt is a small constant (≈ 0.1 second) to allow for slight temporal discrepancies between the data and the music. For each t_i, we combine $peak_k(t_i)$ for each B_k, where $k = 1, \ldots, N$, to obtain an estimate, for a given piece of music, of the independent probability of a peak event occurring at each time slice. That is:

$$peak(t) = \frac{1}{N} \sum_{k=1}^{N} peak_k(t)$$

We repeat this for each of the given music tracks and corresponding sets of animations. This gives us a set of music tracks, each with a corresponding function $peak(t)$ marking possible moments at which such events may occur.

3.2.3 Testing with Example Motions

We now show how well these features of a motion can be used to recreate a new motion, which captures similar *characteristics* of the example motions. First, we extract the parameters $peak(t)$, $\mu_v(t)$, $\sigma_v(t)$, $\mu_m(t)$, and $\sigma_m(t)$ as described above, and then, using these parameters, we will generate a *motion curve* – a path of the hand motion in 3D space. The process of the motion curve creation using these parameters is described in Section 4.2.

Figure 3.9 shows $dist(t)$ function of the motion curve generated by using these

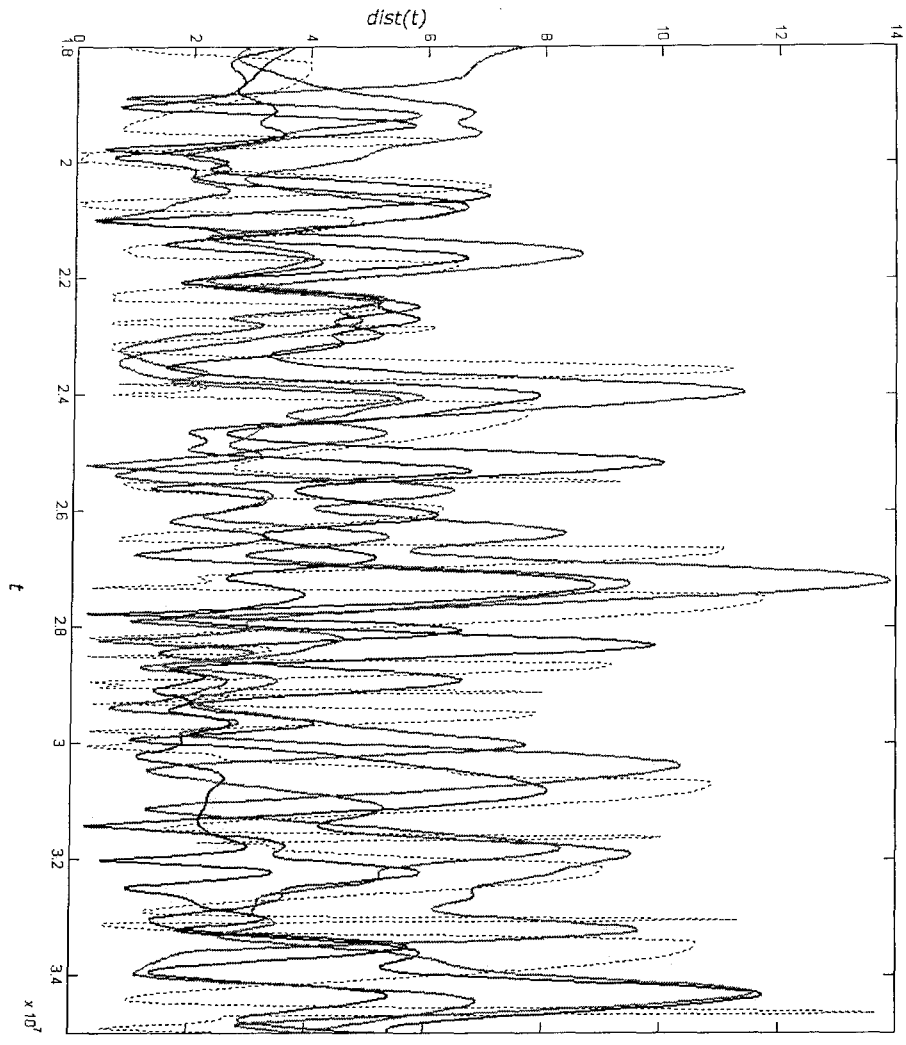


Figure 3.9: $dist(t)$ of the motion curve generated using parameters extracted from examples. The generated motion curve is indicated by the dotted line. Each of the other solid lines corresponds to each example of motions to the same music track. Note that although the generated curve does not follow the exact path of any of the example motions, it roughly emulates the amplitude of the peaks, which approximately happen around the time at which the peaks in the example motions happen.

parameters (dotted line), and the example motions from which we have computed the statistical parameter values. Although the generated curve does not follow the exact path of any of the example motions, it roughly emulates the amplitude of the peaks, which approximately happen around the time at which the peaks in the example motions happen.

As this only suggests the relevance of using those parameters, we now need to show

that by interpolating these peaks, it is possible to create a motion curve very similar to the original motion curve from which it is derived. The interpolation process is also described in Section 4.2.

Figure 3.10 shows the original motion curve of the x-axis component (blue solid line), the original peaks (red x's), and the regenerated motion curve (dotted line) by interpolating those peaks. The interpolated curve follows very closely the original curve. Therefore, if the parameters $peak(t)$, $\mu_v(t)$, $\sigma_v(t)$, $\mu_m(t)$, and $\sigma_m(t)$ are provided, we are able to re-generate motions very close to the original example motion. Figure 3.11 shows the captured images of the original motion (on the left hand side) and the motion created by interpolation between the peaks in the original motion (on the right hand side). It is seen that the regenerated motion is in fact very similar to the original one.

3.3 Music Analysis

Before we can apply a machine learning tool to generate the motion characteristics described above, we need to consider the 'input' variables to the system. That is, we now extract certain characteristics of a given musical track that may influence the resulting animation. The three key concepts in which we are interested are (1) melodic contour, (2) loudness and (3) note density. To realize this, we need to compute various quantities as described below.

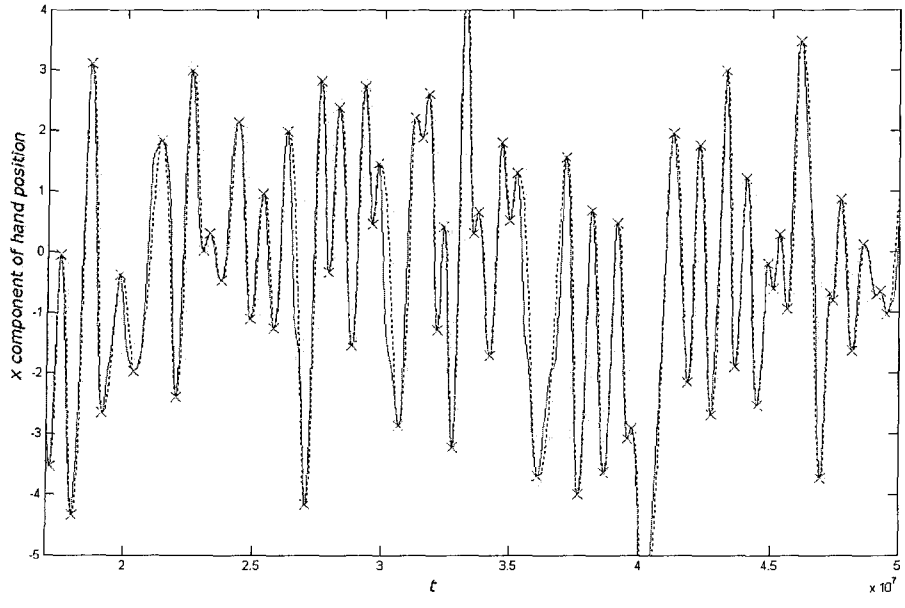


Figure 3.10: The original motion curve on x-axis (blue solid line), the original peaks (red x's), and the motion curve obtained by interpolation those peaks (dotted line).

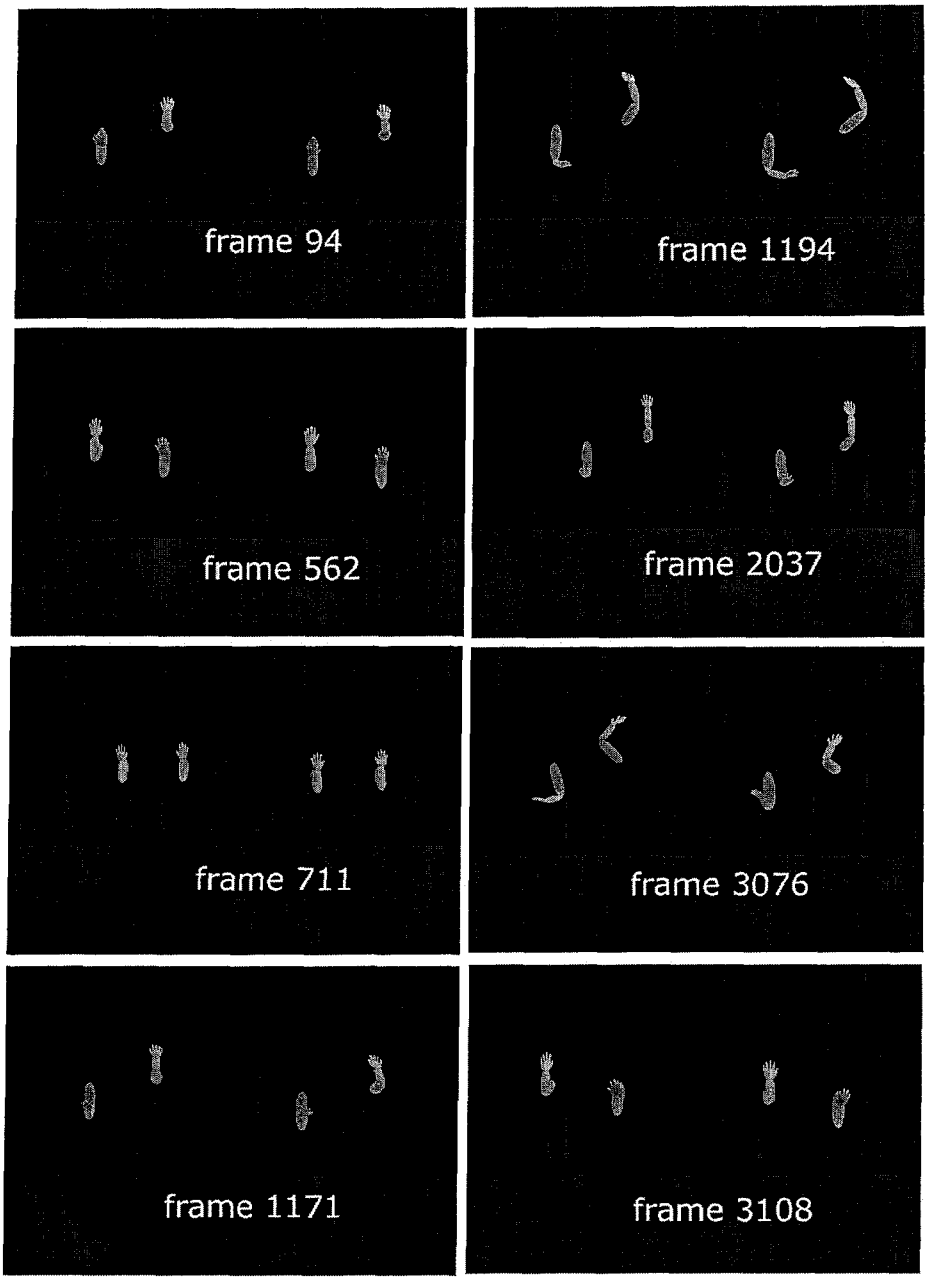


Figure 3.11: Original motions and motions created by interpolated between the original peaks. Within each frame, the pair of hands on the left is the original posture, and the pair on the right the one created by interpolation. It is clear that the motion created by interpolation is in fact very close to the original motion.

3.3.1 Dealing with Multiple Scales

Suppose that the current note is somewhat higher (or louder) than the previous note. What is the significance of this, with the view of choreographing a dance to the music? This would depend not only on the previous note, but also on where this note fits within the larger context of the pitch (or loudness) contour.

Figure 3.12 shows a music excerpt from J. S. Bach's *Choral 250.* Looking at the top part (melody) of this music, there are three occurrences of a note $C^{\#}$. The note first appears at the end of the first bar. This note acts as a *passing tone* between the preceding note B and the following note D, making this $C^{\#}$ as the *leading tone* resolving to the *tonic* note (D) of the key of this music[1]. This is very different from the one on the second beat in the next bar, as it serves as a *neigbouring tone* going back to the same note as its previous note. The chord for this second $C^{\#}$ is not the dominant chord (V) (as in the first case) but III serving as upper 5th of VI (which eventually turns into the diminished chord (VII) of the dominant of the original key by raising the $F^{\#}$ to $G^{\#}$), thus the role of the $C^{\#}$ is different from the previous case. Even though it is rather simple to realize these roles of the note $C^{\#}$ in phrases, it is impossible to perform this analysis by looking at a single note. Furthermore, the detailed analysis of music contour is more likely to have a meaning when the events happen right before the current event. The note events happening in the near past generally have more influence on the current event than the events that happen in the further back in time. This seems a natural phenomenon in most of

[1]For information about this kind of musical analysis, see any standard harmony texts such as [2]

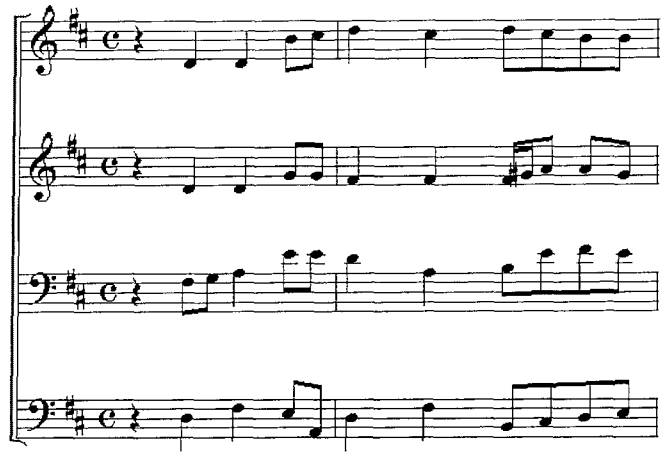


Figure 3.12: Bach, Choral 250

the regression problems. We use this heuristic information to simplify very complex characteristics of music data; we consider musical progression at several time-scales. We approach this by a rough approximation in which we compute characteristics for three distinct time-windows, $W_1 = [w_0, w_1]$, $W_2 = [w_1, w_2]$, $W_3 = [w_2, w_3]$, and refer to the collection of these subwindows as $W = [w_0, w_3]$, with $w_j < w_{j+1}$ and w_3 being the current frame. This is illustrated in Figure 3.13.

Note that the windows used here for the music context are different from the windows used for the motion data analysis, in that the music windows are to capture the music contour that leads up to the current note event, while the motion windows are used to capture the behaviour *around* the current event. This is why the music window ends on the current event, but the motion window is centred on the current event.

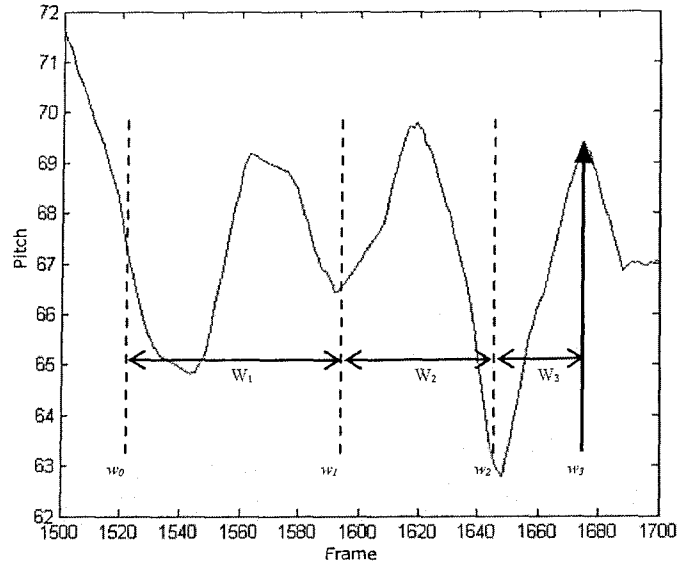


Figure 3.13: The windows W_1, W_2 and W_3 for multiple scales.

3.3.2 Melodic Contour Information

Melodic contour refers to the rise and fall of the pitches (high-low quality of a musical sound [3]). If a passage is played slowly and staccato, then most of the time there is no "active" note, yet a contour is still implied, and animated motion may correspond to this contour. This can be solved by simply keeping track of the previous note. However, suppose a note is embellished with a trill, tremolo or turn (e.g. notes alternating quickly back and forth). The contour could be considered static in this case as opposed to varying along with the trill. Furthermore, in a fast passage, notes may locally be going up and down, but the overall shape might be a gradually rising progression, so that the musical contour is rising. This motivates a weighted moving-average filter, which also allows the contour value to be expressed as a function of frame number. That is, a contour value is expressed even when there is no note being

played. This also reduces the effect of a small mistake in which an additional note is played around the desired note. The resolution of the interpolation is determined according to the sampling rate of the motion capture data, so that each motion frame has a corresponding note event. We can then compute the following characteristics:

1. *Pitch samples.* Once the pitches have been processed as described above, processed pitch values $p(t_i)$ can be sampled as needed from W. This input parameter describes the general contour of the music in window W.

2. *Mean and variance of pitches.* Statistics such as mean μ_p and variance σ_p of the pitch contour $p(t)$ can be calculated for each of the windows W_j. μ_p can be considered as a local phrase centre (i.e., around which register the notes occur) while σ_p gives us the idea of how quickly the notes go up (or down) in the given window.

3. *Pitch-peaks.* Just as peaks are salient features of motion, they are salient features of melodic contour, and therefore a pitch-peak variable is used to flag whether the current event is (close to) a local maximum (1), or a local minimum (-1), and 0 is assigned otherwise.

4. *Duration to next/previous closest peaks.* A peak may be perceived to be more significant if it is more isolated from any other nearby peaks. By including these parameters, one can perceive whether or not the peak is isolated from others or just one of many peaks happening in a short period of time.

5. *Phrase endings and duration of phrase.* Although algorithms to detect phrase endings are widely available ([25, 29]), a very simple procedure of finding two consecutive notes with the inter-onset time larger than a threshold was used in this study. This method is inspired by Pachet's *Continuator* [22]. The threshold is typically based on the average inter-onset time of all the notes. Phrase ending is important as it indicates a *break* in music, and it oftentimes corresponds to a break in dance motion.

3.3.3 Loudness Characteristics

Although loudness information is not present in MIDI data, it can be expressed implicitly as *velocity*, which refers to the speed of a key being pressed when a note is played (see Appendix A.1); The faster the key is pressed, the louder the note sounds.

Loudness information is processed differently from the pitch information. Firstly, we assume that when a note is played, it decays at a fixed rate while the key is pressed, and then as soon as the key is released, the note stops. This phenomenon of decaying sound is observed in many of acoustic instruments such as piano and guitar. Although the MIDI information may request a non-decaying instrument such as organ and wind instruments, we only concentrate on simulating those instruments with decaying sound at this time. (For non-decaying instruments, we can simply keep loudness to be constant while the key is being pressed. Our tests have shown that this works well with our system.)

For each time frame, we compute a sum of the loudness of all the notes that are decaying. The loudness $l_j(t)$ of a jth note at time t_i is given by:

$$l_j(t_i) = l_j(t_{j0})\kappa^{(t_i - t_{j0})}$$

where $l_j(t_{j0})$ and t_{j0} are the initial loudness and NOTE ON time of the jth note, respectively, and κ the *decay* constant ($\kappa < 1$). Note that if the key is held long enough, $l_j(t_i)$ ultimately nears 0 even before the key is released. The sum of the loudness $L(t_i)$ at time t_i is computed as:

$$L(t_i) = \sum_{Note_j \in \Phi_{t_i}} (l_j(t_i))$$

where Φ_{t_i} denotes all the notes that have NOTE ON time prior to time t_i but have not been turned off yet. This *cumulative* loudness is relevant because it captures articulation of notes as well as events that are important for realization of the loudness. For example, in the events such as trills and tremolos, the overall loudness is kept near constant as if a single note ringers without decaying even though there are many notes played one after another. In another case when a group of notes are played simultaneously (e.g. chords), the music is usually louder than one-line melodies. Figures 3.14 and 3.15 show the result of the cumulative loudness and its corresponding analogue audio loudness.

Once we have obtained the cumulative loudness of the music, we then extract the information in a manner similar to the process of pitch information. (i.e. loudness samples, mean and variance, peaks, and duration to neighbouring peaks.)

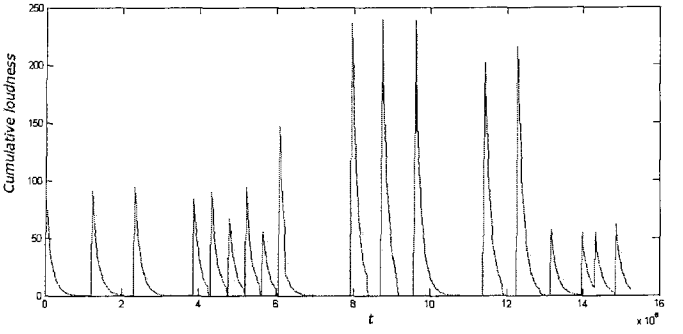


Figure 3.14: Cumulative loudness processed from MIDI data

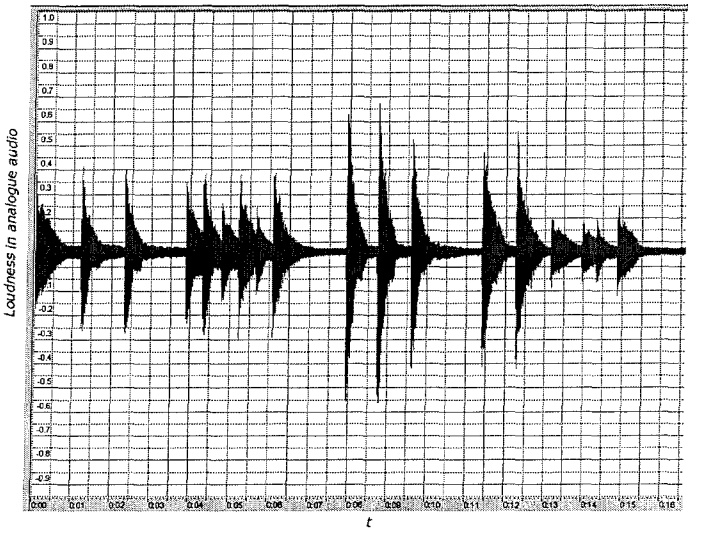


Figure 3.15: Loudness in analogue audio format

3.3.4 Density

At each time t_i, the *density* $\rho(t_i)$ of notes is computed for each of the three windows leading up to that time. Let the number of notes in a window $W_k(t_i)$ and the time duration of $W_k(t_i)$ be, $N_k(t_i)$ and $\Delta T_k(t_i)$, respectively, $\rho_k(t_i)$ is computed as:

$$\rho_k(t_i) = \frac{N_k(t_i)}{\Delta T_k(t_i)}$$

This information describes how many notes are played in a given window, hence, it gives us a general idea of how fast the music is played. Because of the filtering of the pitch information and the cumulation of the loudness, the information of the note density is somewhat lost without explicitly providing it.

Alternately, the average inter-onset time in a given window can be used. It also describes how fast the music is played, and both cases show very similar results in this study.

Chapter 4

Learning and Generation of Output Animation

4.1 Learning

Once the various characteristics have been selected and extracted, standard feed-forward neural network models are used to learn relationships based on the provided example data.

Our system predicts two types of characteristics of the output motion: temporal and spatial characteristics. The input data is based on a series of music tracks, each with a set of corresponding recorded motions. The input data is processed as described in Section 3.3, so that each time slice of the music contributes one training example. For each frame at time t_i, let $u(t_i)$ represent the set of musical characteristics computed as described in Section 3.3. One model is trained to predict the probabilities $peak(t)$ given input $u(t_i)$ (Section 3.2). The other model learns to estimate the spatial characteristics ($\mu_v(t_i)$, $\sigma_v(t_i)$, $\mu_m(t_i)$, and $\sigma_m(t_i)$) of the distribution (also described in Section 3.2). These output values are then used to create motion curves. The details of the motion curve creation are discussed in Section 4.2.

#	Parameter
1 - 10	distances in time to the last 10 peaks in pitch
11	distance in time to the next peak in pitch
12 - 21	distances in time to the last 10 peaks in loudness
22	distance in time to the next peak in loudness

Table 4.1: Input parameters for the temporal predictor

4.1.1 Neural Network Structure

The structure of both the models is a feed-forward network with one hidden layer (6 hidden units). Tanh functions are used in the hidden layer, and logistic functions in the output layer. Although we have tried different numbers of hidden units and different types of output functions such as linear function, this model resulted in the best performance by far. ([5])

The temporal model has 22 input parameters (Table 4.1) and 1 output parameter (Table 4.3), which is the probability $peak(t)$ indicating the likelihood of the frame at time t being a peak in the motion output. The spatial model has 41 input parameters (Table 4.2) and 4 output parameters, $\mu_m(t)$, $\sigma_m(t)$, $\mu_v(t)$, and $\sigma_v(t)$ (Table 4.3), which are used to form distribution functions. How these output parameters are used to generate motion curves is discussed in Section 4.2.

4.1.2 Training of the Models

Each time frame contains the combination of music data and the corresponding motion data (a *posture*). Each of these combinations contributes one example of the training data, thus a single training set contains a number of examples which is the same as the number of frames in the music data. The frames at the beginning and

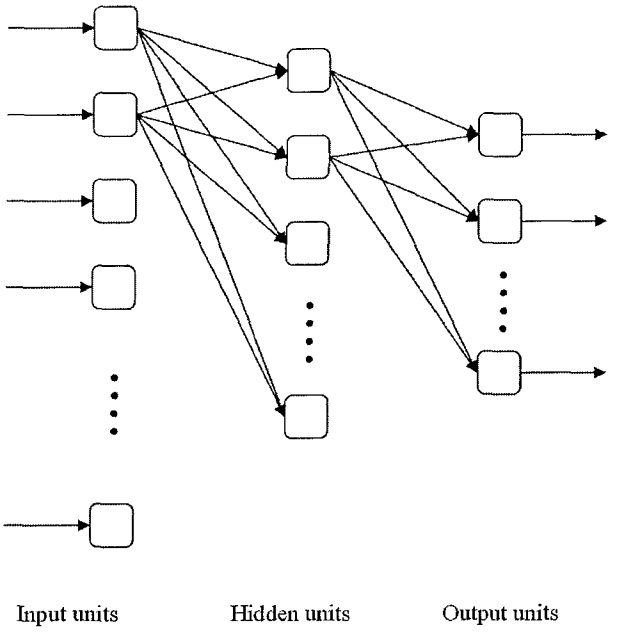


Figure 4.1: Neural Network Structure: feed-forward neural network with one hidden layer.

#	**Parameter**
1 - 10	Pitch values in window W
11 - 20	Velocity values in window W
21 - 23	Density of notes in windows W_1, W_2, *and* W_3
24 - 26	Mean of pitch in windows W_1, W_2, *and* W_3
27 - 29	Variance of pitch in windows W_1, W_2, *and* W_3
30 - 32	Mean of loudness in windows W_1, W_2, *and* W_3
33 - 35	Variance of loudness in windows W_1, W_2, *and* W_3
36	Flags to indicate peaks in pitch
37	Flags to indicate peaks in loudness
38	Time duration of peak in pitch to the next minimum
39	Time duration of peak in loudness to the next minimum
40	Flags to indicate phrase endings
41	Time duration of phrases

Table 4.2: Input parameters for the spatial predictor

#	Parameter	
1	$\mu_m(t)$	(from spatial model)
2	$\sigma_m(t)$	(from spatial model)
3	$\mu_v(t)$	(from spatial model)
4	$\sigma_v(t)$	(from spatial model)
5	$peak(t)$	(from temporal model)

Table 4.3: Output variables

end of each data set are excluded from the training data as they typically contain irrelevant data: due to the lack of a system for cuing exactly when to start moving, it is almost impossible for the dancer to synchronize the motion with music for the first few seconds. Also, for a similar reason, the ending frames usually do not reflect well what the dancer intends to do because the music sometimes stops unexpectedly despite the fact the dancer familiarizes herself prior to the recording sessions.

The models are trained using a scaled conjugate gradient optimization algorithm, which sometimes results in faster convergence than conventional conjugate gradient algorithms ([5]). The training process continues until 1) the number of iterations reaches the maximum value, or 2) the error function returns a value smaller than the threshold. The results of the training are discussed in Section 5.1

4.2 Generating Motion Curves

To generate a motion curve from the parameters obtained from the learning module, the following steps are performed.

1. Stochastically mark *candidate peaks* by using the $peak(t)$ estimator

2. These candidate peaks are further trimmed down based on human physical

ability and the constraints specified by the user

3. For each peak point occurring at time t_q, a local mean distance of the hand position $m(t_q)$ and the variance $v(t_q)$ are sampled from distributions specified by $\mu_m(t_q)$ and $\sigma_m(t_q)$, and by $\mu_v(t_q)$ and $\sigma_v(t_q)$, respectively

4. $dist(t_q)$ is given by: $dist(t_q) = m(t_q) + \sqrt{v(t_q)}$

5. θ_k is randomly chosen from several pre-defined angles. This is the rotation angle of an axis along which the hand motion occurs (to simplify the problem, the hand motion is currently restricted only on the x-y plane)

6. θ_q is then sampled from a distribution specified by a mean at θ_k and a variance σ_θ^2

7. Peak point $q(t_q) = \{x(t_q), y(t_q), z(t_q)\}$ is calculated as:

$$\begin{aligned} x(t_q) &= dist(t_q) \cdot \sin\theta(t_q) \\ y(t_q) &= dist(t_q) \cdot \cos\theta(t_q) \\ z(t_q) &= 0 \end{aligned}$$

8. After determining the locations of the peak points in space and time, another set of peaks are computed so as to create oscillating curves

9. A hand motion is computed by using a quartic polynomial interpolation. This interpolation is performed separately on each axis.

10. For each frame, the final configuration of the hand and arm motion is computed using an inverse kinematics algorithm. Some heuristic methods are used to further determine the orientations of the hand and arm.

Now we discuss each of these steps in details.

4.2.1 Generating and Trimming Candidate Peaks

The generative procedure begins by analyzing a new musical input track, and using the $peak(t)$ estimator (predicted by a sigmoidal unit, which is a value between 0 and 1) to stochastically select certain times as moments where motion peaks can occur. The system then goes through all the candidate peak points to make sure that the consecutive peaks are not too close in time. Whether or not they are too close is determined by two factors; one based on human physical ability and the other specified by the user. The maximum values of velocity and acceleration are obtained from the example motions. If the candidate motion peaks require movements with values higher than these maximum values, one of them is excluded from the list of peaks. The system also gives the user an option to control the number of possible peaks. The more peaks in a motion, the faster the hand has to move in general. Therefore, controlling the number of peaks indirectly corresponds to the control of the speed of the hand motion. The current system allows the user to select from four different settings; fast, medium, slow, and very slow. This is particularly useful when animators desire different dance movements to exactly the same music, depending on circumstances. Figures 4.2 and 4.3 show example motions generated according to the user's choice of the speed. Both are generated from the same musical data.

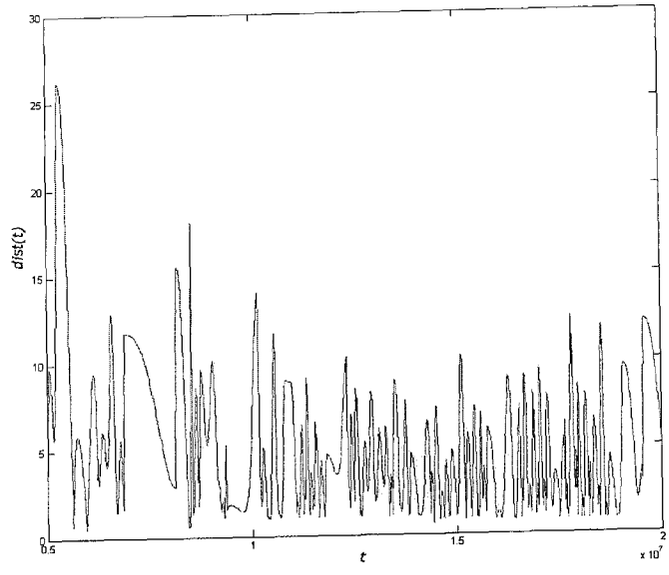


Figure 4.2: Motion with fast movement

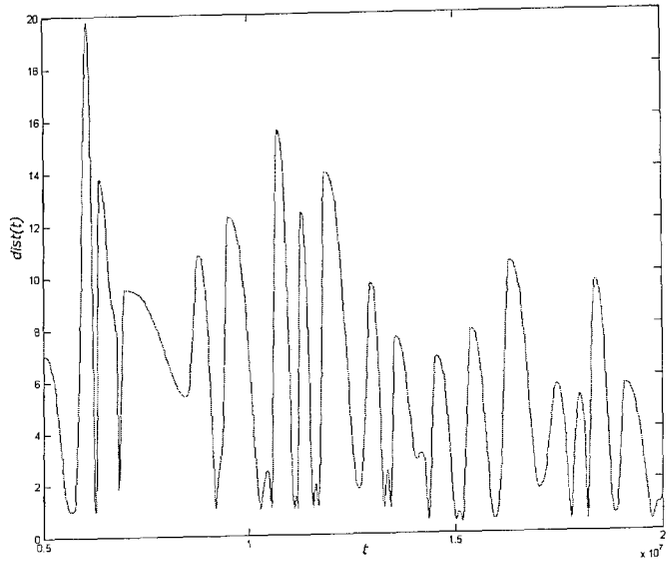


Figure 4.3: Motion with slow movement

A common previous approach to create animation to music is to assume the beat information is embedded in the MIDI file [13, 31], which is contingent on having an unwavering tempo throughout the piece, or by using beat extraction algorithms such as [20]. However, restricting dance motions synchronizing only to beat is merely one aspect of choreography, and the movements seen in dance are not necessarily accented only by the beat factor. Furthermore, consider music played *rubato* (freedom of tempo) as often found in classical and jazz music, or music in tempo but with tuplets (e.g. triplets, quintuplets). In these cases, synchronizing the motion with the beat is only one of the possible options; having motion that can be synchronized with articulation and melodic aspects of music is equally important.

4.2.2 Calculation of Spatial Peak Locations

Once the peaks have been selected, the system computes an exact location of the hand as follows. Let $\mu_m(t_q), \sigma_m(t_q), \mu_v(t_q)$, and $\sigma_v(t_q)$ be predicted by the model according to the music characteristics $u(t_q)$ at time t_q at which a peak occurs. A local mean hand distance and amplitude for $dist(t_q)$ are chosen by sampling a gaussian distribution $p(m)$ and $p(v)$ with these parameters. $p(m)$ and $p(v)$ are given by:

$$p(m) = \frac{1}{\sigma_m(t_q)\sqrt{2\pi}} e^{-\frac{(m-\mu_m(t_q))^2}{2\sigma_m(t_q)^2}} \quad (m > 0)$$

and,

$$p(v) = \frac{1}{\sigma_v(t_q)\sqrt{2\pi}} e^{-\frac{(v-\mu_v(t_q))^2}{2\sigma_v(t_q)^2}} \quad (v > 0)$$

Note that the $dist(t)$ function specifies distance of the arm, but does not indicate the

direction in which the distance should be chosen. While this is an additional characteristic that could be predicted by the system (discussed in Section 5.2), we currently randomly set several primary axes of motion on the x-y plane, with rotation angles around the origin being $\theta_1, \theta_2, \ldots, \theta_r$ for each character's hands as *mean* directions, and then we randomly choose one of them θ_k, and sample from a normal distribution centered at θ_k. θ_k is used for the fixed duration of time ΔT_θ, in order to avoid too much randomness in the resulting motion. ΔT_θ can be fixed for the entire motion, or change according to the length of musical phrases.

Together, the axis and distance specify a point relative to the character. The following are the steps to calculate the peak point $q(t_q) = \{x(t_q), y(t_q), z(t_q)\}$ at time t_q. (Figure 4.4)

1. Sample a local mean distance $m(t_q)$ and variance $v(t_q)$ from $p(m)$ and $p(v)$, respectively.

2. Randomly choose a mean primary axis θ_k from $\theta_1, \theta_2, \ldots, \theta_r$

3. Sample motion axis $\theta(t_q)$ from $p_\theta(\theta)$

$$p_\theta(\theta) = \frac{1}{\sigma_\theta \sqrt{2\pi}} e^{-\frac{(\theta-\theta_k)^2}{2\sigma_\theta^2}}$$

   where σ_θ^2 is the variance of the the distribution $p_\theta(\theta)$, which is randomly chosen $(0 < \sigma_\theta^2 < 5)$.

4. signed distance $dist(t_q)$ is given by:

$$dist(t_q) = m(t_q) + \sqrt{v(t_q)}$$

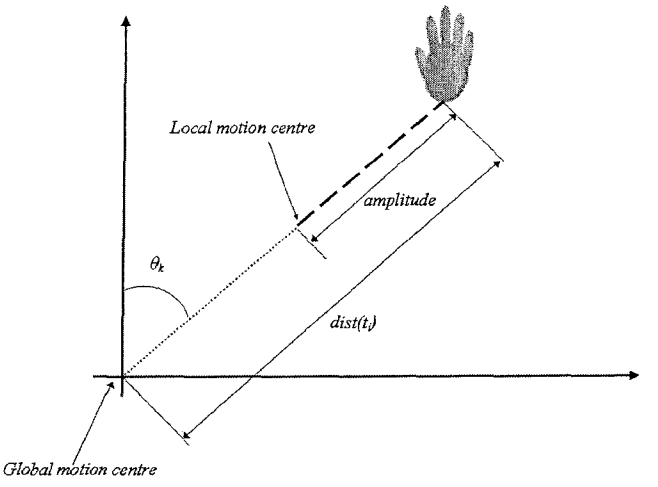


Figure 4.4: Calculation of Peak Locations

5. Finally, $q(t_q) = \{x(t_q), y(t_q), z(t_q)\}$ is calculated as:

$$\begin{aligned} x(t_q) &= dist(t_q) \cdot \sin\theta(t_q) \\ y(t_q) &= dist(t_q) \cdot \cos\theta(t_q) \\ z(t_q) &= 0 \end{aligned}$$

4.2.3 Insertion of Minimum Points

From the peak points computed in Subsection 4.2.2, we now insert another set of peaks, each of which appear in between the pre-defined peak points. The first group of peaks will become the zero-crossing points of the velocity along the motion direction, and near each peak, the first derivative is positive before and negative after the peak

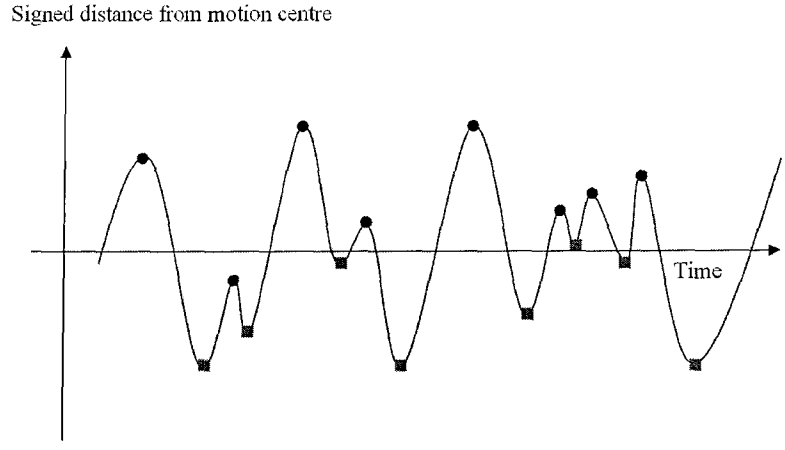


Figure 4.5: Signed distance and two sets of peaks

(black circles in Figure 4.5). On the other hand, the second groups of peaks will be the zero-crossing points of the velocity along the motion direction, but with its function's first derivative being negative before and positive after each peak (red squares in Figure 4.5).

To meet this condition, r_{t_r}, a peak point of the second group at time t_r must have a smaller (signed) distance than the peaks on its neighbouring peaks. The tentative location of this point is given by:

$$x(t_r) = dist(t_r) \cdot \sin\theta(t_q)$$

$$y(t_r) = dist(t_r) \cdot \cos\theta(t_q)$$

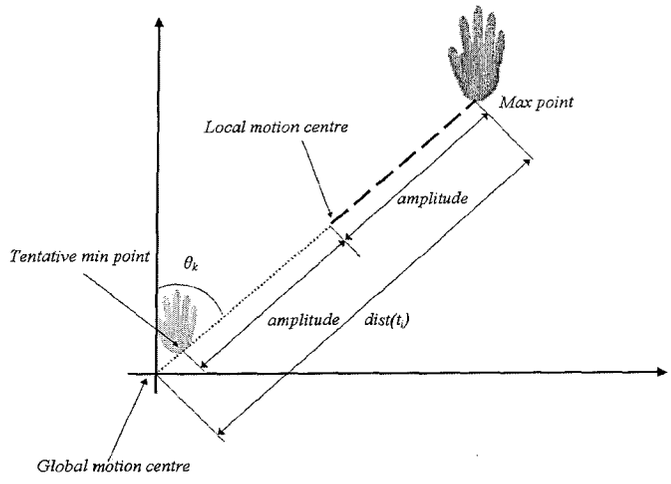


Figure 4.6: Tentative minimum point

$$z(t_r) = 0$$

where

$$dist(t_r) = m(t_q) - \sqrt{v(t_q)}$$

and this is shown in Figure 4.6.

Ideally, this tentative point will have a smaller value than both the neighbouring peaks. However, the signed distance of the next peak may be smaller than this tentative point. In this case, during the interpolation stage, it will create a saddle point, as shown in Figure 4.7, instead of creating an oscillating curve that is desired here.

Therefore, another tentative point that is computed from the next peak point is also considered at this time; that is:

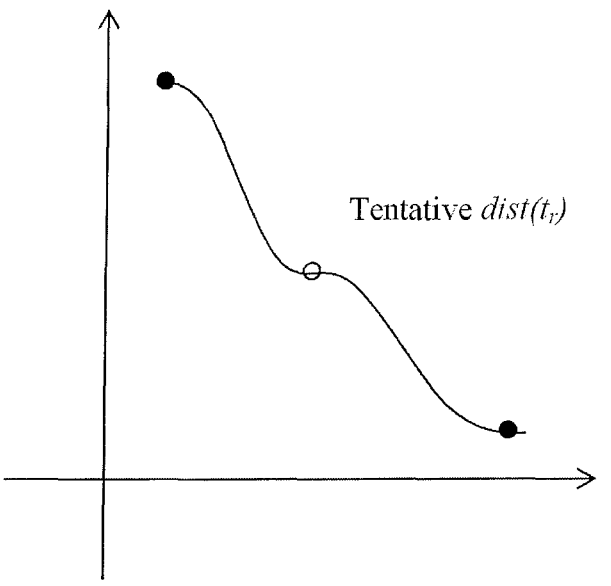


Figure 4.7: Tentative peak point has the larger distance than the next peak

$$
\begin{aligned}
dist(t_{r+1}) &= m(t_{q+1}) - \sqrt{v(t_{q+1})} \\
x(t_{r+1}) &= dist(t_{r+1}) \cdot \sin\theta(t_{q+1}) \\
y(t_{r+1}) &= dist(t_{r+1}) \cdot \cos\theta(t_{q+1}) \\
z(t_{r+1}) &= 0
\end{aligned}
$$

and, choose the appropriate minimum point according to the following criteria.

1. The tentative point with $dist(t_r)$ is chosen if it is smaller than both $dist(t_q)$ and $dist(t_{q+1})$

2. The tentative point with $dist(t_{r+1})$ is chosen if $dist(t_r)$ is larger than $dist(t_{q+1})$, but $dist(t_{r+1})$ is smaller than both $dist(t_q)$ and $dist(t_{q+1})$

Note that there may be a case, in which both $dist(t_r)$ and $dist(t_{r+1})$ are smaller than both $dist(t_q)$ and $dist(t_{q+1})$. This can be caused by the different θ values from which the two consecutive maximum points are computed. In this case, we enforce the minimum value to be zero. In other words, this peak point will be positioned at the global motion centre. This enforcement in fact ignores the amplitude value predicted by the model, but it makes sense as this point will be around the *transition* time (i.e., the time to change the direction of the movement, or the time when a newly computed theta becomes in effect) and it can be considered that the hand *retracts* before moving on to the next motion direction.

4.2.4 Interpolation

With the locations of peaks and their timing information, it is possible to interpolate between peak points using a quartic polynomial interpolation. Although these peaks are used as control points, all of them are also either local maxima or minima. Therefore, the first parametric derivatives of all the control points must be zero. However, this does not guarantee the continuity of the acceleration of the hand movements. To enforce the C^2 continuity, the second derivatives of both sides of a control points should be equal. Let the quartic function p of u defined as:

$$\begin{aligned} p(u) &= c_0 + c_1 u + c_2 u^2 + c_3 u^3 + c_4 u^4 \\ &= \vec{c}\,\vec{u} \end{aligned}$$

where

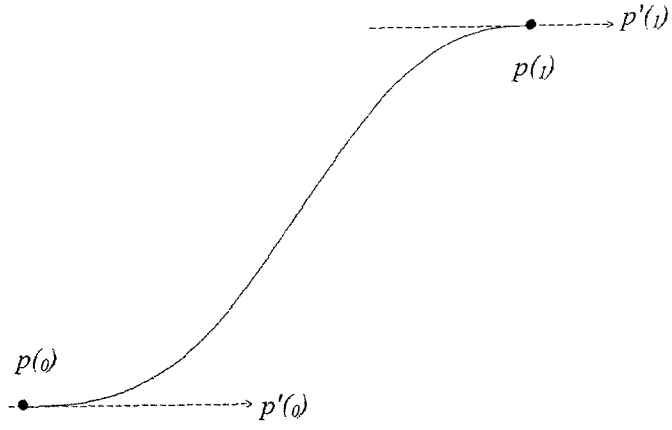


Figure 4.8: Interpolating between control points $p(0)$ and $p(1)$ on the x-coordinates

$$\vec{c} = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & c_4 \end{bmatrix}$$

and

$$\vec{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \\ u^4 \end{bmatrix}$$

Then, the conditions to interpolate between the control points (shown in Figure 4.8) are expressed as:

$$
\begin{aligned}
p(0) &= c_0 \\
p(1) &= c_0 + c_1 + c_2 + c_3 + c_4 \\
p'(0) &= 0 = c1 \\
p'(1) &= 0 = c_1 + 2c_2 + 3c_3 + 4c_4 \\
p''(0) &= \alpha = 2c_2
\end{aligned}
$$

and in a matrix form,

$$
\vec{\lambda} = \begin{bmatrix} p_0 \\ p_1 \\ 0 \\ 0 \\ \alpha \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix} \vec{c}
$$

where

$$
\vec{\lambda} = \begin{bmatrix} p(0) \\ p(1) \\ p'(0) \\ p'(1) \\ p''(0) \end{bmatrix}
$$

and α is the second derivative of the end point of the previous curve segment. Having the second derivative of the beginning point of the current segment equal to α ensures C^2 continuity at the joining points.

Therefore, the interpolation matrix M for this interpolation is given by:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} \\ -4 & 4 & -3 & -1 & -1 \\ 3 & -3 & 2 & 1 & \frac{1}{2} \end{bmatrix}$$

Note that although we have shown the interpolation using the values for the x-axis as an example, the same interpolation matrix can be used for all the axes without losing generality. We perform this interpolation separately for each axis.

Now we define the blending polynomials $b(u)$ of this interpolation as:

$$b(u) = M^T \vec{u}$$

and then we can rewrite the equation for $p(u)$ as:

$$\begin{aligned} p(u) &= b(u)^T \vec{\lambda} \\ &= b_0(u)p(0) + b_1(u)p(1) + b_2(u)p'(0) + b_3(u)p'(1) + b_4(u)p''(0) \end{aligned}$$

Therefore, these blending polynomials are given by the equations:

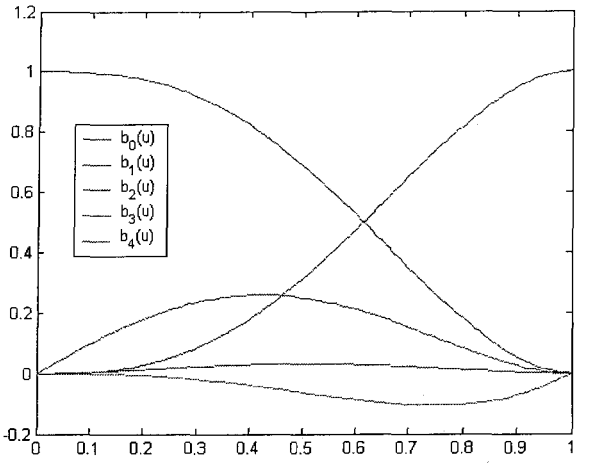


Figure 4.9: Blending functions

$$\vec{b}(u) = \begin{bmatrix} 3u^4 - 4u^3 + 1 \\ -3u^4 + 4u^3 \\ 2u^4 - 3u^3 + u \\ u^4 - u^3 \\ \frac{1}{2}u^4 - u^3 + \frac{1}{2}u^2 \end{bmatrix} = \begin{bmatrix} (u-1)^2(3u^2 + 2u + 1) \\ -3u^3(u - \frac{4}{3}) \\ 2u(u-1)^2(u + \frac{1}{2}) \\ u^3(u-1) \\ \frac{1}{2}(u-1)^2 \end{bmatrix}$$

As seen in Figure 4.9, these five polynomials have none of their zeros inside the interval $(0, 1)$ and they are smooth in the range.

4.3 Inverse Kinematics (IK) and Link Orientations

The orientation angles at the characters' links are computed using an IK algorithm (see Appendix A.2). The shoulder position in 3D space can be specified by the user,

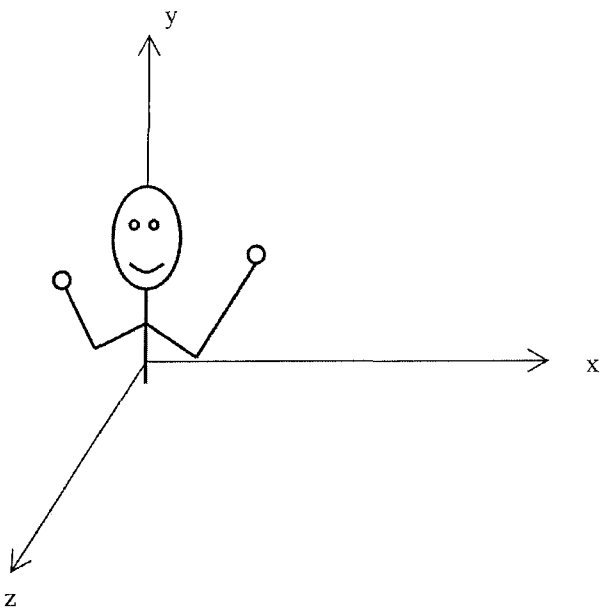


Figure 4.10: Conventional right-hand 3D coordinate system

or by reading in a file containing a specification of the character's body structure (.asf files of Acclaim format). Because the solutions to this IK problem only occur on the plane Ω that is defined by the two links, humerus and radius, we first solve it analytically on Ω (2D), which is kept at an orientation such that Ω is always perpendicular to the x-z plane (the figure is looking in the positive z-axis direction in the conventional right-hand coordinate system, as shown in Figure 4.10), and then find orientation of Ω based on the hand orientation. Even with this constraint, however, there may be two correct solutions, as shown in Figure 4.11. In this case, we use a heuristic approach to choose the solution that has a lower elbow location.

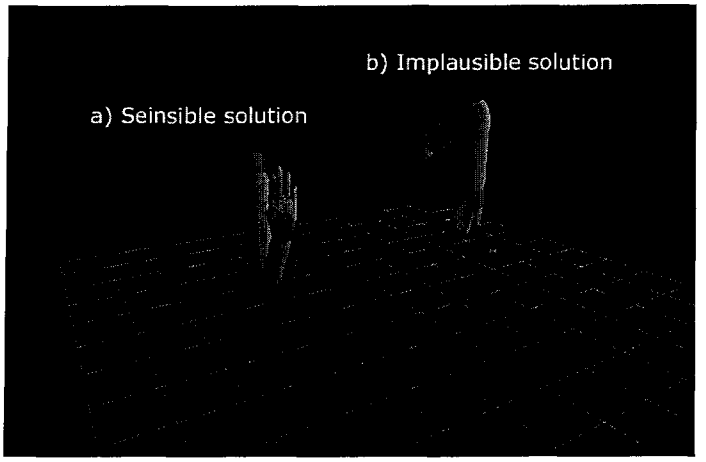


Figure 4.11: Solutions to IK. The red spheres indicate target point and shoulder position

4.3.1 Computing Orientations

Define the hand's default orientation such that the palm faces the positive z-axis direction while the fingers are pointing upwards (the positive y-axis direction) and its bottom center positioned at the origin of its local coordinate system (Figure 4.12). The orientation of the hand is typically determined by three rotation angles but we consider two; one is the rotation around the z-axis and the other around the x-axis in the local system. Let the rotation angles around the local x-axis and z-axis be ϕ_x and ϕ_z, respectively (Figure 4.13 and 4.14). While there are other options, we use two different heuristics to determine ϕ_x and ϕ_z; ϕ_x is determined based on the locations of the hand and shoulder to attain a natural posture, and ϕ_z is obtained from the velocity of the hand movement to add some fluidity to the motion.

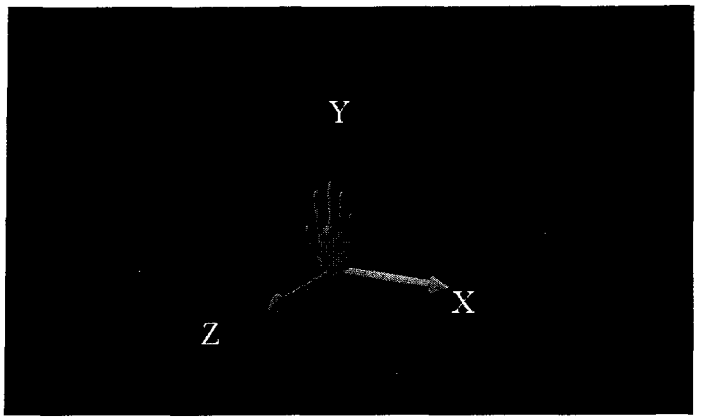


Figure 4.12: Default Hand Orientation

Determining ϕ_x

ϕ_x is determined so that the normal vector of the palm is collinear with the vector from the shoulder position to the hand position. Let the shoulder position in the global coordinate system be $p_s = [x_s, y_s, z_s]$, and the current hand position $p_h = [x_h, y_h, z_h]$, the vector $\vec{w}$ from the shoulder to the hand is given by:

$$\vec{w} = [x_v, y_v, z_v] = p_h - p_s = [x_h - x_s, y_h - y_s, z_h - z_s]$$

Therefore, ϕ_x is computed as:

$$\phi_x = \arccos\left(\frac{y_v}{\sqrt{x_v^2 + y_v^2 + z_v^2}}\right)$$

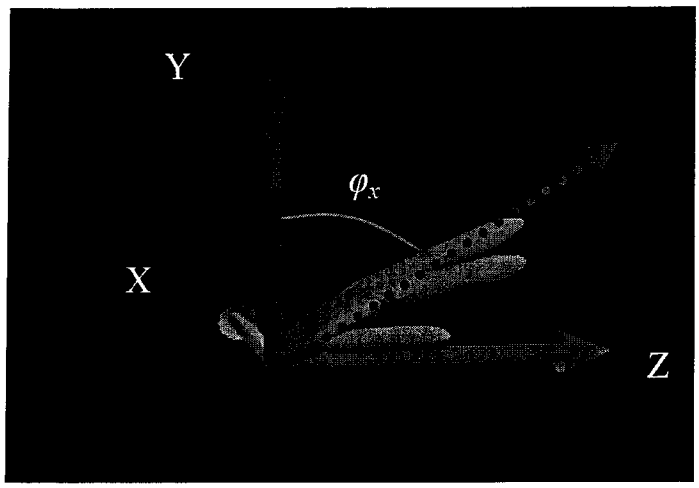


Figure 4.13: Hand orientation about the x-axis ϕ_x

Determining ϕ_z

Unlike ϕ_x, which is computed in relation to the locations of the hand and shoulder, ϕ_z is computed from the x component of the velocity of the hand movement v_x, as:

$$\phi_z = Cv_x$$

where C is a constant value controlling the amplitude of ϕ_z.

By setting this rotation angle ϕ_z proportional to the velocity, the wrist movement looks more fluid as this *imitates* one aspect of the physical properties (inertial property). Although setting C to a value that is too large results in unnatural wrist motion, a larger C usually adds some effects such as *exaggeration* and *appeal* that are part of the principles of animation described in [16].

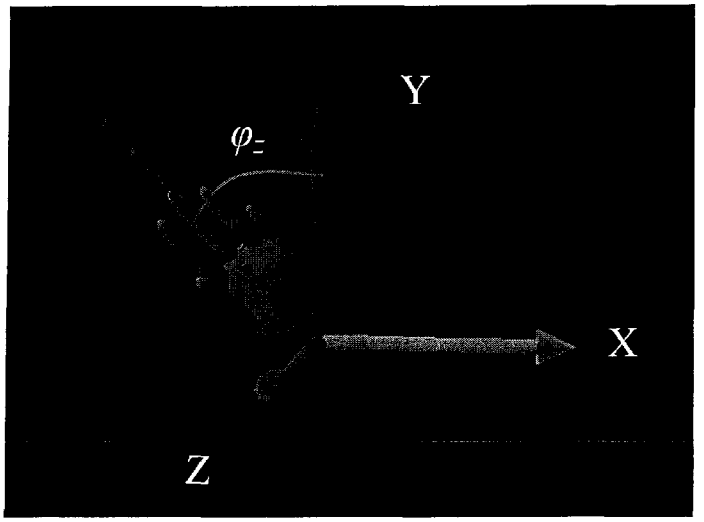


Figure 4.14: Hand orientation about the x-axis ϕ_z

Determining Arm Orientation

Finally, the orientation of the arm is determined based on the orientation of the hand. Let α be the line through the locations of the hand and shoulder, and β the line passing through the center of the hand from the bottom of the palm to the tip of the middle finger, we rotate Ω around α so that Ω contains β. This is shown in Figure 4.15.

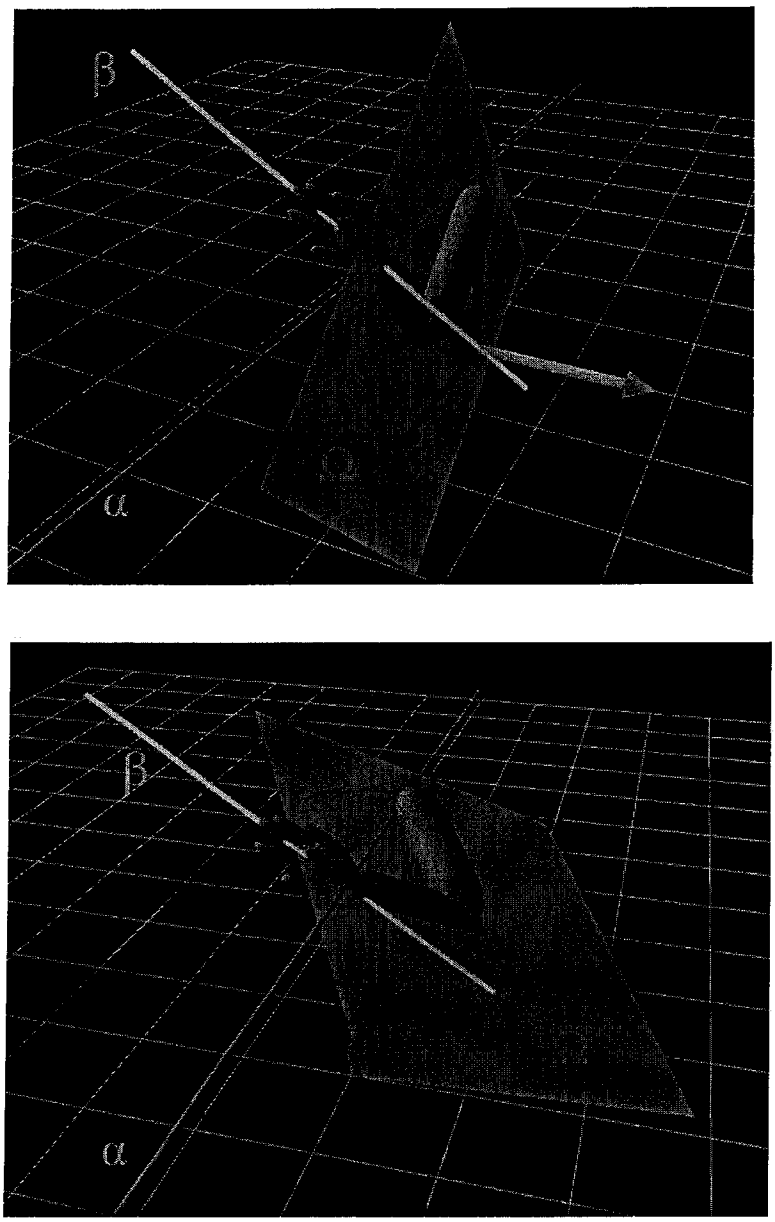


Figure 4.15: Orientation of Arm: Rotating plane Ω to contain line α

Chapter 5

Results and Conclusion

5.1 Results and Discussion

The model was trained on a data set consisting of several music pieces, with multiple motion recordings for each piece. We first show the results of the creation of new motions, corresponding to the music data that were used to train the model, and then the results of the new motions generated using new musical data that the system has not seen during the training process.

The musical pieces range from highly rhythmic with steady tempo, to having rhythmic patterns that shift from slow and rubato to fast, demonstrating that the characters respond appropriately to these complex changes. Table 5.1 shows the music tracks that are used to train the models (and also used to test and evaluate the system's output), and Table 5.2 shows the list of music tracks that the models are tested on.

5.1.1 New Motions to the Music Data Used for Training

Comparing the results with the examples is one of the important ways to evaluate the system. In this section, we train the models on the music listed in Table 5.1, with

#	Genre	Tempo	Description (Metre)
1	Classical/Ethnic	Slow	$\frac{4}{4}$ notes. Exaggerated variation in loudness $(\frac{4}{4})$
2	Popular/Jazz	Medium	My Favorite Things $(\frac{3}{4})$
3	Popular	Fast	8th notes improvised line $(\frac{3}{4})$

Table 5.1: The list of music tracks used to train the model, and also test to evaluate how closely it produces output animation that shares the characteristics of the example motions.

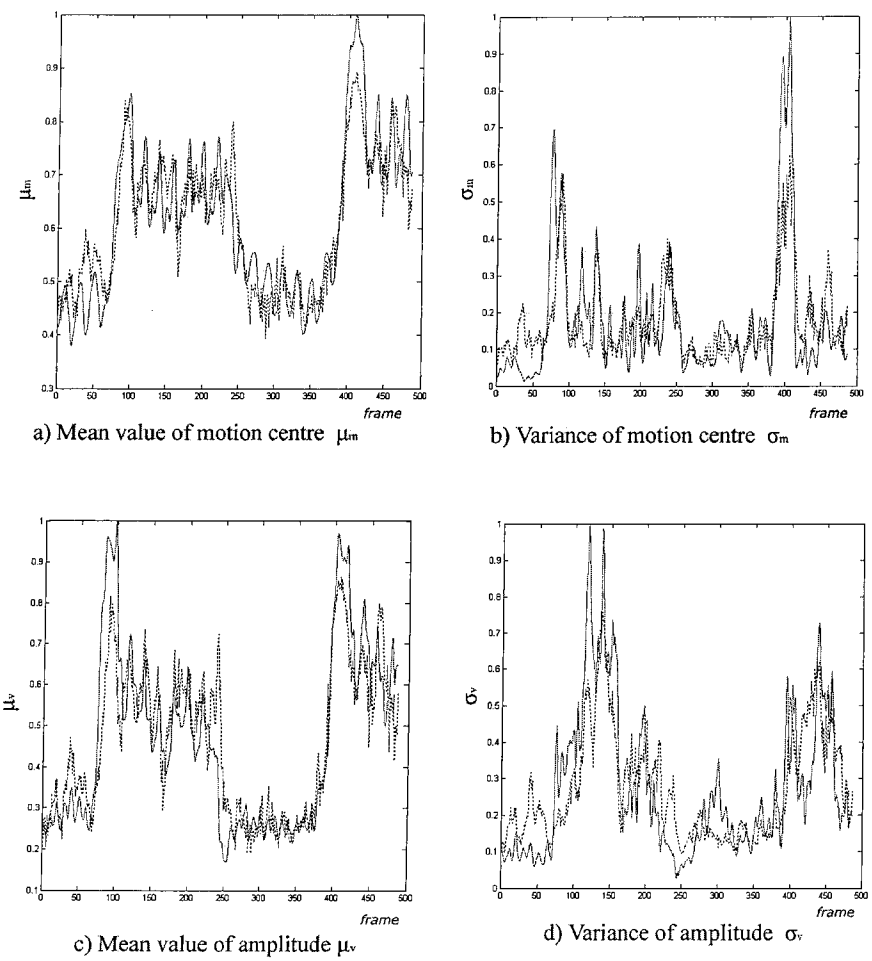
#	Genre	Tempo	Description (Metre)
4	Jazz	Fast/Medium	Four $(\frac{4}{4})$
5	Ethnic	Slow to fast (varying tempo)	$(\frac{4}{4})$
6	Classical/Folk	Medium	Mazurka $\frac{3}{4}$

Table 5.2: The list of new music tracks that are used to generate motion curves. Note that these are not used during the training phase.

the several motion tracks danced to the music, and then use the same music tracks to create new motions.

Figures 5.1, 5.2, and 5.3 show the original values (blue) of spatial parameters μ_v, σ_v, μ_m, and σ_m and the output result from the model (dotted lines), and Figures 5.4, 5.5, and 5.6 show the original and predicted values of temporal parameter *peak*. Figure 5.7 also shows the target and predicted temporal parameter values for the music track 1, but shows only a smaller section for a more detailed view. Although the predicted data are not exactly on the target, the predicted peaks happen in a certain range from the target peaks.

Using these output values from the models, we now generate motion curves as described in Section 4.2. Figures 5.8, 5.9, and 5.10 show several motion curves that are newly created (dotted line), with their original example motion curves for the same music (other solid lines). The peaks of the new motion occur around the frames



a) Mean value of motion centre μ_m

b) Variance of motion centre σ_m

c) Mean value of amplitude μ_v

d) Variance of amplitude σ_v

Figure 5.1: Values learnt by the model for the training music #1

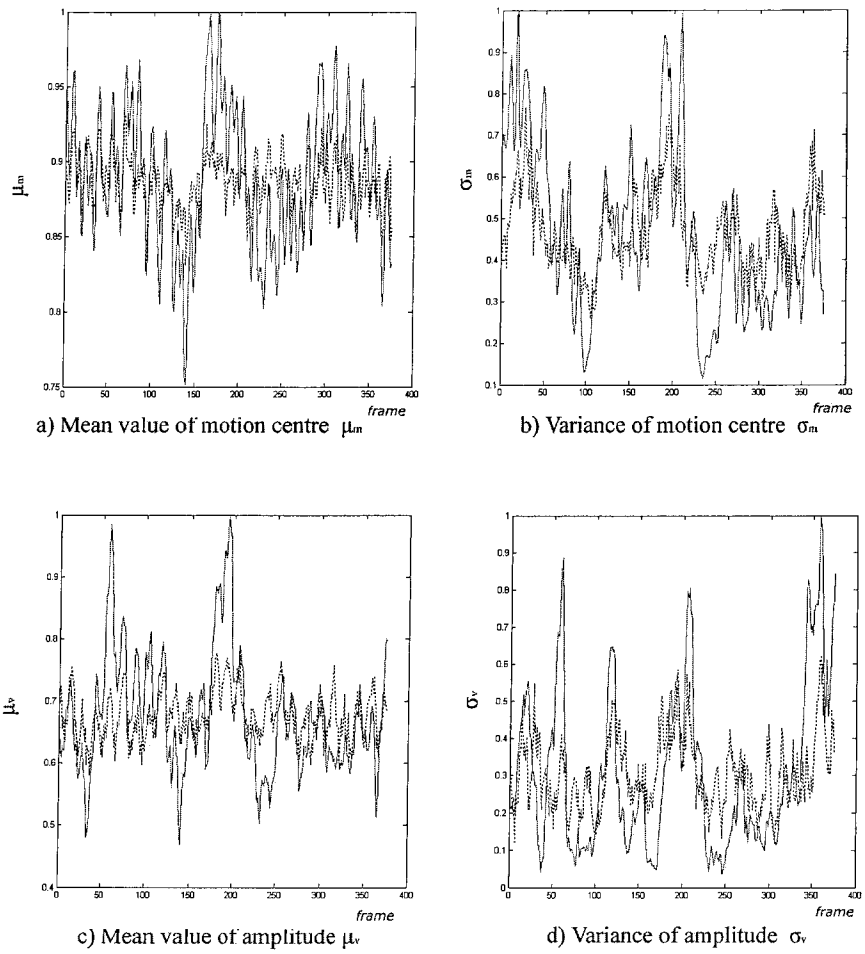


Figure 5.2: Values learnt by the model for the training music #2

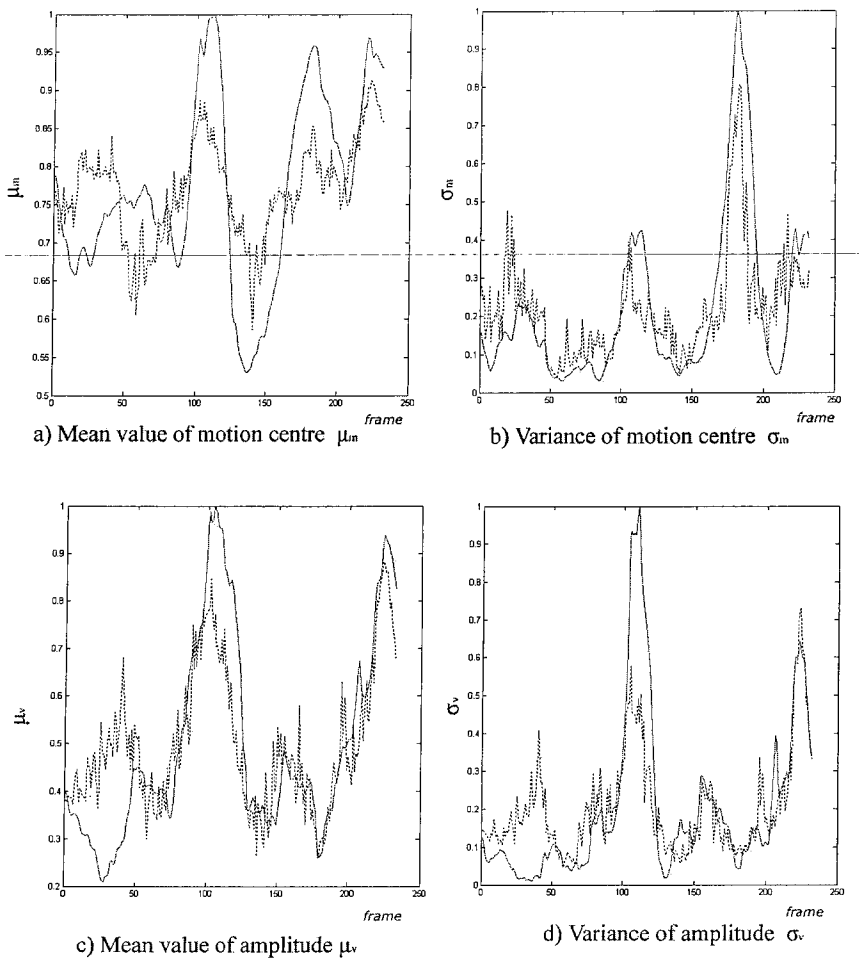


Figure 5.3: Values learnt by the model for the training music #3

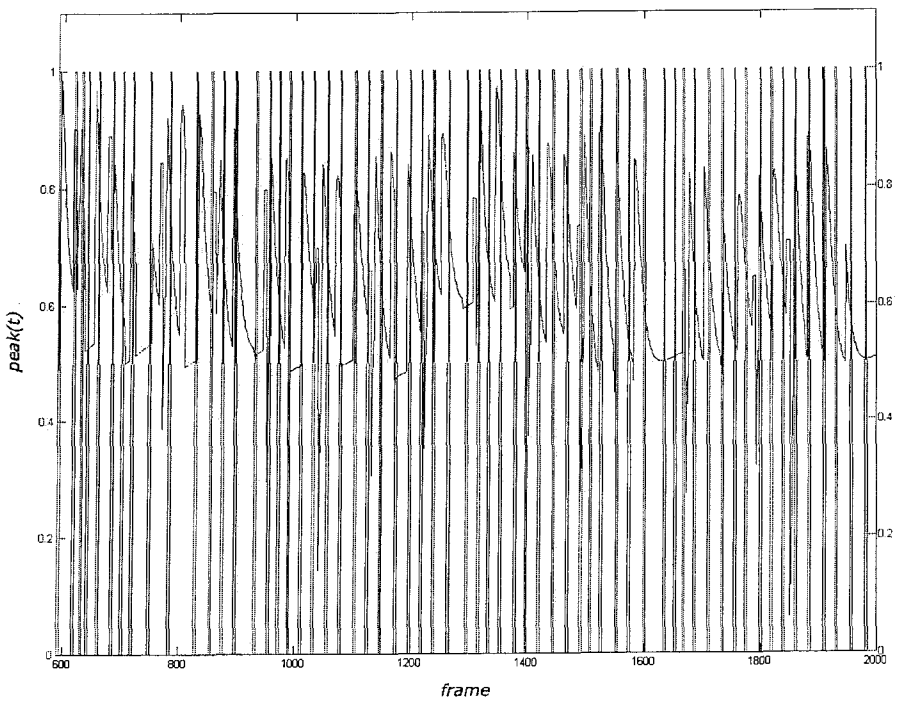


Figure 5.4: Peak probability *peak* for the training music #1: Blue (target data), Green (likelihood predicted by the model)

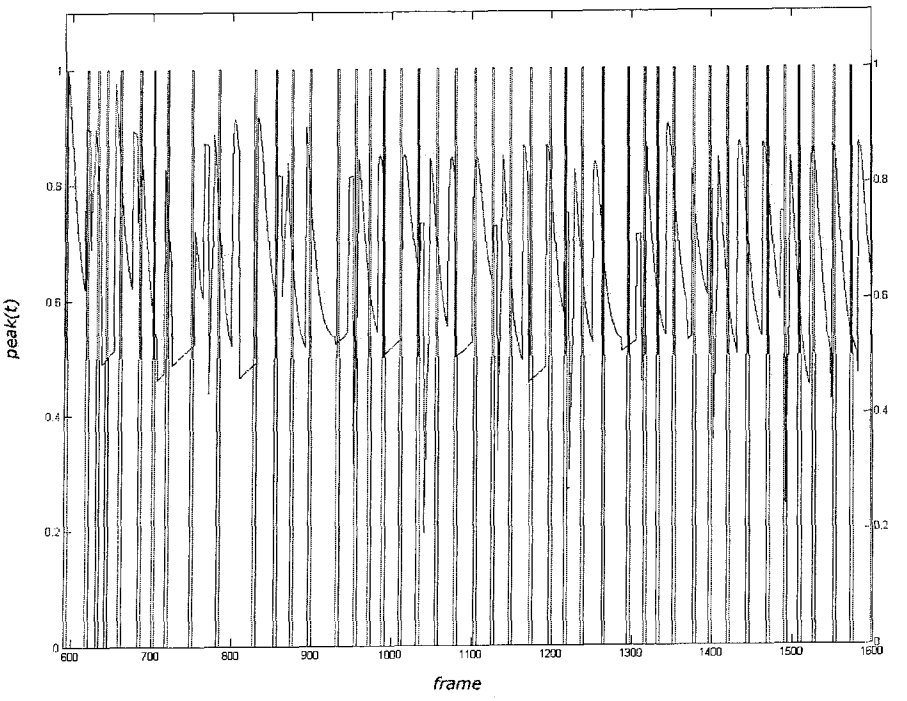


Figure 5.5: Peak probability *peak* for the training music #2: Blue (target data), Green (likelihood predicted by the model)

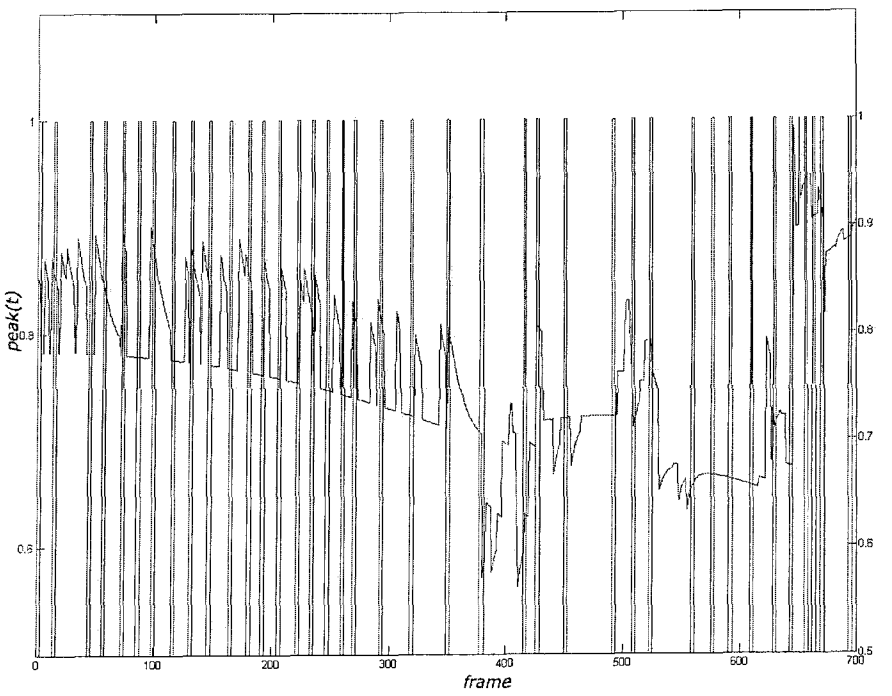


Figure 5.6: Peak probability *peak* for the training music #3: Blue (target data), Green (likelihood predicted by the model)

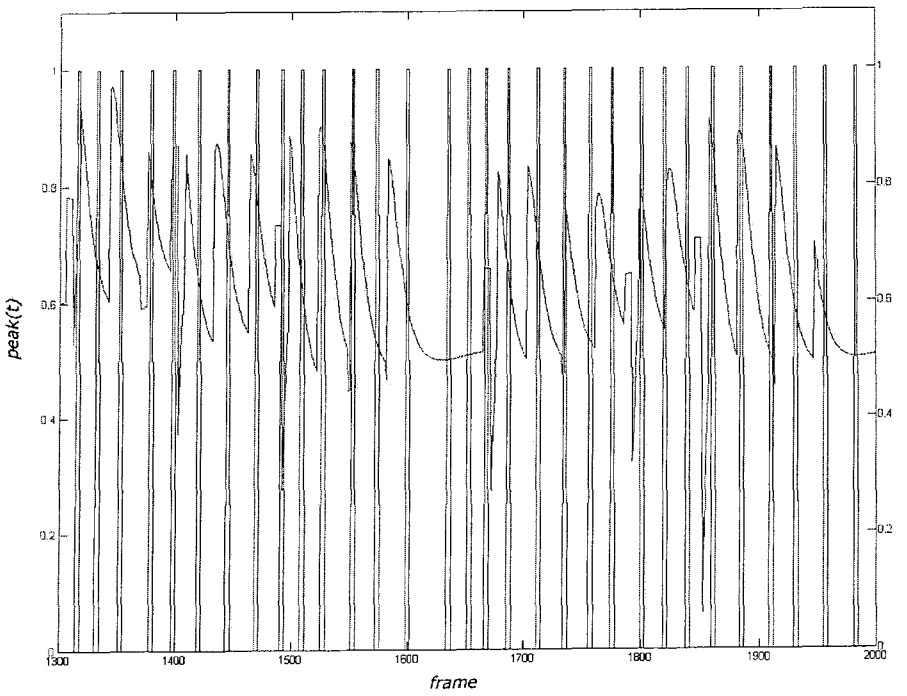


Figure 5.7: Peak probability *peak* for the training music #1, sampled to show in detail: Blue (target data), Green (likelihood predicted by the model). Although the predicted data are not exactly on the target, the predicted peaks happen in a certain range from the target peaks.

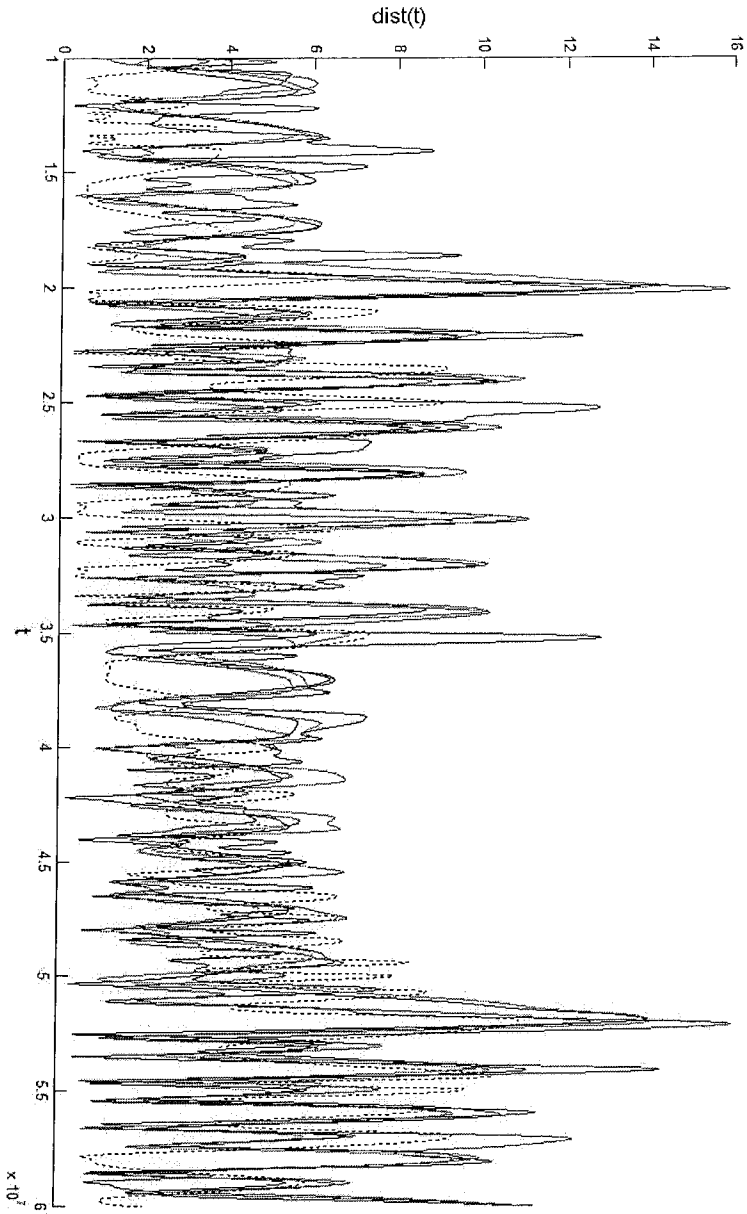


Figure 5.8: New motion ($dist(t)$) and the original example motions for the training music #1

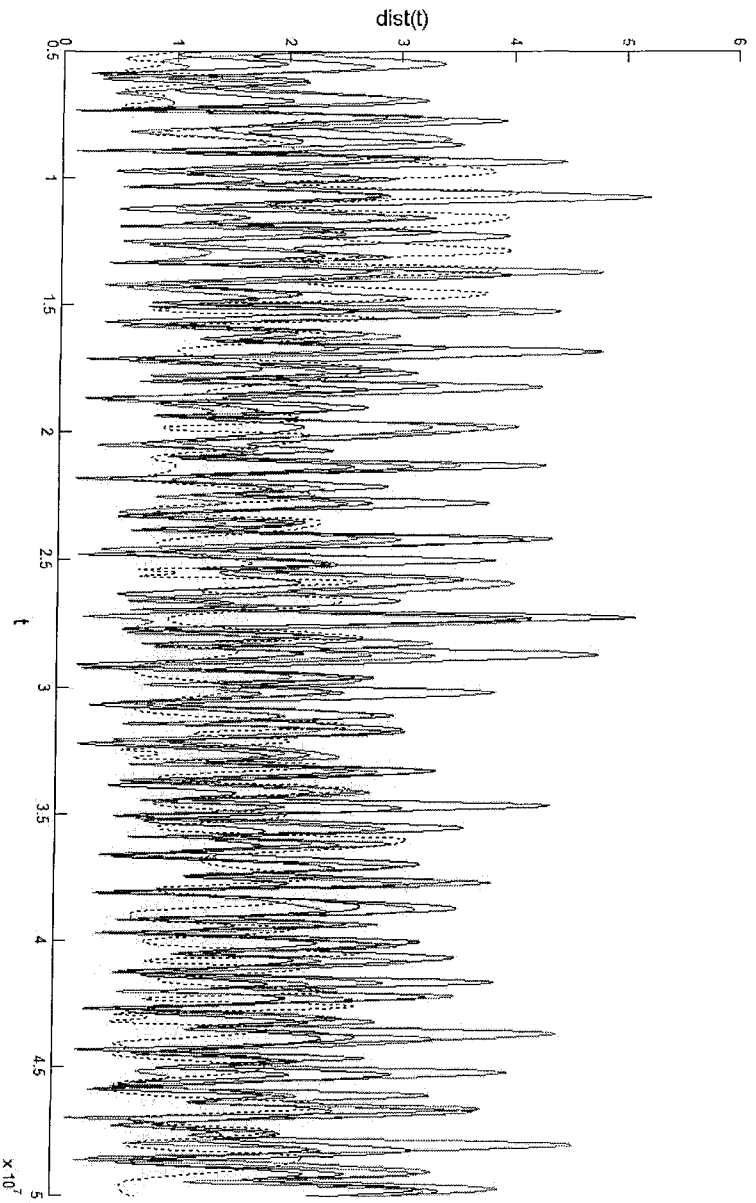


Figure 5.9: New motion ($dist(t)$) and the original example motions for the training music #2

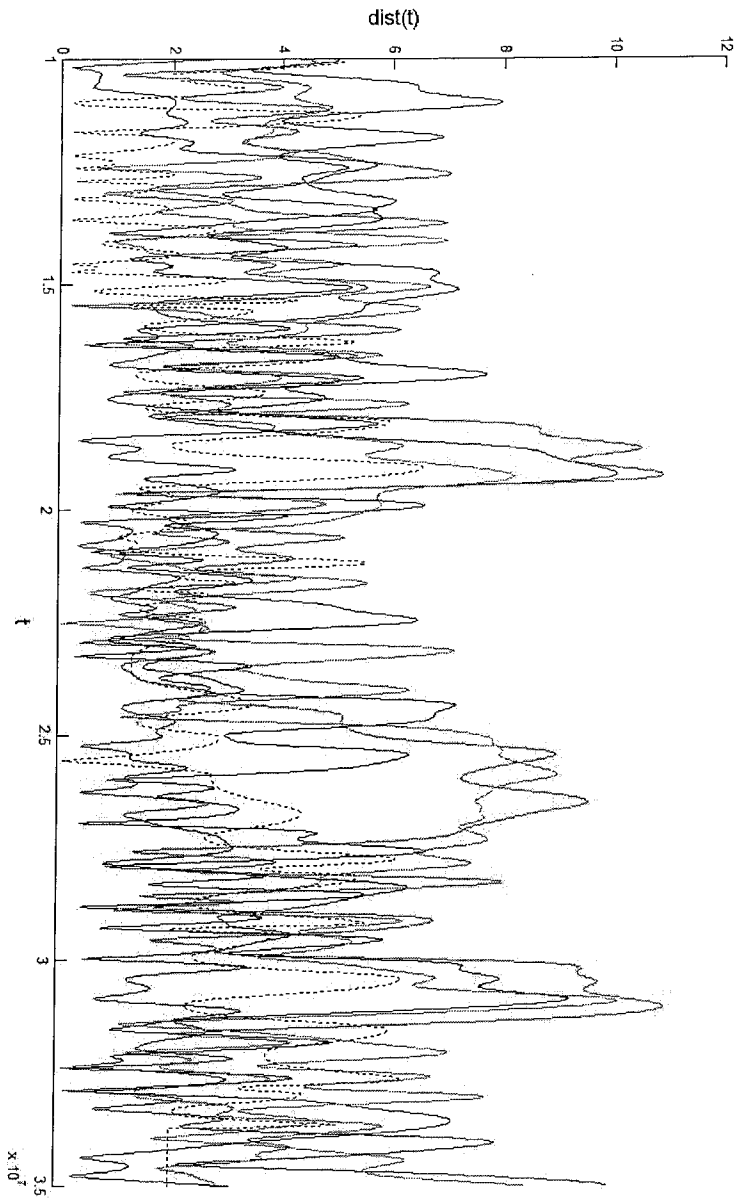


Figure 5.10: New motion ($dist(t)$) and the original example motions for the training music #3

at which many peaks of the example motions occur, and at these peak points, the amplitude of the generated motion curve approximately follows that of the example motions. This suggests the system did indeed generate new motions that share certain characteristics of the original example motions by analyzing the proposed features of music and motions.

5.1.2 New Motions to the New Music Data

One of the features of the system is to be able to create motions to musical lines that were not used in the training process, yet automatically produce resulting motions that are similar in style to the example motions. Using the models trained in the previous section, we now generate new motions to the new music tracks shown in Table 5.2 on Page 56.

Figures 5.11, 5.12, and 5.13 show the motion curves (distances from the global centre) created by analyzing the music tracks in Table 5.2, and Figures 5.14, 5.15, and 5.16 show screenshots of the resulting animation. The system was capable of handling highly rhythmic music (track 4) as well as responding to tempo changes within a single tune (music track 5). Because the peak points are stochastically chosen and the user can select different motion speed, the system has created a few motions, which are not identical yet able to share certain characteristics, to one music track.

5.2 Conclusion and Future Work

We have demonstrated a novel approach to creating animated motion to new musical scores. We propose a set of key features of music and motions, and then extract these

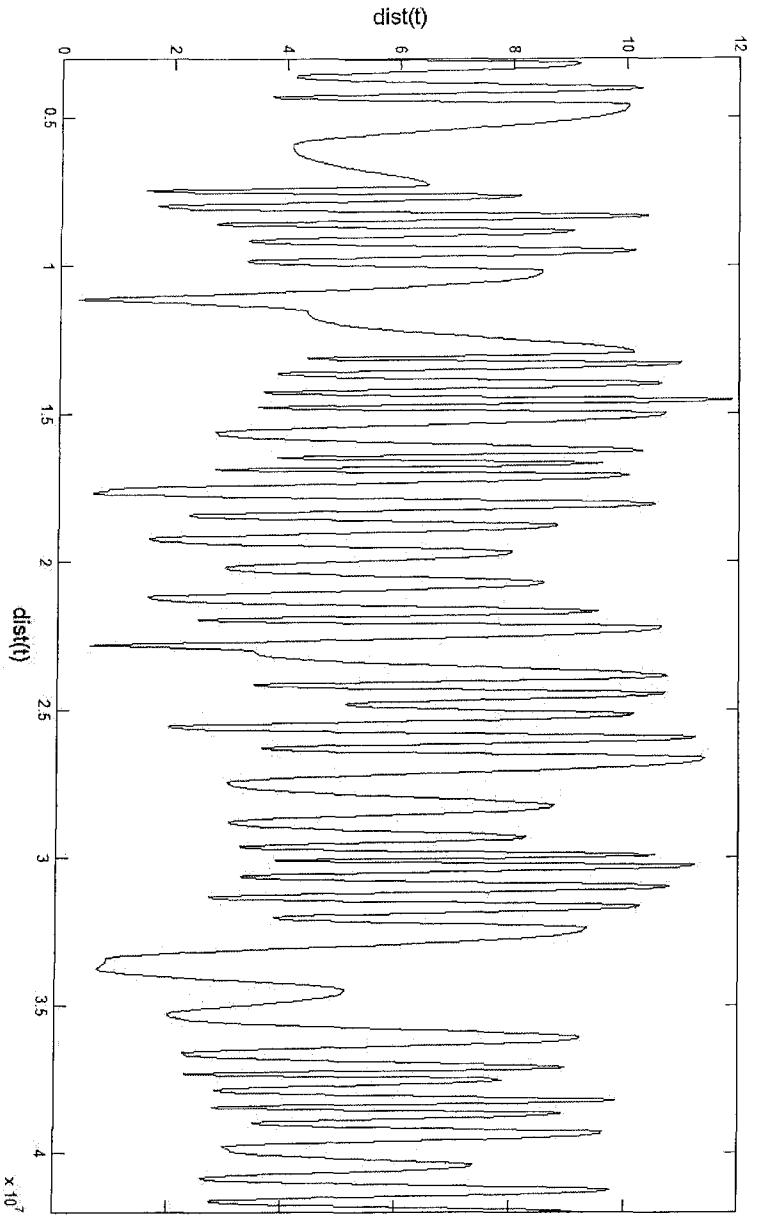


Figure 5.11: New motion curve (distance) to the testing music #4

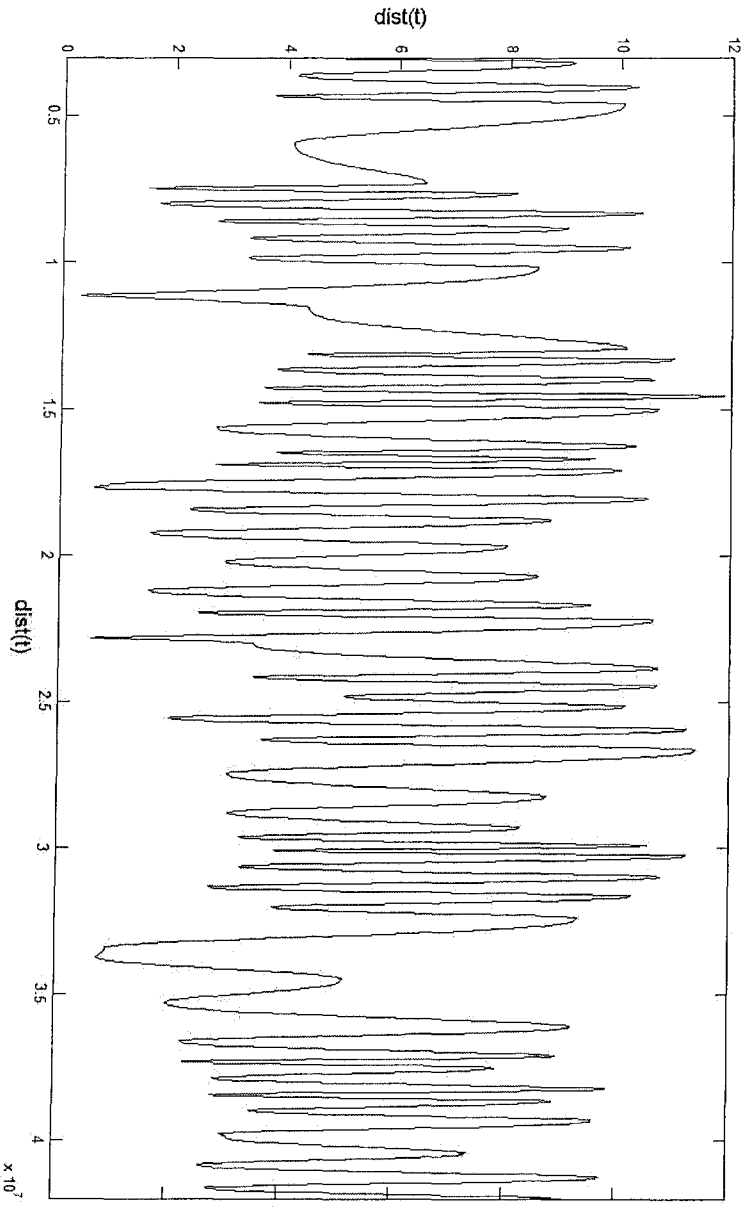


Figure 5.12: New motion curve (distance) to the testing music #5

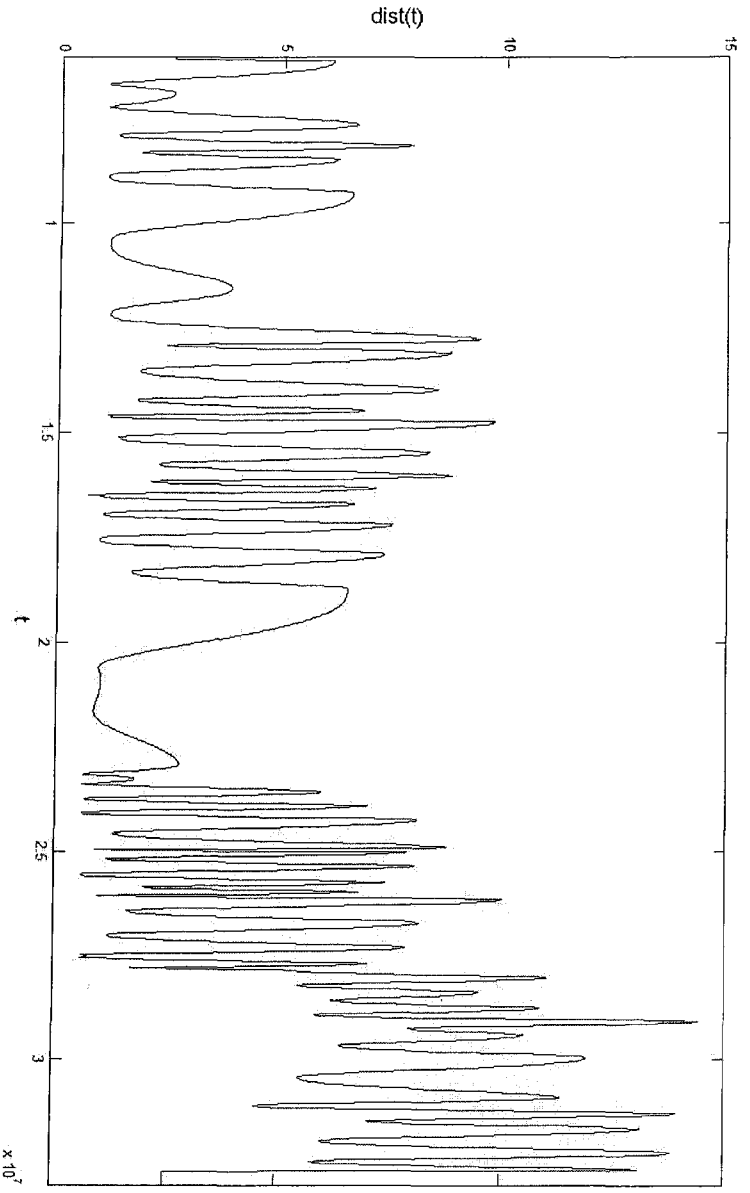


Figure 5.13: New motion curve (distance) to the testing music #6

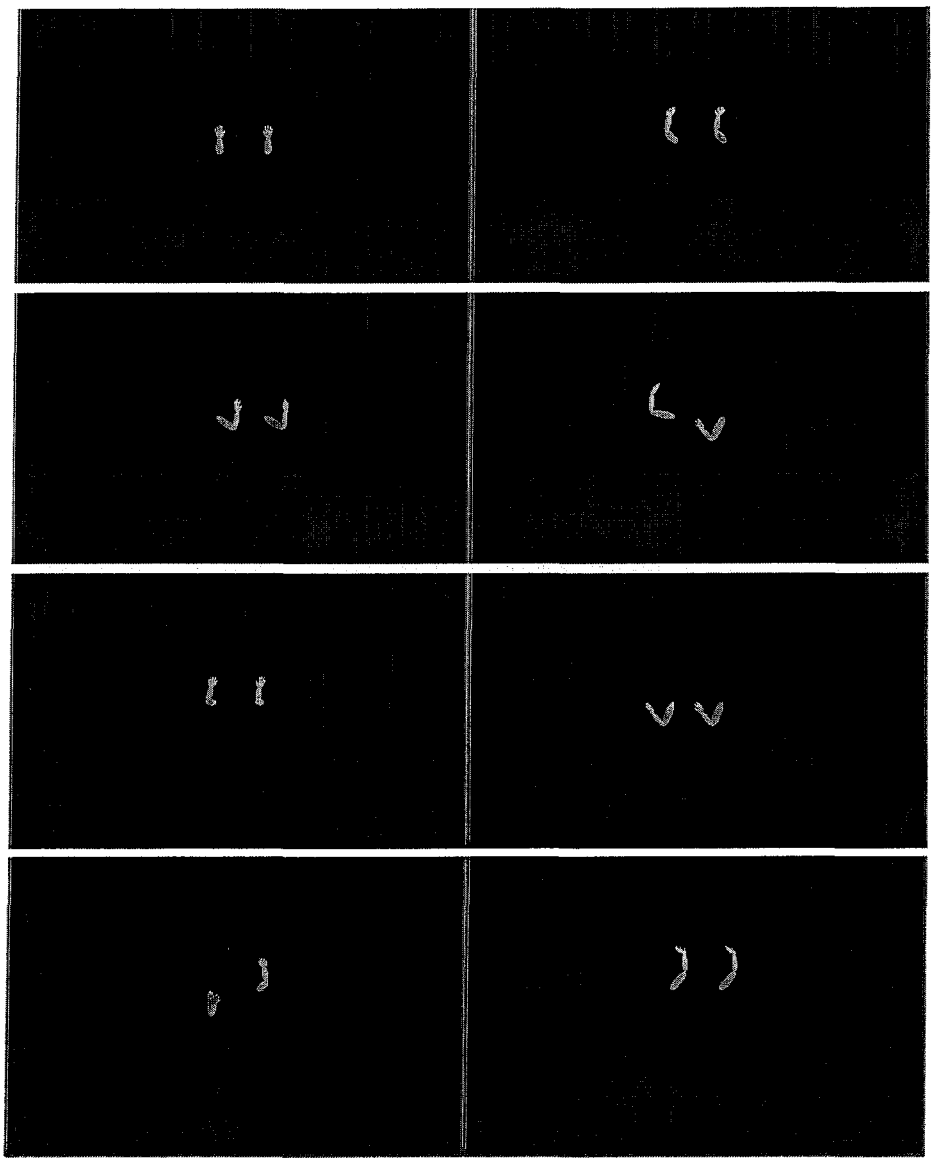


Figure 5.14: Screenshots of the motions to the testing music #4

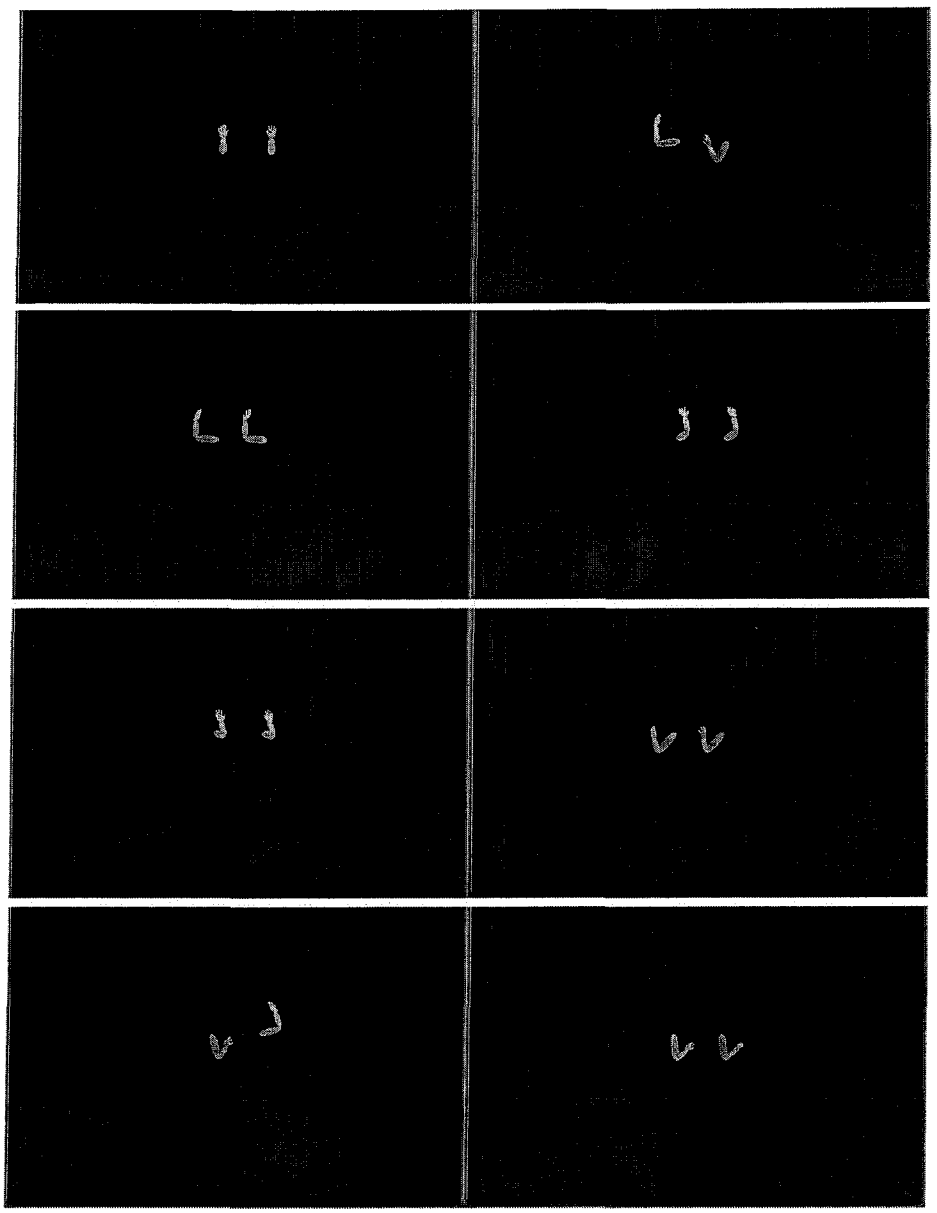


Figure 5.15: Screenshots of the motions to the testing music #5

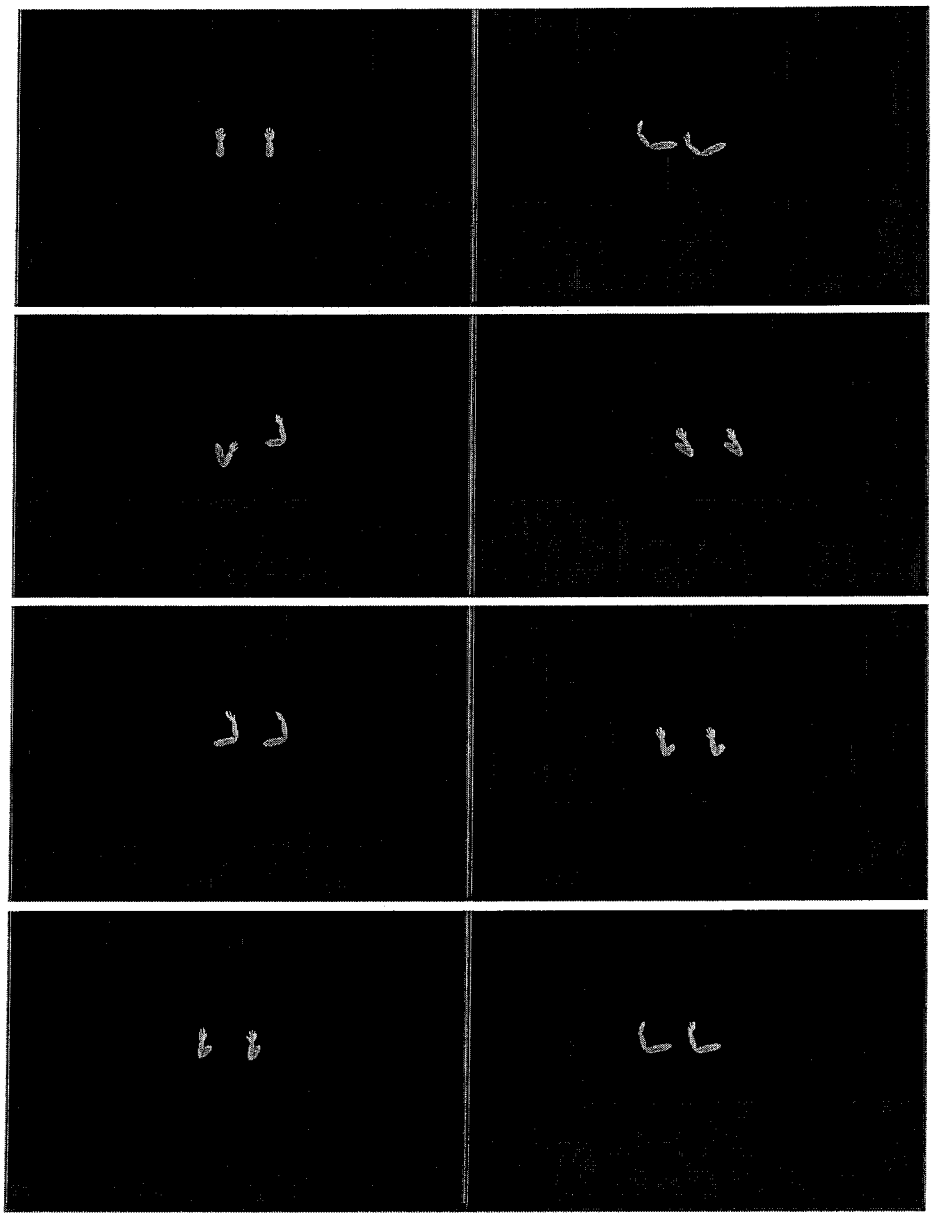


Figure 5.16: Screenshots of the motions to the testing music #6

features. Rather than adapting existing recorded dance motion, we then use recorded motions synchronized to music soundtracks, and automatically analyze salient characteristics of both the motion and the music. A feed-forward neural network can then be used to learn the relationship between these characteristics by example, rather than by requiring an animator to specify mappings explicitly.

Even though it can generate animations automatically, the system provides flexibility for the animator to control the speed of the animation. It is very likely that two different animators would like different motion speeds to the same sound track, depending on their needs.

Our work focuses on animating motion of hand and arm, yet the same ideas could be extended naturally to include a range of body motion. An obvious target would be legs. For this, unlike somewhat free movements of hands, we would have to have more constraints in order to attain physically plausible postures. For example, the maximum extent of legs cannot go lower than the floor level, or at least one of the feet must be touching the floor at all time. Another possible way to include more parts of body is to use techniques such as style-based inverse kinematics [11] to manipulate many more degrees-of-freedom with a low-dimensional representation such as hand position.

Another aspect to be explored is to incorporate a predictor for the direction in which the hands move. The current system chooses a few directions from a Gaussian distribution, but this characteristic, too, can be learnt from examples. It may be the case that the directions and other characteristics are affected not only by the musical features, but are dependent on the temporal information as well.

Currently, our system extracts only a limited number of features of motion and music. We would like to extend our system to include more parameters to capture more features. For investigation purposes to determine which features to be included, only a small number of points indicating some of the body parts such as hands, feet, elbows, and shoulders, rather than rendering full links may be performed. Doing so may help us understand other visually salient features in the motion. Our system could also be extended by further analysis of the musical input, such as harmonic function, and again, use of the temporal information of music may be very useful as the repetitive occurrences of musical phrases may be matched to that of the motions.

Appendix A

Techniques Overview

A.1 Musical Instrument Digital Interface (MIDI)

To understand how music applications can be implemented, one must understand how music data can be transferred between digital instruments and computer hardware media, in addition to the music theory and tonal systems. With its first standardization in 1983 [12], MIDI has gained popularity among musicians, who have been interested in using computer technologies to produce their music. MIDI in its pure format can be thought as the representation of notes that are to be played. There are three kinds of basic information contained in MIDI messages:

1. *Command*: What to do e.g., play a note (*NOTE ON*), or stop a note (*NOTE OFF*)[1]

2. *Pitch*: Which note should be played (*middle C* is set to the value 60, one half-step corresponds to 1 of the pitch value: e.g. $C\# = 61$, $D = 62$)

[1] *Command* can be also a command to switch the sound voice or other controls to affect the system's configurations, but only the NOTE ON and NOTE OFF commands are relevant for the purposes of this research.

3. *Velocity*: How fast a key is pressed; hence, how loud the note should be played (ranging from the minimum value of 0 to the maximum value of 127).

A typical MIDI message is represented with a series of bytes such as:

10010000 00111100 01000000

The interpretation of this message is "Play a note (NOTE ON; the first 4 bits are 1001) with the pitch C4 ($00111100_2 = 60_{10} = C4$) and the velocity 64 ($01000000_2 = 64_{10}$)"

When MIDI messages are played and received in real time systems, such as pressing keys on the keyboard, messages are generated and directly sent to a device that synthesizes sound (*synthesizer*). In this case, it is unnecessary to have the information about timing. However, if the messages are stored and played back later, some sort of the timing information is necessary. In the standard MIDI files, a MIDI message of each note is associated with a time at which the note should be played.

There are three types of the standard MIDI file formats: Type 0, 1, and 2. Files in Type 0 format store all the MIDI messages in a single data stream, or a *track*, while Type 1 files can have multiple tracks, which make it easy to view and edit the music when it has more than one part, such as orchestral and choral music. Type 2 files are very rarely used in MIDI sequencer applications, even though they can store multiple songs or patterns in a single file.

A.2 Inverse Kinematics (IK)

Forward kinematics is the process of computing the position of the end effector of a structure by specifying the angles of all the joints. This is fairly straightforward,

and it always has exactly one solution. *Inverse kinematics*, on the other hand, is the process of figuring out the joint angles when given the position of the end effector and possibly other constraints, such as the orientation of the end effector. There can be zero, one, or more solutions, depending on the constraints given by the user. The term *overconstrained* means that there is no solution because there are too many constraints on the configuration. It is said *underconstrained* when there are more than one solutions because there are not enough constraints.

When the mechanism is simple, then the final configuration of the joint angles can be solved analytically. In this case, the animation of the mechanism can be done by interpolating the angles between the initial and final pose vectors (vector of all the joint angles). When it is much more complicated, the solution is usually obtained by employing the *Jacobian* matrix, which relates the incremental changes in the joint angles to the changes in the position and orientation of the end effector. ([23])

A.2.1 Analytical Solution for a Simple System

Suppose a very simple system in two-dimensional space shown in Figure A.1. L_1 and L_2 are the length of the first and second links respectively, θ_1 the joint angle of the first link relational to the fixed wall, and θ_2 the joint angle of the second link relational to the first link. Assuming the base is fixed at the origin, the values of the $(X,\ Y)$ coordinate is given by the user as the desired position of the end effector.

By computing the distance from the origin to this desired position, it is possible to determine the joint angles θ_1 and θ_2. First, we compute the interior angles of the triangle shown in Figure A.2.

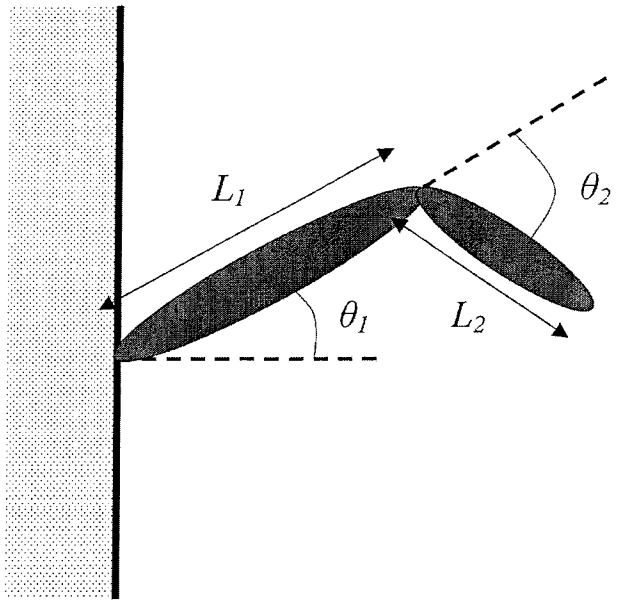


Figure A.1: Analytical solution for a two-link system

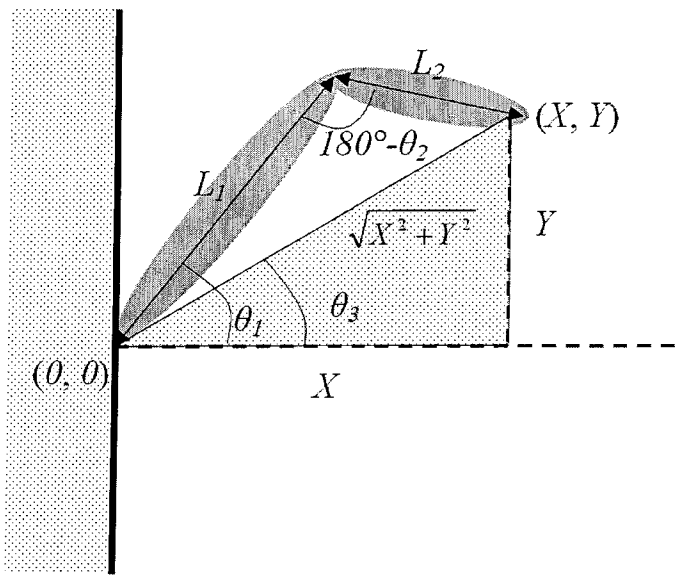


Figure A.2: The triangle used to compute θ_3 (shaded).

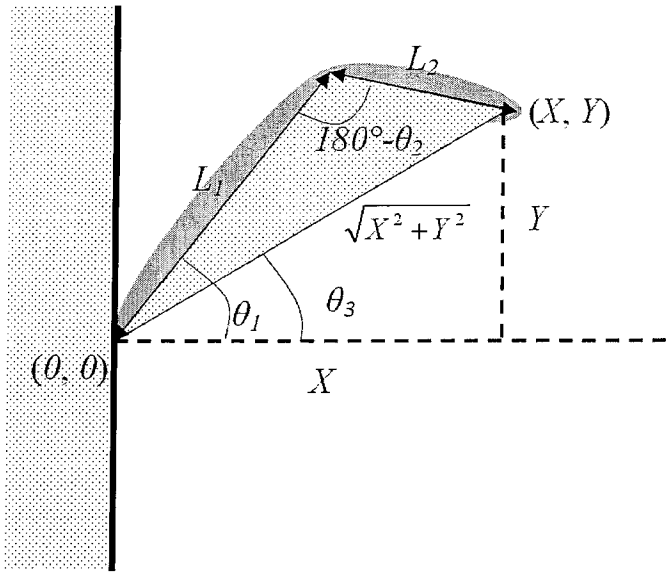


Figure A.3: The triangle used to compute $\cos(\theta_1 - \theta_3)$ and $\cos(180 - \theta_3)$(shaded).

From the law of cosines, θ_3 is given by:

$$\theta_3 = \arccos \frac{X}{\sqrt{X^2 + Y^2}}$$

Similarly considering the triangle shown in Figure A.3,

$$\cos(\theta_1 - \theta_3) = \frac{L_1^2 + X^2 + Y^2 - L_2^2}{2 \cdot L_1 \cdot \sqrt{X^2 + Y^2}}$$

and,

$$\cos(180 - \theta_2) = \frac{L_1^2 + L_2^2 - (X^2 + Y^2)}{2 \cdot L_1 \cdot L2}$$

Therefore,

$$\theta_1 = \arccos \frac{L_1^2 + X^2 + Y^2 - L_2^2}{2 \cdot L_1 \cdot \sqrt{X^2 + Y^2}} + \theta_3$$

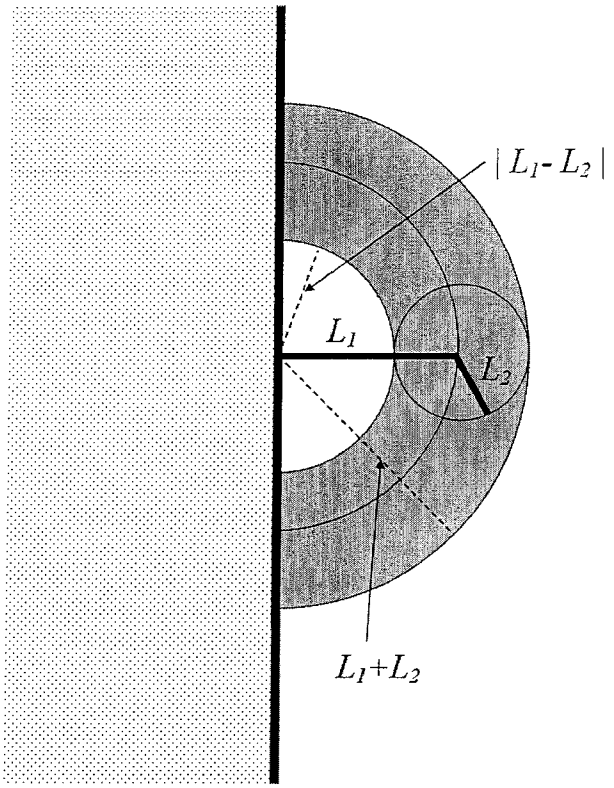


Figure A.4: Reachable area by the system

$$= \arccos \frac{L_1^2 + X^2 + Y^2 - L_2^2}{2 \cdot L_1 \cdot \sqrt{X^2 + Y^2}} + \arccos \frac{X}{\sqrt{X^2 + Y^2}}$$

and

$$\theta_2 = \arccos \frac{L_1^2 + L_2^2 - (X^2 + Y^2)}{2 \cdot L_1 \cdot L_2}$$

Note that we have assumed that the (X,Y) coordinate values given by the user are in the reachable area by this system. The minimum distance this system can reach is $|L_1 - L_2|$ and the maximum is $L_1 + L_2$. This is shown in the Figure A.4

Even for this simple mechanism, there are two correct solutions to satisfy the criteria. These two solutions are symmetric along the line between the origin $(0, 0)$ and the desired position (X, Y), as shown in Figure A.5.

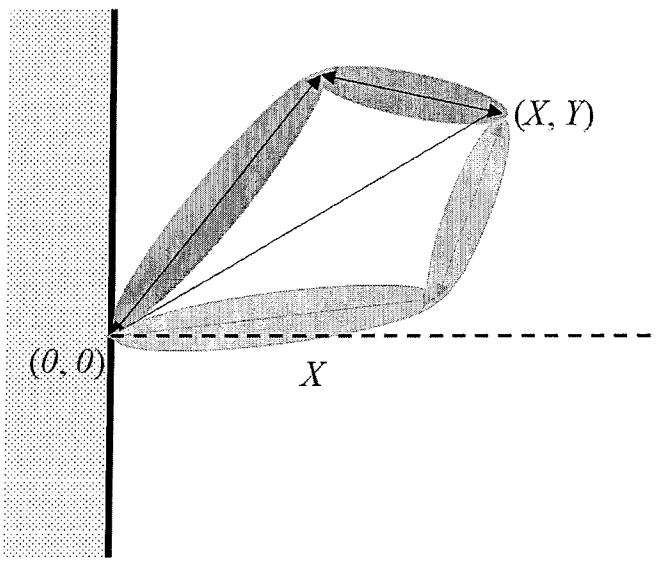


Figure A.5: Two correct solutions for the two-link system

Appendix B

Features of the Software

The complete software program of the system has not only the functionalities to perform the necessary tasks discussed in the main part of the thesis, but it also has useful features that are commonly used for research in interactive animations and computer music.

B.1 Mocap Viewer

This module reads in the motion capture data and plays it back as animation. The file format that this module is capable of handling is called *Acclaim* format ([8]). The Acclaim format requires two types of files; .asf, which contains the specification of the bone structure of the figure, and .amc, which is the joint angle trajectory data for all the frames to animate the figure. (Shown in Figure B.1)

The viewer allows the user to control many properties of the output animation (Figure B.2). These include the animation properties such as animation speed, types of primitive shapes for links, roughness of the primitives, and link size (thickness), as well as some basic on/off settings such as showing of floor/shadow/axes, applying of joint angle limits, effect of the light source. The user can also extract the hand

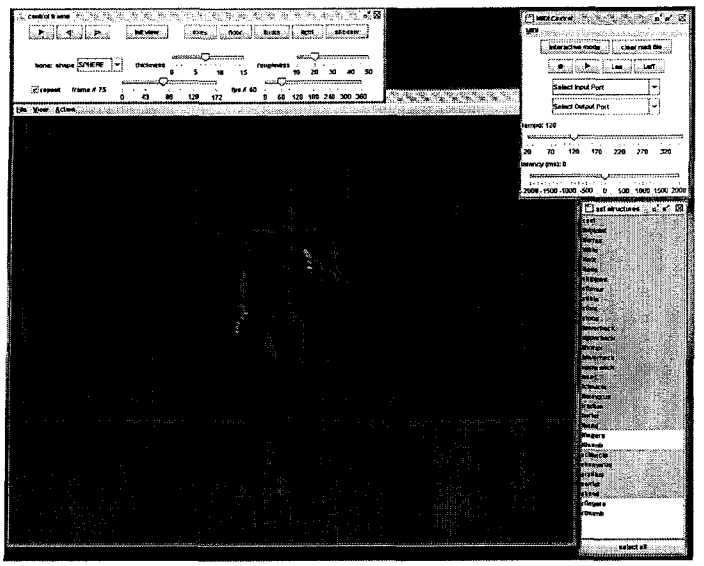


Figure B.1: Mocap Viewer

position data from data of whole body motion.

B.2 Isotrak Controller and 3D Motion Capturer

This module reads in the 12 DOF motion (6 DOF each of two sensors) data from Isotrak device in real-time, and renders the hand movements on the screen (Figure 3.2). The user can save the entire session in a file, and then play it back later.

B.3 Interactive Control of Animation in Low Dimensional Space

Another novel feature of the system is that it can synthesize new postures from the data that are obtained by using algorithms for nonlinear dimensional reduction, such as Isomap ([4]) and Locally Linear Embedding (LLE) ([26]), applied to the original high-dimensional joint trajectory space. Figure B.3 shows two windows, one with

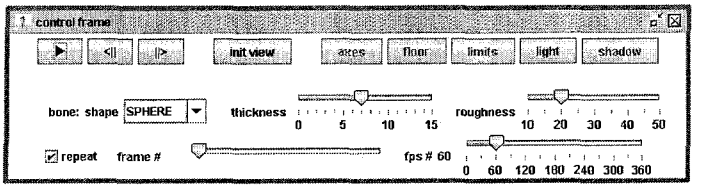


Figure B.2: Control panel for mocap viewer

the low dimensional data and the other the corresponding synthesized posture. By dragging the mouse in the low dimensional space, the system creates a posture, in real time, by linearly interpolating the joint angles of k nearest neighbours of the current location of the mouse. As the number of the nearest neighbours k increases, the flexibility of the figures' posture also increases, at the cost of *realism*.

B.4 MIDI Recorder with Animation Playback

This feature is in fact very useful as the user can record music while she watches the sample animation, thus creating the sample training sets of music-motion combinations. Although we have recorded motions in synchrony with existing music, it is possible to collect data using sample hand motions that are extracted by the feature described in Appendix B.1, and then play new music that matches the sample

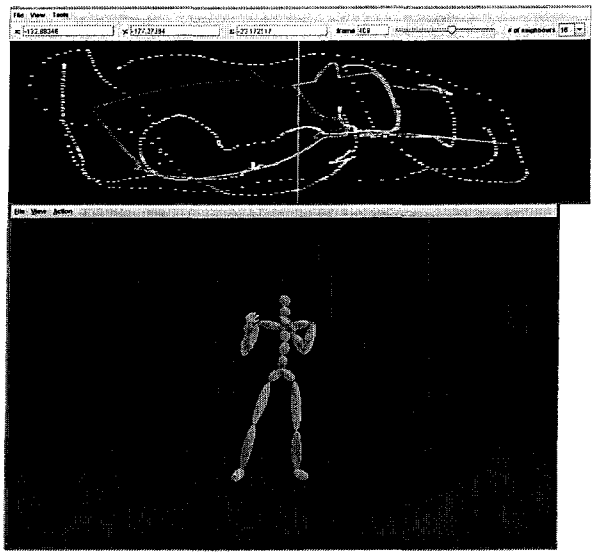


Figure B.3: Controlling animated figure in low dimensional space

motions. This module also includes the feature of MIDI playback synchronized with animation.

Bibliography

[1] G. Alankus, A. A. Bayazit, and O. B. Bayazit, "Automated Motion Synthesis for Virtual Choreography," Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, Tech. Rep. WUCSE-2004-66, 2004.

[2] E. Aldwell and C. Schachter, *Harmony and Voice Leading.* Orlando, FL: Harcourt Brace Jovanovich, Inc., 1989.

[3] W. Apel and R. T. Daniel, *The Harvard Brief Dictionary of Music.* New York, NY: Washington Square Press, 1970.

[4] M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford, "The Isomap Algorithm and Topological Stability," *Science*, vol. 295, p. 7a, 2002.

[5] C. Bishop, *Neural Networks for Pattern Recognition.* Oxford, NY: Oxford University Press Inc, 1995.

[6] M. Cardle, L. Barthe, S. Brooks, and P. Robinson, "Music-Driven Motion Editing: Local Motion Transformations Guided By Music Analysis," in *Eurographics UK Conference (EGUK)*, Leicester, UK, June 2002, pp. 38–44.

[7] G. ElKoura and K. Singh, "Handrix: Animating the Human Hand," in *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, San Diego, CA, 2003.

[8] M. S. Geroch, "Motion Capture for the Rest of Us," *J. Comput. Small Coll.*, vol. 19, no. 3, pp. 157–164, 2004.

[9] M. Goto and Y. Muraoka, "Interactive Performance of a Music-Danced CG Dancer," in *Workshop on Interactive Systems and Software (WISS) '95*, 1995.

[10] C. Greuel, M. T. Bolas, N. Bolas, and I. E. McDowall, "Sculpting 3D Worlds with Music; Advanced Texuring Techniques," in *IST/SPIE Symposium on Electronic Imaging: Science & Technology, The enginerring Reality of Virtual Reality III*, 1996.

[11] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popovic, "Style-based Inverse Kinematics," in *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, 2004.

[12] J. Heckroth, "Tutorial on MIDI and Music Synthesis," 1995. [Online]. Available: http://www.midi.org/about-midi/tutorial/tutor.shtml

[13] T.-H. Kim, S. I. Park, and S. Y. Shin, "Rhythmic-Motion Synthesis Based on Motion-Beat Analysis," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 392–401, 2003.

[14] L. Kovar, M. Gleicher, and F. Pighin, "Motion Graphs," in *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques.* New York, NY, USA: ACM Press, 2002, pp. 473–482.

[15] R. Laban, *The Mastery of Movement*, 4th ed. London: Northcote House, 1988, revised by Lisa Ullman.

[16] J. Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation," *Computer Graphics*, vol. 21, no. 4, pp. 35–44, 1987.

[17] H.-C. Lee and I.-K. Lee, "Automatic Synchronization of Background Music and Motion in Computer Animation," in *EUROGRAPHICS 2005*, vol. 24, no. 3, September 2005.

[18] S. D. Lipscomb and E. M. Kim, "An Empirical Investigation into the Effect of Presentation Mode in the Cinematic and Music Listening Experience," in *The 8th International Conference on Music Perception & Cognition*, 2004.

[19] W. T. Lytle, "Driving Computer Graphics Animation from a Musical Score," in *Scientific Excellence in Supercomputing, The IBM 1990 Contest Prize Papers*, vol. 2, Ithaca, NY, 1990, p. 644.

[20] B. Meudic, "A Causal Algorithm for Beat-Tracking," in *2nd Conference on Understanding and Creating Music*, Caserta, Italy, November 2002.

[21] M. Neff and E. Fiume, "Aesthetic Edits for Character Animation," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003.

[22] F. Pachet, "Interacting with a Musical Learning System: The Continuator," in *ICMAI '02:Proceedings of the Second International Conference on Music and Artificial Intelligence.* London, UK: Springer-Verlag, 2002, pp. 119–132.

[23] R. Parent, *Computer Animation: Algorithms and Techniques.* San Francisco, CA: Morgan Kaufmann Publishers, 2002.

[24] V. Penasse and Y. Nakamura, "Dance Motion Synthesis with Music Synchronization," in *Robotics Society of Japan 2003*, vol. 21, 2003.

[25] C. Raphael, "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models," *ieeepami*, vol. 21, no. 4, pp. 360–370, 1999.

[26] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[27] H.-H. Shih, S. S. Narayanan, and C.-C. J. Kuo, "Automatic Main Melody Eextraction from MiDI Files with a Modified Lempel-Ziv Algorithm," in *International Symposium on Intelligent Multimedia, Video Speech Processing*, 2001.

[28] S. Subramanya, M. Chowdhury, T. Pham, and S. Shankar, "A Scheme for Content Based Retrieval of Music Data in MIDI Format," in *ISCA 16th International Conference - Computer Applications in Industry and Engineering*, 2003, pp. 159–162.

[29] G. Tzanetakis, N. Hu, and R. Dannenberg, "Toward an Intelligent Editor for Jazz Music," in *Workshop on Image Analysis for Multimedia Interactive Systems*, London, UK, 2002.

[30] A. Uitdenbgerd and J. Zobel, "Melodic Matching Techniques for Large Music Databases," in *MULTIMEDIA '99: Proceedings of the Seventh ACM International Conference on Multimedia (Part 1).* New York, NY, USA: ACM Press, 1999, pp. 57–66.

[31] T. Wang, Y. Li, Y.-Q. Xu, and H.-Y. Shum, "Learning Kernel-based HMMs for Dynamic Sequence Synthesis," in *10th Pacific Conference on Computer Graphics and Applications (PG'02)*, 2002.