Learning to Synthesize Arm Motion to Music By Example

Sageev Oore
Saint Mary's University, Halifax, Canada
sageev@cs.smu.ca

Yasushi Akiyama
Saint Mary's University, Halifax, Canada
y_akiyama@cs.smu.ca

ABSTRACT

We present 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 adapting existing motion, as most music-animation systems do, our system is novel in that it analyzes both the supplied motion and music data for certain characteristics (eg. melodic contour, loudness, etc), and learns relationships between the two. 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 curves according to these constraints. A demonstration is presented showing the results on both slow and fast tunes, including an automatically-generated trio of backup dancers for a jazz standard. The system could be extended naturally to other movements beyond arm dancing.

Keywords: animation, learning, motion/music synchronization.

1 INTRODUCTION

There are a myriad ways to create character motion to music. Certain characters may have a 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. How does the nature of some of these dance motions— especially in the "background" contexts that one might want to automate— relate to musical characteristics such as loudness, pitch, and note density? There is no constant relationship, of course, but for a given context or character, there may likely be a particular connection or association. Our paper presents an initial approach towards answering this question by analyzing the characteristics of music and motion data, and then by providing a prototype tool for animators to specify some of these relationships implicitly by example.

Various work [ES03, Lyt90, LL05, CBBR02, KPS03, WLXS02, GBBM96, GM95, PN03] has been 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 for some musical phrases, 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 automatically 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 dance styles. 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. In Section 9 we demonstrate this with automatically generated arm motions for a set of backup dancers for a jazz standard.

2 RELATED WORK

The issue of synchronizing animation to music has been addressed by a number of researchers. Kim et al. [KPS03] and Alankus et al. [ABB04] created systems using motion graphs [KGP02] and beat analysis to synthesize a motion from existing sample motions synchronized to background music. The systems by Cardle et al. [CBBR02] and Lee and Lee [LL05] synchronize motion curves by locally modifying motions using perceptual cues obtained from the music. Lytle's [Lyt90] system creates animation of musical instruments from orchestrated MIDI music. Goto and Muraoka [GM95] have multiple musicians, each assigned to control specific features of a single animated character. Penasse

WSCG 2006 conference proceedings, ISBN 80-86943-03-8
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.

and Nakamura [PN03] used the analog sound signal and adjusted key frames so that they fall on the beats of music. In these systems, using explicit mappings of features between music and animation seems like a standard approach. ElKoura and Singh [ES03] use input music data in an augmented tablature notation for guitar. Their system solves the problem of multiple concurrent reaching tasks by finding the likely hand configurations in the realistic hand configurations obtained from example motions, and minimizing a cost function for hand motions. Wang et al [WLXS02] predict the joint angle trajectories to create conducting motions by using kernel-based Hidden Markov Models. Learning from the music with the time signature that has the same or similar musical accents and relying on beat information provided in the MIDI file may limit its flexibility of creating motions responding to music that is played more freely.

3 SYSTEM OVERVIEW AND OUTLINE

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; scalability is discussed in Section 10.

We now give an outline of this paper, along with an overview of the two phases, in which our system operates.

1) Training Phase: Fig 1

A set of examples is provided consisting of hand-motion recorded in synchrony with music in MIDI format (described in Section 4). Based on our motion model, the motion data is analyzed for certain characteristics: distance-from-body, amplitude, centres-of-motion (described in Section 5). Likewise, the music data is analyzed for characteristics such as loudness and note density (Section 6). A relationship between the input and output characteristics is learned so as to be able to predict a distribution over likely output motion characteristics given music input characteristics (Section 7).

2) Generative Phase: Fig 2

New music input is provided. The system analyzes it as before, and uses the learned model to stochastically generate a set of motion characteristics. A generative process is applied to create motion curves for the final animation (Section 8).

4 DATA COLLECTION

Music Data .

Music is input in a standard MIDI file format. Each note event contains information pertaining to pitch, loudness, time, and a NOTE ON and NOTE OFF signal, indicating whether the note is starting or stopping. The MIDI format we use is Type 1, which allows a file to have multiple tracks. This is convenient as it lets us focus directly on processing individual lines (eg. melody, bass) and avoid the task of separating a given score into parts [SNK01].

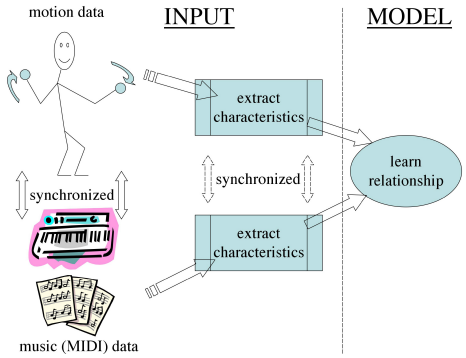

Figure 1: Training Phase: Synchronized examples of motion and music are provided to the system.

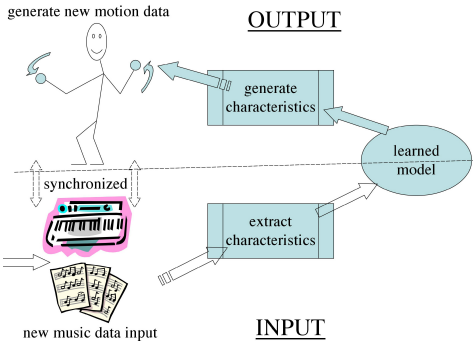

Figure 2: Generative Phase: New music is now provided, and using the learned model, motion is now the output rather than input (as indicated by the reversed arrows).

Motion Capture Two 3D motion capture devices are used to collect the sample motion data. The dancer familiarizes herself with the music prior to the motion recording session. We acquire 3 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 motion examples are representative.

5 DANCE MOTION ANALYSIS

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

5.1 Spatial Characteristics

An important visible feature of the arm motion is *extent*: the distance of the hands from the body [Lab88, NF03].

We first compute $D_k(t)$, the Euclidian distance of the hands from the root node at time t, for motion B_k, where the root can simply be approximated by a point near the middle the torso for this purpose. 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. Figure 4 shows $dist_k(t)$ of four sample captured dance sequences to the same piece of music M.

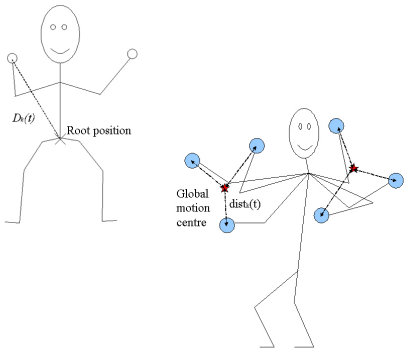

Figure 3: Root position, $D_k(t)$, the global motion centre, and $dist_k(t)$

The motions tend to be comprised of oscillating segments, corresponding to back-and-forth (or side-to-side etc) movement of the hands. We model such oscillations by estimating, for each time frame, an approximate centre of extent 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 5.

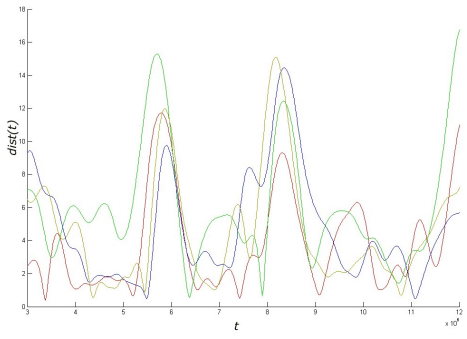

Figure 4: This illustrates 4 different sets of $dist(t)$ for a single music track. Each colour corresponds to one recorded motion. While each curve is different, they share certain characteristics, and those characteristics are what we would like to learn. E.g. the two largest peaks happen around the 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.

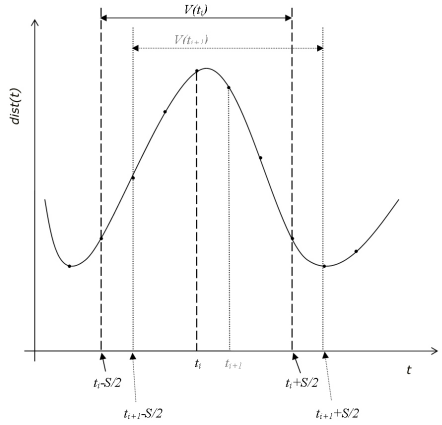

Figure 5: For each frame, $m_k(t_i)$ and $v_k(t_i)$ are computed in the moving window $V(t_i)$.

For each window $V(t_i)$ we compute the mean hand distance (or "center of oscillation"):

$$m_k(t_i) = \frac{1}{S+1} \sum_{t_i \in V(t_i)} (dist_k(t_i)) \qquad (1)$$

Similary, we compute the variance for window $V(t_i)$:

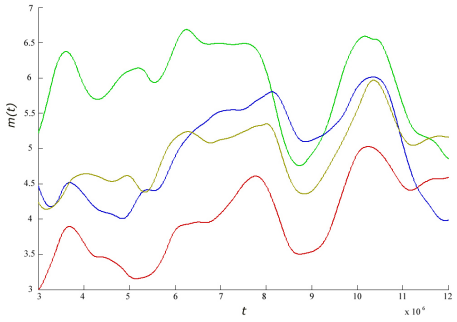

Figure 6: The mean hand distance $m_k(t)$. The same colors correspond to the same motions in the previous and next figures.

$$v_k(t_i) = \frac{1}{S+1} \sum_{t_i \in V(t_i)} (dist_k(i) - m_k(t_i))^2 \qquad (2)$$

The square-root of this value, $\sqrt{v_k(t_i)}$, relates to the amplitude of the oscillations, as it tells us how far the hand moves from the local centre of oscillation. Figures 6 and 7 show $m_k(t)$ and $v_k(t)$, respectively, for clips from a set of motions all corresponding to one piece of music. 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

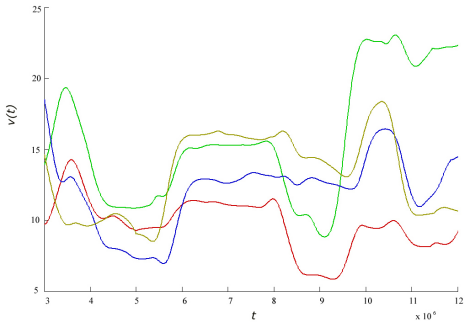

Figure 7: The variance $v_k(t)$

wherein the m_k are relatively tight. Similarly, looking at the estimated amplitudes over the same motion set, more coherence is evident in some areas than in others. This suggests that, if we are to predict $m(t)$ and $v(t)$ for some new piece of music, and if we want to be able to create multiple animations sharing "characteristics", it is not enough to simply predict a single value for $m(t)$, but rather 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 8). Similarly we will predict $\mu_v(t)$ and $\sigma_v(t)$. In order to learn to predict these values for a new piece of music, we first estimate them for the given data:

$$\mu_m(t_i) = \frac{1}{N}\sum_{k=1}^{N}(m_k(t_i)) \tag{3}$$

$$\sigma_m(t_i) = \frac{1}{N-1}\sum_{k=1}^{N}(m_k(t_i) - \mu_m(t_i))^2 \tag{4}$$

and

$$\mu_v(t_i) = \frac{1}{N}\sum_{k=1}^{N}(v_k(t_i)) \tag{5}$$

$$\sigma_v(t_i) = \frac{1}{N-1}\sum_{k=1}^{N}(v_k(t_i) - \mu_v(t_i))^2. \tag{6}$$

In Section 6 we will describe some of the music characteristics that may drive $m(t)$ and $v(t)$, but first we consider some spatial characteristics of the motion.

5.2 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, and we would like to model these *peak points* both in time and in space. The amplitude and oscillation centre predictors $(\mu_v, \sigma_v, \mu_m, \sigma_m)$ 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

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

where Δt is a small constant $\approx 0.1sec$ to allow for slight temporal discrepancies between the data and the music. Combining $peak_k(t)$ for each B_k, where $k = 1, \ldots, N$, we obtain an estimate, for a given piece of music, of the independent probability of a peak event occuring at each time slice. 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.

6 MUSIC ANALYSIS

Before we can apply a machine learning tool to predict 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.

6.1 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. Thus, we consider musical progression at several time-scales, by computing 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 Fig 8.

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.

6.2 Melodic Contour Information

Melodic contour refers to the rise and fall of the pitches. If a passage is played slowly and staccato, then most of

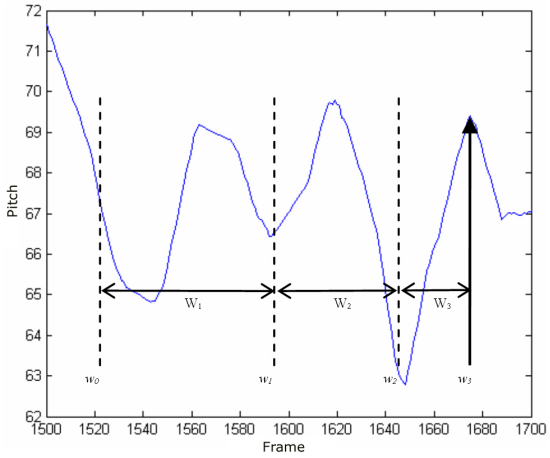

Figure 8: The windows W_1, W_2 and W_3 for multiple scales.

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. 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, 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* [Pac02]. The threshold is typically based on the average inter-onset time of all the notes.

6.3 Loudness Characteristics

Loudness information is processed differently from the pitch information. Firstly, we assume that when a note is played, it decays continuously 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. For non-decaying instruments, we have found it to be sufficient to simply keep loudness constant while the key is being pressed.

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

$$l_j(t_i) = l_j(t_{j0})\kappa^{(t_i - t_{j0})} \tag{7}$$

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 ($\kappa < 1$). 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. Figure 9 shows the result of the cumulative 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.)

6.4 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:

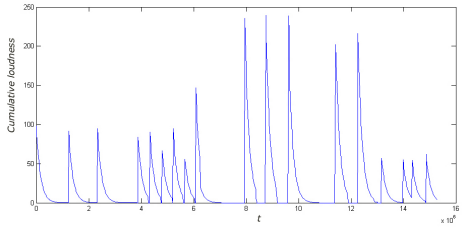

Figure 9: Cumulative loudness processed from MIDI data

$$\rho_k(t_i) = \frac{N_k(t_i)}{\Delta T_k(t_i)} \tag{8}$$

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.

7 LEARNING CHARACTERISTIC RELATIONSHIPS

Once the various characteristics have been selected and extracted, a standard neural network model, with tanh functions in the hidden layer, is used to learn relationships based on the provided example data. Other nonlinear models could be also be used.

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 above, 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 6. One model is trained to predict the probabilities $peak(t_i)$ as described in Section 5 given input $u(t_i)$. The other model learns to estimate the spatial characteristics of the distribution, also described in Section 5.

The temporal model has 22 input parameters (Table 1) and 1 output parameter (Table 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 2) and 4 output parameters, $\mu_m(t)$, $\sigma_m(t)$, $\mu_v(t)$, and $\sigma_v(t)$ (Table 3), which are used to form distribution functions. We next explain how these output parameters are used to generate motion curves.

8 GENERATING MOTION CURVES

The generative procedure begins by analyzing a new musical input track, and using the $peak(t)$ estimator (predicted by a sigmoidal unit, which can be interpreted as a binary probability value) 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 based on the human physical ability. The maximum values of velocity and acceleration are obtained from the example motions. If candidate motion peaks are detected to create movements with higher values than these minimum values, one of them is excluded from the list of peaks. A common previous approach to create animation to music is to assume the beat information is embedded in the MIDI file [KPS03, WLXS02], which also requires an unwavering tempo throughout the piece, or by using beat extraction algorithms such as [Meu02]. 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 that has 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 of music itself is equally important.

Once the peaks have been selected, then for each peak point at time t_i, the system computes an exact location of the hand as follows. Let $\mu_m(t_i), \sigma_m(t_i), \mu_v(i)$, and

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

Table 1: Input parameters for the temporal predictor

#	Parameter
1 - 20	Pitch/velocity samples
21 - 23	Note density in each window
24 - 29	Mean/variance of pitch in each window
30 - 35	Mean/variance of loudness in each window
36 - 37	Flags to indicate peaks in pitch/loudness
38 - 39	Time duration of peaks
40	Flags to indicate phrase endings
41	Time duration of phrases

Table 2: Input parameters for the spatial predictor

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

Table 3: Output parameters

$\sigma_v(i)$ be predicted by the model according to the music characteristics $u(t_i)$ at that time. A local motion centre and amplitude for $dist(t_i)$ are chosen by sampling a gaussian distribution with these parameters. 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 10), we currently randomly set several primary axes of motion $\theta_1, \theta_2, \ldots, \theta_r$ for each character's hands as *mean* directions, and then for any given peak at t_i, we randomly choose one of them, k and sample from a normal distribution centered at θ_k. Together, the axis and distance specifies a point relative to the character.

A quartic polynomial is used to interpolate between the generated peak points. First derivatives are constrained to be 0 at the peaks, and second-derivative continuity is enforced at the join points.

The orientation angles at the characters' links are computed using an IK algorithm.

9 RESULTS

The model was trained on a data set consisting of several pieces of music, each one a few minutes long, together with multiple corresponding samples of recorded hand motion for each piece. Specifically, the average training music sample had 3000-4000 frames, of which we sampled every 10 frames, and used 8 recorded motions per piece, giving roughly 2400 data samples per piece. New pieces of music were then given to the system, which were automatically analyzed, and distribution parameters were predicted for the corresponding motions. Sampling from these parameters and interpolating, as described in above, allowed the creation of multiple motions, sharing certain characteristics, while still differing in the details due to the stochastic nature of the generative process (see the accompanying videos, with sample frames shown in Figure 10). The musical pieces can range from being highly rhythmic, to having rhythmic patterns that shift from slow and rubato to fast, demonstrating that the characters respond appropriately to these complex changes. Testing the system with various pieces shows that the motion can relate to the music even when there is no steady tempo, while at other times it can correspond to certain clear accents in the music. Note that, for each test, the motions of only one dancer were used, so the results corresponded to the patterns specific to that set. This is, indeed, the goal of the system: to learn a mapping based on the motions of a particular choreographer; the system is not intended to learn a general "all-purpose" mapping.

10 CONCLUSION AND FUTURE WORK

We have demonstrated a novel approach to creating animated motion to new musical scores for virtual actors. We have proposed a set of key features of music and motions, and shown how to extract these features. Rather than adapting existing recorded dance motion, we use recorded motions synchronized to music soundtracks, and automatically analyze salient *characteristics* of both the motion and the music. A non-linear parametric model is then used to learn the relationship between these characteristics by example, rather than by requiring an animator to specify mappings explicitly. Once the training data has been specified, then the subsequent musical analysis and stochastic generative process are automated procedures, and thus can be applied in any context where an autonomous animated agent is acting.

Finally, while our present work focuses on animating motion of hand and arm, the same ideas could be extended to include a range of body motion, such as leg and torso motion. In this case, unlike the somewhat free movements of hands, additional constraints would need to be incorporated (e.g. ground contact) in order to maintain physically plausible postures. We expect that further extensions of this sort will contribute significantly to the naturalness of the final synthesized motion.

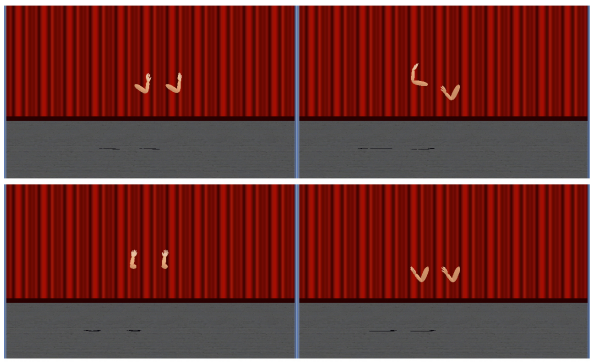

Figure 10: Screenshots of hand motions created stochastically for a new piece of music (see accompanying video).

ACKNOWLEDGEMENTS

We would like to thank Dirk Arnold and the anonymous reviewers.

REFERENCES

[ABB04] Gazihan Alankus, A. Alphan Bayazit, and O. Burchan Bayazit. Automated motion synthesis for virtual choreography. Technical Report WUCSE-2004-66, Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, 2004.

[CBBR02] Marc Cardle, Loic Barthe, Stephen Brooks, and Peter Robinson. Music-driven motion editing: Local motion transformations guided by music analysis. In *Eurographics UK Conference (EGUK)*, pages 38–44, Leicester, UK, June 2002.

[ES03] George ElKoura and Karan Singh. Handrix: Animating the human hand. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, San Diego, CA, 2003.

[GBBM96] Christian Greuel, Mark T. Bolas, Niko Bolas, and Ian 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.

[GM95] Masataka Goto and Youichi Muraoka. Interactive performance of a music-danced cg dancer. In *Workshop on Interactive Systems and Software (WISS) '95*, 1995.

[KGP02] Lucas Kovar, Michael Gleicher, and Frdric 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 Press.

[KPS03] Tae-Hoon Kim, Sang Ill Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*, 22(3):392–401, 2003.

[Lab88] Rudolf Laban. *The Mastery of Movement*. Northcote House, London, 4 edition, 1988. Revised by Lisa Ullman.

[LL05] Hyun-Chul Lee and In-Kwon Lee. Automatic synchronization of background music and motion in computer animation. In *EUROGRAPHICS 2005*, volume 24, September 2005.

[Lyt90] Wayne T. Lytle. Driving computer graphics animation from a musical score. In *Scientific Excellence in Supercomputing, The IBM 1990 Contest Prize Papers*, volume 2, page 644, Ithaca, NY, 1990.

[Meu02] Benoit Meudic. A causal algorithm for beat-tracking. In *2nd Conference on Understanding and Creating Music*, Caserta, Italy, November 2002.

[NF03] Michael Neff and Eugene Fiume. Aesthetic edits for character animation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003.

[Pac02] François Pachet. Interacting with a musical learning system: The continuator. In *ICMAI '02:Proceedings of the Second International Conference on Music and Artificial Intelligence*, pages 119–132, London, UK, 2002. Springer-Verlag.

[PN03] Vincent Penasse and Yoshihiko Nakamura. Dance motion synthesis with music synchronization. In *Robotics Society of Japan 2003*, volume 21, 2003.

[SNK01] H.-H. Shih, S. S. Narayanan, and C.-C. J. Kuo. Automatic main melody extraction from midi files with a modified lempel-ziv algorithm. In *International Symposium on Intelligent Multimedia, Video Speech Processing*, 2001.

[WLXS02] Tianshu Wang, Yan Li, Ying-Qing Xu, and Heung-Yeung Shum. Learning kernel-based hmms for dynamic sequence synthesis. In *10th Pacific Conference on Computer Graphics and Applications (PG'02)*, 2002.