EUROGRAPHICS 2005 / M. Alexa and J. Marks
(Guest Editors)

Volume 24 (2005), Number 3

Automatic Synchronization of Background Music and Motion in Computer Animation

Hyun-Chul Lee † and In-Kwon Lee ‡

Dept. of Computer Science, Yonsei University, Korea

Abstract

We synchronize background music with an animation by changing the timing of both, an approach which minimizes the damage to either. Starting from a MIDI file and motion data, feature points are extracted from both sources, paired, and then synchronized using dynamic programming to time-scale the music and to timewarp the motion. We also introduce the music graph, a directed graph which encapsulates connections between many short music sequences. By traversing a music graph we can generate large amounts of new background music, in which we expect to find a sequence which matches the motion better than the original music.

Categories and Subject Descriptors (according to ACM CCS): J.5 [Computer Applications]: Arts And Humanities

1. Introduction

Background music (BGM) enhances the emotional impact of computer animation. Here, BGM means any kind of music played by one or more musical instrument, and should be distinguished from sound effects, which are usually short sounds made by natural or artificial phenomena. This paper is about synchronizing BGM with motion in computer animation. Well-synchronized BGM helps to immerse the audience in the animation, and can also emphasize features of characters' motions. Successful examples of synchronizing animation with excellent BGM can, for example, be found in many Disney animations.

In many large scale animations, the voice track is often pre-recorded before the animation. A particular piece of music can also sometimes give birth to ideas for the animation (e.g., Fantasia from Disney). However, in most cases, the picture comes first and music and sound effects are usually added once the picture is completed [Lay98]. In order to obtain music that synchronizes with a particular animation, we can hire a composer and musicians. This approach is, of course, expensive and it is more common to fit existing recorded music to the animation after it has been produced. But, it is not simple to find a piece of music that matches the characters' motion. One may have to go through several scores, and listen to many selections in order to find a suitable portion for a given scene. It is still hard to match the music with every significant feature of the motion.

In computer animation, it is possible to generate or modify the animation or the music, by speeding up or slowing down passages, so that the feature points of each occur at the same time. Recent studies on synchronization of music and motion in computer animation can be categorized by how this is done. In some studies [CBBR02,KPS03] the motion data has been synthesized or edited to match the music. Windows media player visualization can be put into this category, since it uses an audio signal to generate an abstract animation. In other studies [NKH*94, DKP01, HFGL95] the opposite approach has been taken, and music has been synthesized to match the completed animation. An alternative idea, modifying existing music to match the scene, has received little attention.

Our goal is to generate synchronized animation by editing both music and motion at the same time, with the aim of avoiding changes that are not so large that they compromise the integrity of each component. Our system analyzes MIDI data and motion signals and finds the optimal matches between features of the music and the animation using DP (Dynamic Programming). This is followed by modification of the music and motion timewarping of the animation, so as to match the features while preventing noticeable dam-

† kennyd@cs.yonsei.ac.kr
‡ iklee@yonsei.ac.kr

age to either music or motion. This framework uses elements of Cardle's analysis techniques [CBBR02] which allows the user to try out numerous combinations of music and motion.

We also introduce a new music rearrangement technique, which uses a directed graph called a Music Graph, which is automatically constructed by our system. A music graph encapsulates connections between several music sequences. Music can be generated by traversing the graph and then smoothing the resulting melody transitions. The music graph can be utilized in our synchronization system to generate new tunes that will match the motion better than the original music.

Our system works well for (motion, music) pairs with similar moods, such as (ballet, waltz) and (dance, pop). However, music without strong beats such as classical music or ballad gave us better results after the time-scaling. One of the limitations of this work is that users have to select the appropriate features for the motion and music case by case. For example, foot-step feature would not well represent a skating motion, thus other features like effector velocity must be chosen. We also assume that the music clips that are used to construct the music graph should have the similar rhythmic styles to avoid abrupt rhythmic variation throughout the generated tune.

The remainder of this paper is organized as follows. In Section 2 we describe related work. In Section 3 we describe the overall structure of our system. In Section 4 we set out a general framework for extracting features from motion and music. In Section 5 we describe how these features are matched using DP matching, and how the music and motion should be synchronized. In Section 6 we describe the process of music graph construction, and show how the music graph can be used as a synchronization method. In Section 7 we look at some results, and we conclude the paper in Section 8 with some ideas for future work.

2. Related Work

There has been a lot of work on synchronizing music (or sound) with animations. In essence, there are two classes of approach, depending on whether the music is altered or generated to match the animation, or vice versa.

One approach is to use musical parameters to control motion. Greuel et. al [GBBM96] used MIDI notes to control objects in a 3D virtual environment, and Hwang and Gerard [HG99] represented music progressions as orbiting planets. Cardle et. al [CBBR02] edited motion curves to match musical features, obtained by analysis of both MIDI data and audio signal. Our system continues this approach with the inverse, because we modify the music as well as the motion.

A great deal of work has also been done on controlling music or generating sound from an existing animation [HFGL95, MH95]. Cardle et. al [CBBJR03] introduced a system that automatically generates sound-tracks for a given movie based on the sound-tracks of other movies. Nakamura et. al [NKH*94] generated background music and sound effects based on the mood and motion of animated characters. Several studies on the automatic generation of sound effects from physical models have achieved realistic association between sound and motion [DYN03, DP96, DKP01, OCE01].

While the techniques described above were focused on the generation of music or sound, the modification of existing music was explored by Tadamura and Nakamae [TN98]. They adjusted the tempo of the music to match the length of the scene, but took little account of features inside that interval.

As regards feature matching Cardle's approach [CBBR02] is closest to ours. Cardle extracted low-level and high-level musical features from a MIDI stream, such as note pitch, note velocity (volume) and chord progression. They were then used to edit motion curves in the animation. We extend Cardle's work by considering music modification as well as motion editing in matching the extracted features.

Yoo et. al [YLC04] suggested a method to generate long background music sequences from a given music clip using a music texture synthesis technique. We introduce our music graph as an alternative approach to music texture synthesis. The idea of the music graph is related to motion graphs for motion synthesis [KGP02, AF02, LCR*02].

3. System Overview

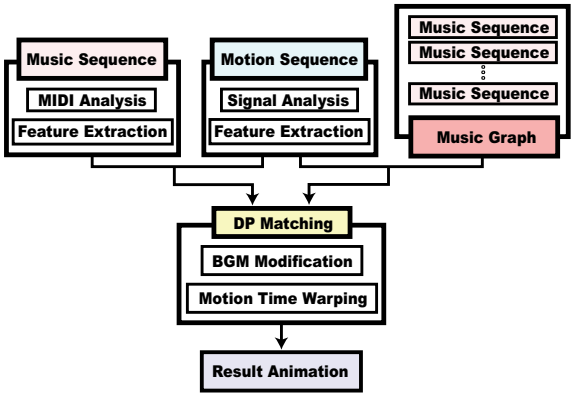

Figure 1: *System Overview.*

Our system consists of four distinct modules:

- Music analysis.
- Motion analysis.
- DP matching.
- Music graph.

Figure 1 shows the relationship between the system components.

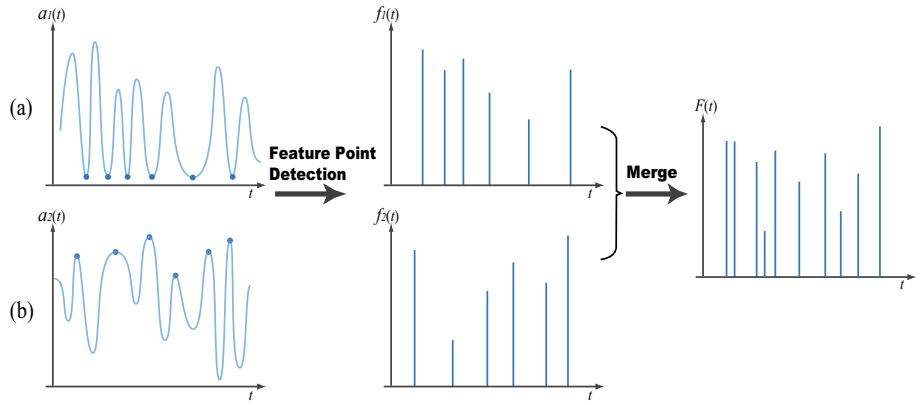

Figure 2: *An example of motion curves and feature point detection: (a) footstep points; (b) arm-swinging points using KCS.*

The music analysis module extracts musical parameters such as note pitch, velocity (volume) and duration from the MIDI score, and hence determines the feature points of the input music sequence. The motion analysis module analyzes various features of the input motion signal, such as end-effector velocity, kinetic energy, footstep and arm-swing, and determines feature points from these data.

The inputs to the DP matching module are various combinations of the outputs from the music and motion analysis modules. By applying DP matching to the music and motion feature points, a partial sequence is extracted from the input music, with the minimum feature value difference and feature time distance. The music and motion features are also paired during this procedure. The system ignores feature pairs that will cause noticeable damage to the music and then locally scales the music so that both features occur at the same time. The feature pairs that were ignored during this procedure are now reconsidered and matched through motion timewarping. The DP matching module then outputs the final animation and its BGM.

The music graph module constructs a music graph that encapsulates connections between several music sequences, which are determined by similarities in chord progression. Then the system traverses the music graph repeatedly so as to achieve a new music sequence which matches the motion better than the original music.

Details of each module will be provided in the following sections.

4. Feature Extraction

We will represent a motion sequence in the time interval $[\ t_b,\ t_e\ ]$ as a multidimensional curve, $A(t) = (a_1(t),\ a_2(t), \cdots, a_n(t))$, $t_b \leq t \leq t_e$, which is called a *motion curve*. Each of the component functions $a_i(t)$ represent a quantitative or qualitative property of the motion sequence, such as:

- Transformation (translation, rotation, scale, and other kinds of deformation) of any parts of the actors.
- Material (such as color or texture) changes of parts of the actors.
- Mission or feelings of the actors (in a script-based animation system).
- An arbitrary function specified by the user.

Like the animation, the BGM can also be represented by a multidimensional *BGM curve*, which we will write $M(s) = (m_1(s),\ m_2(s), \cdots, m_m(s))$, $s_b \leq s \leq s_e$. Each component function $m_i(s)$ represents any quantitative or qualitative property of the music, such as:

- Note pitch, duration, or velocity (volume).
- Inter-onset interval (duration between consecutive notes).
- Register (interval between highest and lowest pitches).
- Fitness for a fixed division (see Equation 4).
- Chord progression.
- Feeling of the music.
- An arbitrary function specified by the user.

Collecting such samples from the BGM is not easy when its source is analogue or digital sound data. A MIDI file makes extraction of the necessary data much easier. In the following subsections we will look at some examples of how feature points are obtained from the motion and BGM curves.

4.1. Motion Feature Detection

The range of human movements is extremely wide, and the features of a motion depends on its characteristics. In general, it is not easy to capture motion features algorithmically; but we have found some general motion analysis techniques that are useful for feature detection.

One of the most obvious motion features are footsteps, especially in dance or marching motions. In our work, we not only find the times of feature points, but also give scores to

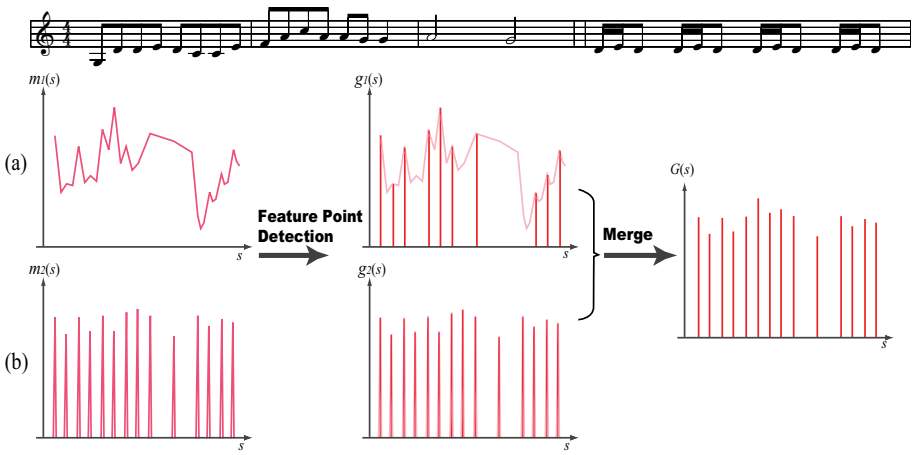

Figure 3: *An example of BGM curves and feature point detection: (a) note velocity (volume); (b) fitness to quarter note.*

the feature points to emphasize more important features. For example, if $a_1(t)$ is the vertical position (y-coordinate) of the foot of a character, the local minimum points are the footstep points, as shown in Figure 2(a). We can get these points by calculating the zero crossing points of the first derivative $a_1'(t)$ where the second derivative $a_1''(t)$ is larger than zero. The score of the footstep point is $a_1''(t)$ which indicates difference in the velocity of the foot before and after the footstep. In other words, measuring how fast the foot hits and leaves the ground determines the score of this type of feature: in general, more dynamic movements make more significant features.

To find motion features for arms, we use the KCS (Kinematic Centroid Segmentation) technique [JM03] which can detect the extreme positions of arm swings. Using KCS, a motion curve $a_2(t)$ can be derived as follows:

$$a_2(t) = (C(t) - B(t))^2, \tag{1}$$

where $B(t)$ is the shoulder position at time t and $C(t)$ is the average of the shoulder, elbow and wrist positions at time t. Then the arm feature points will correspond to the local maximum of $a_2(t)$, which are the points when the arm is at one end of the swing. The derivative $a_2''(t)$ determines the score of the feature which describes how fast the arm reaches and leaves the extreme positions.

Many other feature extraction methods are available. We can extract features from local minima of the end-effector velocity or from local maximum of a kinetic energy function. Also we can use the Z-function [FMJ02] segmentation technique to set feature points at instances where the motion stops. Particular feature scoring methods can be set up by the user.

The motion curves $a_i(t)$ are converted into feature functions $f_i(t)$ that represent the scores of the candidate feature points. For example, each feature function $f_1(t)$ in Figure 2(a) can be defined as follows:

$$f_1(t) = \begin{cases} a_1''(t) & \text{if } a_1'(t) = 0 \text{ and } a_1''(t) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Finally, the motion feature function $F(t)$ can be computed by merging the normalized component feature functions $\hat{f}_i(t)$:

$$\hat{f}_i(t) = \frac{f_i(t)}{max(f_i(t))} \ (i = 0, 1, \cdots, n). \tag{3}$$

The user can select either one feature function or merge several feature functions together to give an overall representation of the original motion.

4.2. Music Feature Detection

Detecting music features is relatively straightforward compared to motion feature detection but the method used still needs to suit the characteristics of the music. As already mentioned, low-level data such as note pitch and note velocity (volume) can be extracted from MIDI files and these data can be used to analyze higher-level data such as chord progressions [Row04]. These data are represented in separate BGM curves that can either be *continuous* or *bouncing*. The note velocity (volume) $m_1(s)$ in Figure 3(a) is a continuous function which represents the change in note volume through time. By contrast, the fitness function $m_2(s)$, which determines whether a note is played near a quarter note (a note played on the beat), is of bouncing type. For example, the fitness function $m_2(s)$ can be defined as follows:

$$m_2(s) = \begin{cases} |s - s_k| & \text{if a note exist in } [s_k - \varepsilon, s_k + \varepsilon] \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

$$s_k = k\Delta s, \ \Delta s = \frac{s_e - s_b}{4N_m} \ (k = 0, 1, 2, \cdots), \tag{5}$$

where ε is a small tolerance, and N_m is the number of bars of the BGM; thus Δs is a length of a quarter note. (Note that the time signature of the BGM in Figure 3 is $\frac{4}{4}$). Feature points can be extracted from the BGM curves in various ways depending on the kind of data we are dealing with. For example, we may consider the local maximum points of the note velocity (volume) curve to be the features of this curve, because these are notes that are played louder than the neighboring notes.

The BGM curves $m_i(s)$ are converted into the feature functions $g_i(s)$, as shown in Figure 3. In some cases, the fitness function can be used directly as the feature function since it represents discrete data. For example, $m_2(s)$ is inverted in order to represent how well the note fits a quarter note:

$$g_2(s) = \begin{cases} \dfrac{1}{m_2(s)} & \text{if } m_2(s) \neq 0 \\ 0 & \text{otherwise} \end{cases}, \quad (6)$$

Finally, the BGM feature function $G(s)$ can be computed by merging the normalized component feature functions. The user can select either one feature function or merge several feature functions together to form the final representation of the music.

5. Synchronization Using DP Matching

DP matching is a well known method for retrieving similarities in time series data such as speech or motion [AFO03, HDL02, SDO*04, YT99]. Using DP matching, we can find the partial sequence from the given music which best matches the motion sequence, while also pairing the feature points from the motion and music. To synchronize the music and motion, we first modify the music using feature pairs that will not cause severe damage to the music. Then we timewarp the motion using the remaining feature pairs. Figure 4 shows the process of synchronization using DP matching.

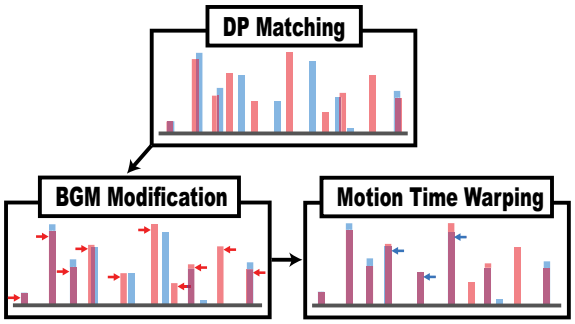

Figure 4: *Synchronization using DP matching. The red lines indicate BGM feature points and the blue lines indicate motion feature points.*

5.1. DP Matching

The DP matching method does not require the motion and music to be of the same length. However, we will assume that the music sequence is longer than the motion so that we are sure there is enough music for the motion. Following Section 4, we assume that $F(t)$, $t_b \leq t \leq t_e$, and $G(s)$, $s_b \leq s \leq s_e$, are the feature functions for the motion and music respectively. For DP matching, we use t_i, $i = 1,...,T$, and s_j, $j = 1,...,S$, which consist, respectively, of T and S sampled feature points of $F(t)$ and $G(s)$, and which satisfy $F(t_i) > 0$ and $G(s_j) > 0$, for all i and all j. Note that we place default sample feature points at the boundary of each feature function, such that $t_1 = t_b$, $t_T = t_e$, $s_1 = s_b$, and $s_S = s_e$. The distance $d(F(t_i),\ G(s_j))$ between a motion feature point and a BGM feature point can be given by the following formula:

$$d(F(t_i),\ G(s_j)) = c_0(F(t_i) - G(s_j))^2 + c_1(t_i - s_j)^2, \quad (7)$$

where c_0 and c_1 are weight constants that control the relative influence of the score difference and the time distance. The DP matching method calculates $d(F(t_i),\ G(s_j))$ using a matching matrix $q(F(t_i),\ G(s_j))$, of size $T \times S$. The matching matrix is calculated as follows:

$$q(F(t_1),\ G(s_j)) = d(F(t_1),\ G(s_j)) \quad (j = 1,...,S) \quad (8)$$

$$q(F(t_i),\ G(s_1)) = d(F(t_i),\ G(s_1)) + q(F(t_{i-1}),\ G(s_1)) \quad (i = 2,...,T) \quad (9)$$

$$q(F(t_i),\ G(s_j)) = d(F(t_i),\ G(s_j)) + min \begin{pmatrix} q(F(t_{i-1}),\ G(s_j)) \\ q(F(t_{i-1}),\ G(s_{j-1})) \\ q(F(t_i),\ G(s_{j-1})) \end{pmatrix} \quad (i = 2,...,T,\quad j = 2,...,S) \quad (10)$$

$$D(F,\ G) = min\{q(F(t_T),\ G(s_j)) \mid 1 \leq j \leq S\}. \quad (11)$$

Here $q(F(t_T), G(s_j))$ is the total distance between the motion feature point sequence F and the partial BGM feature point sequence of G, when $F(t_T)$ matches $G(s_j)$. Moreover, $D(F,G)$ is the total distance between F and the partial sequence of G starting from s_1. In order to find the op-

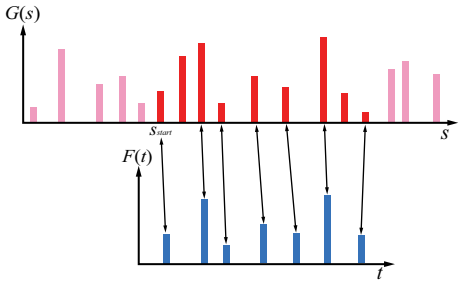

Figure 5: *Feature point paring through DP matching.*

timal match, we increase the starting time of G from s_1 to s_{S-T} and calculate the matching matrix again until we get the minimum value of $D(F,G)$. This dynamic programming algorithm naturally establishes the optimal matching pairs of motion and music feature points with time complexity $O(N^3)$, where $N = \max(T,S)$. Since we are usually dealing with short motion clips which correspond to scenes in computer animation, and different scenes usually have different background music, the time complexity $O(N^3)$ seems to be acceptable (see the timing results in Section 7). Figure 5 shows a set of feature pairs obtained by DP matching.

5.2. Music Modification and Motion Timewarping

Now we synchronize the feature points by time-scaling the music to match the feature pairs obtained from DP matching. First we plot the feature pairs and interpolate the points using a cubic Hermite curve [HL93]. The reason we use curve interpolation is to minimize the perceptual tempo change around the feature pairs. Once an interpolation curve $C(u) = (s(u),t(u))$ has been computed, each music event, occurring at a time $s_* = s(u_*)$, is moved to $t_* = t(u_*)$ (see Figure 6(a)).

Before we apply the scaling to the music, we discard the points that will give large local deformations and lead to abrupt time scaling of the music. The points to be discarded that are further than a user-specified threshold from the least-squares line which approximates all the feature pairs. The red points in Figure 6(a) are removed, producing the new curve illustrated in Figure 6(b). This new curve will change the tempo of the music locally, with natural ritardando and accelerando effects.

Next we timewarp the motion [BW95] in order to synchronize the feature point pairs that were discarded during the time-scaling of the music. Note that the music feature points c and d of the discarded points in Figure 6(b) are shifted to a' and b' during time-scaling of the music. Therefore, the original motion should be timewarped so that the feature points a and b are moved to a' and b', respectively. Figure 6(c) shows the interpolation curve for the motion timewarping. Note that the horizontal axis is the time axis of the original motion and the vertical axis is that of the timewarped motion. After music modification and motion timewarping, we obtain the final synchronized animation and BGM.

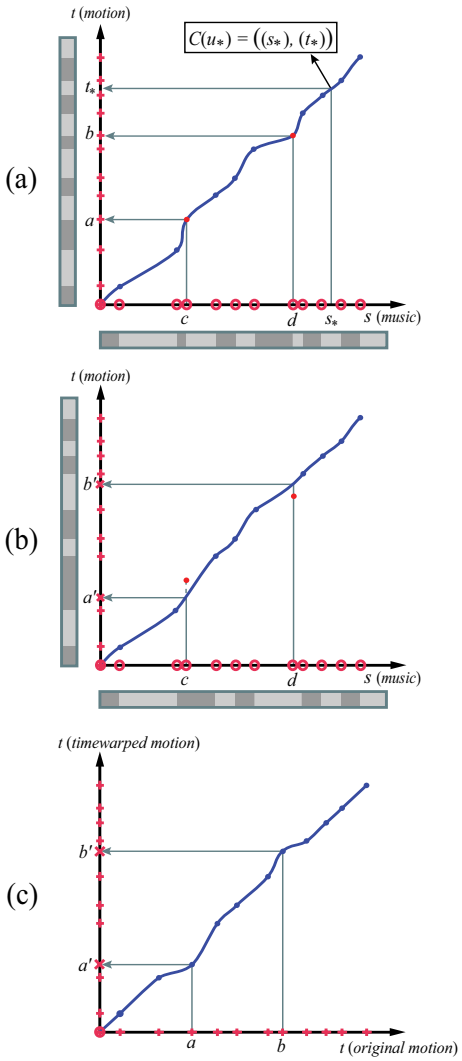

Figure 6: *Music time-scaling and motion timewarping using Hermite curve interpolation. The red circles on s-axis indicate the feature points of the BGM, and the red crosses on t-axis indicate the feature points of the motion: (a) all feature pairs used to interpolate the curve; (b) after removal of feature pairs that will damage the music; (c) interpolated curve for motion timewarping.*

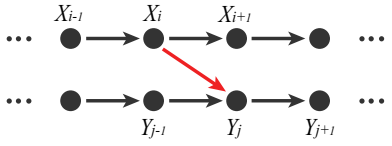

Figure 7: *Adding new transition edges to a music graph.*

6. Music Graph

The music graph encapsulates connections between several music sequences. New sequences of music can be generated by traversing the graph and applying melody blending at the transition points. The goal of the music graph is to retain the natural flow of the original music, while generating many new tunes. The concept of the music graph is similar to the motion graph [KGP02, AF02, LCR*02] which is commonly used for synthesizing motion.

The vertices of a music graph correspond to a user-specified unit into which the music is segmented. In our work, we segmented the music into bars or half-bars which usually form units of chord progression. The 'chord notes' of a specific chord are the set of notes that construct the chord [Cok86]. For example, a C Major chord consists of the notes 'C', 'E', and 'G'. Analyzing the exact chord at a given point in the melody is not easy [Row04]. We assume the following facts about the input music clips to simplify the estimation of chords:

- The MIDI data has at least one chord track (e.g. left hand part for the piano) as well as a melody track.
- All the notes in the chord track of a music segment will be included in the chord note set of the chord corresponding to that segment.

We calculate a distance for each vertex pair (X_i, Y_j) which is defined as follows:

$$\begin{aligned} &Dist(X_i,\ Y_j) \\ &\quad = w \cdot ChordDist(X_{i-1}, Y_{j-1}) + ChordDist(X_i, Y_j) \\ &\quad + w \cdot ChordDist(X_{i+1}, Y_{j+1})), \end{aligned} \tag{12}$$

where

$$\begin{aligned} &ChordDist(X_i,\ Y_j) \\ &\quad = 1\ /\ (\text{the number of same notes in chords } X_i \text{ and } Y_j) \end{aligned} \tag{13}$$

and w is a weight constant between 0 and 1. The difference between the chords at two vertices is estimated by counting the number of notes that are shared between the chords. If $Dist(X_i,\ Y_j)$ meets the threshold requirement, we create an edge in the music graph and a melody transition by blending the melody track (Figure 7), using the following rule:

$$\begin{aligned} &\text{If } |\ Last(X_i) - First(Y_j)| > 12 \\ &\quad First(Y_j) = \\ &\quad ClosestChordNote((Last(X_i) + Second(Y_j))\ /\ 2). \end{aligned} \tag{14}$$

The *First*, *Second* and *Last* functions return the first, second, and last notes of the segmentation unit, respectively. The function $ClosestChordNote(x)$ returns the chord note in Y_j which is closest in pitch to note x. If the pitch interval at a transition point is larger than an octave, the first note of the target node is replaced by the chord note which is closest to the mean of the surrounding notes. Figure 8 shows an example of the melody blending procedure.

A traversal of the music graph begins at a vertex selected by the user. Because every edge in the music graph is weighted by the distance function $Dist(X_i,\ Y_j)$, the next edge in the traversal can be selected by a random process, influenced by the score, as in a Markov chain [Tri82], which is used as a standard tool for algorithmic composition in computer music area [Cam94, TRMB01].

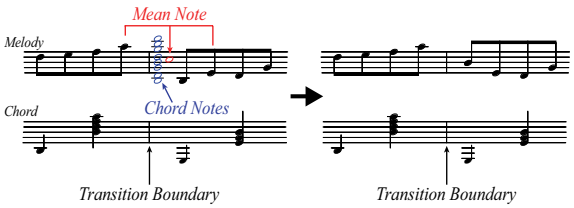

Figure 8: *An example of melody blending.*

The system randomly traverses the music graph repeatedly (100 times in our work) retrieving new music sequences. We expect that some of these will synchronize more effectively with the motion than the original music clips. We measure the disparity between each new music sequence and the given motion using the DP matching distance function, and select the sequence corresponding to the minimum distance.

7. Results

In our experiments, we used three motion capture data files, showing dancing, ballet, and walk-to-run movement; and the music data was of various genres, including classical, waltz, and dance.

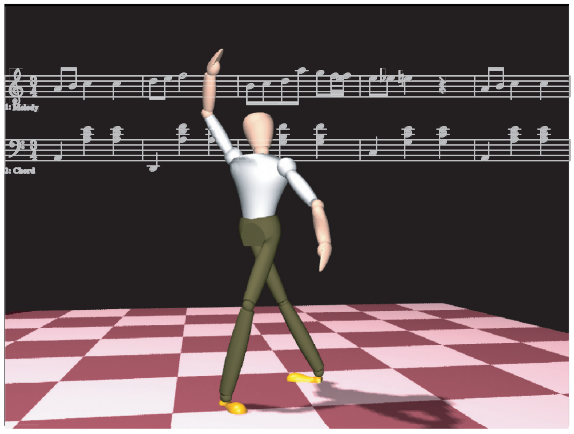

Figure 9: *A ballet motion synchronized with Chopin's waltz.*

In the first test, we attempted to synchronize the footstep points and end-effector (hands and feet) velocity feature points from the ballet motion (Figure 9, 13 sec) with the note velocity (volume) variation points from Chopin's waltz (1 min 40 sec). We extracted 20 feature points from the motion and 398 feature points from the music. DP matching paired 20 feature points from the music with the motion feature points, and 11 feature pairs were used for music modification. The remaining 9 feature pairs were used for motion timewarping. We also tried synchronizing all 20 feature

pairs using only music modification. In this case the synchronized points were clearly apparent but the music was severely damaged. Figure 6 shows parts of the interpolation graphs for this example. To measure how much the music is deformed by time-scaling, we calculated the integral of the second derivative of $s(u)$ from the interpolation curve $C(u) = (s(u), t(u))$ (see Section 5.2). The extent to which the tempo of the music changes our time is thus expressed as:

$$\int s''(u)\, du. \tag{15}$$

Similarly, the overall extent of the deformation of the motion can also be calculated using the interpolated curve in Figure 6(c). Table 1 shows deformation values for this example.

Number of feature pairs	Deformation value	
	(Music)	(Motion)
20	21.047	0.000
11	3.159	8.215

Table 1: *Deformation values of music and motion before and after feature pair removal in the first example.*

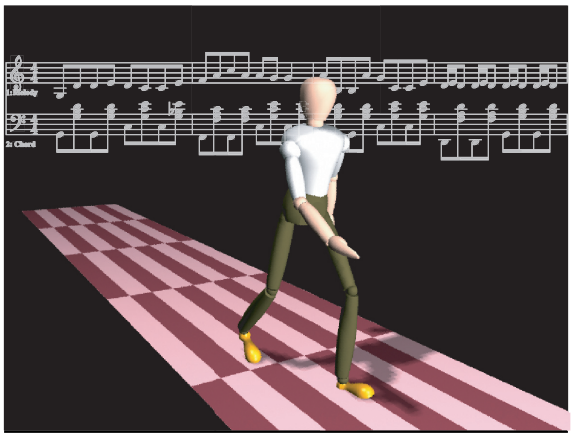

Figure 10: *A walk-to-run motion synchronized with Offenbach's 'Can-can'.*

In the second example, the motion features were the footstep points of the walk-to-run motion (Figure 10, 18 sec) and the music features were obtained from the fitness to half note of the 'Can-can' tune (45 sec) by Offenbach. In this case we ignored the feature time distance (the second term in RHS of Equation 7). The matched feature points, which are shown in Figure 11, gave us gradually accelerating BGM which was more natural than the result of dynamic local time-scaling. Table 2 compares the amount of deformation applied to the music and the motion when the feature time distance is considered and when it is ignored. In this case, ignoring the time distance gives a better result. Table 3 shows the computation time of the first and second example.

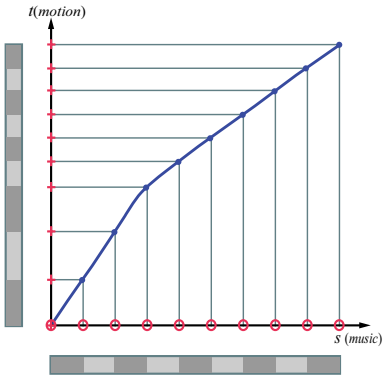

Figure 11: *Example of feature pairing with no time distance penalty.*

Considering time distance		Ignoring time distance	
(Music)	(Motion)	(Music)	(Motion)
7.763	16.292	6.505	0.0

Table 2: *The extent of music and motion deformation, considering and ignoring the feature time distance, in the second example.*

In the third example, we used waltz music, and dance music to construct two separate music graphs. After creating the vertices of the graphs, 1042 new edges were added to the first graph, and 502 to the second. The two music graphs were then tested on ballet and dance motions (Figure 12, 20 sec). The results were new music sequences which filled the motion more closely than the original music sequences. Table 4 compares the DP matching distance of the original music with that of the music synthesized using the music graph.

8. Conclusions & Future Work

We have suggested a method to synchronize background music and motion using DP matching and the music graph. Our method matches the feature points extracted from the music and motion by time scaling the music and timewarping the motion. By modifying each component a little, we can minimize the changes to the original data necessary to synchronize the feature points.

The music graph is a new way to synthesize new music from a directed graph of music clips. It has various applications. In this paper we show how it can be used to generate well-synchronized background music for the given animation. There are several factors that could make the music graph more useful. Replacing random search with systematic traverse methods, as used in motion graph research [AF02, LCR*02], is one possibility. Additionally we could extend the functions for transition distance and melody blending to consider melodic or rhythmic theories.

	Length of Data		Number of feature points		Computation time (sec)		
	Music	Motion	Music	Motion	DP matching	Music time-scaling	Motion timewarping
1st Example	100 sec	13 sec	400	20	0.172	0.469	0.281
2nd Example	45 sec	18 sec	102	37	0.031	0.609	0.375

Table 3: *Computation time for music-motion synchronization examples.*

Figure 12: *A dance motion synchronized with a music generated from music graph.*

Music	Length	DP matching distance (with ballet motion)
Waltz 1	2 min	18.22
Waltz 2	1 min 40 sec	20.93
Synthesized	13 sec	**15.90**
Music graph construction time		0.032 sec
Time for 100 traversals		0.484 sec

Music	Length	DP matching distance (with dance motion)
Dance 1	1 min 25 sec	13.22
Dance 2	1 min 25 sec	13.12
Dance 3	27 sec	12.39
Synthesized	20 sec	**9.99**
Music graph construction time		0.015 sec
Time for 100 traversals		0.781 sec

Table 4: *DP matching distances for the original music and for music synthesized using a music graph.*

The same technique can be extended to synchronize music and video. Video features can be extracted either from overall changes to the contents of the frame, or from movements of individual objects, which can be analyzed using the rotoscoping technique [AHSS04]. At a higher level, it may be possible to parameterize both music and animation in terms of their emotion content [BF00, MK03, UA95]. Synchronizing emotions could be a fascinating project.

Acknowledgement

This work was supported (in part) by the Ministry of Information & Communications, Korea, under the Information Technology Research Center(ITRC) Support Program, and Grant number 2003-1-0362 from Yonsei University.

References

[AF02] ARIKAN O., FORSYTH D.: Interactive motion generation from examples. In *Proceedings of ACM SIGGRAPH* (2002), pp. 483–490.

[AFO03] ARIKAN O., FORSYTH D., O'BRIEN J.: Motion synthesis from annotations. In *Proceedings of ACM SIGGRAPH* (2003), pp. 402–408.

[AHSS04] AGARWALA A., HERTZMANN A., SALESIN D., SEITZ S.: Keyframe-based tracking for rotoscoping and animation. In *Proceedings of ACM SIGGRAPH* (2004), pp. 584–591.

[BF00] BRESIN R., FRIBERG A.: Emotional coloring of computer-controlled music performances. *Computer Music Journal 24*, 4 (2000), 44–63.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *Proceedings of ACM SIGGRAPH* (1995), pp. 97–104.

[Cam94] CAMBOUROPOULOS E.: Markov chains as an aid to computer assisted composition. *Musical Praxis 1*, 1 (1994), 41–52.

[CBBJR03] CARDLE M., BROOKS S., BAR-JOSEPH Z., ROBINSON P.: Sound-by-numbers: motion-driven sound synthesis. In *ACM Symposium on Computer Animation* (2003), pp. 349–356.

[CBBR02] CARDLE M., BARTHE L., BROOKS S., ROBINSON P.: Music-driven motion editing: local motion transformations guided by music analysis. In *Proceeding of Eurographics UK* (2002), pp. 38–44.

[Cok86] COKER J.: *Improvising Jazz*. Fireside Publisher, 1986.

[DKP01] DOEL K. V. D., KRY P. G., PAI D. K.: Foley automatic: physically-based sound effects for interactive simulation and animation. In *Proceedings of ACM SIGGRAPH* (2001), pp. 537–544.

[DP96] DOEL K. V. D., PAI D. K.: Synthesis of shape dependent sounds with physical modeling. In *Proceedings of the International Conference on Auditory Display* (1996).

[DYN03] DOBASHI Y., YAMAMOTO T., NISHITA T.: Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics. In *Proceedings of ACM SIGGRAPH* (2003), pp. 732–740.

[FMJ02] FOD A., MATARIC M., JENKINS O.: Automated derivation of primitives for movement classification. *Autonomous Robots* (2002), 39–54.

[GBBM96] GREUEL C., BOLAS M. T., BOLAS N., MCDOWALL I. E.: Sculpting 3d worlds with music. In *IS and T/SPIE Symposium on Electronic Imaging* (1996), pp. 306–315.

[HDL02] HU N., DANNENBURG R., LEWIS A.: A probabilistic model of melodic similarity. In *Proceeding of International Computer Music Conference* (2002).

[HFGL95] HAHN J., FOUAD H., GRITZ L., LEE J.: Integrating sounds and motions in virtual environments: sound for animation and virtual reality. In *SIGGRAPH Course Notes* (1995), no. 10.

[HG99] HWANG J., GERARD J.: Musical motion: a medium for uniting visualization and control of music in the virtual environment. In *a Conferrence on Virtual Systems and Multimedia* (1999).

[HL93] HOSCHECK J., LASSER D.: *Fundametals of Computer Aided Geometric Design*. AK Peters, 1993.

[JM03] JENKINS O. C., MATARIC M. J.: Automated derivation of behaviour vocabularies for autonomous humanoid motion. In *Proceedings of Autonumous Agents and Multi Agent Systems* (2003), pp. 225–232.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *Proceedings of ACM SIGGRAPH* (2002), pp. 473–482.

[KPS03] KIM T. H., PARK S. I., SHIN S. Y.: Rhythmic-motion synthesis based on motion-beat analysis. In *Proceedings of ACM SIGGRAPH* (2003), pp. 392–401.

[Lay98] LAYBOURNE K.: *The Animation Book*. Three Rivers Press, 1998.

[LCR*02] LEE J., CHAI J., REITSMA P., HODGINS J., POLLARD N.: Interactive control of avatars animated with human motion data. In *Proceedings of ACM SIGGRAPH* (2002), pp. 491–500.

[MH95] MISHRA S., HAHN J.: Mapping motion to sound and music in computer animation and ve. In *Proceedings of Pacific Graphics* (1995).

[MK03] MICHEL P., KALIOUBY R. E.: Real time facial expression recognition in video using support vector machines. In *Proceedings of the 5th International Conference on Multimodal Interfaces* (2003), pp. 258 – 264.

[NKH*94] NAKAMURA J., KAKU T., HYUN K., NOMA T., YOSHIDA S.: Automatic background music generation based on actors' mood and motions. *Journal of Visualization and Computer Animation 5*, 4 (1994), 247–264.

[OCE01] O'BRIEN J. F., COOK P. R., ESSL G.: Synthesizing sounds from physically based motion. In *Proceedings of ACM SIGGRAPH* (2001), pp. 529–536.

[Row04] ROWE R.: *Machine Musicianship*. MIT Press, 2004.

[SDO*04] STONE M., DECARLO D., OH I., RODRIGUEZ C., STERE A., LEES A., BREGLER C.: Speaking with hands: Creating animated conversational characters from recordings of human performance. In *Proceedings of ACM SIGGRAPH* (2004), pp. 506–513.

[TN98] TADAMURA K., NAKAMAE E.: Synchronizing computer graphics animation and audio. *IEEE Multimedia 5*, 4 (1998), 63–73.

[Tri82] TRIVEDI K.: *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice-Hall, 1982.

[TRMB01] TRIVINO-RODRIGUEZ J. L., MORALES-BUENO R.: Using multiattribute prediction surffix graphs to predict and generate music. *Computer Music Journal 25*, 3 (2001), 62–79.

[UA95] UNUMA M., ANJYO K.: Fourier principles for emotion-based human figure animation. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), pp. 91–96.

[YLC04] YOO M. J., LEE I. K., CHOI J. J.: Background music generation using music texture synthesis. In *Proceeding of the International Conference on Entertainment Computing* (2004), pp. 565–570.

[YT99] YABE T., TANAKA K.: Similarity retrieval of human motion as multi-stream time series data. In *DANTE* (1999), pp. 279–286.