Report Number: WUCSE-2004-66

2004-10-19

Automated Motion Synthesis for Virtual Choreography

Gazihan Alankus, A. Alphan Bayazit, and O. Burchan Bayazit

In this paper, we present a technique to automati-cally synthesize dancing moves for arbitrary songs. Our current implementation is for virtual characters, but it is easy to use the same algorithms for entertainer robots, such as robotic dancers, which fits very well to this year's conference theme. Our technique is based on analyzing a musical tune (can be a song or melody) and synthesizing a motion for the virtual character where the character's movement synchronizes to the musical beats. In order to analyze beats of the tune, we developed a fast and novel algorithm. Our motion synthesis algorithm analyze... **Read complete abstract on page 2.**

Automated Motion Synthesis for Virtual Choreography

Gazihan Alankus, A. Alphan Bayazit, and O. Burchan Bayazit

Complete Abstract:

In this paper, we present a technique to automati-cally synthesize dancing moves for arbitrary songs. Our current implementation is for virtual characters, but it is easy to use the same algorithms for entertainer robots, such as robotic dancers, which fits very well to this year's conference theme. Our technique is based on analyzing a musical tune (can be a song or melody) and synthesizing a motion for the virtual character where the character's movement synchronizes to the musical beats. In order to analyze beats of the tune, we developed a fast and novel algorithm. Our motion synthesis algorithm analyze library of stock motions and generates new sequences of movements that were not described in the library. We present two algorithms to synchronize dance moves and musical beats: a fast greedy algorithm, and a genetic algorithm. Our experimental results show that we can generate new sequences of dance figures in which the dancer reacts to music and dances in synchronization with the music.

2004-66

Automated Motion Synthesis for Virtual Choreography

Authors: Gazihan Alankus, A. Alphan Bayazit, O. Burchan Bayazit

Abstract: In this paper, we present a technique to automatically synthesize dancing moves for arbitrary songs. Our current implementation is for virtual characters, but it is easy to use the same algorithms for entertainer robots, such as robotic dancers.

Our technique is based on analyzing a musical tune (can be a song or melody) and synthesizing a motion for the virtual character where the character's movement synchronizes to the musical beats. In order to analyze beats of the tune, we developed a fast and novel algorithm. Our motion synthesis algorithm analyze library of stock motions and generates new sequences of movements that were not described in the library. We present two algorithms to synchronize dance moves and musical beats: a fast greedy algorithm, and a genetic algorithm.
Our experimental results show that we can generate new sequences of dance figures in which the dancer reacts to music and dances in synchronization with the music.

Type of Report: Other

Automated Motion Synthesis for Virtual Choreography

Gazihan Alankus
Computer Science and Engineering
Washington University in St. Louis
gazihan@cse.wustl.edu

A. Alphan Bayazit
Electrical Engineering
Princeton University
alphan@princeton.edu

O. Burchan Bayazit
Computer Science and Engineering
Washington University in St. Louis
bayazit@cse.wustl.edu

***Abstract*— In this paper, we present a technique to automatically synthesize dancing moves for arbitrary songs. Our current implementation is for virtual characters, but it is easy to use the same algorithms for entertainer robots, such as robotic dancers, which fits very well to this year's conference theme.**

Our technique is based on analyzing a musical tune (can be a song or melody) and synthesizing a motion for the virtual character where the character's movement synchronizes to the musical beats. In order to analyze beats of the tune, we developed a fast and novel algorithm. Our motion synthesis algorithm analyze library of stock motions and generates new sequences of movements that were not described in the library. We present two algorithms to synchronize dance moves and musical beats: a fast greedy algorithm, and a genetic algorithm. Our experimental results show that we can generate new sequences of dance figures in which the dancer reacts to music and dances in synchronization with the music.

***Index Terms*— Motion Planning, Virtual Choreography, Motion Analysis, Motion Synthesis, Beat Analysis.**

I. Introduction

Human animation plays an important role in computer graphics, computer games, and virtual reality systems. There are two main methods to generate automated human animation, (i) synthesize new motion from scratch [8], [11], [14], and (ii) synthesize new motion from existing motion data [1], [2], [3], [4]. Both have their advantages and disadvantages, for example, if an algorithm can generate a requested motion automatically from scratch, a wider range of movements can be achieved. However, in most cases due to the high dimensionality of the problem, generating a motion from scratch takes a long time. On the other hand, if the algorithm uses an external motion library, it can quickly generate the motion. The drawback in this approach is, in most of the cases the variety of the generated motion is limited by the library.

Regardless of the method of choice, the goal of the human animation is to simulate humans doing some task. In most cases, visual quality of the animation is used as a metric to evaluate an animation. While vision plays an important role in viewers perception, another sense, hearing is also important. If the animation is in synchronization with the sound, the experience would become more exciting.

In this paper, we present our approach to combine sound with motion synthesis. Our aim is to generate a dance sequence in which the virtual dancer dances to the beats of the music. Our system has three components: (i) music analyzer, (ii) motion analyzer and (iii) motion synthesizer. Through our novel music analyzing algorithm, we find the beats in the music. We do not have any restriction for the type of the music, it can be a song or it can be a melody. We use a stock motion library containing motion capture data. Our motion analyzing algorithm identifies similar frames in the motion capture data and finds dance figures (a sequence of frames that can be grouped together) and moves (sharp changes in the motion of some body parts: hip, hand, elbow, knee or foot). It is our motion synthesis algorithm's job to generate a sequence of dance figures where dance moves are synchronized with musical beats. For motion synthesis, we propose two algorithms: a greedy algorithm and a genetic algorithm which tries to increase the synchronization between moves and beats.

Fig. 1. A dancing figure.

Although our current implementation is for virtual characters using motion capture data (see Figure 1), our algorithms are general and can be easily applied to real robots making them closer to humans as they entertain humans. For example, a motion analyzer component can use a robot's control data instead of motion capture data. Alternatively, the dance sequence for a human character can be used to control a humanoid robot [17], [21].

We would like the emphasize the fact that all the components of our system is fully automated, hence we can generate dance sequence from arbitrary tunes, using arbitrary motion capture data without human intervention. There have

been similar work before, but either the musical beats were periodic, i.e., no need to find them [5] or they were manually selected [9]. To the best of our knowledge, we are the first researchers which have fully automated dance sequence generator for human animation.

We will talk about the related work in the next section. Section III describes our system. Sections IV, V and VI describes the components of our system. We present our experimental results in Section VII. Section VIII concludes our paper.

Our web based dance sequence generator will be available shortly at *http://www.cse.wustl.edu/~bayazit*.

II. Related Work

Although developed independently, some of the key features of our system are similar to Kim et al.'s rhythmic-motion synthesis [5]. We both analyzed some motion capture data, identified motion primitives and dance moves, and synthesized a dance sequence. However, there are three significant differences between the systems. Kim et al. assumed the input sound had periodic beats, restricting the system to musical tunes with repetitive beats. Furthermore, those beats were not identified automatically. In contrast, we capture the arbitrary musical beats from any given tune. They have used a dance-move-music-beat matching algorithm similar to our greedy algorithm. On the other hand, we are presenting a second algorithm based on genetic algorithms. The advantage of using a genetic algorithm is the ability to optimize the dance sequence globally. A greedy algorithm had limited backtrack ability if things go wrong. We noticed that our genetic algorithm generates more variety in the dance figures. Finally, we have implemented a different algorithm to identify dance moves. Jehan et al. [9] developed a system that can generate dance video of real people based on stock video. However, the dance moves in the original video frames were manually selected.

Generating new motion from motion capture data is a well studied problem. Pullen and Bregler presented an animation system which generates new animation through motion data with human assistance [20]. Barbic; et al. [2] segmented motion capture data into distinct behaviors. Wang and Bondeheimer [23] evaluated a new cost metric for selection of transition frames. Kovar et al. [13] developed an algorithm to remove artifacts after merging motion data from different frames. Pettre et al. [19] combined probabilistic motion planning with motion capture data, and used warping technique to blend frames. Kovar et al. [12] presented motion graphs similar to our transition graph. Arikan and Forsyth [1] used randomized search to generate natural looking motion. Wiley and Hahn [24], showed an interpolation algorithm to synthesize desired moves from predefined data. Both Nakaoka et al. [17] and Safonova et al. [21] used motion capture data to generate new moves for humanoid robots.

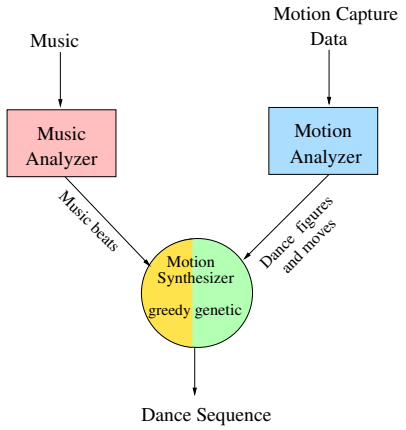


Fig. 2. System overview.

There have been some work on musical beat detection. Harper and Jernigan [7] used recurrent timing networks to detect the beats. Hainsworth and Macleon [6] used particle filters to identify the beats. Laroche [15] used maximum likelihood assuming the tempo was constant. Jensen and Andersen [10] found most probable beat intervals by using a feature extracted from note onsets. Scheirer [22] used filter-banks to predict the beats in the music.

III. System Overview

As discussed in Section I, our system has three components (see Figure 2). The *music analyzer* processes a musical tune and finds the musical beats (low frequency sounds). The *motion analyzer* processes motion capture data to find dance figures and moves. Usually motion capture data are long sequences of frames and similar frames exist in different times. Our motion analyzing component identifies such frames and builds a transition graph. Each node of the graph represents the start of a new dance figure. A dance figure is basically a sequence of consecutive frames. The *motion synthesizer* gets the beats, transition graph and moves and tries to generate a sequence of dance figures that will be synchronized to musical beat. We have developed two algorithms for the motion synthesizer. Our first algorithm is a greedy algorithm. It continuously adds new figures to the dance sequence. Since there may be more than one figure that can be added to the end of a dance sequence, the greedy algorithm selects the figure that will give maximum synchronization between musical beats and dance moves. In order to increase the variety of dance figures, we also consider probabilistic selection. Our second algorithm is a genetic algorithm that tries to increase dance-move music-beat synchronization. The genes in the algorithm represent sequence of dance figures. The algorithm uses the amount of synchronization between moves and beats as the fitness function. If it is necessary either motion synthesizing algorithm can change the timing of the dance figures to fit musical beats.

IV. Music Analyzer

As discussed in III, the goal of the music analyzer is to identify musical beats. We believe that, most of the beats are where the bass sounds are (such as when the drums play). Bass sounds are mostly low frequency sounds. We would like to find the envelope for them and find the peak points in the envelope which are good candidates for possible beat points. In our implementation, any peak point whose value is greater than the average value of peaks of the envelope, is treated as a beat. Figure 3 is a good example of the peak points matching the beat of a sound. So, in order to find beat, we need to find low frequency sounds first, then find their envelope. The easiest way to get low frequency sounds is to apply a low-pass filter to the audio data. There are several such filters. It is important for us to have an algorithm suitable for real-time or fast static applications, so we use a low-pass filter based on convolution which is common in signal processing applications.

Convolution for two discrete signals $x[n]$ and $y[n]$ can be defined as:

$conv(x, y)[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k]$.

It can also be defined as response of a Linear-Time-Invariant system with impulse response $y[n]$ to an input $x[n]$ [18]. In this work, $x[n]$ is always the audio signal whereas $y[n]$ is the Finite Impulse Response (FIR) of a Low-Pass filter system. Please note that $y[n]$ values are independent of the audio signal and can be read from a lookup table. Our beat detection algorithm can be summarized as in Algorithm IV.1. In the algorithm, $lp1$ is an order n-FIR filter for a given frequency cutoff, *conv* is convolution function. *abs* is the absolute value function, *subsample* is the function to reduce sample rate, *max* is the maximum points in the envelope.

Algorithm IV.1 Find-Beats(x)

1: $lx = conv(x, lp1)$ //Initial Low Pass filtering
2: $ax = abs(lx)$ //Extract absolute values
3: $x2 = subsample(ax)$ //Subsampling
4: $e = conv(x2, lp2)$ //Very Low Pass Filtering
5: $averageMax = max(e)/length(max(e))$
6: return $max(e) > averageMax$

Line 1 in Algorithm IV.1 is the initial low pass filtering. There are two reasons for initial low pass filtering. First, low frequencies convey better information regarding the beat points which are mostly effected by bass sounds. Second, there is a practical problem in filtering. If the cut of frequency of envelope detection is too low (i.e. corresponding to a few beats per second), the FIR filter may fail. Recall that typical audio frequency is 44.1KHz. During the envelope detection, in order to filter very low frequencies (less than 4Hz), we need an intermediate signal which has a frequency between 4Hz and 44.1kHz. Hence, the initial low-pass filtering does the anti-aliasing for subsampling for the intermediate signal,

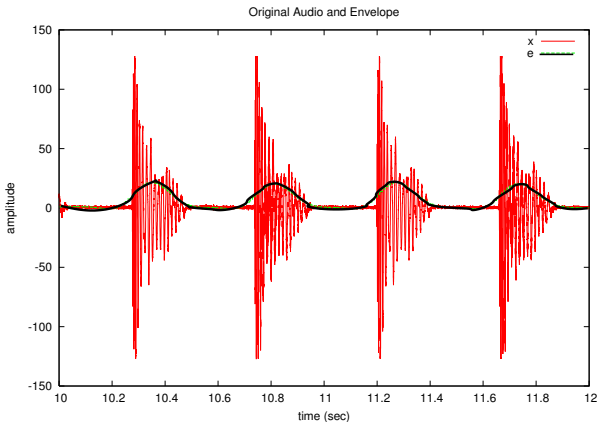


Fig. 3. Original sound and envelope (smooth curve). The envelope would have peaks in beat locations.

which has a sampling frequency of 100 Hz. Line 2 restricts the envelop to above 0 values. Line 3 reduces sampling rate. Line 4 finds the envelop. Line 5 finds the average value for the envelope peaks and Line 6 returns the peaks that are higher than the average.

V. Motion Analyzer

After extracting the beat times from the music, our aim is to create a dance sequence from the given motion capture data and synchronize it with the music. In order to make such a synchronization we need to find suitable frames in motion capture data that we can match with beats in music. A simple approach of assuming the motion capture data is in synchronization with the music beat usually fails, since musical rhythms changes frequently, and unless the motion capture data was recorded using the same music, there will always be synchronization problems. Also, we may want to use motion capture data from the different sources, recorded at different times. Hence, our goal is to analyze motion data and extract the dance figures and moves. We define dance figures and dance moves in the following way:

- A *dance figure*, $\mathbb{F}_i$, is a sequence of motion frames, $f^i{}_0 \dots f^i{}_{last_i}$. A dance figure can be followed by another dance figure, if the last frame of the former figure is similar to first frame of the later. *i.e.*for figures $\mathbb{F}_i$ and $\mathbb{F}_j$, $\mathbb{F}_j$ can follow $\mathbb{F}_i$ if and only if $f^i{}_{last_i} \simeq f^j{}_0$.
- A *dance move* is a motion frame that has significant change in the direction of the movement of some body part (hip, hand, elbow, knee or feet).

Our motion capture data is in Biovision Hierarchical Data format (BVH). This format provides skeleton hierarchy information as well as motion data. Using hierarchical information we can identify individual body parts or overall body pose. Identifying a body pose is very important since, the raw motion data is highly position-based. There may be cases where two frames may have similar poses yet their positions

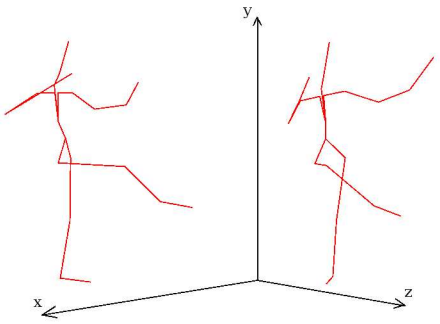


Fig. 4. Two similar dance frames.

and orientation may be far apart due to a dancer's tendency to turn and walk while dancing (see Figure 4). Hence, although the chance of finding the similar frames in motion data is low, it is very likely to find similar body postures due to repetitive nature of dance figures in a performance. Remember that two figures can be consecutive in the dance sequence only if the previous figure's last frame is similar to next figure's first frame. We need to define similarity between two frames in the following way.

Definition: Two frames f_i and f_j a of motion $\mathcal{M}$, are *similar* if and only if there exists an arbitrary translation T_{xz} on the XZ plane and an arbitrary rotation R_y around Y axis such that the points in $T_{xz}(R_y(f_i))$, (*i.e.*, transformed f_i), and f_j are closer to each other than ϵ.

If two frames f_i and f_j are similar, it is easy to observe that we can move from f_i to f_{j+1} and vice versa, provided that we apply the necessary transformation first.

Using this property, we identify similar frames in the motion capture data (see Figure 5(b)). Once similar frames are identified, we can use them as the starting points for dance figures. If f_i and f_j are similar, then we may have a figure starting with f_i, and another figure starting at f_j. Since their starting points are the same, both figures can be interchanged in the animation. A more formal definition of a dance figure is the following:

Definition: A *dance figure* is a sequence of frames $f_i \dots f_j$ from the motion capture data, $\mathcal{M}$, when

- $\forall f_k(f_k \in \{f_i, f_j\}), \exists f(f \in \mathcal{M}) : f_k \simeq f$, *i.e.*, there are some frames in motion capture data that are similar to f_i and f_j,
- $\forall f_k(f_k \in f_{i+1} \dots f_{j-1}), \forall f(f \in \mathcal{M}) : (f_k \neq f) \rightarrow (f_k \not\simeq f)$, *i.e.*, there are no frames in motion capture data that are similar to frames inside the dance figure (other than start and end).

Once we have identified the dance figures, we build the *transition graph* to represent movements between figures (Figure 5(c)).

Our next goal is to identify dance moves in a dance figure. As we have mentioned above a dance move is a significant

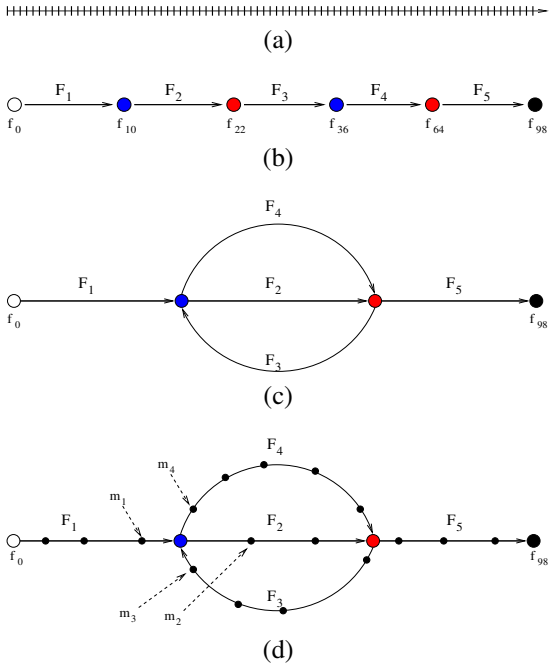


Fig. 5. An example motion capture data with dance figures and moves: (a) frames of initial motion capture data, (b) similar frames ($f_{10} \simeq f_{36}$, $f_{22} \simeq f_{64}$), and dance figures ($F_1, \dots, F_5$), (c) transition graph based, (d) dance moves (m_i) within figures.

change in one of the limbs' movement. In order to decide if there is a significant change, we look for sharp changes in velocity of the body part that we are interested in. We use the current position of the body part, position of the body part in the previous frame and the time rate of motion data to find the velocity vector, $\vec{Vi}$, of the body part in frame f_i. The significance of the change in the velocity vector or a body part, then can be found using vector dot product $\vec{V_{i-1}}.\vec{V_i}$. Note that, $\vec{V_{i-1}}.\vec{V_i} = |V_{i-1}||V_i|\cos(\theta)$, where θ is the angle between the two vectors. If the two vectors are opposite in the direction, (see Figure 6), the dot product becomes negative. The magnitude of the dot product is also important since it is proportional with the scalar velocities and change in direction. We utilize these observations and use the dot product of velocities to find moves. If any of the body parts in a frame has a dot product less than a threshold value τ, that frame is designated as a dance move.

VI. Motion Synthesizer

Once we have identified the beats of the music, dance figures, and dance moves, we can generate a sequence of dance figures where dance moves are synchronized to dance beats. In the simplest form, motion synthesizer moves on the *transition graph* and selects the figures with the maximum synchronization to the data. However, the moves in raw data do not necessarily have to be synchronized with the music beats (Figure 7(a)), motion synthesizer may increase or decrease the speed of a frame to fit the beat interval

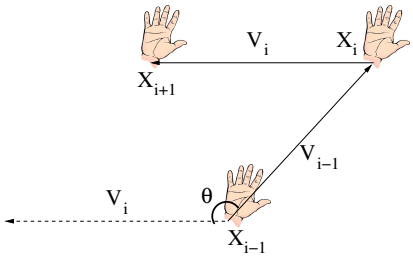


Fig. 6. Dance move for a hand. The hand is located at positions X_{i-1}, X_i, X_{i+1} at the frames f_{i-1}, f_i, f_{i+1}. Since, the direction of velocity vector is significantly changed, there is a dance move at frame f_i.

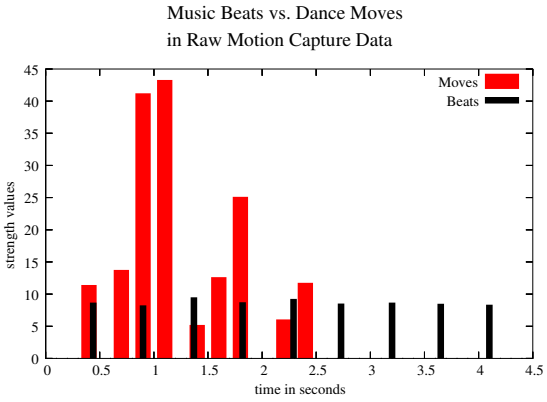
(Figure 7(b)). It may not be possible to match all the beats to all the moves, but synthesizer tries to maximize the matches.

We propose two algorithms to find a good sequence of dance figures that is synchronized with music. The first one is a greedy algorithm with backtracking, which tries to find the best matching frame among the closest dance moves, take it as a greedy choice and repeat the same process. If the algorithm gets stuck in a node it stores the current sequence as a possible solution, backtracks and tries other directions. In the end, the best solution among the possible solutions is returned.

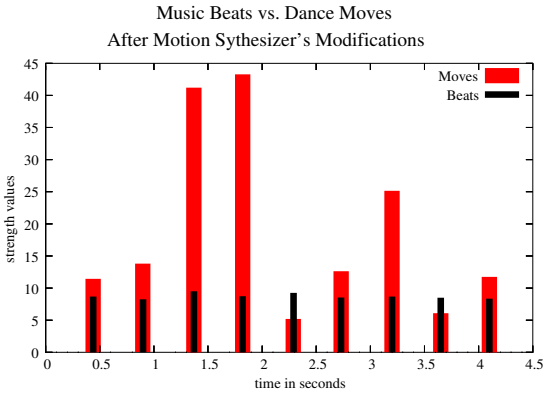
The other algorithm is a genetic algorithm that tries to optimize the dance sequence by taking a number of valid random dance figures as a population, applying the genetic operators of crossover and mutation to create new generations. Both greedy algorithm and genetic algorithm use the same evaluation function, $Score(x)$, to decide how good a solution is. $Score$ takes a sequence of dance figures (not necessarily the same length as the song), considers how much of the song is covered by the sequence, how good the beats and moves match and how much the speed of frames were modified and returns a score. Our score function for a sequence S of dance figures is: $Score(S) = w_1 * beatpercent + w_2 * beatpercent + w_3 * length - w4 * avgspeeddev$, where

- beatpercent: percentage of music beats in S that are synchronized with a dance move.
- movepercent: percentage of dance moves in S that are synchronized with a music beat.
- length: number of beats in the portion of the song which S can cover.
- avgspeeddev: average speed deviation weighted by durations of dance figures in S.

1) Greedy Algorithm: In the greedy algorithm, we walk on the transition graph (see Figure 5). We always keep track of the last synchronized move. When we encounter a new dance move m_i, our goal is to synchronize a beat with this move by changing the speed of the dance. If we can synchronize it with a beat in the music, we record it as the last synchronized move. Otherwise we continue walking on the



(a)



(b)

Fig. 7. Music beats vs. dance moves. (a) Without any modification by motion synthesizer, dance moves and music beats may be asyncronized. (b) Motion synthesizer modifies the length of moves to fit music beats. The strength of moves and beats are not normalized. However, the larger the value the stronger the beat or more dramatic the dance move is.

graph. Depending on the dance figure we are following (arcs in the graph), we may have more than one dance figure to continue on our walk. Our choice of next dance figure greatly affects the success of the solution. Please note that, in order to find the optimum choice we should do an exhaustive search which is not feasible, hence the reason for using a greedy algorithm. So, we select the dance figure which contains the closest dance move we can synchronize with a music beat. Although this greedy choice works well, we may end up in an ending dance figure (no transformation to other figures is possible, *i.e.*motion capture data ends there), or we may not find a dance move that fits to a music beat before the song ends or a time limit reached. In order to overcome these issues, we implemented a backtracking mechanism. If we reach a dead-end before the end of the song, we store the sequence to dead-end as a possible solution and backtrack to closest greedy decision point and follow the second best greedy choice. The greedy algorithm is described in Algorithm VI.2. One drawback of this algorithm is that it

does not promote visiting different figures in the dance graph. Although it generates well synchronized solutions, in same cases it may use a small subset of the dance graph. This means the resulting dance sequence would have the repetition of the same dance figures.

Algorithm VI.1 Greedy_Choice(G,B,lastmove, lastbeat)

```
1: Q=empty priority queue of dance moves sorted by distance to lastnode
2: find all the neighboring dance moves from lastmove and insert them to Q
3: while Q is not empty and iteration limit not reached do
4:    n_i=Q.extract_min()
5:    if dance move can be matched with a beat b_j then
6:       if this this walk is not chosen and rejected before then
7:          lastmove = n_i
8:          lastbeat = b_j
9:          return success
10:      end if
11:   else
12:      find all the neighboring dance moves from n_i and insert them to Q
13:   end if
14: end while
15: return failure
```

Algorithm VI.2 Greedy-Sequence(M,B)

```
1: G=Create-Graph(M)
2: lastmove = first dance move of graph
3: lastbeat = first beat in music
4: while while iteration limit not reached and lastbeat < end of song do
5:    result = Greedy_Choice(G, B, lastmove, lastbeat)
6:    if result = failure then
7:       add the current solution to the set of possible solutions
8:       backtrack to the previous successful greedy choice
9:       if there are no more choices to backtrack then
10:         break the while loop
11:      end if
12:   end if
13: end while
14: for all possible solutions S_i do
15:   fitness = Score(Si)
16: end for
17: return the solution S with the maximum fitness
```

To overcome this problem, we introduce a probabilistic version of our greedy algorithm that probabilistically rejects the best greedy choice based on the number of times that the graph node is visited before. The more number of times a graph node is visited before, the more likely that it will be rejected in the next greedy choice. With this probabilistic rejection included in the algorithm, the resulting solutions usually contain wider selection of dance figures.

2) Genetic Algorithm: The other algorithm we propose for finding a good sequence of dance figures that are synchronized with the music is based on genetic algorithm [16]. We believe our synchronization problem is an optimization problem for which genetic algorithms are well known. They represent possible solutions as genes. They change the gene pool at each iteration. The genes are probabilistically selected using a fitness function, and crossovered to generate new genes. Random mutations modify some genes to increase the entropy of the system. At the end of the each iteration, weakest genes are probabilistically removed from the gene pool.

In our implementation, we used $Score$ function as our fitness function. The genes represent the consecutive dance figures. For a gene to be valid, all the consecutive dance figures in the genes must be neighbors in the transition graph. We restrict our genetic operators to accept and produce valid genes. The size of the genes may vary since the a sequence may contain different number of figures.

We create our initial population as random walks in the motion graph starting from the first figure. We end the walks either upon reaching the end figure in the graph, or slightly exceeding the tune length. Our crossover operator takes a random figure F in the first parent and finds the same figure in the second parent. If there are more than one instance of the same figure, the algorithm probabilistically selects one. In other words, if the first parent represents a sequence $\{{F_0}^1, \ldots {F_i}^1, \ldots {F_k}^1\}$ and the second parent represents a sequence $\{{F_0}^2, \ldots {F_j}^2, \ldots {F_m}^2\}$, where ${F_i}^1 = {F_j}^2$, then the children will be ${F_0}^1, \ldots {F_j}^2, \ldots {F_m}^2$ and $\{{F_0}^2, \ldots {F_i}^1, \ldots {F_k}^1\}$.

For mutation, we select a random dance figure F_i in the gene and try to find a single figure or a sequence of figures that have start and end frames that are similar frames with the start and end frames of F_i. If we can find such a figure or figure sequence, we replace it with the selected figure in the gene, thus protecting the validity of the gene. If we cannot find a replacement for that figure, we try the next figure in the gene and give up when we reach the end of the figure.

VII. Experiments

In our experiments, we want to evaluate the performance of our algorithms under different conditions. We have a set of dance figures (70 total), and two songs. We have run our algorithm on an Athlon 3200++ with 1GB Memory running Linux. The animations of our experiments can be seen at *http://www.cse.wustl.edu/~bayazit*.

In our experiments, we have used two different songs, "Breakdance" which is highly rhythmic, and, "Garden" which is slower. A significant difference between two songs is that, Garden has some long periods without a beat. We designed our experiments to: (i) verify that our genetic algorithm works, (ii) evaluate the frame selection policy for each of the motion synthesizing algorithm, and (iii) compare the efficiency of each algorithm.

In order to verify our genetic algorithm, we investigated the average fitness of the gene pool over each iteration. A well

designed genetic algorithm should converge to some value. Figure 9 shows average fitness value of the gene pool after each generation. x axis represents the generation number and y axis represents the fitness score. We have run the genetic algorithm for 60 generations. The experiment took 5 minutes to finish. As it can be seen in the figure, the average score of the gene pool increases as the number of generations increases.

Next, we wanted to evaluate the figure selection bias of each algorithm. In order to do that, for a given solution, we counted how many times each dance figure occurred in the solution. If an algorithm favors some figures, those dance figures would be encountered more frequently than other dance figures. Figure 8 is the histogram of the dance figures that exist in the solution of each algorithm. This figure helps us visualize how each algorithm selects dance figures. x-axis represents frames. y-axis the frequency of the frames. It is clear from the figure that, the greedy algorithm favors some dance figures, since they have significant number of occurrences. That was expected, since greedy algorithm tries to match a move with the earliest beat, it would be easier, if the algorithm selects the short dance figures. The probabilistic greedy algorithm is more flexible since it has fewer number of repetitive figures. That can be explained by the fact that, although the greedy algorithm favors the earliest beat, the probabilistic greedy algorithm remembers frequency of a figure and is biased towards to less visited figures. Among all three algorithms, the most flexible algorithm is the genetic algorithm, since it selects several different figures. The reason for that is, the genetic algorithm is a global optimization algorithm, so instead of making decisions by evaluating immediate choices, the genetic algorithm evaluates a dance sequence's overall performance.

Finally, we want to compare the solutions of our algorithms for both songs. Table I shows the results. In the table, we show the time to find a dance sequence *Time*, score of the solution, *Score*, number of matched dance moves and music beats, *Matched Beat*, number of beats missed for the current solution *Missed Beat*, the average speed deviation (*Avg.Sp.Dv*), and the length of the dance sequence (*Length*). The solution times for both Greedy and Probabilistic Greedy are very close to each other. They are very fast and their performances are also close (number of matches, number of misses as well as the length of the solution). On the other hand, the genetic algorithm is significantly slower than other two algorithms. And it missed more beats in "Breakdance", however when we look at "Garden", we observe the advantage of the genetic algorithm. Remember that, "Garden" had some long intervals without a beat. The greedy and the probabilistic greedy algorithms fail to pass through such intervals since as they fail to find a beat, they continue to explore the transition graph until the queue becomes full or time out reached. In contrast, genetic algorithm already have

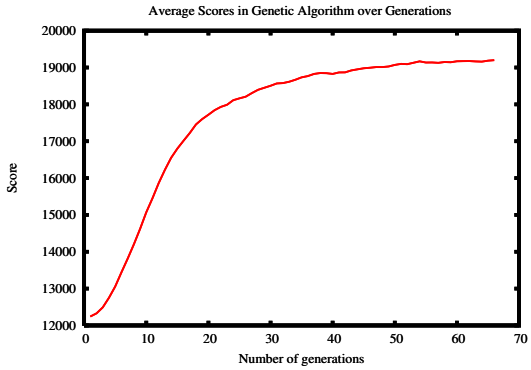


Fig. 9. Average fitness of the genes in "Breakdance".

some dance figures over the beatless intervals so they do not need to explore the transition graph. As a result, the greedy based algorithms could have find a solution to only %60 of the song, yet genetic algorithm found a solution that reaches the end of the song.

Our results suggested that, if a fast solution is required and the variety of the figures are not important, a greedy based algorithm is preferable. However, if the variety of the figures are important or the song has long beatless intervals, a genetic algorithm would be more efficient to find the dance sequence.

VIII. Conclusion

In this paper, we describe a fully automated system to synthesize a dance for virtual characters using motion capture data. We present algorithms to identify the beats in the music, and interesting moves in the motion data. We also propose two algorithms that will synchronize interesting dance moves with the music data to generate a sequence of dance moves. Our system can generate sequence of the dance moves that were not in the original motion capture data and our algorithms are fast.

Our future work includes implementing the addition of multiple characters and their coordination, and improving the matching algorithm by including the beat magnitude and the speed of dance moves.

References

[1] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490. ACM Press, 2002.

[2] J. Barbic, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of the 2004 conference on Graphics interface*, pages 185–194. Canadian Human-Computer Communications Society, 2004.

[3] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192. ACM Press/Addison-Wesley Publishing Co., 2000.

[4] A. Bruderlin and L. Williams. Motion signal processing. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104. ACM Press, 1995.

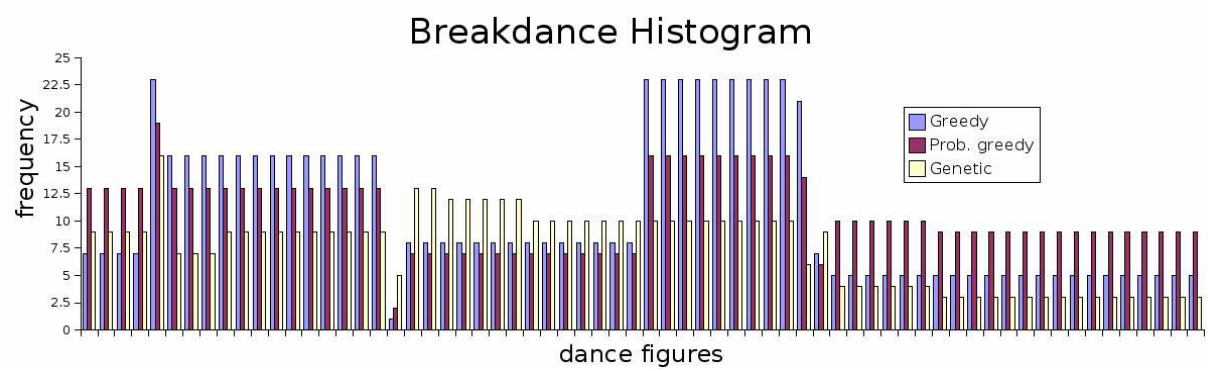


Fig. 8. Histogram of dance figures in Breakdance. The figure shows the frequency of dance figures in solutions of different algorithms.

BREAKDANCE (395 beats long)						
Algorithm	**Time (s)**	**Score**	**Matched Beat**	**Missed Beat**	**Avg.Sp.Dv.**	**Length (in beats)**
Greedy	2.4	20045	232	164	0.1	395
Prob. Greedy	4.1	20046	235	161	0.1	395
Genetic Alg.	1367	19952	278	118	0.1	395
GARDEN (268 beats long)						
ALGORITHM	**Time (s)**	**Score**	**Matched Beat**	**Missed Beat**	**Avg.Sp.Dv.**	**Length (in beats)**
Greedy	7.4	8826	112	60	0.1	171
Prob. Greedy	8.4	8828	116	56	0.1	171
Genetic Alg.	820	13592	201	68	0.09	268

TABLE I

STATISTICS FOR SAMPLE SONGS.

[5] T. h. Kim, S. I. Park, and S. Y. Shin. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Trans. Graph.*, 22(3):392–401, 2003.

[6] S. Hainsworth and M. Macleod. Beat tracking with particle filtering algorithms. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 245 – 248, 2003.

[7] R. Harper and M.E. Jernigan. Self-adjusting beat detection and prediction in music. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04)*, pages 245 – 248, 2004.

[8] J. K. Hodgins and W. L. Wooten. Animating human athletes. In *Robotics Research: The Eighth International Symposium . Y. Shirai and S. Hirose (eds)*. Springer-Verlag, 1998.

[9] T. Jehan, M. Lew, and C. Vaucelle. Cati dance: self-edited, self-synchronized music video. In *Proceedings of the SIGGRAPH 2003 conference on Sketches & applications*, pages 1–1. ACM Press, 2003.

[10] K. Jensen and T.H. Andersen. Beat estimation on the beat. In *IEEE Workshop Applications of Signal Processing to Audio and Acoustics*, pages 19–22, 2003.

[11] Y. Koga, K. Kondo, J. Kuffner, and J.C. Latombe. Planning motions with intentions. In *Proc. ACM SIGGRAPH*, pages 395–408, 1995.

[12] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482. ACM Press, 2002.

[13] L. Kovar, J. Schreiner, and M. Gleicher. Footskate cleanup for motion capture editing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–104. ACM Press, 2002.

[14] J .J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots (special issue on Humanoid Robotics)*, 12:105–118, 2002.

[15] J. Laroche. Estimating tempo, swing and beat locations in audio recordings. In *IEEE Workshop Applications of Signal Processing to Audio and Acoustics*, pages 135 – 138, 2001.

[16] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[17] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 3, pages 3905–3910, 2003.

[18] A. V. OppenHeim and A. S. Willsky. *Signals and Systems, Second Edition*. Prentice Hall, 1997.

[19] J. Pettre;, J.-P. Laumond, and Thierry Simeon. A 2-stages locomotion planner for digital actors. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 258–264. Eurographics Association, 2003.

[20] K. Pullen and C. Bregler. Motion capture assisted animation: texturing and synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 501–508. ACM Press, 2002.

[21] A. Safonova, N. Pollard, and J. K. Hodgins. Optimizing human motion for the control of a humanoid robot. In *2nd International Symposium on Adaptive Motion of Animals and Machines (AMAM2003)*, March 2003.

[22] E. Scheirer. Tempo and beat analysis of acoustic music. *Signal, Journal of Acoustic Society of America*, 103, 1998.

[23] J. Wang and B. Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 232–238. Eurographics Association, 2003.

[24] Douglas J. Wiley and James K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Comput. Graph. Appl.*, 17(6):39–45, 1997.