

# Music-Driven Character Animation

Danielle Sauer

Herb Yang

University of Alberta

## Abstract

In this paper we present a novel system for creating an animation that is synchronized to a piece of music. The system consists of two main stages. In the first stage, musical features such as the tempo, beat positions, and dynamics of an input song are extracted. The features are then mapped to control a character using some pre-defined primitive movements. The mapping is defined by the user using a very simple script. Many-to-one or one-to-many mappings are possible. In the case of one-to-many, the system selects one of the candidate movements randomly. The proposed approach is implemented as a plug-in for Alias-Wavefront's Maya, which allows for interchangeable characters and songs. The system is designed for a wide range of users and encourages experimentation by providing extensive user control. Very interesting animations can be generated using our approach.

Keywords: Character animation, music synchronization, music analysis.

## 1. Introduction

Animations, whether they are in movies, television, or video games, would not capture the viewer's interest if they were not accompanied by music. Music can set the tone for a scene and is generally added after the animation has been completed. It takes a large amount of work to coordinate a piece of music with a final animation, especially when specific movements must occur at certain times in the song. This paper describes a method of automatically synthesizing an animation that synchronizes with the input music. Our system allows the user to choose a piece of music and a character and outputs a synchronized animation that expresses the emotion and intent of the music through movement. The tempo, beat levels, and dynamic levels are extracted from the music and used to drive the animation. The user inputs a music file, a character created in Alias-Wavefront's Maya, and a script file. The script file gives the user the freedom to decide what movements will be used in the animation, when they will be performed, and by which body part of the character. These mappings, as defined in the script file, allow the user to experiment with different ways of arranging movement and music to create a unique animation. Our animation system is designed for use by a wide range of users, not just those who are proficient at animation or musically inclined. Our technique concentrates on simplicity of movement and user control for creating a unique music-driven animation.

## 2. Previous Work

Several different approaches to synchronizing an animation with music have been explored, but the majority of methods are not easy for a user without any musical or animation experience to use. In most systems the user must already have an animation to input for synchronization with the music. One approach uses a

transition graph made of motion capture data to select the frame sequence that best fits the beat pattern [Alankus et al. 2004]. In this case, the movements themselves are not driven by the music, only the order in which they are used. Mapping techniques between musical features and movements have also been attempted in other systems. One such method performs editing techniques on the animation curve according to a user specified mapping between musical features and motion filters [Cardle et al. 2002]. Their mapping technique is similar to ours, but our system creates the animation from scratch based on the music, rather than synchronizing the audio signal with an existing animation. The area of facial animation can also be examined regarding useful mapping methods that allow the input audio file to drive the animation [Brand 1999, Cao et al. 2005].

An additional technique with a mapping method comparable to ours involves making small changes to both the music file and the animation [Lee and Lee 2005]. This method synchronizes musical attributes extracted from the MIDI file with motion attributes, such as footsteps, extracted from the animation. Unlike our approach, they concentrate on changing the music to fit the existing animation, whereas we use the original music to create a novel animation. Another recent approach that maps MIDI data to movements is a system for interacting with a virtual character [Taylor et al. 2005]. A virtual character from the ANIMUS framework reacts to the music based on customized mappings between musical features and behaviours. Our framework has considerably more user control because the user is allowed to choose our system mappings, whereas their virtual character's behaviour mappings are predetermined by the system designer. Unlike our system, the character involved in their animation is not easily interchangeable and has very limited types of movements. We give the user more flexibility by allowing her to plug in her own character.

MIDI files are the most popular input format for songs because it is less difficult to extract musical attributes from the data [Cardle et al. 2002; Taylor et al. 2005]. These files are not easily accessible and imply musical knowledge by the user, so we choose to use .wav files and use signal processing techniques to extract the tempo, beat onsets, and dynamics from the music datum. Beats can be considered regular pulsations of music and determine the tempo of the music, which is the overall pace of a composition of music, e.g. fast, slow. Dynamics represent the variation of loud and soft levels in a song and establish the emotion of the music. Methods for performing beat detection have varied from convolution and subsampling [Alankus et al. 2004] to the Discrete Wavelet Transform [Tzanetakis et al. 2001] to using multiple agents for beat prediction [Goto and Muraoka 1994; Goto 2001]. Our system makes use of Tzanetakis et al.'s [2001] algorithm for determining the tempo and Goto's [2001] algorithm for discovering note onsets. We develop our own algorithm for dynamics extraction using the power spectrum.

The contributions of this paper are as follows:

- We develop a new system that extracts musical attributes and uses them to create an animation that moves directly to the music. We do not simply synchronize music with an animation, but we create an animation from scratch that is precisely tailored to user input.
- We present a new mapping method that allows the user to define how she wants the animation to look without having to create the animation herself. This approach gives maximum user control, encourages experimentation, and allows for interchangeable character figures.
- We present a new method to generate animation based on simple primitive movements, which are controlled by the above mentioned mapping method.

### 3. Proposed Method

Our character animation system is made up of two principal components. The first is musical attribute detection, where the musical attributes used by the system are extracted from the input music file. The second component is a mapping method that uses the results from the first component to map the music to character motion. The result is a unique animation that depends entirely on the music and a set of user-defined mappings.

#### 3.1. Musical Attributes

Once the user has specified the song she wants the animation to be based on, the system analyzes the sound file and extracts the tempo, beats, and dynamic levels of the music. These attributes are the most recognizable aspects of a musical piece, especially to an untrained ear, and important in deciding what types of movements portray the music. All of the code for detecting the musical attributes was written in Matlab using the m-file format. Matlab was chosen over other programming languages due to its excellent signal processing functionality. The tempo of the song is detected according to an audio analysis algorithm based on the discrete wavelet transform [Tzanetakis et al. 2001]. Once the tempo is determined (in beats per minute), we calculate how far apart the beats need to be in order to achieve the correct tempo. The beat increment is determined by converting beats per minute (bpm) into beats per second and dividing the sampling rate of the audio file by the beats per second value.

The mapping component of our animation system relies on having accurate beat positions to which it can map character movements. The tempo value and corresponding beat increment is not reliable enough for all audio files and cannot be solely depended on to provide correct information. The detected tempo value may not be precise enough to calculate the exact distance between beats, so we cannot simply add the estimated beat distance to each beat position to get the next position in the piece. This is due to instances where the tempo value determined by the algorithm is not an integer. Some of the values after the decimal point will be cut off and an error will build up over time in the position estimates because the beat increment will be slightly off. Instead, we use a beat onset algorithm that is based on two papers on beat tracking systems [Goto and Muraoka 1994; Goto 2001]. By performing windowing on the audio data, we can use the onset algorithm in each window and determine where the first beat in each section occurs. From there, our system calculates the

remaining beat positions in the section by adding the beat distance to the last determined beat position. The song is evaluated in sections in order for the algorithm to correct itself if the previous beat onset has strayed. The window size is determined by taking the sampling size of the audio file, 44100Hz, and dividing it by the beats per second value. The result is multiplied by an integer value that is determined by trial-by-error, usually 4 or 5 works best, to get the final window size. The onset algorithm is able to detect both the strong beat positions and the weak beat positions (the sub-beats between the beats that correspond to the tempo). Two arrays are then stored for use in the mapping method, one for the strong beat onsets and one for the weak beat onsets.

The dynamics detection algorithm makes use of the power spectrum to retrieve the peak and valley amplitudes from the song data. The amplitude corresponds to the strength of the signal at a point in time, so the peaks match the loud periods and the valleys match the quiet periods in the song. The algorithm uses a window size of 44100 samples. It performs the Fast Fourier Transform (FFT) on the window and computes the power spectrum. The inverse FFT is then performed on the result and the imaginary parts of the complex values are discarded. The regional maxima are detected from the remaining values, with the highest peak and the lowest peak of the window set aside. This algorithm is performed on each section of the signal and the 10 highest and 10 lowest peaks are used as the frame positions of the loud and soft components in the music. The system uses a dynamic rate scale of 1-5, where 1 denotes a soft dynamic and 5 denotes a loud dynamic. The detected peaks are given a value of 5 and the valleys are given a value of 1. The dynamic rate can alter a movement in order to express the dynamic level at the current frame. For example, the force used to hit a drum will be smaller for soft dynamics and larger for loud dynamics. Interpolation is used to determine the dynamic rate between the detected dynamic frames. By using interpolation, we can take into account crescendos (gradually increasing from soft to loud) and decrescendos (gradually decreasing from loud to soft) based on the dynamic rates on both sides of the current frame. For example, in going from loud to soft over time, the dynamic rate value will decrease as the current frame moves away from the frame in which the loud value originates and towards the frame in which the soft value originates.

#### 3.2. Mapping Method

Our technique for mapping between musical attributes and movements is designed to give the user maximum control over the animation process. For the method to work, the user must provide a character created in Alias-Wavefront's animation system, Maya, as well as fill out a script file. Mappings are used to determine what movements will be used in the animation, when they will be performed, and by which body part on the character. The system needs the names of the body parts, movements, and musical attributes that the user wants involved in the animation. The user specifies these names and their corresponding mappings in the script file, which is then used by the system to create the animation.

The system is designed so that the character being used in the animation is interchangeable. The script file allows the user to call the various body parts of the character rig by any term, rather than forcing the user to adhere to a strict naming policy. The system designer preprograms into the system simple primitive

movements that can be performed by the character. The musical attributes that are extracted from the sound file are also chosen by the system designer and cannot be altered by the user. The names of the body parts are taken directly from the character itself, as defined by the user. The body part names are case-sensitive, and must be spelled exactly as found in the scene or the system will not be able to identify them. The system designer predetermines the movement and musical attribute terms to be used in the script file. For example, the strong beat is referred to in the script file as `STRONG_BEAT` and the loud dynamics are referred to as `LOUD_DYNAMICS`. The twisting movement is represented by `TWIST`, while `THROW` corresponds to throwing an object in the air.

### 3.2.1. Primitive Movements

There are currently 10 primitive movements integrated into the system, with the capability for the system designer to add more. Some of the movements are reusable because they can be paired with several different body parts and still work well. For example, both the character's head and his waist can use the `SWAY` movement to move from side to side. The user controls this reusability, as she defines the movement-body part mappings. The primitive movements are as follows:

- `DRUM`
- `TAP`
- `THROW`
- `HEADBOB`
- `BEND`
- `SWAY`
- `UPDOWN`
- `LIFT`
- `TWIST`
- `JUMP`

The primitive movements are parameterized with respect to the extracted musical features (strong and weak beats, dynamic levels). This allows the system to create variations of the movements based on the content of the music. The dynamic levels drive the magnitude of a movement and the beat positions drive the length of a movement. For example, when the extracted dynamic level is high, indicating loud dynamics, the character bends down further (`BEND`) or throws an object higher in the air (`THROW`). If the dynamic level is low, indicating soft music, the character will throw the object only a short distance from his hand or bend over only slightly. The parameters provide a way to more accurately express the music through movement.

### 3.2.2. Body Part to Movement Mapping

Mappings occur in two ways. The first is between a body part and a movement, and the second is between a movement and a musical attribute. Mappings occur between a body part and a movement by specifying first the movement to be performed and then the body part that will be performing it. When a particular movement is specified for a body part, the body part will perform that movement for the given time interval. The format is as follows:

`MOVEMENT: bodyPartName`

Examples:

- `SWAY: UpperBody`
- `LIFT: RightArmCtrl`

Our system allows for the same movement to be repeated throughout the animation at different times and by different body

parts. For example, both of the character's arms can throw an object into the air at the same time: `THROW: RightArmCtrl, Ball`; `THROW: LeftArmCtrl, Ball2`. This encourages the user to take advantage of movements that work well in the animation by replicating them in different contexts. Some movements need specific objects in order to work properly. For example, the `DRUM` movement needs an object to drum on and the `THROW` movement needs an object to throw in the air.

- `DRUM: BodyPartName, DrumName;`
- `THROW: BodyPartName, ObjectName;`

These movements need special consideration when creating the script file because the system takes into account both the body part and the extra scene object when creating the mapping. The object must be listed in the script file after the body part so that the system can tell the difference between the two components of the movement and perform the correct motion. The length of a movement is based on the available time interval (the time between its current occurrence and its next occurrence). The larger the time interval, the slower the motion will occur, whereas the smaller the time interval, the faster the movement must occur to fit in.

### 3.2.3. Movement to Musical Attribute Mapping

Mappings between movements and musical features can drastically change the look of an animation. The musical features determine over what time period a movement will take place. By changing the musical feature a movement maps to, it will cause the movement to take place at a different time in the animation. The musical features that the system can map to are:

- Strong beats (`STRONG_BEAT`)
- Weak beats (`WEAK_BEAT`)
- Loud dynamics (`LOUD_DYNAMICS`)
- Soft dynamics (`SOFT_DYNAMICS`)

The format is as follows:

```
STRONG_BEAT
MOVEMENT: BodyPartName;
WEAK_BEAT
MOVEMENT2: BodyPartName2;
LOUD_DYNAMICS
MOVEMENT3: BodyPartName3;
SOFT_DYNAMICS
MOVEMENT4: BodyPartName4;
```

Similar to the first type of mapping where a movement can be mapped to more than one body part, the system can also map a movement to more than one musical feature. For example, a foot can tap to both the strong beat and the weak beat.

- `STRONG_BEAT`  
`TAP: LeftFootCtrl;`
- `WEAK_BEAT`  
`TAP: LeftFootCtrl;`

Correspondingly, a musical feature can be mapped to more than one movement. For example, the character can both throw an object and bob her head when the strong beat occurs in the music.

- `STRONG_BEAT`  
`THROW: RightArmCtrl, Ball,`  
`HEADBOB: HeadCtrl;`

The idea behind the mappings is to allow the user to experiment and observe how changing the mappings can drastically change the expression of an animation.

An animation where the same movements are being performed for its entire duration is mundane and uninteresting. Our system allows the user to divide the animation into smaller time periods and produce different mappings at each interval. In the script file, the user specifies the start frame for each interval and the corresponding mappings. This is repeated for the number of intervals chosen by the user. The length of each interval depends on the user, with a section continuing until the next specified start frame has been reached. The movements, music attributes, and body parts used in the animation can be changed in each interval, as well as the manner in which they are mapped to each other. This creates a more interesting animation where the character's movements change over time. The final script file can be seen in Figure 1.

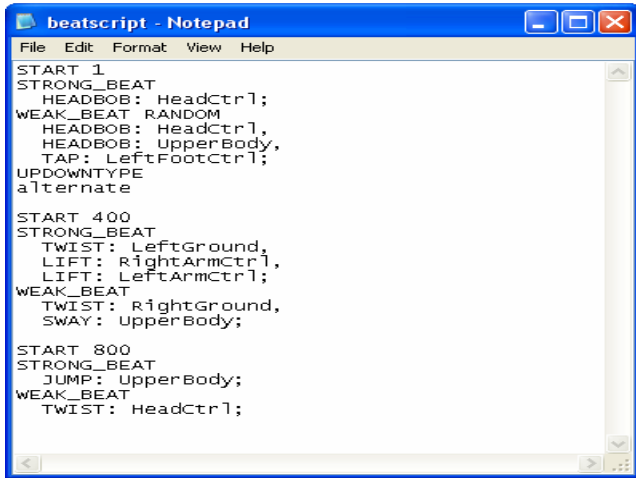


Figure 1: Example script file.

### 3.2.4. Primitive Movements

There are currently 10 primitive movements integrated into the system, with the capability for the system designer to add more. Some of the movements are reusable because they can be paired with several different body parts and still work well. For example, both the character's head and his waist can use the SWAY movement to move from side to side. The user controls this reusability, as she defines the movement-body part mappings. The primitive movements are as follows:

- DRUM
- TAP
- THROW
- HEADBOB
- BEND
- SWAY
- UPDOWN
- LIFT
- TWIST
- JUMP

The primitive movements are parameterized with respect to the extracted musical features (strong and weak beats, dynamic levels). This allows the system to create variations of the movements based on the content of the music. The dynamic levels drive the magnitude of a movement and the beat positions drive the length of a movement. For example, when the extracted dynamic level is high, indicating loud dynamics, the character bends down further (BEND) or throws an object higher in the air (THROW). If the dynamic level is low, indicating soft music, the

character will throw the object only a short distance from his hand or bend over only slightly. The parameters provide a way to more accurately express the music through movement.

### 3.2.5 Random Function

The script file ensures that the user has extensive control over the final animation, but difficulties can occur if the user is unsure of when movements should be performed, or if she cannot formulate good movement combinations. A random function was implemented to resolve this issue. The random function takes a number of movement-body part pairs as input and randomly chooses when each one will be used in the time interval. The user specifies the candidate movements in the script file, just as she would do normally, and indicates that she wants to use the random function by identifying it after the musical attribute it applies to:

MUSICAL\_ATTRIBUTE RANDOM

An example can be seen in Figure 1 under the first interval. The random function will first divide the interval into smaller subintervals. The number of subintervals corresponds to the number of candidate movements, and the length of each subinterval is chosen randomly. The movements that will occur in each interval are then chosen randomly by the system, and can be chosen more than once. Only one movement will be picked for each subinterval. An advantage of using the random function is that it is not predisposed to choose one movement over another, and therefore it can create unique combinations that the user might not otherwise discover on her own. The random function is called each time the system is restarted, so the user can create several different animations from the same script file.

### 3.2.6. Full Body Inverse Kinematics

In order to create a realistic animation we need a character whose movements will be a fairly faithful portrayal of real individuals. This can be achieved through a setup that Maya identifies as full body inverse kinematics (IK), where the inverse kinematics affect the entire body as a whole, rather than distinct body parts separately. Our system was developed in the Maya 6.5 environment, where full body IK was not an option for character rigging. Maya's new version, Maya 7, integrated full body IK, but there is currently not enough documentation provided for us to interface our plug-in with it. We created our own version of full body IK that works in both Maya 6.5 and Maya 7, bypassing the problems that occurred with Maya's version.

By following Maya's tutorial for character setup, we created a skeleton that includes joints and the IK handles to move them. We then parented control icons, which are essentially 3-D polygon shapes, to some of the joints and handles. The icons are more easily selectable by the user and provide excellent skeleton control. Control icons were added for the left and right arms and legs, the hips, the upper body, the ears, and the head of our bunny character. The user moves the objects in the scene, and in turn, these objects move the IK handles and joints to follow the specified motion. To create the semblance of full body IK, we added parent constraints between the arms, legs, and upper body. The upper body control icon is partially constrained to the movement of the character's arms and legs, giving the impression of full body motion. When an arm is pulled to one side, the entire upper body will follow it, just as it does in reality. If the character's leg twists, the rest of the body will twist as well, so that the character is facing the same direction as the leg. The

upper body can still move on its own, but the parent constraint gives the impression that the whole body is responding to a movement, rather than just a piece of it.

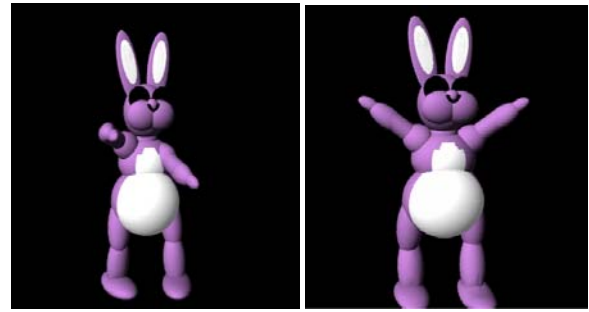
#### 4. Results

One of the main themes of our system is simplicity. Our goal is to create complex and interesting looking animations through the use of simple primitive movements. Our results display our achievement of this goal by providing exciting animations for songs of different styles and tempos. The video accompanying this paper includes four different songs from the styles of hip-hop, country, dance, and rock. Faster music is chosen more often because it generates more interesting animations. Each of the songs was chosen for its steady beat.

The first of the animations in the accompanying video uses a song by the artists Black Eyed Peas entitled 'Let's Get Retarded'. It provides a fairly fast-paced tempo with a strong beat rhythm. The animation's script file was chosen carefully by the user and she selected each time interval and the subsequent mappings. The purpose of this animation is to demonstrate the user control that our system provides. An especially interesting scene occurs within the interval of 0:36 seconds to 0:55 seconds. A TWIST motion is performed with both of the legs, and an UPDOWN motion is performed by both of the arms. The combination of these two simple movements works extremely well and ends up creating an exciting scene in the animation. The TWIST movement is performed by the legs in the interval 0:14 seconds to 0:28 seconds, but due to the difference in arm movement (LIFT), the entire motion looks dissimilar to the one previously mentioned. Figure 2 displays an example from each movement set. This serves to strengthen our belief that simple movements can be combined in different ways to create unique animations.

The random function is an integral part of our system and is used to add exciting and unexpected aspects to the results. We used the function extensively in the animation created from Kylie Minogue's 'Can't Get You Out of My Head'. The purpose of the random function is to create a different animation each time it is used. We demonstrated this by using identical mappings in two different interval periods: 1:14 to 1:22 and 1:38 to 1:51. The system created a different animation for each interval, despite the specified mappings being exactly the same for the strong and weak beats. The random function also chooses the amount of time a movement will be performed within an interval. The length of each subinterval determines how many movement changes occur, which results in scenes where not many movements are performed and scenes where several movements are performed. The random function was excellent at creating new and interesting combinations of movements, and if a certain combination appeared uninteresting it was easy to create a new combination by rerunning the system.

We used a slow country song for one of our animations to demonstrate the versatility of the system with respect to different types and tempos of music. Lee Ann Womack's 'I Hope You Dance' provided a good beat for the music analysis extraction algorithms. The tempo of the song directly affects the speed of the movements in the animation, so this animation was significantly slower in appearance than our other results. The animation used a combination of user chosen mappings and randomly chosen mappings. The second purpose of the animation

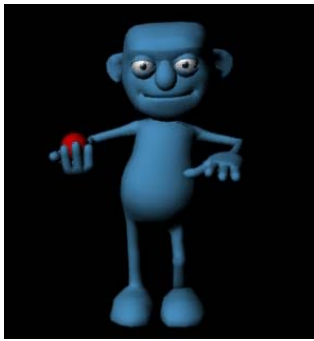


**Figure 2: Motion resulting from combining TWIST and UPDOWN movements (left) and TWIST and LIFT movements (right).**

is to show that different characters can be successfully used by the system. The Generi rig [Silke], as shown in Figure 3, has less body parts to animate than the bunny, and the setup of his character rig is quite different, but he still works effectively and gives a unique appearance to the movements.

We made use of Michael Jackson's song 'Billie Jean' to demonstrate the use of scene objects in an animation. This song was chosen for its quick tempo and noticeable drumming sounds. The most interesting part of this animation was the character's ability to interact with scene props. The bunny character was fitted with a drum and a pair of drumsticks. The DRUM movement was used extensively for pounding on the drum along with the beat. The dynamics level exclusively determined the strength of the movement, giving the animation a more realistic appearance. The dynamics also determine the height of the drumsticks when the character throws them in the air. The use of the scene objects produced an animation that was both interesting and exciting, and allowed for the use of primitive movements different from those used in the other animations.

The resulting animations show how a combination of simple primitive movements can create a complex looking and unique animation. As demonstrated by the random function, different combinations of mappings and time periods can create completely different scenes. The random function is able to provide interesting combinations that the user might not have considered on her own. The majority of the movements in the system are reusable by several different body parts, and because dynamics drive the magnitude of the movements, a movement can be more or less pronounced depending on the dynamics level at the point where it occurs. The user decides when to change the movements in the animation through the use of intervals, allowing her to tailor the animation to the song. The system is extremely versatile in changing the character involved in the animation, encouraging experimentation with not only the motion, but also the appearance of the animation.



**Figure 3: The Generi rig, created by Andrew Silke [Silke], demonstrates the interchangeability of the animated characters.**

The main limitation of our system can be found in the music analysis component. The beat onset algorithm restricts the success that the input music will have in the system. If the algorithm does not correctly detect the beats, the animation will fail to accurately depict the musical structure. The onset detection algorithm has been implemented in Matlab and is slow due to its numerous loops through the data. Another limitation lies in the process of adding new movements and musical features to the system. An experienced programmer must directly implement all new features in the system itself, as there is currently no method for an inexperienced user to add her own movements through Maya.

## 5. Conclusions and Future Work

Planned future work for our system includes improving the music analysis component. We plan to implement a more accurate beat onset detection algorithm that will work with a larger range of music, including music without a distinct beat sound. The addition of more musical attributes and primitive movements is also intended. Physical-based movements and reactions to movements, such as balancing on one leg, are not incorporated, but will be at a later date.

Our system provides a way for users to synchronize music with animation while creating a unique result that is well tailored to their input. A user is given extensive control over the system through the use of a script file, and she can choose movement combinations that best suit her needs and preferences. Simplicity is one of our design principles and we focus on using simple primitive movements to create complex animations. It is our hope that this system contributes to the marriage of Science and Art, and makes both more accessible to users of all types.

## 6. References

ALANKUS, G., BAYAZIT, A.A., and BAYAZIT, O.B. 2004. *Automated Motion Synthesis for Virtual Choreography*. Technical Report WUCSE-2004-66, Washington University.

BRAND, M. 1999. Voice puppetry. In *Proceedings of ACM SIGGRAPH 1999*, ACM Press/Addison-Wesley Publishing Co. 21-28.

CARDLE, M., BARTHE, L., BROOKS, S., and ROBINSON, P. 2002. Music-Driven Motion Editing: Local Motion Transformations Guided by Music Analysis. In *Proceedings of Eurographics UK 2002*, Leicester.

CAO, Y., TIEN, W.C., FALOUTSOS, P., and PIGHIN, F. 2005. Expressive Speech-Driven Facial Animation. In *ACM Transactions on Graphics*, 24, 1283-1302.

GOTO, M. and MURAOKA, Y. 1994. A Beat Tracking System for Acoustic Signals of Music. In *Proceedings of ACM Multimedia 1994*, San Francisco, 365-372.

GOTO, M. 2001. An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds. *Journal of New Music Research*, 30, 159-171.

LEE, H.C., and Lee, I.K. 2005. Automatic Synchronization of Background Music and Motion in Computer Animation. *Eurographics 2005*, 24, 353-362.

SILKE, A. [http://andrewsilke.com/generi\\_rig/generi\\_rig.html](http://andrewsilke.com/generi_rig/generi_rig.html)

TAYLOR, R., TORRES, D., and BOULANGER, P. 2005. Using Music to Interact with a Virtual Character. 5<sup>th</sup> International Conference on New Interface for Musical Expression, Vancouver, 220-223.

TZANETAKIS, G., ESSL, G., and COOK, P. 2001. Audio Analysis using the Discrete Wavelet Transform. In *Proceedings WSES Int. Conf. Acoustics and Music: Theory and Applications*, Skiathos, Greece.