

# Cyclification of Human Motion for Animation Synthesis

Amr Ahmed <sup>†</sup>, Farzin Mokhtarian, and Adrian Hilton

Centre for Vision, Speech, and Signal Processing,  
Department of Electronic Engineering,  
University of Surrey, Guildford, Surrey GU2 7XH, UK.  
Fax: +44 (0) 1483 686031 Tel.: +44 (0) 1483 686030

---

## Abstract

*In this paper, new techniques are introduced for matching the motion cycle boundaries by using simple animation processing algorithms based on some observed characteristics from different families of cyclic motions. Matching the cycle boundaries is one of the essential stages in generating a stream of animation of cyclic human movements. If the cycle boundaries are not matched, noticeable artifact and flickers will appear at these boundaries regardless of how realistic is the available motion cycle. The proposed techniques are developed to modify the motion cycle even in the case, where only one cycle is provided. These techniques do not only match the cycle boundaries but also correct other parts of the cycle and motion constraints. The techniques are computationally efficient as they work directly on the animation data (without conversion to intermediate representations such as spline or time-frequency transforms). Moreover, many aspects of animation (including joint limits and important motion constraints) are preserved as the modification is guided by the original motions.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism — Animation

---

## 1. Introduction

Automatic generation of long realistic animation sequences has attracted the attention of researchers for the last few years. Although motion captured animation has proven to produce realistic animation, due to some limitations, it is not always the case that the whole sequence can be captured. Especially for cyclic movements such as walk and run. Motion captured animation can be used to build a database of basic motions which can be used for character animation. The challenge is to find suitable tools and techniques to edit, modify, and re-use these data.

Cyclic movements represent a major sector of human movements. This includes many of our daily activities such as walking, running, waving, and reaching. In order to generate longer sequence of animation for such cyclic movements, trivially, motion cycles are to be repeated or concatenated to each other.

Concatenating cycles is a trivial task assuming that the

available cycles are complete and perfect (i.e. the cycle start and end postures are perfectly matched) which is rarely the case in human motions. Due to the nature of human being and some internal and external factors (including muscle forces, torques and balance mechanism), human cyclic movements are usually not perfectly cyclic.

Matching the cycle boundaries -the process known as the cyclification <sup>1</sup>- is one of the essential stages in generating a stream of animation of cyclic human movements. If the motion cycle boundaries are not matched (which is usually the case in the real human movements), the generated sequence will possess noticeable artefacts and flickers at these boundaries regardless of how realistic is the available motion cycle.

In this paper, we present new techniques for matching cycle boundaries (cyclification) that employ simple and fast animation processing techniques such as blend, mirror, and reverse of motion cycle's parts. While the first technique assumes that there is enough data (i.e. motion frames) to correct the cycle, the other techniques are based on the availability of only one motion cycle. In such cases, where only

---

<sup>†</sup> email: A.Ahmed@eim.surrey.ac.uk

one motion cycle is provided, the technique benefits from cycle characteristics in order to achieve the cyclification.

In the next section, we give an overview of related work. Our proposed techniques are discussed in section 3. Section 4 shows our cyclification results of different types of cyclic human movements. Then, the conclusion and future work are presented in section 5.

## 2. Related Work

Motion capture animation and its editing and re- using, as techniques for realistic human animation, have been the focus of many research <sup>2, 3, 4, 5, 6</sup>. However, the cyclification problem has received only limited investigation. In this section, we give an overview of related key research in motion cyclification.

Within the context of motion captured animation and editing, Sudarsky et. al. <sup>7</sup> introduced a cyclification operator. They represent motion curves as splines. Given a periodic curve, the cyclification operator is the one that generate a new spline which has the same values at the cycle period and its multiples. If the given curve is not accurately periodic, another processing is needed such as blending. In <sup>8</sup>, Golam et. al. also used blending to cyclify their unit cycle, which consisted of two or more actual motion cycles. In their analysis, the upper and lower body halves are decoupled and processed independently. This introduces a phase difference that needs correction. The above techniques have in common the assumption that there are at least two actual motion cycles available. Although they are similar to our first approach, the difference is that we don't require a second complete cycle to be available. Only a small part of it (either at the beginning or end). Also, our unit cycle is the actual cycle itself and body parts are not decoupled.

The cyclification problem has also been addressed within the motion transition research. Rose et. al. <sup>9</sup> presented a cyclification method based on fitting a least squares cyclic B-spline to a modified motion curve. The modified motion curve is constructed according to minimising the difference of position, speed, and acceleration between the cycle boundaries. However, using B- spline fitting may lead to exceeding the joints limits, and violating some kinematics constraints as mentioned in <sup>7</sup>. These constraints need to be corrected later on using IK algorithms and enforcing the joint limits. Gleicher et al. <sup>10</sup> used a similar approach based on finding a common pose between similar frames from different motion clips. Then the motion clips are modified - using displacement map- to start and/or end with the common pose. This results in violating some motion constraints at the modified parts. They concentrated on the foot-plant constraints only and employed a correction algorithm from <sup>11</sup>. In our techniques, the original motions are synchronised before blending to ensure that the constraints timing and important events are aligned. This automatically preserves the

motion constraints (provided that they are correctly defined in the original motions).

The work of Silva et. al. <sup>1</sup> is the most relevant to this paper. Their curve cyclification algorithm is based on using a windowing technique (LCT; Lapped Cosine Transform) to accomplish time warping whilst preserving frequency contents. This includes automatic detection of the cycle period through the circular autocorrelation function. To apply this algorithm on motion of articulated figures, some considerations should be taken into account. First, the large number of segments and connecting joints in the articulated figure hierarchy and the multiple DOFs at each joint results in large amounts of data that should be processed. Second, and more importantly, synchronisation between body parts in the human motion and dependencies between different joints or group of joints, which should be preserved. Also, in their algorithm, selecting the window size is critical as it should be equal to the cycle period. If it is larger or smaller than the cycle period, it will result in discontinuity and/or noise.

The main advantages of techniques proposed in this paper are that they are applied directly to the animation data (joints DOFs) without fitting (spline,...etc.) or any time-frequency transforms, which reduces the processing time. Instead, these techniques benefit from some observed characteristics of motion cycles for different families of motions and employ these observations with simple animation processing algorithms to match the cycle boundaries. The proposed techniques preserve many aspects of animation such as joints limits and constraints because they are guided by original motions and observing their constraints (through the key events). Also, a simple framework for automatically detecting the gait phases timing is briefed in section 3.1.

## 3. Our Cyclification Techniques

In this section, the proposed cyclification techniques are presented. Section 3.1 shows the cyclification guided by part of additional motion cycle. Self-cyclification techniques are described in section 3.2.

### 3.1. Cyclification guided by part of additional motion cycle

The first and simplest technique tries to treat the cycle by another part of existing data using a synchronised blending technique. Our motivation in this technique is that in many cases, when we capture data, we usually capture more than one cycle of the cyclic motion (depending on the capturing volume), before and/or after the cycle we are focusing on. Due to some measurement noise and missing data, captured movement data may be slightly corrupted. The extra data (or some parts of it) can be utilised carefully to modify the cycle boundaries and make them matched. It can also be used in correcting some other corrupted motion constraint [feet-

floor touch/lock,...etc.] near the edges, which could avoid repeating the motion capture process.

In our system, the unit cycle consists of only one actual motion cycle. So, any other part of a cycle, preceding or following our cycle, is considered as additional data that can be utilised in this technique. It is worth mentioning that our technique does not require a complete additional cycle. Just part of it (up to one fifth of the cycle time) is enough.

Given a motion cycle  $MC[1 : m]$  and an additional part of motion cycle  $pMC[1 : n]$ , where  $m$  and  $n$  are the number of segments (divided by the key event times; the times of important events and/or postures during the motion which imply motion constraints) of  $MC$  and  $pMC$  respectively, the following procedure is applied to match the cycle boundaries:

1. Assuming that  $pMC$  is part of the preceding cycle of  $MC$ , the first step is to get its corresponding part from  $MC$ :

$$sMC = MC[m - n + 1 : m]$$

If  $pMC$  is part of the following cycle, then:

$$sMC = MC[1 : n]$$

2. Blend the aligned segments:

$$cMC1 = SyncBlend(sMC, pMC, X)$$

Where  $X$  is the blending vector such as

$$X[1 : length(sMC)] = [1 : 0]$$

3. Construct the corrected cycle:

$$cMC = Append(MC[1 : m - n + 1], cMC1)$$

If  $pMC$  is part of the following cycle, then:

$$cMC = Append(cMC1, MC[n : m])$$

Due to the importance of the key-event times, a simple framework is developed for automatically detecting the gait phases timing (which represents the majority of cyclic movements). Given a motion clip  $M$ , the procedure below is used to automatically detect the key-event times (including cycle boundaries).

1. Find distance between the left and right feet at each frame of the given motion:

for  $i = 1 : NumOfFrames$

$$D(i) = CalcFeetDistance(Frame(i));$$

end;

2. Scan the distance vector  $D$  for the local minimums and maximums:

$N = 1;$  //number of found key-event times

for  $i = 2 : NumOfFrames$

$$KET(N) = FindCriticalPoint(D(i : end));$$

Update  $i;$  // scan after last Max/Min

$$N = N + 1;$$

end;

3. From the list of local Max. and Min., the key event times can be detected as follows:

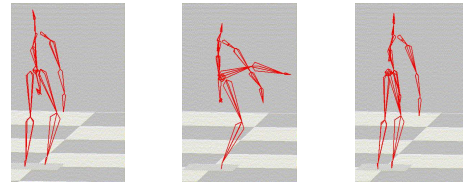
- The first local maximum is corresponding to the touching of 1st foot. [KET1]
- The first local minimum is corresponding to the mid-swing of the other foot.[KET2]
- The second local maximum is corresponding to the touching of 2nd foot. [KET3]
- The second local minimum is corresponding to the mid-swing of the 1st foot.[KET4]
- The 2nd local maximum is again corresponding to the touching of 1stfoot. [KET5]

The above detection procedure is successfully applied to many gait motions with normal and up-normal cycles (like limb and corrupted cycles). For some noisy data, applying simple threshold for accepted local max/min is found to be useful.

The above cyclification technique assumes that we have enough data to be utilised to modify our cycle. It also assumes that the extra data is directly preceding or following the data in focus (our cycle). However, if this extra data is not available (for example we don't have access to the raw captured data, the data is corrupted or not preceding/following our cycle) that approach may not have much to do. This is where the other proposed techniques can help.

### 3.2. Self Cyclification

The second technique works on the provided cycle only (assuming no extra data is provided). This technique utilises some animation processing techniques to manipulate different parts of the given cycle. As there is no additional animation data available in this case, the technique tries to benefit from the cycle characteristics of human movement. In this section, we describe our technique on two major families of cyclic human movements.



**Figure 1:** Example of 'Reversed Half-cycle' motion category (Kicking)

#### 3.2.1. "Reversed Half-cycle" Category

In this category, it has been observed that the second half of the cycle is similar to the reverse of the first half of the

Input:	One motion cycle ' $M$ ' The corresponding Key Event Times ' $KETS$ '
Output:	Corrected motion cycle with its boundaries matched together ' $Mcm$ '
Procedure:	<ol style="list-style-type: none"> <li>1. Get the 1st half of the provided cycle. <math>A = M[1 : KETS(CycleKET/2)];</math> // <math>CycleKET</math> = No. of KETs/Cycle</li> <li>2. Get the 2nd half of the provided cycle. <math>B = M[KETS(CycleKET/2) : KETS(CycleKET)];</math></li> <li>3. Get the reverse of 1st half <math>Brv = Reverse(A, Root);</math></li> <li>4. Prepare the blending factor (<math>X = 1 \rightarrow 0</math> over the original 2nd half) <math>X[1 : length(B)] = [1 : 0];</math></li> <li>5. Blend original 2nd half &amp; aligned, reversed 1st half. <math>Bcm = SyncBlend(B, Brv, X);</math></li> <li>6. Concatenate both halves; Original 1st half, and matched 2nd half <math>Mcm = Append(A, Bcm);</math></li> </ol>

**Figure 2:** Self Cyclification algorithm for 'Reversed Half-cycle' motion category (Sec. 3.2.1)

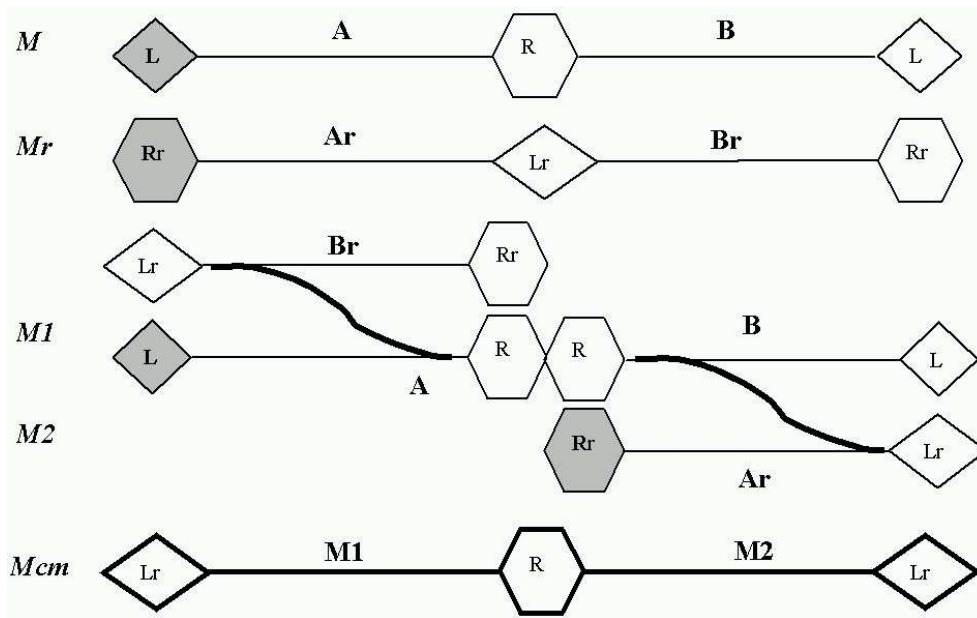
Input:	One motion cycle ' $M$ ' The corresponding Key Event Times ' $KETS$ ' ( $CycleKET$ = No. of KETs/Cycle)
Output:	Corrected motion cycle with its boundaries matched together ' $Mcm$ '
Procedure:	<ol style="list-style-type: none"> <li>1. Get the mirror of the provided cycle (around its forward direction). <math>Mr = Mirror(M);</math> <math>rKETS = Mirror(KETS);</math></li> <li>2. Define the different parts of the given cycle (and its mirror) that is going to be manipulated: <math>A = M[KETS(1) : KETS(CycleKET/2)];</math> // 1st half of <math>M</math> <math>B = M[KETS(CycleKET/2) : KETS(CycleKET)];</math> // 2nd half of <math>M</math> <math>Ar = Mr[rKETS(1) : rKETS(CycleKET/2)];</math> // 1st half of <math>Mr</math> <math>Br = Mr[rKETS(CycleKET/2) : rKETS(CycleKET)];</math> // 2nd half of <math>Mr</math></li> <li>3. Construct the corrected 1st half '<math>M1</math>' as follows: <math>X[1 : length(Br)] = [1 : 0];</math> // prepare <math>X = 1 \rightarrow 0</math> over the length of <math>Br</math> <math>M1 = SyncBlend(Br, A, X);</math> // SyncBlend; start by <math>Br</math> &amp; end with <math>A</math></li> <li>4. Construct the corrected 2nd half '<math>M2</math>' as follows: <math>X[1 : length(B)] = [1 : 0];</math> <math>M2 = SyncBlend(B, Ar, X);</math> // SyncBlend; start by <math>B</math> &amp; end with <math>Ar</math></li> <li>5. Construct the corrected cycle by concatenating both halves (<math>M1</math> and <math>M2</math>) <math>Mcm = Append(M1, M2);</math></li> </ol>

**Figure 3:** Self Cyclification algorithm for 'Mirrored Half-cycle' motion category (Sec. 3.2.2)

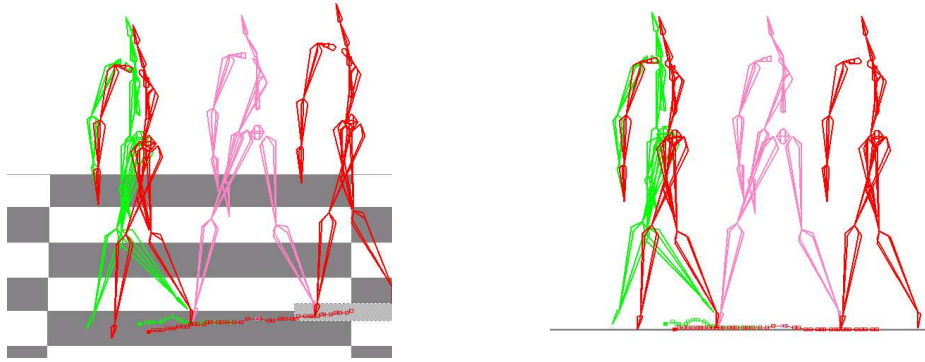
same cycle. 'Kick' and 'jump-turn' are some examples of this category of motions (see figure 1 for kicking example).

Knowing these characteristics, some simple and fast animation processing techniques, such as Reverse and Synchronised Blending, are utilised to modify the cycle's edges in order to match its boundaries for seamless appending and/or transition as shown in figure 2. The synchronised blending first align the corresponding motion segments together using time-warp to maintain motion constraints

As mentioned in previous research, in dealing with the articulated figure, the root joint -which has 6 DOFs- usually needs special attention as it represents the overall translation of the whole hierarchy. Depending on many factors (including the motion type and the required cyclification), the root joint information can either receive the same processing as the rest of the joints or not. This option is implemented in the 'reverse' module as indicated by 'Root' in step 3 of the algorithm. For example, in 'Kick', the cyclification is required to



**Figure 4:** This diagram explains the self-cyclification algorithm for 'Mirrored Half-cycle' category



(Original avatar [A:Green] and corrected avatar [B:Red])

**Figure 5:** Corrected walking cycle (by part of preceding cycle. Sec. 3.1)

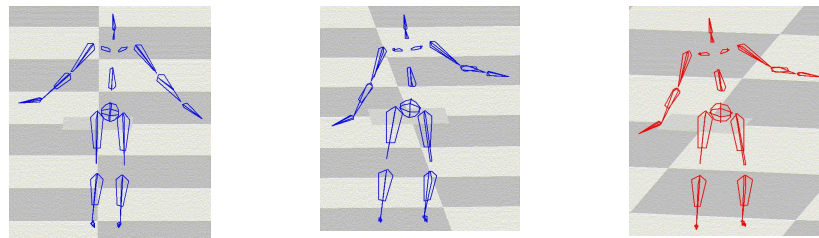
maintain the same root position and orientation at both ends of the cycle. To achieve that, the root joint is processed normally with the rest of the joints. However, this condition is not required in the 'Jump-Turn' as the initial and final root positions and orientations are expected to be different. So, the root joint just follows the original cycle.

### 3.2.2. "Mirrored Half-cycle" category

The observed characteristic of the motion cycle of this category is that the second half of the cycle is similar to the mirror of the first half of the same cycle. A very common example of this category of motions is 'Walk' and 'Run'.

Based on these characteristics, a similar approach that applies Mirror and Synchronised Blending techniques to update and match the cycle boundaries is developed. Figure 3 presents the proposed cyclification algorithm for this category of motion. The diagram in figure 4 also explains the algorithm.

Given a motion cycle 'M', its mirror is calculated and the two halves (of both motions) are separated. The first half of the corrected cycle is constructed by blending between the second and first halves of the mirrored cycle and the original cycles respectively. Similarly, the second half of the cor-



Original final posture

Initial posture

Corrected final posture

**Figure 6:** Corrected jump-turn cycle (self-cyclification technique. Sec. 3.2.1)

(a) Original Kick cycle

(b) Corrected Kick cycle

**Figure 7:** Corrected kicking cycle (self-cyclification technique. Sec. 3.2.1)

rected cycle is constructed by blending between the second and first halves of the original cycle and its mirror respectively. Then both corrected halves are appended together to construct the corrected cycle with matched boundaries.

#### 4. Results

In this section, some results are shown for the proposed cyclification techniques. In figure 5, a corrected walking cycle is shown. It has been corrected using part of the cycle just preceding the shown cycle (using the technique in section 3.1). It can be noticed that not only the cycle boundaries has been matched but also the feet touching to floor is corrected towards the end of the cycle

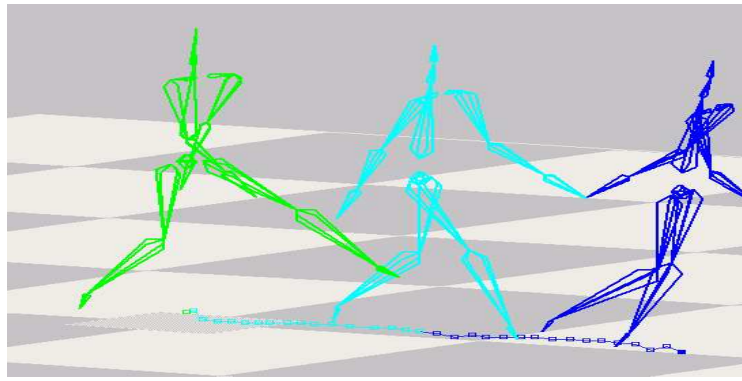
Figure 6 shows the initial and final frames of a jump-turn cycle. The difference in the posture between these two frames produces noticeable flickers when the cycle is repeated. Final frame of the corrected cycle (using the self cyclification technique described in section 3.2.1) is also presented in the figure.

A similar example is shown in figure 7 for kicking cycle. Again the initial and final postures of the original

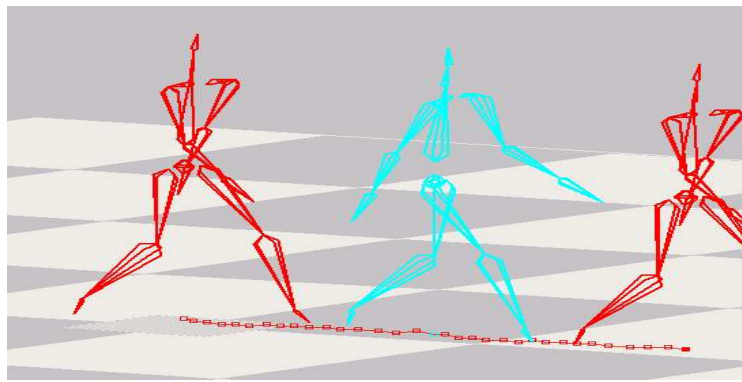
and corrected cycles are presented. A more interesting example is portrayed in figure 8 where the original cycle is not only having unmatched boundaries but also corrupted in some regions near the boundaries as depicted in figure 8-a. The self-cyclification technique, described in section 3.2.2, has been able to successfully correct both cycle boundaries and the corrupted regions of the motion cycle. Without this correction, the motion could not be used to achieve a satisfactory cycle. Some demos of these results can be found in the following URL: "[http : //www.ee.surrey.ac.uk/Personal/A.Ahmed/EG03/Demos/](http://www.ee.surrey.ac.uk/Personal/A.Ahmed/EG03/Demos/)"

#### 5. Conclusion

This paper presents new techniques for matching the motion cycle boundaries by using simple animation processing algorithms based on observation of characteristics for different categories of cyclic motions. The first technique uses a small part of a cycle just preceding or following the provided cycle in order to match the boundaries. The other techniques assume no extra data is available and modify the given motion cycle based on its characteristics. The proposed techniques do not only match the cycle boundaries but also correct other



(a) Original corrupted walking cycle



(b) Corrected walking cycle

**Figure 8:** Corrected walking cycle using the self cyclification technique (Sec. 3.2.2). Note that both boundaries and corrupted regions are corrected

parts of the cycle and motion constraints where captured data is corrupted.

The presented techniques are computationally cheap. They work directly on the animation data (without conversion to intermediate representations such as spline or time-frequency transforms). Moreover, they maintain many aspects of animation (including joint limits and important motion constraints) as the modification is guided by the original motions (according to the key event times). Moreover, a simple framework for automatic detection of key-event times is developed.

Our aim for future work is to extend these algorithms to other motion categories. Another interesting enhancement could be the automatic type classification of the given motion cycle.

## References

1. F. Silva, L. Velho, J. Gomes, and S. Golden-Stein. Motion cyclification by time x frequency warping. In *Proceedings of SIBGRAPI'99, XII Brazilian Symposium of Computer Graphics and Image Processing*, pages 49–58, 1999.

2. J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figure. In *Proceedings of SIGGRAPH'99*, pages 39–48, 1999.
3. W. Sun. *Modelling Bipedal Locomotion using Wavelets for Figure Animation*. PhD thesis, Department of Computer and Information Sciences, De Montfort University, UK, May 2000.
4. M. Brand and A. Hertzmann. Style machines. In *SIGGRAPH'00. Proceedings of the 27th Annual ACM Conference on Computer Graphics and Interactive Techniques*, pages 183–192. ACM Press/ Addison-Wesley Publishing Co., 2000.
5. K. Pullen and C. Bregler. Motion capture assisted animation: Texturing and synthesis. In *SIGGRAPH'02. Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 501–508, 2002.
6. Amr Ahmed, Farzin Mokhtarian, and Adrian Hilton. Parametric motion blending through wavelet analysis. In *Eurographics'01. Short Presentations Proceedings of Eurographics 2001*, pages 347–353, 2001.
7. S. Sudarsky and D. House. Motion capture data manipulation and reuse via b-splines. In *Proceedings of CAPTECH'98*, 1998.
8. Ashraf Golam and Cheong Wong Kok. Constrained framespace interpolation. In *Proceedings of Computer Animation*, pages 61–73, 2001.
9. C. F. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH '96. Proceedings of the 23rd Annual ACM Conference on Computer Graphics and Interactive Techniques*, pages 147–154, 1996.
10. Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap-together motion: Assembling run-time animation. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics*, April 2003.
11. Lucas Kovar, John Schreiner, and Michael Gleicher. Footskate cleanup for motion capture editing. In *ACM Symposium on Computer Animation 2002*, July 2002.