Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Synchronization for dynamic blending of motions

S. Ménardais[1], R. Kulpa[1,2], F. Multon[1,2] and B. Arnaldi[1]

[1] SIAMES Project - IRISA, Rennes, France
[2] LPBEM - University of Rennes 2, France

Abstract

In this paper we present a new real-time synchronization algorithm. In dynamic environments, motions need to be continuously adapted to obtain realistic animations. We propose an advanced time warping algorithm to synchronize such motions. This algorithm uses the sequence of support phases of the motions. It also takes into account the priority associated to each motion. It is based on an algebraic relation to detect incompatible motions and to select elements of the sequence to be enlarged. The resulting time warping function can be non-derivable so it is corrected by using a cardinal spline interpolation. In this paper, we demonstrate that our algorithm always finds at least one solution. This synchronization module is part of a complete animation engine called MKM already used in production.

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

1. Introduction

Realistic human animation is widely used in multimedia applications such as video games and movies. At the same time, it remains a difficult and labouring task. The most popular way to achieve such animations is the use of captured motions. Indeed, they implicitly respect the dynamic laws. Moreover, it is now associated to motion editing techniques that make the data set more flexible. One of these techniques, motion blending, allows the creation of complex animations from short captured motions. Unfortunately, the blending algorithms can produce unexpected motions. For example, if a movement with the left foot on the ground is blended with another with the right foot on the floor, the resulting motion will not respect any of the support constraints of the two original ones. Consequently, the foot will slide on the floor. To avoid such problems, the motions have to be previously synchronized.

The work presented in this paper is placed in a context of dynamic environments. Hence, the motions used during the animations are interactively adapted. They can also be adapted to external constraints specified by the user in real-time (e.g., height of a chair while sitting). Moreover, we want to animate several characters with different morphologies in real-time. To achieve all these requirements, we propose a new real-time synchronization algorithm based on the blending module described in [MMKA04]. It uses an advanced time warping technique based on the support phases of the motions. It also uses priorities defined interactively for each motion. In addition to priorities, the user can activate or deactivate the motions by using the start/stop commands at any time. The algorithm automatically synchronizes motions and delays the start of a new one if it cannot be immediately synchronized. Since our work is already used by industrial partners, this intuitive control of the synchronization is really important. The current areas of application of our works are videogames, interactive fictions and virtual reality.

The remainder of this paper is organized as follows. We first review related works in section 2. In Section 3, we describe the framework in which our synchronization algorithm is introduced. In Section 4, we define the creation of the sequence of the support phases associated to motions and its interpretation to detect incompatible motions and then, in Section 5, we detail the synchronization algorithm using these previously defined sequences. We then present results in Section 6 and conclude with a brief discussion in Section 7. At the end of the paper, appendix A is given to make a recursive proof of our algorithm.

delivered by EUROGRAPHICS DIGITAL LIBRARY
www.eg.org diglib.eg.org

2. Related works

Synchronizing motions has been studied using two main approaches. The first one is to identify correspondences between motions. These correspondences are used to find a unique synchronization. The second one is to find transitions between postures of motions. The transitions are then a set of possible solutions. A cost function can be used to find the most appropriate solution.

Correspondences between motions were first identified using frequential information. The first synchronization works were limited to specific motions like locomotion. Unuma et al. [UAT95] used Fourier techniques to interpolate and extrapolate walking and running motions to give emotions. Bruderlin et al. [BW95] made a multiresolution filtering algorithm to modify motions using the low, middle or high frequencies. They also applied non-linear time warping to make temporal correspondences between motions. Finally, Guo and Robergé [GR96] proposed an interpolation method based on a set of keyframes. They synchronized locomotions by modifying the temporal parameters of the keyframes. The main limitation of such studies is that motions have to be periodic and of the same kind (e.g., locomotion).

Recent techniques have automatized the identification of correspondences between a limited number of motions. To achieve this, Ashraf et al. [AW00] used the kinematic information contained in motions. These different kinds of information are for example zero-crossing speed or angular minima. They extended their work using a multilayered semantic representation of the motions [AW03]. The drawback of such methods is that motions must be similar regarding dynamic information. As we want to synchronize really different motions that are interactively adapted, such techniques are not suitable. Moreover, these works were based on joints angles. They imply the use of identical morphologies.

Other authors have tried to solve a global optimization problem in order to synchronize (and adapt) general human motions with spacetime constraints [RGBC96]. These techniques, based on non-linear systems, involve problems such as local minima and lack of interactivity since complete motions are needed. Moreover, the specification of the constraints that are dependent on the kind of motions is difficult.

More recently, new probabilistic algorithms were widely used to find transitions between motions in a database and create a motion graph [KGP02]. These transitions mainly preserve dynamics by associating similar skeleton postures. The database is precomputed and allows interactive animations. Some authors [LCR*02], [BH00] use hidden Markov chains to define these transitions. Nevertheless, the main drawback of this kind of technique is that transitions are computed automatically. Forecasting which motions will be used to transit from one motion to another is difficult. Moreover, bad transitions can be chosen if motions have very different dynamic information or if the database is too limited. Wang and Bodenheimer [WB03] proposed a set of optimized weights for the cost function used by Lee et al. [LCR*02]. To minimize the choice of bad transitions, Ashraf et al. [AW01] tried to correct the problem by dividing the human skeleton into two groups: the upper and the lower body. Schödl et al. [SSSE00] used this kind of technique to analyze streams of video and drive these streams through high-level user input. Despite these improvements, this kind of work often requires the use of huge databases to ensure a minimal set of good transitions between postures. Nevertheless, such databases are not suitable with highly interactive animations. Indeed, interactivity deals with a large number of postural configurations. Moreover, they are not suitable for crowd animations either. Captured motions implicitly contains their own morphology. Each motion must be used on characters with the same kind of morphology. We avoid this problem using a normalized representation of the skeleton that allows to share automatically a same motion with different characters (and different morphologies) [MMKA04]. Furthermore, adding motions in the database requires the computation of the set of all the possible transitions with the other motions. Its cost is directly dependant on the size of the database and consequently increases with it. Finally, these databases contain predefined motions. Interactively adapted motions cannot be managed with such techniques.

After the search in the database, real-time adaptations were proposed in order to enhance the interactivity of such systems. Lee et al. [LCR*02] used time warping to handle the time of the animation and to adapt motions to the ground. Kovar et al. [KG03] proposed registration curves to automatically determine the relationships between timing and local coordinate frame in order to animate a character who follows a specified trajectory. Current studies still remain limited since they synchronize fixed motions.

The study presented in this paper deals with a new synchronization algorithm based on time warping [WP95], [RCB98] with additional information: the sequence of support phases and a set of priorities associated to the motions. Previous studies [AW01] have emphasized the importance of the sequence of support phases. Zeltzer et al. [Zel82], [Zel86] have used finite state machines where states represent the different phases of the locomotion cycle. For example, this kind of information avoids the blending of two motions: one with the right foot on the ground and one with the left foot on the ground. It also allows to synchronize motions of same kind (such as locomotion [MFCGD99], [PSS02]).

Using the sequence of support phases means dealing with concurrent constraints. There are two classical solutions: the use of ordered priorities and the computation of weights. The first one classifies constraints using their priorities and tries to respect the highest one first [BBET97]. Consequently the lower constraints may not be respected. The second one interpolates constraints using a weighted sum [MMKA04]. This last solution does not respect any of the original constraints but offers an intermediate solution that ensures continuity. In this paper, we synchronize the support phases of the different motions to ensure that we have a sequence of unique constraints before blending the motions. Indeed, ordered constraints are not suitable because the support constraints for one leg is like a boolean: a leg is supporting the body or not. Using weights does not ensure the respect of the support phases.

3. Overview

The synchronization module presented in this paper is part of a complete real-time kinematic animation engine called MKM (Manageable Kinematics Motions). Its goal is to allow and simplify the animation of different kinds of humans in dynamic environments. It uses morphological and environmental adaptations. The latter are done in real-time using an adimensional and normalized skeleton [MMKA04]. This representation of the motion automatically allows it to be independent from the original morphology and then to be shared between a lot of characters with different skeletons. The environmental adaptations are made using kinematic constraints and are computed after the blending to ensure they are respected (e.g., contact of the hands during applause). The normalized skeleton is described in cartesian space allowing an easier and more intuitive set of constraints. Finally, this representation limits the number of parameters to compute at each time and consequently is less computation expensive. With such a technique, we can animate a large number of characters in a dynamic environment.

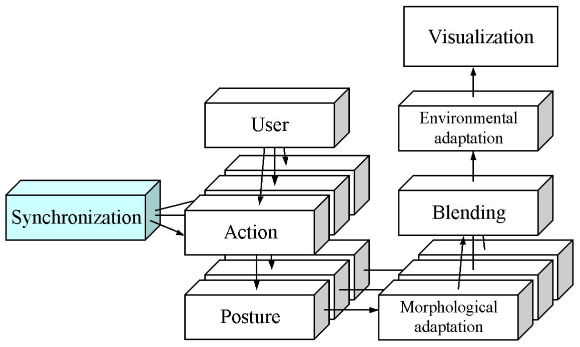


Figure 1: *MKM real-time animation engine. Synchronization module ensures that actions can be blended.*

As shown in Figure 1, the blending module blends motions (hence postures at a given time) issued from actions. These actions (A_i in the remainder of this paper) can simply replay a motion or can adapt it to parameters as parametric models do. Hence, information on used motions cannot be precomputed since they can be modified in real-time. Indeed, these actions are activated and deactivated at any time by using the start/stop commands and priorities as shown in Figure 2 and detailed in [MMKA04]. In this figure, $t1$ is the starting time of the action (start command) and $t3$ its ending time (stop command). $t2$ (resp. $t4$) is the complete activation (resp. deactivation) of the action.

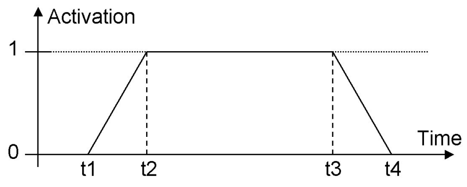


Figure 2: *Activation and deactivation of actions. $t1$ is the starting time of the action and $t3$ its ending time. $t2$ (resp. $t4$) is the complete activation (resp. deactivation) of the action.*

To manage such actions, we introduce a new synchronization algorithm based on the sequence of the support phases of the motion. The output of an action is then the motion and its associated sequence (S_i in the remainder of the paper where i is the index of its action A_i).

4. Definition and analysis of the sequence

Our synchronization algorithm is based on sequences of support phases. The first part of this section deals with the representation of these sequences and the detection of incompatibilities between them. The second part explains how the sequences are generated using this representation. Finally, the last part describes how the detection process manages motions with a different number of elements.

4.1. Algebraic relation

To determine if nA actions can be blended or not, we introduce an algebraic relation $\bigoplus$. It uses the sequence of support phases S_i associated to the action A_i. The latter is defined as a sequence of elements from the set $FS = \{RS, LS, DS, \emptyset, Err\}$ (Figure 3 shows a locomotion cycle described using our representation) where *RS* (resp. *LS*) is the period of time when the right (resp. left) foot is on the ground, *DS* is used to define that the two feet are on the ground (double support), $\emptyset$ is during a jump (no foot contact on the ground) and *Err* is the result of two not-synchronizable motions. Let $S_i(k)$ be the k^{th} element of this sequence.

The algebraic relation $\bigoplus$ from $FS^2 \rightarrow FS$ is defined such as a$\bigoplus$b represents the resulting strike constraints from strike a and strike b (cf. Figure 4).

This algebraic relation has been defined following these rules:

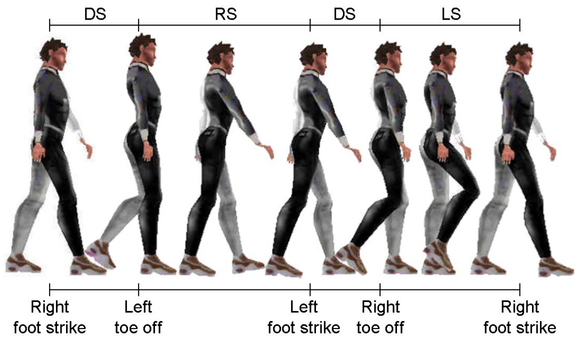


Figure 3: *Definition of support phases of a locomotion. The feet events are under the figure and our representation is above.*

$\oplus$	DS	RS	LS	Err	Ø
DS	DS	RS	LS	Err	Ø
RS	RS	RS	Err	Err	Ø
LS	LS	Err	LS	Err	Ø
Err	Err	Err	Err	Err	Ø
Ø	Ø	Ø	Ø	Ø	Ø

Figure 4: *Algebraic relation that determines if motions can be blended.*

- *DS* is the neutral element because our blending module uses a neutral motion with two feet on the ground. This motion always has the lowest priority and is overridden by any other motion. It is useful when there is no active motion to play. For example, if a locomotion with the sequence $S_i = \{RS, DS, LS, DS\}$ is started, our algebraic relation ensures that its sequence overrides the neutral motion's sequence.
- $\emptyset$ is the absorbing element. Contrary to the previous element, it represents the jump motion during which support constraints are disabled.
- Err is the absorbing element of the subset $\{RS, LS, DS\}$. It means for example that even if only two motions are not synchronizable amongst several others, the relation will set an error Err.
- *LS* and *RS* are the elements that are completely incompatible and that set an error Err if they are both to be verified at the same time.

As our relation is commutative, two motions A_i and A_j are synchronized if and only if:

$$\forall k, \quad S_1(k) \bigoplus S_2(k) \neq Err \tag{1}$$

Moreover, as our algebraic relation is associative, we can compute the nA sequences in any order with this equation:

$$\bigoplus_{i \in [1,nA]} S_i = ((((S_1 \bigoplus S_2) \bigoplus S_3) \bigoplus \ldots \bigoplus S_{nA-1}) \bigoplus S_{nA}) \tag{2}$$

Thus, from the two previous equations (1) and (2), the nA actions are synchronizable if and only if:

$$\forall k, \quad \bigoplus_{i \in [1,nA]} S_i(k) \neq Err \tag{3}$$

4.2. Generation of the sequence

In this section, we present our generation of sequences of support phases. However, the synchronization algorithm uses the support phases in whatever way they were generated. It only considers that two successive elements in a sequence are never identical. Currently, our sequences are generated using two parameters (cf. Figure 5):

- the maximum authorized height for the lowest extremity of the foot. It represents the height over which the foot is not in contact with the ground;
- the maximum authorized speed when the foot is under the previous threshold.

The foot is considered in contact with the ground if the two previous parameters are respected. The second parameter is useful to discriminate the foot strike (when speed is really low) from moments when the foot quickly passes near the ground (i.e. during the swing phase of a walk for example).

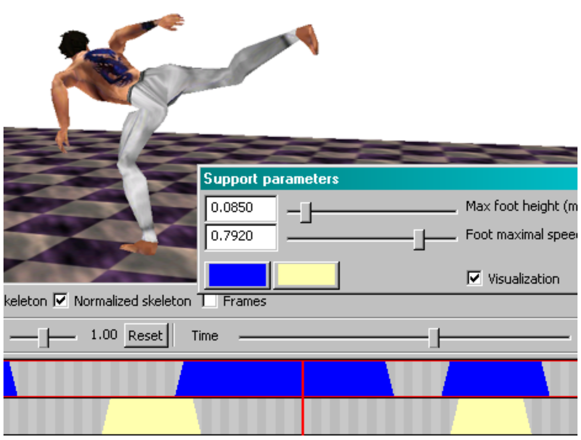

Support parameters
0.0850 Max foot height (m
0.7920 Foot maximal spee
☑ Visualization
keleton ☑ Normalized skeleton ☐ Frames
1.00 Reset Time

Figure 5: *Interface of the support phases generator. At the bottom, the two horizontal areas define the support phases for each foot.*

4.3. Extension of motions

Consider two actions A_i and A_j with sequences S_i (with five elements) and S_j (with only three elements) respectively. To use our algebraic relation after the third element, we need to extend the sequence of action A_j. Moreover, with cyclified motions such as locomotion, we need to extend its sequence as long as the action is running. To this end, we consider two kinds of motions (cf. Figure 6):

- Cyclified motions: their sequences are repeated as a whole;
- Other motions: at the end of these motions, the actions always return the last posture. So only the last element of their sequence is repeated as long as necessary.

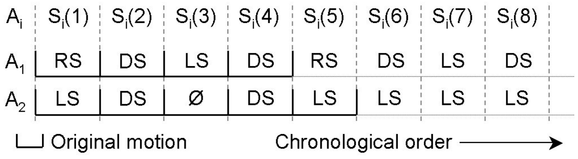
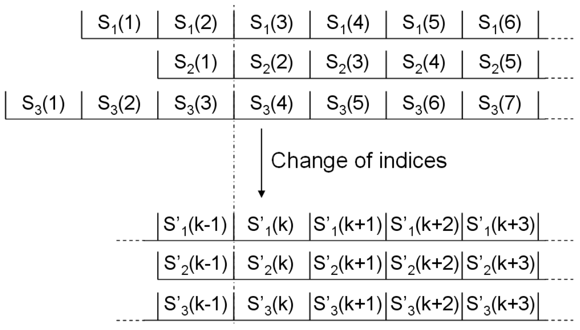


Figure 6: *Extension of cyclic (A_1) and non-cyclic (A_2) actions. A_1 is repeated as a whole; only the last element of A_2 is repeated.*



$S_1(1)$ $S_1(2)$ $S_1(3)$ $S_1(4)$ $S_1(5)$ $S_1(6)$
$S_2(1)$ $S_2(2)$ $S_2(3)$ $S_2(4)$ $S_2(5)$
$S_3(1)$ $S_3(2)$ $S_3(3)$ $S_3(4)$ $S_3(5)$ $S_3(6)$ $S_3(7)$
Change of indices
$S'_1(k-1)$ $S'_1(k)$ $S'_1(k+1)$ $S'_1(k+2)$ $S'_1(k+3)$
$S'_2(k-1)$ $S'_2(k)$ $S'_2(k+1)$ $S'_2(k+2)$ $S'_2(k+3)$
$S'_3(k-1)$ $S'_3(k)$ $S'_3(k+1)$ $S'_3(k+2)$ $S'_3(k+3)$

Figure 7: *Change of variable for the indices. All the current sequences are indexed by k (bottom of figure) whenever the corresponding actions were started (top of figure).*

5. Synchronization algorithm

The synchronization algorithm uses three main steps to achieve its work. The first step is the reorganization of the elements of the sequences in order to synchronize them. The second step is the tuning of the duration of the resulting sequence of support phases. Finally, the algorithm assures that the temporal deformation resulting from the time warping function does not engender speed discontinuities in the animation.

5.1. Chronological organization

Due to real-time interactions, the motions can start at any moment (by using the start/stop commands). To use our relation, we first make a change of variable to ensure that the current sequence of each action is indexed by the same value k as shown in Figure 7.

Always regarding the interactive and real-time constraint, we do not want to apply our algebraic relation to all the sequence but only to the next successive nk elements. The following algorithm is then applied iteratively during the animation.
Consider that the motion is synchronized until the step $k+nk$ (the animation is currently at the step k):

$$\forall 1 \leq j \leq k+nk, \quad \bigoplus_{i\in[1,nA]} S_i(j) \neq Err \qquad (4)$$

Now, when the animation enters step $k+1$, our algorithm must synchronize step $k+nk+1$:

$$\bigoplus_{i\in[1,nA]} S_i(k+nk+1) \neq Err \qquad (5)$$

If this previous equation (5) is not verified, our synchronizing algorithm proposes to enlarge some of the last synchronized elements at the step $k+nk$ to respect the equation. For example, consider two motions A_1 and A_2 (cf. top of Figure 8) such as:

- $S_1(k+1) = DS$ and $S_1(k+2) = RS$;
- $S_2(k+1) = \emptyset$ and $S_2(k+2) = LS$.

Let nk be equal to 1 (synchronization is just made over one element of the sequence). In this example, S_1 and S_2 are not synchronized at step $k+nk+1 = k+2$. Deleting or reducing one of the two concerned elements does not solve the problem. However, the animation is about to enter step $k+1$ when the synchronization process is activated, so we can enlarge one of the element $S_i(k+1)$ without creating discontinuities in the animation. In this example, extending element $S_1(k+1)$ solves the problem (cf. bottom of Figure 8).

Enlarging the last synchronized element of the sequence always gives at least one solution to the synchronization problem. Appendix A details the recursive proof of this algorithm. Now, one solution has to be chosen from the set of possible ones. To achieve this, the algorithm uses the priority of the motions. From the highest priority to the lowest one, it takes the first element of type RS or LS. This element gives the type of constraint to be respected. Hence, extending all the last synchronized elements of the other type (i.e., RS if the element is LS) ensures that the constraint associated with the motion with the highest priority is respected. Figure 9 shows an example of such a synchronization.

Now, we are sure that active motions can be synchronized at nk steps from the current element of the sequence. But there remains the start of the actions. When an action

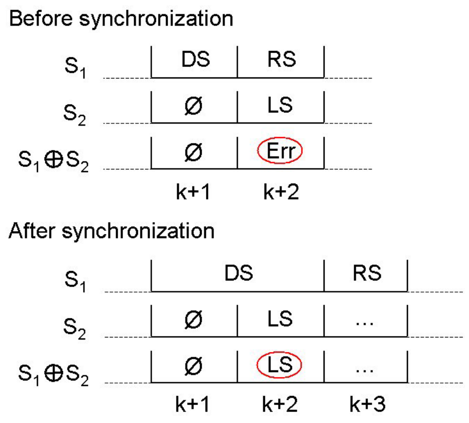


Figure 8: *Example of extension of an element of the sequence. After synchronization, all the actions can be blended and the relation determines the new support phase.*

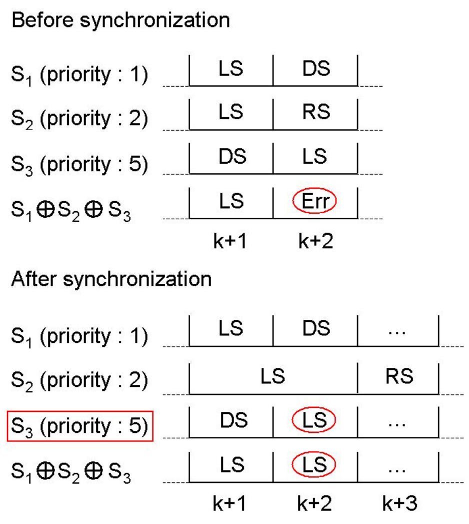


Figure 9: *Example of synchronization using priorities. The resulting support phase respects the action with the highest priority.*

A_i is activated, the algorithm tries to synchronize from the first element $S_i(k)$ to the element $S_i(k+nk+1)$. If there is no error during this synchronization, the action is started at the first synchronized element and the synchronization iterative algorithm can be applied for the rest of its activation. If a synchronization error occurs (the relation returns Err), it means that there is no possible synchronization with the current active actions. Then, the start of the action is delayed until the next element of the sequence is reached.

The value of the number nk of support phases synchronized by our algorithm can be discussed. When it is small, the temporal deformation can be important to ensure the compatibility of the motions. On the contrary, when it is large, the knowledge of the future is required. Consequently, it is more difficult to start a new action since the algorithm must synchronize more support phases at the beginning of the motion. The default value used for all our animations inside MKM is 5. It allows to synchronize more than a complete cycle of locomotion.

5.2. Duration of an element of the sequence

Now that elements of the sequence of support phases are correctly reorganized, there remains to compute the duration of these elements. To achieve this, the synchronization process uses the priority P_i defined for each motion A_i in order to compute the normalized weight w_i associated to the legs [MMKA04]. Let $\Delta S_i(k)$ be the duration of the k^{th} element of the sequence S_i. The computed duration of all the k^{th} elements of all the sequence is then defined as:

$$\forall k, \quad \Delta S(k) = \sum_{i \in [1,nA]} w_i \times \Delta S_i(k) \qquad (6)$$

This duration is then used in the time warping process to adapt the motion.

5.3. Temporal deformation

Changing the duration of the elements of the sequences involves a compression or a dilation of time. Figure 10 shows the temporal deformations due to the use of time warping. Let $t' = H_i(t)$ be the time warping function that gives the animation time from the real time. This function is not derivable and engenders discontinuities in speed and acceleration. After the computation of this initial time warping, we estimate a function that allows to slow down or accelerate the motion in order to avoid these discontinuities during animation.

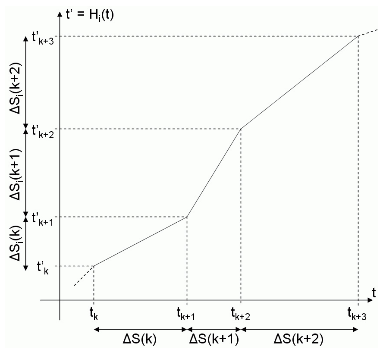


Figure 10: *Non derivable initial time warping.*

Let t and $v(t)$ be the current time (in element k) and the corresponding derivative of the spline (Figure 11A). If the duration $\Delta S(k)$ of element k has changed since the last simulation step, then t_{n+1} has also changed (Figure 11B).

In this case, we recompute the derivative of the extremity of element k by taking the symetric of $v(t)$ (Figure 11D) and born it (Figure 11E). Finaly, the function on element k is recovered by using a cardinal spline (Figure 11F). This process gives a monotonic time warping (Figure 12) and is performed each time elements' durations vary.

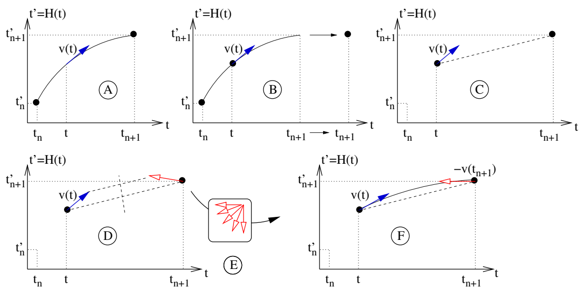


Figure 11: *Monotonic spline construction process.*

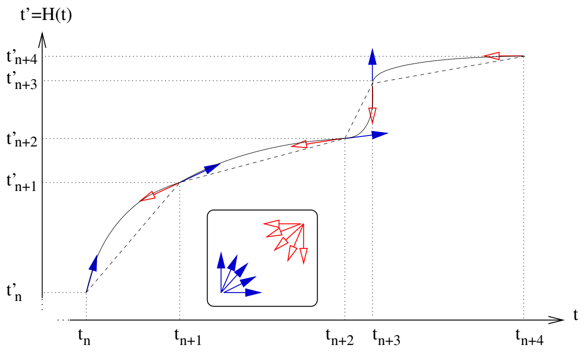


Figure 12: *Final monotonic time warping.*

6. Results

One of the most visual artifacts produced by a bad transition between motions is sliding feet. Figure 17 compares a same sequence of different locomotions using the synchronization algorithm or not. The red parts of the animation represent the phase of transition between motions. At the top of this figure, the different locomotions are activated and deactivated at predefined times. The three ellipses stress the moments when the feet are sliding on the floor. The bottom of the figure shows the same sequence of locomotions but using the synchronization algorithm. This time, the feet do not slide and the support constraints are respected. This figure shows that even with motions of the same kind (such as locomotion in this example), the synchronization of motions is really important.

Following the same wireframe representation, Figure 18 shows a more complex animation. It is composed of a locomotion, a fighting motion, another locomotion and finally a jump. This figure shows the delay that can appear during the synchronization of complex motions. The last row gives the same synchronized animation using a skinned character.

The synchronization module has been tested and validated in several applications using usual motions (walks and runs) and specific ones (sports motions, dance). Common bipedic sequences (including right and left supports) have enough variations to ensure several solutions in our synchronization algorithm. Figure 13 shows some examples of handball shots synchronized with a run.

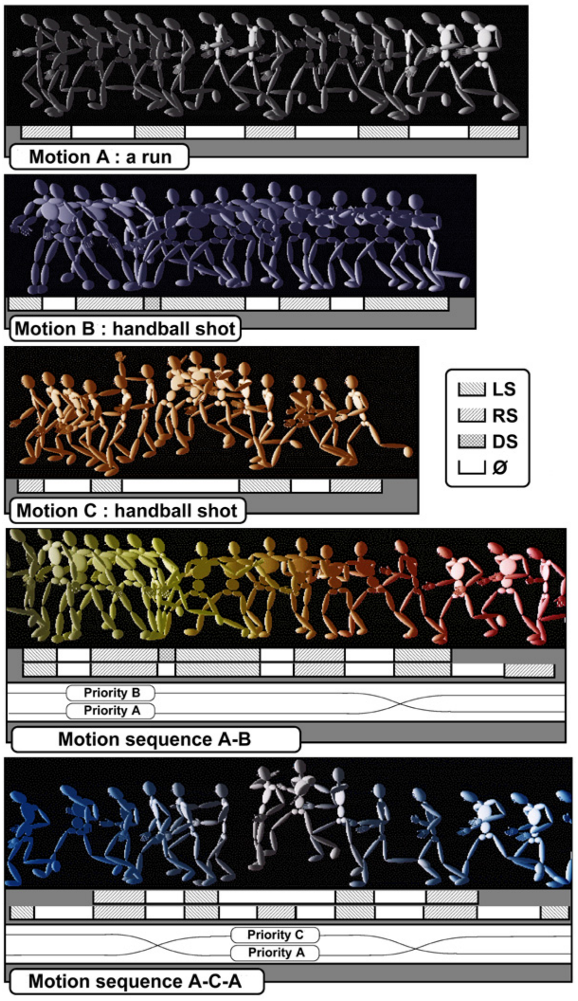


Figure 13: *Synchronization sequences with handball shots.*

However, it is still possible to fail synchronizing critical motions like a right hopping with a left hopping motion. In this case, the last motion waits for a better instant to begin and checks the synchronization possibilities at each new element step. These cases can be avoided by coupling normal motions with critical ones to give more solutions to our synchronization algorithm. Figure 14 shows an example of a walk-to-hop sequence.

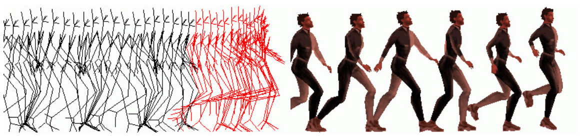


Figure 14: *Walk-to-hope sequence.*

The dynamic synchronization associated with footprint handling gives the opportunity to change the original motions and actions priorities in real time while preserving valid support constraints. Figure 15 shows an example of dynamic blending between a side step motion and a front/back walk in order to force the avatar's displacement on a linear path whatever its local orientation. The same example could be easily extended to a non-linear path.

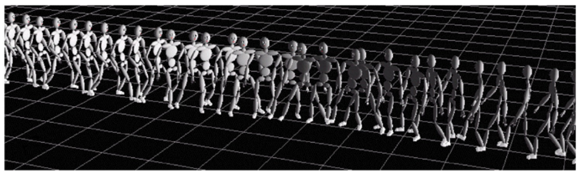


Figure 15: *Dynamic blending and synchronization.*

The last example shows an animation resulting from the complete use of our system. The floor is interactively modified and the character is animated using several motions. These motions are synchronized, blended and adapted to the environment in real-time. The morphological adaptation is still made automatically with the normalized skeleton. In figure 16, the floor is stationary only to allow a clear illustration. The video (ftp.irisa.fr/local/siames/rkulpa/SCA04/dynamicEnv.avi) gives the complete animation with the moving floor. Another video (synchronization.avi) shows the original motions and the result of their synchronization and their blending. The last video clip (hopping.avi) show the automatic synchronization of walks with right hopping motions.

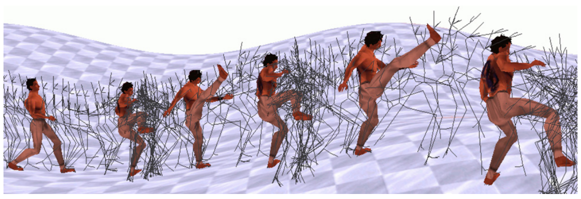


Figure 16: *The synchronization module is part of our complete kinematical animation engine MKM. In this example, the character is animated on an interactively changing ground. For the picture, the floor was stationary.*

7. Discussion

In this paper, we have presented a new real-time algorithm that synchronizes several interactively adapted motions. This algorithm is based on the time warping technique with additional information: the sequence of support phases for each motion. We also introduced an algebraic relation between support phases to verify if they are compatible for motion blending. Its use is very easy since motions are basically controlled by simple start/stop commands. The duration used to activate or deactivate the motion is parameterized for each action as well as the priorities. The weights associated to motions are then computed automatically and are used by our synchronization algorithm. The latter is part of a complete kinematical animation engine: MKM. Using an adimensional and normalized skeleton, a motion can be automatically shared by several characters with different morphologies. Moreover, the synchronized motions are blended by a specific blending module [MMKA04].

This complete real-time animation engine is already used by industrial partners. Consequently, this package has been widely tested with motions involving really different kinds of dynamics and of support phases (such as fighting, walking, running and dancing). It also works while using parametric models such as locomotion or grasping models. The activation time is instantaneous except for rarely critical situations such as simultaneously right and left unilateral jumps. Even in this kind of situation, our synchronization technique always finds a solution contrary to other current studies using motion graphs or predefined correspondences between motions. The computation cost of this algorithm is negligible since it only synchronizes sequences when motions change their support phases. Moreover, it does not need information on all the sequence of support phases because it only synchronizes a limited number of its elements (i.e. the support phases).

Like several systems based on kinematics, our system suffers from some limitations. Indeed, nothing ensures that the weighted sum of two motions that both respect dynamics also respects dynamics. To take this kind of constraints into account, there is some low computation cost kinematic solutions. For example, the time warping technique can engender large dilations of the motion duration. When it is applied while jumping, the fundamental laws of dynamics are not respected. One solution can be to freeze the weights associated to the action during a jump phase. It means that the blending is interrupted during a jump. This problem can also be corrected by controlling the parabolic trajectory of the root during the jump. Our perspectives are to incorporate such techniques inside our system in order to improve the respect of the fundamental laws of dynamics that already exist intrinsically in the original motions. We are also currently working on a dynamic controller to reinforce the results.

Our system also lets the user choose the motions to blend to achieve better interactivity. If the user asks for starting a special motion we directly try to synchronize it without using another transitional one. Indeed, intelligent choices of motions can be helpful to avoid large time warping but can also produce unwanted transitional motions if only kinematics rules are taken into account (ie if motion nature is ignored). Hence, we prefer synchronizing user motions directly while preserving from sliding feet problems. We also add some algorithmic rules which help avoid large time warping, like the possibility to split neutral constraint phases (double support) or choose best solutions while taking account of time dilatation.

Finally, our synchronization algorithm is only based on the sequence of support phases and limited to bipedic humans. It could be possible (mathematicaly) to extend the algorithm (considering hands for example) if an new algebraic relation is proposed (associative and commutative). Nevertheless, the more constraints we have, the more difficult it becomes to extract a general relation. We have decided to ensure secondary constraints (feet are primaries) with another part of MKM, independant of the synchronization module described in this paper.

Acknowledgements

We would like to thank gratefully Stéphane Donikian and Marwan Badawi for their helpful reviews of this paper and their suggestions. We also thank Canal Numedia for their avatar generation software "Avatar Studio". This work was partially funded by the French research program RIAM within the AVA-MOTION project initiated by Dæsign corp.

References

[AW00] ASHRAF G., WONG K.: Generating consistant motion transition via framespace interpolation. *Eurographics, Computer Graphics Forum 19*, 3 (Aug. 2000).

[AW01] ASHRAF G., WONG K.: Constrained framespace interpolation. *Computer Animation 2001* (Nov. 2001), 61–72.

[AW03] ASHRAF G., WONG K.: Semantic representation and correspondence for state-based motion transition. *IEEE Transactions on Visualization and Computer Graphics 9*, 4 (2003), 481–499.

[BBET97] BOULIC R., BECHEIRAZ P., EMERING L., THALMANN D.: Integration of motion control techniques for virtual human and avatar real-time animation. In *Proceedings of the ACM symposium on Virtual reality software and technology* (1997), ACM Press, pp. 111–118.

[BH00] BRAND M., HERTZMANN A.: Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 183–192.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, pp. 97–104.

[GR96] GUO S., ROBERGÉ J.: A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96* (1996), Springer-Verlag New York, Inc., pp. 95–107.

[KG03] KOVAR L., GLEICHER M.: Flexible automatic motion blending with registration curves. Eurographics Association, pp. 214–224.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 473–482.

[LCR*02] LEE J., CHAI J., REITSMA P., HODGINS J., POLLARD N.: Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 491–500.

[MFCGD99] MULTON F., FRANCE L., CANI-GASCUEL M.-P., DEBUNNE G.: Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation 10*, 1 (1999), 39–54.

[MMKA04] MÉNARDAIS S., MULTON F., KULPA R., ARNALDI B.: Motion blending for real-time animation while accounting for the environment. In *Computer Graphics International* (June 2004).

[PSKS] PARK S., SHIN H., KIM T., SHIN S.: Online motion blending for real-time locomotion generation. *CASA04*.

[PSS02] PARK S., SHIN H., SHIN S.: On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), ACM Press, pp. 105–111.

[RCB98] ROSE C., COHEN M., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (1998), 32–40.

[RGBC96] ROSE C., GUENTER B., BODENHEIMER B., COHEN M. F.: Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM Press, pp. 147–154.

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 489–498.

[UAT95] UNUMA M., ANJYO K., TAKEUCHI R.: Fourier principles for emotion-based human figure animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, p-p. 91–96.

[WB03] WANG J., BODENHEIMER B.: An evaluation of a cost metric for selecting transitions between motion segments. *Eurographics/SIGGRAPH Symposium on Computer Animation 2003* (2003), 232–238.

[WP95] WITKIN A., POPOVIĆ Z.: Motion warping. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, pp. 105–108.

[Zel82] ZELTZER D.: Motor control techniques for figure animation. *IEEE Computer Graphics and Applications 2*, 9 (Nov. 1982), 53–59.

[Zel86] ZELTZER D.: Knowledge-based animation. In *Proc. of the ACM SIGGRAPH/SIGART interdisciplinary workshop on Motion: representation and perception* (1986), Elsevier North-Holland, Inc., pp. 318–323.

Appendix A: Recursive proof of the synchronization algorithm

The first part of this appendix shows that, considering that an error has occurred during the synchronization process, at least one solution is found.

Consider that the animation is entering step k. It means that all the nA actions are synchronized until step $k+nk-1$:

$$\forall j \in [k-1, k+nk-1], \quad \bigoplus_{i\in[1,nA]} S_i(j) \neq Err \tag{7}$$

Consider that the actions are not synchronized at step $k+nk$ (else the solution is already found):

$$\bigoplus_{i\in[1,nA]} S_i(k+nk) = Err \tag{8}$$

The algorithm tries to extend some of the last synchronized elements. Thus, there is a solution if there is at least one element at step $k+nk-1$ that can be extended to verify the following equation (9).

$$\bigoplus_{i\in[1,nA]} S_i(k+nk) \neq Err \tag{9}$$

The first solution can be to extend all the nA last synchronized elements:

$$\bigoplus_{i\in[1,nA]} S_i(k+nk) = \bigoplus_{i\in[1,nA]} S_i(k+nk-1) \neq Err \tag{10}$$

Unfortunately, this solution does not solve the problem but just delays it.

Let A be the maximal set with the elements at step $k+nk$ that can be synchronized with the result of the previous step $k+nk-1$.

$$\left(\bigoplus_{i\in[1,nA]} S_i(k+nk-1)\right) \bigoplus \left(\bigoplus_{i\in A} S_i(k+nk)\right) \neq Err \tag{11}$$

This set cannot be empty since the error at step $k+nk$ (cf. equation (8)) involves the presence of RS and LS. So there are compatible elements with one of them and so with all the existing elements. Then, this set is also synchronizable with the result of the previous step $k+nk-1$. Now, two cases have to be considered:

- $\bigoplus_{i\in[1,nA]} S_i(k+nk-1) = \emptyset$: in this case, extending one of the absorbing elements ($\emptyset$) is enough to find a solution (even if the extension of a jump is not an ideal solution);
- $\bigoplus_{i\in[1,nA]} S_i(k+nk-1) \neq \emptyset$: since the result is not the absorbing element ($\emptyset$) then it is not in set A. Moreover, there is no element of type $\emptyset$ at step $k+nk$ since the result is an error Err. So the following equation is verified:

$$\left(\bigoplus_{i\in[1,nA]} S_i(k+nk-1)\right) \bigoplus \left(\bigoplus_{i\in A} S_i(k+nk)\right) \neq \emptyset \tag{12}$$

As A defines the elements compatible with the previous synchronization, then $\{i \notin A\}$ represents the indices of the motions in which last synchronized elements have been extended to solve the synchronization problem.

Set A does not represent all the solutions of the synchronization. As it is never empty, it is only used to demonstrate that there is always at least one solution.

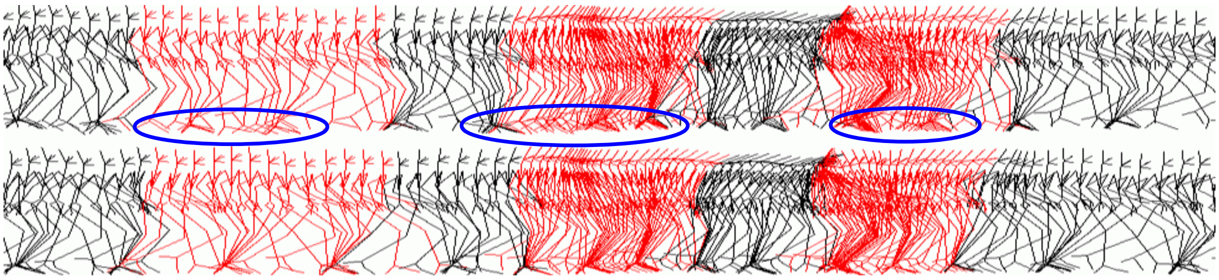


Figure 17: *The first row shows the animation without synchronization. When there are transitions between motions (in red) the feet are sliding on the floor. With synchronization (second row), the foot strikes are respected.*

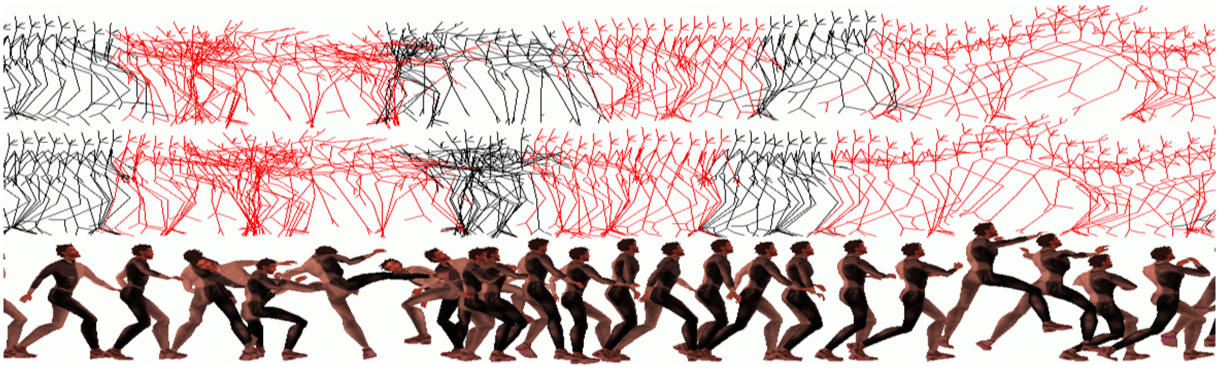


Figure 18: *This animation with motions such as walking, fighting, another walking and a jump shows delayed actions. Contrary to the first row, the two others show the animation with synchronization.*

© The Eurographics Association 2004.