Interactive Motion Synthesis with Optimal Blending

Masaki Oshita
Kyushu Institute of Technology
oshita@ces.kyutech.ac.jp

Abstract

In this paper, we propose an interactive motion synthesis technique that synthesizes a continuous motion sequence from given elementary motions. A user can either specify the execution timings of motions, or execute motions in a sequence with automatically determined execution timings. Our method is based on a previous approach that determined the appropriate synthesis method and blending range for each pair of sequential motions, while considering the constraints between the foot and ground in the elementary motions to prevent foot sliding. However, using only the foot-ground constraints to determine the blending range may generate unnatural motions such as non-smooth, too fast, or too slow transitions. Moreover, simple motion blending with a regular weight function can generate unnatural motions. To solve these problems, we have introduced an optimal blending range and a weight function, which are determined for each blending segment for the upper and lower body. We also introduced extensions for applying this method to interactive character control. We have successfully applied our method to both animation generation and interactive character control.

Keywords: motion synthesis, motion control, motion transition, motion blending

1 Introduction

A set of short elementary motions can be used to generate a long motion sequence. This is a common approach for both off-line animation production and on-line character control in interactive applications such as computer games. However, it is not easy to make a smooth connection between two motions. Many parameters must be tuned by an animator, including the alignment of motions, blending range, and additional constraints.

State machines are often used to apply this approach in interactive applications. A state machine is represented by a graph structure that contains information about possible connections between elementary motions and the specific motion blending parameters for each connection. Constructing such a state machine takes time, especially when there are many elementary motions. Because of this, it is difficult to include many elementary motions in a state machine. As a result, characters in interactive applications can currently only perform a limited number of actions. A method that enables elementary motions to be executed over any execution time and automatically synthesized continuous motion sequences will be beneficial for interactive character control and animation production. Then, a large number of pre-created (captured) elementary motions will be fully utilized.

In this paper, we propose an interactive motion synthesis technique that synthesizes a continuous motion sequence from given elementary motions (Figure 1). A user can either specify the execution times of motions or execute motions in a sequence with automatically determined execution times. Our method is based on a previous approach in [1], which determined the appropriate synthesis method and blending range for each pair of sequential motions, considering the constraints between the foot and the ground

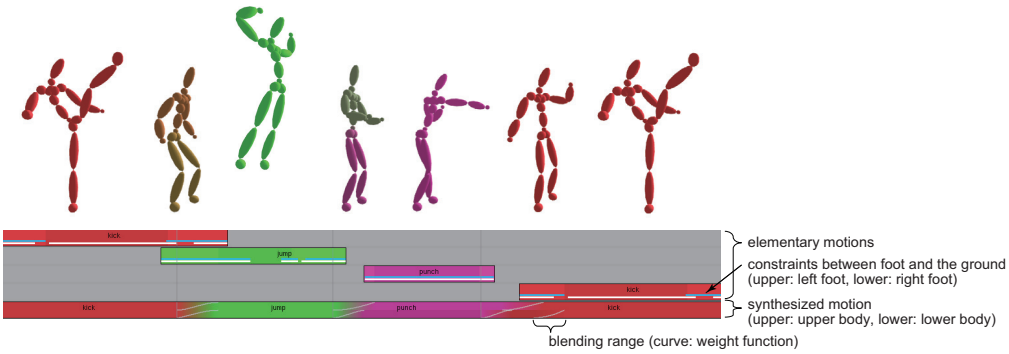

Figure 1: Example of interactive motion synthesis. When elementary motions are placed on the timeline, a synthesized motion sequence is generated.

to prevent foot sliding. However, determining the blending range using only foot-ground constraints may generate unnatural motions such as non-smooth, too fast, or too slow transitions. Moreover, simple motion blending with a regular weight function can also generate unnatural motions. To solve these problems, we have introduced an optimal blending range and a weight function, which are determined for each blending segment for the upper or lower body. We propose new criteria and methods for determining these values.

We also introduce extensions for applying this method to interactive character control. When an elementary motion is added and the synthesized motion is updated while being played, the part of the motion that has been already played shouldn't be changed. We extended our method so that it considers this constraint.

Our method can be used for both animation generation and interactive character control. For animation generation, a user can arrange elementary motions on the timeline in a similar way as existing non-linear animation editing systems. Then, our system interactively generates synthesized motion as the user changes the execution timing of an elementary motion. For interactive character control, our system can work like a game engine. Our system updates the synthesized motion when a new elementary motion is input by a user,

The results of our experiments are promising for both animation editing and interactive character control with dancing and fighting motions, which are dynamic movements that are considered difficult for motion synthesis methods.

2 Related Work

2.1 Motion Synthesis

There are many approaches for synthesizing a continuous motion by making smooth transitions between input motions.

Motion blending is one commonly used technique [2, 3], and is available in many animation editing systems. However, making a smooth connection between two motions is not that easy. Many parameters, such as the alignment of motions, blending range, and additional constraints, must be tuned by an animator. Wang and Bodenheimer [3] proposed a method for determining an appropriate blending duration. They assumed that the execution timing of the motion is determined by the method and did not consider user defined execution timings. Moreover, they did not consider the constraints between the foot and ground. In general, motion blending techniques are applicable when the previous and next motions are similar poses. In particular, it does not work well when the foot positions are different.

Motion interpolation [4, 5] can be used to create smooth transitions. However, this approach is only applicable when the previous and following motions are of the same category and are parameterized for interpolation in advance.

Motion synthesis methods based on the constraint between the foot and ground have been proposed [1, 5, 6]. They can automatically determine the appropriate blending method and range so as to prevent foot sliding. However, as explained in Section 1, this approach has problems. Unnatural motions may be gener-

ated if we determine the blending range based on only the foot-ground constraints or use simple motion blending with a regular weight function. Our method extends the method in [1] and solves these problems. Other previously proposed methods [6, 5] use only one type of motion blending and cannot handle various cases. Moreover, they always determine the execution timings, so they cannot deal with user-defined timings.

There are approaches for generating smooth transitions that considering the physics of the motion. Rose et al. [7] considered joint torques for making transition between motions. They used inverse dynamics to compute the required joint torques, and optimized a short transition motion to minimize the joint torques. This kind of optimization is computationally intensive, and it is difficult to compute joint torques when a character is standing and moving their foot. Zordan et al. [8] used physics simulations to generate smooth transitions from a motion to a falling down motion. They first applied motion blending between two motions to generate an initial target motion. They then applied proportional derivative (PD) controllers and physics simulations. However, the quality of the resulting motions rely on the motion blending method and PD controllers. This technique is suitable for transitions to falling motions, but not for other kinds of motions. Shum et al. [9] proposed a motion concatenation based on the angular moment of motions. This method finds an appropriate timing for the blending that maintains the angular moment of motions during the transition. However, this method is targeted at motions that share the same rotational axis and cannot be applied to other kinds of motions.

Another approach is to find an intermediate motion in a database [10]. This is effective, especially when the poses of the two motions are not similar. However, this approach requires a lot of data, because there may be many possible combinations of poses. Moreover, it takes some time to calculate the transitions.

2.2 Interactive Motion Generation

As explained in Section 1, state machines are commonly used in current computer games. Recently developed game engines such as Mecanim from Unity and Euphoria from Natural Motion provide editing tools for designing a state machine using a graphical user interface. However, it is still difficult to construct a state machine with a large number of motions, because all the possible connections between motions must be tuned. Moreover, because motion blending can only be applied to the connections between two motions that share similar poses, the elementary motions must be carefully created or captured.

A motion graph is a technique for automatically constructing a graph structure [11]. It is automatically constructed from a set of long motion sequences. However, this approach requires a lot of motion data that contain similar poses. Moreover, there is no guarantee that an expected action can be executed quickly, and so it is difficult to interactively control a character. Recently, statistical models that contain transitions of poses (smaller motion segments) have been proposed [12], but they have the same problems. Unfortunately, these methods have not been widely implemented in current computer games.

There are other approaches for using sets of elementary motions that do not split them into short segments. A method for constructing a state machine-like graph structure of elementary motions was proposed in [13]. There are also methods for planning character motions using elementary motions [14, 15, 16]. However, these methods use similar poses from the given sets of motions to make smooth transitions, and have the same problems as motion graphs.

In our method, a quick and smooth transition is generated between any pair of motions without preparing an additional data structure.

3 System Overview

Our system generates a continuous motion sequence from a number of elementary motions with any execution timings, as shown in Figure 1. A user can either specify the execution timings of the input motions or execute the input motions in sequence with automatically determined execution timings.

The input motions must be created in advance. Many methods can be used, for example.

motion capture and keyframing. In addition, the user can specify the core part of each input motion. Normally, a motion of a single action contains preparation and recuperation phases that occur before and after the core part. By specifying the time period, our system tries to preserve the core part while making smooth transitions.

When multiple motions are given with the same execution timing, not all motions may be executable. In this case, our system rejects the unexecutable motions.

4 Motion Synthesis Method

This section describes our motion synthesis method. We focus on our method for making smooth connections between two elementary motions. We assume that we are given two elementary motions and their execution timings. The extension for automatically determining execution timings is explained in Section 5.

To combine multiple elementary motions into a motion sequence, the elementary motions are sorted by their execution times and the motion synthesis method is repeatedly applied to each pair of sequential motions to make smooth connections.

4.1 Fundamental Method

Our method is based on the motion synthesis method proposed in [1]. In this section, we briefly describe an overview of the existing method.

It determines the appropriate blending method and segment based on constraints between the foot and ground in the input motions, which are automatically analyzed based on the heights and velocities of the feet. There are four types of blending methods: motion transition, motion connection, and motion adaptation with, and without, the lower body pose transition. These are shown in Figure 2. Unlike other motion synthesis methods that only use motion-to-motion blending, our method choses an appropriate type of blending for each segment, namely motion-to-motion, pose-to-motion, motion-to-pose, or pose-to-pose. In motion-to-motion blending, two segments from two motions are blended. Motion-to-pose blends a segment of the previous motion and a pose from the following motion. The four types of blending methods and their conditions are as follows.

Motion transition is applied if the same foot is moved in two motions at the same time. A motion-to-motion blending is applied to the segment of the foot-moving phase of the motions (Figure 2(a)).

Motion connection is applied if a foot is moved in the previous motion. A motion-to-pose blending is applied (Figure 2(b)).

Motion adaptation is applied if a foot is moved in the previous motion. The lower body pose of the previous motion is preserved in the following motion. If a foot is moved during the following motion, a pose-to-motion blending is applied (Figure 2(c)). Otherwise, the lower body pose of the previous motion is preserved until the end of the following motion (Figure 2(d)). During motion adaptation, the lower body pose is deformed to maintain the foot positions on the previous motion and the pelvis position is adjusted so that it maintains its relative position to the feet (based on the modified foot positions).

Note that terms such as motion transition, connection, adaptation, blending, and synthesis may have been used to represent different concepts in previous studies. There are no clear definitions of these terms. In this paper, we follow the terms and definitions used in [1].

The initial position and orientation of the following motion is automatically determined so that the supporting foot positions and body orientations of the two motions are aligned. The body orientation during a motion is computed based on the direction of the pelvis movement.

4.2 Blending Methods and Segments

Our method determines the appropriate blending method using the approach described above. Several internal blending segments are created for each pair of sequential motions.

The differences between the previous method and ours can be summarized as follows. First, our method treats the upper and lower body separately and generates individual blending segments, because the appropriate blending ranges are often different. In addition, the weight function used in the blending is determined for each

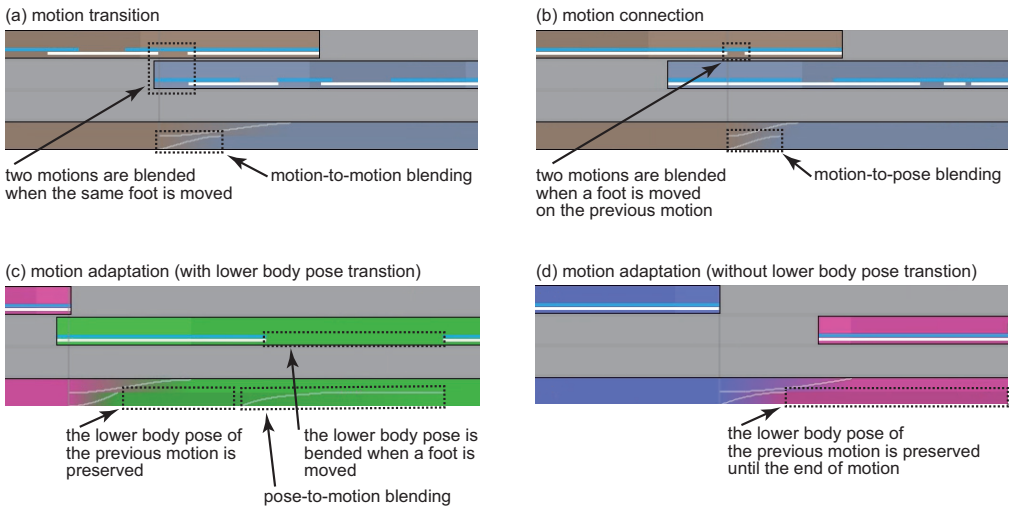

Figure 2: Motion synthesis method. One of four types of blending methods is applied to a pair of sequential motions.

segment.

Second, our method considers the core part of the motion. As explained in Section 3, the user can specify the core part of each input motion. Normally, an elementary motion has a preparation phase and a recuperation phase before and after the core. During motion synthesis, the core parts of the input motions should be preserved, and the preparation and recuperation phases can disappear. However, the preparation and recuperation phases are important for smooth transitions, particularly when there is some interval between the motions. The synthesized motions look unnatural if we only use the core parts as the input motions. Therefore, we allow users to manually specify the core part of each input motion.

We create the internal blending segments for each type of blending method considering these factors.

4.2.1 Motion Transition

Motion transition is applied when the same foot is moved in two motions. These foot-moving phases must be outside of the core parts.

In motion transition, two blending segments are created (Figure 2(a)). For lower body motions, the phases where the same foot is moved in two motions are blended using a motion-to-motion blending. For upper body motions, a pose-to-pose blending is applied at the same time, because the input motions in the blending segment are not relevant to the synthesized motion and the blending between the initial and terminal poses produces a smooth motion. These two blending segments can have different ranges.

Let $C_0(t_{c0,begin} \sim t_{c0,end})$ and $C_1(t_{c1,begin} \sim t_{c1,end})$ be the core parts of the previous and following motions, and $P_0(t_{p0,begin} \sim t_{p0,end})$ and $P_1(t_{p1,begin} \sim t_{p1,end})$ be the foot-moving phases in the previous and following motions.

When constructing the blending segment for the lower body using a motion-to-motion blending, we consider that the overlap of phases P_0 and P_1 $(t_{p0,begin} \sim t_{p1,end})$ is the core part of the blending segment. Therefore, the segment for the lower body (S_0 $(t_{s0,begin} \sim t_{s0,end})$) must satisfy the following conditions.

$$\begin{aligned} &t_{c0,end} < t_{s0,begin} < t_{p1,end}, \\ &t_{p0,begin} < t_{s0,end} < t_{c1,begin}, \\ &t_{s0,begin} < t_{s0,end} \end{aligned} \tag{1}$$

Similarly, the blending segment for the upper body with pose-to-pose blending S_1 $(t_{s1,begin} \sim t_{s1,end})$ must satisfy the following conditions.

$$\begin{aligned} &t_{c0,end} < t_{s1,begin} < t_{p1,end}, \\ &t_{p0,begin} < t_{s1,end} < t_{c1,begin}, \\ &t_{s1,begin} < t_{s1,end} \end{aligned} \tag{2}$$

Within these possible ranges, an optimal blending range is determined for each segment.

The details are explained in Section 4.3.

4.2.2 Motion Connection

In motion connection, two blending segments are created (Figure 2(b)). A motion-to-pose blending is applied for lower body motions, because the foot-moving phase in the previous motion and the terminal pose in the following motion are important.

Let $P_0(t_{p0,begin} \sim t_{p0,end})$ be the foot-moving phase in the previous motion. The blending segment for the lower body with motion-to-pose blending S_0 must satisfy the following conditions.

$$\begin{aligned} &t_{s0,begin} = t_{p0,begin}, \\ &t_{p0,begin} < t_{s0,end} < t_{c1,begin}, \\ &t_{s0,begin} < t_{s0,end} \end{aligned} \tag{3}$$

The segment for the upper body with pose-to-pose blending S_1 must satisfy the following conditions.

$$\begin{aligned} &t_{p0,begin} < t_{s1,begin} < t_{c1,begin}, \\ &t_{p0,begin} < t_{s1,end} < t_{c1,begin}, \\ &t_{s1,begin} < t_{s1,end} \end{aligned} \tag{4}$$

4.2.3 Motion Adaptation with the Lower Body Pose Transition

In motion adaptation with a lower body pose transition, four blending segments are created (Figure 2(c)).

Let $P_1(t_{p1,begin} \sim t_{p1,end})$ be the foot-moving phase in the following motion $M_1(t_{m1,begin} \sim t_{m1,end})$. P_1 may be within or after the core part of the following motion. The blending segment for the lower body with pose-to-motion blending S_0 must satisfy the following conditions.

$$\begin{aligned} &t_{s0,begin} = t_{p0,begin}, \\ &t_{p0,begin} < t_{s0,end} < t_{m1,end}, \\ &t_{s0,begin} < t_{s0,end} \end{aligned} \tag{5}$$

The blending segments for the transition between motions of the lower body S_1 and the upper body S_2 must satisfy the following conditions.

$$\begin{aligned} &t_{c0,end} < t_{s1,begin} < t_{c1,begin}, \\ &t_{c0,end} < t_{s1,end} < t_{c1,begin}, \\ &t_{s1,begin} < t_{s1,end} \end{aligned} \tag{6}$$

The same conditions are applied to S_2.

Finally, the segment for deforming the lower body pose is applied to the segment S_3 where

$$\begin{aligned} &t_{s3,begin} = t_{s1,end}, \\ &t_{s3,end} = t_{s0,begin}. \end{aligned} \tag{7}$$

4.2.4 Motion Adaptation without the Lower Body Pose Transition

In motion adaptation without the lower body pose transition, three blending segments are created (Figure 2(d)). The segments in Section 4.2.3 are used, except that the foot-moving phase S_0 cannot exist and the segment for deforming the lower body pose lasts until the end of the motion. That is,

$$t_{s3,end} = t_{m1,end}. \tag{8}$$

4.2.5 Selection of Blending Method

When there are several options available, we choose one based on their priorities: motion transition > motion connection > motion adaptation. If there are several options for foot-moving phases in the same blending method category, we use the option with the longest blending range. In other words, we use the option that has the closest foot-moving phase to the core of the input, because it is expected to produce the smoothest transition.

On the other hand, if the core parts of motions overlap with each other, the following motion cannot be executed and will be rejected.

4.3 Optimal Blending Range

For each of the blending segments described in Section 4.2, an optimal blending range is determined within the possible range. The initial time of the range is searched for within the previous motion, while the terminal time of the range is searched for within the following motion. In some cases, either the beginning or end time has already been determined and we only need to find the other, as defined in Section 4.2.

A blending range that is defined by its initial (t_0) and terminal (t_1) times is determined by solving the following optimization problem.

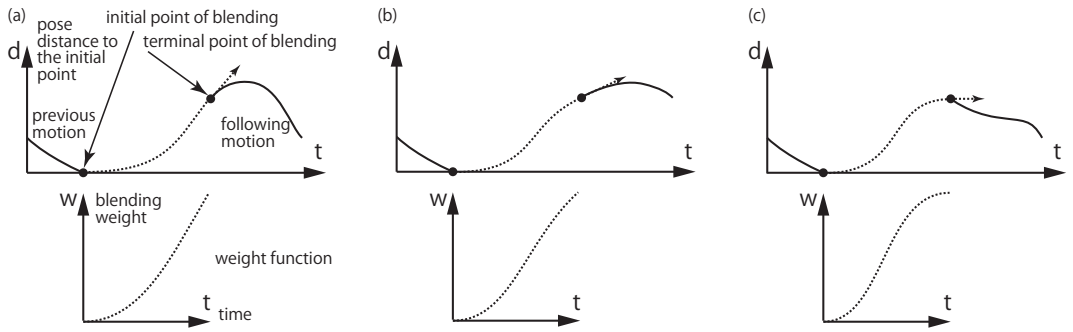

Figure 3: Weight function for pose-to-pose blending.

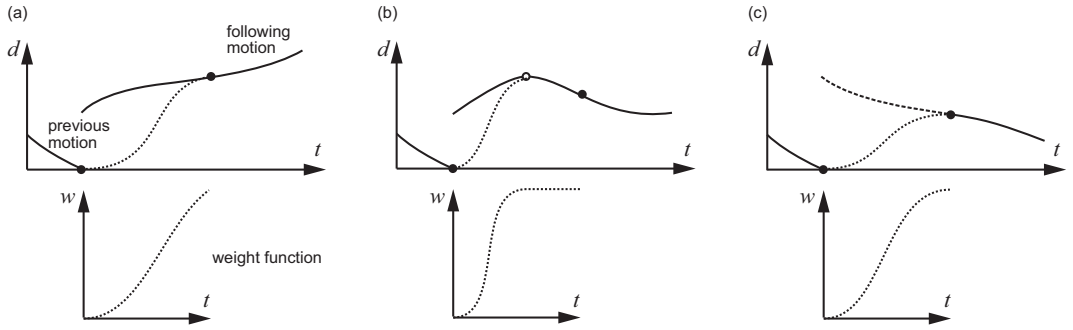

Figure 4: Weight function for pose-to-motion blending.

$$\min_{t_0 \in S, t_1 \in S}(w_t Q_t(t_0,t_1) + w_p Q_p(t_0,t_1) + w_v Q_v(t_0,t_1)) \tag{9}$$

where S is the possible range defined in Section 4.2. Q_t, Q_p, Q_v represent the objective functions for the time, pose, and velocity, respectively. w_t, w_p, w_v are the weights for these terms.

The objective function for the time evaluates the distance between the interval of the foot-moving phase $P(t_{p,begin} \sim t_{p,end})$ and the interval of the blend range,

$$Q_t(t_0,t_1) = |(t_{p,end} - t_{p,begin}) - (t_1 - t_0)| \tag{10}$$

This means that the closer they are to each other, the better. For a pose-to-pose blending segment, this term evaluates the interval of the blending range,

$$Q_t(t_0,t_1) = t_1 - t_0 \tag{11}$$

This means that a shorter blending range is better.

The objective function for the pose evaluates the difference between the poses from two motions at t_0 and t_1. There are several ways to compute a distance between poses. As in [11], we have used the average distance between the positions of primary joints after two poses are aligned so that their pelvis positions and body orientations match. The distance between the pose from the previous motion at time t_0 and the pose from the following motion at time t_1 is

$$Q_p(t_0,t_1) = D(t_0,t_1) = \sum_{i=1}^{n} |\mathbf{p}_i(t_1) - \mathbf{p}_i(t_0)|/n \tag{12}$$

where $\mathbf{p}_i$ is the position of the i-th primary joint at the specified time, and n is the amount of primary joints. When determining a blending segment for the lower body, only the joints in the lower body are used. When determining a blending segment for the upper body, only the joints in the upper body are used.

In general, it is difficult to define the velocity of a motion, because a motion sequence is a multi-dimensional signal. In our method, we

use the distance between poses for computing the velocity in a similar way as in [3]. The objective function for the velocity evaluates the difference between the average velocity of the blending range and the velocity at the pose. The velocity at t in an input motion is

$$V(t) = D(t, t+\epsilon)/\epsilon \tag{13}$$

where ϵ is a small time step. The average velocity between t_0 and t_1 is computed using the distance equation

$$\begin{aligned} Q_v(t_0, t_1) = \quad & |V(t_0) - D(t_0, t_1)/(t_1 - t_0)| + \\ & |V(t_1) - D(t_0, t_1)/(t_1 - t_0)| \end{aligned} \tag{14}$$

We use both the objective functions for pose and velocity, although they may conflict, because the velocity evaluates only the changes of the distance between poses, and we also need the distance.

The weights w_t, w_p, w_v in Equation (9) must be specified. A user may want to control these weights, because motion synthesis involves a trade-off between responsiveness and smoothness. However, manually controlling all these parameters is difficult. As mentioned in Section 3, we allow a user to specify a parameter W between 0.0 (most responsive) and 1.0 (smoothest). We prepare two sets of weights that are linearly blended based on the parameter W.

The optimization problem (Equation (9)) is easily solved by searching for an optimal combination of variables (t_0, t_1) with small intervals, because the parameters have at most two dimensions. A simple coarse-to-fine approach further facilitates the process, first searching for candidates with a large interval, and then narrowing them down to an optimal answer with a small interval.

4.4 Optimal Weight Function

A weight function is used to blend two motions or poses. It is defined as

$$w = f(t) \tag{15}$$

where t is the normalized time, and w is the normalized weight (between 0.0 and 1.0). A linear function or a fixed smooth-in and -out function is typically used for blending. However, these weight functions can generate unnatural motions. Our method determines an optimal weight function for each blending segment.

4.4.1 Pose-to-Pose and Motion-to-Motion Blending

For pose-to-pose and motion-to-motion blending, we consider the velocity at the initial time of the blending segment in the previous motion and the terminal time of the blending segment in the following motion. The weight function is defined as a Hermite curve based on the velocities at the beginning and the end.

The velocity at the terminal time t_1 is computed based on the distance between poses. Using the ratio between the average velocity over the blending range and the velocity at the terminal time, the velocity (slope) at the end time of the weight function $w'(t_1)$ is

$$w'(t_1) = V(t_1)/(D(t_0, t_1)/(t_1 - t_0)) \tag{16}$$

as shown in Figure 3 (a),(b).

However, if the direction of the movements before and after the terminal time are different, the velocity should become zero for a moment at the terminal time. Then, the velocity (slope) is set to 0, i.e., $w'(t_1) = 0$, as shown in Figure 3 (c).

The same process is applied to the velocity (slope) at the initial time. A Hermite curve is defined using the slopes at the initial and terminal points. This ensures continuity in the synthesized motion at the initial and terminal points of the blending range.

The same method is also applied to motion-to-motion blending.

4.4.2 Pose-to-Motion and Motion-to-Pose Blending

For pose-to-motion and motion-to-pose blending, in addition to the velocities at the initial and terminal points, we use the distances between the pose and the poses during the motion to adjust the initial or terminal times.

Here, we consider the pose-to-motion blending example shown in Figure 4. We compute the distances between the initial pose from the previous motion and the poses in the following

motion. Based on the trajectory of the distance, one of three types of weight functions is applied.

If the distance increases monotonically, a weight function is determined so that it also increases monotonically between the initial time in the previous motion and the terminal time in the following motion, as shown in Figure 4 (a). In the same way as pose-to-pose blending, the weight function is represented by a Hermite curve and we determine the slopes at the initial and terminal points using the velocity of the distances. Because the motion segment in the following motion reaches the terminal pose in the same direction as the initial pose from the previous motion, this blending generates a smooth transition.

If the end point is in the middle of following the motion segment, we use a weight function that becomes 1.0 at this point, as shown in Figure 4 (b). A Hermite curve is created in the same way. In this case, the slope at the point becomes zero. If we simply blend the motion segment until the terminal point, the blended motion goes past the terminal point and then comes back. Then the pose at the end point is not realized. Therefore, we adjust the terminal point.

If the distance trajectory decreases monotonically, it is difficult to avoid this problem with any weight function. In this case, instead of pose-to-motion blending, we only use the end pose in the following motion and apply pose-to-pose blending (Figure 4 (c)).

The same process is applied to motion-to-pose blending.

5 Interactive Motion Control

This section describes our extension that automatically determines the execution timing of the input motion so that the new input motion is executed as soon as possible after the current synthesized motion. This feature is particularly useful for interactive motion control.

The important constraint when adding an input motion while the synthesized motion is being played is that the new input motion should not change any already played synthesized motions. Because if the current pose is changed, it would produce a discontinuity. To prevent this, all the new blending segments must only exist after the current time. This constraint is applied to determine the blending method and range. Then, we must add the following constraints to the possible ranges of the blending segment S $(t_{s,begin} \sim t_{s,end})$.

$$\begin{aligned} t_{current} &< t_{s,begin}, \\ t_{current} &< t_{s,end} \end{aligned} \tag{17}$$

where $t_{current}$ is the current time when playing the synthesized motion.

When we add an input motion, the system searches for foot-moving phases in the synthesized motion after the core of the last input motion, and before the new input motion's core. If there are foot-moving phases, we either apply motion transition or connection. If not, we apply motion adaptation.

When the blending method and corresponding phases of the previous and following motions have been determined, the following motion is temporally placed based on the phases with a fixed interval (one second in our experiments). Then, we compute the optimal blending ranges. Finally, the execution timing of the following motion is adjusted to an earlier timing, so that there is no gap between the blending ranges and the core parts of the motions.

6 Results and Discussion

We have applied our system to animation editing and interactive character control with dancing and fighting motions, which contain dynamic movements that are considered difficult for motion synthesis methods. For dancing, we chose to use hip-hop dance and Japanese traditional Noh dance motions, because they contain complex and independent movements of the upper and lower body. These motion capture data are either purchased or provided from a research project [17].

Some motion synthesis results are presented in the accompanying video. We compared our results with two conventional methods: pose-based and constraint-based. The pose-based method determines the blending timing based on the pose similarities between two motions. This approach is commonly used in previous presented methods [11, 13, 14, 15, 16]. The constraint-based method [1] determines the

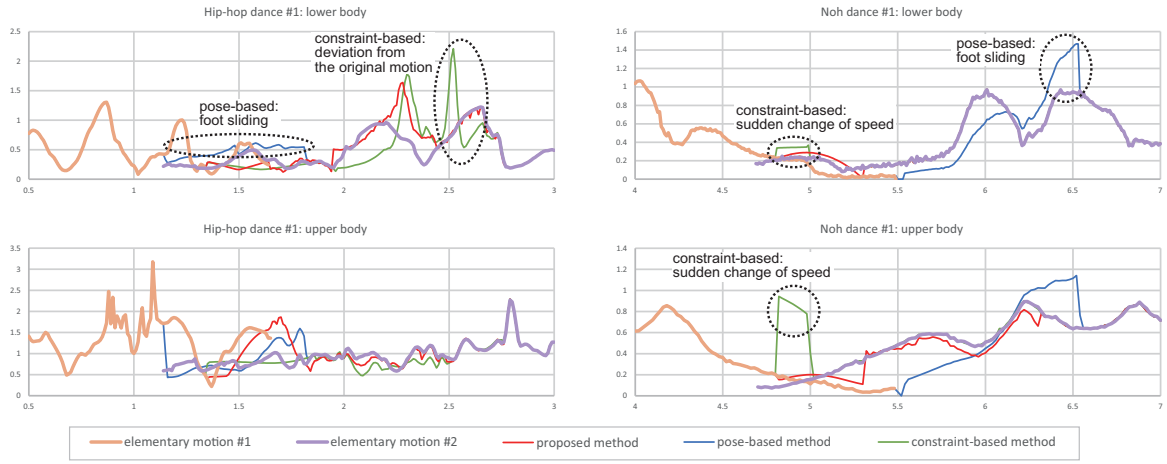

Figure 5: Analysis of synthesized motion. The velocities of lower and upper body over time are plotted.

blending timing based on the foot-ground constraints.

Our method performed well in these results and comparisons. For example, for dance motions with motion transition and connection (the result of blending the foot-moving phase of the previous motion with the following motion) the moving foot moved further than the original motion. Our method determined the blending range for the lower body (the foot-moving phase is elongated) so that the velocity was maintained. In addition, smooth motions were generated for the upper body with a wider blending range. In the fighting example, when blending the flying phase of a jumping motion, the weight function was properly determined so that the pose of at the highest point of the jump was realized.

To evaluate their quality, we analyzed the synthesized motions by calculating the velocities of the lower and upper body over time. By investigating the velocity trajectories, we can find unnatural artifacts such as foot sliding and sudden changes of motion. The velocity of the lower body is calculated from the average velocities of the left foot, right foot, and pelvis. The velocity of the upper body is computed from the average velocities of the right hand, left hand, chest, and head. The results from two cases (a hip-hop dance and a Noh dance) are presented in Figure 5. Several artifacts can be seen in the synthesized motions from the pose-based and constraint-based methods. The pose-based method caused unnatural foot sliding during the support phase. The constraint-based caused a sudden change of speed and deviations from the original motion, because of an inappropriate blending range and weight function.

We also discussed our Noh dance results with some expert researchers and performers. They commented that the synthesized motions look very natural. A Noh dance consists of a series of motion units called Shosa [17]. Even though the core parts of the motion units (Shosa) are formalized, the transitions are not clearly documented and are only learnt through extensive practice. In general, the fluent movements between the core parts of the motions without unnecessary deviations are considered beautiful Noh dance moves. Our method simulated these beautiful movements by professional performers, even though it is not specialized to this kind of dance. We believe that this is very promising result. In the future, we plan to apply our method to other forms of dance and sports.

In terms of computational efficiency, our method works in real-time. In general, optimization processes can take a long time. However, our method is fast because there are at most two variables (the initial and terminal times of a blending segment).

7 Conclusion

In this paper, we proposed an interactive motion synthesis technique. We introduced the optimal blending range and weight function, which are determined for each blending segment of the upper and lower body. Our method can be used for both animation editing and interactive character control. We believe that our method is particularly useful as a game engine for interactive character control, because it does not require a state machine and can use a set of existing motions.

Acknowledgment

This work was supported in part by a Grant-in-Aid for Scientific Research (No. 24500238) from the Japan Society for the Promotion of Science (JSPS). The author would like to thank Reiko Yamanaka, Yukiko Nakatsuka, Takeshi Seki and Masami Iwatsuki for their help in evaluating the Noh dance motions and their valuable comments.

References

[1] M. Oshita. Smart motion synthesis. *Computer Graphics Forum (Pacific Graphics 2008)*, 27(7):1909–1918, 2008.

[2] A. Witkin and Z. Popović. Motion warping. In *SIGGRAPH '95*, pages 105–108, 1995.

[3] J. Wang and B. Bodenheimer. Synthesis and evaluation of linear motion transitions. *ACM Transactions on Graphics*, 27(1):Article No. 1, 2008.

[4] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

[5] P. Glardon, R. Boulic, and D. Thalmann. On-line adapted transition between locomotion and jump. In *Computer Graphics International 2005*, pages 44 – 50, 2005.

[6] S. Ménardais, R. Kulpa, F. Multon, and B. Arnaldi. Synchronization for dynamic blending of motions. In *Symposium on Computer Animation*, pages 325–335, 2004.

[7] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH '96*, pages 147–154, 1996.

[8] V. B. Zordan, A. Majkowska, B. Chiu, and M. Fast. Dynamic response for motion capture animation. *ACM Transactions on Graphics*, 24(3):697–701, 2005.

[9] H. P. H. Shum, T. Komura, and P. Yadav. Angular momentum guided motion concatenation. *Computer Animation and Virtual Worlds*, 20(2-3):385–394, 2009.

[10] L. Ikemoto, O. Arikan, and D. Forsyth. Quick transitions with cached multi-way blends. In *Symposium on Interactive 3D Graphics 2007*, pages 145–151, 2007.

[11] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[12] Y. Lee, K. Wampler, G. Bernstein, J. Popovic, and Z. Popovic. Motion fields for interactive character animation. *ACM Transactions on Graphics*, 29(6):Article No. 138, 2010.

[13] M. Gleicher, H. J. Shin, L. Kovar, and A. Jepsen. Snap-together motion: Assembling run-time animations. In *Symposium on Interactive 3D Graphics 2003*, pages 181–188, 2003.

[14] J. McCann and N. S. Pollard. Responsive characters from motion fragments. *ACM Transactions on Graphics*, 26(3):Article No. 6, 2007.

[15] A. Treuille, Y. Lee, and Z. Popović. Near-optimal character animation with continuous control. *ACM Transactions on Graphics*, 26(3):Article No. 7, 2007.

[16] J. Min and J. Chai. Motion graphs++: a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics*, 31(6):Article No. 153, 2012.

[17] M. Oshita, T. Seki, R. Yamanaka, Y. Nakatsuka, and M. Iwatsuki. Easy-to-use authoring system for noh (japanese traditional) dance animation and its evaluation. *The Visual Computer*, 29(10):1077–1091, 2013.