Splicing Motion Graphs: Interactive Generation of Character Animation

Kiyotaka Tamada [†] Shinya Kitaoka [‡] Yoshifumi Kitamura [*]
Graduate School of Information Science and Technology,
Osaka University, 1-1 Yamadaoka, Suita, Osaka, 565-0871
Japan
Tel.: +816-6877-5111
E-mail: [†]tamada. kiyotaka@gmail.com, [‡]skitaoka@gmail.com, [*]kitamura@riec.tohoku.ac.jp

***Abstract*—We present a novel method of which synthesizes a variety of transitions for multiple actions of human characters from a small amount of motion data. This method effectively combines both motion graphs and a splicing technique. So we call the method *splicing motion graphs*. The motion graph represents a set of motion segments using graph structure to store the transition possibilities from one segment to another. We independently construct motion graphs for the upper- and lower-bodies to splice them. Furthermore, we propose a method to synchronize the infinite length motions of the upper- and lower-bodies. We show our method can efficiently generate animations by illustrating synthesized motions and by evaluating the generation speed and the generation accuracy of the animations. We conclude our method is fast enough to generate animations at interactive rates, allowing us to generate infinite-length character animations with specified constraints.**

Keywords-motion capture; motion synthesis; character animation; motion graph; splicing

I. INTRODUCTION

Human characters with realistic motions are widely used in a variety of media such as movies, video games, Metaverse, and so on. To reproduce such motions, motion capture is often used because it is reliable and useful. Motion sequences can also be synthesized with a set of motion capture data by using motion graphs [1] which preserve the quality of the original motion data. Here users can control characters who can perform a transition of multiple actions, and even infinite-length animation of a human character can be generated.

A variety of character motions can be reproduced from a limited amount of motion data stored in a database using a splicing technique [2]; $n \times m$ character motions can be generated from n locomotion data and m upper-body actions. This technique is useful but it does not allow us to specify such constraints as splicing points and paths along which to move that are drawn on the ground.

This paper proposes *splicing motion graphs* which synthesizes a variety of transitions of multiple actions of human characters from a small number of motion data by effectively combining both motion graphs and a splicing technique. In addition to simply combining these two techniques, a method to synchronize the motions of the upper- and lower-bodies is proposed. Thus the method can synthesize the realistic motions of human characters by interactively specifying constraints.

II. RELATED WORK

Various techniques have been proposed for generating human animations based on motion graphs [1]. Shin proposed a fat graph that parameterizes similar transitions in the graph [3]. Animations can be controlled by a mouse and a joystick. Moreover, Heck introduced a parametric motion graph that generates a parametrical space from a similar motion segment [4]. In this way, it can generate human animation in real-time. However, these algorithms lose detailed behavior in the motion data because they merge nodes that represent original motion in the motion graphs.

In addition, many techniques for achieving synchronization between two bits of motion data have been proposed. Kovar synchronized with Dynamic Time Warping (DTW) based on posture distance [5]. To generate natural movement animations, this technique only synchronized similar motions. Menardais proposed a real-time synchronization approach [6] that achieved synchronization to combine the supporting legs and DTW. However, this technique needs to search for the ranges that can be synchronized beforehand. In this paper, we achieve synchronization by combining [5] and [6].

III. ANIMATION GENERATION

A. *Method Overview*

To interactively generate animations based on user-specified constraints, we have to execute processing that searches for necessary motions from motion graphs and synchronizes the velocity differential between upper- and lower-body motions after the user draws a path. So we execute other processing as preprocessing. Fig. 1 shows the flow of our method and its two parts: preprocessing and runtime. In preprocessing, we calculate some data that need for generating animation. In runtime, we generate animation based on user-specified constraints.

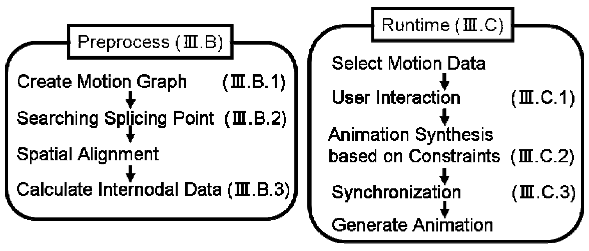

Preprocess (Ⅲ.B)
Create Motion Graph (Ⅲ.B.1)
Searching Splicing Point (Ⅲ.B.2)
Spatial Alignment
Calculate Internodal Data (Ⅲ.B.3)
Runtime (Ⅲ.C)
Select Motion Data
User Interaction (Ⅲ.C.1)
Animation Synthesis based on Constraints (Ⅲ.C.2)
Synchronization (Ⅲ.C.3)
Generate Animation

Figure 1. Flow of our method.

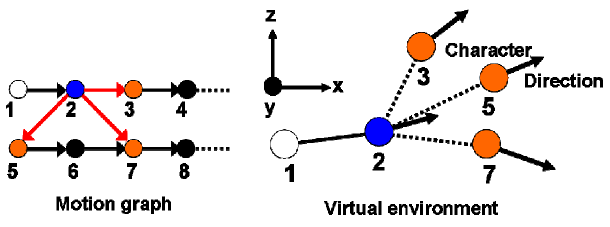


Figure 2. Overview of internodal calculation.

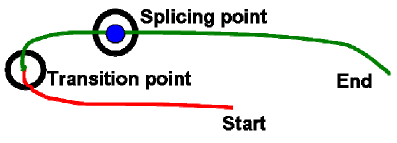


Figure 3. User interaction example.

B. *Preprocess*

1) Creation of Motion Graphs

We represent motions using a standard skeleton model. For splicing, we divide the motion data into upper- and lower-body motions with a central focus on the pelvis based on the method [2].

After defining the motion data, we independently construct two types of motion graphs: an action for the upper-body and a movable one for the lower-body. These motion graphs are constructed from a given motion data set by the standard construction technique [1]. Here, we also use mirroring motion data to deal with the user-specified constraints flexibly. Then, we simplify the constructed motion graphs to reduce calculation costs by merging nodes without branch edges in the motion graph.

2) Searching Splicing Points

Heck et al. [2] spliced two motions of finite length by temporal synchronization. But we cannot use this method because we treat motions of infinite length. Therefore, we search for splicing candidate points based on the degree of similarity between upper- and lower-body motions. First, we calculate the posture distance between the upper- and lower-body motion graphs using the method of Kovar et al. [1]. We divide the posture distance into upper- and lower-body parts. Because the posture distance of the upper-body has a large value, the posture distance of the lower-body is not applied the calculation of degree of similarity. In addition, we use a segment center of gravity of the specific upper-body motions to calculate the posture distance of the upper-body motions. The segment center of gravity is one body segment parameter, calculated by dividing the point of a vector based on the mass ratio and its center [7].

Next, in searched splicing candidate point, we calculate an amount of rotation of a pelvis joint based on the technique of Horn et al. [8], and correct spine of upper-body to satisfy an interlocking relation between upper- and lower-body.

3) Calculation of Internodal Data

Motion data acquired by the motion capture device only maintain the position data in the skeleton data for each frame and the rotation data for each joint. Therefore, we need to do a conversion and a reverse conversion for the skeleton data for every intermodal transition when we search for motion graphs to generate animations shown in Fig. 2. However, such processing takes a long time. To reduce the searching time for the motion graphs, we calculate the intermodal data that are needed to search for the motion graphs. We calculate the number of frames, the Euclidean distance between nodes, and a transform matrix.

C. *Runtime*

1) User Interaction

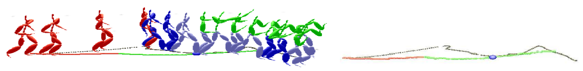
Our method provides three interactions: drawing a path, setting transition points, and setting splicing points. Figure 3 shows a user interaction example. Users can freely draw a path (red line) for generating animation on a 2D ground. Additionally, they can freely set transition points (the point changing line color) and splicing points (blue point) on the drawn path. The transition points change the lower-body motion patterns and the splicing points change the combination between the lower- and upper-body motions. Therefore, we can generate animation in which the user edits the action timing and the moving direction on a 3D virtual space.

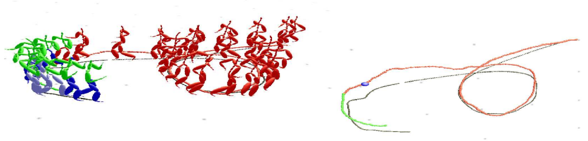
2) Animation Synthesis based on Constraints

To interactively generate animations based on the user-specified constraints, we need to quickly search for the nodes of motion graphs. We search for them by a greedy algorithm to decrease the number of searches. However, if we choose the shortest transition distance for the user-drawn path in all cases, the distance errors of the transition increase. Therefore, we set a threshold of distance errors to prevent an increase of the distance errors between generating the animation's moving direction and the user-drawn path. If the distance errors exceed the threshold when searching for the shortest distance of transitions, we return to a former node and restart the search from the second shortest transition distance. As a result, we can prevent increased distance error.

3) Synchronization between Upper- and Lower-bodies

Spliced motion has a different velocity between the upper- and lower-bodies, because those bodies have different motion data. However, as described in Section Ⅲ .B.2, because animation generated from motion graphs is of infinite lengths, we cannot calculate a number of frames. So we cannot use temporal synchronization for the generated animation. Thus,


(a) straight (3.2 sec)


(b) left (22.4 sec)

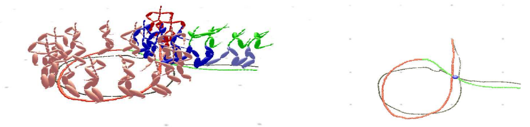

(c) right (18.8 sec)

Figure 4. Generated animations.

we synchronize the motion timing of infinite-length animation based on constraint points.

First, we search splicing point from generated node sequence in Section Ⅲ.C.2. We use splicing point as a starting point of synchronization because splicing point has been already synchronized upper- and lower body. Then, if we can find second splicing point or transition point, we make the synchronizing range between splicing points or between splicing point and transition point. If we cannot find any splicing point and transition point, we set the synchronizing range based on supporting legs proposed by Menardais [6].

Next, we achieve synchronization using the technique of Kovar [5] within the set range. Finally, in order to correct the matching of the grounding timing in a motion, we correct the generated animation with the technique of Shin [9] and Inverse Kinematics (IK) based on the Cyclic-Coordinate Descent (CCD) method. Because the amount of calculation in the CCD method is light, convergence of the solution is fast.

IV. RESULTS AND DISCUSSION

We evaluate the generation speed and the generation accuracy of the animations and show the generated animations based on the user-drawn path and the user-specified constraints.

We used motion data published in the Motion Capture Database of Carnegie Mellon University (http://mocap.cs.cmu.edu). The CPU and PC memory were Core 2 Duo 2.4 GHz and 2 GB RAM. But we used a 1 core CPU with single thread programming. In this evaluation, we used a walking motion (1506 frames) and a running motion (761 frames) for the lower-body motions and a boxing motion (1500 frames) for the upper-body. The walking motion goes forward and backward, and the running motion runs in a circular pattern.

We used three kinds of paths for the evaluation: straight line, turns to the left, and turns to the right. These paths were

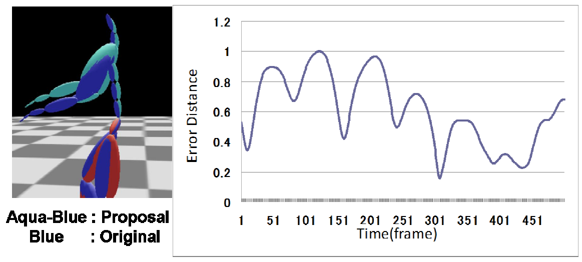


Figure 5. Error distance of spine between spliced and original carry motions.

also set by splicing and transition points. Moreover, we used the improved greedy algorithm to search for user-drawn paths.

In addition, we show the computation time for each preprocessing and simplification result of the motion graphs. The construction times of the lower- and upper-body motion graphs were 70 and 10 minutes, respectively. The search time for the splice candidate points was 30 minutes, and the computation time for the information between nodes in the motion graphs was three hours. After simplification, the lower-body motion graph reduced 240 nodes, the upper-body motion graph reduced 64 nodes.

Figure 4 shows generated animations for three kinds of paths. The red motion is running, the blue motion is walking, and the green motion is boxing. These result show generated animations along the user-drawn paths with the transition and the splicing as the constraints. In addition, there are no velocity differentials of upper- and lower-body motions and no foot slidings. These results show that our method generated animations of basic movement patterns from several bits of motion data. Thus, we can generate complex animations by mixing these basic operations. In all the generated animation results, computation times from when the user started to draw a path and when the animation was generated were between 3 and 23 seconds. So our method is fast enough to generate animations at interactive rates.

Next, we show the animation generation accuracy and calculate the sum of the Euclidean distance between the original upper-body and the spliced upper-body spine. Then we calculate the error distance by normalizing them, because our method correctly spliced the upper-body spine by approaching the original upper-body. Fig. 5 shows the error distance of the spine between the spliced and the original carry motions. The left figure shows the posture difference when the error distance is the worst. From the left figure, the two postures are similar even in the worst case. The right figure shows that the average error distance is small. Our method can generate animation while preserving the posture features possessed by the original motion data.

V. CONCLUSION AND FUTURE WORK

This paper presented a novel method for editing human motion data. Our method interactively generates infinite-length character animations with user-specified constraints. To generate the animations, we construct two motion graphs for the upper- and lower-bodies. Then we splice the generated upper- and lower-body motions. Our method can generate a variety of motions from a small amount of motion data. Moreover, we synchronized the upper- and lower-body motions to align the velocity differential. Therefore, we illustrated that our method can intractably generate animations along a user-specified path with user-specified transition and splicing points.

Our method has some limitations. For example, it does not strictly consider constraints by environment. In future work, we will adapt such constraints as obstacles and stairs to our method and extend it for handling human motion data that do not stand on feet, e.g., forward rolling, crawling, and sliding.

REFERENCES

[1] L. Kovar, M. Gleicher, and F. Pighin, “Motion graphs,” *ACM Transactions on Graphics (SIGGRAPH '02)*, Vol. 21, No. 3, pp. 473-482, 2002.

[2] R. Heck, L. Kovar, and M. Gleicher, “Splicing upper-body actions with locomotion,” *Computer Graphics Forum (Eurographics '06)*, Vol. 17, No. 3-4, pp. 219-227, 2006.

[3] H. J. Shin and H. S. Oh, “Fat Graphs: Constructing an interactive character with continuous controls,” In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*, pp. 402-408, 2006.

[4] R. Heck and M. Gleicher, “Parametric motion graphs,” In *Proc. of the Symposium on Interactive 3D Graphics and Games (i3D '07)*, pp. 129-136, 2007.

[5] L. Kovar and M. Gleicher, “Flexible automatic motion blending with registration curves,” In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*, pp. 214-224, 2003.

[6] S. Menardais, R. Kulpa, F. Multon, and B. Arnaldi, “Synchronization for dynamic blending of motions,” In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*, pp. 325-335, 2004.

[7] Michiyoshi Ae, Hai peng Tang, and Takashi Yokoi, “Estimation of Inertia Properties of The Body Segments in Japanese Athletes,” *Japanese Journal of Sports Science*, Vol. 15, No. 3, pp. 155-162, 1996. (in Japanese)

[8] B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *Journal of the Optical Society of America A*, Vol. 4, No. 4, pp. 629-642, 1987.

[9] H. Shin, J. Lee, S. Shin, and M. Gleicher, “Computer puppetry: an importance-based approach,” *ACM Transactions on Graphics (SIGGRAPH '01)*, Vol. 20, No. 2, pp. 67-94, 2001.