http://dx.doi.org/10.5392/IJoC.2015.11.1.015

Linear Interpolation Transition of Character Animation for Immediate 3D Response to User Motion

Sooyeon Lim
School of Liberal Arts
Dongyang University, Yeongju, 750-711, South Korea

ABSTRACT

The purpose of this research is to study methods for performing transition that have visual representation of corresponding animations with no bounce in subsequently recognized user information when attempting to interact with a virtual 3D character in real-time using user motion. If the transitions of the animation are needed owing to a variety of external environments, continuous recognition of user information is required to correspond to the motion. The proposed method includes linear interpolation of the transition using cross-fades and blending techniques. The normalized playing time of the source animation was utilized for automatically calculating the transition interpolation length of the target animation and also as the criteria in selecting the crossfades and blending techniques. In particular, in the case of blending, the weighting value based on the degree of similarity between two animations is used as a blending parameter. Accordingly, transitions for visually excellent animation are performed on interactive holographic projection systems.

Key words*: Linear Interpolation, Animation Transition, Crossfade, Blending, Source Animation, Target Animation.*

1. INTRODUCTION

The emergence of computer technologies provides opportunities for scholars to explore human-computer interaction [16]. The studies for natural and intuitive interaction have been made constantly in the human-computer interaction and the field of computer graphics which leverages that. Especially the motion-based 3D interaction which interacts with a digital content by using the human body has been applied in various fields such as the game, education, simulation and so on. In order to implement the reliable and efficient 3D interactive application systems, increasing the recognition rate of the 3D motion and at the same time, the study on the method for natural interaction are necessary and recently, there has been the various studies tried.

The goal of this study is to design the flexible and simple interface where the user can be friendly-accessed to 3D system. Therefore, the method to allow the natural transition between animations is proposed for the immediate response to the dynamic motion of the user in the interaction system between the user and the 3D contents.

The animation system universally widely used in the game engine is equipped with the system to move the bone by having the character's skeleton, which is generally called the bone, in conjunction with three-dimensional information of the character through the technique called skinning. The problem of this animation system is that the animation work is growing exponentially as the animation being used and the kind of the utilized character increase, because the animation must be made separately in the case that there is even a little difference in the structure of the bone. In order to reduce the excessive animation work, in recent years, the study on the animation synthesis [8], [11] which mixes two or more animation and generate a different animation and the study on retargeting [4], [12] which allows the recycling in the various characters by making the animation based on the generalized animation system for one character model.

It is difficult to make a natural transition in the data-driven animation synthesis study. The work for a natural transition is needed especially when connecting the two animations because there is the low degree of similarity between the last frame of the source animation and the first frame of the target animation.

This study is proposing the transition method that allows the animation transition, which happens in real-time and dynamically in the interaction system between the user and the 3D contents, to occur smoothly without bouncing phenomenon.

The proposed transition method is basically using the linear interpolation. The natural and satisfactory transition could be produced by generating the intermediate posture and doing the linear interpolation from automatically calculating the length of the transition length and performing a weighted blending.

* *Corresponding author, Email: sylim@knu.ac.kr*
Manuscript received Sep. 22, 2014; revised Mar. 17, 2015; accepted Mar. 24, 2015

2. BACKGROUNDS

Recently, the studies on the characters where the real-time control is possible using the existing pieces of the motion are actively in progress focusing on the game field. But, interactive character animation still lacks the fluidity and variability of real human motion despite of decades of advances.

In today's game, the animation blending is an essential element for the smooth animation of the character. Animators use two types of animation data. One is making separately for each of the basic animations (for example, walking, running, idle, shoot, etc.), the other is using a corpus of motion capture data [2]. At any time in game play, the users can want to perform the multiple animations in succession or do a mixing of them and will want the smooth transition with no sudden movement bounce.

This can be resolved with the animation transition techniques which generate a long animation by connecting the short animated pieces seamlessly and the animation blending techniques which mix the similar animation pieces each other based on the blending weights and generate the new animation. The blending-based parametric synthesis techniques, which calculate the blending weights corresponding to the given input parameter by interpolating the existing motions, are widely used because of its excellent control and efficiency.

Kovar [11] automatically generates the motion graph which controls the motion of the three dimensional character and creates the new motion by blending based on the weighting. It is difficult for the user to manually input a weight vector for the motion control and has the disadvantage which is difficult to predict the result motions. Thus, while the parametric function is sometimes exploited for the intuitive motion control, Kovar [10] proposed the weighting calculation function based blending synthesis technique by K-nearest neighbors interpolation.

And Rose [3] set the critical moment in the motion as the key time and proposed the way of finding the corresponding motion frame by normalizing the motions with respect to time based on this and created the intermediate actions that allow the natural connection and inserted in the transition interval using the weighting vector calculation function.

Ikemoto [9] created all possible transitions in advance and studied on the method of automatically selecting most natural transition in execution phase and Wang [7], [8] recognized that the length of blending interval is very important element in the visual representation of animations when transition and then conducted the study on that. He used the empirical methods to optimize the weighting which is for changing the length of a transition interval dynamically according to the degree of similarity between the source and the target or calculated the blending period assuming a transition point is given.

Lee, Y [17] proposed a structure called a motion field that finds and uses motion capture data similar to the character's current motion at any point. He used reinforcement learning to choose between these possibilities at runtime the direction of the state space can be altered, allowing the character to respond optimally to user commands.

In this study, the moment when the new motion of the user is recognized is set to the transition point and the length of the transition interval is automatically calculated according to the remaining playing time of a source animation and then one of two methods (crossfade and blending) is applied to the target animation and gets transitioned. Especially the weighting when performing the transition using blending was set directly using the similarity between the source animation and the target animation

3. ANIMATION TRANSITION

3.1 Linear Interpolation Transition

The most basic way of the continuous process of the animation is the linear interpolating transition. The linear transition is the method to interpolate linearly the positions and rotation values at the last frame of one animation and the first frame of the next animation. It has the advantages of the easy implementation and short execution time as the animation method traditionally used in the field of animation.

In 3D interaction system, users are constantly producing the new motions and voices information. When one motion/voice is recognized, the system plays the corresponding animation/ sounds for it. When the new user's information is detected before the end of playback of corresponding contents, the system needs to stop playing the contents currently being performed and play the new corresponding contents. For the example that the corresponding contents is the recognized motion, it corresponds to that the character is greeting with right hand raised and suddenly calling hurray with both hands raised. In such situations, the users want the natural connection without bouncing phenomenon in the transition from one animation to another animation. The moment the user's new motion is recognized becomes the transition point so that the fragmentation of the animation should be implemented. In addition, the animation's transition needs to be done in real time and should not lose continuity.

If you switch to another animation, you can specify a new animation using the play() function the way you are running one of the animation. As a result, the run immediately stops and the new animation is executed in animation system. However this method seems unnatural. So Unity provides two special animation functions. One is the CrossFade() function which changes two animations gradually and the blend() function which generates an intermediate motion using a weight function.

Kinect sensor sends and receives the image information at 30 frames per second and each frame has the three-dimensional coordinate for 20 joints. It is a very important issue to determine the frame of the target animation, to which the transition happens, from the particular frame of the source animation. We set the blending length of using the normalized remaining time of the source animation based on the linear interpolation and thereby, select and apply one of two functions, the cross fade or blending and implement the transition.

In this study, it was processed as shown in Fig. 1 using Unity's animation control functions which have the function of easily controlling and interpolating the animation's transition.

```
while(animation.IsPlaying() && user.MotionCheck())
{
    If (source_animation.normalizedPlayTime) > T
        animation.CrossFade(animation, fadeLength)
    else
        animation.Blend(animation, targetWeight,
                        fadeLength)
}
```

Fig. 1. Proposed algorithm for animated transition

If the certain animation is running and the new motion of the user is recognized, the normalized play time and threshold value are compared and if the play time is identified above the threshold value, then the transition is implemented with the crossfade and if it's less than the threshold value, the transition is implemented using blending function.

3.2 Crossfade and Blending

As mentioned earlier, when the sudden and unpredictable animation transition happens by the user, we distinguish the crossfade and blending and use them depending on the situation in order to overcome the degree of bouncing of the animation and make the smooth animation. This distinction is to implement the crossfade only without the blending if the animation's remaining time is short. It means that performing a quick transition to the target animation gives more smooth visual effects to the user no matter of the source animation reaching the end of play.

In this study, the threshold value *T* was tested in Figure 1 through the repeated experiments based on the user experience. As a result, it can be found that if the threshold value is more than 0.6, then the crossfade or if not, applying the blending has shown the natural animation representation.

The transition length (*fadeLength* in Fig. 1) is determined using a normalized playing time of the source animation like the following equations (1) ~ (3). The normalized playing time of the source animation is getting by normalizing the ratio of the play time to the length of total source animation and the remaining time is 1.0 minus the normalized playing time. Lastly, the transition length is set to the time of the total length of the target animation multiplied by the normalized remaining time.

$$s\_normailzedPlayTime = s\_currentPlayTime / s\_PlayTime; \quad (1)$$

$$s\_normailzedRemainTime = 1.0 - s\_normailzedPlayTime; \quad (2)$$

$$t\_TransitionLength = t\_PlayTime \times s\_normailzedRemainTime; \quad (3)$$

As Fig. 2 is the case where the normalized playing time of the source animation is 0.86, it shows the animation transition process applying the crossfade. In a picture, the source animation is representing for the appearance of the right hand greeting and the target animation is expressing for the appearance of shaking hands with the right hand.

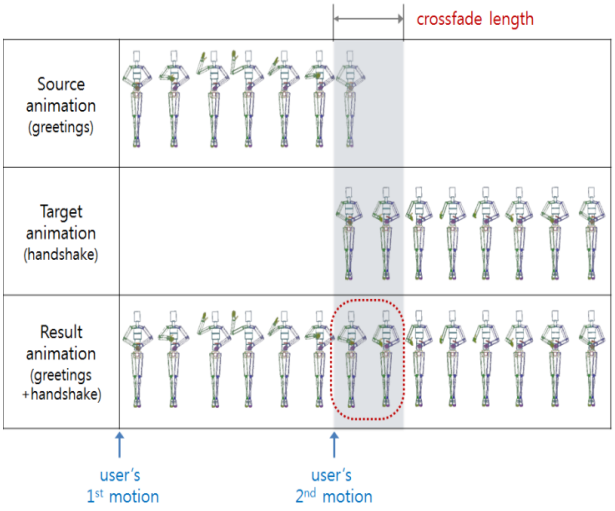

Fig. 2. Animation transition by Crossfade (*s_normailzedPlayTime* =0.86)

Fig. 3 shows that the normalized playing time of the source animation is 0.57 and shows the process of the transition using the target animation and motion blending scheme. The blending weighting based on the vector is set directly using the similarity of the source animation and the target animation.

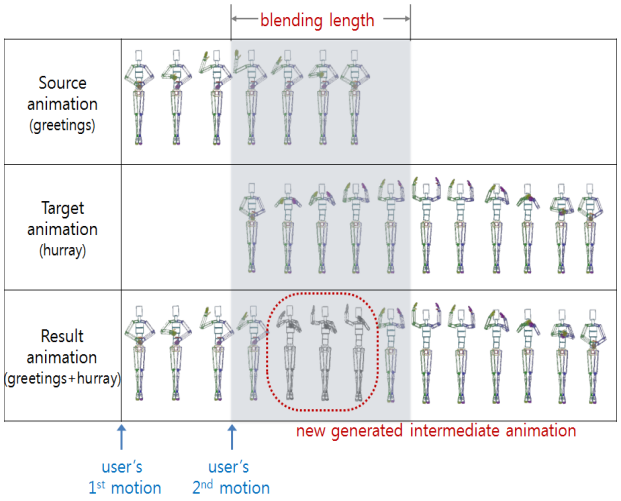

Fig. 3. Animation transition by Blending (*s_normailzedPlayTime*=0.57)

The animation similarity is calculated using the change information of 20 joints used in Kinect. The motion at time t can be expressed as follows.

$$m(t) = \{J_1(t), J_2(t), J_3(t), \ldots, J_{20}(t)\} \quad (4)$$

In the equation above, $J_i(t)$ is the information of Joint *i* received from Kinect.

The animation can be described as the vector shown below and each state s_i consists of a pair (m_i, m_{i+1}) of the consecutive frames. We have all animation states starting from the ready state (s_0) of the hands together and going to original position, starting posture (s_0) after performing a particular movement.

$$s = (s_0, s_1, s_2, s_3, \ldots, s_n) \quad (5)$$

The similarity of animation s and t is easily found through using vector cosine similarity calculation method. In the similarity calculation method, the animation expresses in vector and calculates its degree to find out the distance of the animations similarity. This method is generally used in information retrieval and so on.

It is defined by the motion and the number of frames like the animation state $s_i = (m_i, f)$ and the similarity measurement of the vector information of animation s and t is set after pre-calculating using Cosine Similarity [7].

$$Similarity_{cos}(s,t) = \frac{\sum_{i=1}^{n} s_i t_i}{\sum_{i=1}^{n} {s_i}^2 \ \sum_{i=1}^{n} {t_i}^2} \quad (6)$$

The users can implement the various motions possible at each frame while performing the animation and our system calculates the weighting of the blending in real time and implements the transition by referencing the degree of similarity between the pre-set source and the target animation.

4. IMPLEMENTATION AND RESULT

4.1 Interaction system with user's motion/voice

We develop direct and natural interaction system with 3D digital content(character) using user's motion and voice. We use a depth camera to track user's voice and motion. The information obtained is used to manipulate the state of the animations of character.

Kinect is one of the integrated sensing devices and uses the skeleton algorithm for the motion capture. Kinect consists of RGB camera, depth perception sensors and multi-array microphone. And using this system, the overall shape of the person's Kinect 3D motion capture, facial recognition, and voice recognition is possible [14].

The development of our system uses the libraries such as Kinect SDK [18], KUInterface [19], etc based on Unity 4.0. In this study, 16 user's motions and 21 corresponding animations are set as the test object.

Fig. 4 is showing the 3D interactive system overview using the virtual character and motion/audio which this study has proposed.

4.1.1 Animation import using the animation separation

An animation that is made through External graphic software goes through FBX format and users setting to converting in Unity's animation system. Unity provides two conversion methods.

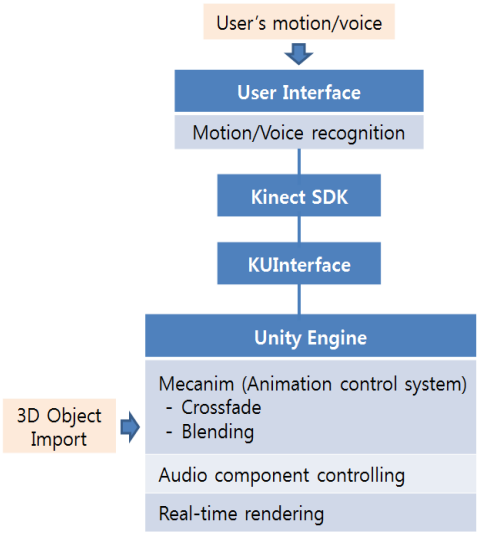

Fig. 4. Configuration of 3D Interaction System

The first is to save all animation in a FBX file. User specifies a frame cycle of the animation and each of the animation to be generated automatically in separated small clips or a single total clip. Although it has advantage that all animations is included a single file, the FBX file size is increased. And the management of animations becomes more complex in proportion to the number of animation. The second is to save each animation in separate FBX files. User extracts from the animation information in Unity system.

For the implementation of the previously introduced method, the character created by Autodesk's 3dsMax is exported to FBX format and then it's imported in the Unity engine. We saved animations which are about the corresponding motions to the separate FBX files by category and imported. This has the advantage to quickly determine whether each animation is playing accurately.

After connecting the animation files and audio files which includes all types of information on the imported 3D objects, it's rendering in real time by performing the response animation according to the user's motion which is recognized from Kinect, depth camera. 'KUInterface' of University of Central Florida was used as the plugin for the interworking between Unity 3D engine and Kinect SDK.

4.1.2 Animation transition using Unity

Unity has the key frame type of the animation system built in and provides the animation component which enables the user to control the play of the animation and the animation view interface where the user can create the animation curve directly.

As Unity has the animation system called Mecanim. Mecanim can be used for all animations by calling the script function during playing animation. Animation component is used to play the animation. The user can assign the animation clip to the animation component and control the play of the animation through the script. Fig. 5 is showing the process of allocating all animation clips to the character which is the animation component in Unity.

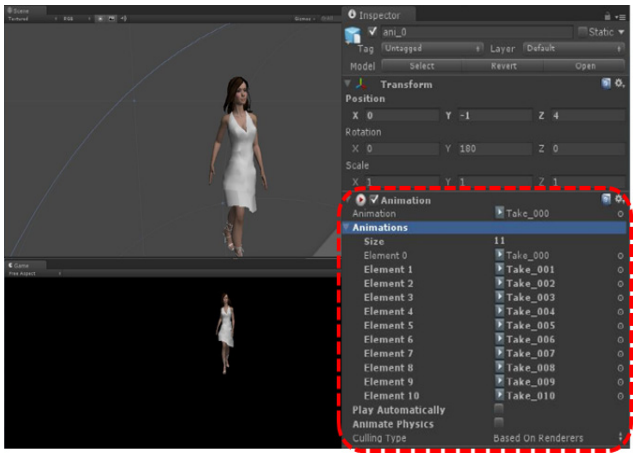

Fig. 5. Process of allocating animation clips to the components

The animation system in Unity is weight-based and implements results by combining each animation clips based on this value. While one animation is running, if the new animation is indicated using play() function, the currently playing animation must stop and perform the new animation immediately. As such method looks unnatural, it supports the control functions such as the animation crossfade, animation blending, animation mixing, additive animation, etc.

4.2 Exhibition result

We have the composition of the interactive environment with holographic characters using a transparent screen in order to verify the animated transition technology of the proposed system. The system outputs a corresponding animation and reply voice for input audio and motion information of the user.

Fig. 6 shows the actual appearance of the properties of the 3D character which change according to the user's motion in the exhibition.

Fig. 6. Property change of the character according to the user's motion information

Table 1 below shows recognition result in motion and voice information user tried in front of the camera.

Table 1. Recognition results in motion and voice of user

	Number of trial	Number of recognition	recognition rate
Single_motion	581	462	79.5%
Blended_motion	738	579	78.5
voice	785	343	47.5%

The number of the blended animation generated by the user's intention is 738 (56.0%) out of a total of 1,319 input motion. User has responded that natural animation transition took place for the 78.5% of the total blended animation.

5. DISCUSSIONS AND CONCLUSION

Recently, many researchers are having the studies on the motion production technology in the various 3D wide contents fields of Games, movies and Virtual reality. Many studies on the animation synthesis using the existing parameter space mostly has the objective of reducing the time and cost required to produce by using the blending to derive the new animation from the motion capture animation.

In this study, it seeks to provide an efficient way which allows the average user who does not have expertise to interact with the virtual 3D character in real time using their motion and voice. The method which enables the smooth transition without the bouncing of the corresponding animation was investigated for the immediate response to the user information which is recognized from the outside successively. This can be used in the real-time system such as a variety of 2D/3D interaction systems or the games.

In the experimental results of the crossfade transition and blending based interpolation for the sample data, the users have shown the considerable satisfaction. Computing the transition length of the target animation dynamically and applying the similarity-based weighting with the target animation in the blending parameter allow the transition of the visually excellent animation.

However, it's unfortunate that the animation's location of the sample data is fixed and the upper body-oriented tendencies are outstanding. It's considered that it's because the development goal of the system was set on the face-to-face conversation with the virtual character. Thus, together with the introduction of the story which considers the various situations, it needs the corresponding animation contents. In addition, as the semi-automatic input of the animation similarity takes a lot of money and effort, the further study in the future for calculating the similarity between animations in real time is necessary.

REFERENCES

[1] A. Egges, G. Papagiannakis, and N. Magnenat-Thalmann, "An Interactive Mixed Reality Framework for Virtual Humans," Cyberworlds 2006, CW'06. International Conference on IEEE, 2006, pp. 165-172.

[2] A. Treuille, Y. Lee, and Z. Popović, "Near-optimal character animation with continuous control," ACM Transactions on Graphics (TOG), vol. 26, no. 3, 2007, p. 7.

[3] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen, "Effective Generation of Motion Transitions using Spacetime Constraints," SIGGRAPH'96, 1996, pp. 147-154.

[4] C. Hecker, B. Raabe, R. Enslow, J. Weese, J. Maynard, and K. Prooijen, "Real-time Motion Retargeting to Highly

Varied User-Created Morphologies," ACM Transactions on Graphics (TOG), vol. 27, issue 3, Article no. 27, 2008.
[5] G. Salton, A. Wong, and CS. Yang, "A vector space model for automatic indexing," Communications of the ACM, vol. 18, no. 11, 1975, pp. 613-620.
[6] J. Lee, J. Chai, PSA. Reitsma, JK. Hodgins, and NS. Pollard, "Interactive control of avatars animated with human motion data," ACM Transactions on Graphics (TOG), vol. 21, no. 3, 2002, pp. 491-500.
[7] J. Wang and B. Bodenheimer, "Computing the Duration of Motion Transitions: An Empirical Approach," ACM SIGGRAPH Symposium on Computer Animation, 2004, pp. 335-344.
[8] J. Wang and B. Bodenheimer, "Synthesis and evaluation of linear motion transitions," ACM Transactions on Graphics (TOG), vol. 27, issue 1, Article no. 1, 2008.
[9] L. Ikemoto, O. Arikan, and D. Forsyth, "Quick Transitions with Cached Multi-way Blends," ACM Symposium on Interactive 3D Graphics, 2007, pp. 145-151.
[10] L. Kovar and M. Gleicher, "Automated extraction and parameterization of motions in large data sets," ACM Transactions on Graphics (TOG), vol. 23, no. 3, 2004, pp. 559-568.
[11] L. Kovar and M. Gleicher, "Flexible Automatic Motion blending with Registration Curves," ACM SIGGRAPH/ Eurographics Symposium on Computer Animation, 2003, pp. 214-224.
[12] M. Poirier and E. Paquette, "Rig retargeting for 3D animation," Proceedings of Graphics Interface 2009, 2009, pp. 103-110.
[13] S. Lim and S. Kim, "A Study on Case Analysis of media art works using Spatial Interaction," Journal of Digital Design, vol. 13, no. 2, 2013, pp. 255-264.
[14] S. Lim and S. Kim, "Holographic Projection System with 3D Spatial Interaction," Proceedings of the 1st International Conference on Advanced Data and Information Engineering (DaEng-2013), vol. 285, 2013, pp 409-416.
[15] Y. Chow, "3D spatial interaction with the Wii remote for head-mounted display virtual reality," Proceedings of World Academy of Science, Engineering and Technology, 2009, pp. 377- 383.
[16] Y. Kim and M. Kim, "An Interactivity-based Framework for Classifying Digital Games," International Journal of Contents, vol. 6, no. 4, 2010, pp. 35-38.
[17] Y. Lee, K. Wampler, G. Bernstein, J. Popović, and Z. Popović, "Motion fields for interactive character locomotion," ACM Transactions on Graphics (TOG), vol. 29, no. 6, 2010, p. 138.
[18] Kinect SDK, http://www.microsoft.com/en-us/kinect for-windows/
[19] Kinect SDK-Unity 3D Plugin, http://www.eecs.ucf.edu/ isuelab/

Sooyeon Lim
She received the B.S. in electronics engineering from Kyungpook National University, Korea in 1988, respectively and also received M.S., Ph.D. in computer engineering from Kyungpook National University, Korea in 1991, 2006 respectively. Currently she is adjunct professor in Dongyang University and Ph. D. Candidate of DigitalMediaArt department in Kyungpook National University. Her research interests are Interactive Art, Motion Graphics and 3D Holographic Projection System.