A NOVEL FRAMEWORK FOR 3D COMPUTER ANIMATION SYSTEMS FOR NONPROFESSIONAL USERS USING AN AUTOMATIC RIGGING ALGORITHM

Natapon Pantuwong and Masanori Sugimoto
The University of Tokyo

ABSTRACT

This paper presents a novel framework for developing automatic animation systems, which accept a 3D model that is created at runtime. Previous systems cannot deal with such a 3D model because it needs to be prepared by a manual process (rigging) that may not be suitable for nonprofessional users. The proposed framework solves this problem by employing an automatic rigging algorithm. Our algorithm can generate an animation skeleton for a given 3D model automatically, including the anatomical meaning of each joint. The relationship between motion data and this animation skeleton is created by identifying the corresponding joints in the motion data and the skeleton. The motion data for each joint is transferred automatically to the created skeleton using the correspondence between the animation skeleton and the motion data, and the animation can be generated without any user intervention.

In this paper, we also discuss a method for increasing the number of motions in the database. We develop several motion editing techniques, such as editing-by-propagation and editing-from-record, which can reduce the difficulties with typical keyframe-based motion editing. In this way, nonprofessional users can create new motion by editing previously stored motion data.

The remainder of this paper is organized as follows. Related works are discussed in Section 2. The proposed framework is described in Section 3. Experimental results are shown and discussed in Section 4 and we conclude the paper in Section 5.

1. INTRODUCTION

Automatic animation systems have been developed to help nonprofessional users create their own animations. Although previous work [1, 2] has discussed various methods to reduce the difficulty of creating animations, these systems cannot deal with a 3D model that is created at runtime, because the initial preparation of the model involves a rigging process that requires skilled users. Therefore, these methods may not be suitable for nonprofessional users, because some complex tasks remaining to be performed.

This paper presents a novel framework for developing an automatic animation system that can deal with a newly created 3D model. The problem is resolved by using an automatic rigging algorithm to create an animation skeleton for the given 3D model. As distinct from previous systems, such as [3, 4, 5], which still require user intervention, our automatic rigging algorithm can create an animation skeleton automatically, with the name of each joint corresponding to a joint name in the motion data. Therefore, motion data can be transferred automatically to the created skeleton using the correspondence between the animation skeleton and the motion data, and the animation can be generated without any user intervention.

2. RELATED WORK

2.1. Automatic Animation System

Most previous work involves storing 3D models and motions separately. A computer animation is created by transferring motion data to the previously rigged 3D model [1]. This strategy requires a relationship between motion data and the 3D model to ensure that the motion of each joint in the motion data is transferred to the correct joint of the 3D model. Some researchers use real performance data to generate the animation [2]. Motion data for each joint is acquired via a motion capture system at runtime, and then used to create the motion of the corresponding joint of the previously rigged 3D model. The implication is that, if the 3D model is created at runtime, it cannot be directly applied to these previous automatic animation systems because it has to be rigged manually. Although users may not need to create the animation when creating the 3D model, they will still need skilled users to complete the rigging process. Therefore, it may be too difficult for nonprofessional users to add new 3D models to an automatic animation system by themselves.

2.2. Automatic Rigging

This technique converts a curve skeleton extracted from a given 3D model to an animation skeleton. Typically, only
local features, such as the bending angle [3], are used to determine the position of each joint. Although this strategy can correctly create an animation skeleton, it contains information about the joint positions only, with the relationship between motion data and the created animation skeleton having to be set manually. Therefore, users themselves need to identify the joints to prepare the model for the automatic animation system. Template-based approaches [4, 5] have been developed to solve this problem. Here, a template skeleton is located on the extracted curve skeleton to create an animation skeleton, with the meaning of each joint then being identified automatically from the template. However, these systems are not fully automatic, and are not robust against differences of orientation between the 3D model and the template skeleton. We solve this problem by creating a set of rules based on the general characteristics of real animals that enable analysis of the extracted curve skeleton. These rules can correctly identify a suitable template, and then extract the meaning of each skeleton segment without requiring any user intervention. Furthermore, all features used by our rules are invariant against orientation. Therefore, a newly created 3D model can be rigged automatically, and the relationship between the 3D model and the motion data is specified without any user intervention.

2.3. Motion Editing

Motion capture is one of the methods used to create new motion, but several complex postprocessing tasks can be too difficult for novice users. Some previous work parameterizes the style of the motion data in a mathematic model [6]. This strategy requires a large set of motion data. To create new motion, users need to specify parameter values, possibly several times, to achieve an optimal parameter value. Direct editing may be suitable for creating new motion because users can directly create new motion from existing primitive motions. However, direct motion editing requires users to perform the same procedure repeatedly for several keyframes. In [7], an editing technique called “editing by propagation” was discussed. This enables users to edit the motion for one interval, and then propagate the editing values to every similar interval of that motion. This can reduce the difficulty and editing completion time of typical keyframe-based editing techniques. However, this research discussed only temporal editing. We extend the idea of editing-by-propagation to geometry editing, and develop other motion editing functions that will help novice users to create new motion.

3. THE PROPOSED FRAMEWORK

We propose a novel framework for developing an automatic animation system, as shown in Figure 1. It consists of three main modules: an animation module, a modeling model, and a motion module. In this section, the details of each module and all required algorithms are described.
This module receives a newly created 3D model from the application. A 3D scanning system may be used to help the user create the new 3D model. Because the newly created 3D model needs to be rigged, and the name of each joint has to be set according to the set of joint names in the motion data, automatic rigging is required to enable nonprofessional users to use the system without help from skilled users. However, existing systems for automatic rigging have several limitations, as discussed in Section 2. Therefore, those algorithms cannot be applied to the proposed framework, which requires this process to be completed automatically.

The requirement of an automatic rigging algorithm for the proposed framework is that it must be able to locate all joints of the animation skeleton at their correct positions, and be able to correctly specify the name of each joint from the set of joint names in the motion data. Furthermore, these processes must be performed automatically. To achieve this requirement, we have developed an automatic rigging algorithm that automatically creates an animation skeleton from the extracted curve skeleton of a given 3D model. This algorithm analyzes several features of the junction points on the symmetry axis of the extracted curve skeleton to identify a character class, which is then used to retrieve a suitable template skeleton. This algorithm can also extract anatomical meanings for each skeleton segment (such as recognizing left-arm or right-leg segments). In this way, each joint on each segment of the template skeleton can be automatically located on the corresponding segment of the animation skeleton, and the name of each joint can be specified in terms of the corresponding joint of the template skeleton.

The first step is extracting a curve skeleton from the 3D model. Any skeletonization algorithm can be used to extract the curve skeleton, if it can generate a clean curve skeleton. Furthermore, because the shortest distance from each position along each skeleton segment to the 3D model’s boundary is required for the automatic rigging algorithm, a preferred skeletonization algorithm should have the potential to supply that information. In our implementation, a skeleton-growing algorithm [10] is used to extract a curve skeleton from a pseudonormal vector field [11] created inside the 3D object. This algorithm generates a clean curve skeleton without any user intervention. Furthermore, because this method propagates a set of normal vectors at the surface to the center of the 3D model, shortest-distance data is also available.

After the curve skeleton is extracted, it can be analyzed to generate an animation skeleton. Because the junction points on the symmetry axis are the main feature points used by the algorithm, we first merge junction points of the curve skeleton that are sufficiently close together to ensure that a pair of symmetry segments, such as the left and right leg, connects to the symmetry axis at the same position, as shown in Figure 2(a). A symmetry axis is then extracted from the curve skeleton. We start from the junction point that is nearest to the centroid of the bounding box of the given 3D model. We then decompose all pairs of symmetry segments from that junction point, with the remaining segments being part of the symmetry axis. We determine a pair of segments as being symmetry segments if they have the same length and the same number of child nodes. If a junction point is found on the extracted part of the symmetry axis, a similar process is performed until there are no newly found junction points on the symmetry axis.

After the symmetry axis is decomposed, the curve skeleton is roughly classified by the number of symmetry junctions. For example, if there is one symmetry junction in the skeleton graph, the given 3D model could be a bird (with a wing junction), a bird (with a leg junction), or a fish (with a tail junction). We then use several features of symmetry junctions that are related to the characteristics of each kind of real animal, such as the position of the symmetry junction, the length of symmetry segments, the angle between symmetry segments, and the shortest distance to the boundary, to achieve a more precise classification. For example, in the case of one symmetry junction, if the junction is located at the end of symmetry axis, it is classified as a fish. Otherwise, we can distinguish between the two bird cases by determining
the angle between symmetry segments, because an angle between bird’s wing should be larger than 90° whereas a bird’s leg will be smaller. These features are also used to extract an anatomical meaning for each skeleton segment. The selected features are invariant against pose and orientation, so this algorithm can deal with the 3D model created by any system.

After the anatomical meaning of each segment has been extracted, we can retrieve a suitable template skeleton. We scale the length of each segment in the template skeleton to match the length of its corresponding segment in the curve skeleton. Each joint can be located in terms of its proportional distance along each skeleton segment. An example of the resulting skeleton is shown in Figure 2(b). To complete the rigging process, the Heat Equilibrium algorithm [4] is used to attach each vertex to the animation skeleton. The resulting 3D model is sent to the animation module to generate an animation and stored in the database. Since this algorithm can be completed without user intervention, a newly created 3D model can be rigged and used to generate animations automatically.

3.3. Motion Module

This module enables users to create new motion for animation purposes. As discussed in Section 2, direct editing is a suitable method for generating new motion. We can assume that all basic motions, such as walking and running, have already been stored in the database. Therefore, users may need to change only the style of the motion, such as changing the arm-swing angle of a walking motion. Clearly, if users require a motion that is very different from the basic motions in the motion database, a motion capture system will be the best method of obtaining good-quality motion. Typically, direct editing of motion data can be performed by manipulating the pose of several keyframes in the motion. The resulting motion can then be generated by interpolating between each pair of keyframes. However, this method is not immediately suitable for editing captured motion data, because the set of keyframes has not yet been specified for such motion data. In [7], this problem was solved by applying the algorithm introduced in [12] to extract a set of main poses for the given motion. Those poses are then identified as keyframes, and shown via a 2D timeline-based interface. In [7], editing-by-propagation was also proposed to reduce the difficulty of the editing task, but this method offers only temporal editing, as discussed in Section 2. That is, users can change only temporal information about the motion.

In this paper, we also use a timeline-based interface to develop a motion editing system. The algorithm in [12] is used to extract keyframes from the motion data. Each keyframe is illustrated in 3D space, so it is easy for a user to directly manipulate the pose of each keyframe. In addition to temporal editing, we also apply the idea of editing-by-propagation to pose editing. Figure 3 illustrates this function. The user chooses one keyframe and all similar keyframes are then selected automatically by using the features previously used to extract all keyframes. (This selection can be adjusted by the user if desired.) The user then edits the chosen key pose via the editing window. These editing operations will then be propagated to the selected set of key poses automatically. The user can set the blending interval for each edited keyframe by adjusting interval bars. The overall motion is then obtained by cubic spline interpolation between the two keyframes at the starting point and the endpoint of each interval bar.

Creating cyclic motion from an arbitrary motion is also a difficult task. For example, if a user needs to create a walking-plus-hand-waving motion from a normal walking motion, the user would need to perform similar editing operations (creating several cycles of hand waving) in several keyframes. We propose “editing from record” to simplify this task. The user creates each key pose in the editing window and records that pose. In this example, the user could record three editing operations (lift right arm, swing right, and swing left) in the editing window. The user would then locate these records within specific intervals by adjusting the interval bar. The same record could be used for several different intervals. Finally, the system would interpolate between all editing durations to generate the overall motion.

We also provide a function to transfer geometry information about the source motion to the target motion. For example, walk-plus-punch motions could be created by adding punching-motion data for arm segments to a walking motion. The user would select arm segments for a specific interval in the source motion and select an editing interval in the target motion. The motion data for the source arm segments would then be transferred directly to the target arm segments. If the sizes of the selected parts of the source and target motions differed significantly, a motion retargeting algorithm [9] would be required to generate satisfactory motion.

4. RESULTS AND DISCUSSION

4.1. Results

We developed the proposed framework as a plug-in for Blender software. The framework is explained in this section...
Fig. 4: Joints (red) are located on each segment of the extracted curve skeleton.

Fig. 5: Left: the curve skeleton of each 3D model is converted to an animation skeleton. Right: 3D models are deformed automatically by transferring motion data to the animation skeleton.

We used 100 character models from several 3D-model databases such as the Princeton Shape Benchmark [13] to test the automatic rigging algorithm. Our algorithm can generate correct results for all 3D models that show their symmetry properties. Examples of the skeleton and joints calculated by our algorithm are shown in Figure 2(b) and Figure 4. The created skeleton is converted to an animation skeleton by the Blender software, as shown in Figure 5(left). Each joint of the animation skeleton is specified by the name of the corresponding joint in the selected template skeleton. The Bone Heat function provided by Blender, developed using the Heat Equilibrium algorithm [4], is used to attach the created animation skeleton to the 3D model.

In the animation module, users can select a 3D model and motion from the databases. The selected motion is automatically transferred to the selected 3D model. Currently, inverse kinematics is used to ensure that the feet of the 3D model do not penetrate the floor. The rendering engine in the Blender software is used to render the resulting animation. Figure 5 (right) shows an example of the deformed 3D model created by our proposed system. In our experiment, motion data was collected from several motion databases, such as the CMU Motion Capture database [14].

We evaluated our motion editing system in terms of completion time and user feedback. Seven male and three female computer users participated in the evaluation. They were 25–30 years of age, and were unfamiliar with computer animation and motion editing. They were asked to complete three motion editing tasks. In the first task, users were to manipulate the maximum angle of the right shoulder as it swings forward during walking. The second task was to create a walking-plus-hand-waving motion from normal walking. In the final task, users were to create a walk-plus-punch motion. Users used our timeline interface to edit the motion by using typical keyframe-based editing techniques and our proposed editing techniques. Using typical techniques, average completion times were 221.7 s ($SD = 19.68$), 756.5 s ($SD = 63.85$), and 1652.1 s ($SD = 61.93$) for the first, second, and third tasks, respectively. Using our system, the times were reduced to 62.1 s ($SD = 5.20$), 245.2 s ($SD = 16.70$), and 57.3 s ($SD = 16.55$), respectively. Seven questions from the Questionnaire for User Interaction Satisfaction (QUIS) [15] were used to rate both editing systems. Figure 6 shows the average scores for each question. The Wilcoxon signed-rank test was used to analyze the QUIS and indicated a significant difference between the two methods ($p < .005$).

4.2. Discussion

The experimental results confirm that the proposed framework can be used to develop automatic animation systems. Nonprofessional users can generate the animation easily because they need to provide only a sequence of motions and a list of 3D models, with an animation then being generated automatically. The proposed framework can deal with newly created 3D models because the proposed automatic rigging can generate correct results for both joint positions and joint meanings. Although user intervention is required to edit motion data in the motion module, the proposed system reduces the difficulty and completion time of this editing task. The experimental results confirm that a nonprofessional user can use...
the proposed editing system to edit motion data into a desired form, and then use it to generate a computer animation.

Although the proposed framework can generate correct results for all tested 3D models, it may generate incorrect results if the 3D model does not provide appropriate symmetry properties (a human being with asymmetric arms or legs, for example) because symmetry junctions are the main feature points used by our algorithm. However, stretching or bending each symmetry segment in different ways does not change the result, as shown by human-being models in Figure 4. Moreover, the results may be incorrect if a 3D model is posed like a 3D model from another character class. For example, a 3D model of a human being might be classified as a four-legged animal if it is created in a crawling pose. Another limitation concerns the editing of motion. Users are required to edit directly via the keyframe-based interface, even though some editing information, such as speed, can be parameterized without using a large set of examples. It would be more useful in such cases if the system enabled users to edit motions by editing these parameter values directly.

It is possible to improve this framework to generate better results. To date, this framework can generate animations for 3D characters, but it cannot generate scenes for the animations. A scene module, which places all objects in the scene, could be added to this framework. An additional algorithm would be required in the animation module to ensure that the 3D model moves about the scene without colliding with scene objects. This might be solved by asking the user to specify a path through the simple map interface, or by automatic calculations involving the position of each object in the scene.

5. CONCLUSION

This paper presents a novel framework for developing automatic animation systems that can deal with a newly created 3D model by employing our proposed automatic rigging algorithm. An animation skeleton and its relationship with motion data can be created automatically by the algorithm. Animation can be generated by transferring motion data according to this relationship. Starting with a static 3D model, animation can be generated without any user intervention. Several motion editing techniques are also proposed to reduce the difficulty and completion time of typical keyframe-based editing. The proposed editing system can be used by nonprofessional users to modify basic motions into desired forms. In future work, we would like to improve the motion editing process, and to develop the scene module as discussed in the previous section.

6. REFERENCES