Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2009, Article ID 592812, 9 pages
doi:10.1155/2009/592812

Research Article

Motion Editing for Time-Varying Mesh

Jianfeng Xu,[1] Toshihiko Yamasaki,[2] and Kiyoharu Aizawa[2]

[1] *Department of Electronic Engineering, The University of Tokyo, Engineering Building no. 2, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*
[2] *Department of Information and Communication Engineering, The University of Tokyo, Engineering Building no. 2, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*

Correspondence should be addressed to Kiyoharu Aizawa, aizawa@hal.t.u-tokyo.ac.jp

Received 30 September 2007; Revised 26 January 2008; Accepted 5 March 2008

Recommended by A. Enis Çetin

Recently, time-varying mesh (TVM), which is composed of a sequence of mesh models, has received considerable interest due to its new and attractive functions such as free viewpoint and interactivity. TVM captures the dynamic scene of the real world from multiple synchronized cameras. However, it is expensive and time consuming to generate a TVM sequence. In this paper, an editing system is presented to reuse the original data, which reorganizes the motions to obtain a new sequence based on the user requirements. Hierarchical motion structure is observed and parsed in TVM sequences. Then, the representative motions are chosen into a motion database, where a motion graph is constructed to connect those motions with smooth transitions. After the user selects some desired motions from the motion database, the best paths are searched by a modified Dijkstra algorithm to achieve a new sequence. Our experimental results demonstrate that the edited sequences are natural and smooth.

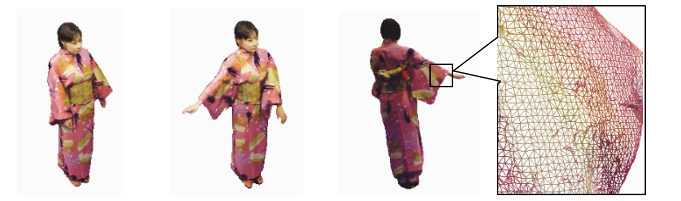
1. Introduction

Over the past decade, a new media called *time—varying mesh (TVM)* in this paper has received considerable interest from many researchers. TVM captures the realistic and dynamic scene of the real world from multiple synchronized video cameras, which includes a human's shape and appearance as well as motion. TVM, which is composed of a sequence of mesh models, can provide new and attractive functions such as free and interactive viewpoints as shown in Figure 1. Potential applications include movies, education, CAD, heritage documentation, broadcasting, surveillance, and gaming.

Many systems to generate TVM sequences have been developed [1–4], which made use of multiple cameras. The main difference between these systems is in their generating algorithms. A recent comparison study is reported by Seitz et al. [5]. Each frame in TVM is a 3D polygon mesh, which includes three types of information: the positions of the vertices, represented by (x, y, z) in a Cartesian coordinate system, the connection information for each triangle that provides topological information of the vertices as shown in Figure 1(d), and the color information attached to each vertex. Two sample frames are given in Figures 1(a)–1(c). Table 1 shows the lengths of our original sequences from four people. The frame rate is 10 frames per second. "Walk" sequence lasts about 10 seconds, "Run" sequence lasts about 10 seconds, and "BroadGym" sequence is broadcast gymnastics exercise, which lasts about 3 minutes.

There are several challenging issues in our TVM data. For instance, each frame is generated independently. Therefore, the topology and number of vertices vary frame by frame, which poses the difficulty in utilizing the temporal correspondence. TVM contains noise, which requires the proposed algorithms to be robust. Another issue is the algorithm efficiency to deal with the huge data. As shown in Table 1, average vertices in a frame are more than 15000.

In conventional 2D video, it is demonstrated that video editing has been widely used. Many technologies have been developed to (semi-)automatically edit the home video such as AVE [6]. In the professional field of film editing, video editing such as montage is surely necessary, which is still implemented mainly by experts using some commercial softwares such as Adobe Premiere. Similarly, editing is necessary in TVM sequences because it is very expensive and time consuming to generate a new TVM sequence. By editing, we can



(a) Frame no. 0 Front view (b) Frame no. 30 Front view (c) Frame no. 30 Back view (d) Part in detail from frame no. 30

Figure 1: Sample frames in TVM; (a) a sample frame in the front view, (b) another sample frame in the front view, (c) the same frame as (b) in the back view, (d) part of the frame in (c).

Table 1: The number of frames and average number of vertices (shown in parenthesis) in TVM sequences.

	Person A	Person B	Person C	Person D
Walk	105(16991)	105(15232)	117(16163)	113(16465)
Run	106(17101)	107(14972)	96(16025)	103(16051)
BroadGym	1981(17681)	1954(15233)	1981(16149)	1954(16834)

reuse the original data for different purposes and even realize some effects which cannot be performed by human actors.

In this paper, a complete system for motion editing is proposed based on our previous works [7–10]. The feature vectors proposed in our previous work [7] are adopted, which is based on histograms of vertex coordinates. Histogram—based feature vectors are suitable for the huge and noisy data of TVM. Like video semantic analysis [11], several levels of semantic granularity are observed and parsed in TVM sequences. Then, we can set up the motion database according to the parsed motion structure. Therefore, the user can select the desired motions (called *key motions*) from the motion database. A motion graph is constructed to connect the motions with smooth transitions. Then the best paths are searched between key motions by a modified Dijkstra algorithm in the motion graph to generate a new sequence. Because the editing operation is on the motion level, the user can edit a new sequence easily. Note that the edited sequence is only reorganized from the original motions, namely, no new frame is generated in our algorithm.

The remainder of this paper is organized as follows. First, some related works are introduced in Section 2. Section 3 describes the feature vectors extracted from mesh models. Section 4 presents the process of parsing the motion structure. Then, motion database is set up in Section 5. Section 6 describes the concept and construction method of motion graph followed by Section 7, where the modified Dijkstra algorithm is proposed to search the best paths in motion graph. Our experimental results are reported in Section 8. Finally comes our conclusions and future work in Section 9.

2. Related Works

2.1. Related works. Motion editing of TVM remains an open and challenging problem. Starck et al. proposed an animation control algorithm based on motion graph and a motion-blending algorithm based on spherical matching in geometry image domain [12]. However, only genus-zero surface can be transfered into geometry image, which limits the adoption in TVM.

Many editing systems are reported on 2D video editing. The CMU Informedia system [13] was a fully automatic video-editing system, which created video skims that excerpted portions of video based on text captions and scene segmentation. Hitchcock [14] was a system for home-video editing, where original video was automatically segmented into the suitable clips by analyzing video content and users dragged some key frames to the desired clips. Hua et al. [6] presented a system for home-video editing, where temporal structure was extracted with an *importance* score for each segment. They also considered the beats and tempos in music. Schodl et al. proposed an editing method in [15], where "video texture" was extracted from video and reorganized into the edited video.

Besides 2D video editing systems, motion capture data editing is another related research topic [16–19], where motion graphs are widely applied, proposed independently by Arikan and Forsyth [16], Lee et al. [17], and Kovar et al. [18]. A motion graph for motion capture data is a graph structure to organize the motion capture data for editing. In [16, 17], the node in motion graph is a frame in motion capture data and an edge is the possible connection of two frames. In [18], the edge is the clip of motion and the node is the transition point which connects the clips. A cost function is employed as the weight of the edge to reflect how good the motion transition is. Motion blending is also used to smooth the motion transition in [17, 18]. The edited sequence is composed by the motion graph with some constraints and some search algorithms. Lai et al. proposed a group motion graph by a similar idea to deal with the groups of discrete agents such as flocks [19]. The larger the motion graph is, the better the edited sequence may be, because the variety of motions contained in the motion graph is higher. However, the search algorithm will take longer time in a larger motion graph.

2.2. Originality of our Motion Graph. A directed motion graph in this paper is defined as $G(V, E, W)$, where the node $v_i \in V$ is a motion in the motion database, the edge $e_{i,j} \in E$ is the transition from the node v_i to v_j, and the weight $w_{i,j} \in W$

is the cost to transit from v_i to v_j (detailed in Section 6). A cost function for a path is defined in Section 7. In our system, the user selects some motions, which are called *key motions* in this paper. The best path between two neighboring key motions is searched in the motion graph. Therefore, the edited sequence is obtained after finishing the searches.

Obviously, our motion graph is different from those in motion capture data. In our motion graphs, a node is a motion instead of a frame, which reduces greatly the number of nodes in motion graph. Therefore, we need to parse the motion structure. To reduce the motion redundancy, the best motion is selected into the motion graph in each motion type, which results in the reduction of the size of motion graph. Therefore, only a part of frames in original frames is utilized in our motion graph, which is different from other motion graphs [16–19]. In addition, TVM is represented in mesh model. Unlike motion capture data, mesh model has no kinematic or structural information. Therefore, it is difficult to track and analyze the motion.

3. Feature Vector Extraction

As described in Section 1, TVM has a huge amount of data without explicit corresponding information in the temporal domain, which makes geometric processing (such as model-based analysis and tracking) difficult. On the other hand, a strong correlation exists statistically in the temporal domain, therefore, statistical feature vectors are preferred [7, 20]. We adopt the feature vectors that were proposed in [7], where the feature vectors are the histograms of the vertices in the spherical coordinate system. A brief introduction is as follows.

Among the three types of information available in mesh models, vertex positions are regarded as essential information for shape distribution. Therefore, only vertex positions are used in the feature vector [7]. However, vertex positions are unsuitable for reflecting both translation and rotation in the Cartesian coordinate system. In [7], the authors proposed to transform them to the spherical coordinate system. To find a suitable origin for the whole sequence, the center of vertices of the 3D model in (and only in) the first frame is calculated by averaging the Cartesian coordinates of vertices in the first frame. Then, the Cartesian coordinates of vertices are transformed to the spherical coordinates frame-by-frame by using (1) after shifting to the new origin.

$$\begin{aligned} r_i(t) &= \sqrt{x_i^2(t) + y_i^2(t) + z_i^2(t)}, \\ \theta_i(t) &= \operatorname{sign}\,(y_i(t)) \cdot \arccos\left(\frac{x_i(t)}{\sqrt{x_i^2(t) + y_i^2(t)}}\right), \\ \phi_i(t) &= \arccos\left(\frac{z_i(t)}{r_i(t)}\right), \end{aligned} \tag{1}$$

where $x_i(t)$, $y_i(t)$, and $z_i(t)$ are the Cartesian coordinates with the new origin for the ith vertex of the tth frame. $r_i(t)$, $\theta_i(t)$, and $\phi_i(t)$ are the spherical coordinates for the same vertex. *sign* is a sign function.

A histogram is obtained by splitting the range of the data into equally sized bins. Then, the points from the data set that fall into each bin are counted. The bin sizes for r, θ, and ϕ are three parameters in the feature vectors, which are kept the same for all frames in a sequence. That causes the bin numbers $J(\sigma, t)$ in (3) to be different in different frames. Therefore, the histograms of the spherical coordinates are obtained, the feature vectors for a frame comprise three histograms, for r, θ, and ϕ, respectively.

With the feature vectors, a distance is defined in (2), called a *frame distance* in this paper. The frame distance is the base of our algorithms:

$$d_f(t1, t2) = \sqrt{d_f^2(r, t1, t2) + d_f^2(\theta, t1, t2) + d_f^2(\phi, t1, t2)}, \tag{2}$$

where $t1$, $t2$ are the frame IDs in the sequence, $d_f(t1, t2)$ is the frame distance between the $t1$th and the $t2$th frames, and $d_f(\sigma, t1, t2)$ is the Euclidean distance between the feature vectors, calculated by

$$d_f(\sigma, t1, t2) = \sqrt{\sum_{j=1}^{\max(J(\sigma,t1), J(\sigma,t2))} \left(h_{\sigma,j}^*(t2) - h_{\sigma,j}^*(t1)\right)^2}, \tag{3}$$

where σ denotes r, θ, or ϕ. $d_f(\sigma, t1, t2)$ is the Euclidean distance between histograms in the $t1$th frame and the $t2$th frame with respect to σ. $J(\sigma, t)$ denotes the bin number of histogram in the tth frame for σ. $h_{\sigma,j}^*(t)$ is defined as

$$h_{\sigma,j}^*(t) = \begin{cases} h_{\sigma,j}(t) & j \le J(\sigma, t), \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

where $h_{\sigma,j}(t)$ is the jth bin in the histogram in the tth frame for σ.

4. Hierarchical Motion Structure Parsing

Many human motions are cyclic such as walking and running. There is a basic motion unit which repeats several times in a sequence. If there are more than one motion types in a TVM sequence, a basic motion unit will be transfered to another after several periods such as from walking to running. Therefore, we define a basic motion unit as the term *motion texton*, which means several successive frames in TVM that form one period of the periodic motion. And several repeated motion textons will be called a *motion cluster*. Thus, TVM is composed of some motion clusters, and a motion texton is repeated several times in its motion cluster. This is the motion structure of our TVM sequences as shown in Figure 2.

An intuitive unit to parse the motion structure is a frame. However, motion should include not only the pose of the object but also the velocity and even acceleration of motion. For example, two similar poses may have different motions with inverse orientations. Therefore, we have to consider several successive frames instead of a single frame. As shown in Figure 2, *motion atom* is defined as successive frames in a fixed-length window, which are our unit to parse the motion structure. Another benefit from motion atom is that noise can be alleviated by considering several successive frames. Some abbreviations will be used in this paper: motion atom will be called as atom or MA, motion texton as texton or MT, and motion cluster as cluster or MC. The motion is analyzed in hierarchical fashion from MA to MC. Therefore, an *atom distance* is defined to measure the similarity of two motion atoms as

$$d_A(t1, t2, K) = \sum_{k=-K}^{K} w(k) \cdot d_f(t1 + k, t2 + k), \qquad (5)$$

where $w(k)$ is a coefficient of a window function with length of $(2K + 1)$. $t1$ and $t2$ are the frame IDs of the atom centers, which show the locations of motion atoms with $(2K + 1)$ frames. $d_A(t1, t2, K)$ is the atom distance between the $t1$th and the $t2$th atoms. In our experiment, a 5-tap Hanning window is used with the coefficients of $\{0.25, 0.5, 1.0, 0.5, 0.25\}$ as it is popular in signal processing. The window size should be larger than 3. The longer the window is, the smoother the atom distances are. However, due to the low frame rate (10 fps) in our sequence, five frames are recommended for the window size, which equals 0.5 seconds. From now on, we will simplify $d_A(t1, t2, K)$ as $d_A(t1, t2)$ since K is a fixed window length.

To parse the hierarchical motion structure, we have to detect the boundaries of motion textons and motion clusters. As shown in Figure 2, motion texton and motion cluster are not in the same level. Namely, a motion cluster is composed of a group of similar motion textons. Therefore, the main idea to detect motion textons is that the first motion atoms are similar in two neighboring motion textons that are in the same motion cluster. And the main idea to detect motion clusters is that there should be some motion atoms which are very different from those in the previous motion cluster. Figure 3 shows the procedure of motion structure parsing. From the beginning of a sequence, a motion texton and a motion cluster begin at the same time in the different levels. For each motion atom, we will determine if it is the boundary of a new motion texton or even a new motion cluster. When a new MT or MC begins, some parameters will be updated. If the current MA is similar to the first MA in the current MT, a new MT should begin from the current MA. Therefore, the atom distance $d_A(t, t_{\text{first}})$ between the current MA at t and the first MA at t_{first} in the current MT is calculated. Then, if $d_A(t, t_{\text{first}})$ reaches the local minimum and the difference between the maximum and minimum in the current MT is large enough (since unavoidable noise may cause a local minimum), a new motion texton is defined.

Figure 4 shows the atom distance $d_A(t, t_{\text{first}})$ in "Walk" sequence by Person D, where all the motion textons are in a motion cluster. Periodic change in Figure 4 shows the motion textons repeat. A distance in the following equation is then defined as *texton distance*, which is the atom distance between the first and last atoms in the texton:

$$d_T(T_i) = d_A(t_{\text{last}}, t_{\text{first}}), \qquad (6)$$

where $d_T(T_i)$ is the texton distance for the ith texton, t_{first} is the first atom in the ith texton, and t_{last} is the last atom in the ith texton. Texton distance measures how smooth the texton repeats by itself.

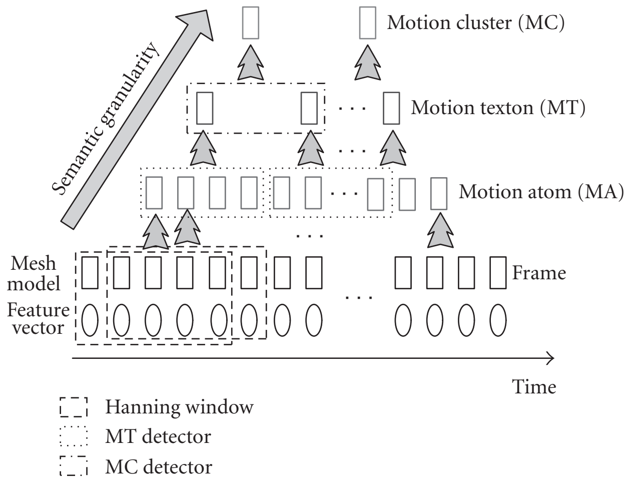


Figure 2: Hierarchical motion structure in TVM.

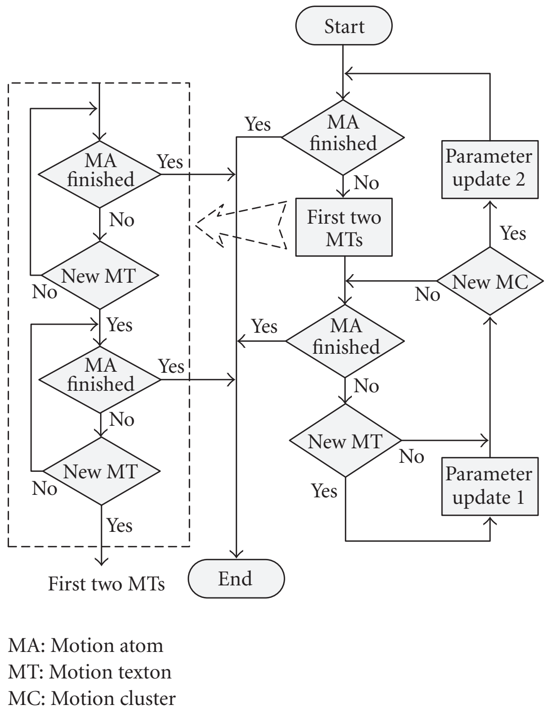


Figure 3: Motion structure parsing procedure in TVM; left: the detail of the *first two MTs*, right: the whole procedure.

On the other hand, if there is no similar MA to the current MAs in the current MC, a new MC should begin from the current MA. Therefore, a minimal atom distance will be calculated as (7), which tries to find the most similar MA in the current MC $[t_{\text{inf}-C}, t_{\text{sup}-T}]$:

$$\begin{aligned} d^{\min}(t) &= d^{\min}(t_{\text{inf}-C}, t_{\text{sup}-T}, t) \\ &= \min_{t_{\text{inf}-C} \le t_k \le t_{\text{sup}-T}} d_A(t, t_k), \end{aligned} \qquad (7)$$

where $t_{\text{inf}-C}$ is the first MA in current MC. $t_{\text{sup}-T}$is the last

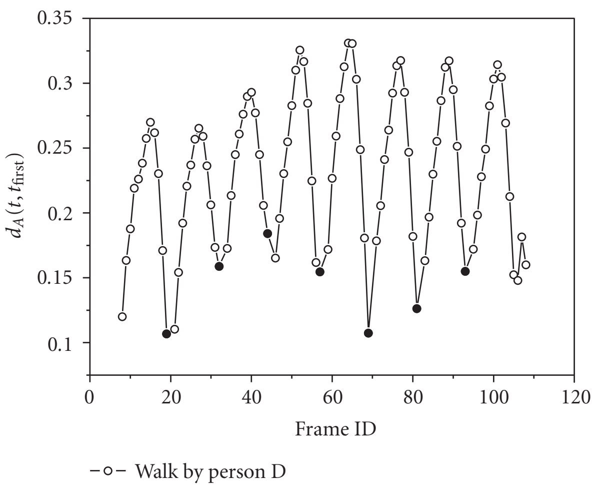


FIGURE 4: Atom distance $d_A(t, t_{\text{first}})$ from the first atom in its motion texton in "Walk" sequence by Person D, the black points denote the first atom in a motion texton.

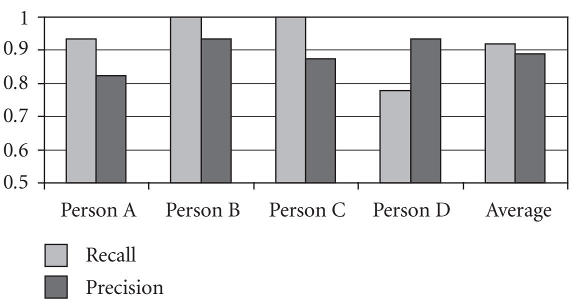


FIGURE 5: Precision and recall for motion cluster detection in "BroadGym" sequences.

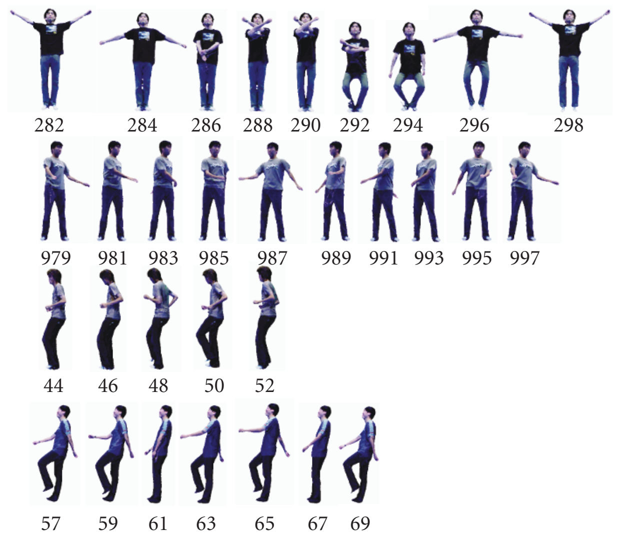


FIGURE 6: Samples of selected motion textons, only every two frames are shown for simplicity.

MA in the previous MT. Then, if two successive motion atoms satisfy (8), a new motion cluster is defined:

$$\begin{aligned} d^{\min}(t-1) &> \beta, \\ d^{\min}(t) &> \beta, \end{aligned} \tag{8}$$

where β is a threshold and set as 0.07 empirically in our experiment. Equation (8) infer that the two motion atoms are different from those in the current MC. We adopt two successive MAs instead of one to avoid the influence of noise. High precision and recall for motion cluster detection are achieved as shown in Figure 5. β surely depends on the motion intensity in two neighboring MCs. It should be set as a smaller value in the sequence with small motions than those with large motions. However, our experiments show that 0.07 can achieve a rather high performance in the most common motions as walking and running.

To initialize $t_{\text{inf}-C}$ and $t_{\text{sup}-T}$, it is assumed that there are at least two motion textons in a motion cluster. Therefore, we detect the boundaries of MC after detecting two motion textons and regard them as the initial reference range of $[t_{\text{inf}-C}, t_{\text{sup}-T}]$ in (7).

5. Motion Database

In Section 4, the hierarchical motion structure is parsed from the original sequences. Since the motion textons are similar in a motion cluster, we only select a representative motion texton into our motion database to reduce the redundant information. The requirement of the selected motion texton is that it is cyclic or it is repeated seamlessly so that the user can repeat such a motion texton many times in the edited sequence. Therefore, we select the motion texton with the minimal texton distance as shown in

$$T_i^{\text{opt}} = \underset{T_i \in C_j}{\arg} \min d_T(T_i), \tag{9}$$

where T_i and C_j are the motion texton and motion cluster. $d_T(T_i)$ is the texton distance for the motion texton, defined in (6). T_i^{opt} is the representative texton, which has minimal texton distance. Figure 6 shows some examples of selected motion textons, where we can see the motion textons are almost cyclic.

6. Motion Graph

To construct a motion graph, we find a possible transition between the motion textons in the motion database. A transition is allowed if the transition between the two motion textons (or two nodes in the motion graph) is smooth enough. A complete motion graph is firstly constructed. Then, some impossible transitions, whose costs are large, are pruned to get the final motion graph. Therefore, a reasonable cost definition is an important issue in motion graph construction, which should be consistent with the smoothness of transition.

Since the node is a motion texton, a transition frame should be chosen in the motion texton. A distance of two textons is defined as the minimal distance of any two frames in the two separate textons as

$$d_V(T_i,T_j) = \min_{t_i\in T_i, t_j\in T_j} d_f(t_i,t_j),$$
$$\{t_i^*, t_j^*\} = \arg_{t_i\in T_i, t_j\in T_j} \min d_f(t_i,t_j), \tag{10}$$

where T_i and T_j are two nodes in the motion graph. t_i and t_j are two frames in the nodes T_i and T_j, respectively. $d_f(t_i,t_j)$ is *frame distance*. $d_V(T_i,T_j)$ is the distance of two nodes, called *node distance*. $\{t_i^*, t_j^*\}$ are the transition frames in the nodes T_i and T_j, respectively, which are calculated by (10).

Another factor that affects the transition smoothness is the motion intensity in the node. By human visual perception, a large discontinuity in transition is acceptable if the motion texton has a large motion intensity, and vice versa. An average frame distance in the node is calculated to reflect the motion intensity of motion texton T_i:

$$\overline{d(T_i)} = \frac{1}{n(T_i)-1}\cdot\sum_{t_i\in T_i \& t_{i+1}\in T_i} d_f(t_i,t_{i+1}), \tag{11}$$

where $n(T_i)$ is the number of frames in node T_i, $d_f(t_i,t_{i+1})$ is the frame distance between two neighboring frames, and $\overline{d(T_i)}$ is the motion intensity of T_i. Thus, the ratio of node distance and motion intensity is defined as the weight of the edge $e(i,j)$ in motion graph:

$$w(T_i,T_j) = \begin{cases} \dfrac{d_V(T_i,T_j)}{\overline{d(T_i)}} & i \neq j, \\ \infty & i = j, \end{cases} \tag{12}$$

where $w(T_i,T_j)$ is the weight of edge $e_{i,j}$ or the cost of transition. Notice that the motion graph is a directed graph: $w(T_i,T_j) \neq w(T_j,T_i)$.

After calculating the weights for all edges, the complete motion graph will be pruned. Considering a node v_i in the complete motion graph, all the edges for the node v_i are classified into two groups, one includes possible transitions and another includes pruned transitions. The average weight of all edges for v_i is adopted as the threshold for the classifier. However, a parameter is given for the user to control the size of motion graph:

$$\overline{w(T_i)} = \frac{1}{N(T_i)-1}\sum_{T_j\in E(T_i)} w(T_i,T_j), \tag{13}$$

where $N(T_i)$ denotes the number of edges which connect with T_i, and $E(T_i)$ denotes the set of edges which connect with T_i. Then, the edge $e_{i,j}$ will be pruned if

$$w(T_i,T_j) \geq \mu\overline{w(T_i)}, \tag{14}$$

where μ is the parameter which controls the size of motion graph.

After pruning the edges, the motion graph is constructed as shown in Figure 7. Note that the IDs of two transition frames are attached to each edge. And the motion graph is constructed in an offline processing.

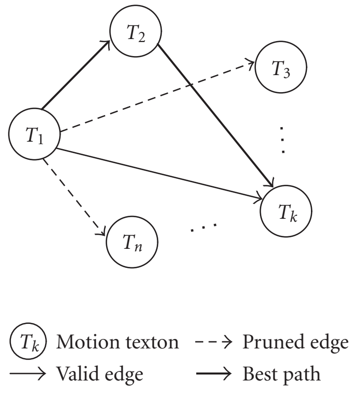


Figure 7: Motion graph concept.

7. Motion Composition

Motions are composed in an interactive way by the desired motion textons. The selected motion textons are similar to the key frames in computer animation and therefore called *key motions*. Between two key motions, there are many paths in the motion graph. A cost function of the path is defined to search the best path. The edited sequence is composed of all the best paths searched in every two neighboring key motions in order.

The perceptional quality of a path should depend on the maximal weight in the path instead of the sum of all weights in the path. For example, the quality of a path will become bad if there is a transition with a very large cost even if other transitions are smooth. Therefore, the cost function is defined as

$$\text{cost}(p(T_m,T_n)) = \max_{e_{i,j}\in p(T_m,T_n)} w(T_i,T_j), \tag{15}$$

where $p(T_m,T_n)$ is a path from the node v_m to v_n. T_m, T_n are two key motions. However, by this definition, more than one path may have the same costs. The best path is required to be shortest, that is, it has the least edges. Then, given the motion graph $G(V, E, W)$ and two key motions T_m and T_n, the problem of the best path can be represented as

$$p(T_m,T_n)^* = \arg_G \min \text{cost}\,(p(T_m,T_n))$$
$$\text{s.t. } p(T_m,T_n) \text{ is shortest.} \tag{16}$$

Dijkstra algorithm can work in the problem of (16) after some modifications. Algorithm 1 lists the algorithm, where the part in italic font is the difference from the standard Dijkstra algorithm. Lines 6, 15, and 17–19 are from the requirement of the shortest path; lines 13 and 14 are from the cost function in (15). The constraint in (16) does not change the cost of the path. Therefore, the only difference from the standard Dijkstra algorithm is our cost function of a path, which uses the maximal weight in the path instead of the sum of the weights. However, because the following property still

```
(1)  function Dijkstra($G$, $w$, $s$)
(2)    for each node $v$ in $V[G]$                       // Initialize
(3)      $d[v]$ := infinity
(4)      previous[$v$] := undefined
(5)    $d[s]$ := 0
(6)    length[$s$] := 0
(7)    $S$ := empty set
(8)    $Q$ := $V[G]$
(9)    while $Q$ is not an empty set                     //Loop
(10)     $u$ := Extract-Min($Q$)
(11)     $S$ := $S$ union $u$
(12)     for each edge $(u, v)$ outgoing from $u$
(13)       if max($d[u]$, $w(u, v)$) < $d[v]$           // Cost function
(14)         $d[v]$ := max($d[u]$, $w(u, v)$)
(15)         length[$v$] := length[$u$] + 1
(16)         previous[$v$] := $u$
(17)       else if max($d[u]$, $w(u, v)$) = $d[v]$      //Shortest path
(18)         if length[$u$] + 1 < length[$v$]
(19)         length[$v$] := length[$u$] + 1
(20)         previous[$v$] := $u$
```

Algorithm 1: Modified Dijkstra algorithm.

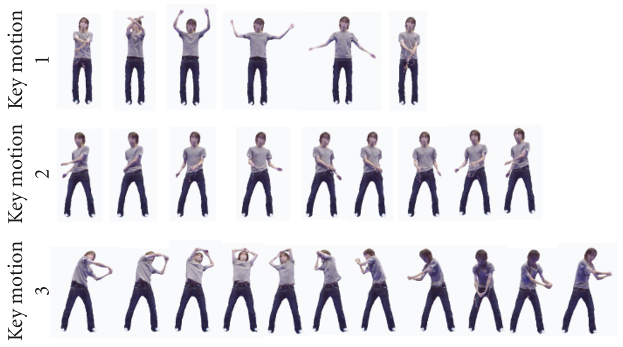


Figure 8: Three key motions in a case study, each row shows a key motion.

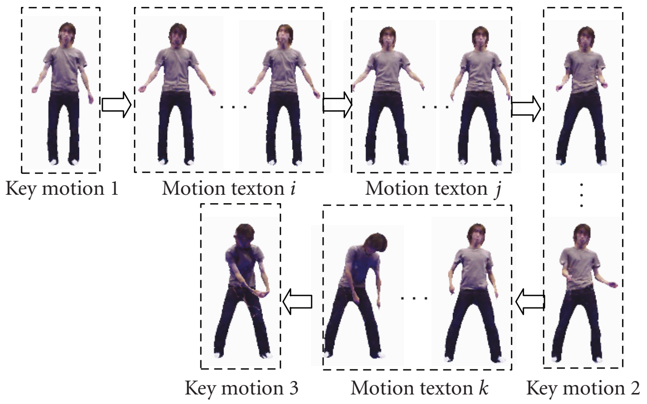


Figure 9: Transitions (denoted by arrows) in two best paths.

holds, we can prove our modified Dijkstra algorithm in the same way as proving the standard Dijkstra algorithm [21]:

$$\text{cost } (p(s, u)) \geq \text{cost } (p(s, x)) \quad \text{if } x \in p(s, u). \tag{17}$$

8. Experimental Results

The original TVM sequences used in the experiments are shown in Table 1. As described above, the user selects the desired motions as key motions. At least, two key motions are required. If more than two motions are selected, the best paths will be searched in every two neighboring key motions. And the ID indices of motion textons in the best paths and their transition frames are calculated to render the edited sequence. The final composite sequence is played using OpenGL.

In our experiments, the parameter μ is set as 0.9. As a case study, Figure 8 shows the three key motions randomly selected by the authors. And our modified Dijkstra algorithm searches two best paths between the three key motions. Figure 9 shows the transitions of the best paths. Our method achieves natural transitions. In the attached video, the whole edited video is played, where the transition is as fast as possible but every frame in the motion texton is rendered at least once before transition (as described in Section 5, the motion textons are cyclic). It is demonstrated that the realistic sequence is achieved.

In our experiments, it is observed that the best path does not exist in some cases because the key motion is unreachable from the previous key motion. The problem can be solved by selecting a new key motion or a larger μ in (12). Although a larger μ means more edges in the motion graph, the path may include some transitions with large weights so that the motion blending is required, which is our future work.

Some extensions are possible in our system. For example, the user can decide some forbidden motions in the edited sequence. For all edges to the forbidden motions, their weights are set as ∞. Therefore, the cost of any path including a forbidden motion will be ∞.

Another issue is how to evaluate the performance of the system, which is rather subjective. However, it is very difficult to design the metric like PSNR in video coding due to the absence of ground truth although it is surely important and meaningful. No report is found in the literature as [12, 16–18], leaving it an open question until now. Generally speaking, it depends on the users and applications: different users have different criteria in different applications. Moreover, the edited sequence also depends on the key motions and motion database. If a key motion has too few edges to connect with, the edited sequence may suffer from a worse quality.

9. Conclusions and Future Work

In this paper, a system for motion editing has been proposed, where the best paths are searched in the motion graph according to the key motions selected by the users. In the original sequences, the hierarchical motion structure is observed and parsed. Then, a motion database is set up with a graph structure. In our motion graph, the node is the motion texton, which is selected from the motion cluster. Therefore, the size of the motion graph is reduced. After the user selects the desired motions, the best paths are searched in the motion graph with a path cost by a modified Dijkstra algorithm.

However, some improvements are possible. In the current system, the length of edited sequence is out of control. In (15), the length error should be considered if necessary. In addition, motion blending at the transitions with large costs will be useful as Kovar et al. [18] did. Motion textons cannot be smoothly transited to others especially when the motion database is relatively small. Also, we believe that the system should take into account the graphical interface design.

Acknowledgments

This work is supported by Ministry of Education, Culture, Sports, Science and Technology, Japan within the research project "Development of fundamental software technologies for digital archives." The generation studio is provided by Japan Broadcasting Corporation (NHK). And the volunteers are greatly appreciated to generate the original sequences.

References

[1] T. Kanade, P. Rander, and P. J. Narayanan, "Virtualized reality: constructing virtual worlds from real scenes," *IEEE Multimedia*, vol. 4, no. 1, pp. 34–47, 1997.

[2] K. Tomiyama, Y. Orihara, M. Katayama, and Y. Iwadate, "Algorithm for dynamic 3D object generation from multi-viewpoint images," in *Three-Dimensional TV, Video, and Display III*, vol. 5599 of *Proceedings of SPIE*, pp. 153–161, Philadelphia, Pa, USA, October 2004.

[3] T. Matsuyama, X. Wu, T. Takai, and T. Wada, "Real-time dynamic 3-D object shape reconstruction and high-fidelity texture mapping for 3-D video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 3, pp. 357–369, 2004.

[4] J. Starck and A. Hilton, "Surface capture for performance-based animation," *IEEE Computer Graphics and Applications*, vol. 27, no. 3, pp. 21–31, 2007.

[5] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, vol. 1, pp. 519–528, New York, NY, USA, June 2006.

[6] X. Hua, L. Lu, and H. J. Zhang, "AVE-automated home video editing," in *Proceedings of the 11th ACM International Conference on Multimedia (MULTIMEDIA '03)*, pp. 490–497, Berkeley, Calif, USA, November 2003.

[7] J. Xu, T. Yamasaki, and K. Aizawa, "Histogram-based temporal segmentation of 3D video using spherical coordinate system," *Transactions of Information Processing Society of Japan*, vol. 47, no. SIG10 (CVIM15), pp. 208–217, 2006, Japanese.

[8] J. Xu, T. Yamasaki, and K. Aizawa, "Motion composition of 3D video," in *Proceeding of the 7th Pacific Rim Conference on Multimedia (PCM '06)*, vol. 4261 of *Lecture Notes in Computer Science*, pp. 385–394, Springer, Hangzhou, China, November 2006.

[9] J. Xu, T. Yamasaki, and K. Aizawa, "Motion structure parsing and motion editing in 3D video," in *Proceedings of the13th International Multimedia Modeling Conference (MMM '07)*, vol. 4351 of *Lecture Notes in Computer Science*, pp. 719–730, Springer, Singapore, January 2007.

[10] T. Yamasaki, J. Xu, and K. Aizawa, "Motion editing for 3D video," in *Proceedings of the Digital Contents Symposium (DCS '07)*, Tokyo, Japan, June 2007, paper # 8-1.

[11] G. Xu, Y.-F. Ma, H.-J. Zhang, and S.-Q. Yang, "An HMM-based framework for video semantic analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 11, pp. 1422–1433, 2005.

[12] J. Starck, G. Miller, and A. Hilton, "Video-based character animation," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '05)*, pp. 49–58, Los Angeles, Calif, USA, July 2005.

[13] M. Christel, M. Smith, C. R. Taylor, and D. B. Winkler, "Evolving video skims into useful multimedia abstractions," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '98)*, pp. 171–178, Los Angeles, Calif, USA, April 1998.

[14] A. Girgensohn, J. Boreczky, P. Chiu, et al., "A semi-automatic approach to home video editing," in *Proceeding of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST '00)*, pp. 81–90, San Diego, Calif, USA, November 2000.

[15] A. Schodl, R. Szeliski, D. H. Salesin, and I. Essa, "Video texture," in *Proceedings of the 27th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pp. 489–498, New Orleans, La, USA, July 2000.

[16] O. Arikan and D. A. Forsyth, "Interactive motion generation from examples," in *Proceedings of the 29th ACM International Conference on Computer Graphics and Interactive Techniques*

(SIGGRAPH '02), pp. 483–490, San Antonio, Tex, USA, July 2002.

[17] J. Lee, J. Chai, and P. S. A. Reitsma, "Interactive control of avatars animated with human motion data," in *Proceedings of the 29th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*, pp. 491–500, San Antonio, Tex, USA, July 2002.

[18] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," in *Proceedings of the 29th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*, vol. 21, pp. 473–482, San Antonio, Tex, USA, July 2002.

[19] Y. C. Lai, S. Chenney, and S. H. Fan, "Group motion graphs," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '05)*, pp. 281–290, Los Angeles, Calif, USA, July 2005.

[20] T. Yamasaki and K. Aizawa, "Motion segmentation and retrieval for 3D video based on modified shape distribution," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, Article ID 59535, 11 pages, 2007.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Mass, USA, 2nd edition, 2001.