Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Motion texture using symmetric property and graphcut algorithm*

SHEN Jian-bing†, JIN Xiao-gang, ZHOU Chuan, ZHAO Han-li
(*State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, China*)
†E-mail: shenjianbing@cad.zju.edu.cn

Received Apr. 10, 2006; revision accepted Apr. 19, 2006

Abstract: In this paper, a novel motion texture approach is presented for synthesizing long character motion (e.g., kungfu) that is similar to the original short input motion. First, a new motion with repeated frames is generated by exploiting the symmetric properties of the frames and reversing the motion sequence playback in a given motion sequence. Then, the order of the above motion sequence is rearranged by putting the start and the end frames together. The graphcut algorithm is used to seamlessly synthesize the transition between the start and the end frames, which is noted as graphcut-based motion-texton. Finally, we utilize the motion-textons to synthesize long motion texture, which can be patched together like the image texture synthesis method using graphcut algorithm, and automatically form a long motion texture endlessly. Our approach is demonstrated by synthesizing the long kungfu motion texture without visual artifacts, together with post-processing including our new developed graphcut-based motion blending and Poisson-based motion smoothing techniques.

Key words: Motion capture, Motion texture, Character animation, Graphcut
doi:10.1631/jzus.2006.A1107 **Document code:** A **CLC number:** TP39

INTRODUCTION

Recent advances in motion capture techniques and other motion editing software facilitate character animation with unprecedented ease and realism. It is still an important and active research area to synthesize the realistic character animation with the motion capture data. Due to the restriction in recorded motion capture data, the direct reuse of long motion capture data is unacceptable, and thus a novel method to edit them is needed. In response, motion textures (Li *et al*., 2002) were introduced.

Motion textures are long motion sequences, often repeatable over time without noticeable artifacts. Similar to image textures, motion textures can be regarded as stochastic process. However, motion textures assume 1D temporal distribution, not like image textures, which display 2D spatial distribution. Motion capture databases such as Carnegie Mellon's (http://www.mocap.cs.cmu.edu) divide the motion clips into different actions. It is still difficult to create a desired long motion texture from the existing short motion capture data.

Several methods for the creation of motion textures have been proposed, but the applicability of motion textures is still restricted to a small range of motion patterns (such as disco-dance) (Li *et al*., 2002). The most important reason is the specific requirement of the original motion capture data to satisfy certain requirements to be acceptable when the input training motion data are limited.

The first algorithm to create motion textures was proposed by Pullen and Bregler (2002) who used a multi-level sampling approach to synthesize new motions that are statistically similar to the original. Similar to multi-resolution representations of video textures (Schödl *et al*., 2000), the term "motion texture" was originally used by Pullen and Bregler (2002) as their project name. They modelled cyclic motions with multi-resolution signals. The main shortcoming of their method was that it is only explicitly modeled as local dynamics of the repetitive patterns.

* Project supported by the National Natural Science Foundation of China (Nos. 60573153 and 60533080), and Program for New Century Excellent Talents in University (No. NCET-05-0519), China

An alternative method was presented by Li *et al*.(2002), who created motion textures using a linear dynamic system (LDS) while the texton distribution is represented by a transition matrix indicating how likely each motion-texton is switched to another. They model not only local dynamics but also global dynamics on how these patterns are linked together. The motion textures are represented by a set of motion-textons and their distribution. When the motion texture is learnt, their method can synthesize sequences of visually compelling dance motion. Our motion texture method is completely different from their methods, inspired by the existing symmetric-based and graphcut-based image and video textures method (Schödl *et al*., 2000; Haevre and Reeth, 2005; Kwatra *et al*., 2003). We explicitly model not only local dynamics of those repetitive patterns, but also global dynamics on how these patterns are linked together. By combining the re-sequencing of existing frames using symmetric property (Phase 1), with the automatic generation of new seamless transition frames using graphcut algorithm (Phase 2), into a two-phase animation pipeline, new long motion textures can be synthesized simply and automatically by patching the above produced motion-textons together.

MAIN IDEA

This paper describes a novel method for creating motion textures, both by exploiting the symmetric properties of short motion capture sequences and synthesizing the frames between the start and the end frames using graphcut algorithm. Fig.1 (see page 1112) shows the framework of our proposed algorithm, which consists of the following three main steps:

(1) Highly symmetric frames are located within the motion capture data by the Mahalanobis distance (Section 3.3), and using derived probabilities from this data to reverse motion playback, which produces a good transition before and after the symmetric frames (Section 3.5);

(2) Using the graphcut algorithm to synthesize the motion capture data between the beginning and end frames, with some post-processing, a fine transition motion is generated from the end to the start frames, which we defined as motion-textons (Section 4).

(3) Patch the above motion-textons to generate an infinite motion texture stream or long motion texture loop without noticeable visual inconsistencies (Section 5).

ANALYZING SYMMETRIC MOTION

As mentioned before, we are interested in finding the specific positions in a motion capture clip from which we can reverse the playback without noticeable visual inconsistencies. With traditional video textures (Schödl *et al*., 2000), a good transition is found when the succeeding frames before and after the transition almost completely match. The use of video textures can thus be extended to motion textures containing no smooth transitions. Inspired by the video symmetric properties described in (Haevre and Reeth, 2005), we propose a method to model the symmetric motion textures using the Mahalanobis distance.

The movement away from a reference position and the returning movement are opposites. In the case of symmetric motions, this is not always true for the complete recorded motion capture data, but most of the time it holds for several frames in the neighborhood of the point where the motion turns. This implies that while playing the motion sequence, a large number of frames can be reused when an extreme position in the captured motion is reached, resulting in a reversed playback of that motion (Fig.1). An important property of a motion texture is the possibility to repeat the generated sequence over time without noticeable, disturbing repetitions of the same movements. This requires the displayed motions to be made of actions that remain natural looking when repeated continuously.

It is very difficult to find exactly the same configuration of the joint structure on different locations in a dancing motion sequence, due to their complex structure, turbulent motions and the incoherence of the movement of the individual joint parts. A typical subset of this category includes the movements of disco-dance. Such movements are often focused on one or more fixed positions from which the joint part returns from time to time.

Whenever a similar position of a joint or set of different joints is detected on different motion-frames,

a motion transition can be made, resulting in a rearrangement of the frames of the original motion sequence. Standard motion textures exploit these reference locations by means of transitions. On the other hand, joints tend to move very symmetrically by means of successive decreasing in symmetric movements.

Distance metrics

The key to locate highly symmetric frame precisely, is the proper use of the distance metrics. We compared two different distance metrics: the L2 distance (Schödl *et al*., 2000; Haevre and Reeth, 2005) and Mahalanobis distance (Hastie *et al*., 2002). We compare the frames which locate over the same distance between and after the test position. A symmetric position is marked when they match sufficiently according to a user-specified threshold.

L2 distance

The first distance metric is the L2 distance between all-pairs of frames. Given two relevant clips $\boldsymbol{M}_i$ and $\boldsymbol{M}_j$:

$$D(i,j)=\exp\left(-\sum_k \| \boldsymbol{M}_i(k)-\boldsymbol{M}_j(k) \|_2 / \sigma\right), \quad (1)$$

where $M_i(k)$ represents the kth joint of the ith frame of the original motion $\boldsymbol{M}$, The parameter σ controls the mapping between L2 distance and relative probability of a given transition (Schödl *et al*., 2000). The L2 distance is simple and works well for large motion datasets with incremental changes between frames, but cannot handle the inherently sparse and small changes between motion sequences.

Mahalanobis distance

The Mahalanobis distance (Hastie *et al*., 2002) is superior to the L2 distance because it takes into account all the correlations between the variables. Considering the covariance, it accounts for ranges of acceptability in the sample data, and thus makes it very suitable for calculating the distance of frame difference. We will use the Mahalanobis distance (Hastie *et al*., 2002) of frame difference for locating the symmetric motion frames later (shown in Fig.2, see page 1112). Assuming the joint variables of the motion capture data of the adjacent frames are uncorrelated, the Manalanobis distance of $\boldsymbol{M}_i$ and $\boldsymbol{M}_j$ is defined as:

$$D(i,j)=(\boldsymbol{M}_i-\boldsymbol{M}_j)^{\mathrm{T}}\boldsymbol{C}^{-1}(\boldsymbol{M}_i-\boldsymbol{M}_j), \quad (2)$$

where $\boldsymbol{C}$=diag(d$\boldsymbol{M}$/dt) is the diagonal covariance matrix computed from the motion capture data with the derivative representation d$\boldsymbol{M}$/dt, which mean the velocity of the joints.

Locating highly symmetric frames

Once the Mahalanobis distance matrix of a motion clip is computed, we analysis the symmetric property to obtain the locations of the motion sequences. In our experiment, we just include a few frames because the symmetric property of a location with a motion sequence can sometimes be very local. Considering the influence of nearby motion frames, similar to Haevre and Reeth (2005) having done with the symmetric video texture, a weighted sum of the Mahalanobis distance is used to find the location of the symmetric frames as follows:

$$sym_i=\sum_{r=1}^{n} C_{2(n-s)}^{n-r}\cdot D_{i-r,i+r}, \quad (3)$$

where n represents the amount of frames of the original motion clip, s defines the smoothness of the binomial coefficients and $D_{i-r,i+r}$ equals to the Mahalanobis distance between the frames located r positions before and after test-frame i.

In order to make the *sym* values more meaningful, it is then mapped onto the unit interval [0..1] as Eq.(3) shows:

$$sym_i=1-sym_i/\max(sym_i). \quad (4)$$

Next, we extract the local maxima probability values to reduce the symmetric candidates, the frames with a high sym_k ($i-r\le k\le i+r$) value were kept as the candidate symmetric positions as follows:

$$P_{\max_i}=\begin{cases} sym_i/2, & \text{if } sym_i=\max(sym_k),\\ 0, & \text{else.}\end{cases} \quad (5)$$

As a result, a sufficient symmetric value indicates that motion playback should reverse at these positions, which can prevent rapid repetitions of the

same short action while producing a sufficient amount of smooth transiting frames between symmetric turning positions.

Symmetric-based motion synthesis

In order to synthesize new sequences from the original motion clip, we just simply use the pre-calculated probabilities to decide whether a reverse is going to be made at a certain frame position. The main problem with this approach is that most of the time the same turns are at the positions with the highest probabilities. However, the same part of the original motion clip will be reused several times while a portion of the available frames are unused (Haevre and Reeth, 2005).

An extended algorithm with a changing behavior of the turning-points is introduced by increasing or decreasing the probability of a point. When a turn is taken, increasing the possibility to continue the next time without a turn, the probability to turn again at this point is lowered a fixed amount (Haevre and Reeth, 2005), When a turn is not taken, the opposite behavior is provoked, by increasing a controlled probability with parameter k as follows:

$$P_i=k\cdot P_{\max_i}. \tag{6}$$

The resulting rearrangement of the motion frames can be done in real-time and results in a visually satisfying motion texture sequence.

GRPAHCUT-BASED MOTION TEXTURE

In image synthesizing, small blocks from the sample texture image are copied onto the output texture image. The first block is copied at random, and then subsequent blocks are placed such that they partly overlap with previously placed blocks of pixels. This process is repeated for each block as they are placed in the larger image. Kwatra *et al*.(2003) used the graph cut algorithm to choose the minimum cost path from one end of this overlap region to the other. That is, the chosen of the optimal path is through those pixels where the old and new patch colors are similar. The path determines which patch contributes pixels at different locations in the overlap region.

We use a standard articulated skeleton as an avatar model. Modelling of a human body attached to the skeleton is beyond the scope of our work. We use unit quaternions instead due to the problems with rotations parameterized by Euler angles.

Applying the graph cut algorithm to motion capture data, we take only the beginning m frames, noted as $\boldsymbol{M}_1=\boldsymbol{M}_i(1{:}m)$, and the end m frames, noted as $\boldsymbol{M}_2=\boldsymbol{M}_i((end-m+1){:}end)$, from the original input motion $\boldsymbol{M}_1=\boldsymbol{M}_i(1{:}end)$, then rearrange the sequence of the original motion to generate a new motion $\boldsymbol{M}_{\text{texture}}=\boldsymbol{M}_i((m+1){:}(end-m))\cup\boldsymbol{M}_2\cup\boldsymbol{M}_1$.

For the above rearranged motion $\boldsymbol{M}_{\text{texture}}$ is not a smooth motion clip, we need to use a graphcut algorithm to synthesize seamless motion between $\boldsymbol{M}_2\cup\boldsymbol{M}_1$. Now we first parameterize the motion $\boldsymbol{M}_{\text{texture}}=\boldsymbol{M}_2\cup\boldsymbol{M}_1$ using the unit quaternions, then search for the best overlap region over n_{overlap} frames. The segmented seam occurs at frame t_1 of $\boldsymbol{M}_1$ and t_2 of $\boldsymbol{M}_2$. The optimal seam (path) between $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ is then computed joint by joint. The whole algorithm can be described as follows:

```
for i=1: n_overlap
  for j=1: n_overlap
    using our new cost function M_new(s,t,M_2,M_1) to
      calculate the distance between M_1(i) and M_2(i)
    return i and j of the minimum distance
  end
end
```

Kwatra *et al*.(2003) defined the SSD-based patch matching quality cost function between two adjacent pixels for image and video texture synthesis. Similarly, we solve this problem by joint sampling and joint matching algorithm. However, we do not use the common Sum of Squared Difference (SSD), which is widely used in image and video texture to measure the similarity between space patches. The reason is that the SSD does not always suffice to provide the desired texture results as described in (Wexler *et al*., 2004).

Since a well-suited similarity measurement between joint patches is the heart of the graphcut texture synthesis algorithm that directly influences the final motion texture synthesis result, we define a more accurate joint cost function as an exponential similarity measure as follows:

$$M_{\text{new}}(s,t,\boldsymbol{M}_2,\boldsymbol{M}_1)=\exp(-M(s,t,\boldsymbol{M}_2,\boldsymbol{M}_1)/\nabla G^d(s,t,\boldsymbol{M}_2,\boldsymbol{M}_1), \tag{7}$$

$$M(s,t,\boldsymbol{M}_2,\boldsymbol{M}_1)=\|\boldsymbol{M}_2(s)-\boldsymbol{M}_1(s)\|+\|\boldsymbol{M}_2(t)-\boldsymbol{M}_1(t)\|, \quad (8)$$

$$\nabla G^d(s,t,\boldsymbol{M}_2,\boldsymbol{M}_1)=\|G^d_{\boldsymbol{M}_1}(s)\|+\|G^d_{\boldsymbol{M}_1}(t)\|+\|G^d_{\boldsymbol{M}_2}(s)\|+\|G^d_{\boldsymbol{M}_2}(t)\|, \quad (9)$$

here, d indicates both the direction of the joint-gradient and the direction of the changes between the joint DOFs (degree of freedoms) between joint s and joint t. $G^d_{\boldsymbol{M}_1}(s)$ and $G^d_{\boldsymbol{M}_2}(s)$ are the joint gradients in the motion clips $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ along the direction d. $M_{\text{new}}(s,t,\boldsymbol{M}_2,\boldsymbol{M}_1)$ penalizes seams going through high frequency motion regions.

Each degree of freedom over the n_{overlap} frames could transit between $\boldsymbol{M}_2$ and $\boldsymbol{M}_1$ at different frames for a smoother transition. In our experiment, we set m=100 and n_{overlap}=50, in order to accelerate the speed for calculating the joint cost function $M_{\text{new}}(s,t,\boldsymbol{M}_2,\boldsymbol{M}_1)$. Similar to (Kwatra *et al*., 2003), an FFT-based method can be used. A frame of the motion is viewed as a column of the joints-graph as Fig.3a (see page 1112) shows, and then directly applied to the above described graphcut algorithm to get the synthesized motion texture as Fig.3b shows.

For synthesizing a long motion texture, we just need to patch the above motion-textons ($\boldsymbol{M}_{\text{texton}}=\boldsymbol{M}_{\text{new}}$) together along the time axis like image synthesis method (Kwatra *et al*., 2003), where k is the *patch* numbers for motion-textons. The final long seamless motion texture is represented as follows:

$$\boldsymbol{M}_{\text{texture}}=patch(\boldsymbol{M}_{\text{texton}}, k). \quad (10)$$

POST-PROCESSING

Our method produces a high quality motion texture when a good input motion is given, but may violate kinematics constraints imposed by the environment. Usually, the most visible artifact is footstake or foot-sliding, which is described and corrected by using the existing methods if the footplants are annotated (Lucas *et al*., 2002a; 2002b), as the simple graphcut synthesis may produce some inconsistent transitions between two adjacent clips $\boldsymbol{M}_2$ and $\boldsymbol{M}_1$. In this section, we describe an efficient graphcut-based blending algorithm for cleaning up the footstake phenomenon.

Graphcut-based motion blending

If we get the graphcut seam locations set s, we create a transition by blending the frames $\boldsymbol{M}_1$ to $\boldsymbol{M}_1^{s+k-1}$ with frames $\boldsymbol{M}_2^{s+k-1}$ to $\boldsymbol{M}_1^{s}$ along the direction of the graphcut seam, inclusively. This is similar to the approach in (Lucas *et al*., 2002a; 2002b), but we use the graphcut to determine the transition points while they use the minimal weighted SSD-based error function to detect the candidate transition points. Then the root positions in frame p of the transition ($0\le p<k$) is linearly interpolated and the joint rotations are performed with spherical linear interpolation as follows:

$$M^{\text{R}}_p=\alpha(p)M^{\text{R}}_{\boldsymbol{M}_1^{s+p}}+(1-\alpha(p))M^{\text{R}}_{\boldsymbol{M}_2^{s-k+1+p}}, \quad (11)$$

$$\alpha(p)=2(p+1/k)^3-3(p+1/k)+1, \quad (12)$$

$$q^i_p=slerp(q^i_{\boldsymbol{M}_1^{s+p}},q^i_{\boldsymbol{M}_2^{s-k+1+p}},\alpha(p)), \quad (13)$$

where M^{R}_p is the root position on the pth transition frame, q^i_p is C^1 continuous, and $\alpha(p)$ is a blending function according to the conditions that $\alpha(p)=0$ for $p\ge k$, $\alpha(p)=1$, for $p\le -1$, and q^i_p is the rotation of the ith joint on the pth transition frame (Lucas *et al*., 2002a; 2002b). In order to get a smoother motion texture, we finally use the Poisson equation for smoothing the motion.

Motion smoothing using Poisson equation

Image reconstruction from gradients fields is an approximate invariability problem, and it can be extended for smoothing the motion capture data when the frame of the motion is viewed as a column of graph (or a matrix) in 2D. In 2D, a modified joint-gradient vector field $\boldsymbol{G}'=[Gx',Gy']$ may not be integrable. Let $\boldsymbol{M}_{\text{new}'}$ denote the smoothed reconstructed motion propagated from $\boldsymbol{G}'$, we can use one of the direct methods recently proposed (Fattal *et al*., 2002) to minimize $|\nabla\boldsymbol{M}_{\text{new}'}-\boldsymbol{G}|$, so that $\boldsymbol{G}=\nabla^2\boldsymbol{M}_{\text{new}'}$. Involving a Laplacian and a divergence operator, $\boldsymbol{M}_{\text{new}'}$ can be obtained by solving the Poisson differential equation:

$$\nabla^2\boldsymbol{M}_{\text{new}'}=\text{div}([Gx',Gy']). \quad (14)$$

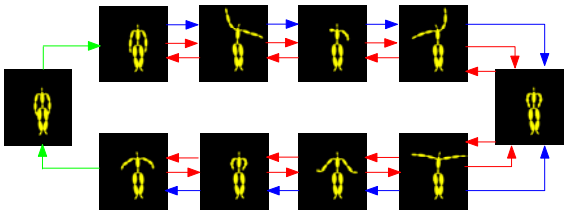

Fig.1 The framework of reversing motion playback at a symmetric frame position, while the start and the end frames are synthesized by graphcut: (blue) original motion capture sequences; (red) reusing the symmetric frames; (green) graphcut synthesis frames

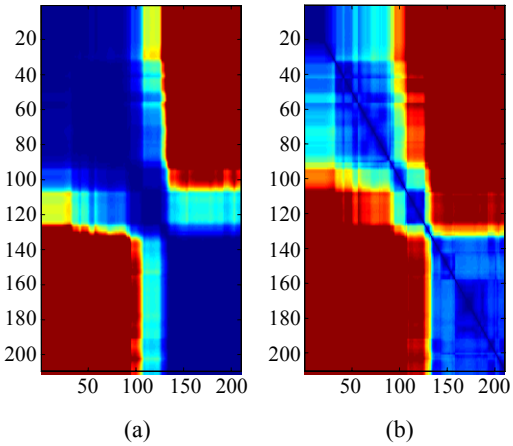

Fig.2 L2 distance metric (a) and Mahalanobis distance metric (b) for the dance motion capture data

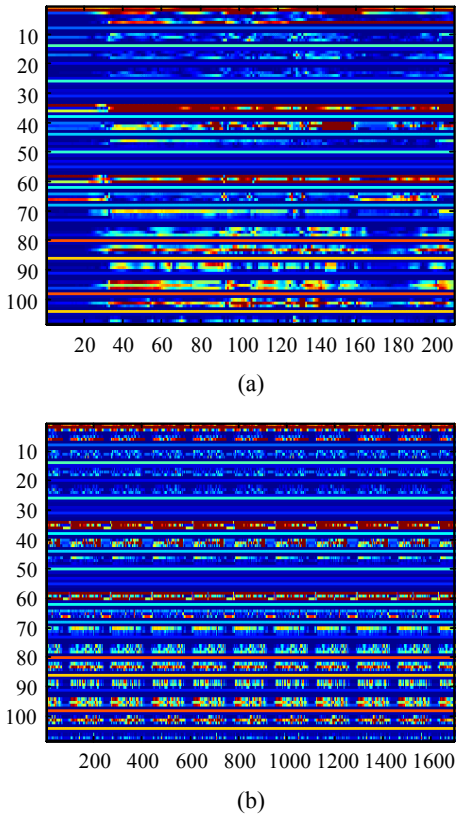

Fig.3 Illustration of graphcut-based motion texture. A frame of the motion is represented as a column of the joints-graph. (a) The original dance motion (210 frames); (b) The textured motion using graphcut (with Poisson smoothing), total 1700 frames

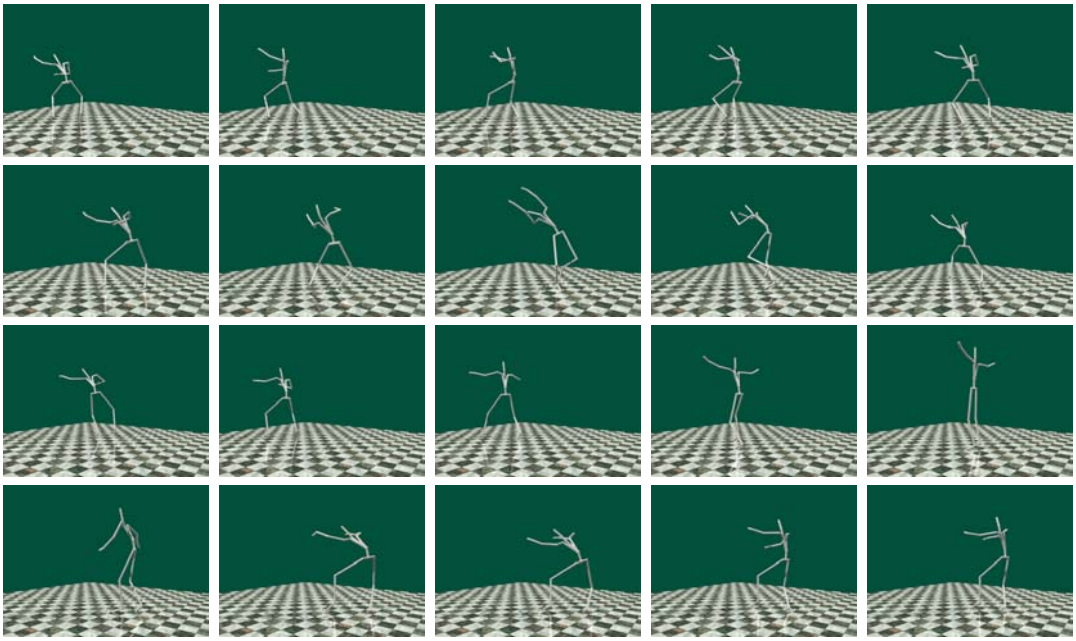

Fig.4 Graphcut-based kungfu motion texture

Since both the Laplacian ∇^2 and div are linear operators, approximating them using standard finite differences yields a large system of linear equations. We use the full multigrid method (Agrawal *et al*., 2005) to solve the Laplacian equation with Gaussian-Seidel smoothing iterations. This leads to $O(n)$ operations to reach an approximate solution, where n is the number of DOFs in the motion.

To solve the Poisson equation more efficiently, an alternative is to use a "rapid Poisson solver", which uses a 'sine' transform based on the method in (Agrawal *et al*., 2005) to invert the Laplacian operator. However, the complexity with this approach will be $O(n(\log(n)))$. The graphcut parts of the motion was zero-padded on both sides, and Dirichlet boundary conditions instead of Neumann boundary conditions were used to avoid the scale/shift ambiguity (Agrawal *et al*., 2005) in the gradient reconstruction.

EXPERIMENTAL RESULTS

In all our evaluations, the motion capture data is acquired from Carnegie Mellon's motion capture database and other public motion database from Web site, preprocessed with standard commercial tools. Our motion capture dataset contained various typical human motions (such as running, walking, dance, kungfu).

Fig.4 (see page 1112) demonstrates an application of our method to modify and texture the length of a kungfu motion clip. This is a rather challenging test case, since the input sequence contains only 209 kungfu frames, almost none of which is very similar to another because of the kicking and turning nature of the kungfu. Therefore, we believe it would be difficult to construct a good high-level motion model (Li *et al*., 2002) from such a training motion dataset. Our method, however, is still able to generate visually continuous and naturally looking motions by locally smoothing over discontinuous transitions using our post-processing methods.

Although our approach has been effective on generating realistic motion texture, there remain several areas for improvement to synthesize the motion texture seamlessly. Currently our method does not ensure various desirable properties, such as that the feet of a character do not penetrate the ground while walking. Another limitation of our method is that interacting with environment objects is not taken into consideration.

Our method is best suited for motions consisting of frequently repeated patterns such as disco-dance, kungfu. The synthesized motion textures may lack global variations when the original data is limited. For complex motion textures with fewer repetitions it is difficult for our method to synthesize natural-looking transitions without post-processing for footskate cleanup.

CONCLUSION AND FUTURE WORK

In this paper, we have developed a new novel algorithm for generating long motion texture from existing captured motion sequences, named motion texture using symmetric property and graphcut algorithm. In order to produce the seamless motion textures, we use the graphcut algorithm to find the synthesis points to concatenate the rearranged frames. Our technique is not intended as a replacement for previously developed methods for motion texture (Li *et al*., 2002; Moradoff and Lischinski, 2004), rather it is meant to complement them, adding a new useful component into the animator's motion texture synthesis.

Although our approach has been effective on generating realistic and dynamic motion texture, there remain several areas for further improvement. In future work, we plan to consider sophisticated types of constraints, such as the soft constrains described by Arikan and Forsyth (2002). We also plan to extend our method to mix elements from several different style input motion sequences, perhaps in a manner similar to the style translation (Hsu *et al*., 2005), together combining our graphcut motion texture algorithm.

References

Agrawal, A., Raskar, R., Nayar, S., Li, Y., 2005. Removing flash artifacts using gradient analysis. *ACM Transactions on Graphics*, **24**(3):828-835. [doi:10.1145/1073204.1073269]

Arikan, O., Forsyth, D.A., 2002. Interactive motion generation from examples. *ACM Transactions on Graphics*, **21**(3):483-490. [doi:10.1145/566654.566606]

Chun, J.S., Kim, M.K., Jung, H.K., 1997. Shape optimization

of electromagnetic devices using immune algorithm. *IEEE Transactions on Magnetics*, **33**(2):1876-1879. [doi:10.1109/20.582650]

Fattal, R., Lischinski, D., Werman, M., 2002. Gradient domain high dynamic range compression. *ACM Transactions on Graphics*, **21**(3):249-256. [doi:10.1145/566654.566573]

Haevre, W.V., Reeth, F.V., 2005. Video Textures Exploiting Symmetric Movements. EUROGRAPHICS Short Presentations, p.489-498.

Hastie, T., Tibshirani, R., Friedman, J., 2002. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, New York, Berlin, Paris.

Hsu, E., Pulli, K., Popovic, J., 2005. Style translation for human motion. *ACM Transactions on Graphics*, **24**(3): 1082-1089. [doi:10.1145/1073204.1073315]

Kwatra, V., Schödl, A., Essa, I.A., Turk, G., Bobick, A.F., 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics*, **22**(3): 277-286. [doi:10.1145/882262.882264]

Li, Y., Wang, T., Shum, H.Y., 2002. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, **21**(3):465-472. [doi:10.1145/566654.566604]

Lucas, K., Schreiner, J., Gleicher, M., 2002a. Footskate cleanup for motion capture editing. *Proceedings of ACM SIGGRAPH'02 Symposium on Computer Animation*, **21**(3):97-104.

Lucas, K., Gleicher, M., Pighin, F., 2002b. Motion graphs. *ACM Transactions on Graphics*, **21**(3):473-482.

Moradoff, S., Lischinski, D., 2004. Constrained synthesis of textural motion for articulated characters. *The Visual Computer*, **20**(4):253-265. [doi:10.1007/s00371-003-0231-1]

Pullen, K., Bregler, C., 2002. Motion capture assisted animation: texturing and synthesis. *ACM Transactions on Graphics*, **21**(3):501-508. [doi:10.1145/566654.566608]

Schödl, A., Szeliski, R., Salesin, D., Essa, I.A., 2000. Video textures. *ACM Transactions on Graphics*, **19**(3):489-498.

Wexler, Y., Shechtman, E., Irani, M., 2004. Space-time Video Completion. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition'04, p.120-127.