An Evolution Computation Based Approach to Synthesize Video Texture

Yu Meng, Wen-hui Li, Yan Wang, Wu Guo, and Wei Pang

College of Computer Science and Technology, Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Jilin University, Changchun, 130012, P.R.China
myu_jlu@126.com

Abstract. Texture synthesis is one of the hottest areas in computer graphics, computer vision and image processing fields, and video texture synthesis is one subset of it. We bring forward a new method on video texture synthesis, in which evolution computing technique is introduced into the processes of synthesizing videos. In the method, by analyzing and processing a finite source video clip, Infinite video sequences obtained can be played smoothly in vision. Comparing with many existing video texture synthesis algorithms, this method can not only get high-quality video results without complicated pre-processing of source video, but also improve the efficiency of synthesis.

1 Introduction

Texture synthesis is of great importance in computer vision and graphics. Many methods on texture synthesis were introduced in the last decade. Heeger & Bergen [1] use color histograms across frequency bands as a texture description. Bonet[2], who first brought non-parameter texture synthesis method, uses a multiresolution filter-based approach to generate textures. Efors & Leung[3] are first to copy pixels directly from input textures to generate new textures. Portilla & Simoncelli[4] use a synthesis procedure by decomposing texture images to complex wavelets and synthesize new images by matching the joint statistics of these wavelets. Wei & Levoy[5] have improved Efros & Leung's method by using a multiresolution image pyramid based on a hierarchical statistical method. Liang [6] and Efors & Freeman[7] copy whole patches from input textures to synthesize new textures.

The work described above mainly deals with static scenes. Recently has been appeared many video texture synthesis methods. Arno Schödl creates long videos by rearranging original frames from a short source video[8], and extends video textures to allow for control moving objects in video[9]. Wang&Zhu[10] models the motion of texture particles in source video to synthesize new videos. Doretto[11] uses Auto-Regressive filters to model and edit the complex motion of fluids in video. Kwatra[12] combines volumes of pixels along minimum error seams to create new sequences that are longer than the original video. Bhat[13] analyzes the motion of textured particles in the input video along user-specified flow lines, and synthesizes seamless video of arbitrary length by enforcing temporal continuity long a second set of user-specified flow lines.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, pp. 223 – 230, 2006.

2 Evolution Computation-based Video Texture Synthesis

We suggest a video texture synthesis method using Genetic Algorithm[14], which is inspired by Schödl's algorithm[8]. Compared with other algorithm, this method uses an appropriate fitness function in the processes of creating video sequences instead of much complex pre-processing of source video clip. Additionally, constraints of selection operator, crossover operator and mutation operator ensure that this algorithm can get high-quality video results quickly.

2.1 Video Textures Algorithm

The original idea of Video Texture Algorithm supposes that there are two similar frames in a given video clip: frame *i* and frame *j*. Naturally, the switch from frame *j* to frame *j*+1 is very smooth, we can gain the unaffected video from switching frame *i* to frame *j*+1. Here *i* is called "transition" point. When the video is played frame *i*, the next frame can be frame *i*+1 or frame *j*+1. If there are several such transition points in the source clip, videos with arbitrary length can be synthesized from it by switching at these points continuously.
The original procedure of Video Textures Algorithm is described as follows:

1. Calculate the similarities between all pairs of frames in the input sequence *S*. Calculate frame-to-frame distances and stored them in the similarity matrix of *D* with the following formula:

$$D_{ij}=\sum_{k=1}^{m}\sum_{l=1}^{n} sqrt\left\{\left[R\left(P_{k,l}^{i}\right)-R\left(P_{k,l}^{j}\right)\right]^2+\left[G\left(P_{k,l}^{i}\right)-G\left(P_{k,l}^{j}\right)\right]^2+\left[B\left(P_{k,l}^{i}\right)-B\left(P_{k,l}^{j}\right)\right]^2\right\} \quad (1)$$
$$0<i,j\le len;\quad 0<k\le m;\quad 0<l\le n;$$

   Where $P_{k,l}^{i}$ denotes the pixel of coordinates (*k*,*l*) in the *i*-th frame with size m×n in video sequence *S*. *R()*, *G()*, *B()* represent the RGB values of a pixel. And *len* is the frame number of *S*;
2. Calculate the transition cost matrix between two frames. Transition cost is a sum of image differences around a transition. Transition cost from frame F_i to frame F_j can be calculated as below and stored in a transition cost matrix of $D_{ij}^{'}$:

$$D_{ij}^{'}=\sum_{k=-m}^{m-1} w_k D_{i+k+1,j+k};\quad 0<i,j\le len; \quad (2)$$

   Where m=1 or 2, corresponding to a sum over 2 or 4 image differences with weights w_k (1 1) or (1 2 2 1), and *len* is the frame number of *S*;
3. Avoid dead-ends and prune part of transitions. The former operation aim at preventing the video from being lead to regions without transitions, and the latter operation can save on storage space and improve the quality of result videos.

4. Calculate future cost for each transition using the following formula, that reflects the expected average cost of future transitions and store it in future cost matrix of $D_{ij}^{"}$.

$$D_{ij}^{"} = (D_{ij}^{'})^{p} + \alpha \min_{k} D_{jk}^{"}; \quad 0 < i, j, k \le len; \tag{3}$$

   Here, p is a constant controlling the tradeoff between multiple good (low-cost) transitions and a single poorer one. The constant α is used to control the relative weight of future transitions in the metric. In order to reach convergence, it should be limited to (0,1). *len* is the frame number of *S*;
5. Create video loops, called compound loops, which can be combined to create additional cyclic sequences. The final video sequences are composed of several compound loops.

Applying Video Textures Algorithm to process not only random textures but also structured textures can obtain high-quality result video. However, the pre-processing to source clip before creating video sequences is too much and complex. Furthermore, the Dynamic Programming Algorithm(DPA) used to create video sequences only allows backward transition so that its synthesizing results have less diversity. We find that creating of video sequences can be considered as a combinatorial optimum problem and solved by Genetic Algorithm. Using Genetic Algorithm to synthesize videos only need an appropriate fitness function to preserve dynamics and continuity instead of much complex pre-processing. In addition, the GA-based algorithm allows not only backward transition but also forward transition so that its synthesizing results have better diversity.

2.2 Genetic Algorithm (GA)

GA is based on the biological phenomenon of genetic evolution. The GA maintains a set of solutions which is known as a population or a generation. GA operates in an iterative manner and evolves a new generation from the current generation by application of genetic operators. The process of basic Genetic Algorithm is as below:

1. Generate random population of n chromosomes (suitable solutions for the problem)
2. Evaluate the fitness f(x) of each chromosome x in the population
3. Create a new population by repeating following steps until the new population is complete
   - 3.1 Select two parent chromosomes from a population according to their fitness. For each chromosome, the better fitness it has, the bigger chance it would be selected;
   - 3.2 Cross over the parents to form new offspring with a crossover probability. If no crossover was performed, offspring is the exact copy of parents.
   - 3.3 Mutate new offspring at each position in chromosome with a mutation probability;
   - 3.4 Place new offspring in the new population

If the end condition is satisfied, stop, and return the best solution in current population; otherwise, go to step 2.

2.3 Evolution Computation-Based Video Texture Synthesis

Now we apply GA to Synthesize videos:

Encoding. Given a video sequence S with l frames, S={F_1, F_2…F_l}, a solution to video texture synthesis will be represented as an ordered list of numbers. Define each decimal number i(i∈ [1,l])as a gene and let it denote a frame F_i. For instance, (3 4 5 1 2 3 6 7) is the chromosome representation of video sequence {F_3, F_4, F_5, F_1, F_2, F_3, F_6, F_7}. In the following algorithm description, we assume that there are m chromosomes in the population and each chromosome C_k (k∈ [1,m]) denotes a solve to video texture synthesis. The simple permutation representation is a common and popular representation for solving the order-based problems.

Fitness. In order to make new video sequence continuous in vision, we should compute similarities among frames and cost of transitions. First we should calculate similarity matrix of D and transition cost matrix of D'_{ij}. Then we use formula(4) to calculate mathematical expectation E_k and standard error σ_k of transition cost for each chromosome C_k.

$$E_k = \left(\sum_{j=1}^{n-1} D'_{c_j, c_{j+1}} + D'_{c_n, c_1} \right) \times n^{-1} ,$$
$$\sigma_k = \sqrt{\left(\sum_{j=1}^{n-1} (D'_{c_j, c_{j+1}} - E)^2 + (D'_{c_n, c_1} - E)^2 \right) \times n^{-1}} \tag{4}$$

Here, n is the number of frames in C_k, c_j represents the j-th(j ∈ [1,n]) gene of C_k For each chromosome C_k if its E_k is small and its σ_k is large, there are less transitions with high transition costs in the video sequence corresponding to C_k. And thus there are some obvious abruptions in the video sequence; if its E_k is large and its σ_k is small, there are more transitions with low transition costs in the video sequence corresponding to C_k. The video sequence is regarded as repeatedly playing a small clip. It seems that when its E_k and σ_k are both relative small, chromosomes have better continuity and diversity. Therefore, we can define fitness function as a weighted sum of E_k and σ_k. Furthermore, considering that E_k and σ_k may be not at the same quantity-level, we should normalize them at first. Formula of fitness function is as below:

$$Fit(C_k) = \left(\alpha \cdot E_k + \beta \cdot \frac{\sigma_k \cdot \overline{E}}{\overline{\sigma}} \right)^{-1} , \tag{5}$$

Where $\overline{E} = \left(\sum_{t=1}^{m-1} E_t \right) \times m^{-1}$, $\overline{\sigma} = \left(\sum_{t=1}^{m-1} \sigma_t \right) \times m^{-1}$,and $\alpha + \beta = 1$

m is the number of chromosomes in the population. α and β are weight coefficients. In our experiments, we obtain the best result videos in vision when α =0.7 and β =0.3.

Selection Operator. Selection Operator gives preference to better individuals, allowing them to pass on their genes to the next generation. In our experiments, we use roulette wheel method for selection operation. Survivable probability Ps of each individual *C* is directly determined by its fitness.

$$Ps = \frac{Fit(C)}{\sum_{k=1}^{m} Fit(C_k)} \tag{6}$$

Here, *m* is the amount of individuals in population. C_k means the *k*-th individual in population.

Crossover Operator. This process recombines portions of good existing individuals to create even better individuals, it leads population to move towards the best solution. Single-point crossover is adopted and each time two genes from different individuals are exchanged. Crossover probability is set as a random float in [0.5,1].

Mutation Operator. The purpose of Mutation Operation is to introduce new gene and maintain diversity within the population and inhibit premature convergence. Selection of Mutation probability is much important for GA. If its value is too great, the population will not be converged, if its value is too small, the population may be led to premature convergence. In this paper, we set the value to 0.05.

During each iteration of GA, the individual having the largest fitness within all individuals in every generation is selected and checked whether its fitness is larger than a given threshold value of convergence V_t. If the fitness is larger or the number of generations is beyond a specified value G_{max}, the algorithm stops iterations and reports the individual having the largest fitness as the best solution.

3 Results and Analysis

3.1 Results

The current video synthesizing system is implemented in Visual C++6.0. Some video clips designed using the system are shown in the following figures. The system maintains reasonably smooth frames on a PC (Pentium4 2.0GHz). These video clips demonstrate several different video textures produced by the above-mentioned methods.

Fig. 1. a 63-frame source video of a man speaking, result video synthesized by the algorithm in this paper maintains continuity of his head and mouse moving.

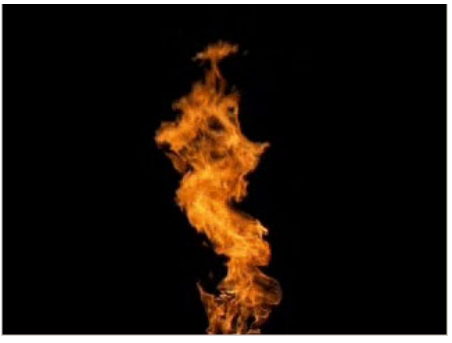

Fig. 2. an 89-frame source video of flame, result video synthesized by the algorithm in this paper maintains continuity of flame jumping

Fig. 3. a 61-frame source video of waterfall, result video synthesized by the algorithm in this paper maintains continuity of water flowing.

Fig. 4. a 63-frame cartoon source video, result video synthesized by the algori-thm in this paper maintains flying regularity of hair and scarf in the source video.

3.2 Analysis of Population Size

Population size is a main key to influence the efficiency and quality of our results. If its value is too small, the algorithm may fail to give us a satisfied solution of video texture synthesis; On the other hand, if its value is too large, the algorithm may waste much of calculating time. Therefore, it is more important to set a proper population size for the efficiency and quality of our algorithm.

In our experiments for each source clip, we set different population sizes to test. When processing the example in Fig.1, we set the number of chromosomes to 30, 50, 100, 150, 300, 500, 800, 1000 and 1500 respectively. Results of our experiments show that the algorithm quickly converges to an unsatisfied solution when the number of chromosomes is tiny; while the number of chromosomes is 500, the algorithm is able to do a fairly work; as the population size is larger than 500, fitness of the best

solution has unobvious change. For the rest figures, satisfactory solutions can be obtained when the population sizes are 800, 500 and 500 respectively.

3.3 Analysis of Max Generation Number

Number of Max generation is another main key to influence the efficiency of our results. When processing the example in Fig.1, we specify that the population size is 500, and set the number of Max generations to 150, 300, 500, 800, 1000, 1500, 2000 and 3000 respectively. Results of our experiments show that when the number of Max generations is 1000, satisfied solutions can be obtained. For the rest Figures, we specify that the three population sizes are 800, 500 and 500 respectively, satisfied solutions can be obtained when the number of Max generations are 1500, 1000, and 1000.

3.4 Analysis of Computation Complexity

In Schödl's algorithm[8], the computation complexity of pre-processing is much more than that of creating video sequence. Without computing the part time of pre-processing, the total computation complexity of Video Textures Algorithm is $O(L^2N^2)$, where L is the length of result video sequence and N is the number of transitions. However the total computation complexity of our GA-based algorithm is $O(CG)$,where C is the population size and G is the number of Max generation. Only when the computation complexity of pre-processing is ignored, the two algorithms have comparable efficiency. Obviously, GA-based method is more efficient.

4 Conclusions

In this paper, evolution computation technique is introduced into the processes of synthesizing videos, and a new video texture synthesis method is suggested. This method uses an appropriate fitness function instead of much complex pre-processing of source video clip. Additionally, constraints of selection operator, crossover operator and mutation operator can ensure this algorithm to get high-quality video results quickly. Compared with other existing representative video texture synthesis algorithms, this algorithm has less computational complexity and improves the efficiency of synthesis.

Although we put more emphasis on the continuities and regularities of video sequences in our algorithm, we need to care for the structures of them. A more appropriate fitness function is anticipated to be found in future research work.

References

1. Heeger D.J. and Bergen, J.R.: Pyramid-based texture analysis/synthesis. Computer Graphics, ACM SIGGRAPH. (1995), 229-238
2. Bonet J.S.D.: Multiresolution sampling procedure for analysis and synthesis of texture images. Computer Graphics, ACM SIGGRAPH. (1997) 361-368
3. Efros A. and Leung T.: Texture synthesis by non-parametric sampling. International Conference on Computer Vision, Vol 2. (1999) 1033-1038

4. Portilla J. and Simoncelli E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. International Journal of Computer Vision. vol.40, No.1. (2000) 49-70
5. Wei L.Y. and Levoy M.: Fast texture synthesis using tree-structured vector quantization. Computer Graphics, ACM SIGGRAPH. (2000) 479-488
6. Liang L., Liu C., Xu Y.Q., Guo B. and Shum H.Y.: Real-time texture synthesis by patch-based sampling. ACM Transactions on Graphics, Vol. 20, No. 3. (2001) 127-150
7. Efros A.A. and Freeman W.T.: Image quilting for texture synthesis and transfer. Proceedings of SIGGRAPH 2001. (2001) 341-346.
8. Schödl A., Szeliski R., Salesin D.H. and Essa I.A.: Video textures. Proceedings of SIGGRAPH 2000. (2000) 489-498.
9. Schödl A., ESSA I. A. Controlled animation of video sprites. Proceedings of ACM Symposium on Computer Animation 2002, (2002) 121~127
10. Wang Y., Zhu S. C. A generative model for textured motion: Analysis and synthesis. In: Proceedings of European Conf. on Computer Vision (ECCV)2002, Copenhagen, (2002) 583~598
11. Doretto G., Chiuso A., Soatto S., And Wu Y. Dynamic textures. International Journal of Computer Vision2003, (2003) 51(2): 91~109.
12. Kwatra V., Schödl A., Essa I. A., Turk G., and Bobick A. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. In Proceedings of ACM SIGGRAPH 2003. (2003) 277~286
13. Bhat K.S., Seitz S.M., Hodgins J.K., and Khosala P.K., Flow-based video synthesis and editing. In Proc. of ACM SIGGRAPH 2004. (2004), 360~363
14. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press. (1975)