Video Input Synthesize Waterfall Scenes

Yu Guan, Wei Chen*, Long Zhang, Chengfang Song, Yi Gong, Qunsheng Peng
{guanyu,chenwei,lzhang,songchengfang,ygong,peng}@cad.zju.edu.cn

State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310058, China.

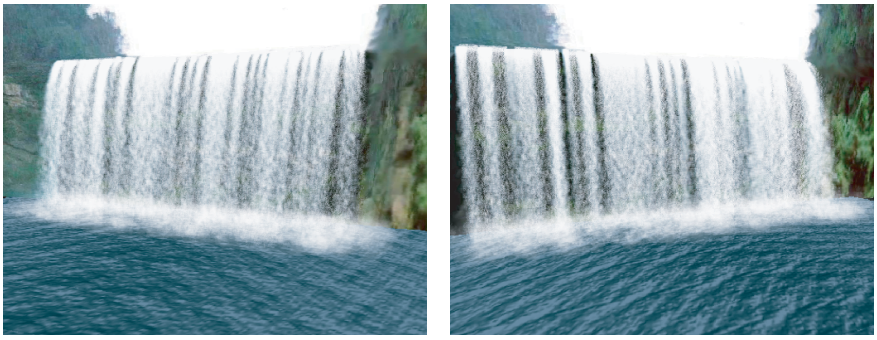


Fig. 1. Two different waterfall scenes produced by our algorithm. The image resolution is 512×400.

Abstract. In this paper, we propose an new approach for modelling and rendering dynamic and realistic waterfall scenes that from both texture and motion analysis based on acquired video sequences. We first generate a set of basis texture sprites, which capture inherent appearances and motions of waterfall scenes contained in video sequences. To model the shape and motion of new waterfall scenes, we construct a set of flow lines taking account of physical principles. Along each flow line, the basis texture sprites synthesize a sequence of dynamic texture sprites in 3D space according to spatio-temporal dynamic synthesis model (STDS). These texture sprites are further displayed with point splatting technique, which can be accelerated in programmable graphics hardware. By choosing different sets of basis texture sprites, waterfall scenes with different appearances and shapes can be conveniently simulated. Experimental results demonstrate that our approach achieves realistic effect and real-time frame rates in consumer PC platform, and can be adopted in entertainment-relating applications such 3D simulator, video games, *etc.*

1 Introduction

Waterfall scenes contain rich stochastic motion patterns which are characterized by the movement of a large number of small elements. In this paper, we call these motion patterns texture sprites. Consider a sequence of images of a moving

* Corresponding author

scene. Each image is an array of positive numbers that depend upon the shape, pose and motion of the scene. Traditional approaches to simulate fluid scenes can be divided into two categories, namely, particle system based methods and image or video synthesis methods. Particle system methods model the scenes as a sequence of animated particles. However, they make use of one texture for all particles and are difficult to simulate the fluid scenes which possess time-varying appearance. On the other hand, video synthesis methods decompose image sequences into many small 2D samples and analyze the implied patterns. By means of texture synthesis techniques, these 2D samples can be used to generate varied new scenes. One main disadvantage of these methods lies in that it can not provide free viewpoint simulation, which is mandatory for most entertainment-relating applications. Note that, both methods lack a means to model dynamic motions and render realistic appearances of fluid scenes simultaneously. In this paper we first analyze sequences of images which are contained in video, capture inherent appearances and motions of fluid, generate a set of basis texture sprites. To model the shape and motion of new waterfall scenes, we construct a set of flow lines taking account of physical principles. Along each flow line, the basis texture sprites synthesize a sequence of dynamic texture sprites in 3D space according to spatio-temporal dynamic synthesis model (STDS). These texture sprites are further displayed with point splatting technique, which can be accelerated in programmable graphics hardware. By choosing different sets of basis texture sprites, waterfall scenes with different appearances and shapes can be conveniently simulated.

2 Related Work

Creating realistic animations of fluid flow has been an active research area in the past decade. Particle system methods [9–11] are commonly used to simulate complex group dynamics such as flocking birds. A particle system is composed of one or more individual particles, each of which has attributes that affect the behavior itself or how and where it is displayed. Particles are normally represented by graphical primitives such as points or lines, leading to high performance. In addition, particle system methods introduce some type of random element, which is used to control the particle attributes such as position, velocity and color. However, particle system methods concentrate on the behaviors of particles solely. Typically, only one color or texture is adopted for the appearances of all particles. In this way, achieving realistic effects with particle system methods are far from be solved, as shown in Figure 9(a).

There has been extensive work [8, 6, 1, 5] on simulating the motion of fluid flow based on image or video synthesis techniques. Probably the simplest way is the video texture approach introduced by Schödl *et al.* [15]. Based on an input video clip, its motion cycles is first analyzed and extracted. New and similar-looking video clips of arbitrary length are then synthesized by re-arranging image frames. This idea is further extended [13] to allow for high level control over moving objects in video sequences. Rather than using an image frame as the

synthesis element, the approach introduced by Wei and Levoy [16] makes use of Markov Random Field texture models to generate textures through a deterministic searching process. It can be applied to create 3D temporal textures of fluid-like motion.

On the other hand, Doretto *et al.* [4] use Auto-Regressive filters to model and edit the complex motion of fluids contained in video sequences. To model the motion of texture particles in video sequences, Wang and Zhu [17] propose to adopt a second order Markov chain. Szummer and Picard [14] propose to capture and synthesize dynamic textures by means of the spatio-temporal auto-regressive model (STAR). Bar-Joseph *et al.* [2] present an algorithm based on statistical learning for synthesizing static and time-varying textures matching the appearance of an input texture. Kwatra *et al.*[7] demonstrate an algorithm for image and video synthesis using graph cut approach.

Recently, Bhat *et al.* [3] present an algorithm for synthesizing and editing video sequences of natural phenomena that exhibit continuous flow patterns. The algorithm analyzes the motion of textured particles in the input video sequence along user-specified flow lines, and synthesizes new video sequence with arbitrary length by enforcing temporal continuity along a second set of user-specified flow lines. However, it is limited to input sequences with nearly stationary flow patterns as shown in Figure 9(b).

Note that, aforementioned methods based on video textures, video sprites or video synthesis are all restricted to one fixed view point because they are essentially image synthesis techniques. Consequently, they can not provide true 3D effects and fail to satisfy the requirements of spatial walkthrough under arbitrary viewpoints.

3 Modelling and Rendering of Waterfall Scenes

To fix notation, let $I[0,\tau]$ denote a texture sprite sequence in a discretized time interval $[0,\tau] = \{0,1,2,\cdots,\tau\}$. Each texture sprite $\Lambda = \{(x,y) : 0 \leq x,y \leq L\}$, L is the size of texture sprite. For $(x,y) \in \Lambda$ and $t \in [0,\tau]$, $I(x,y,t)$ denotes the pixel color. $I(t) \in I[0,\tau]$ is a single texture sprite. Let $B[0,\tau]$ denote a basis texture sprite sequence. $B(x,y,t)$ denotes the pixel color. $B(t) \in B[0,\tau]$ is a single basis texture sprite.

3.1 Conceptual Overview

Video input synthesize waterfall scenes can be divided into four steps. First, we construct a sequence of texture sprites based on the analysis to the input video sequence. Second, we interactively design flow lines by adjusting physical parameters, including gravity, wind and height etc. The flow lines act as the skeleton of the expected waterfall. Third, new dynamic texture sprites are generated automatically along each flow line, which constitute the running waterfall. Finally, dynamic texture sprites are displayed by means of point splatting technique in programmable graphics hardware. Figure 2 illustrates the whole pipeline of our approach.

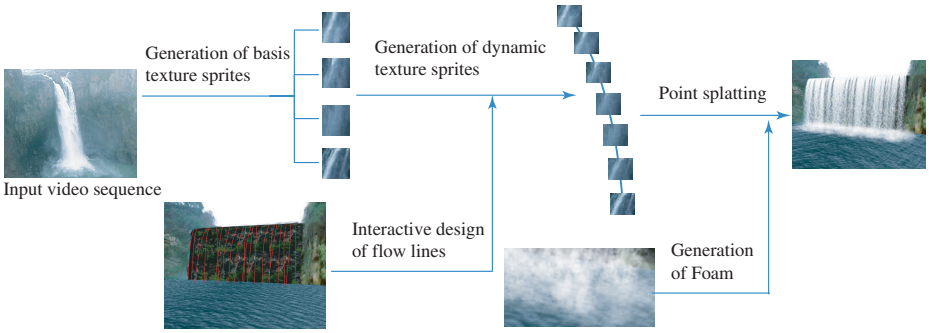


Fig. 2. The conceptual overview of our approach.

3.2 Extraction of Basis Texture Sprites

Natural scenes contain rich stochastic motion patterns which are characterized by the movement of a large number of small deformable. Note that, the time-varying waterfall moves from up to down and its shape can be decomposed into multiple small components, i.e., 2D samples. We select representative samples from the input video sequence manually and analyze their shape variations, yielding a set of basis texture sprites. For one given waterfall scene, the extracted basis texture sprites represent all possible shape variations.

Let $N(t)$ denotes the number of different shape basis texture sprites at time t. Obviously, $N(t)$ will increase stepwise in process of time. However, at each fixed time t, $N(t)$ has relative stability. Certainly, $N(t)$ does not grow infinitely. It has an upper limit.

When $t \rightarrow t + \Delta t$, the rate of growth of $N(t)$ is $\lambda(t)$. $\lambda(t)$ is related to time. The rate of change of $\lambda(t)$ is in direct proportion to time. Basis texture sprites set model as follows:

$$\begin{cases} d\lambda(t)/dt = -a\lambda(t) \\ \lambda(0) = b \end{cases} \qquad (a, b \; are \; constant) \qquad (1)$$

We obtain:

$$\lambda(t) = b * exp(-at) \qquad (2)$$

In regard to $N(t)$:

$$\begin{cases} \frac{dN(t)}{dt}/N(t) = \lambda(t) \\ N(0) = c \end{cases} \qquad (c \; is \; constant) \qquad (3)$$

Substitute equation (2) into equation (3):

$$N(t) = c * exp[b(1 - exp(-at))/a] \qquad (4)$$

At fixed time t, we compute the number of basis texture sprites according to equation (4). Then, we extract the $N(t)$ basis texture sprites from video. The amount of basis texture sprites N:

$$N = \sum_{t=0}^{\tau} N(t) \tag{5}$$

The N basis texture sprites compose a basis texture sprites set. The left image of Figure 3 shows one frame of input video sequence, while the right image illustrates the extracted basis texture sprites.

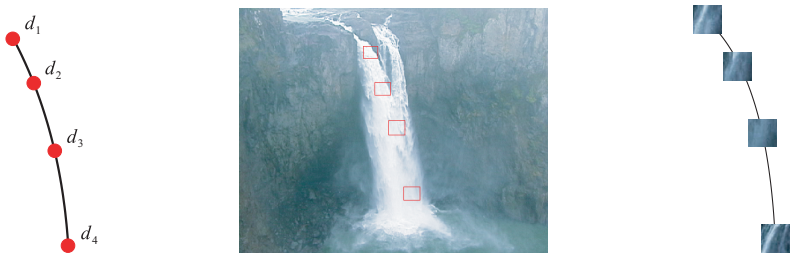


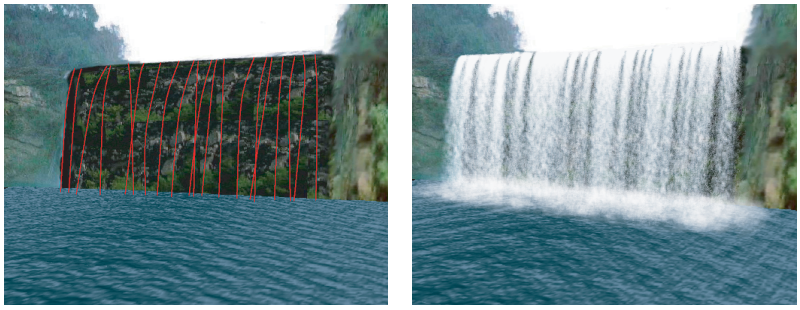
Fig. 3. Interactive generation of basis texture sprites. (a) One frame of input video sequence. (b) Input video. (c) The extracted basis texture sprites.

3.3 Interactive Design of Flow Lines

The second step is to interactively design the flow lines. The overall shape of the underlying waterfall is controlled by determining the parameters, such as the number of flow lines and the approximated length. We take a single flow line shown in Figure 3 (a) as our example. Any particle begins at the start of the flow line d_1 and passes through a sequence of positions during its moving. Its corresponding texture sprite varies as shown in Figure 3 (c). To represent the temporal evolution of particles, we assume that Y axis is the direction of gravity and the direction of the wind is parallel to XZ plane. The particles are affected by the gravitational acceleration g, the air resistance f and the wind velocity $windv$. Thus, the velocity $(V_x(t), V_y(t), V_z(t))$ and position $(P_x(t), P_y(t), P_z(t))$ of one particle can be written as follows:

$$\begin{cases} V_x(t) = v_x + windv \cdot cos\theta \\ V_y(t) = v_y - \int_0^t (g - f/m)dt \\ V_z(t) = v_z + windv \cdot sin\theta \end{cases} \qquad \begin{cases} P_x(t) = p_x + \int_0^t V_x dt \\ P_y(t) = p_y + \int_0^t V_y dt \\ P_z(t) = p_z + \int_0^t V_z dt \end{cases} \tag{6}$$

Here, (v_x, v_y, v_z) and (p_x, p_y, p_z) are initial velocity and position. θ is angle between the wind direction and X axis.



(a) (b)

Fig. 4. (a) Designed flow lines by modifying parameters. (b) Final generated waterfall.

With given control parameters, our algorithm automatically generates flow lines according to Equation 6. Users are allowed to interactively modify these parameters to achieve desired effect. One example is demonstrated in Figure 4.

3.4 Spatio-Temporal Dynamic Synthesis Model

In process of time and spatial position, each texture sprite varies significantly along its flow line. However, the transition between basis texture sprites is not smooth. To generate all particles along the flow lines, we have to produce dynamic texture sprites in virtue of the basis texture sprites. One simplest way is to linearly interpolate the basis texture sprites. Note that the appearance of each texture sprite dependents on its position, we use the tangent information of the flow line between two adjacency basis texture sprites. This modification produces smooth transition between two basis texture sprites along the flow line. This method can be formulated as:

$$I(x,y,t) = (t^3, t^2, t, 1)M \begin{pmatrix} B(x,y,t_i) \\ B(x,y,t_{i+1}) \\ p_i^{'} \\ p_{i+1}^{'} \end{pmatrix} + n \tag{7}$$

In equation (7), $B(t_i)$ and $B(t_{i+1})$ are two adjacent basis texture sprites, whose position are p_i and p_{i+1}. The $p_i^{'}$ and $p_{i+1}^{'}$ are the tangents at p_i and p_{i+1} respectively. M is a given transition matrix, and n is a noise process for the residues. The generative texture sprites sequence sort in chronological order. Figure 5 gives one synthetic example.

The texture sprite set generated in the previous step can be extended to support deformed variations. We can rotate and scale texture sprite at one point along the flow line as shown in Figure 6.

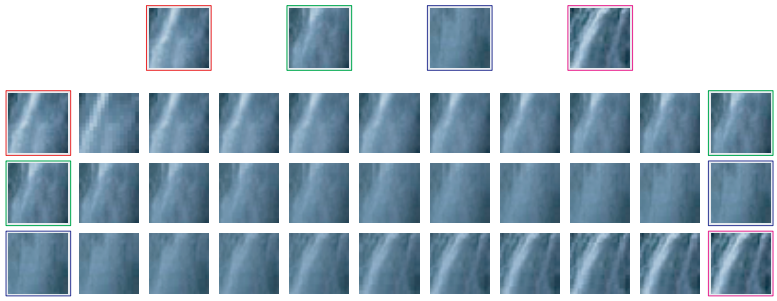


Fig. 5. The top four images are selected basis texture sprites. By interpolating them sequentially, a set of dynamic texture sprites are generated.

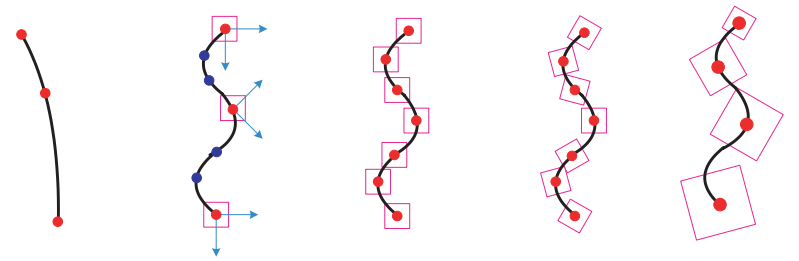


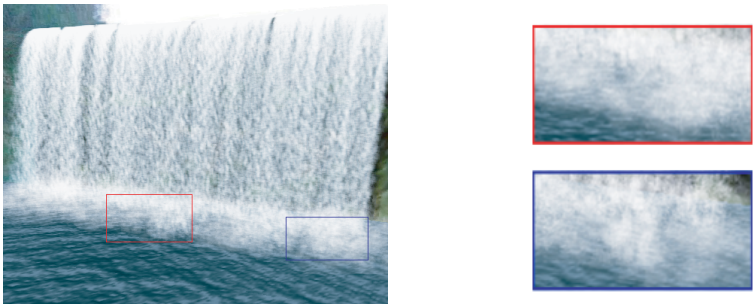
Fig. 6. Five steps to generate the falling waterfall. (a) Extract basis texture sprites from natural waterfall or video. (b) Compute the pixel color of other texture sprites along the user-specified flow lines over time. (c) Generate dynamic texture sprites. (d) Rotate dynamic texture sprites along the flow line. (e) Scale dynamic texture sprites along the flow line.

3.5 The Simulation of Foam

To simulate the foam caused by the collision between the falling waterfall and the water pool, we construct a set of radial flow lines, yielding a great number of foam patterns. They are simulated using another type of texture sprite as shown in Figure 7(b).

3.6 Rendering with Point Splatting Technique

To display the falling waterfall scene, each resultant texture sprite is represented by a textured quad parallel to the view plane. Generally speaking, all dynamic texture sprites can be divided into two classes, namely, axis-aligned and non-axis-aligned sprites. For each axis-aligned sprites, it is rendered as a single textured rectangle in 2D screen directly, which can be accelerated greatly in GPU. In this way, the bandwidth between CPU and GPU is saved remarkably. Our experiments demonstrates that this scheme increases about the frame rates by up



(a) (b)

Fig. 7. (a) One waterfall scene with foam. (b) Local enlargement of foams.

to 50%. In order to draw non-axis-aligned sprites, a rotated quad is created for each sprite. The cost to render a rotated quad is much larger than that of the axis-aligned sprite. Fortunately, most of dynamic texture sprites are axis-aligned.

Note that, the generation of dynamic texture sprites is performed on-the-fly in programmable graphics hardware. The calculation is accomplished completely by means of a simple vertex shader. To exploit the power of retained rendering mode, all texture sprites are loaded in video memory in advance and updated on-the-fly.

4 Experimental Results

We implemented the proposed approach in a PC equipped with P4 1.7 HZ CPU and an NVidia 6800 GT video card. The vertex shader and pixel shader are written in OpenGL Shading Language. The average frame rates of our algorithm is about 30 fps at the image resolution of 512×400.

The extraction of the basis texture sprites is very simple. We can obtain different types of basis texture sprites from acquire video sequences, leading to different waterfall scenes as shown in Figure 8.We also compare our results with that of particle system method [12] and 2D video synthesis techniques [3]. Corresponding images are illustrated in Figure 9. It is obvious that our algorithm not only achieves realistic effect, but also provides more flexibility.

5 Conclusions and Future Work

In this paper we present a simple and efficient method *Video Input Synthesize Waterfall Scenes* for simulating waterfall animation. On the issue of representation, we present a novel algorithm of manipulating and animating dynamic texture sprites in 3D space. They are used to generate animated waterfall with arbitrary length, appearances at the cost of little user interaction. Experimental results show that our algorithm achieves real-time simulation and realistic effect. As further work is concerned, we would like to introduce some parameters for

(a) (b)

Fig. 8. Different waterfall scenes with different basis texture sprites.

enhancing the irregularity of texture sprites because our current solution is limited to simulate some waterfall whose appearance varies intensely. Furthermore, we intend to extend the algorithm to other types of fluid, including fountains, smokes, clouds, *etc.*

References

1. Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. *In Proceedings of SIGGRAPH 1997, Computer Graphics Proceedings, Annual Conference Series*, 1997, 361-368. ACM, ACM Press / ACM SIGGRAPH.
2. Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *In IEEE Transactions on Visualization and Computer Graphics*, 7(2) 1996, 120-135.
3. Kiran S.Bhat, Steven M. Seitz, Jessica K. Hodgins, Pradeep K. Khosla. Flow-based Video Synthesis and Editing. *ACM Transactions on Graphics*, 23(3) 2004, 360-363. ACM Press.
4. G. Doretto, A. Chiuso, S. Soatto, Y. N. Wu. Dynamic Textures.*International Journal of Computer Vision*, 51(2) 2003, 91-109.
5. Gianfranco Doretto, Stefano Soatto. Editable Dynamic Textures. *In Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003*. 2003, 137-142.
6. David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. *In Proceedings of SIGGRAPH 1995, Computer Graphics Proceedings, Annual Conference Series*, 1995, 229-238. ACM, ACM Press / ACM SIGGRAPH.
7. V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts.*In ACM SIGGRAPH 2003*, 277-286.
8. Randal C. Nelson and Ramprasad Polana. Qualitative recognition of motion using temporal texture. *CVGIP: Image Understanding*, 56(1) 1992, 78-89.
9. W.T.Reeves, Particle Systems-A Technique for Modelling a Class of Fuzzy Objects, *Computer Graphics*, 1983, 17(3): 359-376.
10. W.T.Reeves, Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems, *Computer Graphics*, vol. 19, no. 3, 1985, 313-322.

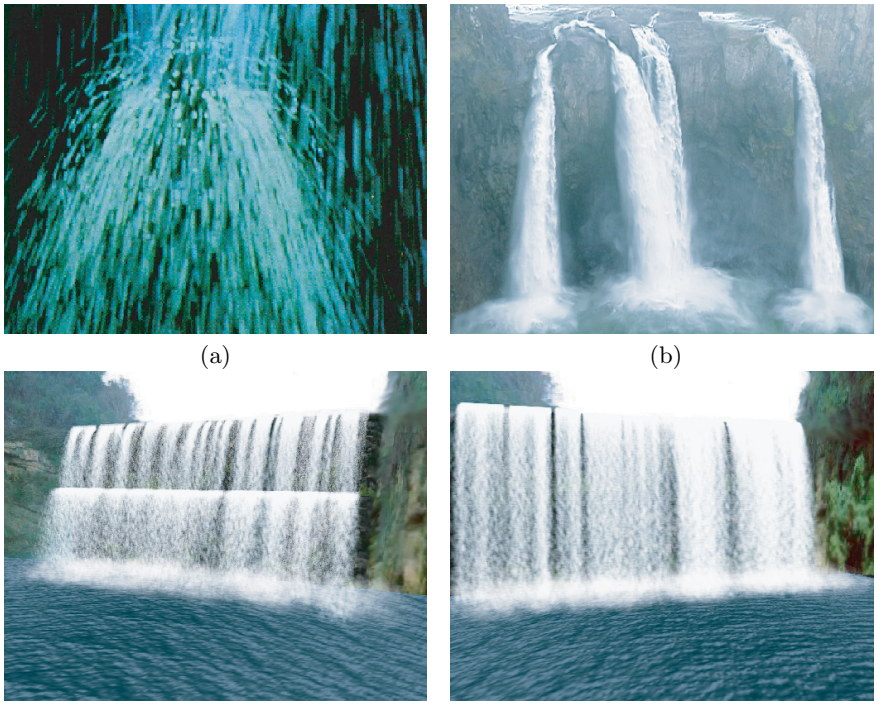


Fig. 9. (a) Result by particle system method.(b) Result by video synthesis method based on the input video sequence shown in Figure 3 (b). (c) and (d) are our results based on the same video sequence.

11. C. W. Reynolds, Flocks, Herds, and Schools: A Distributed Behavioral Model, *Computer Graphics*, vol. 21, no. 4, 1987, 25-34.
12. Karl Sims. Particle Animation and Rendering Using Data Parallel Computation. *Computer Graphics*, 24(4) 1990, 405-413. ACM Press.
13. A.Schödl, I.A.ESSA, 2002. Controlled animation of video sprites, In *ACM SIGGRAPH Symposium on Computer Animation*, 121 - 128.
14. M. Szummer and R. W. Picard. Temporal texture modeling. *In IEEE International Conference on Image Processing*, Lausanne, Switzerland, volume 3, Sept 1996.
15. A. Schödl, R. SZELISKI, D. H. SALESIN, I. ESSA. Video textures. *In Proceedings of ACM SIGGRAPH 2000*, 489-498.
16. Li-Yi Wei,Marc Levoy. Fast texture synthesis using tree-structured vector quantization. *In Proceedings of ACM SIGGRAPH 2000*, 2000, 479-488.
17. Y. Wang, S. C. Zhu. A generative model for textured motion: analysis and synthesis. *In Proceedings of European Conference on Computer Vision (ECCV) 2002*, 582-598.