A Generative Method for Textured Motion: Analysis and Synthesis

Yizhou Wang and Song-Chun Zhu

Dept. of Comp. and Info. Sci., Ohio State Univ., Columbus, OH 43210, USA
{wangyiz, szhu}@cis.ohio-state.edu

Abstract. Natural scenes contain rich stochastic motion patterns which are characterized by the movement of a large number of small elements, such as falling snow, raining, flying birds, firework and waterfall. In this paper, we call these motion patterns *textured motion* and present a generative method that combines statistical models and algorithms from both texture and motion analysis. The generative method includes the following three aspects. 1). Photometrically, an image is represented as a superposition of linear bases in atomic decomposition using an over-complete dictionary, such as Gabor or Laplacian. Such base representation is known to be generic for natural images, and it is low dimensional as the number of bases is often 100 times smaller than the number of pixels. 2). Geometrically, each moving element (called moveme), such as the individual snowflake and bird, is represented by a deformable template which is a group of several spatially adjacent bases. Such templates are learned through clustering. 3). Dynamically, the movemes are tracked through the image sequence by a stochastic algorithm maximizing a posterior probability. A classic second order Markov chain model is adopted for the motion dynamics. The sources and sinks of the movemes are modeled by birth and death maps. We adopt an EM-like stochastic gradient algorithm for inference of the hidden variables: bases, movemes, birth/death maps, parameters of the dynamics. The learned models are also verified through synthesizing random textured motion sequences which bear similar visual appearance with the observed sequences.

1 Introduction: objectives and previous work

Natural scenes contain rich stochastic motion patterns which are characterized by the movement of a large number of small deformable elements (or particles). For example, raining, snowing, bird flock, moving crowd, firework, waterfalls, and so on. The analysis and synthesis of such motion patterns, called *textured motion* in this paper, are important for a variety of applications in both vision and graphics, and stimulate growing interest of the two communities.

Graphics methods. In graphics, the objective is to render textured motion in video or cartoon animation, and the quality of the rendered motion is usually measured by three basic criteria.

1. It should be *realistic*. This motivates work for modeling and learning the photometric and dynamic properties from real video due to the complexity of textured motion. Usually, data driven statistical modeling is often more appropriate than physically-based modeling.
2. It should be *stylish*. This is required for applications in non-photo realistic rendering (NPR), for example, rendering a waterfall in a cartoon movie. It is desirable to separate the dynamics of motion from its photometric appearances, so that the video appears symbolic but with realistic motion.
3. It should be *controllable*. For a better blending of the motion with other 3D objects in a scene, one should increase the degree of freedoms in maneuvering the motion. For example, it is desirable to control the sources and sinks where the motion elements appear and disappear, to control the individual moving elements, to change its motion direction etc.

In the graphics literature, both physically-based and data driven models are reported. The former include the work which create animations of fire and gaseous phenomena with particles [12, 5]. The latter include the 1). video texture[14] which finds smooth transition points in a video sequence from which the video could be replayed with minimum artifacts; 2). 3D volume texture[17] which generates motion through non-parametric sampling from an observed video motivated by recent work on texture synthesis. Though the statistical models in the video texture or 3D volume texture can render some realistic animations, such models do not model the dynamic and geometric properties of the moving elements.

Vision methods. In computer vision, the analysis of textured motion has applications for video analysis, such as motion segmentation, annotation, recognition and retrieval, detecting abnormal motion in a crowd, and so on. Needless to say that a good vision model of textured motion is useful for animation in graphics as mentioned above. For such applications, a vision model should satisfy the following properties.

1. It should be *sufficient* and *general*. It is not enough to just render a synthesized sequence that look like the original as the video texture do, the model should also be able to capture the variability and therefore can be generalized to new data.
2. It should be *parsimonious* and *low dimensional* for computation. This requests the model capture the semantics of the motion. This also requests the modeling of photometric, geometric, and dynamic aspects of the motion — consistent with the graphics criteria.

In the vision literature, as these motion patterns lie in the domains of both motion analysis and texture modeling, statistical models are proposed from both directions with a trend of merging the two. In the following, we briefly review these work to set the background for our method.

Early vision work on textured motion was done by (Szummer and Picard, 1996)[16] who adopt a spatial-temporal auto-regression (STAR) model from

(Cliff and Ord, 1976)[4]. Let $\mathbf{I}(x, y, t)$ be the intensity of a pixel (x, y) at time t, a STAR model assumes that $\mathbf{I}(x, y, t)$ is a regression of its neighboring pixels

$$\mathbf{I}(x, y, t) = \sum_{i=1}^{p} a_i \mathbf{I}(x + \delta x_i, y + \delta y_i, t + \delta t_i) + N(0, \sigma^2), \tag{1}$$

where $(\delta x_i, \delta y_i, \delta t_i)$ is the displacement of a neighboring pixel in space and time, and $a_i, i = 1, ..., p$ are parameters to be fit. A linear (or partial) order is imposed so that $\mathbf{I}(x, y, t)$ only depends on pixels at previous frames $\delta t_i < 0, \forall i$ for fast synthesis. Such model can be considered as an extension from a causal Gaussian Markov random field model (GMRF) used in texture modeling by adding the time dimension. Along the line of texture modeling, Bar-Joseph et.al.[1] extended the work by Heeger and Bergen (1995) and others[18] to multi-resolution analysis in a tree structured representation, in a similar spirit to (Wei and Lovoy, 2000).

Although these algorithms can show synthesis of good motion, we argue that the concept of treating a motion pattern as a solid texture is perhaps not appropriate. Because textures are physically the status of systems with massive elements at thermodynamic equilibrium characterized by maximum entropy distributions[18]. This assumption is not observed in textured motions, for example, fire or gaseous turbulence, these patterns are clearly not at equilibrium.

The recent work (Soatto, Doretto, and Wu, 2001)[15] engages the motion dynamics explicitly. By a SVD analysis, Soatto et al. represent an image $\mathbf{I}(t)$ by a small number of principal components. The projections of $\mathbf{I}(t)$ on these components, denoted by $x(t)$, is modeled by a Markov model,

$$x(t+1) = Ax(t) + Bv(t), \quad \mathbf{I}(t) = Cx(t) + n(t), \tag{2}$$

where $v(t)$ is the noise driving the motion and $n(t)$ is the image noise for the reconstruction residues. The parameters A, B, C are learned by maximum likelihood estimation (MLE). This model can generate impressive synthesis for a variety of motion patterns and can also be used for recognition[13].

Our method. In this paper, we present a generative method for the analysis and synthesis of textured motion, motivated by the vision and graphics criteria discussed above. Our model includes the following three aspects.

1. *Photometrically*, an image is represented as a superposition of linear bases in atomic decomposition using an over-complete dictionary, such as Gabor or Laplacian. Such base representation is known to be generic for natural images, and it is low dimensional as the number of bases is often 100 times smaller than the number of pixels.
2. *Geometrically*, each moving element (called moveme), such as the individual snowflake, bird, is represented by a template which is a group of several spatially adjacent bases. Such templates are deformable to account for the variabilities of the elements and are learned through clustering.
3. *Dynamically*, the movemes are tracked through the image sequence by a stochastic algorithm maximizing a posterior probability. A classic Markov

chain model is adopted for the motion dynamics, as in[15]. The sources and sinks of the movemes are modeled by birth and death maps.

We adopt an EM-like stochastic gradient algorithm for inference of the hidden variables: bases, movemes, birth/death maps, parameters of the dynamics.

2 A generative model for images and video

To fix notation, let $\mathbf{I}[0,\tau]$ denote an image sequence on a 2D lattice $\Lambda = \{(x,y) : 0 \le x, y \le L\}$ in a discretized time interval $[0,\tau] = \{0,1,2,...,\tau\}$. For $(x,y) \in \Lambda$ and $t \in [0,\tau]$, $\mathbf{I}(x,y,t)$ denotes the pixel intensity, and $\mathbf{I}(t) \in \mathbf{I}[0,\tau]$ is a single image frame.

2.1 Image representation: from pixels to bases

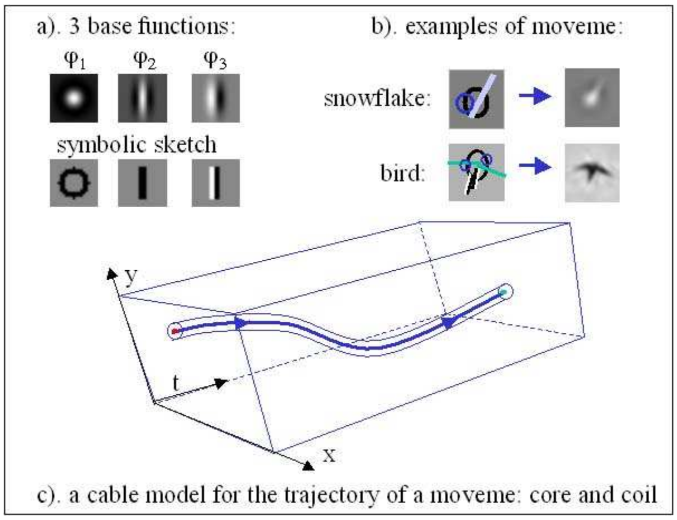

Fig. 1. A "cable model" for movemes.

In this section, we study the representation of a single image frame $\mathbf{I} \in \mathbf{I}[0,\tau]$. For clarity, we remove the time index. We represent an image as a superposition of a small number of image bases, in a scheme which is often called *atomic decomposition* in wavelets and image coding[9, 10, 3].

$$\mathbf{I} = \sum_{j=1}^{N} \alpha_j \mathbf{b}_j + \mathbf{n}, \quad \mathbf{b} \in \Delta. \tag{3}$$

In equation (3), $\mathbf{b}_j$ is an image base from a dictionary Δ, α_j is its coefficient, and $\mathbf{n}$ is a noise process for the residues. The dictionary includes all bases which are transformed versions of three base functions (mother wavelets) $\psi_\ell, \ell = 1, 2, 3,$

$$\Delta = \{T_{x,y,\theta,\sigma} \circ \psi_\ell \ : \ (x,y) \in \Lambda, \theta \in [0, 2\pi), \sigma \in [\sigma_{\min}, \sigma_{\max}], \ell = 1, 2, 3\}$$

$T_{x,y,\theta,\sigma}$ denotes a transform with (x, y, θ, σ) for translation, rotation, and scaling respectively.

We denote the set of base functions by $\Psi = \{\psi_\ell, ell = 1, 2, 3\}$. We choose the Laplacian of Gaussian (LoG), Gabor cosine (Gcos), and Gabor sine (Gsin) shown in Figure 1.a. These base functions represents blobs, bars and step edges respectively (see the symbolic sketches in Figure 1.a). We choose 8 scales, and 12 orientations.

Thus we transform an image $\mathbf{I}$ into a base representation, called a *base map*.

$$\mathbf{B} = (\mathbf{b}_j = (\alpha_j, \ell_j, x_j, y_j, \theta_j, \sigma_j) : \ j = 1, 2, ..., N).$$

As Δ is over-complete, we should discuss how $\mathbf{B}$ is inferred from $\mathbf{I}$ later. We choose the base representation for two reasons.

1. *Low dimensionality.* The number of bases is usually 100-fold smaller than the number of pixels. Figure 2 shows a snowing sequence, each frame can be approximated by $N \approx 100$ bases (see Figure 2.b). When N increases to 800 bases, the reconstructed images in Figure 2.c) are of very high precision. This also introduces a *coarse-to-fine* strategy for computation.
2. *Generality.* It is well known that the LoG and Gabor bases are generic representations for the ensemble of natural images[11], and are also fundamental to human visual perception.

2.2 Image representation: from bases to movemes

In natural image sequences, the image bases often form spatially coherent groups. This is most evident in sequences where the moving elements (or "movemes") are identifiable, such as the individual snow flakes, and flying birds. Figure 1.b shows two examples. A snow flake is a sum of three bases: 2 ψ_1's and 1 ψ_2 at various scales and space displacements. A bird consists of 7 bases: 3 ψ_1's, 2 ψ_2's 2 ψ_3's. The number of bases, and their relative positions and coefficients may various between the movemes. By defining a distance between the movemes, one can cluster the movemes into a small number of deformable templates.

$$\Phi = \{\phi_\ell(\beta) : \ell = 1, 2, ..., n\}$$

with ℓ indexing the moveme types and β being the parameters for relative deformations of the bases within a moveme. Thus we obtain a dictionary of movemes with some transformations,

$$\Pi = \{ \ T_{x,y,\theta,\sigma} \circ \phi_\ell \ : \ (x,y) \in \Lambda, \theta \in [0, 2\pi), \sigma \in [\sigma_{\min}, \sigma_{\max}], \ell \ \}. \tag{4}$$

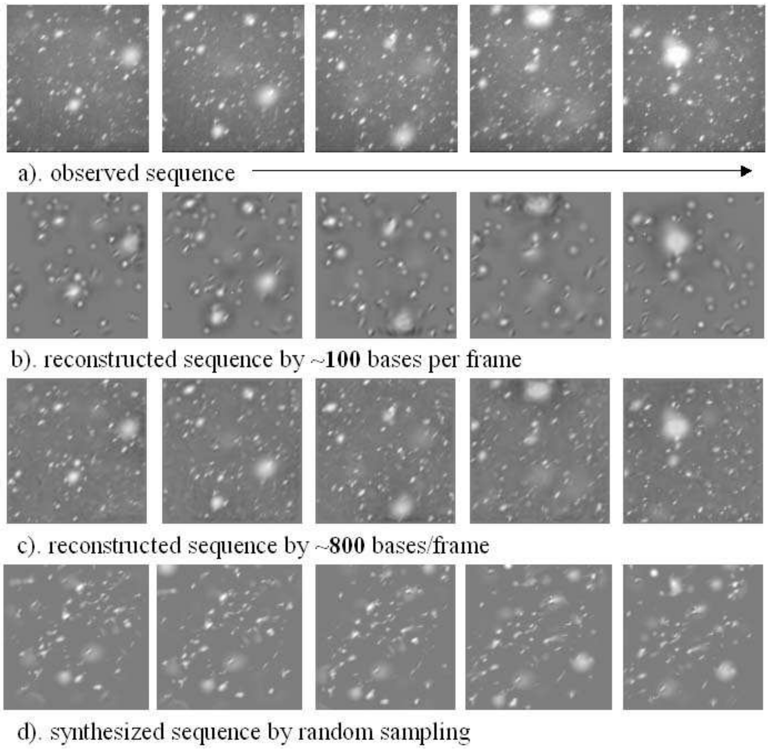

Fig. 2. Example of a snowing sequence. (see snow_obs.avi and snow_syn.avi for movies)

In practice, not all bases are necessarily grouped into movemes, we call the ungrouped ones *free bases*, and for clarity of notation, we treat them as degenerated movemes, i.e. each moveme has one base. For the N bases in the base map $\mathbf{B}$, suppose we group them into K movemes, then we arrive at a more meaningful representation of the image, with dimensions further reduced than $\mathbf{B}$.

$$\mathbf{M} = (\pi_j = (\ell_j, x_j, y_j, \theta_j, \sigma_j, \beta_j), j = 1, 2, ..., J), \qquad J \ll N.$$

Each moveme π_j is represented by $1 \leq \ell_j \leq n$ for the type of the deformable template, $x_j, y_j, \theta_j, \sigma_j$ for the position, orientation, and scale of the overall moveme, and β for the deformable within the moveme.

During the computation, we should learn the deformable templates Ψ_ℓ and compute the movemes and free bases $\mathbf{M}$ from images. For example, Figure 3.a displays the symbolic sketches for a set of typical deformable templates of the snowing sequence shown in Figure 2. Figure 3.b shows 120 random movemes sampled from the moveme dictionary Π. Each moveme is a snow flake. This sample show the variety and generality of the deformable models learned with bases.

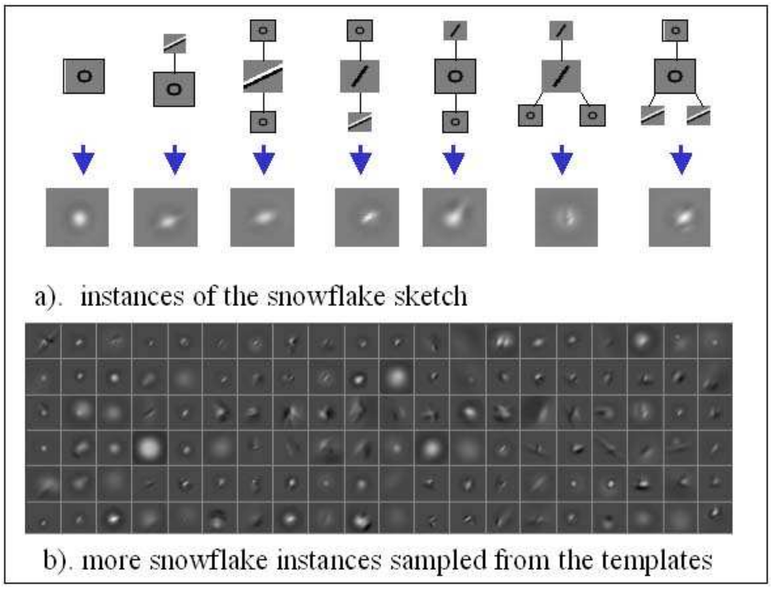

Fig. 3. The computed motion elements: snow flakes and random examples.

To summarize, we have a following generative model for an image $\mathbf{I}$, with dimensions reduced sequentially,

$$\mathbf{M} \xrightarrow{\Phi} \mathbf{B} \xrightarrow{\Psi} \mathbf{I}$$

2.3 Motion representation: dynamics, sources, and sinks.

Now we turn to the image sequence $\mathbf{I}[0, \tau]$. As Figure 1.c shows, a moveme π can be traced over a certain time interval $[t_b, t_e]$ and thus its trajectory is what we call a "cable". Typically in a moveme template, one base has relatively large coefficient and scale, such as the main body of the bird or snow flake, its trajectory forms the *core* of the cable. The core base is surrounded by a number of minor bases which account for the deformations. Due to self-rotation, the trajectories of these minor bases form the *coil* surrounding the cable core. In a coarse-to-fine computation, we can compute the trajectories of the cores first, and then add the coils sequentially. Thus we denote a cable by

$$C = C[t^b, t^e] = (\pi(t^b), \pi(t^b + 1), ..., \pi(t^e)). \tag{5}$$

In practice, the core of a moveme is relatively consistent through its life span, and the number of coil bases may change over time, due to self-occlusion etc. Since these bases are often minor, we assume the number of coil bases are fixed in a cable for simplicity.

We adopt a classic 2nd order Markov model which is sufficient for the dynamics of a moveme $C[t^b, t^e]$. In other words we fit the trajectory (the cable) $C[t^b, t^e]$ by regression. Such models are extensively used in tracking[8].

$$\begin{aligned} \pi(t) &= A\pi(t-1) + B \cdot \pi(t-2) + C + DN(0, \sigma_0^2) \quad t \in [t^b + 2, t^e] \\ \pi(t^b+1) &= A'\pi(t^b) + C' + D\omega \\ (\pi(t^b), t^b) &\sim P_B(\pi, \lambda), \ \ (\pi(t^e), t^e - t^b) \sim P_D(\pi, \lambda). \end{aligned}$$

One can simplify the equation in a canonical form expressed in equation (2). As $\pi(t)$ is a vector representing a number of bases including both the photometric (by base coefficients) and geometric information. The matrices A, B, C, D, A', C' capture the change of image appearances and the motion of the movemes, and these matrices are usually diagonal.

The first moveme $\pi(t^b)$ and its timing t^b follows a probability $P_B(\pi, \lambda)$ which we call the *birth map* for movemes. P_B specifies the "sources" of the movemes where the movemes are often originated. Similarly, the end of the trajectory $\pi(t^e)$ and its life span $t^e - t^b$ are governed by a *death map* $P_D(\pi, \lambda)$. P_D reveals the "sinks" in a lattice. As π is a long vector, P_B and P_D are high dimensional. But we are most interested in the location (x, y), other attributes in π can be modeled if necessary.

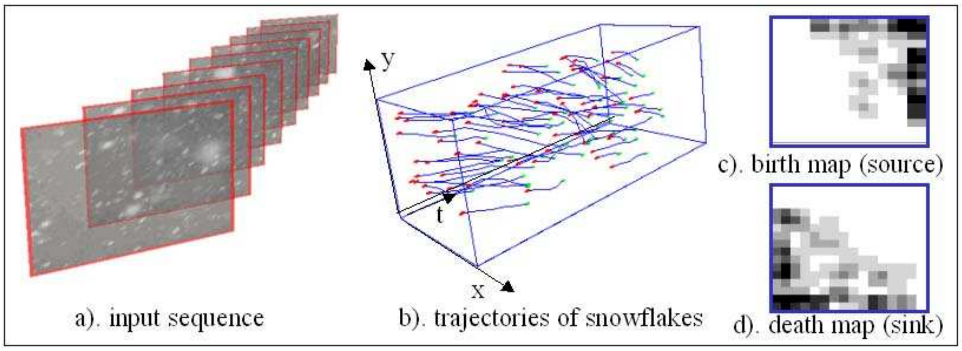

Fig. 4. The computed trajectories of snow flakes and the source and sink maps.

For example, Figure 4 displays the computed trajectories (4.b), birth (source) map (4.c), and death (sink) map (4.d) of the snowing sequence shown in Figure 2. The dark locations at the death/birth maps indicate high probabilities. Thus the algorithm "understands" that the snow flakes enter mostly from the upper-right corner and disappear around the lower-left corner. We sum over the other variables at each (x, y).

During the learning process, suppose we have computed K cables from a sequence $\mathbf{I}[0, \tau]$, $C_k[t_i^b, t_i^e], i = 1, 2, ..., K$, we represent P_B and P_D in a non-parametric form,

$$P_B(\pi,\lambda)=\frac{1}{K}\sum_{i=1}^{K}\delta(\pi-\pi_i(t_i^b),\lambda-t_i^b),\quad P_D(\pi,\lambda)=\frac{1}{K}\sum_{i=1}^{K}\delta(\pi-\pi_i(t_i^b),\lambda-(t_i^e-t_i^b))$$

where $\delta()$ is a Parzen window centered at 0. Then we can project P_B and P_D to the (x,y) dimensions as marginal probabilities.

In practice, the death and birth of movemes may be synchronized, for example, in the firework scene shown in Figure 8, a large number of movemes can come and go together. This requests the P_B and P_D to be joint probabilities for a large number of movemes.

To summarize, we denote all parameters in the motion equation above by,

$$\Gamma=(A,B,C,D,B,A',C',P_B,P_D)$$

3 Problem formulation

Given an observed image sequence $\mathbf{I}^{\mathrm{obs}}[0,\tau]$ as training data, we want to achieve two objectives.

1. Make inference about all the hidden (latent) variable which are represented by an unknown of K cables,
$$W[0.\tau]=(K,\{(t_i^b,t_i^e,C_i):\ \ [t_i^b,t_i^e]\subset[0,\tau],i=1,2,...,K\}\ ).$$
2. Compute the optimal fit for all parameters in the generative model $\Theta=(\Phi,\Gamma)$, with Φ being the set of deformable templates for the movements, and Γ governing the birth, death, and motion of the movemes.

The formulation is standard in statistics for learning a model with latent variables (missing data), that is, the maximum likelihood estimate (MLE),

$$\Theta^*=(\Phi^*,\Gamma^*)=\arg\max\log p(\mathbf{I}^{\mathrm{obs}}[0,\tau];\Theta). \qquad (6)$$

The likelihood is computed from the generative model with latent variables integrated (summed) out, For clarity of notation, we assume W are continuous variables.

$$p(\mathbf{I}^{\mathrm{obs}}[0,\tau];\Theta)=\int p(\mathbf{I}^{\mathrm{obs}}[0,\tau]|W[0,\tau];\ \Phi)p(W[0,\tau];\ \Gamma)dW.$$

Let $\mathbf{B}(t)=\{\mathbf{b}_{t,j},j=1,2,...,N(t)\}$ be the collection of all bases in the K movemes (cables) at time t, then we can re-express $W[0,\tau]$ as $(\mathbf{B}(0),...,\mathbf{B}(\tau))$, by equation (3), $p(\mathbf{I}[0,\tau]|W[0,\tau];\ \Phi)$ is the product Gaussians,

$$p(\mathbf{I}^{\mathrm{obs}}[0,\tau]|W[0,\tau];\ \Phi)=\prod_{t=0}^{\tau}G(\mathbf{I}^{\mathrm{obs}}(t)-\sum_{j=1}^{N(t)}\alpha_{t,j}\mathbf{b}_{t,j};\ \ \sigma_o^2),$$

as we assume iid Gaussian noise $G(0, \sigma_o^2)$ for $\mathbf{n}$.

Following the motion representation, we have $p(W[0,\tau];\ \Gamma)$ is also a product of Gaussians,

$$p(W[0,\tau];\ \Gamma) = \prod_{i=1}^{K} P_B(\pi(t_i^b), t^b) P_D(\pi(t_i^e), t_i^e) p(\pi(t_i^b+1)|\pi(t_i^b);\ A', C', D)$$
$$\times \prod_{t=t_i^b+2}^{t_i^e} p(\pi(t)|\pi(t-1), \pi(t-2);\ A, B, C, D).$$

4 Computation: learning and inference

To solve the MLE in eqn. (6), we set $\frac{\partial \log p(\mathbf{I}^{\text{obs}};\Theta)}{\partial \Theta} = 0$. This leads to

$$\int [\frac{\partial \log p(\mathbf{I}^{\text{obs}}|W;\Phi}{\partial \Phi} + \frac{\partial \log p(W;\Gamma)}{\partial \Gamma}] p(W|\mathbf{I};\ \Theta) dW. \tag{7}$$

Instead of using the classic EM algorithm, we adopt the stochastic gradient algorithm[6] which is capable of global optimal Θ. It iterates three steps with s indexing steps.

Step 1. Sampling $W^{\text{syn}}[0,\tau] \sim p(W|\mathbf{I}^{\text{obs}};\beta)$. This includes computing the bases, grouping bases into movemes, and tracking the movemes. The computation is realized by a data driven Markov chain Monte Carlo techniques, including the following reversible dynamics.

1). The death or birth of a motion trajectory Γ of length one.

2). Extending or shrinking a trajectory.

3). Mutate two nearby trajectories at a certain base.

4). Diffuse the location of a base in a trajectory.

Step 2. Updating the motion dynamics parameters Γ by regression,

$$\Gamma(s+1) = (1-\rho)\Gamma(s) + \rho \frac{\partial \log p(W^{\text{syn}}[0,\tau];\Gamma)}{\partial \Gamma}.$$

Step 3. Updating the moveme parameters Γ by clustering and grouping,

$$\Phi(s+1) = (1-\rho)\Phi(s) + \rho \frac{\partial \log p(\mathbf{I}^{\text{obs}}|W^{\text{syn}};\Phi)}{\partial \Phi}.$$

The algorithm is initialized by a stochastic version of match pursuit[9] for the base maps which is often very effective. We adopt a coarse-to-fine scheme. We track the core bases whose coefficients and scales are higher than a threshold, and learn the motion dynamics Γ. Then we lower the threshold to add the coil bases quickly following the learned trajectory.

Our method for tracking movemes is similar to the condensation algorithm[8], and it is different in two main aspects. Firstly, we have a full generative model of image, unlike the tracking model whose likelihood can only be evaluated relatively. Secondly, we are optimizing the whole trajectories and thus will trace back in time during the computation. Thus we don't have to remember a huge samples for each movemes. This, in combination with the generative model, saves large amount of time and memory.

For a typical sequence of 30 frames, the learning takes about 10-20 minutes in a Pentium IV PC, and the synthesis of sequence can be done in nearly real-time.

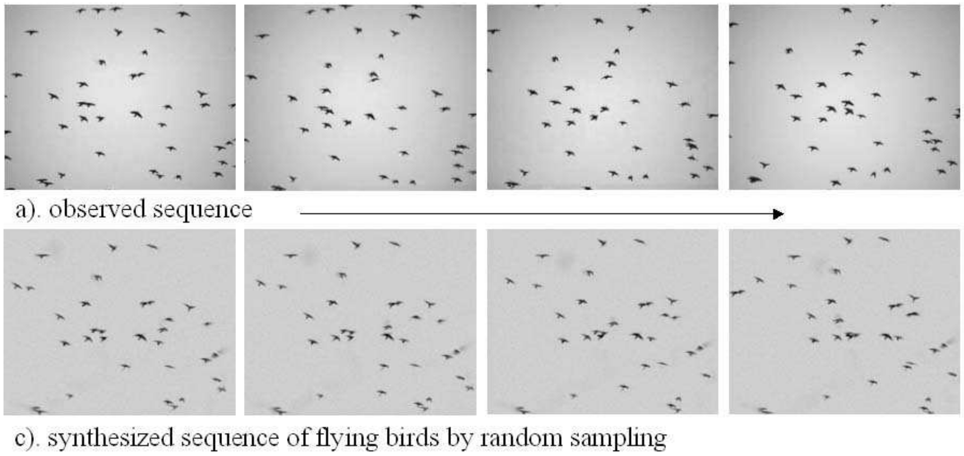

Fig. 5. Example of the bird sequence (see bird_obs.avi, bird_syn.avi).

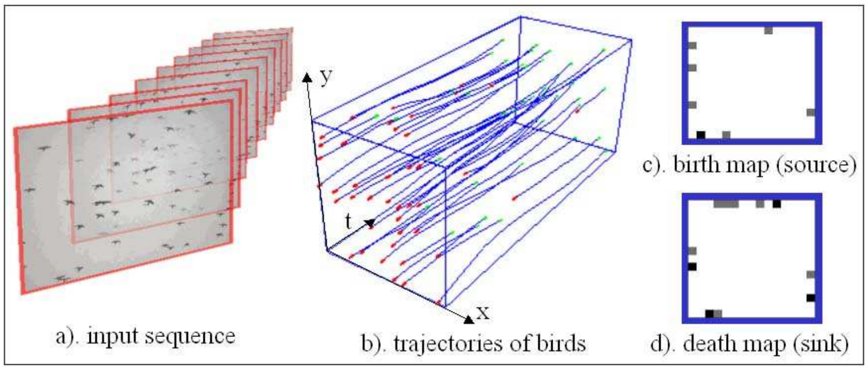

Fig. 6. The computed trajectories of flying birds and the source and sink maps.

5 Experiments

We report the results on four textured motion sequences.

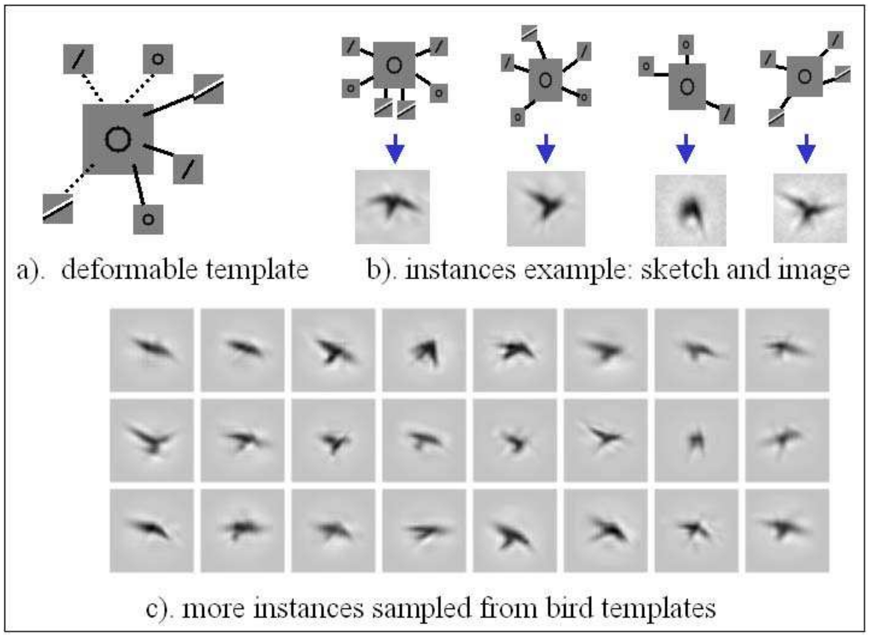

Fig. 7. The computed motion elements: flying birds and random examples

1. The snowing sequence. Fig. 2 shows the reconstruction of the snowing images by bases, and a synthesized sequence. For movie, see the observed and synthesized sequence at the attached avi files snow_obs.avi and snow_syn.avi respectively. The algorithm also computes the movemes (snow flake) templates, and random samples are shown in Fig. 3 The trajectories and source/sink maps are shown in Fig 4.

2. The flying bird sequence. Fig. 5.a and b show the observed and synthesize sequences. The animation can be seen at the attached avi files bird_obs.avi and bird_syn.avi. The trajectories and source/sink maps are shown in Fig.6. The birds enter and exit the picture from the image boundary. The maps are rather sparse due to the small number of birds in a short sequence. Fig.7 shows the deformable templates (a) where a core base is surrounded by a number of small coil bases. The dashed connection means the coil base may or may not appear all the time. A variety of templates and image instances of birds (movemes) are shown in (b) and (c).

3. The firework sequence. Fig. 8.a and b show the observed and synthesized sequences. See attached firework_obs.avi and firework_syn.avi for the movie. The trajectories and source/sink maps are shown in Fig.9. In the synthesis, we edit the birth map $P_B(\pi, \lambda)$ by changing its birth rate, assume a uniform distribution for the sources over then lattice. Thus the synthesis has more fireworks.

4. The waterfall sequence. Fig.10 shows the observed and synthesized sequences. See attached waterfall_obs.avi and waterfall_syn.avi for movies. The trajectories and source sinks are shown in Fig.11. Fig.12 shows 10 typical water drops in the waterfall what are a cluster of bases.

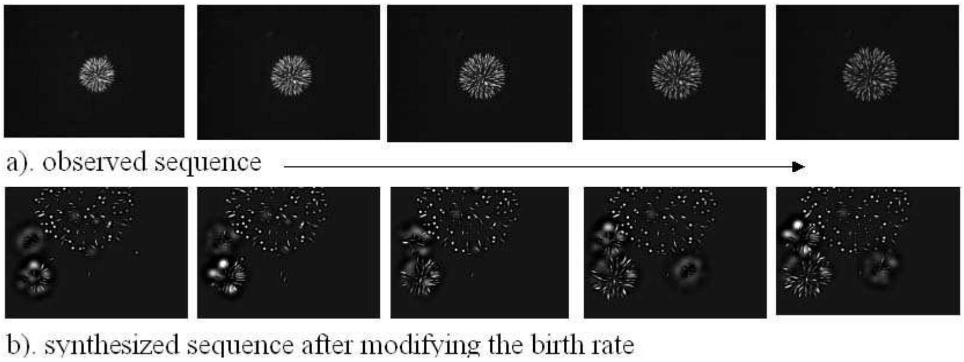

Fig. 8. Example of the firework sequence (see firework_obs.avi, firework_syn.avi)

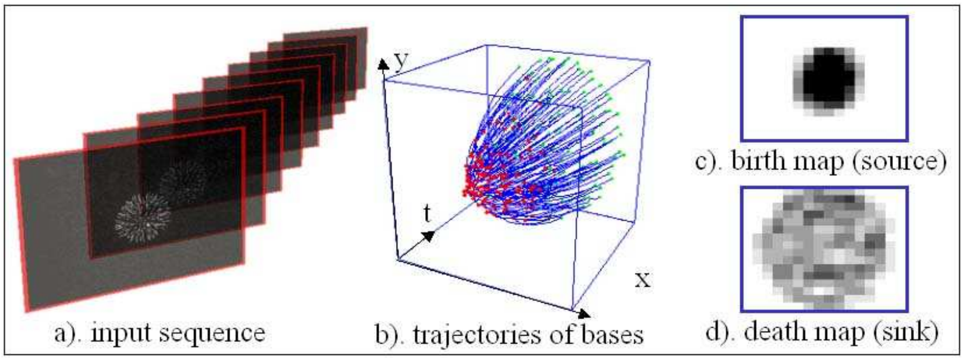

Fig. 9. The computed trajectories of fireworks and the source and sink maps.

6 Discussion and future work

The generative model in this paper is motivated by the graphics and vision criteria discussed in Section (1). It learns realistic motion patterns from real data, separates the motion dynamics with photometric and geometric styles, and thus achieve good controllability. For example, we can change the source/sink, alter the dynamics or geometry of movemes by group or by individuals. The representation is semantically meaningful for vision applications as well because the recovered trajectories etc. Needless to say that the generative description

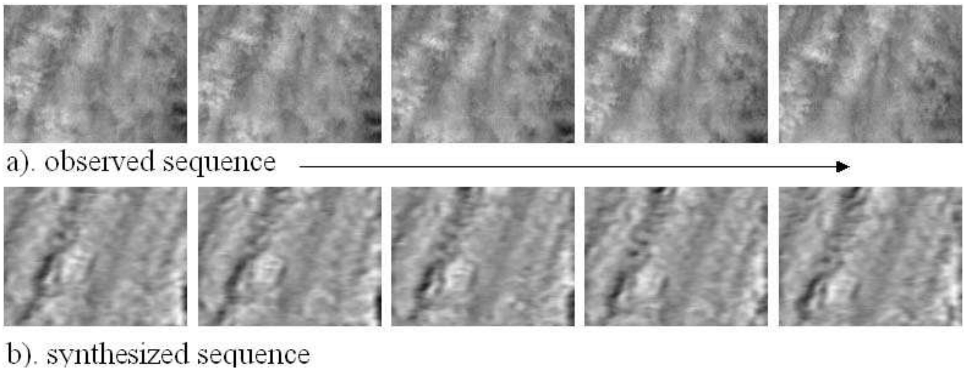

Fig. 10. Example of the waterfall sequence (see waterfall_obs.avi, waterfall_syn.avi)

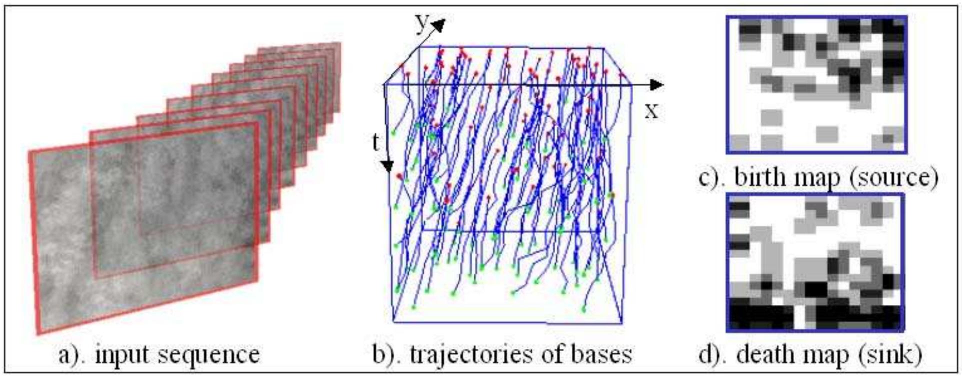

Fig. 11. The computed trajectories of waterfalls and the source and sink maps.

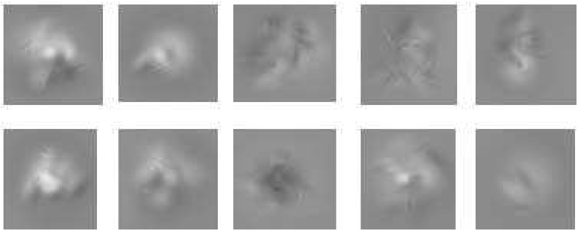

Fig. 12. The random examples of water drops.

$W[0, \tau]$ achieves tremendous often 10^2-fold compression in comparison to image $\mathbf{I}[0, \tau]$.

In future work we should extend the model in the following aspects.

1. We shall study the spatial interactions of the moving elements, bifurcation and merging of trajectories, and thus integrate good properties of the STAR model to account for lighting variation in motions such as water.
2. We shall model the state transition of moving elements, such as the flapping wings of a bird, and thus extend the motion dynamics model with more states.
3. We shall study the 3D positions of the moving elements (structure from motion).

References

1. Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. "Texture mixing and texture movie synthesis using statistical learning", *IEEE Trans on Visualization and Computer Graphics*, (to appear).
2. M. J. Black and A. Jepson, "Estimating optical flow in segmented images using variable-order parametric models with local deformations", *IEEE Trans on PAMI*, Vol. 18, No. 10, pp. 972-986, 1996.
3. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decompostion by basis pursuit", *Technical preprint*, Dept. of Stat., Stanford Univ., 1996.
4. A. D. Cliff and J.K. Ord, "Space-time modeling with an application to regional forecasting", *Trans. Inst. British Geographers*, 66:119-128, 1975.
5. D. S. Ebert and R. E. Parent, "Rendering and animation of gaseous phenomena by combining fast volume and scaleline A-buffer techniques", *SIGGRAPH*, 1990.
6. M. G. Gu, "A stochastic approximation algorithm with MCMC method for incomplete data estimation problems", *Preprint*, Dept. of Stat., McGill Univ. 1998.
7. D. Heeger and J. Bergen, "Pyramid-based texture analysis and synthesis", *Proc. of SIGGRAPH*, 1995.
8. M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density", *ECCV*, 1996.
9. S. Mallat and Z. Zhang, "Matching pursuit in a timef-requency dictionary", *IEEE trans. on Signal Processing*, vol.41, pp3397-3415, 1993.
10. Y. Meyer, *Wavelets: Algorithm and Applications*, SIAM, Philadelphia, 1993.
11. B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?", *Vision Research*, Vo.37, No. 23, pp3311-3325, 1997.
12. W. T. Reeves and R. Blau, "Approximate and probabilistic algorithms for shading and rendering structured particle systems", *Proc. of SIGGRAPH*, 1985.
13. P. Saisan, G. Doretto, Y.N. Wu, S. Soatto, "Dynamic Texture Recognition," *CVPR*, 2001.
14. A. Schodl, R. Szeliski, D. Salesin, and I. Essa, "Video texture", *Proc. of SIGGRAPH*, 2000.
15. S. Soatto, G. Doretto, and Y.N. Wu, "Synamic texture", *ICCV*, 2001.
16. M. O. Szummer and R. W. Picard, "Temporal texture modeling", *Proc. of Int'l Conf. on Image Processing*, Lausanne, Switzerland, 1996.
17. L.Y. Wei and M. Levoy, "Fast texture synthesis using tree structured vector quantization", *Proc. of SIGGRAPH*, 2000.
18. S. C. Zhu, Y.N. Wu, and D. B. Mumford, "Minimax entropy principle and Its Applications to Texture Modeling", *Neural Computation*, Vol. 9, Nov. 1997