Manifold Based Dynamic Texture Synthesis from Extremely Few Samples

Hongteng Xu[1,2], Hongyuan Zha[1], Mark A. Davenpot[2]
[1] CSE, Georgia Tech, [2] ECE, Georgia Tech
{hxu42,mdav}@gatech.edu, zha@cc.gatech.edu

Abstract

In this paper, we present a novel method to synthesize dynamic texture sequences from extremely few samples, e.g., merely two possibly disparate frames, leveraging both Markov Random Fields (MRFs) and manifold learning. Decomposing a textural image as a set of patches, we achieve dynamic texture synthesis by estimating sequences of temporal patches. We select candidates for each temporal patch from spatial patches based on MRFs and regard them as samples from a low-dimensional manifold. After mapping candidates to a low-dimensional latent space, we estimate the sequence of temporal patches by finding an optimal trajectory in the latent space. Guided by some key properties of trajectories of realistic temporal patches, we derive a curvature-based trajectory selection algorithm. In contrast to the methods based on MRFs or dynamic systems that rely on a large amount of samples, our method is able to deal with the case of extremely few samples and requires no training phase. We compare our method with the state of the art and show that our method not only exhibits superior performance on synthesizing textures but it also produces results with pleasing visual effects.

1. Introduction

Dynamic texture synthesis has many applications including graphics rendering [26, 19], video analysis [4], interpolation [14, 9, 27] and compression [1]. This problem has been widely studied from a number of perspectives in the past [10, 14, 9, 29, 26, 12, 19]. However, even the best existing methods still exhibit an obvious limitation — none of them can cope with the situation when there are only extremely few samples available, which characterizes several important practical applications. For example, we might capture two images of a dynamic scene at different times. How to synthesize a long realistic image sequence between these two frames is still a challenging open problem. In contrast to the problem of synthesizing a long sequence from a large number of frames [29, 19], the temporal information of the texture is completely lost in this case, and no external dataset can be readily used. Our goal is to fill this gap by developing algorithms that can synthesize long image sequences for natural dynamic textures merely from the first and last frames. The developed techniques will have potential applications in video interpolation, video hallucination and computer graphics.

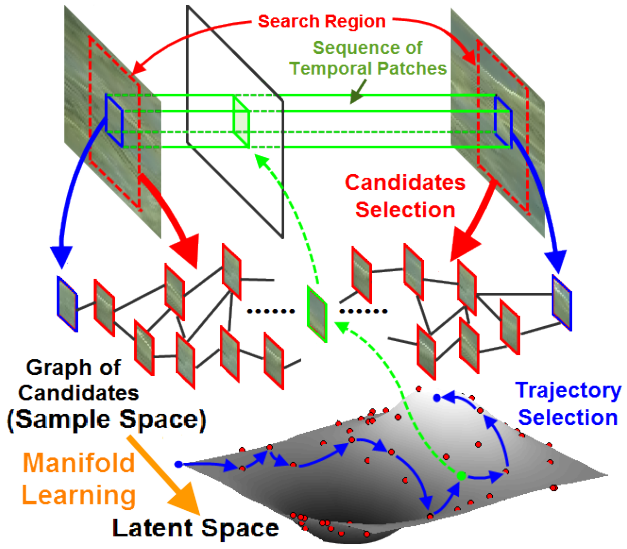

Figure 1. **Overview.** The synthesis process includes: 1) selecting candidates for temporal patches based on MRFs; 2) selecting trajectory for temporal patches based on manifold learning.

Illustrated in Fig. 1, we synthesize dynamic textures by estimating the sequences of in-between temporal patches. Our approach is driven by two key questions: how to find good estimates for temporal patches and how to decide their temporal order? To this end, we select candidates locally for the sequence of temporal patches from the given source images. To each sequence of temporal patches, its candidates are assumed to be on a low-dimensional manifold, and then the estimation and the permutation of the temporal patches is equal to finding an optimal trajectory in the latent space defined by the low-dimensional manifold.

We treat the selection of candidates as a patch matching task based on measuring the spatio-temporal similarity between patches. For synthesizing a long sequence of tem-

poral patches, we search candidates simultaneously along both the forward and backward directions of the sequence. The search regions are local regions of the two given source images. This strategy allows us to find the candidate similar to its spatial and temporal neighbors quickly. To avoid over-smoothing of the generated images, we select multiple candidates and update the set of samples (sampled from the search regions) for each temporal patch. After selecting the candidates, we construct a graph over the candidates which contains a trajectory corresponding to the estimated sequence of temporal patches. We analyze the properties of the trajectory in the latent space with the help of manifold learning [30]. The key observation reflects that the step length of trajectory is small while the direction of the trajectory is oscillatory. Guided by this two critical properties, we propose a curvature-based trajectory selection algorithm, which jointly optimizes the step length and the direction. Finally, candidates corresponding to the selected trajectory are stitched orderly in synthesized frames.

We evaluate our approach using the DynTex dataset [20] which contains multiple challenging dynamic textures from natural scenes. Given just two disparate frames, our method is still able to synthesize visually pleasing dynamic textures. We investigate the effect of parameters and find the optimal configuration of parameters based on quantitative analysis. We demonstrate that our method captures the dynamic of texture, and exhibits substantial improvement over the state of the art methods. Further, results on generating transitions between totally different images illustrate the broader applicability of our method in image morphing.

Our contribution is to synthesize dynamic texture from few samples without a time-consuming training phase using a manifold-based approach, which is not available for other existing methods. We creatively partition the synthesis process into the candidate selection phase and trajectory selection phase. Specifically, we propose a curvature-based trajectory selection algorithm for deciding optimal temporal patches and their order, which is a new strategy for dynamic texture synthesis.

Related Work. Synthesis methods relying on MRFs are popular in practice. They achieve texture synthesis by patch matching. In the early works [11, 10], the patch matching is based on the illuminance of patches. The following works [14, 26, 12, 1, 19] take the rotation, the scale, the gradient and the structural information of the patch into account. In the case of dynamic texture, the patch matching is extended to be spatio-temporal block matching. Differing from MRFs, dynamic system (DS) based methods view dynamic texture as a dynamic system controlled by low-dimensional latent variables. These methods firstly estimate latent variables by linear [9, 29] or nonlinear [7, 17] dimension reduction algorithms. Then they establish the status equation of latent variables and the mapping from latent variables to samples by parametric [9, 29, 21] or non-parametric models [17]. As a result, given latent variables, textures can be synthesized. Although these methods get encouraging synthesis results, none of them can cope with the case of few samples. For MRFs based methods, even the state of the art needs at least several consecutive frames [19] or external dataset [26] for synthesis. On the other hand, the feasibility of DS based methods [9, 29, 21, 17] is more questionable. These methods generally need a sequence containing hundreds even thousands of frames for training the model. What is worse, for different dynamic textures we have to train their DS models respectively.

Image morphing methods are also used for synthesis problems. The methods in [24, 2, 23, 5] achieve synthesis by bi-directional similarity (BDS) based patch matching — the distance between the patch of source image and the synthesis result and the distance between the patch of synthesis result and the source image are minimized iteratively. Although these methods can synthesize a sequence from two images, they often lead to over-smoothness in the case of dynamic texture. Methods for computing diffeomorphic deformations between images are proposed in [3, 8, 22, 28]. Unfortunately, transitions between textures are not diffeomorphic in general, and can not be synthesized by [3, 8, 22].

2. The Analysis of Temporal Patches

Given two texture images I_1 and I_T, we want to synthesize the in-between $\{I_t\}_{t=2}^{T-1}$. An arbitrary patch in I_t, $t = 1, ..., T$, is denoted as $\mathbf{p}_t(x) \in \mathbb{R}^{s^2}$, where x is the coordinate in the image and s is the size of patch. Our goal is to estimate temporal patches $\{\mathbf{p}_t(x)\}_{t=2}^{T-1}$ for each x.

2.1. Matching Temporal Patches Locally

Because dynamic textures have local self-similarity (the basic assumption of MRFs), we can find a good estimate of $\mathbf{p}_t(x)$ locally in the source images. We did a verification experiment to prove the feasibility of matching temporal patches locally. For each x, define search regions $R_1(x)$ and $R_T(x)$ in I_1 and I_T respectively. Given $\mathbf{p}_t(x)$, we searched its matching patch in $R_1(x)$ and $R_T(x)$, and then calculated the energy of the matching error. The relative matching error is measured by the proportion between the energy of the matching error and the energy of $\mathbf{p}_t(x)$, and the average relative error is the average of all relative matching errors. We measured average relative errors for different configurations of the patch size[1] and the sequence length using the DynTex dataset [20]. The results are shown in Table 1. Even if we choose large patch size (e.g., $s = 13$), the average relative error is still smaller than 11%. In other words, we are able to find a patch in $R_1(x)$ or $R_T(x)$ matching well with $\mathbf{p}_t(x)$.

[1] In our work, we choose the radius of search region is the twice of the radius of patch.

Table 1. Average Relative Errors (%) of Patch Matching

	$s=5$	$s=7$	$s=9$	$s=11$	$s=13$
$T=5$	04.53	05.73	06.52	07.59	08.01
$T=10$	05.21	06.93	07.35	08.68	09.69
$T=20$	05.97	07.50	08.15	09.63	10.29
$T=30$	06.36	07.61	08.66	09.94	10.56

2.2. The Dynamics of Temporal Patch Sequences

After sampling $R_1(x)$ and $R_T(x)$ with overlap, we obtain a set of patches $\mathbf{S}$ containing the estimate of $\{\mathbf{p}_t(x)\}_{t=2}^{T-1}$. Now, two problems are still left to be tackled: 1) not all patches in $\mathbf{S}$ are useful for synthesizing $\{\mathbf{p}_t(x)\}_{t=2}^{T-1}$; 2) even if we find the useful patches, the temporal order of them is still unknown. In [7, 15], it is shown that textural patches are approximated well by a low-dimensional manifold, and we model patches of textures based on a manifold whose samples include $\mathbf{p}_1(x)$, $\mathbf{p}_T(x)$ and $\mathbf{S}$. According to Section 2.1, $\{\mathbf{p}_t(x)\}_{t=2}^{T-1}$ can be estimated by a trajectory on the manifold connecting $\mathbf{p}_1(x)$ with $\mathbf{p}_T(x)$. So, the two problems above are equivalent to finding an optimal trajectory on the manifold.

The definition of optimality in our work is based on the assumption that an optimal trajectory should have the properties of real trajectories corresponding to the sequence of temporal patches. After analyzing the DynTex dataset, we conclude that the step length of real trajectories is small and its direction is oscillatory in most situations. To illustrate these two properties, Fig. 2 presents several typical examples. Using the Local-Tangent-Space-Alignment (LTSA) algorithm [30], we visualize the temporal patches in 2D latent space and mark the trajectories by a series of arrows. Although the four dynamic textures in Fig. 2(a) are very different from each other, their trajectories of temporal patches in the latent space share several general properties. In Fig. 2(b), the patches having different structures are clustered separately in the latent space. This phenomenon shows that the direction of the trajectories (the angles of arrows in the plane) to be quite oscillatory and the step length of trajectory to be small in most situation. In Fig. 2(c), the directions of the arrows is plotted as blue curve. We can find that the directions of trajectories changed frequently. On the other hand, we plot the normalized step length of trajectory as stems. The green stems represent the steps smaller than 1% of the total length of the trajectory while the red ones represent the steps larger than 1%. We can find that most of steps are small while only few large steps correspond to the "jumps" from a cluster to another.

3. The Proposed Method

According to the analysis in the former section, the properties of temporal patches can be summarized as follows: 1) temporal patches can be estimated by spatial patches locally; 2) the trajectory of temporal patches in the latent space has small step length and oscillatory directions. Taking advantages of these two properties, we propose a manifold based algorithm for dynamic texture synthesis.

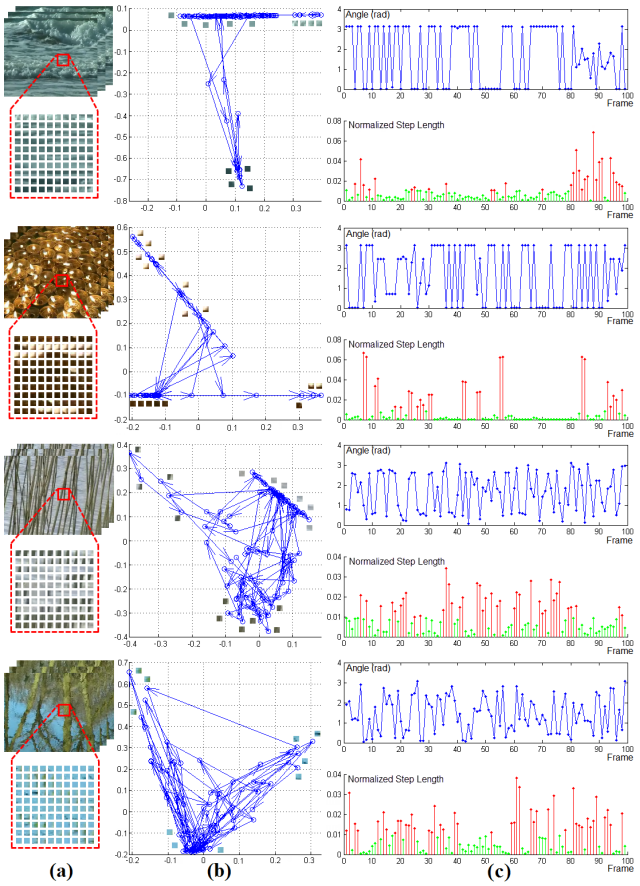

Figure 2. (a) The sequences of temporal patches are shown as zigzag patterns, each of which is obtained from 100 frames of the corresponding dynamic textures. (b) The 2D visualization of temporal patches is shown, in which arrows mark the trajectory of the latent variables. Some latent variables are labeled by the corresponding patches. (c) The direction and the normalized step length of the trajectory are plotted.

3.1. Candidate Selection

Given I_1 and I_T, we are always able to find a coordinate $\hat{x}$ where the residual $\|\mathbf{p}_1(\hat{x}) - \mathbf{p}_T(\hat{x})\|$ is minimum. The initialization of our method is synthesizing $\{\mathbf{p}_t(\hat{x})\}_{t=2}^{T-1}$ by following linear interpolation,

$$\mathbf{p}_t(\hat{x}) = [(T-t)\mathbf{p}_1(\hat{x}) + (t-1)\mathbf{p}_T(\hat{x})]/(T-1).$$

Given $\{\mathbf{p}_t(\hat{x})\}_{t=2}^{T-1}$, the overlapped part between $\mathbf{p}_t(\hat{x})$ and its neighbor $\mathbf{p}_t(x)$, denoted as $\mathbf{q}_t(x) \in \mathbb{R}^m$, is known. Then, for each $\mathbf{p}_t(x)$ we select K candidates from $\mathbf{S}$ according to following spatio-temporal similarity. **Spatial similarity:** define a matrix $\mathbf{M} \in \mathbb{R}^{m \times s^2}$ indicating the

elements of $\mathbf{p}_t(x)$ corresponding to $\mathbf{q}_t(x)$. If $\mathbf{s} \in \mathbf{S}$ is a candidate of $\mathbf{p}_t(x)$, $\mathbf{Ms}$ should be similar to $\mathbf{q}_t(x)$. **Temporal similarity:** the candidates of $\mathbf{p}_t(x)$ should be similar to the candidates of $\mathbf{p}_{t-1}(x)$ or $\mathbf{p}_{t+1}(x)$. We propose our candidate selection algorithm with details in the following table. For the convenience of representation, we omit the coordinate x in the algorithm and following analysis.

Algorithm 1: Candidate Selection
Inputs: $\mathbf{p}_1$, $\mathbf{p}_T$, $\{\mathbf{q}_t\}_{t=2}^{T-1}$, $\mathbf{S}$.
Outputs: candidates $\{\mathbf{c}_{t,k}\}$.
Create two sets $\mathbf{S}_1 = \mathbf{S}$, $\mathbf{S}_T = \mathbf{S}$.
For $k = 1 : K$
$\mathbf{c}_{2,k} = \min_{\mathbf{s}\in\mathbf{S}_1}\{\frac{s^2}{m}\|\mathbf{Ms} - \mathbf{q}_2\|_2^2 + \|\mathbf{s} - \mathbf{p}_1\|_2^2\}$ (1)
$\mathbf{c}_{T-1,k} = \min_{\mathbf{s}\in\mathbf{S}_T}\{\frac{s^2}{m}\|\mathbf{Ms} - \mathbf{q}_{T-1}\|_2^2$ (2)
$+\|\mathbf{s} - \mathbf{p}_T\|_2^2\}$,
$\mathbf{S}_1 = \mathbf{S}_1 \setminus \mathbf{c}_{t,k}$, $\mathbf{S}_T = \mathbf{S}_T \setminus \mathbf{c}_{T-t+1,k}$.
End
For $t = 3 : T/2$
For $k = 1 : K$
$\mathbf{c}_{t,k} = \min_{\mathbf{s}\in\mathbf{S}_1}\{\frac{s^2}{m}\|\mathbf{Ms} - \mathbf{q}_t\|_2^2$ (3)
$+\frac{T-t}{K(T-1)}\sum_{k=1}^{K}\|\mathbf{s} - \mathbf{c}_{t-1,k}\|_2^2$
$+\frac{t-1}{K(T-1)}\sum_{k=1}^{K}\|\mathbf{s} - \mathbf{c}_{T-t+2,k}\|_2^2\}$,
$\mathbf{c}_{T-t+1,k} = \min_{\mathbf{s}\in\mathbf{S}_T}\{\frac{s^2}{m}\|\mathbf{Ms} - \mathbf{q}_{T-t+1}\|_2^2$ (4)
$+\frac{T-t}{K(T-1)}\sum_{k=1}^{K}\|\mathbf{s} - \mathbf{c}_{t-1,k}\|_2^2$
$+\frac{t-1}{K(T-1)}\sum_{k=1}^{K}\|\mathbf{s} - \mathbf{c}_{T-t+2,k}\|_2^2\}$,
$\mathbf{S}_1 = \mathbf{S}_1 \setminus \mathbf{c}_{t,k}$, $\mathbf{S}_T = \mathbf{S}_T \setminus \mathbf{c}_{T-t+1,k}$.
End
If $\#\{\mathbf{S}_1\} < K$, $\mathbf{S}_1 = \mathbf{S}$.
If $\#\{\mathbf{S}_T\} < K$, $\mathbf{S}_T = \mathbf{S}$.
End

Here, we use $\mathbf{c}_{t,k}$ to represent the kth candidate of $\mathbf{p}_t$. $\mathbf{S}_1$ and $\mathbf{S}_T$ are sample sets for selecting candidates along forward and backward directions respectively. We find candidates for $\mathbf{p}_2$ and $\mathbf{p}_{T-1}$ by Eq. (1) and Eq. (2). In each equation, the first term corresponds to the residual in the overlapped region, and the second term corresponds to the residual between the candidate and the source patch. These two terms measure the spatial and the temporal similarity respectively, whose importance is leveraged by the weight $\frac{s^2}{m}$. The candidates of $\{\mathbf{p}_t\}_3^{T-2}$ are selected by Eq. (3) and Eq. (4). The spatial similarity is still measured by the residual in the overlapped region, while the temporal similarity is measured by the residual between the candidates and existing candidates selected in the former iteration. The weight of the residual between candidates is proportional to the temporal distance between them. In summary, the proposed algorithm takes advantages of spatio-temporal similarity to select candidates for the sequence of temporal patches.

It should be mentioned that we update sample sets $\mathbf{S}_1$ and $\mathbf{S}_T$ during the iteration. Samples having been selected as candidates will be removed from set so that they will not be used repeatedly. Only when the size of set is too small to find candidates do we update the set by original $\mathbf{S}$. This update strategy preserves the dynamics of temporal patches.

3.2. Trajectory Selection

3.2.1 Manifold Learning for Candidates

According to the candidate selection algorithm above, each candidate is connected with its forward and backward neighbors, which can be described by the graph $\mathcal{G}$ in Fig. 3. The distance between candidates is measured by a distance matrix $\mathbf{D}$, whose element is

$$D(i,j) = \begin{cases} \|\mathbf{c}_i - \mathbf{c}_j\|_2, & \mathbf{c}_i \text{ links to } \mathbf{c}_j, \\ 0, & \text{otherwise}, \end{cases} \quad i,j \in \{(t,k)\}.$$

For the convenience, we default to use i or j to represent the tuple (t,k) for candidates in the following content.

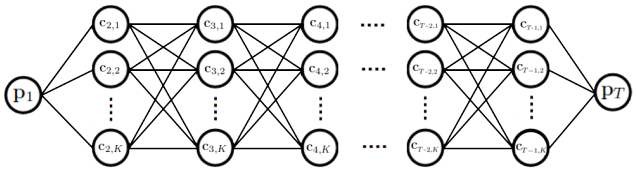

Figure 3. The graph $\mathcal{G}$ of candidates.

According to Section 2.2, $\{\mathbf{p}_t\}_{t=2}^{T}$ can be synthesized by finding a trajectory from $\mathbf{p}_1$ to $\mathbf{p}_T$ in graph $\mathcal{G}$. Facing up to K^{T-2} potential trajectories, we need a criterion for trajectory selection. Although selecting the shortest path according to $\mathbf{D}$ seems to be a reasonable choice, it results in an over-smoothed trajectory, which is insufficient to reflect the oscillation of trajectory. So, we derive a trajectory selection algorithm based on manifold learning.

Suppose that candidates are samples on a manifold, the relationship between $\mathbf{c}_i$ and its latent variable $\mathbf{y}_i \in \mathbb{R}^d$ ($d \ll s^2$) is described as follows,

$$\mathbf{c}_i = f(\mathbf{y}_i) + e,$$

where f is a nonlinear bijection mapping and e is sample noise. Denote the N neighbors of $\mathbf{c}_i$ as $\mathbf{C}_i$, the corresponding N neighbors of $\mathbf{y}_i$ as $\mathbf{Y}_i$. We can get the linear approximation of f at $\mathbf{c}_i$ by following Taylor expansion.

$$\mathbf{C}_i - \mathbf{c}_i\mathbf{1}^T = \mathbf{J}_i(\mathbf{Y}_i - \mathbf{y}_i\mathbf{1}^T) + o(\|\mathbf{Y}_i - \mathbf{y}_i\mathbf{1}^T\|_F).$$

Here $\mathbf{J}_i \in \mathbb{R}^{s^2\times d}$ is the Jacobian matrix of f at $\mathbf{y}_i$, whose columns span the local tangent space on the manifold. $\mathbf{1}$ is a vector whose elements are all equal to 1.

Assume the noise e is with i.i.d. homogeneous Gaussian distribution, $\mathbf{J}_i$ can be estimated by minimizing following objective function,

$$\|(\widetilde{\mathbf{C}}_i - \mathbf{J}_i\widetilde{\mathbf{Y}}_i)\mathbf{W}\|_F, \tag{5}$$

where $\widetilde{\mathbf{C}}_i = \mathbf{C}_i - \mathbf{c}_i\mathbf{1}^T$ and $\widetilde{\mathbf{Y}}_i = \mathbf{Y}_i - \mathbf{y}_i\mathbf{1}^T$. $\mathbf{W} \in \mathbb{R}^{N\times N}$ is a diagonal matrix indicating the importance to minimizing the error. We choose the element of $\mathbf{W}$ as follows,

$$w_n = \exp(-\|\mathbf{c}_i^n - \mathbf{c}_i\|_2^2/\sigma^2), \tag{6}$$

where $\mathbf{c}_i^n$ is the nth neighbor of $\mathbf{c}_i$. According to [30], Eq. (5) can be solved efficiently as follows,

$$\begin{aligned} &\widehat{\mathbf{J}}_i = \arg\max_{\mathbf{J}_i} \|\mathbf{J}_i^T\widetilde{\mathbf{C}}_i\mathbf{W}\mathbf{W}^T\widetilde{\mathbf{C}}_i^T\mathbf{J}_i\|_* \\ &\text{s.t.} \quad \mathbf{J}_i^T\mathbf{J}_i = \mathbf{I}. \end{aligned} \tag{7}$$

$\|\cdot\|_*$ is the nuclear norm of matrix, which calculates the sum of singular values. The solution is the largest d eigenvectors of the matrix $\widetilde{\mathbf{C}}_i\mathbf{W}\mathbf{W}^T\widetilde{\mathbf{C}}_i^T$.

For adjacent candidates $\mathbf{c}_i$ and $\mathbf{c}_j$, the change of their tangential directions can be measured by the principal angle between their Jacobian matrices. Letting $\mathbf{J}_i = \mathbf{Q}_i\mathbf{R}_i$ and $\mathbf{J}_j = \mathbf{Q}_j\mathbf{R}_j$ be the QR-factorization, the principal angle, denoted as $\alpha(i,j)$, is calculated as the arc cosine of the largest singular value of $\mathbf{Q}_i^T\mathbf{Q}_j$. So, we can further define an angle matrix $\mathbf{A}$, whose element is

$$A(i,j) = \begin{cases} \alpha(i,j), & \mathbf{c}_i \text{ links to } \mathbf{c}_j, \\ 0, & \text{otherwise}, \end{cases} \quad i,j \in \{(t,k)\}.$$

3.2.2 Curvature Based Trajectory Selection

Denote a trajectory of candidates as $\mathcal{P} = \{i_1, ..., i_T\}$, where i_t represents the index of candidate on the trajectory. The information of its step length and direction is in the distance matrix $\mathbf{D}$[2] and the angle matrix $\mathbf{A}$ respectively. Define the length and the integrated principal angle of $\mathcal{P}$ as follows,

$$\begin{aligned} \text{Angle}(\mathcal{P}) &= \sum\nolimits_{t=2}^{T} |A(i_{t-1}, i_t)|, \\ \text{Length}(\mathcal{P}) &= \sum\nolimits_{t=2}^{T} D(i_{t-1}, i_t). \end{aligned}$$

We already know that an optimal trajectory should have small steps and its direction should change frequently. This requires us to minimize Length($\mathcal{P}$) and maximize Angle($\mathcal{P}$) simultaneously. In our work, we achieve this aim by minimizing following objective function,

$$\sum\nolimits_{t=2}^{T} \frac{D(i_{t-1}, i_t)}{|A(i_{t-1}, i_t)|^\beta + \epsilon}, \tag{8}$$

where ϵ avoids the case of zero denominator. β is a non-negative parameter achieving a trade-off between the influence of step length and that of principal angle. Define $\frac{D(i,j)}{|A(i,j)|^\beta+\epsilon}$ as the weight of edge in the graph $\mathcal{G}$, then minimizing Eq. (8) equals to finding the shortest path from $\mathbf{p}_1$ to $\mathbf{p}_T$ in the graph, which can be solved efficiently by Dijkstra's algorithm [6]. It is easy to find that minimizing Eq. (8) tends to maximize principal angle (denominator) and minimize step length (numerator) jointly, which preserves the properties of trajectory. In summary, the details of our trajectory selection algorithm is proposed as follows.

Algorithm 2: Trajectory Selection
Inputs: $\mathbf{p}_1$, $\mathbf{p}_T$, $\mathbf{c}_{t,k}$, $t = 2, .., T-1$, $k = 1, .., K$.
1. Create the graph $\mathcal{G}$ as Fig. 3 shows:
   $\mathbf{p}_1$ is connected with $\{\mathbf{c}_{2,k}\}_{k=1}^K$;
   $\mathbf{p}_T$ is connected with $\{\mathbf{c}_{T-1,k}\}_{k=1}^K$;
   $\mathbf{c}_{t,k}$ is connected with $\{\mathbf{c}_{t-1,k}\}_{k=1}^K$ and $\{\mathbf{c}_{t+1,k}\}_{k=1}^K$.
2. According to the connections of candidates, calculate distance matrix $\mathbf{D}$.
3. To each $\mathbf{c}_{t,k}$, calculate Jacobian Matrix $\mathbf{J}_{t,k}$ by solving (7).
4. According to $J_{t,k}$, calculate angle matrix $\mathbf{A}$.
5. Calculate the graph matrix $\mathbf{G}$. The element is $G(i,j) = \frac{D(i,j)}{|A(i,j)|^\beta+\epsilon}$, $i,j \in \{(t,k)\}$, which defines the weight of edge in graph $\mathcal{G}$.
6. Find the shortest trajectory from $\mathbf{p}_1$ to $\mathbf{p}_T$ by Dijkstra Algorithm.
7. The candidates on the trajectory are matched patches.

It should be noted that Eq. (8) has following physical meaning. According to [13], the integrated principal curvature of $\mathcal{P}$ is defined as follows,

$$\text{Curve}(\mathcal{P}) = \sum_{t=2}^{T} \text{Curve}(i_{t-1}, i_t) = \sum_{t=2}^{T} \frac{|A(i_{t-1}, i_t)|}{D(i_{t-1}, i_t)}. \tag{9}$$

It is obvious that maximizing the integrated principal curvature also ensures the step length (denominator) to be small and the principal angle (numerator) to be large. However, maximizing Curve($\mathcal{P}$) directly requires us to find the longest path from $\mathbf{p}_1$ to $\mathbf{p}_T$ in the graph, which is an NP-hard problem. Compared with Eq. (9), the $\frac{D(i_{t-1},i_t)}{|A(i_{t-1},i_t)|^\beta+\epsilon}$ in Eq. (8) can be viewed as a generalized reciprocal of curvature Curve(i_{t-1}, i_t). From this view, our trajectory selection algorithm minimizes the integrated reciprocal of curvature, which is actually an alternative way to maximize the principal curvature of trajectory.

3.3. Patch Fitting

After finding the optimal trajectory, candidates on the trajectory are regarded as good estimates of temporal the patches. We stitch them together in order to produce the synthesized frames. The overlapped region is processed by graph-cuts [14] to achieve seamless fitting. Repeatedly applying candidate selection algorithm and trajectory selection algorithm for all sequences of temporal patches, we finally obtain the synthesis results.

[2]Because of the local isometry assumption, the distance between candidates is proportional to the distance between latent variables.

Table 2. Comparison for Various Configurations of Parameters (PSNR (dB))

	wave			fur			rain			ripple		
	K=2	K=5	K=10	K=2	K=5	K=10	K=2	K=5	K=10	K=2	K=5	K=10
$\beta = 0$	17.70	18.01	17.31	26.92	**27.03**	26.68	23.78	23.85	23.74	18.48	18.78	18.49
$\beta = 0.5$	19.06	19.55	18.71	26.86	26.85	26.83	23.89	**23.93**	23.81	19.04	19.04	19.04
$\beta = 1$	19.52	**19.79**	19.10	26.69	26.83	26.70	23.85	**23.93**	23.83	19.06	**19.42**	18.91
$\beta = 2$	19.16	19.65	18.80	26.69	26.80	26.65	23.79	23.81	23.76	19.09	19.28	18.89
$\beta = 15$	19.00	19.51	18.52	26.70	26.92	26.59	23.63	23.77	23.60	19.02	19.30	18.88

Table 3. Comparison for Various Configurations of Parameters (SSIM)

	wave			fur			rain			ripple		
	K=2	K=5	K=10	K=2	K=5	K=10	K=2	K=5	K=10	K=2	K=5	K=10
$\beta = 0$	0.724	0.748	0.700	0.964	**0.965**	0.962	0.951	**0.952**	0.950	0.761	0.777	0.774
$\beta = 0.5$	0.804	0.827	0.786	0.963	0.963	0.963	0.951	**0.952**	0.951	0.791	0.789	0.793
$\beta = 1$	0.826	**0.838**	0.805	0.962	0.963	0.962	0.951	**0.952**	0.951	0.785	**0.803**	0.777
$\beta = 2$	0.810	0.832	0.792	0.962	0.962	0.961	0.950	0.950	0.950	0.788	0.797	0.778
$\beta = 15$	0.803	0.827	0.777	0.962	0.963	0.961	0.948	0.950	0.948	0.788	0.802	0.781

4. Experimental Results

4.1. Implementation Details

We apply our method to the DynTex dataset [20] in which the resolution of the video frame is 352×288. The testing set contains many natural dynamic textures including ripples, waves, raining scenes, bushes, and furs.[3] To each dynamic texture, we synthesize 30 frames using only the first and the last frame of the video clip. In the experiment, we set the relevant parameters empirically as follows: the dimension of latent space of candidates is $d = 3$; the σ in Eq. (7) is 0.1 while the ϵ in Eq. (8) is 0.01. A significant parameter for texture synthesis is the size of the patch, which impose constraints on other parameters, e.g., the size of search window and the moving step of patch during synthesis (the width of the overlapped region is equal to the size of the patch minus the moving step length). In [11, 14, 15], the size of the patch is set to be quite large for preserving the structural information of the texture. Similar to those works, the size of the patch is $s = 13$ in our work, leading to the size of corresponding search region being 27, and the moving step of the patch being 8, respectively.

4.2. Dynamic Texture Synthesis

Besides the size of the patch, another two critical parameters in our method are: 1) the number of candidates, K; 2) the parameter β in Eq. (8). For analyzing their influence on synthesis results, we select $K = 2, 5, 10$, $\beta = 0, 0.5, 1, 2, 15$ respectively, and test our method with different configurations. The quantitative comparison for various configurations is based on two objective measurements. Given the ground truth of dynamic textures, we measure the PSNR/SSIM [25] of synthesis results corresponding to different configurations, which are listed in Table 2 and Table 3 respectively.

K decides the degrees of freedom for candidate selection. If K is too large, each temporal patch will have many candidates. In such a situation, temporal patches will risk being estimated by unsuitable candidates so that the dynamics of the synthesis result will be out-of-control. On the contrary, if K is too small, there will be too few candidates to learn the manifold. In our experiment, setting $K = 5$ achieves the best PSNR/SSIM in most situations.

The parameter β controls the contribution of the principal angle in Eq. (8). The importance of principal angle will be enhanced with the increase of β. When $\beta \gg 1$, the influence of numerator (the length of trajectory) in Eq. (8) can be ignored, so minimizing Eq. (8) is equal to maximizing the integrated principal angle. On the contrary, if we decrease β, the importance of principal angle will be weakened. When $\beta = 0$, the denominator in Eq. (8) is just $1 + \epsilon$, so minimizing Eq. (8) is equal to minimizing the length of trajectory. Experimental results in Table 2 and Table 3 demonstrate that $\beta = 1$ is a reasonable trade-off between the two extreme cases discussed above. In such a situation, Eq. (8) is similar to $\sum_{t=2}^{T} \frac{1}{\text{Curve}(i_{t-1}, i_t)}$, which maximizes integrated principal curvature indirectly. Besides the four dynamic textures in Table 2 and Table 3, we test our method with different β's ($K = 5$) on the DynTex dataset and plot average PSNR/SSIM curves in Fig. 4(a). The PSNR/SSIM curves corresponding to $\beta = 1$ are higher than those given by other configurations indeed. In summary, according to Table 2 and Table 3 we set $\beta = 1$, $K = 5$ for our method.

Fig. 4(b) further illustrates the difference among various configurations. The ground truth of ripple sequence shown in Fig. 4(b), in which there are approximately 3 periods of fluctuation. We synthesize the sequence by various configurations and list enlarged temporal patches respectively. Tak-

[3] We strongly recommend viewing the video of synthesis results in the supplemental file for clear visual effects.

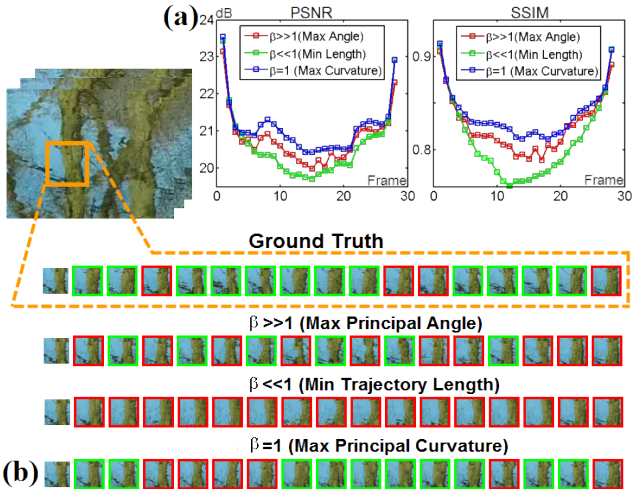

Figure 4. (a) The average PSNR curves for different β. (b) Comparison for various methods on textures having periodic pattern.

ing the energy of the first patch as the reference, we mark the following patches having higher energy by red squares, the rest by green squares. The fluctuation corresponding to $\beta \gg 1$ looks rather jittery. This is because the algorithm tends to merely maximize the integrated principal angle. On the contrary, the result corresponding to $\beta \ll 1$ does not have a periodic pattern anymore because minimizing trajectory length is equivalent to minimizing the Euclidean distance between adjacent temporal patches. In such a situation, the selected trajectory would be over-smoothed and the oscillation is eliminated. Only the method with the proposed configuration achieves suitable fluctuation. More comparison results can be found in the supplemental file.

4.3. Comparisons with Image Morphing Methods

In the case of merely two images, both MRFs based methods [11, 10, 14, 26, 19] and DS based methods [9, 29, 17] do not work well. As a result, only image morphing methods [23, 5, 3, 22] are available as our competitors. To demonstrate the superiority of our method, we compare our method with the state of the art BDS based method [5] and classical diffeomorphism based method LDDMM (Large-Deformation-Diffeomorphic-Metric-Mapping) [3].

Fig. 5 and the supplemental file show that both BDS based morphing [5] and LDDMM [3] lead to over-smoothed synthesis results for dynamic textures. The reason for this phenomenon is that their smoothness assumptions do not adequately capture the nature of dynamic texture. The bi-directional similarity in [5] is a measurement based on Euclidean distance, which aims to achieve a smooth morphing process by minimizing the distance between the target image and the source image. LDDMM [3] assumes that there exists a diffeomorphic path between the target image and the source image. Generally, this type of smoothness assumptions are reasonable for many images like cartoons, medical images and synthetic graphics. However, as we have shown in Section 2.2, the dynamic process of natural textures is complicated with oscillatory patterns, defeating priors attempts based on smoothness assumptions [5, 3].

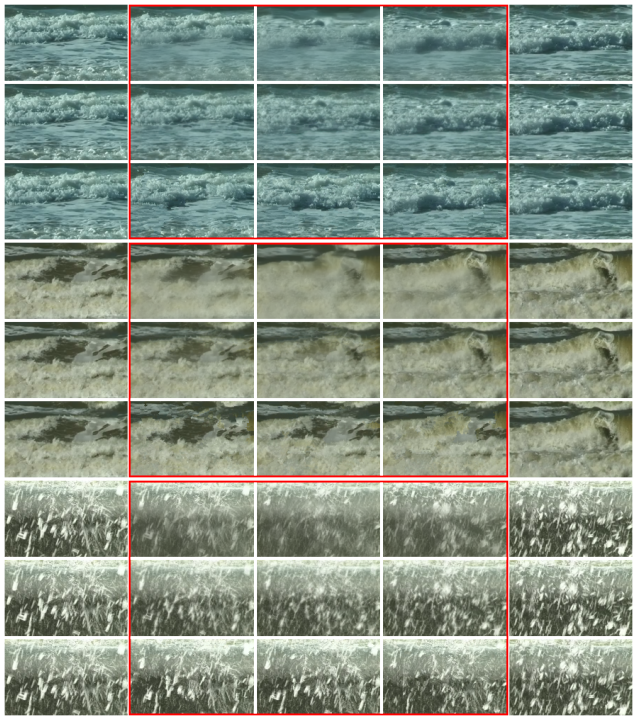

Figure 5. In each subfigure, synthesis results are in the red square. Rows from top to bottom correspond to [5], [3] and our method.

Instead of smoothing dynamic processes like [5, 3], our method synthesizes textures by selecting temporal patches whose trajectory has large curvature. The curvature-based trajectory selection algorithm preserves the properties of temporal patches, which helps us to synthesis textures having suitable dynamics. As a result, our method has much better synthesis results.

Additionally, we test our method on generating transitions between totally different images. In Fig. 6, although our method achieves image synthesis by finding temporal patches locally, transitions obtained by our method are still comparable to those obtained by morphing method [23, 5][4]. In the supplemental file we further show that, differing from [5], which generates a smooth transition, our method creates a transition with high degree of dynamic variations.

5. Conclusion and Future Work

We proposed a dynamic texture synthesis method for dealing with the case of extremely few samples. Combin-

[4]Because of the lack of source code, the result of [23] is downloaded directly from the website http://grail.cs.washington.edu/projects/regenmorph/.

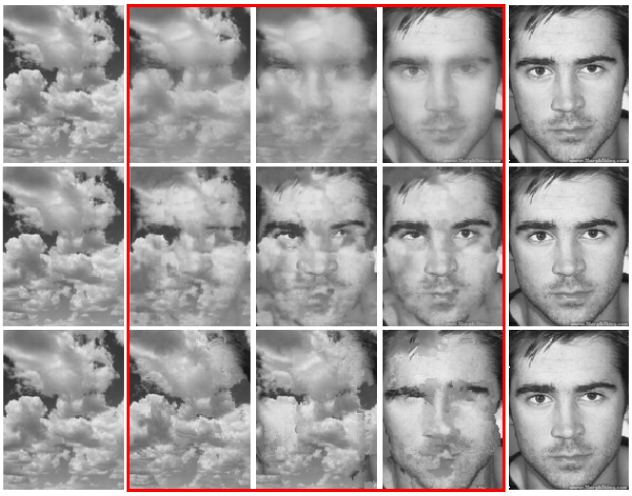

Figure 6. Images in the red square are the synthesis results. Rows from top to bottom correspond to [5], [23] and our method.

ing MRFs with manifold learning, our approach delivers promising texture synthesis results using just two frames. Our method works well especially for textures with prominent periodic patterns, e.g., ripples, raining scenes, etc. However, in the case of textures having persistent motions, our method is less successful in capturing the motion patterns, which is also shown in the supplemental file. For future work, we plan to explore more intrinsic properties of dynamic textures and develop methods that can handle persistent motions. We are considering using optical flow for the initialization step to provide the synthesis process with useful prior knowledge of motion [18, 16].

Acknowledgement: This work is supported in part by NSF grant DMS-1317424, CCF-1350616, NRL grant N00173-14-2-C001 and NSFC-61129001/F010403. Thanks Dr. Eli Shechtman and Dr. Soheil Darabi for providing us with their code about image morphing [5].

References

[1] J. Balle, A. Stojanovic, and J.-R. Ohm. Models for static and dynamic texture synthesis in image and video compression. *IEEE Journal of Selected Topics in Signal Processing*, 5(7):1353–1365, 2011. 1, 2

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM ToG*, 28(3):24, 2009. 2

[3] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *IJCV*, 61(2):139–157, 2005. 2, 7

[4] A. B. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *PAMI*, 30(5):909–926, 2008. 1

[5] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding: combining inconsistent images using patch-based synthesis. *ACM ToG*, 31(4):82, 2012. 2, 7, 8

[6] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. 5

[7] P. Dollár, V. Rabaud, and S. J. Belongie. Learning to traverse image manifolds. In *NIPS*, pages 361–368, 2006. 2, 3

[8] A. Dominitz and A. Tannenbaum. Texture mapping via optimal mass transport. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):419–433, 2010. 2

[9] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. Dynamic textures. *IJCV*, 51(2):91–109, 2003. 1, 2, 7

[10] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *ACM SIGGRAPH*, pages 341–346, 2001. 1, 2, 7

[11] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, volume 2, pages 1033–1038. IEEE, 1999. 2, 6, 7

[12] B. Galerne, Y. Gousseau, and J. Morel. Random phase textures: Theory and synthesis. *TIP*, 20(1):257–267, 2011. 1, 2

[13] D. Gong, X. Zhao, and G. Medioni. Robust multiple manifolds structure learning. In *ICML*, pages 321–328, 2012. 5

[14] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM ToG*, volume 22, pages 277–286, 2003. 1, 2, 5, 6, 7

[15] D. Lin and J. Fisher. Manifold guided composite of markov random fields for image modeling. In *CVPR*, pages 2176–2183. IEEE, 2012. 3, 6

[16] D. Lin, E. Grimson, and J. Fisher. Modeling and estimating persistent motion with geometric flows. In *CVPR*, pages 1–8. IEEE, 2010. 8

[17] R.-S. Lin, C.-B. Liu, M.-H. Yang, N. Ahuja, and S. Levinson. Learning nonlinear manifolds from time series. In *ECCV*, pages 245–256. Springer, 2006. 2, 7

[18] C. Liu and W. T. Freeman. A high-quality video denoising algorithm based on reliable motion estimation. In *ECCV*, pages 706–719. Springer, 2010. 8

[19] C. Ma, L.-Y. Wei, S. Lefebvre, and X. Tong. Dynamic element textures. *ACM ToG*, 32(4):90, 2013. 1, 2, 7

[20] R. Péteri, S. Fazekas, and M. J. Huiskes. Dyntex: A comprehensive database of dynamic textures. *Pattern Recognition Letters*, 31(12):1627–1632, 2010. 2, 6

[21] A. Rahimi, T. Darrell, and B. Recht. Learning appearance manifolds from video. In *CVPR*, volume 1, pages 868–875. IEEE, 2005. 2

[22] D. Seo, J. Ho, and B. C. Vemuri. Computing diffeomorphic paths for large motion interpolation. In *CVPR*, pages 1227–1232. IEEE, 2013. 2, 7

[23] E. Shechtman, A. Rav-Acha, M. Irani, and S. Seitz. Regenerative morphing. In *CVPR*, pages 615–622. IEEE, 2010. 2, 7, 8

[24] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR*, pages 1–8. IEEE, 2008. 2

[25] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. 6

[26] L.-Y. Wei, S. Lefebvre, V. Kwatra, G. Turk, et al. State of the art in example-based texture synthesis. In *Eurographics, State of the Art Report*, pages 93–117, 2009. 1, 2, 7

[27] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *CVPR*, volume 1, pages I–120. IEEE, 2004. 1

[28] H. Xu and H. Zha. Manifold based face synthesis from sparse samples. In *ICCV*, pages 2208–2215. IEEE, 2013. 2

[29] L. Yuan, F. Wen, C. Liu, and H.-Y. Shum. Synthesizing dynamic texture with closed-loop linear dynamic system. In *ECCV*, pages 603–616. Springer, 2004. 1, 2, 7

[30] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing*, 26(1):313–338, 2004. 2, 3, 5