Vis Comput
DOI 10.1007/s00371-009-0409-2

ORIGINAL ARTICLE

Fast multi-scale joint bilateral texture upsampling

Chunxia Xiao · Yongwei Nie · Wei Hua · Wenting Zheng

Abstract We present a new approach using a multi-scale joint bilateral filter for upsampling the synthesized texture generated by optimization-based methods. Our method is based on the following motivation: if the available exemplar texture is used as a priority to upsample the synthesized texture, a high resolution result that prevents image blurring can be obtained. Our multi-scale joint bilateral upsampling applies a spatial filter on each multi-scale layer of the synthesized texture, and jointly applies a similar range filter on exemplar texture, which guides the interpolation from low to high resolution, by magnifying and combining the upsampled information; the details of the upsampled texture are progressively enhanced, and the image blurring artifacts can be effectively avoided. We offer an accelerated joint bilateral filter, which enables our upsampling method to interactively generate a large texture. In addition, we propose a detail-aware texture optimization approach that incorporates image detail in texture optimization to improve the quality of the synthesized texture, on which the multi-scale joint bilateral filter works to generate a more convincing result. We show results for upsampling image and video textures and compare them to traditional upsampling methods, by this demonstrating that with low computational and memory costs, our method achieves better results.

Keywords Texture synthesis · Global optimization · Image upsampling · Bilateral filter

C. Xiao (✉) · Y. Nie
School of Computer, Wuhan University, Wuhan, China
e-mail: cxxiao@whu.edu.cn

Y. Nie
e-mail: nieyongwei@gmail.com

W. Hua · W. Zheng
State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China

W. Hua
e-mail: huawei@cad.zju.edu.cn

W. Zheng
e-mail: wtzheng@cad.zju.edu.cn

1 Introduction

Texture synthesis is a useful technique in computer graphics and computer vision, which synthesizes a large texture based on a small input sample while exhibiting the same stochastic features of the exemplar. A great deal of existing works synthesize texture using either parametric [1, 2], non-parametric [3–5], or patch-based [6–9] approaches. Among these methods, optimization-based methods [10, 12, 13], which belong to a kind of patch-based methods, have been proved very successful in terms of the quality of the synthesized results, and have been efficiently applied in image and video synthesis [10, 12], solid texture synthesis [13], inverse texture synthesis [18], and geometry-surface texture synthesis [11]. However, the synthesis procedure of these optimization-based methods is done by minimizing a global texture energy function, which is a time and memory consuming process, and it limits the wider applications of optimization-based methods.

To efficiently synthesize a large and high resolution texture using optimization-based methods [10, 12], an intuitive approach is to first synthesize a relative small and low resolution texture, then upsample this generated texture to the high one (Fig. 1). When the exemplar texture is large, especially for video exemplar texture for which it is time and memory consuming to construct tree-based acceleration structure for nearest neighbor search [10, 12], in this

Published online: 08 December 2009

case, an alternative method is to first downsample the exemplar texture to a lower resolution one, then synthesize a low resolution texture using this low resolution texture as an input, and then the synthesized low resolution texture is finally interpolated to generate a high resolution texture (Fig. 3). This interpolation operation is also called an upsampling procedure. To upsample the low resolution to a high one, many methods have been proposed, while most methods, such as the commonly-used Gaussian and bicubic interpolation, generally assume a smoothness priority for the interpolation and the results usually suffer from blurring edges, as illustrated in Figs. 2(b) and 4(b). Thus, a better interpolation method should not only require low computational and memory costs, but also achieve good interpolation result from the synthesized texture.

To address these problems, inspired by joint bilateral filtering techniques [23–25], we develop a novel algorithm for texture upsampling using multi-scale joint bilateral filtering. Our method is based on the following observation: if the available exemplar texture is used as a priority to upsample the synthesized texture, a high resolution result which prevents the image blurring can be obtained. We use joint bilateral filter to perform the upsampling: a spatial filter is applied to the synthesized texture, while a similar range filter is jointly applied on the exemplar texture. Using these techniques, the image blurring artifacts in upsampling process can be effectively prevented (Figs. 2(c) and 4(c)). When the features of the exemplar are relatively weak, the optimization-based methods [10, 12] do not work well to reconstruct the texture details. In this situation, to further improve the upsampling results, we maximize the multi-scale details of the synthesized texture, and propose a multi-scale joint bilateral upsampling method which progressively enhances the details of the upsampled texture (Fig. 6) using the exemplar texture as the priority. Applying the original input exemplar as the guide reference, our work addresses this deficiency of the existing interpolation methods such as bicubic and Gaussian interpolation which suffer from blurring edges.

Because the joint bilateral filter is a non-linear operator, a direct implementation of the 2D convolution costs $O(k^2N^2)$ operations, where k is the width of the spatial filter kernel and N is the width of the image. To make our joint bilateral upsampling procedure more efficient, the joint bilateral upsampling has to be accelerated. Although many methods have been proposed to accelerate the bilateral filtering [29–31], these methods are difficult to be adapted directly to our joint bilateral texture upsampling. We presented an acceleration technique which enables the joint bilateral filtering to perform in constant time, that is, the computation time of filtering remains unchanged even if the filter size becomes very large. This acceleration technique enables our upsampling method to interactively generate a large texture.

The traditional approach in texture synthesis is to compare an image patch with that of an exemplar. For textures with strong structures, feature maps can be particularly helpful. It has been shown that some textures can be better synthesized with the aid of feature maps which provide feature information. In our method, differently from the binary feature maps used in [14–16], we extract the detailed layer of the exemplar as an extra channel for patch similarity computing. Then, a detail-aware texture optimization is applied, which combines both texture optimization and histogram matching of the image detail to furthermore improve the quality of the synthesized results. When our multi-scale joint bilateral upsampling operation performs on these better synthesized textures, we can receive more conceiving results.

Overall, this paper presents the following three main contributions:

Multi-scale joint bilateral upsampling: we propose a multi-scale joint bilateral upsampling approach, which utilizes the exemplar texture as a priority to progressively interpolate the synthesized texture from low resolution to high resolution for producing a larger and better texture.

Accelerated joint bilateral filter: we present an accelerated joint bilateral filter, which enables the joint bilateral filtering to be performed in constant time, that is, the computation time of filtering remains unchanged even if the filter size becomes very large.

Detail-aware texture optimization: we incorporate the high frequency detail of the texture and its spatial variation into the texture optimization to further improve the synthesized results.

The rest of our paper is organized as follows. Section 2 reviews related work. Section 3 gives a brief description for the texture optimization, on which our upsampling algorithm is based. In Sect. 4, multi-scale joint bilateral texture upsampling algorithm is presented. In Sect. 5, we propose a fast joint bilateral texture upsampling approach, and in the subsequent Sect. 6, a detail-aware texture optimization method is presented. We extend our methods to video texture upsampling in Sect. 7. In Sect. 8, we give the experimental results and discussions. Finally, we conclude our paper in Sect. 9.

2 Related work

Our work is built on recent literature in texture synthesis and image upsampling. Numerous approaches have been proposed for both topics, and an exhaustive survey is beyond the scope of this paper. In the following, we review some recent and most related works.

The texture synthesis methods have shifted from parametric methods [1], to non-parametric methods [3], including pixel-based methods [4, 5], patch-based methods

[7, 9], and most recently to appearance-space texture synthesis [15], and optimization-based methods [10, 12, 13]. Parametric methods attempt to construct a parametric model of the texture based on the input sample, which have been proven to be a challenging task, and are mostly successful for homogeneous and stochastic textures. Non-parametric methods have demonstrated the ability to handle a much wider variety of textures, by growing the texture one pixel/patch at a time. Appearance-space method improves the quality by replacing pointwise color neighborhoods with appearance vectors that incorporates non-local information such as feature- and radiance-transfer data. We refer the reader to [13] and [15] and references therein for more complete surveys of texture synthesis.

Optimization-based methods [10, 12, 13] evolve the texture as a whole, further improving the quality of the results and making the synthesis more controllable. By defining a Markov Random Field (MRF) based similarity metric for measuring the quality of synthesized texture with respect to a given input sample, the synthesis problem is formulated as minimization of an energy function, which is optimized using an algorithm similar to Expectation Maximization (EM). Wexler et al. [12] defined an energy function for the completion of missing information of the video. Kwatra et al. [10] defined global texture optimization for image texture synthesis. More recently, Kopf et al. [13] extended global texture optimization method [10, 12] to the task of solid texture synthesis; in addition, Kopf et al. [13] integrated histogram matching the texture optimization, which improved the convergence of the synthesis process and partially addressed the issue that the optimization process could get stuck in a local minimum. More recently, differently from the traditional texture synthesis, Wei et al. [18] presented an inverse texture synthesis method, which used an optimization framework to produce a small texture compaction that best summarized original large globally varying texture.

It has been shown that some textures can be synthesized better with the aid of feature maps, which provide non-local feature information [13–16]. Texture synthesis guided by feature maps gives rise to more satisfactory results by reducing the number of feature discontinuities and artifacts. Zhang et al. [16] used binary texton mask as a control channel for color synthesis, and the mask extraction [16] needed manual intervention. Wu and Yu [14] use binary image which is obtained automatically to guide the synthesis with the feature matching and alignment operations. Lefebvre and Hoppe [15] also applied binary feature mask that encodes the signed feature distance to better preserve semantic texture structures. Instead of using the binary feature maps, in this paper, we incorporate detailed layer to the texture optimization, which generates more convincing results.

Image upsampling is a fundamental processing operation in the computer graphics and image processing. Many techniques have been proposed [19–21]. For example, the classical approaches such as the Nearest-Neighbor, Bilinear, Bicubic, Hamming, and Lanczos interpolation kernels, are very popular in commercial software. The construction of these kernels relies strongly on the assumption that the image data is either spatially smooth or band-limited. While, as is well known, these assumptions are not true for most images, and the solutions obtained using these methods tend to produce visual artifacts such as aliasing, ringing, blocking, and blurring. A more detailed survey of these techniques and their shortcomings is given in [22]. To reduce the blurriness and other artifacts, Su et al. [20] adjusted the interpolation weights locally by choosing three out of the four nearest pixels to reduce the number of variables that are averaged, while Fattal [19] proposed an image upsampling technique which was based on the edge characteristic. For more image upsampling methods please refer to the related work section in [19].

Differently from the existing methods which rely on the smoothness assumptions, more recently, based on the joint bilateral filters [23, 24], Kopf et al. [25] presented a joint bilateral upsampling method (JBU) which applied the available high resolution input image as a priority to produce a better high resolution image and which received satisfactory results. Inspired by the joint bilateral filter [23–25] techniques, we upsample the synthesized texture using the high resolution exemplar texture as a priority. Compared with traditional upsampling methods, our method achieves convincing results. As will be shown in the related section, the adaption is not trivial and our method is novel for texture upsampling.

3 Texture optimization using EM

Optimization-based texture synthesis methods [10, 12] work as follows. The optimization process begins with a context region where the value of each pixel is randomly chosen from the exemplar. The goal is to iteratively increase the similarity between the synthesized texture on the context region and the exemplar by minimizing an energy function that measures the differences between the two. For simplicity, we take the case of 2D image texture synthesis as an example.

Formally, let X denote the context region over which we want to synthesize the texture and Z the input exemplar. Let X_p be the vectorized neighborhood of pixel p in X, and Z_p be the vectorized pixel neighborhood in Z whose appearance is most similar to X_p under the Euclidean distance. Then, the texture energy over X is defined as

$$E_t\big(X;\{Z_p\}\big)=\sum_p \|X_p - Z_p\|^r. \tag{1}$$

Using iteratively re-weighted least squares (IRLS) [10] to minimize the energy, the right term of the energy function (1) can be rewritten as follows:

$$\|X_p - Z_p\|^r = \underbrace{\|X_p - Z_p\|^{r-2}}\|X_p - Z_p\|^2$$
$$= \omega_p \|X_p - Z_p\|^2$$

and minimize the following quadratic function:

$$E_t\big(X, \{Z_p\}\big) = \sum_p \sum_{u \in N(p)} \omega_{p,u}(X_{p,u} - Z_{p,u})^2, \tag{2}$$

here, $N(p)$ denotes the neighborhood of the pixel p and $\omega_{p,u} = \omega_p$, $Z_{u,p}$ denotes the exemplar pixel in the neighborhood Z_u (the nearest neighborhood of X_u in Z) that corresponds to p . Assuming that the weights $\omega_{p,u}$ are constant during the optimization phase, and setting the derivative of (2) with respect to p to zero, yields the following solution:

$$p = \frac{\sum_{u \in N(p)} \omega_{u,p} Z_{u,p}}{\sum_{u \in N(p)} \omega_{u,p}}. \tag{3}$$

Thus, the optimal value of each pixel is simply a weighted average of a collection of pixels from different exemplar neighborhoods. Similar energy function can be defined for solid texture synthesis [13] and video completion [12].

The energy function (1) is minimized similarly to Expectation Maximization (EM) [26]. The estimation of X is obtained by minimizing the texture energy in (1), which corresponds to the E-step, while finding the set of closest input neighborhoods, $\{Z_p\}$, corresponds to the M-step.

4 Multi-scale joint bilateral texture upsampling

Optimization-based texture synthesis methods [10, 12, 13] are time-consuming and large memory is required using tree-based acceleration structure such as ANN [27] for nearest neighbor search. Sometimes, the large data may make the computation untractable, especially for video texture synthesis. Thus, to produce a large and high resolution texture, it is an intuitive idea to upsample from the synthesized texture computed in a smaller one. The traditional methods, such as bilinear interpolation or Gaussian interpolation, usually assume smoothness prior to the interpolation, and may suffer from blurred images. Inspired by the joint bilateral methods [23–25], we upsample the synthesized texture to a large and higher resolution one using the original exemplar image as the priority to produce better results.

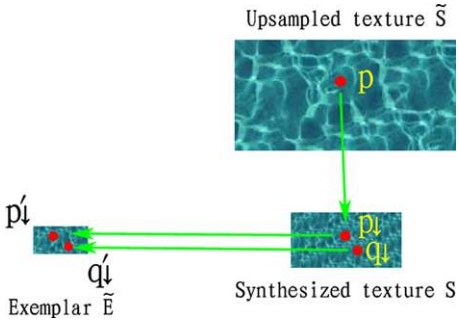


Fig. 1 Overview of the texture interpolation using joint bilateral upsampling method

4.1 Joint bilateral texture upsampling

Let $\tilde{E}$ be the input exemplar texture, image S being the synthesized texture for applying optimization algorithm [10]. To upsample the solution S to a higher resolution texture $\tilde{S}$, we use $\tilde{E}$ as the priority for upsampling operator, as illustrated in Fig. 1. The joint bilateral filter applies a spatial filter on the small texture S, while a similar range filter is jointly applied on the high resolution exemplar $\tilde{E}$. More specifically, let p be a pixel in $\tilde{S}$, $p_\downarrow$ be its corresponding pixel in the image S, and $q_\downarrow$ be the neighboring pixel of $p_\downarrow$. Let $p'_\downarrow$ and $q'_\downarrow$ be pixels in $\tilde{E}$ that contribute to the value of pixel $p_\downarrow$ and $q_\downarrow$ during texture optimization procedure, respectively, then the value of pixel p in the upsampled solution $\tilde{S}$ is computed as:

$$\tilde{S}_p = \frac{1}{k_p} \sum_{q_\downarrow \in \Omega} S_{q_\downarrow} f\big(\|p_\downarrow - q_\downarrow\|\big) g\big(\|\tilde{E}_{p'_\downarrow} - \tilde{E}_{q'_\downarrow}\|\big), \tag{4}$$

where $k_p = \sum_{q_\downarrow \in \Omega} f(\|p_\downarrow - q_\downarrow\|) g(\|\tilde{E}_{p'_\downarrow} - \tilde{E}_{q'_\downarrow}\|)$ and Ω is the neighborhood of $p_\downarrow$ in S, and $\tilde{E}_{p'_\downarrow}$ and $\tilde{E}_{q'_\downarrow}$ are the values of pixels $p'_\downarrow$ and $q'_\downarrow$. Note that $q_\downarrow$ takes only integer coordinates in S, so the guidance image $\tilde{E}$ is only sparsely sampled. This joint bilateral texture upsampling operator (4) is almost identical to standard bilateral filter [28], except that a high resolution texture is constructed by operating at two different images simultaneously, rather than filtering a single image.

Note that in texture optimization, as shown in (3), the optimal value of each pixel $p_\downarrow$ and $q_\downarrow$ is a weighted average of a collection of pixels from different exemplar neighborhoods in $\tilde{E}$, that is:

$$S_{p_\downarrow} = \sum_{u \in N(p_\downarrow)} \omega_{u,p_\downarrow} \tilde{E}_{u,p'_\downarrow} \Big/ \sum_{u \in N(p_\downarrow)} \omega_{u,p_\downarrow}, \tag{5}$$

$$S_{q_\downarrow} = \sum_{v \in N(q_\downarrow)} \omega_{v,q_\downarrow} \tilde{E}_{v,q'_\downarrow} \Big/ \sum_{v \in N(q_\downarrow)} \omega_{v,q_\downarrow}, \tag{6}$$

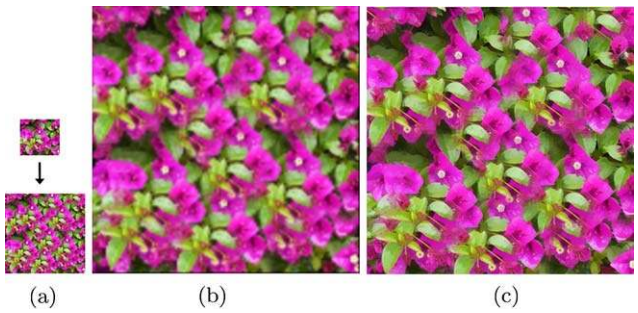


Fig. 2 (**a**) The exemplar (64 × 64) is synthesized to texture (128 × 128). (**b**, **c**) The synthesized texture (128 × 128) is upsampled to the images (512 × 512) using Gaussian interpolation and joint bilateral upsampling, respectively

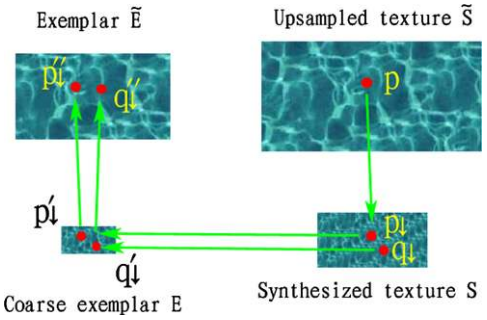


Fig. 3 Overview of the texture interpolation using hierarchical joint bilateral upsampling methods

where $S_{p\downarrow}$ and $S_{q\downarrow}$ are the texture values of pixels $p\downarrow$ and $q\downarrow$, respectively. So in (4), we have to process $\tilde{E}_{p'\downarrow}$ and $\tilde{E}_{q'\downarrow}$ with more techniques.

To efficiently compute this non-linear problem, we use the following strategy. We compute the final value $\tilde{S}_p$ in the following two steps.

In the first step, we fix $\tilde{E}_{p'\downarrow}$, and replace $\tilde{E}_{q'\downarrow}$ in (4) with $S_{q\downarrow}$, a weighted average of pixels $\tilde{E}_{v,q'\downarrow}$ computed in (6); we obtain

$$\tilde{S}_{u,p} = \frac{1}{k'_p} \sum_{q\downarrow \in \Omega} S_{q\downarrow} f\big(\|p\downarrow - q\downarrow\|\big) g\big(\|\tilde{E}_{u,p'\downarrow} - S_{q\downarrow}\|\big), \quad (7)$$

where k'_p is a normalizing factor. The joint bilateral texture upsampling operator (7) is called JBTU.

In the second step, according to (3), the optimal value of each pixel $p\downarrow$ is a weighted average of a collection of pixels from different exemplar neighborhoods in $\tilde{E}$, and we compute the final value of pixel p in $\tilde{S}$ as:

$$\tilde{S}_p = \frac{\sum_{u\in N(p\downarrow)} \omega_{u,p\downarrow} \tilde{S}_{u,p}}{\sum_{u\in N(p\downarrow)} \omega_{u,p\downarrow}}. \quad (8)$$

The above ideas can also be described as the following: we first perform the weighted average for $q\downarrow$ in the inner loop, and then perform the joint bilateral operation, and finally, we perform the weighted average step for $p\downarrow$. By using this technique, the final upsampled value of pixel can be computed efficiently. As shown in Fig. 2, using the proposed method, the blurring artifacts are better avoided and the sharp edges are successfully preserved.

Note that $p\downarrow$ is an integer coordinate, while $p\downarrow$ possibly be fractional coordinates in S, so it does not correspond to an integer coordinate in $\tilde{E}$, which makes it difficult to apply (4) directly. In this case, we let $p\downarrow$ be its closest integer coordinate pixel. Notice also that the weights $\omega_{u,p\downarrow}$ and $\omega_{v,q\downarrow}$ in (5) and (6) have been computed during the optimization step of the last optimization iteration, so they do not need to be recomputed at the upsampling step.

4.2 Hierarchical joint bilateral texture upsampling

When the exemplar texture is large, especially for large video exemplar texture, it is usually time and space consuming to work directly on the original exemplar texture. In the M-step of texture optimization, we have to find a nearest neighborhood in the exemplar texture for each patch of the context region. Even using acceleration methods such as tree-construction of approximate nearest neighbors [27] and dimensionality reduction (PCA) [13], the time and memory requirements remain the bottleneck of optimization-based texture synthesis, preventing them from attaining desirable speeds. To accelerate texture synthesis as well as to generate good results, we propose a hierarchical joint bilateral texture upsampling (hierarchical JBTU) method: producing high resolution texture results by upsampling the texture synthesized using the low resolution exemplar texture, while using the original exemplar texture as the priority, as illustrated in Fig. 3. More technical details are described in the following.

Let E be the low resolution version of the exemplar texture $\tilde{E}$ (e.g. Gaussian downsampling), the texture S being synthesized using E as the exemplar. To upsample the texture S to a higher resolution texture $\tilde{S}$, we use $\tilde{E}$ instead of E as the priority for upsampling, as illustrated in Fig. 3. The joint bilateral filter applies a spatial filter to the low resolution texture S, while a similar range filter is jointly applied to the original exemplar $\tilde{E}$. More specifically, let p be a pixel in $\tilde{S}$, $p\downarrow$ be its corresponding pixel in the image S, and $q\downarrow$ be the neighboring pixel of $p\downarrow$. As described in the EM optimization process, the value of each output pixel is a weighted average of a collection of pixels from different exemplar neighborhoods. For simplicity, let $p'\downarrow$ and $q'\downarrow$ be pixels in E that contribute to the value of pixels $p\downarrow$ and $q\downarrow$, respectively, pixels $p''\downarrow$ and $q''\downarrow$ be the corresponding pixels of $p'\downarrow$ and $q'\downarrow$ in the $\tilde{E}$. Similarly as in the previous subsection, the upsampled solution $\tilde{S}$ is obtained as:

$$\tilde{S}_p = \frac{1}{k''_p} \sum_{q\downarrow \in \Omega} S_{q\downarrow} f\big(\|p\downarrow - q\downarrow\|\big) g\big(\|\tilde{E}_{p''\downarrow} - S_{q\downarrow}\|\big) \quad (9)$$

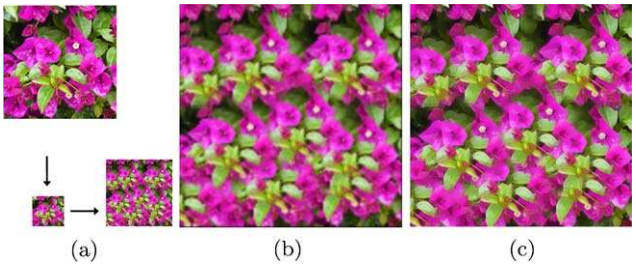


Fig. 4 (**a**) The exemplar (256 × 256) is downsampled to an exemplar (64 × 64), and a small texture (128 × 128) is synthesized using the small exemplar as input. (**b**, **c**) The synthesized texture is upsampled to the images (512 × 512) using Gaussian interpolation and hierarchical joint bilateral upsampling, respectively

where k''_p is the normalizing factor. Note that $q_\downarrow$ takes only integer coordinates in the low resolution solution, so the guidance image $\tilde{E}$ is only sparsely sampled. Similarly to the JBTU method described in Sect. 4.1, the final value of pixel p can be computed efficiently in two steps.

As illustrated in Fig. 4, using the proposed hierarchical JBTU method, compared with the Gaussian upsampling, our method prevents the image deblurring artifacts. We synthesize the texture S from the low resolution exemplar E which makes the synthesis procedure much faster, while using the original texture as the priority for upsampling; the upsampling result is satisfying. One difference from JBTU method is that, using the hierarchical JBTU, if the proportion that the exemplar is downsampled to coarse exemplar is the same as that of the synthesized texture is upsampled to the final image, the texture feature scale of the upsampled texture can be the same as in the original full resolution exemplar. Using JBTU, however, as shown in Fig. 2, the texture feature scale of the JBTU results is larger than the exemplar texture.

4.3 Multi-scale joint bilateral upsampling

Although the joint bilateral sampling method generates satisfactory results, in some situations, when the detail features of the exemplar are relatively weak, the details of the upsampled texture may not be well-reconstructed as expected. Furthermore, since the joint bilateral filter is an edge-preserving operator, the upsampling results may not be smooth enough. To further enhance the texture details of upsampled texture as well as keeping the result smooth, we present a multi-scale joint bilateral texture upsampling technique (multi-scale JBTU). We compute a multi-scale decomposition for the synthesized low resolution texture S using the bilateral filter, and then extract the progressive detailed layers of S. Guided by the exemplar image $\tilde{E}$, we upsample each layer of the synthesized low resolution, and reconstruct a final enhanced upsampled texture image that combines all the upsampled information of each scale across the texture S. More technique details are given as follows.

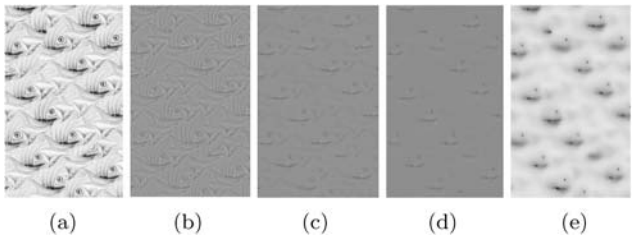


Fig. 5 Image multi-scale decomposition, the image (**a**) is decomposed into detailed layers (**b**, **c**, **d**) and base image (**e**)

The multi-scale bilateral decomposition of input image S is to build a series of filtered images S^j that preserve the strongest edges in S while smoothing small changes in intensity. At the finest scale $j = 0$, we set $S^0 = S$ and then iteratively apply the bilateral filter to compute

$$S^{j+1}_{p_\downarrow} = \frac{1}{k^{j+1}_{p\downarrow}} \sum_{q_\downarrow \in \Omega} S^j_{q\downarrow} g_{\sigma_s}\big(\|p_\downarrow - q_\downarrow\|\big) \times g_{\sigma_r}\big(\|S^j_{q_\downarrow} - S^j_{p_\downarrow}\|\big), \tag{10}$$

with $k^{j+1}_{p\downarrow}$ the normalizing factor. Inspired by [29], we compute a set of detail images as differences between successive levels of these bilateral filtered images $d^j = S^{j-1} - S^j$ for $j = 1, \ldots, m$. Thus, the S^j retains the strongest edges in the image and the detailed layers d^j contain the smaller changes in intensity. We can reconstruct the image from this decomposition as $S = \sum_{j=1}^{m} d^j + S^m$. Figure 5 illustrates the multi-scale decomposition of an image.

The multi-scale decomposition of the synthesized texture S can be used to progressively enhance the upsampled texture $\tilde{S}$. We enhance the details of $\tilde{S}$ by combining the magnified upsampled details $\tilde{S}^{\text{Detail}}$ across entire scale of the detailed layers of image S and the upsampled base image $\tilde{S}^{\text{Base}}$ of image S. We generate the enhanced output texture image $\tilde{S}^{\text{Result}}$ as

$$\tilde{S}^{\text{Result}} = \tilde{S}^{\text{Detail}} + \beta \cdot \tilde{S}^{\text{Base}}, \quad 0 < \beta \leq 1. \tag{11}$$

The parameter β is the trade-off between detail image and base image.

The base image $\tilde{S}^{\text{Base}}$ sets the coarsest level image of $\tilde{S}^{\text{Result}}$ and retains the strongest edges, which has a strong effect on the overall appearance of the resulting image $\tilde{S}^{\text{Result}}$. Image $\tilde{S}^{\text{Base}}$ is computed as follows:

$$\tilde{S}^{\text{Base}} = \frac{1}{k^B_p} \sum_{q_\downarrow \in \Omega} S^m_{q\downarrow} f\big(\|p_\downarrow - q_\downarrow\|\big) g\big(\|\tilde{E}_{p'\downarrow} - S^m_{q\downarrow}\|\big).$$

The upsampled detail $\tilde{S}^{\text{Detail}}$ maximizes each upsampled detail D^j of the d^j. We compute $\tilde{S}^{\text{Detail}}$ as a weighted sum

of the upsampled difference images D^j,

$$\tilde{S}^{\text{Detail}} = \sum_{j=1}^{m} U^j D^j \Big/ \sum_{j=1}^{m} D^j,$$

where $D_p^j = \frac{1}{k_p^D} \sum_{q_\downarrow \in \Omega} d_{q_\downarrow}^j f(\|p_\downarrow - q_\downarrow\|) g(\|\tilde{E}_{p'\downarrow} - d_{q_\downarrow}^j\|)$, and U^j is the weight for different layer D^j. Since we wish to maximize small changes in intensity, the weight U^j rewards pixels with large D^j, the weight is computed as $U^j = g_{\sigma_d} * e^{|D^j|}$, the spatial convolution with a Gaussian g_{σ_d} is used to locally smooth the weight and reduce the noise. There are also other methods to determine the weight U^j in [29]. In our method, we find the above weight works well.

Our experiments show that multi-scale JBTU can effectively enhance and exaggerate details of the upsampled texture. Figure 6 shows one comparison results between JBTU and multi-scale JBTU. A close examination reveals that, compared with the original exemplar texture, many of the edges of the result generated using JBTU are not clear enough; in contrast, the edges in multi-scale JBTU appear much clearer. Figure 7 gives the upsampled results using proposed multi-scale JBTU with different scales. Other comparisons are shown in Fig. 10, where we demonstrate that we can generate highly enhanced detail even from the synthesized low resolution texture using multi-scale JBTU.

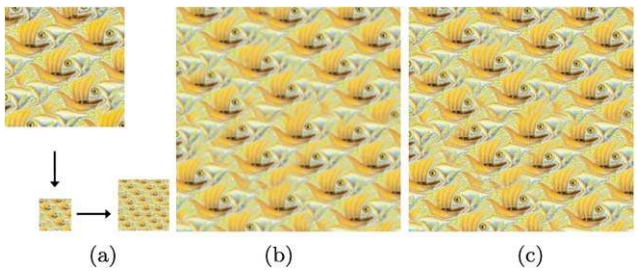


Fig. 6 (**a**) The exemplar (128 × 128) is downsampled to coarse exemplar (32 × 32), which is used to synthesize a small texture (64 × 64). (**b**, **c**) The synthesized texture is upsampled to the images (512 × 512) using hierarchical JBTU and multi-scale JBTU, respectively

5 Fast joint bilateral texture upsampling

The joint bilateral filter is non-linear, a coarse-force implementation of the 2D convolution costs $O(k^2N^2)$ operations, where k is the width of the spatial filter kernel and N is the width of the image. Many methods have been proposed to accelerate the bilateral filtering [29–31], while these methods are difficult to be adapted to our joint bilateral texture upsampling operator (7), (9). To make our multi-scale joint bilateral upsampling operator more efficient, inspired by [17], we present a constant time 0(1) joint bilateral filter, which enables the joint bilateral filtering to perform in constant time, that is, the computation time of filtering remains even if the filter size become very large. This acceleration method makes our upsampling process interactive to generate a large texture.

The joint bilateral texture upsampling operator (7) can be rewritten into the following formula:

$$\tilde{S}_p = \frac{1}{k'_p} \sum_{k \in \Omega} f(k) S_{p_\downarrow + k} g\big(\|\tilde{E}_{p'_\downarrow} - S_{p_\downarrow + k}\|\big), \tag{12}$$

where we choose the spatial filter f to be polynomial filters, and the range filter g to be Gaussian filters. We apply the Taylor series expansion of the Gaussian function g to approximate the bilateral filters. Gaussian filters are differentiable and can be expressed in terms of linear transforms. Let filter g be Gaussian range filter $g = \exp(-a[\tilde{E}_{p'_\downarrow} - S_{p_\downarrow + k}]^2)$, then g can be rewritten as

$$g = \exp\big(-a\tilde{E}_{p'_\downarrow}^2\big) \exp\big(-aS_{p_\downarrow+k}^2 - 2\tilde{E}_{p'_\downarrow} S_{p_\downarrow+k}\big), \tag{13}$$

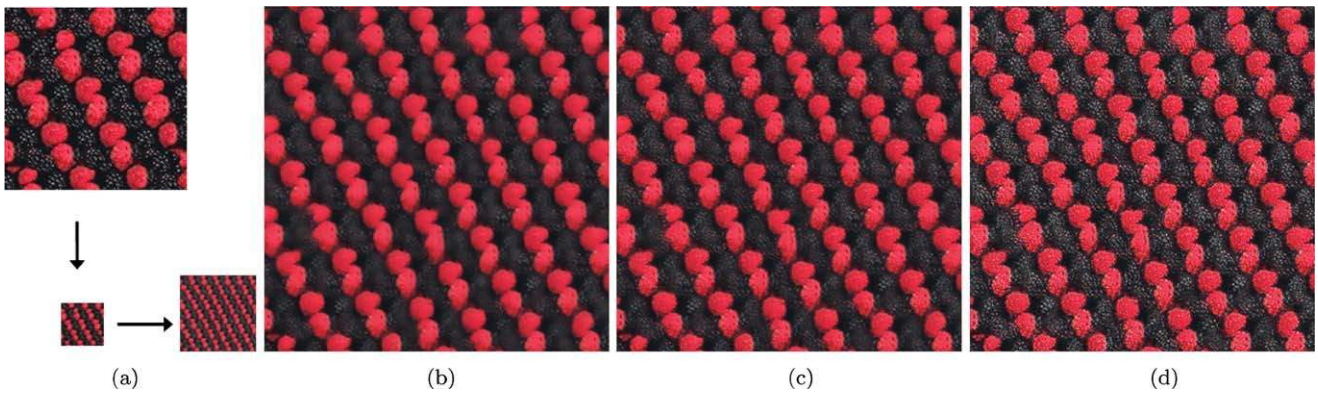


Fig. 7 Multi-scale joint bilateral upsampling. (**a**) The exemplar (256 × 256) is downsampled to small exemplar (64 × 64), and a small texture (128 × 128) is synthesized using small exemplar as input; (**b**) the synthesized texture is upsampled to the image (512 × 512) using hierarchical JBTU; (**c**, **d**) the synthesized texture is upsampled applying multi-scale JBTU, using 2 and 4 levels of upsampling, respectively

where the first term $\exp(-a\tilde{E}^2_{p'_\downarrow})$ does not depend on the range kernel. This term also appears in the normalizing term k'_p, thus it does not have to be computed separately.

By applying the second-order Taylor expansion to the Gaussian range filter (13), we obtain

$$\tilde{S}_p = (k'_b)^{-1}\big[y_1 + 2aRy_2 + a\big(2aR^2 - 1\big)y_3 - 2a^2Ry_4 + 0.5a^2y_5\big], \quad (14)$$

where $R = \tilde{E}_{p'_\downarrow}$, $R^2 = \tilde{E}^2_{p'_\downarrow}$, $y_i = \sum f(k)S^i_{p_\downarrow+k}$, the normalizing terms k'_b have similar forms. Therefore, a bilateral filter can be interpreted as the weighted sum of the spatial filtered responses of the powers of the original image.

Using the polynomial filters $f(k) = 1 - k^n$, the term y_i can be computed in constant time 0(1) using a set of integral image [17]. For squire distance norm, we get

$$\begin{aligned} y_1(p_\downarrow) &= \sum_{k\in\Omega} f(k)S_{p_\downarrow+k} \\ &= \sum_{z\in\Omega^z} f(z - p_\downarrow)S_z \\ &= \big[1 - p^2_\downarrow\big](A + B - C), \end{aligned}$$

where $A = \sum_{z\in\Omega^z} S_z$, $B = 2p_\downarrow \sum_{z\in\Omega^z} zS_z$, and $C = \sum_{z\in\Omega^z} z^2S_z$, Ω^z is the new kernel around $z - p_\downarrow$. The sums $\sum S_z$, $\sum zS_z$, $\sum z^2S_z$ can be computed directly from the corresponding integral images. The other terms y_i can be computed in a similar way.

An integral image is the accumulated sum of original image intensities. The sum of any region $\sum S_z$ can be computed by three arithmetic operations involving the values of the integral image at the corners of the region. Integral image takes advantage of the overlapping kernels to avoid redundant computing, it enables fast computation of sum if image pixel intensities in a rectangular region. The normalizing factor k'_p can be efficiently computed in a similar way; more details refer to [17]. Since these sums require only fixed number of operations at the corner points of the rectangular regions in integral images, the total computation time is independent of the region size.

The introduced Gaussian range and arbitrary spatial joint bilateral filters (12) are expressed by Taylor series, which results in a linear filter decompositions without any noticeable degradation; we call this method (14) a Fast JBTU. The proposed methods drastically decrease the computational time by cutting it down constant times (0.1 s for 1 MB image). The complexity is $O(1)$, which makes our JBTU operator fast even the filter size becomes very large, while achieving satisfying results. The Fast JBTU also can be directly used to accelerate the multi-scale JBTU since each upsampled detailed layer and the upsampled based image can be computed using Fast JBTU.

(a) (b) (c) (d)

Fig. 8 (**a**) Original image, (**b**) intensity, (**c**) large-scale, (**d**) detailed layer

6 Detail-aware texture optimization

It has been shown that some textures can be synthesized better with the aid of feature maps [14, 15] by preventing the feature breaking at the boundary of adjacent patches. To handle textures with strong large structures, and to reduce the number of feature discontinuities, we also provide a feature map as an extra channel. Differently than in the previous methods [14, 15] that use binary image as the feature maps, we consider the feature maps as detailed layer G_d of the texture obtained using the non-linear decomposition, as described in [30].

The detailed layer G_d extraction is similar to that in Sect. 4.3, but has some difference. More formally, let G_o be the original image, and G_i be the intensity of G_o, by filtering G_i using the bilateral filter results in large-scale layer G_f. The detailed layer G_d of G_o is deduced by a division of the intensity G_i by the large-scale layer G_f, that is $G_d = G_i/G_f$. Then, G_d is used to describe the feature of the exemplar, as illustrated in Fig. 8. In practice, we perform the detail calculations on the logs of pixel intensities, which yields a more uniform treatment of the whole range, then the detailed layer G_d corresponds to pixel differences.

Once the detailed layer texture is extracted, each of its pixels encodes both color and feature value, which is more effective for neighborhood matching than binary image. These detailed layer components are scaled into [0, 255] and added to the RGB measurements to obtain a four-dimensional representation for each pixel $(\mathrm{R}, \mathrm{G}, \mathrm{B}, \alpha\mathrm{D})$, where D is the detailed layer value of the image, α is the importance weight and is set between 1 and 2 in our paper. We apply an L2-norm (SSD) to this 4D representation in order to capture color-feature similarities between texture neighborhoods.

On the other hand, the histogram built on detailed layer can capture detail information of the image better than the binary image. For many textures the optimization process [10] may converge to a wrong local minimum, because the energy function measures only the similarity of local neighborhoods, without accounting for any global statistics. To address this issue, inspired by Heeger

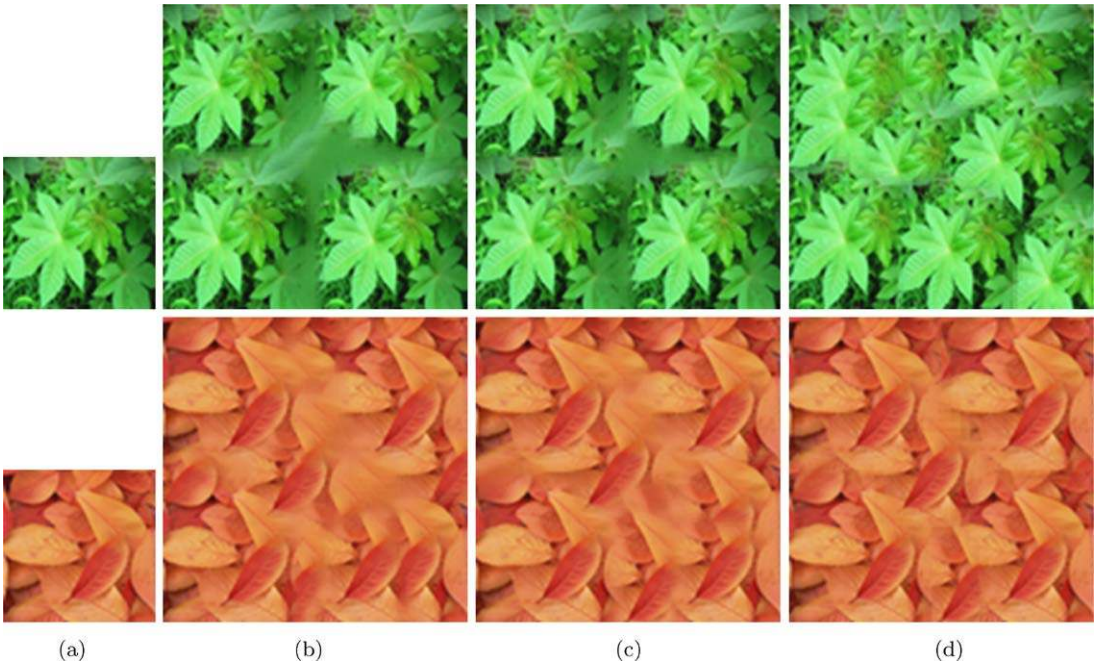


Fig. 9 (**a**) Exemplar, (**b**) result using [10], (**c**) result using EM involving histogram matching [13], (**d**) result involving the texture detailed layer

and Bergen's work [1], Kopf et al. [13] introduced a re-weighting scheme to make the result preserve the global statistics of the exemplar. More specifically, they adjust the weights in (3) so as to effectively ensure that certain histograms of the synthesized texture match the exemplar.

We also construct and keep track of histogram H for each of the texture's four channels (R, G, B and detailed layer D). Let $H_{s,j}$ and $H_{e,j}$ denote the jth histogram (R, G, B, and D) of the synthesized texture S and the exemplar texture E, respectively, and let $H(b)$ denote the value of bin b in a histogram H. Similarly to re-weighting method [13], we modify the weights for (3) as follows: $\omega'_{(u,j)} = \omega_u/(1 + \sum_{j=1}^{K=4} \alpha_j \max[0, \Theta])$, where $\Theta = H_{S,j}(b_j(E_u)) - H_{E,j}(b_j(E_u))$ and weights $\alpha_1 = \alpha_2 = \alpha_3 = 1$, α_4 is a parameter that can be tuned by the user. In our experiments, we set $\alpha_4 = 1$ and receive good results. To receive better results, the histograms must be kept up to date as the synthesis progresses, we employ the fast bilateral filter [31] to compute the detailed layer for the exemplar and synthesizing result in real time, which makes it fast to construct histogram for detailed layer.

As illustrated in Fig. 9, compared with the existing methods [10] and [13], using our method, the feature structures are better synthesized. In Fig. 12, we present video completion results; similarly, the proposed detail-aware completion method further improves the completion result. We upsample the completed low resolution result using hierarchical JBTU and receive convincing results.

7 Extension to video texture upsampling

Our multi-scale joint bilateral image texture upsampling can be easily extended to upsample the synthesized video texture using optimization [10, 12]. To allow for a uniform treatment of dynamic and static video texture information, we treat video sequences as space–time volumes. The main difference between 2D image texture synthesis and 3D video texture synthesis is that the 2D neighborhood X_p and Z_p in (1) are replaced with 3D space–time neighborhood. Compared with image texture synthesis, the computational time and memory requirements in 3D video synthesis increased accordingly.

As shown in Fig. 11, in our hierarchical multi-scale joint bilateral video texture upsampling, as the data in the exemplar video is large, to synthesize a large video texture using the original example as input is time and memory consuming. Similarly to image texture upsampling case, we downsample the original example into a smaller one, using this as input, we synthesize a low resolution video texture, then guided by the original high resolution example, we generate a better and high resolution video texture. Note also that to receive a smaller exemplar video texture, we only downsample original video in space dimension, not in time dimension, and we apply the Gaussian downsamping operator.

The real time detail extraction of the video sequence is the main consideration for multi-scale joint bilateral video upsampling. To filter the video in real time to receive the video texture detailed layer, we apply the Bilateral Grid

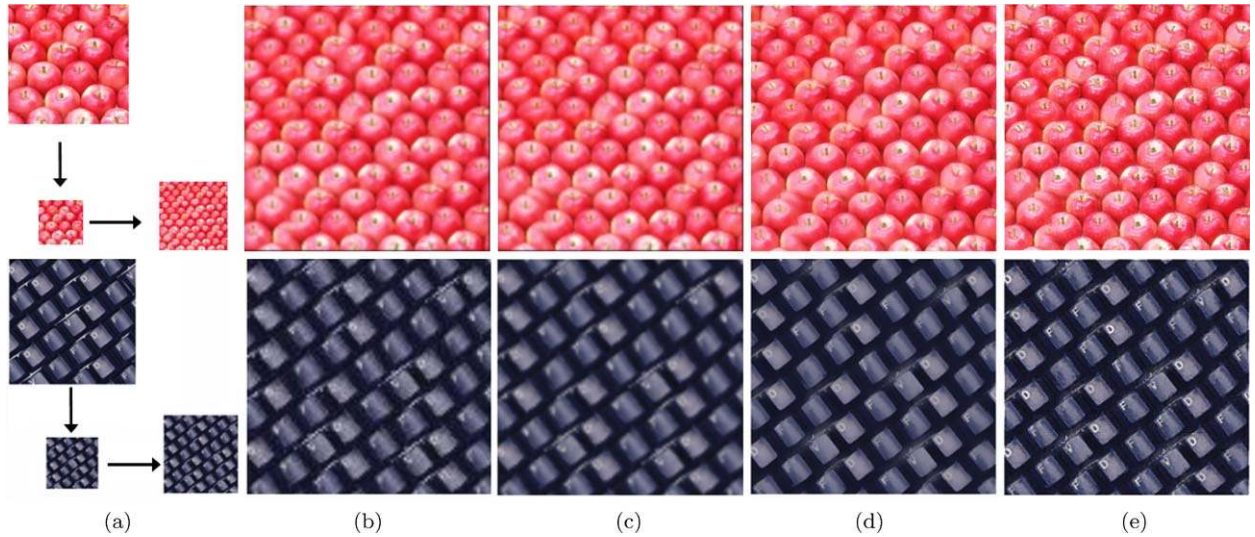


Fig. 10 (**a**) The exemplar (256 × 256) is downsampled to small exemplar (64 × 64), and a small texture (128 × 128) is synthesized using small exemplar as input. (**b**) The synthesized small texture is upsampled to the image (512 × 512) using Gaussian interpolation. (**c**) Upsampled result using bicubic interpolation. (**d**) Upsampled result using hierarchical JBTU. (**e**) Upsampled result using multi-scale JBTU

Fig. 11 Hierarchical video upsampling. (**a**) The exemplar (96 × 96 × 32) is downsampled to a small exemplar (48 × 48 × 32), which is used as input to synthesized a video (150 × 70 × 90), the synthesized video is upsampled to video (**b**) using Gaussian interpolation, video (**c**) using hierarchical JBTU, and video (**d**) using multi-scale JBTU (350 × 170 × 90)

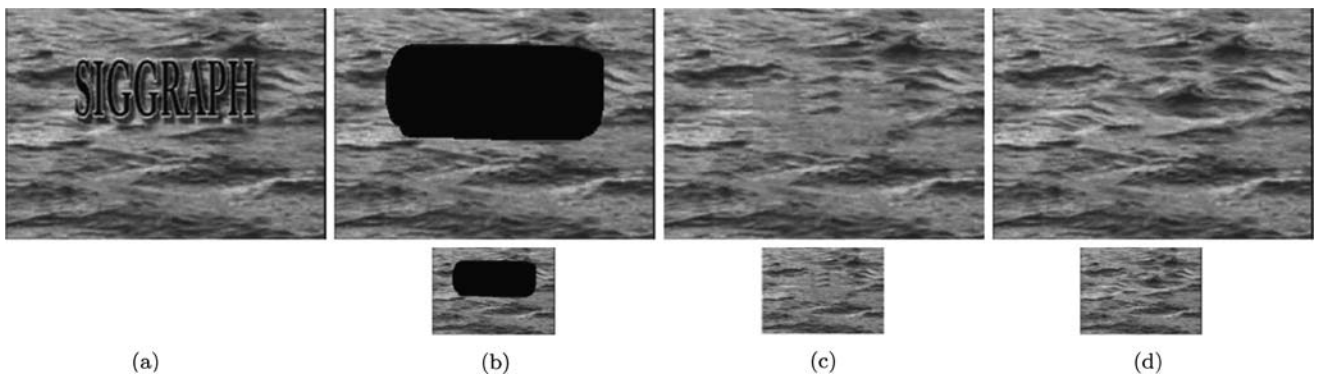


Fig. 12 Video completion. The 85th frame of the video (240 × 180 × 119). (**a**) The original video, (**b**) masked video and its downsampled video (80 × 60 × 119), (**c**) completion result and its upsampled result using (**a**) as the priority, (**d**) completion result that incorporate the detailed layer, and the result is upsampled using original full solution masked video as the priority

techniques presented in [31], then we can apply our Fast JBTU to efficiently perform upsampling.

Our detail-aware patch similarity matching can also be used in video texture synthesis, as illustrated in Fig. 12. Similarly to the image case, the 3D patch matching also incorporates the detail channel, and histogram matching is also implied in the video texture optimization procedure.

8 Experimental results and discussions

We give the experimental results of our methods and compare with the popular upsampling method such as Gaussian upsampling. The results include image and video synthesis upsampling, video completion upsampling. We implemented our approach in C++ on a 2.8 GHz Pentium 4 PC with 1 GB of RAM.

In the M-step of the texture optimization, it has to find the best matching exemplar neighborhood for each neighborhood in the context region. This is a standard nearest neighbor search in a high-dimensional space, and it dominates the running time of the texture optimization. For neighborhood with large data (e.g. 3D neighborhood for video texture synthesis) we apply a PCA projection to the neighborhood vectors in the exemplar [15]. We keep only the number of coefficients sufficient to preserve 95 percent of the variance; thus, the dimensionality reduction drastically improves performance for nearest neighbor search. In all the results, we use the ANN approximate nearest neighbor library [27] for the nearest neighbor search.

In Fig. 9, we present the texture optimization results that incorporate detailed layer, and compare our method with the latest texture method [10, 13] using optimization. In Fig. 9, the result of [13], which employs color histogram matching, looks quite similar to the exemplar, but neither of these two methods [10, 13] is good at maintaining the continuity of structural features as well as the shapes of individual objects in the textures. In bottom row of Fig. 9, it shows that if the color information of the exemplar is similar, the color histograms matching [13] do not work well, while our detail-aware patch similarity matching and detail histograms matching generate better results.

In Fig. 10, we compare our hierarchical JBTU method and multi-scale JBTU with the Gaussian upsampling and bicubic upsampling. Notice that compared with our method, Gaussian upsampling produces much more blurred results. In our experiments, we generally set the domain filter's Gaussian σ_d of the joint bilateral filtering operator to 0.5 with 5×5 support. The range filter Gaussian σ_r is strongly application-dependent. For the images with color values normalized to the [0,1] interval, setting σ_r to the standard deviation of the values has always given good results. In Fig. 10, it takes about 0.6 s to upsample the synthesized texture (128×128) to the result (512×512) using the multi-scale JBTU.

We give multi-scale joint video texture upsampling results in Fig. 11. As the exemplar video texture is large, we downsample it into a smaller one, then we apply the hierarchical video texture upsampling to generate a large video texture. Compared with image texture upsampling case, video upsampling is much more time consuming, it takes about 2 min to upsample the low resolution video texture ($150 \times 70 \times 90$) to a larger one ($350 \times 160 \times 90$). As illustrated in Fig. 11, compared with the upsampled results using Gaussian interpolation, video blurring artifact is better prevented using JBTU method, and video texture is further enhanced applying multi-scale JBTU.

We also present a video completion example with large data set (Fig. 12). This example is difficult to process by traditional texture optimization method due to memory and computation constraints. In Fig. 12, we first downsample the masked video, and complete the low resolution video using the optimization-based space–time video completion method [12], then the completed video is upsampled to the full solution video using the original masked video as the priority. In addition, in this example, we also present a low resolution video completion result that incorporates the video detailed layer; as show in Fig. 12(d), the salient structure is better reconstructed.

Limitation: Our multi-scale JBTU applies base-detail decomposition technique which is based on the bilateral filtering. However, the currently used bilateral filtering is limited in its ability to extract detail at arbitrary scales, which makes it difficult to capture all the high-frequency content of the image. Recently, Farbman et al. [32] proposed an edge-preserving smoothing operator based on a weighted least squares optimization framework. This method is well suited for progressive coarsening of image and for multi-scale detail extraction; however, this method is time consuming, especially for large data video. The second limitation is that, although our accelerated JBTU is fast to upsampling the image texture, to upsample the video texture, especially for large data video texture, the results cannot be interactively generated.

9 Conclusion and further work

In this paper, we propose a novel technique for upsampling synthesized texture using multi-scale joint bilateral filter, which is an efficient method for both image and video texture to better prevent image blurring. To accelerate the upsampling process, we present an accelerated joint bilateral filtering method, which makes our upsampling operator interactive even for large images. We also propose a detail-aware texture optimization approach to improve the optimization process.

In the future, more sophisticated work can be performed to further improve the quality and the speed of the texture upsampling. Instead of using uniformed joint bilateral filtering, to upsampling the video with moving objects, we can use the adaptive bilateral filter, that is, using dynamic function of both the spatial neighborhood and temporal history at each pixel, which makes a smoother result for consequent frames. We would like to find a more effective detail delayer extraction method that can control the spatial variation of

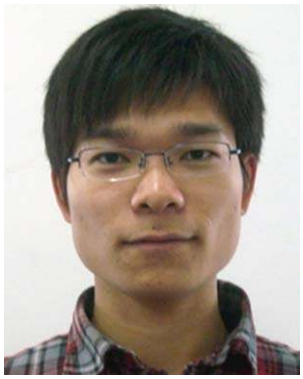
detail for better video texture synthesis and upsampling, especially for video with moving objects. Finally, we would extend our texture upsampling to texture synthesis methods.

Acknowledgements This work was partly supported by NSFC (No. 60803081), State Key Lab of CAD&CG (No. A0808), State Key Laboratory of Software Engineering (SKLSE) (No. SKLSE2008-07-08), Ph.D. Programs Foundation of Ministry of Education of China (No. 200804861038), Natural Science Foundation of Hubei Province (2008CDB350), National High Technology Research and Development Program of China (863 Program) (No. 2008AA121603).

References

1. Heeger, D.J., Bergen, J.R.: Pyramid-based texture analysis/synthesis. In: ACM SIGGRAPH 95 (August), pp. 229–238 (1995)
2. Portilla, J., Simoncelli, E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. Int. J. Comput. Vis. **40**(1), 49–70 (2000)
3. De Bonet, J.S.: Multiresolution sampling procedure for analysis and synthesis of texture images. In: SIGGRAPH 1997, pp. 361–368 (1997)
4. Efros, A., Leung, T.: Texture synthesis by non-parametric sampling. In: International Conference on Computer Vision 1999, pp. 1033–1038 (1999)
5. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH 2000, pp. 479–488 (2000)
6. Praun, E., Finkelstein, A., Hoppe, H.: Lapped textures. In: SIGGRAPH 2000, pp. 465–470 (2000)
7. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of SIGGRAPH 2001, Los Angeles, CA, pp. 341–346 (2001)
8. Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. ACM Trans. Graph. **20**(3), 127–150 (2001)
9. Kwatra, V., Schodl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. ACM Trans. Graph. **22**(3), 277–286 (2003)
10. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. ACM Trans. Graph. **24**(3), 795–802 (2005)
11. Xiao, C., Zheng, W., Miao, Y., Zhao, Y., Peng, Q.: A unified method for appearance and geometry completion of point set surfaces. Vis. Comput. **23**(6), 433–443 (2007)
12. Wexler, Y., Shechtman, E., Irani, M.: Space–time completion of video. IEEE Trans. Pattern Anal. Mach. Intell. **29**(3), 463–476 (2007)
13. Kopf, J., Fu, C.W., Cohen-Or, D., Deussen, O., Lischinski, D., Wong, T.T.: Solid texture synthesis from 2d exemplars. In: ACM SIGGRAPH 2007, pp. 21–29 (2007)
14. Wu, Q., Yu, Y.: Feature matching and deformation for texture synthesis. ACM Trans. Graph. **23**(3), 364–367 (2004) (Proc. SIGGRAPH 2004)
15. Lefebvre, S., Hoppe, H.: Appearance-space texture synthesis. ACM Trans. Graph. **25**(3), 541–548 (2006) (Proc. SIGGRAPH 2006)
16. Zhang, J., Zhou, K., Velho, L., Guo, B., Shum, H.-Y.: Synthesis of progressively-variant textures on arbitrary surfaces. In: ACM SIGGRAPH 2003, pp. 295–302 (2003)
17. Porikli, F.: Constant time O (1) bilateral filtering. In: CVPR 2008, pp. 1–8 (2008)
18. Wei, L., Han, J., Zhou, K., Bao, H., Guo, B., Shum, H.-Y.: Inverse texture synthesis. ACM Trans. Graph. **27**(3), 52 (2008) (Proc. SIGGRAPH 2008)
19. Fattal, R.: Image upsampling via imposed edge statistics. ACM Trans. Graph. (TOG) **26**(3) (2007)
20. Su, D., Willis, P.: Image interpolation by pixel-level data-dependent triangulation. Comput. Graph. Forum **23**(2), 189–201 (2004)
21. Aly, H.A., Dubois, E.: Image up-sampling using total-variation regularization with a new observation model. IEEE Trans. Image Process. **14**(10), 1647–1659 (2005)
22. Thvenaz, P., Blu, T., Unser, M.: Image interpolation and resampling. In: Bankman, I. (ed.) Handbook of Medical Imaging, Processing and Analysis, pp. 393–420. Academic Press, San Diego (2000)
23. Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., Toyama, K.: Digital photography with flash and no-flash image pairs. ACM Trans. Graph. **23**(3), 664–672 (2004)
24. Eisemann, E., Durand, F.: Flash photography enhancement via intrinsic relighting. ACM Trans. Graph. **23**(3), 673–678 (2004) (Proc. SIGGRAPH 2004)
25. Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint bilateral upsampling. ACM Trans. Graph. **26**(3), 839–846 (2007)
26. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions. Wiley, New York (1997)
27. Mount, D.M., Arya, S.: Ann: A library for approximate nearest neighbor searching (2006). Available at: http://www.cs.umd.edu/~mount/ANN/
28. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Sixth International Conference on Computer Vision, 1998, pp. 839–846 (1998)
29. Fattal, R., Agrawala, M., Rusinkiewicz, S.: Multiscale shape and detail enhancement from multi-light image collections. ACM Trans. Graph. **26**(3) (2007)
30. Durand, F., Dorsey, J.: Fast bilateral filtering for the display of high-dynamic-range images. ACM Trans. Graph. 257–266 (2002) (Proc. SIGGRAPH 2002)
31. Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. ACM Trans. Graph. **26**(3) (2007)
32. Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. In: SIGGRAPH 2008, ACM, New York (2008)

Chunxia Xiao received the Ph.D. from the State Key Lab of CAD& CG, Zhejiang University, in 2006, and is currently an Associate Professor at Computer School, Wuhan University, China. During October 2006–April 2007, he worked as a Postdoc at the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology. His research interests include image and video precessing, digital geometry processing and computational photography.



Yongwei Nie is an undergraduate student at Computer School, Wuhan University, China. His research interests include image and video editing, computational photography.

Wei Hua is an Associate Professor of State Key Lab of CAD&CG at Zhejiang University, China. His research interests include designing image-based modeling tools, real-time graphics systems and virtual reality applications.

Wenting Zheng was born in 1974. He received a Ph.D. from the State Key Lab of CAD&CG, Zhejiang University in 1999, and is currently an Associate Professor at the State Key Lab of CAD&CG, Zhejiang University, China. His research interests include computer graphics, virtual reality and programmable graphics hardware.