Mesh-Guided Optimized Retexturing for Image and Video

Yanwen Guo, Hanqiu Sun, *Member*, *IEEE*, Qunsheng Peng, and Zhongding Jiang, *Member*, *IEEE*

Abstract—This paper presents a novel approach for replacing textures of specified regions in the input image and video using stretch-based mesh optimization. The retexturing results have the similar distortion and shading effect conforming to the unknown underlying geometry and lighting conditions. For replacing textures in a single image, two important steps are developed: The stretch-based mesh parameterization incorporating the recovered normal information is deduced to imitate perspective distortion of the region of interest; the Poisson-based refinement process is exploited to account for texture distortion at fine scale. The luminance of the input image is preserved through color transfer in YCbCr color space. Our approach is independent of the replaced textures. Once the input image is processed, any new textures can be applied to efficiently generate the retexturing results. For video retexturing, we propose key-frame-based texture replacement extended and generalized from the image retexturing. Our approach repeatedly propagates the replacement results of key frames to the rest of the frames. We develop the local motion optimization scheme to deal with the inaccuracies and errors of robust optical flow when tracking moving objects. Visibility shifting and texture drifting are effectively alleviated using graphcut segmentation algorithm and the global optimization to smooth trajectories of the tracked points over temporal domain. Our experimental results showed that the proposed approach can generate visually pleasing results for retextured images and video.

Index Terms—Texture replacement, parameterization, Poisson equation, graphcut segmentation.

1 Introduction

Editing contents of photos/footages by changing the appearances of some regions with new textures is a common task for creating visual effects. This process is commonly referred to as retexturing or texture replacement. The key issue of texture replacement is how to preserve the original shading effect and texture distortion without knowing the underlying geometry and lighting conditions. Retexturing objects of images and video clips has wide applications in digital entertainment, virtual exhibition, art, and industry design.

For retexturing image, two fundamental issues must be addressed: how to deform the new texture for conforming to scene geometry and how to keep shading effect encoded in the original image for consistent lighting conditions. One possible solution to the first issue is recovering 3D surface geometry using shape-from-shading techniques, then establishing parameterization between the surface and the texture. Unfortunately, shape-from-shading techniques for a single image cannot accurately recover the 3D geometry with high efficiency. Even with multiple images, full recovery of 3D geometry is still an open problem in the computer vision community. For the second issue, relighting techniques can be adopted to change intensities of pixels of the new texture when properties of light sources and surface appearances are known beforehand. However, accurate recovery of these properties from a real-world image is more difficult than geometry recovery. Hence, relighting techniques are impractical for texture replacement.

For generating plausible visual effects, full recovery of 3D geometry and lighting conditions can be relaxed in practice. Fang and Hart proposed one normal-guided texture synthesis method that produced compelling replacement effects [1]. This method works well when a 3D surface is untextured, nearly diffuse, and illuminated by a single directional light source. One limitation of this texture synthesis approach is that the synthesis process must be repeated when a new texture is applied. For regular/near-regular textures that are popular in the real world, Liu et al. suggested an interactive scheme to extract the deformation field of texture image with respect to the original sample [2]. The extraction of lighting information can also benefit from this restriction through a Markov process [3]. Nevertheless, this approach usually needs tedious user interaction with high accuracy.

Video clip is an image sequence in time domain, which usually contains dynamic objects and lighting changes. Retexturing video is more challenging than retexturing image due to these dynamic phenomena. In particular, keeping the texture coherence over time is more challenging. The temporal coherence demands that the new texture should be perceptually fixed on 3D surface when an object or camera moves. For achieving temporal coherence, the key-frame-based methods [4], [5] consist of two steps. First, a few of video frames are selected as key frames on which

- *Y. Guo is with the National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, P.R. China. E-mail: ywguo@nju.edu.cn.*
- *H. Sun is with the Department of Computer Science and Engineering, the Chinese University of Hong Kong, Shatin, N.T., Hong Kong. E-mail: hanqiu@cse.cuhk.edu.hk.*
- *Q. Peng is with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, P.R. China. E-mail: peng@cad.zju.edu.cn.*
- *Z. Jiang is with the Software School, Fudan University, Shanghai, 201203, P.R. China. E-mail: zdjiang@fudan.edu.cn.*

Manuscript received 26 Dec. 2006; revised 11 July 2007; accepted 27 Aug. 2007; published online 17 Sept. 2007.
Recommended for acceptance by J. Dorsey.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0231-1206.
Digital Object Identifier no. 10.1109/TVCG.2007.70438.

texture replacements are conducted [4], [5]. Second, the generated results are propagated to the rest frames. These methods either need cumbersome interaction to locate the region of interest (ROI) frame by frame [4] or utilize special color-coded pattern as input [5].

In this paper, we propose one novel approach for optimized retexturing on image and video. For image texture replacement, we formulate texture distortion as one stretch-based parameterization. The ROI is represented as a feature-mesh coupled with normal field. The corresponding mesh of the feature-mesh is computed in texture space during parameterization. For simulating the effect of texture deformation at fine scale, one Poisson-based refinement process is developed. Based on our image-retexturing scheme, we design one key-frame-based video retexturing approach similar to RotoTexture [4]. Once replacing textures of the specified regions of key frames, these generated effects are iteratively transferred to the rest frames. For achieving temporal coherence, mesh points of key frame serve as features that are tracked and further optimized using motion analysis as well over the whole image sequence through temporal smoothing. Graphcut segmentation is adopted for handling object occlusions by extracting new appearing parts in each frame.

Our optimized retexturing approach has the following new features:

- *A two-step mesh guided process for image texture replacement.* Coupled with recovered normal field, visually pleasing deformation effect of the replaced texture is produced by performing stretch-based mesh parameterization. Furthermore, a Poisson-based refinement process is used to improve the effect and enhance the efficiency.
- *Creation of special retexturing effects.* Based on mesh parameterization, we can easily generate a replacement effect with progressively variant texton scales. In addition, texture discontinuities can be realistically simulated in self-occlusion regions, which are usually difficult to produce for most previous approaches.
- *An optimized framework of video retexturing.* We extend and generalize our image retexturing approach to video. Rather than presegmenting the ROI throughout the whole image sequence, our approach only needs to select a few of key frames. The generated results are optimally propagated to the rest frames. Texture drifting and visibility shifting are also tackled effectively.

The rest of the paper is organized as follows: The related work is described in Section 2. Our optimized retexturing approach for image is presented in Section 3. In Section 4, the image retexturing approach is extended and generalized to video. The experimental results are presented in Section 5. Finally, we draw conclusions and point out the future work.

2 Related Work

This paper is made possible by many inspirations from previous work on image and video texture replacement. Since tracking objects throughout video sequence is one important step of our approach, it is briefly reviewed.

2.1 Image Texture Replacement

The pioneering work on texture replacement dealt with extracting lighting map from given image [3]. Based on certain lighting distribution models, Tsin et al. introduced one Bayesian framework for near-regular texture, which relies on the color observation at each pixel [3]. Oh et al. assumed that large-scale luminance variations are due to geometry and lighting [6] and presented an algorithm for decoupling texture luminance from image by applying an improved bilateral filter. Currently, accurate recovery of lighting from natural image is still a challenging problem.

Assuming that object appearance satisfies the Lambertian reflectance model, Textureshop [1] recovered normal field of specified region using a simplified shape-from-shading algorithm [7]. One propagation rule of adjacent texture coordinates is deduced to guide a normal-related synthesis process. The limitation of employing texture synthesis is that the synthesis process must be re-executed when a new texture is applied. The work developed by Zelinka et al. [8] is an improvement over Textureshop. It reduces user interaction of object specification by employing efficient object cutout algorithm [9]. In addition, jump map-based synthesis [10] is adopted to speed up the computation process. Instead of texture synthesis, our method belongs to the texture mapping approach. Texture replacement can be efficiently carried out after the mesh parameterization is completed.

The method presented in [4] warps an entire texture onto photographed surface. It minimizes one energy function of a spring network with known evenly distributed rectilinear grid in texture space. In most cases, the specified region of the image is a usually irregular grid. Hence, it is difficult for this approach to accurately control the mapping position of the replaced texture.

For extracting deformation fields of textures in natural images, Liu et al. introduced one user-assisted adjustment scheme on the regular lattice of real texture [2]. A bijective mapping between the regular lattice and its deformed shape on the surface image is obtained. Any new texture can thus be replaced onto the source image by exerting the corresponding mapping. Since this method often requires elaborate user interaction, it is more suitable for regular/near-regular textures.

Besides image texture replacement, recent research demonstrated that material properties of objects can be changed in image space [11]. Exploiting the fact that human vision is surprisingly tolerant of certain physical inaccuracies, Khan et al. reconstructed depth map of the concerned object with other environment parameters [11] and realized compelling material editing effects using complex relighting techniques.

2.2 Video Texture Replacement

Rototexture [4] generalized the method of Textureshop [1] to video. It provides two means of texturing a raw video sequence, namely, texture mapping and texture synthesis. The texture mapping method uses one nonlinear optimization of a spring model to control the behavior of texture

image that is deformed to match the evolvement of normal field throughout the video. For the synthesis method, the minimum advection tree is constructed to deal with the visibility issue due to dynamic motions of moving objects. Such tree determines the initial frame for each image cluster and the advection for clusters among frames. The main challenge of video texture replacement is how to stably track the moving objects and their interior regions. At present, accurately tracking moving objects of dynamic video is an open problem. The replaced textures drift in the experimental results [4].

For stably tracking moving objects and their interior parts, Scholz and Magnor presented one system of video texture replacement [5] using color-coded patterns. The deformation process of the texture throughout video clip can be accurately extracted. Since the deformation is accurate, compelling results can be achieved. However, videos are usually captured by off-the-shelf camera without the color-coded patterns, the system is not applicable to them. Our approach is designed for those videos in which the special patterns are unavailable.

Recently, White and Forsyth proposed another video retexturing method [12]. At coarse scale, old texture is replaced with a new one by tracking deforming surface in 2D. At fine scale, local irradiance is estimated to preserve the structure information in real lighting environment. Since local irradiance estimation is difficult and unreliable, the approach is limited to screen printing with a finite number of colors. Our method can be applied to video sequences with rich color details.

2.3 Object Tracking

Object tracking is the process of locating moving object throughout the whole image sequence taken by video camera. For general object motion, the nonparametric algorithm such as optical flow [13] can be applied. When the motion can be described using simple models, methods based on feature points and parametric models are more preferable [14]. For instance, Jin et al. presented one combined model of geometry and photometry to track features and detect outliers in video [15]. Contour tracking can be more effective for nonrigid objects than isolated point tracking. Agarwala et al. [16] and Wang et al. [17] introduced frameworks for tracking contours of moving objects in video sequences, which are based on spatiotemporal optimization and user assistance. Chuang et al. described one method of accurately tracking specified trimap for video matting [18]. A trimap is one labeling image in which 0 stands for background, 1 stands for foreground, and the rest is the unknown region to be labeled. Stably tracking of the trimap is carried out based on robust optical flow algorithm [19].

3 Image Retexturing

The key issue of image texture replacement is how to preserve the distortion effect of texture, as well as the shading effect encoded in the original image. Texture distortion is mainly caused by the undulation of underlying surface of object in the given image. Assume that the surface where texture replacement is performed on, is nearly developable. Otherwise, the surface can be divided into several nearly developable parts, each of them is handled using the texture replacement. Based on this assumption, the basic idea is converting reconstruction of the underlying 3D surface of ROI into computation of its corresponding mesh in texture space. Using projective geometry, we further formulate the retexturing task as a stretch-based mesh parameterization problem. After the parameterization is completed, the result is further refined with one Poisson-based refinement process.

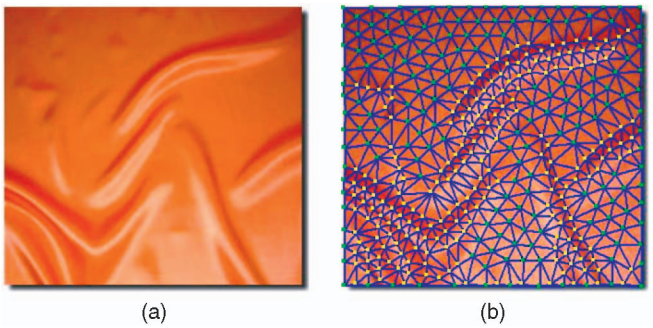


Fig. 1. Mesh generation. (a) The input image. (b) The generated mesh. The yellow dots are detected by Canny operator, whereas the green ones are added automatically with a distance threshold to maintain mesh uniformity.

3.1 Mesh Generation

We first generate an initial mesh on the concerned region and make its shape consistent with the underlying geometry of this region. Mesh generation for image was addressed in motion compensation for video compression [20]. The content-based mesh was computed by extracting a set of feature points followed by Delaunay triangulation.

Our algorithm as follows shares the same idea in [20]. First, the concerned region is specified interactively by outlining the boundary of ROI using snakes. For reducing user intervention, our approach supports extracting ROI using the up-to-date segmentation techniques [21], [9]. Second, we employ the edge detection operator, for example, Canny operator, for extracting some feature points inside ROI. For keeping uniformity of the points, some auxiliary ones are usually added. Finally, the constrained Delaunay triangulation algorithm is adopted to generate a feature-consistent mesh. Fig. 1 shows an example of mesh generation.

3.2 Mesh parameterization

Let $\mathcal{M}$ denote the generated mesh of input image. $\mathcal{M}$ is one 2D mesh that represents the 2D projection of 3D surface of ROI. If normal vector of every mesh point of $\mathcal{M}$ is recovered, the normal field of $\mathcal{M}$ will encode the geometry shape of the underlying surface. For obtaining the distortion effect of the new texture, it is feasible to first parameterize $\mathcal{M}$, then map the new texture onto ROI. Since $\mathcal{M}$ is one 2D mesh, parameterizing $\mathcal{M}$ onto the texture space is 2D-to-2D, which can be computed using the geometry information of $\mathcal{M}$.

Let $\mathcal{M}'$ be the parameterized mesh in texture space. Theoretically, $\mathcal{M}'$ can be completely determined by lengths of all edges and topology of $\mathcal{M}$. For avoiding artifacts, the topology of $\mathcal{M}'$ should be the same as that of $\mathcal{M}$. The length of each edge in $\mathcal{M}'$ should be ideally equal to the 3D length

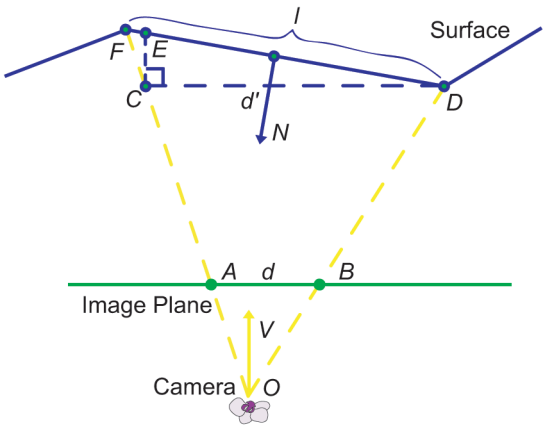


Fig. 2. Calculation of the 3D length $\|FD\|$ for the observed edge $e(AB)$. V: view direction, N: the projection of edge $e(FD)$'s normal vector on the plane OAB, $CD//AB$, and $CD \perp CE$.

of its corresponding edge in $\mathcal{M}$. The 3D length reflects the edge length on the underlying surface. In the following sections, we first introduce our method of computing the 3D length of each edge in $\mathcal{M}$, then present the stretch-based parameterization scheme for computing $\mathcal{M}'$ in texture space.

3.2.1 Computing Length of 3D Edge

Since normal field encodes the 3D geometry information of $\mathcal{M}$, we first recover one rough normal field of $\mathcal{M}$ using the approach in [1]. The shape-from-shading algorithm in [1] has a linear complexity when recovering the normal field for diffuse surface. It is easy to implement and quite effective. For more details, please refer to [1] and [8].

With the recovered normal vector, 3D length of each edge in $\mathcal{M}$ can be calculated, as illustrated in Fig. 2. Suppose the observed length of edge $e(AB)$ in the input image is d. Its corresponding 3D length is the distance between F and D on the underlying surface. The length is denoted as $\|FD\|$.

Let N be the projection of normal vector of edge $e(FD)$ on the plane OAB. It can be calculated first by averaging the recovered normal vectors of A and B, then projecting the average vector onto plane OAB. From the geometry relationship in Fig. 2, we derive the length of $\|ED\|$:

$$\|ED\| = \|CD\|/(V \cdot N) = d'/(V \cdot N), \tag{1}$$

where V is the view direction of camera, both V and N have been normalized. With an approximation $\|FD\| \doteq \|ED\|$, the 3D length of edge $e(AB)$ can be expressed as follows:

$$l = \|FD\| \doteq d'/(V \cdot N). \tag{2}$$

In the above equation, d' is determined by the observed length d of $e(AB)$ and scene depth of $e(FD)$. In most cases, the object is relatively far from camera. It is reasonable to assume that the scene depths for all edges in $\mathcal{M}$ are close enough with small variation. Consequently, the homogeneous scale factor of every edge of $\mathcal{M}$ can be eliminated. The surface length of edge $e(AB)$ can then be approximated as

$$l = \|FD\| \doteq \|AB\|/(V \cdot N) = d/(V \cdot N). \tag{3}$$

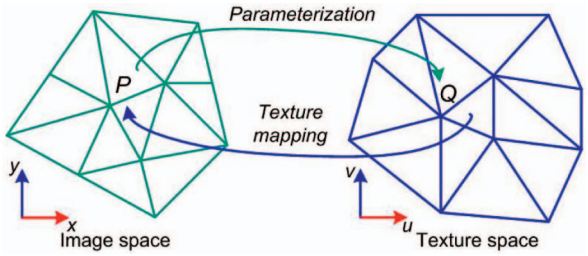


Fig. 3. In virtue of the normal field, simulating texture distortion is converted into one 2D-to-2D parameterization, after which texture mapping is then applied.

The 3D length of each edge in $\mathcal{M}$ can be figured out using the above equation. Although some approximations are made, our experimental results show that the method can generate visually pleasing effects.

3.2.2 The Stretch-Based Parameterization

With the calculated edge lengths, $\mathcal{M}'$ can be obtained by one stretch-based parameterization method.

Mesh parameterization has been extensively studied in computer graphics [22], [23], [24], [25], [26], [27], [28], and [29]. These methods take different metric criterions as the objective of energy minimization process. Among them, stretch-based methods work well when reducing the global mesh distortion [22], [25], [29]. Our concern is computing one 2D-to-2D parameterization, which is different from the aforementioned 3D-to-2D process (Fig. 3).

We first describe some related notations. Let $\{P_i = (x_i, y_i)|i = 1, \ldots, n\}$ denote the nodes of $\mathcal{M}$. $\{Q_i = (u_i, v_i)|i = 1, \ldots, n\}$ represent the corresponding nodes of $\mathcal{M}'$ to be solved in texture space. $\mathcal{M}'$ has the same topology as $\mathcal{M}$, Q_i corresponds to P_i, and edge $e(Q_iQ_j)$ corresponds to $e(P_iP_j)$. The 3D length l_{ij} of $e(Q_iQ_j)$ is obtained using the aforementioned method.

$\mathcal{M}'$ is completely characterized by the lengths of its edges. As each edge of $\mathcal{M}'$ has been figured out, $\mathcal{M}'$ can be computed by minimizing the following energy function:

$$E_l = \sum_{(i,j)\in edges} \left(\|Q_i - Q_j\|^2 - l_{ij}^2\right)^2 \Big/ l_{ij}^2. \tag{4}$$

Exploiting the symmetry of the above equation, energy gradients on point Q_i are

$$\frac{\partial E_l}{\partial u_i} = 8 \sum_{(i,j)\in edges} \left(\|Q_i - Q_j\|^2 - l_{ij}^2\right)^2 \cdot (u_i - u_j)/l_{ij}^2, \tag{5}$$

$$\frac{\partial E_l}{\partial v_i} = 8 \sum_{(i,j)\in edges} \left(\|Q_i - Q_j\|^2 - l_{ij}^2\right)^2 \cdot (v_i - v_j)/l_{ij}^2. \tag{6}$$

When $\mathcal{M}$ is dense, directly solving the above equations may cause adjacent triangles of $\mathcal{M}'$ to flip over, leading to invalid topology. This is caused by inverting the orientation of the three points of triangle. To tackle this issue, we revise the energy function by penalizing the orientations of triangles using the sgn function [30].

Assume that the adjacent triangles incident upon edge $e(Q_iQ_j)$ are $T_{Q1} = T(Q_iQ_{k1}Q_j)$ and $T_{Q2} = T(Q_iQ_jQ_{k2})$. Their corresponding triangles in the input image are

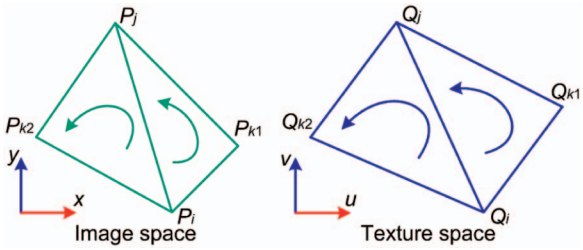


Fig. 4. The orientation of the triangles (a) in image space should keep consistent with that of their corresponding ones (b) in texture space.

$T_{P1} = T(P_iP_{k1}P_j)$, and $T_{P2} = T(P_iP_jP_{k2})$ (Fig. 4). For each pair of corresponding triangles, the orientations of points should be equal. To achieve this, we define

$$w_{ij} = sgn\ min\Big(det\Big(\overrightarrow{Q_iQ_{k1}}, \overrightarrow{Q_jQ_{k1}}\Big) \cdot det\Big(\overrightarrow{P_iP_{k1}}, \overrightarrow{P_jP_{k1}}\Big), det\Big(\overrightarrow{Q_iQ_{k2}}, \overrightarrow{Q_jQ_{k2}}\Big) \cdot det\Big(\overrightarrow{P_iP_{k2}}, \overrightarrow{P_jP_{k2}}\Big)\Big). \tag{7}$$

The energy function is then transformed into

$$E_l = \sum_{(i,j)\in edges} \Big(w_{ij} \cdot \|Q_i - Q_j\|^2 - l_{ij}^2\Big)^2 \Big/ l_{ij}^2, \tag{8}$$

where the coefficient w_{ij} penalizes the triangle in $\mathcal{M}'$ whose orientation of points flips over with respect to its corresponding triangle in $\mathcal{M}$. If so, w_{ij} is chosen as -1, otherwise, $+1$. With this energy function, a valid mesh in texture space is obtained.

The minimal value of (8) is computed by the multidimensional Newton's method. For each iteration of Newton's method, one multigrid solver is adopted for solving the sparse linear equations. In practice, it converges to the final solution within several seconds.

Once $\mathcal{M}'$ is obtained with the parameterization process, a new texture can be mapped onto the ROI of the input image. Since the parameterization takes into account the underlying geometry of ROI, the new texture deforms naturally with respect to the underlying surface. Our experimental results demonstrate that the distortion effects of the new textures are visually pleasing.

3.3 Poisson-Based Refinement

After the stretch-based parameterization, texture coordinates of the nodes in $\mathcal{M}$ have been obtained. Texture coordinates of the interior pixels of triangles in $\mathcal{M}$ can be computed by interpolating the obtained ones using barycenter coordinates or the Radial Basis Functions (RBFs). However, such interpolation techniques cannot reflect the distortion effect of the new texture in the interior of each triangle. For obtaining natural and smoother distortion, we design a Poisson-based refinement process instead of using interpolation techniques with barycenter coordinates or the RBFs.

The origin of Poisson equation is from Isaac Newton's laws of gravitation [31]. It has been widely used in computer graphics, including seamlessly image editing [32], digital photomontage [33], gradient field mesh manipulation [34], and mesh metamorphosis [35]. The main principle of Poisson equation lies in how to compute the interior values of the function with known Dirichlet boundary condition and a guidance vector field. Due to its sparse linear property, the Poisson equation can be solved efficiently using conjugate gradient method or multigrid method [34].

Our Poisson-based algorithm adopts the normal-guided propagation rules for texture synthesis developed in Textureshop [1]. These rules describe the offsets of texture coordinates among adjacent pixels. We rewrite them as follows:

$$u(x+1,y) - u(x,y) = f_{ux}(N(x,y)), \tag{9}$$

$$v(x+1,y) - v(x,y) = f_{uv}(N(x,y)), \tag{10}$$

$$u(x,y+1) - u(x,y) = f_{uv}(N(x,y)), \tag{11}$$

$$v(x,y+1) - v(x,y) = f_{vy}(N(x,y)), \tag{12}$$

where $(u(x,y), v(x,y))$ is the texture coordinate of the pixel (x,y) in the concerned region, $N(x,y) = (N_x, N_y, N_z)$ is the normal vector at pixel (x,y), and

$$f_{ux}(N(x,y)) = (1 + N_z - N_y^2)/((1+N_z)N_z), \tag{13}$$

$$f_{uv}(N(x,y)) = (N_xN_y)/((1+N_z)N_z), \tag{14}$$

$$f_{vy}(N(x,y)) = (1 + N_z - N_x^2)/((1+N_z)N_z). \tag{15}$$

Since the texture coordinates of nodes in $\mathcal{M}$ are available, (9)-(12) can be used directly to calculate texture coordinates of the interior pixels of triangles in $\mathcal{M}$. In practice, this may result in a wired mapping. For avoiding it, the texture coordinates are obtained by solving the energy minimization problem below with respect to the u component. The v component can be computed in a similar way:

$$min_{u(x,y)} \int_{\mathcal{M}} |\nabla u(x,y) - D_u(x,y)|^2, \tag{16}$$

where $\nabla u(x,y) = (u(x+1,y) - u(x,y), u(x,y+1) - u(x,y))$, and $D_u(x,y) = (f_{ux}(N(x,y)), f_{uv}(N(x,y))$.

Minimizing (16) yields a set of Poisson equations:

$$\Delta u(x,y) = \mathrm{div} D_u(x,y), \tag{17}$$

where Δ and div represent the Laplacian and divergence operators, respectively. We adopt a multigrid solver to obtain the solution with high efficiency.

Unlike the widely used Dirichlet Boundary conditions [32], [34] of the generic Poisson process, the external force in our Poisson equations is imposed by the discrete texture coordinates of nodes of $\mathcal{M}$.

3.4 Lighting Effect Transfer

Using texture coordinates deduced by the stretch-based parameterization and Poisson-based refinement process, a new texture is mapped onto ROI of the input image to overwrite the old one. Due to the lack of simulating lighting effect exhibited in ROI, the mapping result looks flattening. For realistic appearance, transferring the lighting effect must be considered.

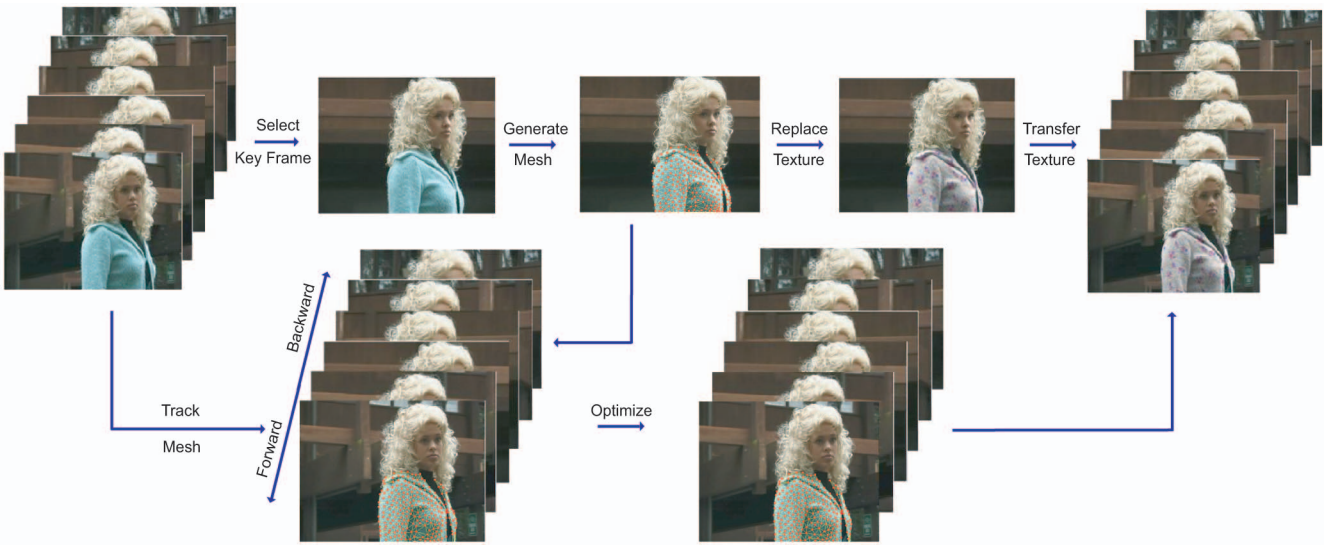


Fig. 5. Video retexturing pipeline. Retexturing is first applied to each key frame selected from the input sequence. The mesh generated for the key frame is tracked backward/forward throughout the video. After a temporal smoothing optimization, the replaced texture of the key frame is transferred onto the rest frames. In the above process, the lighting effect is transferred independently for each frame.

Accurate recovery of the lighting conditions of a real-world image is an open problem in computer vision. Researchers developed methods [3], [6] that are tailored for planar patterns exhibited in an image. Although luminance can be extracted from curved surfaces [2], the method needs one regular texture as reference. The general solution of decoupling illuminance from image demands further investigations.

Based on the above analysis, an approximate method is designed to transfer the lighting effect for texture replacement. In color theory, the intensity of each pixel can be regarded as one combination of chromaticity and luminance. $YCbCr$ is one color space described by luminance and chromaticity. The Cb and Cr components represent the chromaticity, and Y component encodes the luminance. This color space is widely employed in video systems.

For texture replacement, we transform the color of each pixel into $YCbCr$ space. The Y component of each pixel inside ROI of input image is fused with the Y component of the corresponding pixel in the new texture. This scheme preserves luminance of the lighting effect. For transferring chromaticity, the Cb, Cr components of each pixel inside ROI are overwritten by the Cb, Cr components of the corresponding pixel of the new texture.

Let Y_t, Cb_t, Cr_t, Y_i, Cb_i, Cr_i, and Y_r, Cb_r, Cr_r represent the Y, Cb, Cr components of the pixel color in the new texture, its corresponding one in the input image, and the one in the replacement result. The above rules of transferring light effect can be expressed as follows:

$$Cb_r = Cb_t, \tag{18}$$

$$Cr_r = Cr_t, \tag{19}$$

$$Y_r = (1 - m_t) \cdot Y_t + m_t \cdot Y_i, \tag{20}$$

where m_t is one weight factor balancing the luminance between the new texture and the input image. Increasing m_t will make the luminance component of the replacement result more similar to that of the input one. Through extensive experiments, it is set to 0.9 for results in Section 5.

Our lighting preservation method is unsuitable for input images with strong textures. For these images, the luminance components are dominated by contributions from textures instead of from lighting effects. Fortunately, our approach separates the task of image replacement into two independent tasks: simulating texture distortion and preserving lighting effect. It allows other lighting preservation method [3] to be incorporated for enhancing the results.

4 Video Retexturing

Texture replacement of video frames is more complicated than retexturing one image. For retexturing video, applying the image retexturing approach frame by frame will cause the drifting phenomenon. The main reason is that the generated mesh in each frame may be quite different. After stretch-based parameterization and Poisson-based refinement process, the texture coordinate computed at each pixel of ROI usually changes over frames. Hence, the region with new texture is flickering when playing the retextured video.

We propose one key-frame based video retexturing approach. Fig. 5 outlines the processing pipeline of our approach. Some key frames are first selected from the input video frames with user interaction. In these selected frames, the concerned objects usually have maximal visibility. Second, the correspondences between key frames are established, which are triangulated into one coarse mesh. The coarse mesh serves as the initial mesh of ROI of key frame that will be refined using the aforementioned mesh generation technique. For each key frame, the image retexturing approach is applied. The replaced results are propagated to other frames repeatedly. For preventing texture drifting, the texture coordinates of mesh points should be tracked accurately in consecutive frames. The generated mesh of ROI of key frame serves as an object that is tracked backward/forward throughout the video using the robust optical flow algorithm [19]. Although the robust

optical flow algorithm is proven to be of better performance [18], [36], the estimated motion field still contains artifacts and errors. Hence, the trajectories of mesh points are further optimized by a temporal smoothing operation. After these steps, the replaced results are transferred from the key frames to the rest frames by taking the texture in each triangle of the tracked mesh as transfer unit.

For video clip containing dynamic objects, visibility shifting often occurs at or near object boundaries. It is generally caused by moving objects, changing viewpoint, and occlusion/disocclusion events. Since it usually leads to variations of the replaced textures, visibility shifting must be dealt with carefully. We design one graphcut-based algorithm for handling visibility shifting.

4.1 Local and Global Optimization for Object Tracking

In our approach, the robust optical flow algorithm [19] is adopted to track the positions of mesh points. Although the robust optical flow algorithm can handle large-scale motion, it cannot handle image changes caused by phenomena that cannot be modeled by optical flow algorithm. Hence, the tracked points may deviate from their correct positions to a certain extent. It is indispensable to optimize the tracking results for avoiding texture drifting. We propose to *locally* analyze the motion information of each mesh point and optimize its position for each frame. Once the whole sequence is processed, the trajectories of mesh points are further refined *globally* throughout the video.

4.1.1 Local Optimization

The optical flow algorithm takes the brightness matching criterion when computing motion field between two frames. The computed motion of each mesh point can be stabilized by considering the geometry and topology of the mesh. For a mesh point, three tracking cases are usually considered as bad tracking:

- *Color inconsistency.* It happens when the color difference of a mesh point between the current frame and reference frame is greater than a given threshold.
- *Motion inconsistency.* It happens when the motion vector of a mesh point differs too much from its neighbors'.
- *Orientation inconsistency.* It occurs when any of the orientations of the triangles incident upon a mesh point is flipped over with respect to its corresponding triangles in the reference frame.

One mesh point is considered to be unstable when any of the three bad-tracking cases occurs. Otherwise, it is a stable one. The position of an unstable point needs to be recalculated to remedy inconsistency. We employ inverse distance interpolation [37] to recompute the motion vector of unstable point using its stable neighbors.

Assume P_0 is an unstable point with neighbors $P_{0j}{}_{j=1,\ldots,k}$. $(t_{x0_j}, t_{y0_j})_{j=1,\ldots,k}$ represent their corresponding motion vectors, and $d_{0j}{}_{j=1,\ldots,k}$ denote their distances from P_0, respectively. The new motion vector of P_0 is computed as follows:

$$t_{x0} = \frac{\sum_{j=1}^{k} \sigma(j) t_{x0_j}/d_{0j}}{\sum_{j=1}^{k} \sigma(j)/d_{0j}}, \tag{21}$$

$$t_{y0} = \frac{\sum_{j=1}^{k} \sigma(j) t_{y0_j}/d_{0j}}{\sum_{j=1}^{k} \sigma(j)/d_{0j}}, \tag{22}$$

with

$$\sigma(j) = \begin{cases} 1 & if\ P_{0j}\ is\ a\ stable\ point \\ 0 & otherwise. \end{cases}$$

If the neighbors of P_0 are unstable, mesh points that are relatively farther are treated as new neighbors for the interpolation process. In addition, the new position of P_0, calculated with the interpolated motion vector, should not destroy the local topology around P_0 with respect to the reference frame. Otherwise, a random searching operation around the new position is performed to find another suitable one.

4.1.2 Global Optimization

The *local* optimization mainly accounts for the shifting points in each frame. The global optimization aims to optimize the mesh points throughout the video sequence to ensure the temporal coherence.

Taking the video sequence as one 3D volume, the motion trajectory of each mesh point throughout the video can be regarded as a piecewise motion curve. For the unstable point, it may *jump* suddenly or frequently along its motion curve. The motion trajectory of unstable point can be refined using the curve smoothing technique.

We generalize the bilateral denoising algorithm [38], [39] to smooth trajectories of all mesh points. Bilateral filter has been successfully applied to remove noise from images/meshes while retaining details [38], [39]. It is intuitive to extend the algorithm from mesh to curve. Before bilateral smoothing is performed, mesh points deviated significantly from the ideal positions are manually dragged near the ideal ones.

Combining the trajectories generated through bilateral smoothing, we compute the final trajectories of all mesh points by solving an energy minimization problem. Suppose the number of video frames is T between two key frames. Let $P_i(t)_{i=1,\ldots,n;t=1,\ldots,T}$ denote the positions of mesh points tracked by robust optical flow with *local* optimization, and $\widetilde{P_i(t)}_{i=1,\ldots,n;t=1,\ldots,T}$ represent the positions refined by bilateral smoothing. $\overline{P_i(t)}_{i=1,\ldots,n;t=1,\ldots,T}$ are the ideal positions, which are solved by minimizing the following objective function:

$$E_t = \sum_{i=1}^{n} \sum_{t=1}^{T} \left[\lambda_i(t) \cdot (\overline{P_i(t)} - P_i(t))^2 + (\overline{P_i(t)} - \widetilde{P_i(t)})^2 \right], \tag{23}$$

with

$$\lambda_i(t) = exp\left(-\left(P_i(t) - \widetilde{P_i(t)} \right)^2 \right) \Big/ (2\sigma_i^2), \tag{24}$$

where σ_i is the standard deviation of the offsets between $P_i(t)_{t=1,\ldots,T}$ and $\widetilde{P_i(t)}_{t=1,\ldots,T}$. λ is one weight factor that takes greater effect on the unstable point. The weight factors are

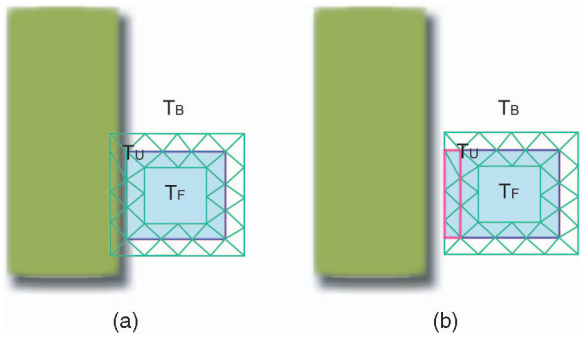


Fig. 6. Trimap is constructed near the object boundary, which is partly occluded by the shelter in the reference frame (a). In the current frame (b), the new appearing textured part in trimap (surrounded by the red rectangle) is extracted by graphcut. This region should be replaced with a new patch of texture.

set to small costs when optimizing (23). Instead of trying to find the global minimum of (23), every mesh point is repeatedly perturbed toward its ideal position with a small incremental step until one local minimum is achieved.

Once obtaining the final positions of mesh points in each frame, the replaced texture adhering to the key frame is iteratively transferred to the other frames. During this process, the retextured effect of each frame incorporates the lighting effect encoded by the current frame using the algorithm described in Section 3.4.

4.2 Graphcut Segmentation for Visibility Shifting

Visibility shifting is one common phenomenon of dynamic video sequence. It may lead to variations of replaced textures. For example, if the concerned object moves into a shelter, its boundary triangles will be reduced because parts of the texture change from visible to invisible. In contrast, when the object moves out, some previously occluded parts will shift from invisible to visible. The shifting parts should be replaced accordingly with patches of the new texture.

In RotoTexture [4], an advection tree is constructed to deal with the dynamic visibility of texture. It is mainly designed for the cluster-based synthesis method. Since our approach is mesh-guided, the tree scheme [4] is unsuitable for our framework.

We propose a graphcut-segmentation-based algorithm to address the visibility shifting issue of video retexturing, which can keep the consistencies of object movements. Graphcut algorithm has been proven to be very effective when handling partial occlusions in multiview stereo or volumetric rendering. In our scenario, graphcut [40] is adopted to precisely locate the new boundary of concerned region in the current frame.

We first build a trimap near the boundary of concerned region (Fig. 6). The interior region of the trimap belongs to the concerned region T_F. The exterior region is grouped into the background T_B. The rest region is called unknown region T_U, which includes the boundary triangles together with their mirrored ones. Graphcut is applied to the trimap for extracting the texture part in the unknown region. To enhance efficiency, color distributions of the foreground and background used in graphcut are learned once via the key frame. They are then used when processing the other frames. Since the texture part is extracted inside the trimap, mesh points are either deleted or appended according to whether there exists a triangle whose variation of texture occupancy exceeds 90 percent. Accordingly, a remeshing operation is performed. For the appearing region, the texture is replaced by fetching a new texture patch from boundary of the concerned region in texture space.

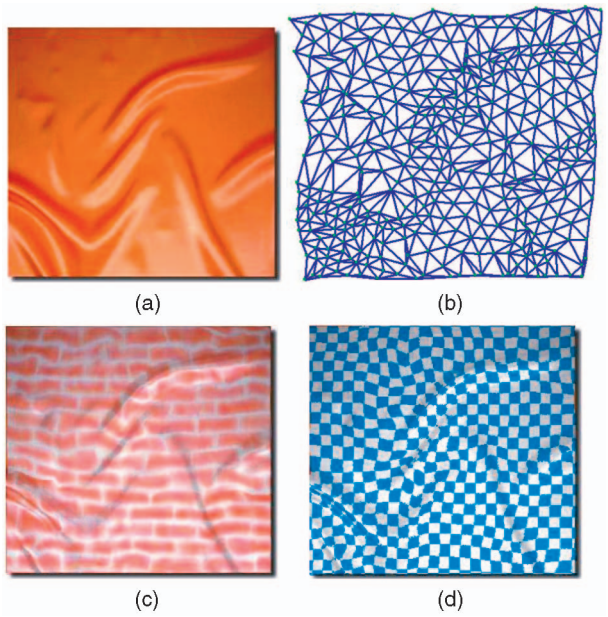


Fig. 7. Retexturing results for the image of (a). (b) is the computed mesh in texture space corresponding to the mesh in Fig. 1b. (c) and (d) are the replacement results.

5 Experimental Results

Extensively, experiments are carried out to verify our mesh-guided retexturing approach. The results are generated using an Intel Pentium IV 2.4-GHz PC with 512 Mbytes of main memory.

5.1 Results on Image

One texture replacement result for a single image is shown in Fig. 7. The input image is Fig. 7a. Fig. 7b is the computed mesh in texture space corresponding to the generated mesh in Fig. 1b. The mesh in Fig. 1b has 385 points. It takes 9 seconds to solve the nonlinear equation (8) using the multidimensional Newton's method. The brick texture and checkboard texture show regular/near regular patterns. They are used as new textures for replacing the input image. Figs. 7c and 7d are the replacement results with the brick and checkboard textures, respectively.

Fig. 8 shows another image retexturing result. Fig. 8a is a screenshot of one 3D cube rendered by *Deep Exploration* [41]. Figs. 8b, 8c, and 8d are the retexturing results with three new textures. The results show that the stretch-based parameterization can produce realistic distortion effects incurred by perspective projection. For this example, the Poisson-based refinement process is not applied because the foreshortening effect can be obtained with RBF interpolation. Figs. 9 and 10 present image retexturing results for curved surfaces using our mesh-guided approach.

Results with variant texton scales. Progressively variant textures can be produced by texture synthesis technique [42]. However, generating progressively variant textures poses challenges for previous methods of texture replacement [1], [2], [3], [8]. The main difficulty lies in how to effectively combine the algorithm of creating variant

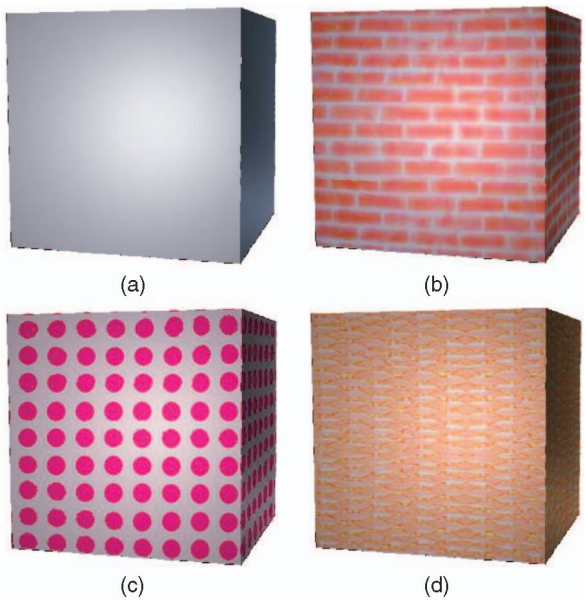


Fig. 8. (b), (c), and (d) The replacement results for a screenshot image (a) of a 3D cube. This example illustrates that our approach retains the perspective distortion convincingly.

replaced texture with the means of simulating texture distortions. With the stretch-based parameterization, our retexturing approach, however, can create such effects with ease (Figs. 11c and 11d).

In our approach, we only need to specify a few key vectors over the input image (Fig. 11a) for describing texton scales. The scale of each mesh edge computed by the inverse distance interpolation is integrated into the objective function:

$$E_l = \sum_{(i,j)\in edges} \left(w_{ij} \cdot \|Q_i - Q_j\|^2 - (1/s_{ij}^2) \cdot l_{ij}^2 \right)^2 \Big/ l_{ij}^2, \qquad (25)$$

where $1/s_{ij}$ represents the ideal scale interpolated for edge $e(Q_iQ_j)$. The smaller the edge scale is, the bigger the texton scale is. Minimizing the function yields the parameterized mesh in texture space (Fig. 11b).

Results with self-occlusion. In a self-occluded region, texture patches may be covered by their neighboring ones,

Fig. 9. Image retexturing result of clothes.

Fig. 10. Image retexturing result of curtains.

so the replaced textures may appear discontinuous in the region. It is mainly caused by the discontinuities of visual appearance of the underlying 3D surface. Unlike previous methods that cannot create such effects easily, our mesh-guided retexturing approach can generate the effects with convenient user interaction.

In our approach, visual discontinuities are handled in the process of stretch-based parameterization. We interactively specify relevant points around the boundary of occluded region (for example, the red points in Fig. 12a). Each of these points is accompanied with a virtual point. The original point and its virtual reflection represent the adjacent invisible/visible triangles, respectively. In the process of mesh parameterization, a virtual distance is valued to the original specified point and its virtual one. Fig. 12b is the generated mesh in texture space. The holes in the mesh generate discontinuities of the replaced result in self-occlusion region. Figs. 12c and 12d show two replacement results.

Fig. 13 compares our method with Textureshop [1]. The input image is the inset in Fig. 13b. Fig. 13a is our retexturing result, and Fig. 13b is directly copied from that generated by Textureshop [1, Fig. 7]. Texture discontinuities are created in drapes of the sculpture in both results. The result generated by our mesh-guided approach is visually comparable to that

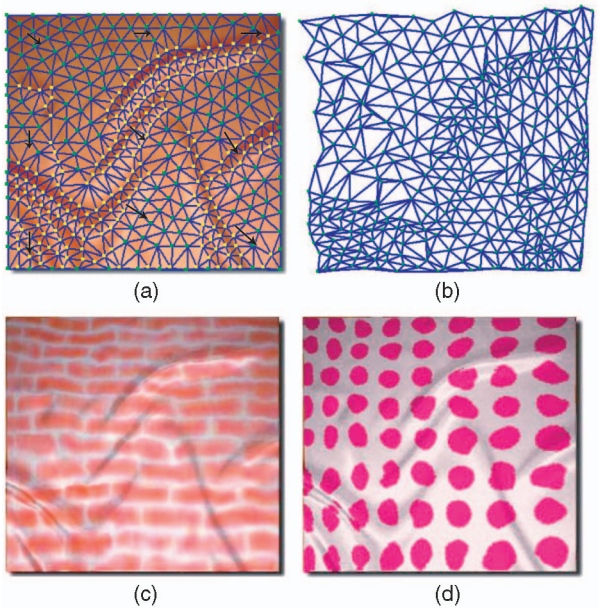


Fig. 11. Results with variant texton scales. The key vectors specified are shown in (a). (b) shows the computed mesh in texture space. From top left to bottom right, texton scales in (c) and (d) vary progressively from small to big along the horizontal and vertical directions.

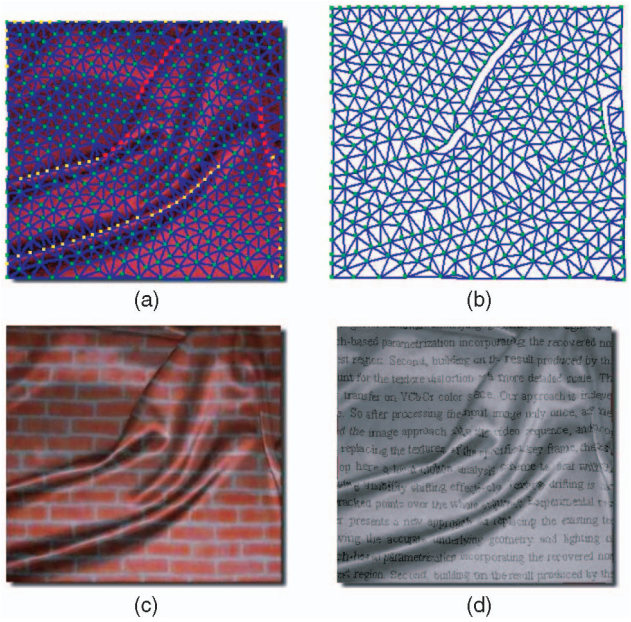


Fig. 12. Results with self-occlusion. (a) The relevant points specified over the mesh. (b) The computed mesh in texture space. (c) and (d) show the retexturing results. Texture discontinuities in the occluded regions are obvious.

of cluster-based texture synthesis method [1]. Both methods can generate good texture discontinuities in the drape regions. Figs. 13c and 13d show more replacement results generated by our approach using checkboard and text images. Fig. 14 gives another result generated by our mesh-guided approach in which the brick and checkboard textures follow the surfaces in the photo.

5.2 Results on Video

Figs. 15, 16, and 17 show some frames selected from three video retexturing results. The dynamic effects of these results can be found in the accompanying video. In the experiments, processing one video clip takes from several minutes to tens of minutes according to the length and resolution of the video clip. The computation intensive steps of our approach are tracking mesh points of moving objects, local and global smoothing of the tracked points, and texture sampling when propagating the replaced results between adjacent frames.

Fig. 15 shows some frames of one retextured video to verify the effectiveness of our visibility shifting method. The waving arm of the announcer leads to visibility changes in front of the body (please refer to the accompanying video). The video contains 50 sequential frames, among which the first frame is selected as the key frame to initialize the retexturing process. The graphcut segmentation is adopted to tackle the visibility issue. Although graphcut segmentation is incorporated into our method, the new appearing regions in some frames cannot be extracted accurately. Hence, artifacts may appear in the retextured video. One solution of reducing such artifacts is improving the accuracy of boundary segmentation of moving object. With more precise models such as Gaussian Mixture Model (GMM) [21] to compute the foreground/background distributions, the result can be improved.

Fig. 16 shows six frames from one video retexturing result. The input image sequence contains 118 sequential frames. One model walks through the stage with a silk

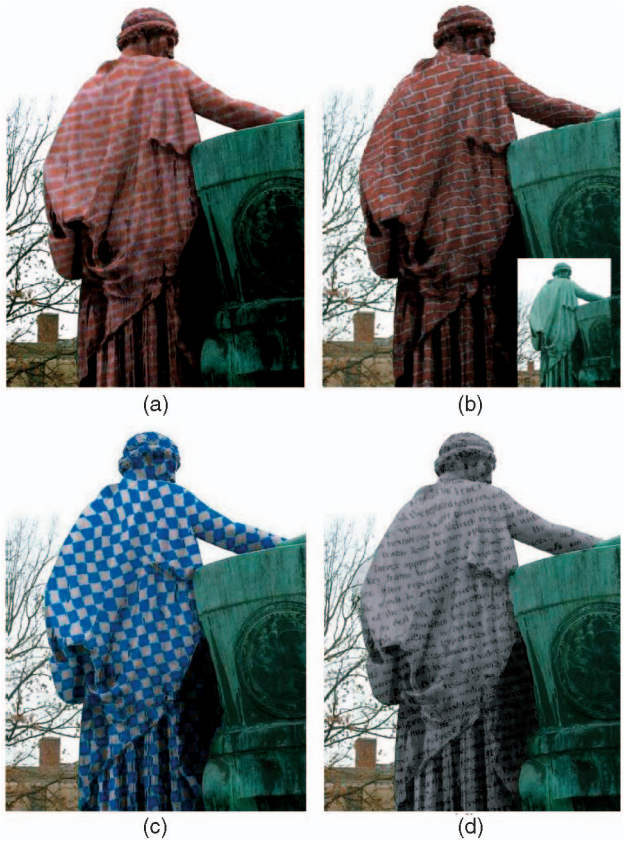


Fig. 13. Comparison with Textureshop. The input image is the inset in (b). (a) is our result and (b) is the result of Textureshop. The result generated by our mesh-guided approach is visually comparable to Textureshop's cluster-based texture synthesis method. Both methods generate good texture discontinuities in the drape regions.

dress deforming largely. In this example, eight key frames are selected in time domain. The odd rows show some input frames. The even rows show the corresponding frames of retextured video with checkboard texture. Even with the robust optical flow algorithm and local optimization, perfect tracking is impossible. There are few mesh points jumping suddenly along their motion curves. Before the global optimization of motion trajectories, these points are dragged to the user-specified positions with little editing operations. With the above scheme, our approach can generate a visually pleasing result with slight texture

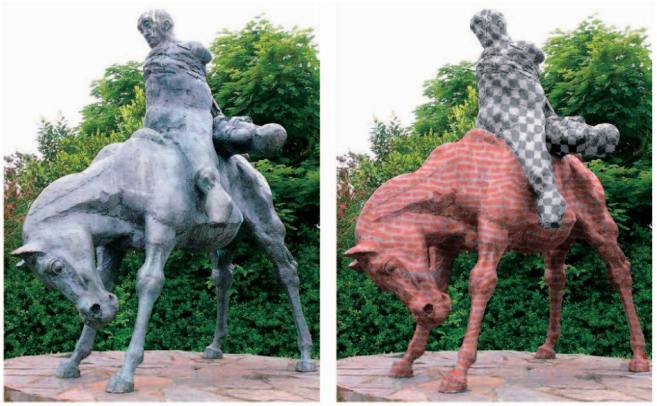


Fig. 14. Image retexturing result for a sculpture.

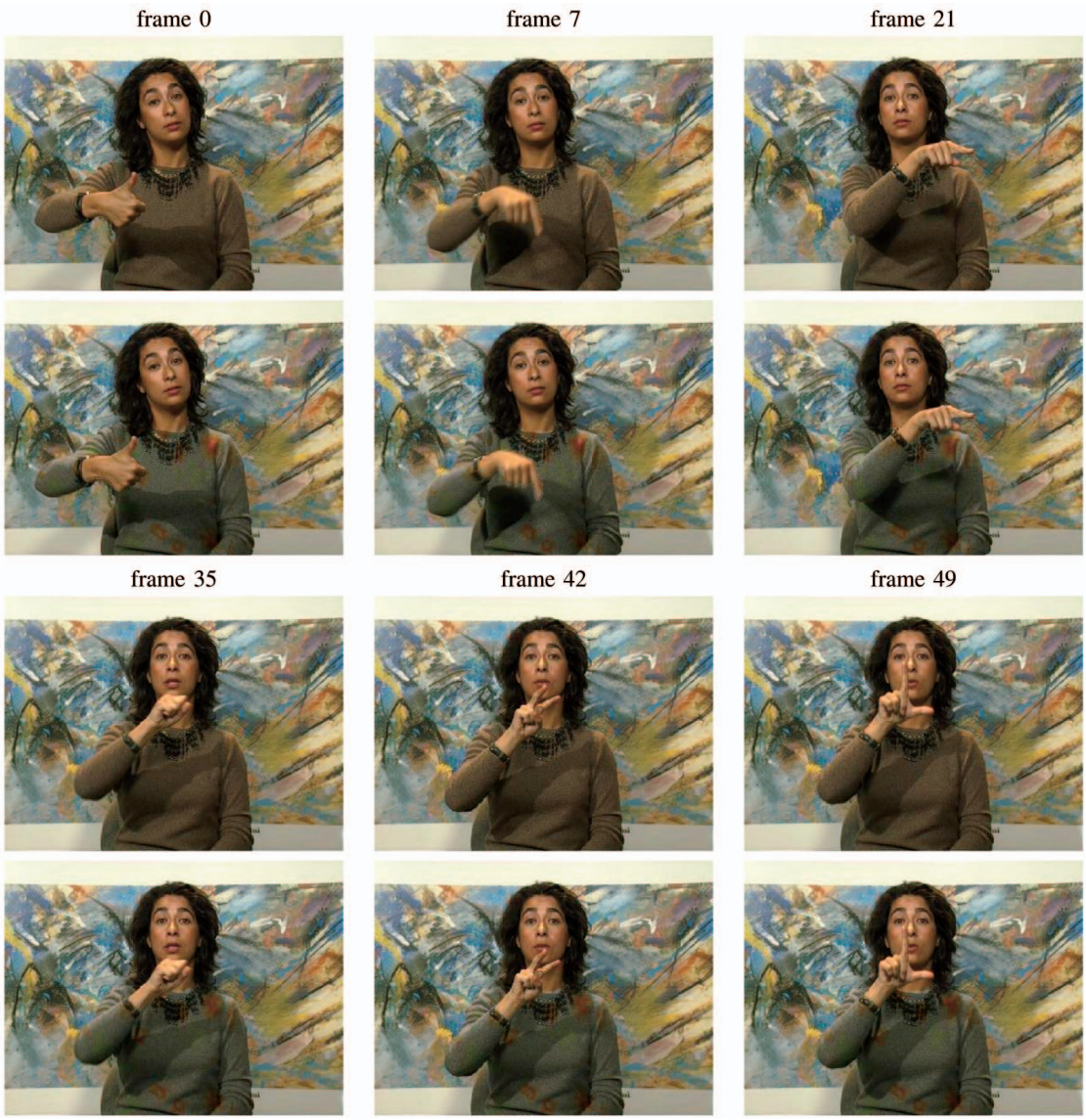


Fig. 15. Results with visibility shifting. The waving arm leads to disoccluded regions in front of the body. The first and third rows list the six frames selected. The second and fourth rows list their corresponding replacement results.

drifting. In most cases, the intersection parts of tracked meshes have the same topology. When visibility shifting occurs, the tracked meshes between the reference and current frames are slightly different. The wireframe of the tracked mesh is shown in the attached video for frame-by-frame comparison.

Fig. 17 presents one video retexturing example with metamorphing effect. The dress gradually metamorphs from the original texture to the retextured one with small body movements while the person is walking. The input video consists of 50 frames. The metamorphosis ranges from the 13th frame to the 19th frame. This special metamorphing effect of video retexturing is very useful in film production. For the whole metamorphing effect, please refer to the attached video.

5.3 Limitations

The approach presented in the paper is mesh guided, which computes texture coordinates of the generated mesh using stretch-based parameterization. This global method allows mapping one texture onto user specified region. However, it may stretch texture at the adjacent places when it compresses texture at some places. When 3D surface is severely curved and the recovered normal field is inaccurate, the stretching effects become severe. This is a limitation of our approach. However, our method can balance texture compression and texture dilation. For most tested images, texture dilations are slight, and visually pleasing results are achieved.

When retexturing an image, the self-occlusion region is tackled by incorporating user interaction into the parameterization framework. For video retexturing, this preprocess is not carried out for every frame. Actually, once the self-occlusion effect of the key frame is imitated, such effect can be propagated to the rest frames when self-occlusion does not change dramatically. If the self-occlusion region changes significantly in successive frames, a new key frame should be selected to catch this change and repropagate it. However, when the self-occlusion changes frequently, the above method will be overburdened. Generating visually

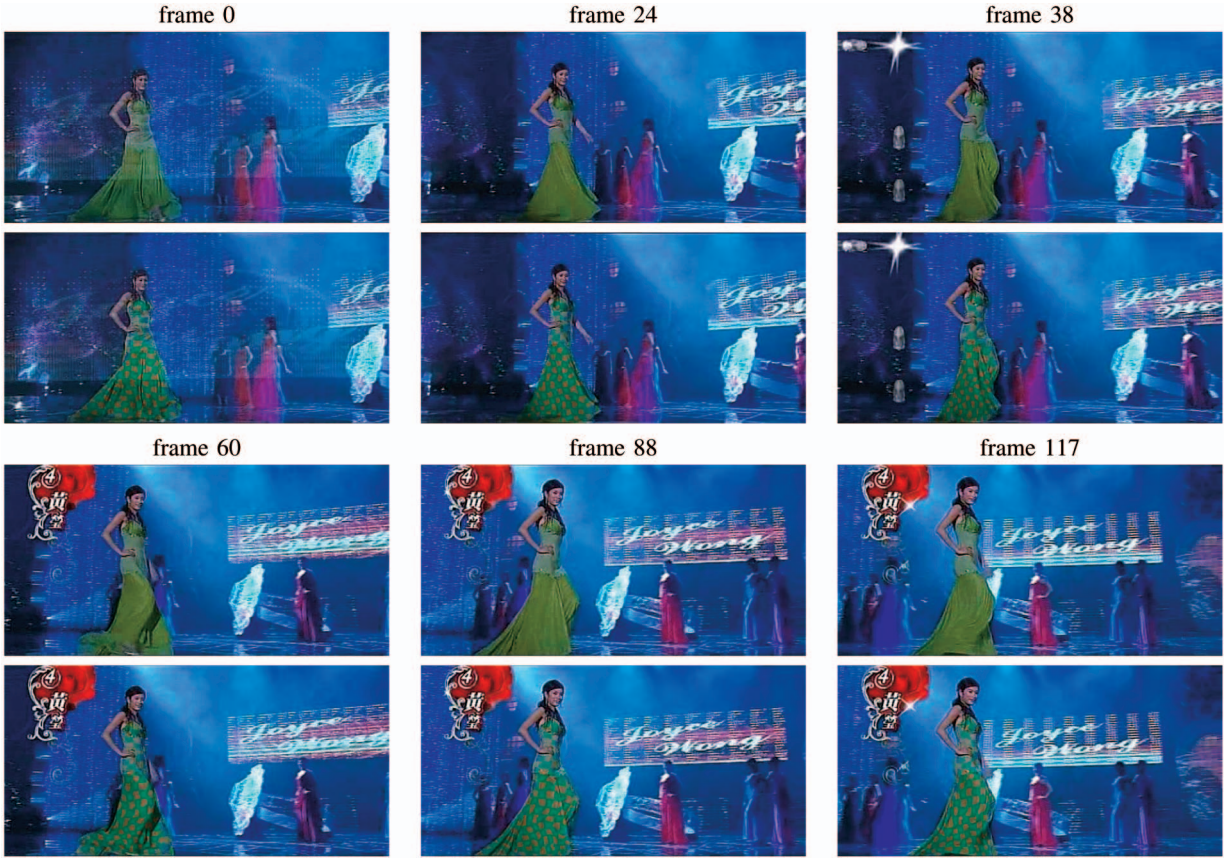


Fig. 16. Video retexturing of deforming surface. Eight frames shown in odd rows are selected from the input video. The corresponding retexturing results of the selected frames are shown in even rows. Checkboard texture is used for overwriting the dress of the model walking through the stage.

Fig. 17. Video retexturing for metamorphosis effect. Six frames are selected from the retextured video exhibiting the metamorphosis.

pleasing results for self-occlusion regions of complex dynamic video is one of the future works to be done.

6 Conclusions and Future Work

This paper presents the novel mesh-guided optimization approach for retexturing image and video. Our approach performs the task of preserving texture distortion in two steps: stretch-based mesh parameterization for perspective foreshortening and large distortion, and Poisson-based refinement for distortion at fine scale. Further, the key-frame-based video retexturing approach is devised by generalizing and extending the image retexturing method. For input video sequence, the replaced results of key frames are propagated to the rest ones. To deal with texture drifting and visibility shifting of moving objects, local and

global optimizations are designed for achieving accurate and smooth motion information of dynamic objects.

Except lighting preservation, our approach for image and video retexturing is independent of the new textures. For any input image and video, the sampling computation of texture coordinates is performed only once as a preprocessing step. The new textures can then be exerted with consistent lighting and deformation effects as the input one. This characteristic endows our algorithm with high efficiency, which makes it have wide applications including online and offline virtual exhibition, industry design, and image editing.

Our approach can effectively produce a progressively variant version of the new texture with key vectors. In general, it is difficult to create global or sudden orientation variation. Tackling this issue and allowing for more user control are the concerns of our future work. The method of synthesizing progressively variant textures [42] may be incorporated into our framework. In addition, the lighting preservation method is unsuitable for input images with strong textures. For these images, the luminance components are dominated by contributions from textures instead of from lighting effects. We will investigate more general solutions to the problem. Our current work on video retexturing is still preliminary. More investigations [43] are needed to realize more robust and efficient approach for handling video sequences with complex motions.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their constructive comments, which helped improve this paper. The image in Fig. 13b is the copy in [1, Fig. 7]. The video used in Fig. 17 is obtained from that in [16]. This work is supported by the National Natural Science Foundation of China (Grants 60703084 and 60723003), the Natural Science Foundation of JiangSu Province of China (Grant BK2007571), CUHK direct grant for research (2050349), RGC Research Grant (416007), the National Basic Research Project of China (Grant 2002CB312101), and the 863 Program of China (Grant 2006AA01Z325).

REFERENCES

[1] H. Fang and J. Hart, "Textureshop: Texture Synthesis as a Photograph Editing Tool," *Proc. ACM SIGGRAPH '04/ACM Trans. Graphics,* vol. 23, no. 3, pp. 354-359, 2004.

[2] Y. Liu, W.-C. Lin, and J. Hays, "Near Regular Texture Analysis and Manipulation," *Proc. ACM SIGGRAPH '04/ACM Trans. Graphics,* vol. 23, no. 3, pp. 368-376, 2004.

[3] Y. Tsin, Y. Liu, and V. Ramesh, "Texture Replacement in Real Images," *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '01),* vol. 2, pp. 539-544, 2001.

[4] H. Fang and J. Hart, "Rototexture: Automated Tools for Texturing Raw Video," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 6, pp. 1580-1589, Nov./Dec. 2006.

[5] V. Scholz and M. Magnor, "Texture Replacement of Garments in Monocular Video Sequences," *Proc. 17th Eurographics Symp. Rendering,* pp. 305-312, 2006.

[6] B. Oh, M. Chen, J. Dorsey, and F. Durand, "Image-Based Modeling and Photo Editing," *Proc. ACM SIGGRAPH '01,* pp. 433-442, 2001.

[7] B. K. Horn, "Height and Gradient from Shading," *Int'l J. Computer Vision,* vol. 5, no. 1, pp. 37-75, 1990.

[8] S. Zelinka, H. Fang, M. Garland, and J. Hart, "Interactive Material Replacement in Photographs," *Proc. Graphics Interface,* pp. 227-232, 2005.

[9] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy Snapping," *Proc. ACM SIGGRAPH '04/ACM Trans. Graphics,* vol. 23, no. 3, pp. 303-308, 2004.

[10] S. Zelinka and M. Garland, "Towards Real-Time Texture Synthesis with the Jump Map," *Proc. 13th Eurographics Workshop Rendering Techniques,* pp. 99-104, 2002.

[11] E.A. Khan, E. Reinhard, R.W. Fleming, and H.H. Bülthoff, "Image-Based Material Editing," *Proc. ACM SIGGRAPH '06/ACM Trans. Graphics,* vol. 25, no. 3, pp. 654-663, 2006.

[12] R. White and D. Forsyth, "Retexturing Single Views Using Texture and Shading," *Proc. European Conf. Computer Vision (ECCV '06),* vol. 3, pp. 225-239, 2006.

[13] S.S. Beauchemin and J.L. Barron, "The Computation of Optical Flow," *ACM Computing Surveys,* vol. 27, no. 3, pp. 433-467, 1995.

[14] J. Shi and C. Tomasi, "Good Features to Track," *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '94),* pp. 593-600, 1994.

[15] H. Jin, P. Favaro, and S. Soatto, "Real-Time Feature Tracking and Outlier Rejection with Changes in Illumination," *Proc. IEEE Int'l Conf. Computer Vision (ICCV '01),* vol. 1, pp. 684-689, 2001.

[16] A. Agarwala, A. Hertzmann, D.H. Salesin, and S.M. Seitz, "Keyframe-Based Tracking for Rotoscoping and Animation," *Proc. ACM SIGGRAPH '04/ACM Trans. Graphics,* vol. 23, no. 3, pp. 584-591, 2004.

[17] J. Wang, Y. Xu, H.-Y. Shum, and M. Cohen, "Video Tooning," *Proc. ACM SIGGRAPH '04/ACM Trans. Graphics,* vol. 23, no. 3, pp. 574-583, 2004.

[18] Y.-Y. Chuang, A. Agarwala, B. Curless, D. Salesin, and R. Szeliski, "Video Matting of Complex Scenes," *Proc. ACM SIGGRAPH '02/ ACM Trans. Graphics,* vol. 21, no. 3, pp. 243-248, 2002.

[19] M.J. Black, D.J. Fleet, and Y. Yacoob, "Robustly Estimating Changes in Image Appearance," *Computer Vision and Image Understanding,* vol. 78, no. 1, pp. 8-31, 2000.

[20] Y. Altunbasak and A.M. Tekalp, "Closed-Form Connectivity-Preserving Solutions for Motion Compensation Using 2-D Meshes," *IEEE Trans. Image Processing,* vol. 6, no. 9, pp. 1255-1269, 1997.

[21] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut Interactive Foreground Extraction Using Iterated Graph Cuts," *Proc. ACM SIGGRAPH '04/ACM Trans. Graphics,* vol. 23, no. 3, pp. 309-314, 2004.

[22] J. Maillot, H. Yahia, and A. Verroust, "Interactive Texture Mapping," *Proc. ACM SIGGRAPH '93,* pp. 27-34, 1993.

[23] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Mutiresolution Analysis of Arbitrary Meshes," *Proc. ACM SIGGRAPH '95,* pp. 173-182, 1995.

[24] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle, "Conformal Surface Parametrization for Texture Mapping," *IEEE Trans. Visualization and Computer Graphics,* vol. 6, no. 2, pp. 181-189, Apr.-June 2000.

[25] P.V. Sander, J. Snyder, S.J. Gortler, and H. Hoppe, "Texture Mapping Progressive Meshes," *Proc. ACM SIGGRAPH '01,* pp. 409-416, 2001.

[26] B. Levy, S. Petiejean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *Proc. ACM SIGGRAPH '02/ACM Trans. Graphics,* vol. 21, no. 3, pp. 362-371, 2002.

[27] A. Sheffer and E. de Sturler, "Smoothing an Overlay Grid to Minimize Linear Distortion in Texture Mapping," *ACM Trans. Graphics,* vol. 21, no. 4, pp. 874-890, 2002.

[28] C. Gotsman, X. Gu, and A. Sheffer, "Fundamentals of Spherical Parameterization for 3D Meshes," *Proc. ACM SIGGRAPH '03/ACM Trans. Graphics,* vol. 22, no. 3, pp. 358-363, 2003.

[29] K. Zhou, J. Snyder, B. Guo, and H.-Y. Shum, "Iso-Charts: Stretch-Driven Mesh Parameterization Using Spectral Analysis," *Proc. ACM Symp. Geometry Processing,* pp. 45-54, 2004.

[30] http://mathworld.wolfram.com/, 2007.

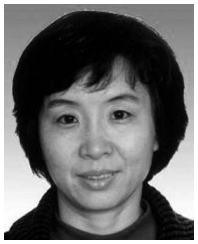
[31] J. Tohline, *Origin of the Poisson Equation,* http://www.phys.lsu.edu/astro/HBook.current/Context/PGE/poisson.origin.text.pdf, 1999.

[32] P. Perez, M. Gangnet, and A. Blake, "Poisson Image Editing," *Proc. ACM SIGGRAPH '03,* vol. 22, no. 3, pp. 313-318, 2003.

[33] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, "Interactive Digital Photomontage," *Proc. ACM SIGGRAPH '04/ACM Trans. Graphics,* vol. 23, no. 3, pp. 294-302, 2004.

[34] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh Editing with Poisson-Based Gradient Field Manipulation," *Proc. ACM SIGGRAPH '04/ACM Trans. Graphics,* vol. 23, no. 3, pp. 641-648, 2004.
[35] D. Xu, H. Zhang, Q. Wang, and H. Bao, "Poisson Shape Interpolation," *Proc. ACM Symp. Solid and Physical Modeling,* pp. 267-274, 2005.
[36] Z. Jiang, T.-T. Wong, and H. Bao, "Practical Super-Resolution from Dynamic Video Sequences," *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '03),* vol. 2, pp. 549-554, 2003.
[37] D. Shepard, "A Two Dimensional Interpolation Function for Irregularly Spaced Data," *Proc. 23rd Nat'l Conf. ACM,* pp. 517-524, 1968.
[38] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '98),* pp. 839-846, 1998.
[39] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral Mesh Denoising," *Proc. ACM SIGGRAPH '03/ACM Trans. Graphics,* vol. 22, no. 3, pp. 950-953, 2003.
[40] Y. Boykov and M.P. Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in n-d Images," *Proc. IEEE Int'l Conf. Computer Vision (ICCV '01),* vol. 1, pp. 105-112, 2001.
[41] *Right Hemisphere,* http://www.righthemisphere.com/products/dexp/, 2007.
[42] J. Zhang, K. Zhou, L. Velho, B. Guo, and H.-Y. Shum, "Synthesis of Progressively-Variant Texture on Arbitrary Surfaces," *Proc. ACM SIGGRAPH '03/ACM Trans. Graphics,* vol. 22, no. 3, pp. 295-302, 2003.
[43] Y. Li, J. Sun, and H.-Y. Shum, "Video Object Cut and Paste," *Proc. ACM SIGGRAPH '05/ACM Trans. Graphics,* vol. 24, pp. 595-600, 2005.

Yanwen Guo received the PhD degree in applied mathematics from State Key Lab of CAD & CG, Zhejiang University in 2006. He is currently an assistant professor at the National Laboratory of Novel Software Technology, Nanjing University. He worked as a visiting researcher in the Department of Computer Science and Engineering, Chinese University of Hong Kong in 2006. His main research interests include image and video processing, geometry processing, and face-related applications.



Hanqiu Sun received the MS degree in electrical engineering from the University of British Columbia and the PhD degree in computer science from the University of Alberta, Canada. She is currently an associate professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong. Her research interests include virtual and augmented reality, interactive graphics/animation, hypermedia, Internet-based visualization and navigation, computer-assisted surgery, and touch-enabled simulations. She has published more than 100 technical papers refereed at prestigious VR/interactive Graphics journals and international conference proceedings. She has served as a guest editor of *MIT PRESENCE* and the *Journal of CAVW*, conference general chair of ACM VRCIA 2006, program cochair of ACM VRST 2002, organization cochair of PG 2005, CGI 2006, and director of the advanced workshop on VR 2007. She is a member of the IEEE.

Qunsheng Peng received the PhD degree in the School of Computing Studies, University of East Anglia, UK, in 1983. He is currently a professor in the State Key Laboratory of CAD &CG, Zhejiang University and serves as the vice chairman of the lab academic committee. His research interests include realistic image synthesis, virtual reality, biomolecule graphics, and scientific visualization. In these fields, he is the author or coauthor of more than 100 journal and conference proceedings, including ACM SIGGRAPH, ACM VRST, ICCV, Eurographics, Pacific Graphics, *The Visual Computer*, etc. He received the best paper award of Eurographics '89 and received the Computer Graphics Achievements Award of China at Chinagraph 2000. He is a member of the editorial board of *The Visual Computer* and the *Journal of Computer Science and Technology*.

Zhongding Jiang received the BS and PhD degrees from the State Key Lab of CAD & CG, Zhejiang University in 1998 and 2004, respectively. He is currently an assistant professor at the Software School, Fudan University, and serves as the director of the Computer Graphics Laboratory, where he does research on computer graphics, multimedia, human-computer interaction, and Web science. He interned at Microsoft Research Asia in 2000. He worked in the Department of Computer Science and Engineering, the Chinese University of Hong Kong, in 2002 as visiting researcher. His current research interests include projector-camera system, image-based modeling and rendering, mobile and gaming graphics, and Internet advertising. He is a member of the ACM and the IEEE.