# Inference of Segmented Color and Texture Description by Tensor Voting

Jiaya Jia, *Student Member*, *IEEE*, and Chi-Keung Tang, *Member*, *IEEE Computer Society*

**Abstract**—A robust synthesis method is proposed to automatically infer missing color and texture information from a damaged 2D image by $N$D tensor voting ($N > 3$). The same approach is generalized to range and 3D data in the presence of occlusion, missing data and noise. Our method translates texture information into an adaptive $N$D tensor, followed by a voting process that infers noniteratively the optimal color values in the $N$D texture space. A two-step method is proposed. First, we perform segmentation based on insufficient geometry, color, and texture information in the input, and extrapolate partitioning boundaries by either 2D or 3D tensor voting to generate a complete segmentation for the input. Missing colors are synthesized using $N$D tensor voting in each segment. Different feature scales in the input are automatically adapted by our tensor scale analysis. Results on a variety of difficult inputs demonstrate the effectiveness of our tensor voting approach.

**Index Terms**—Image restoration, segmentation, color, texture, tensor voting, applications.

## 1 Introduction

The human visual system enables us to make robust inference from insufficient data. We can perform an amazing job of extrapolating shape and texture information despite severe occlusion and noise. In this paper, we present a computational framework to fill in missing geometry, color, and texture information in large image holes, present encouraging results in 2D and 3D, and propose useful applications:

- 2D data—from only a single image with large holes, our method synthesizes missing color and texture information seamlessly.
- Range or 3D data—from a single depth map, stereo data, or noisy 3D data, we recover missing data and correct erroneous geometry, and repair colors and textures on surfaces.

No a priori complex scene or texture model is assumed in our *tensor voting* approach, which adopts an adaptive continuity constraint in 2D, 3D, and $N$D. Because of this, our method has certain limitations which are absent from our powerful visual system where prior knowledge are employed. They will be discussed in the conclusion section. In this paper, we first approach the more difficult problem in 2D and then generalize our synthesis method to range and 3D data, when some insufficient depth or geometry information exists. By adopting recent texture segmentation algorithms, we describe how tensor voting provides a robust and effective methodology to perform image segmentation and to synthesize missing geometry as well as color and texture information.

## 2 Motivation

Let us first motivate our work by studying the 2D *image repairing problem* [21]. Today, powerful photo-editing softwares and large varieties of retouching, painting, and drawing tools are available to assist users to refine or redefine images manually. For instance, we can easily outline and remove a large foreground object by intelligent scissors [30] or JetStream [32], leaving behind a large background hole. However, filling the background hole seamlessly from existing neighborhood information of damaged images is still a difficult problem. A skilled art designer repairs them mostly through his experience and knowledge. Alternatively, some stereo algorithms [22], [37] use a dense set of images to infer the color of occluded pixels.

Given as few as one image and no additional knowledge, is it possible to automatically repair it? The main issues to be addressed are as follows:

- How much color and shape information in the existing part is needed to seamlessly fill the hole?
- How good can we achieve in order to reduce possible visual artifacts when the information available is not sufficient?

Inpainting [8], [2] is an automatic method to fill small holes by exploiting color and gradient information in the neighborhood of the holes. It successfully restores images with little visual artifact. However, when the method does not take texture information into account, blurring in a textured region will be introduced.

In this paper, we propose a unified approach to gather and analyze statistical information so as to synthesize proper pixel values in image holes. This is achieved by computing a tensor to encode the underlying point and texture distribution (smooth or discontinuous), in the geometry and texture space respectively. Our method generalizes well to range and 3D data under certain conditions to be discussed. Instead of simply extending neighboring color information, we use the robust, noniterative $N$D tensor voting [29] to globally infer the most suitable pixel value in the neighborhood under the Markov Random Field (MRF) constraint. A texture-based segmentation [13] is adopted to partition images into different region segments. Depending on the data dimensionality, 2D or 3D curve connections are voted for, in order to complete the segmentation in image holes. Missing colors are synthesized

• *The authors are with the Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: {leojia, cktang}@cs.ust.hk.*

*Manuscript received 19 Apr. 2003; revised 5 Nov. 2003; accepted 13 Nov. 2003. Recommended for acceptance by A. Khotanzad.*
*For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 0043-0403.*

by $N$D tensor voting, which adapts to different feature and texture scales.

The rest of this paper is organized as follows: We discuss and compare related work in Section 3. In Section 4, we review the tensor voting algorithm. In Section 5, we describe our 2D method in detail, where the adaptive $N$D tensor voting for color and texture synthesis is introduced. In Section 6, 3D/range data repairing on geometry, color, and texture is described, which is a generalization of 2D repairing. Detailed time complexity analysis is given in Section 7. Our 2D and 3D results are presented and explained in Section 8. Finally, we discuss the limitation of our approach, propose future work and conclude our paper in Section 9.

The conference version of this paper first appeared in [21], which mainly focuses on 2D image repairing. In this coverage, we provide full details, expand our approach to range and 3D data, and present many new results.

## 3 Previous Work

One contribution of our work is the robust synthesis of missing image or texture information in large holes. To extrapolate details, the Markov Random Fields (MRF) have been widely used to describe textures [5], [15], [43], including this work. Texture synthesis techniques can be roughly categorized into three classes. The first class uses a parametric model to describe statistical textures. Portilla and Simoncelli [35] used joint wavelet coefficients to parameterize and synthesize textures. Heeger and Bergen [18] made use of Laplacians, steerable pyramids, and histogram analysis of a texture sample to synthesize textures. While impressive results are obtained for highly stochastic textures, these methods are not well-suited to represent highly structured textures such as that of a brick wall.

The second class consists of nonparametric sampling and synthesis techniques. Efros and Leung [15] synthesized textures by copying pixels that are matched with the neighborhood of a pixel being synthesized. It works very well to reproduce structured textures as well as stochastic textures.

The third class is to synthesize textures by procedural means. Solid textures, proposed independently by Perlin [33] and Peachey [31], involve a function that returns a color value at any given 3D point. Reaction-diffusion techniques [44] build up spot and stripe patterns which can be used to synthesize textures.

To accelerate the synthesis process and avoid the problems inherent in pixel-wise generation, Xu et al. [45] proposed a block-wise synthesis algorithm to seamlessly copy sample patches in an overlapping manner to create a new image. Image quilting [16] extends this algorithm by a minimum error boundary cut so that synthesized patches naturally connect to each other. Wei et al. [43] described another hierarchical texture synthesis algorithm to speed up the per-pixel generation process by matching the neighborhood of lower resolution image pixels with the synthesized ones, and applying tree-structured vector quantization to accelerate the process.

Masnou and Morel [28] presented a variational formulation to fill in, or *inpaint* regions to be disoccluded by joining points of the isophotes. Inpainting [8], [2] is an algorithm to restore damaged 2D images. Instead of performing matching and copying, the basic idea of image inpainting is to smoothly propagate information from the surrounding areas in the isophotes direction. This algorithm generates satisfactory results when holes are small and the gradient information in holes does not change abruptly. In more recent development, textures as well as structures are taken into consideration [3]. However, in a situation where large holes and complex scene components (e.g., single image input as shown in Fig. 19) exist, we argue that segmentation information (such as discontinuity curves to partition distinct geometry and textures) should be considered. Moreover, in the texture synthesis process in [3], it requires either manual parameter adjustment for different texture patterns, or simple constant texture/color assumption. Although there are some preset parameters in our method, our adaptive high dimensional tensor can automatically measure different feature scales in the input image. Our goal is similar to image completion [14], which also recovers a large portion of disoccluded missing background. One major difference is that explicit segmentation is considered and robustly inferred in this work, in order not to blur important edges. In [46], images are restored by combining texture synthesis and image inpainting in multiresolution. Critical edge regions are still blurred. Criminisi et al. [10] combined texture synthesis and image inpainting. Levin et al. [24] inpaint images by taking global image statistics into consideration. A training image is used.

The secondary contribution is the robust inference of geometric features, such as curves inside image holes and missing data in range and 3D images, to assist texture and color synthesis. In [29], a survey and comparison of tensor voting with representative computer vision approaches in surface inference from a 3D cloud of points, e.g., deformable models [23] and computational geometry [19], are given.

For range and 3D data, unorganized sampled data are common input. To infer missing geometry or depth information, many algorithms have been proposed to regularize and reorganize the points [38], [20], [6], [17]. Turk and O'Brien [40] used variational implicit surfaces. Hoppe et al. [20] adopted a combinational energy function which takes connectivity and distance factors into consideration to refine the original meshes. Curless and Levoy [11] set some additional unseen and empty states, and fill holes by a space carving step. Therefore, after isosurface extraction, aliasing artifacts in the holes are observed. In order to smooth these areas, postfiltering is performed to blur the hole-filled surface for generating a smooth connection. In their examples, holes are relatively small so that they can be filled by simple smooth surface elements. In the postrendering 3D warping algorithm [27], to fill an uncovered portion in image without obvious artifact, splat and mesh reconstructions are implemented. For each gap area with high-connectedness, a set of quadrilateral meshes are stretched to cover the hole, and the vertex colors of the reconstructed triangles are linearly interpolated. For low-connectedness, a heuristic is used to flat-shade the triangle using vertex color, if it is far away from the reference frame viewpoint. Chang et al. [9] extended the layered depth image (LDI) tree algorithm to fill the gaps by two-pass pixel filtering. Instead of meshing the foreground and background, their algorithm fills the gap by splatting the filtered samples from surrounding surface patches in the output. Therefore, a better result can be obtained. However this splatting may not respect surface orientation discontinuities (e.g., corners).

In [1], a variational approach for filling in missing data in 2D and 3D digital images is proposed, which is based on a joint interpolation of the image gre-levels and gradient/isophotes direction. In [41], inpainting surface holes is achieved by extending the image inpainting algorithm. Hole filling algorithm in [42] deals with locally smooth structures, while our hole filling can deal with discontinuities. A locally

smooth surface is also a requirement in [7], which uses radial basis function for interpolating missing information from depth maps resulting from medical data. Pfeifle and Seidel [34] used B-splines for hole filling.

## 4 Review of Tensor Voting

In our method, *Tensor Voting* [29] is used to infer missing curves and pixel values. It makes use of a *tensor* for token representation, and *voting* for communication among tokens. Tensor and voting are brought together by a voting field, which is a dense tensor field for postulating smooth connections among tokens. In this section, we first give a concise review of the stick tensor and the stick voting field, which are used in the following sections. The key idea is the propagation of an adaptive continuity or smoothness constraint in a finite neighborhood.

### 4.1 Token Representation and Communication

We are interested in answering the following geometric question in 2D, which can be generalized to higher dimension readily. Suppose there exists a smooth curve connecting the origin $O$ and a point $P$. Suppose also that the normal $\vec{N}$ to the curve at $O$ is known. What is the most likely normal direction at $P$? Fig. 1a illustrates the situation. We claim that the osculating circle connecting $O$ and $P$ is the most likely connection since it keeps the curvature constant along the hypothesized circular arc. The most likely normal is given by the normal to the circular arc at $P$ (thick arrow in Fig. 1a). This normal at $P$ is oriented such that its inner product with $\vec{N}$ is nonnegative. The length of this normal, which represents the vote strength, is inversely proportional to the arc length $OP$ and also to the curvature of the underlying circular arc. The decay of the field takes the following form:

$$\overline{DF}(r, \varphi, \sigma) = e^{-\left(\frac{r^2+c\varphi^2}{\sigma^2}\right)}, \tag{1}$$

where $r$ is the arc length $OP$, $\varphi$ is the curvature, and $c$ is a constant which controls the decay with high curvature. $\sigma$ controls smoothness, which also determines the effective neighborhood size. If we consider all possible locations of $P$ with nonnegative $x$ coordinates in the 2D space, the resulting set of normal directions thus produced constitutes the 2D stick voting field, Fig. 1b.

Given an input token $A$, how can it cast a stick vote to another token $B$ for inferring a smooth connection, assuming that $A$'s normal is known? It is illustrated in Fig. 1b. First, we fix $\sigma$ to determine the size of the voting field. Then, we align the voting field with $A$'s normal, simply by translation and rotation. If $B$ is within $A$'s voting field neighborhood, $B$ receives a stick vote $[v_x\ v_y]^T$ from the aligned field. Hence, voting is similar to convolution, and the voting field is like a convolution mask, except that the voting result is not a scalar.

Other input tokens cast votes to $B$ as well. Second order tensor sums of all votes received at $B$ are collected into a covariance matrix

$$\mathbf{S} = \begin{bmatrix} \sum v_x^2 & \sum v_x v_y \\ \sum v_y v_x & \sum v_y^2 \end{bmatrix}.$$

The corresponding eigensystem consists of two eigenvalues $\lambda_1 \geq \lambda_2 \geq 0$ and two corresponding eigenvectors $\hat{e}_1$ and $\hat{e}_2$. Therefore, $\mathbf{S}$ can be rewritten as $\mathbf{S} = (\lambda_1 - \lambda_2)\hat{e}_1\hat{e}_1^T + \lambda_2(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T)$. $\hat{e}_1\hat{e}_1^T$ is a 2D stick tensor with $\hat{e}_1$ indicating curve normal direction. $\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T$ is a 2D ball tensor.

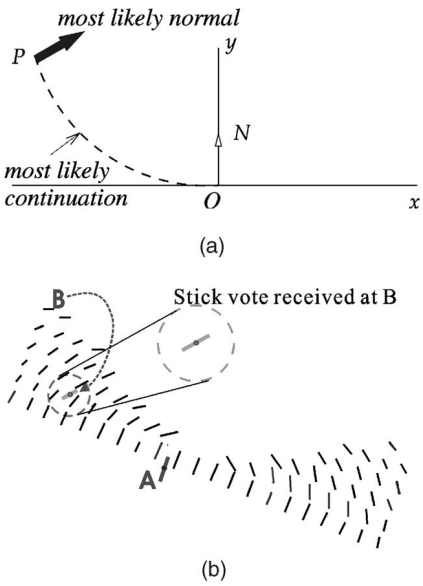


Fig. 1. (a) Design of the 2D stick voting field. (b) $A$ casts a stick vote to $B$, using the 2D stick voting field. An adaptive smoothness constraint is propagated by voting, where we do not smooth everywhere.

If $A$'s normal is unknown, we need to estimate it first. Normal estimation is also achieved by tensor voting: Each point votes with the *2D ball voting field*, which is obtained by integrating or summing up the vote contributions from a rotating 2D stick voting field.

In 3D,

$$\mathbf{S} = (\lambda_1 - \lambda_2)\hat{e}_1\hat{e}_1^T + (\lambda_2 - \lambda_3)(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T) + \lambda_3(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T + \hat{e}_3\hat{e}_3^T),$$

where $\hat{e}_1\hat{e}_1^T$ is a 3D stick tensor, $\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T$ is a 3D plate tensor, and $\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T + \hat{e}_3\hat{e}_3^T$ is a 3D ball tensor.

### 4.2 Feature Extraction

When we have obtained normal directions at each input site, we can infer a smooth structure that connects the tokens with high feature saliencies (in 2D (respectively, 3D), curve (respectively, surface) saliency is represented by $\lambda_1 - \lambda_2$ [29]). Feature extraction can be achieved by casting votes to all discrete sites (input and noninput sites), using the same voting fields and voting algorithm.

Given this dense information, in 2D (respectively, 3D), we extract true curve (respectively, surface) points and connect them. The curve or surface mesh (triangular mesh) extraction algorithms are modified marching cubes algorithms [29]. Alternatively, we can use B-splines to obtain a smoother representation, by treating the inferred points as control points.

## 5 2D Repairing

In this section, we describe image repairing [21] for 2D image data, which generalizes well to 3D (Section 6).

### 5.1 Algorithm Framework

Fig. 2 gives the overview of the whole process, which uses a single damaged image as input. To robustly generate pixel colors, a two-phase process is performed (highlighted in the

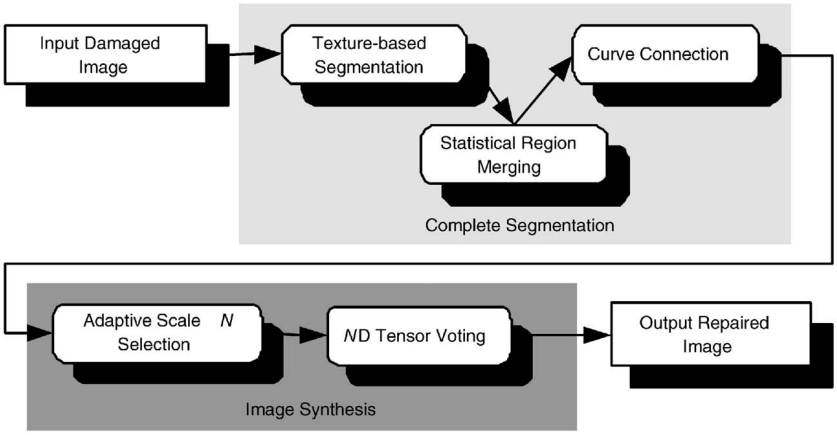


Fig. 2. Overview of our 2D image repairing algorithm.

figure). We call the two phases *complete segmentation* and *color synthesis*, respectively.

**Complete segmentation**. Usually, images reveal a cluttered scene with indistinct edge features, leading to a complex image structure. In order not to mix up different information, we perform texture-based segmentation in the damaged image, followed by extrapolating the resulting partitioning curves into an image hole. To this end, 2D tensor voting is used to infer the most likely connection between two curve endpoints. A complete segmentation for the damaged image is achieved.

In case of range or 3D data, we apply 3D tensor voting first to infer missing surface patches/depth values. Junction curves that localize depth or surface orientation discontinuities are also found by tensor voting.

**Color synthesis**. Let us call a pixel in an image hole (including the one lying on the hole boundary) a *defective pixel*. When an image hole has been segmented, we synthesize the color value of a defective pixel using only existing pixels belonging to the same group as shown in Fig. 3. In other words, we qualify the MRF constraint by complete segmentation. Adaptive scale adjustment further eliminates visual artifact.

## 5.2 Complete Segmentation

We adopt the unsupervised segmentation algorithm described in [13] to perform pattern segmentation. Initially, we assign defective pixels in an image hole to the same segment $P_0$. Then, seed growing and region merging are performed. In the process, we can assign merge threshold for small regions so as to generate partitions with finer details.

### 5.2.1 Region Merging

In real images, complex occlusion and object discontinuities are typical. For instance, the building in the flower garden is occluded and partitioned into several parts by the foreground tree in Fig. 19a. In Fig. 18, the silhouette of the tree is not a smooth blob, but contains many curve orientation discontinuities. We perform region merging to group similar objects together. The grouping is performed in color and gradient space.

To measure color and pattern feature of different regions, the statistics of zeroth and first order image intensity derivatives are considered. We construct a $(M+1)$D intensity vector $V_i^{M+1}$ for each region $P_i$ $(i \neq 0)$ where $M$ is the maximum color depth in the whole image. The first $M$ components in the intensity vector, i.e., $V_i(1)$ to $V_i(M)$, represent the histogram of region $i$. The last component is defined as

$$V_i(M+1) = \frac{\alpha}{R_i} \sum_{j \in P_i} \| \nabla j \|, \tag{2}$$

where $R_i$ is the number of points in region $P_i$ and $\| \nabla j \|$ is the intensity gradient magnitude of point $j$. $\alpha$ can be thought as a controlling weight to adjust the significance of the gradient information for measuring intensity similarity. The larger $\alpha$ is, the more important intensity gradient becomes. In our experiments, $\alpha$ is set to normalize $V_i(M+1)$ so that the largest gradient magnitude equals to the color depth, and the smallest one equals to zero.

Therefore, we merge two similar regions $P_i$ and $P_k$ into $P$, or $P = P_i \cup P_k$, if

$$s_{i,k} = \| V_i - V_k \| \leq Threshold, \tag{3}$$

where $s_{i,k}$ is the similarity score for the region pair and $V_i$ and $V_k$ are intensity vectors for $P_i$ and $P_k$, respectively. Hence, we match not only the color information but also some nearest neighborhood statistics, i.e., pattern information, to obtain a more reliable measurement.

Next, we work with the region boundary to infer partitioning curves inside an image hole.

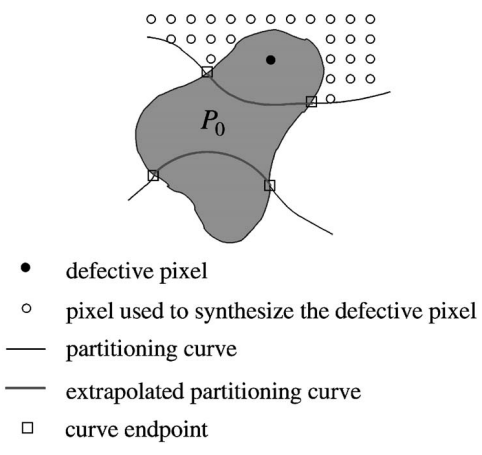


Fig. 3. Color synthesis constrained by complete segmentation.

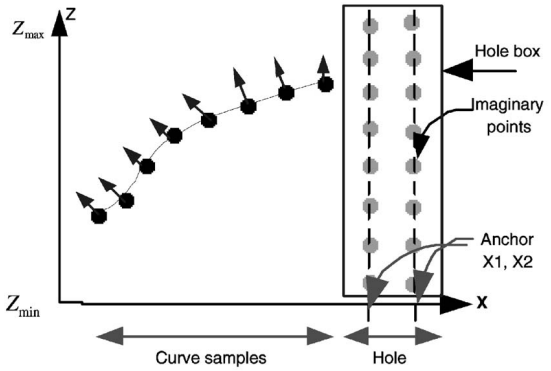


Fig. 4. Hole filling by hypothesizing depth values.

### 5.2.2 Curve Connection by 2D Tensor Voting

After merging regions in the damaged image, we want to extrapolate the partitioning curves into the holes to complete our segmentation by inferring the most likely connection between curve endpoints. Normally, we do not have any additional knowledge of the missing data. The only available constraint for extrapolating boundary curves is to maintain existing geometrical features, e.g., curvature and curve shape. In order not to oversmooth the synthesized curves and to preserve the shape of the segmented region, the robust tensor voting algorithm is used. Tensor voting votes for the missing curve elements in image holes by gathering tensorial support in the neighborhood.

Fig. 4 shows a simplified hole filling example. First, we place *imaginary points* (lightly shaded dots in Fig. 4) evenly distributed in the hole (the rectangle in Fig. 4). Denote the sampling density by $S = Z_{max} - Z_{min} + 1$. Our goal is to identify true curve points from the set of imaginary points in the hole. First, we infer normal directions on the existing curve points by tensor voting [29] (black arrows on the curve, in Fig. 4). These points with normal directions $n$ are then encoded into 2D stick tensors $nn^T$, which cast votes to each imaginary point.

After receiving and accumulating the collected votes (Section 4.1) cast by existing curve points, each imaginary point obtains a curve saliency value, $\lambda_1 - \lambda_2$ (Section 4.2). Assuming that the curve has no intersection and that it is a function in $x$ (otherwise, the hole can be split to enforce this function requirement), only one point $P_{x_i,z_j}$ is selected as the curve point for each *anchor* $x_i$ (Fig. 4), whose other coordinate is equal to $Z_{min}$.[1] The larger the curve saliency the point receives, the more likely the point is actually lying on the curve. For each anchor $x_i$, the curve saliencies of all imaginary points along the $z$-axis are compared. The point $P_{x_i,z_j}$ having the highest curve saliency is selected as the optimal curve point $P_{x_i}$ at anchor $x_i$. The process can be formulated as follows:

$$P_{x_i} = \max\{P_{x_i,z_j}(\lambda_1 - \lambda_2)\} \quad 1 \le j \le S, \tag{4}$$

where $x_i$ is an anchor and $S$ is the sampling density (in pixels) along the $z$-direction. Note that curve normals at $P_{x_i}$s have been computed at the same time after tensor voting is run.

Once the discrete curve points are synthesized, we connect them together using a B-spline. In case of a large hole, we perform curve connection at successive scales by a Gaussian pyramid: From the optimal curve points we inferred at one scale, we warp and upsample inferred curve points in the next finer scale. The same hole filling is then applied, with the upsampled points casting votes as well. This process is repeated until the bottom of the Gaussian pyramid has been reached.

1. If the curve is not a function in $x$, it means that more than one $P_{x_i,z_j}$ has curve saliency maxima. The curve (hole) can be split at $x_i$ to enforce this function requirement.

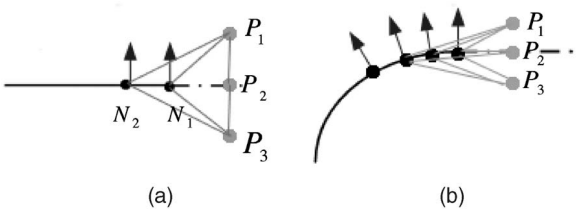


Fig. 5. An analysis of 1D hole filling.

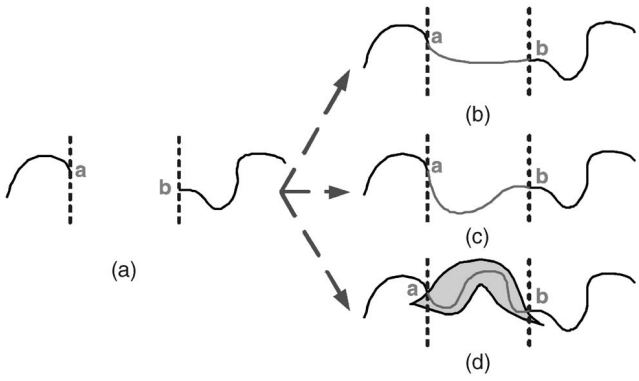


Fig. 6. Hole filling examples. (a) The broken curve. (b) Curve connection result with a large $\sigma$. (c) Result obtained by using a smaller $\sigma$. (d) Curve result, constrained by the shape of the hole.

Since the voting algorithm is dominated by the design of the voting field, we give a qualitative analysis of the correctness of the above method. Consider a simple situation that two point tokens are on a straight line, with three imaginary points: above, on, and below the line, respectively, (Fig. 5b). Since $|\overline{P_1N_1}| > |\overline{P_2N_1}|$ and $|\overline{P_1N_2}| > |\overline{P_2N_2}|$, the received vote strength of $P_2$ is stronger than that of $P_1$. Hence, $P_2$ has the largest curve saliency and the line segment is extended naturally. If $N_1$ and $N_2$ are on a curve, the analysis is similar (Fig. 5b) and $P_3$ receives the largest saliency, due to the sign of the curvature along the existing curve.

There are two advantages to apply discrete tensor voting to synthesize optimal curve points. One is that the $\sigma$ parameter of the voting field can be used to control the smoothness of the resulting curve. Fig. 6 shows one curve connection example. The larger the scale is, the smoother the curve $ab$ is. Thus, we can easily set the appropriate $\sigma$ to avoid oversmoothing. In our experiments, $\sigma$ is set to be a user-defined global value for a whole image. The selection criteria is as follows: For a highly structured image (e.g., buildings and brick walls), a small $\sigma$ is selected to maintain sharp connection. On the other hand, a large $\sigma$ is chosen for natural image to construct smooth curve. Since the Gaussian decay function used in (1), $\sigma \approx 3|V_s|$, where $V_s$ is the size of the voting field. $|V_s|$ is usually set to be a larger value than 1 if a Gaussian pyramid is used, say 2, to obtain a smooth connection curve.

The other advantage is related to the fact that different shapes of image holes constrain different sets of the imaginary points. The imaginary points outside the designated area are

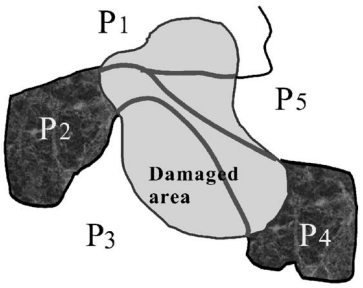


Fig. 7. A hole connection example.

automatically omitted in the voting process. Therefore, various optimal curves can be synthesized accordingly. One example is shown in Fig. 6d. The shaded region represents an image hole. The optimal curve $ab$ synthesized is different from that in Figs. 6b and 6c without the hole constraint, but it is still a natural connection with respect to the hole shape.

#### 5.2.3 Connection Sequence

We facilitate the discussion in the previous section by using an unambiguous case where only two salient curve endpoints are present. In practice, we have the configuration similar to Fig. 7. To complete the segmentation, we connect all similar but disjoint regions surrounding the hole by inferring the most likely smooth connection curves.

To reduce the instabilities and avoid ambiguities, we propose the following connection scheme implemented as a greedy algorithm:

1. Find all disjoint region pairs around an image hole which belong to the same merged segment (Section 5.2.1). Sort these pairs by similarity scores in descending order and put them into a queue.
2. If the queue is not empty, fetch the region pair at the queue head:
   a. If the two regions are already connected, skip them and back to Step 2.
   b. If there exists a possible boundary connection which does not intersect any existing curves, infer the connection curve. Else, skip the current region pair and go back to Step 2.
3. If there exists a single region unclosed, infer new curves to close it (by constraining the shape of the hole).

To illustrate, we examine the partition situation depicted in Fig. 7. Assume that regions $P_2$ and $P_4$ are the most similar, while $P_3$ and $P_5$ are less similar regions. Initially, we enqueue two 2-tuples, region pairs ($P_2$, $P_4$) and ($P_3$, $P_5$), according to the similarity scores. After that, we dequeue them one by one. First, we fetch ($P_2$, $P_4$) from the queue and connect these two regions according to Step 2b. Then, we dequeue ($P_3$, $P_5$). Note that connecting them has to intersect the existing curves from region $P_2$ to $P_4$. According to Step 2b of our method, we skip this pair. Finally, we close region $P_1$ according to Step 3.

### 5.3 Image Synthesis

Once the image has been completely segmented, we synthesize missing data with existing color and texture information. By the MRF assumption, texture is defined by a neighborhood. We propose to use $N$D tensor voting to infer color and texture information, where $N$ indicates neighborhood scale.

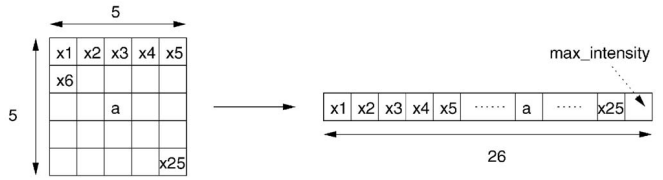


Fig. 8. Encoding a subimage centered at $a$ by a $N$D vector ($N = 26$ here).

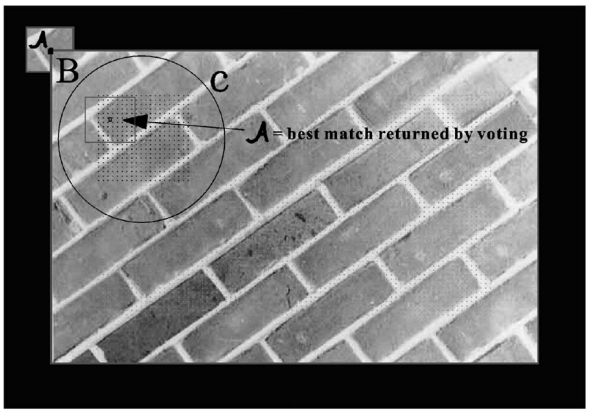


Fig. 9. Synthesizing one pixel. C is the candidate set of pixels considered by $\mathcal{A}$, where $\mathcal{A} \in C$.

#### 5.3.1 $N$D Tensor Representation and Voting

Given a neighborhood window of size $n \times n$ centered at a pixel $A$, we first translate the subimage into a stick tensor (Section 4.1), by producing a feature vector of dimension $N = n \times n + 1$ (Fig. 8) in a lexicographical ordering. Converted grey levels are used if the input is a color image. Thus, a feature vector is represented by homogeneous coordinates, so that zero intensity can be dealt with uniformly (max_intensity in Fig. 8 is equal to the maximum color depth, 256 for instance).

Refer to Fig. 9 which depicts the synthesis of one pixel to illustrate the relationship between the symbols in the following. Suppose we have two pixels $A = (x_a, y_a)$ and $B = (x_b, y_b)$ with their respective $N$D feature vectors $\vec{\mathcal{T}}_A$ and $\vec{\mathcal{T}}_B$, which are encoded as described in Fig. 8. Suppose also that we want to synthesize color at $B$. Therefore, some of the $N$ components in $\vec{\mathcal{T}}_B$ may not have color information. We zero out corresponding components in both vectors before matching $A$ and $B$. We denote the modified vectors by $\vec{\mathcal{T}}_A$ and $\vec{\mathcal{T}}_B$, respectively (note the change of the subscripts), and name $A$ as *sample seed*. In order to avoid too many zero components in both vectors, the number of zero components should be less than $\frac{n}{2}$ in $\vec{\mathcal{T}}_A$ and $\vec{\mathcal{T}}_B$ before synthesis. Else, pixel $B$ will not be synthesized for now. Later, when the colors of other pixels in its neighborhood have been synthesized, the number of zeros in $B$'s feature vector will ultimately be less than $\frac{n}{2}$.

The matching between $\vec{\mathcal{T}}_A$ and $\vec{\mathcal{T}}_B$ is translated into tensor voting for a *straight line in the $N$D space*. Geometrically, this line defines a family of hyperplanes, and this 1D manifold is the minimal constraint on continuity in the high dimensional space, without any a priori assumption. The computation is straightforward in tensor voting terms and is as follows: First, we need to perform the following $N$D encoding for $A$ and $B$ into $A_N$ and $B_N$, respectively:

1. Convert $A$ into $N$D coordinates, denoted by $A_N$. Without losing generality, we can choose

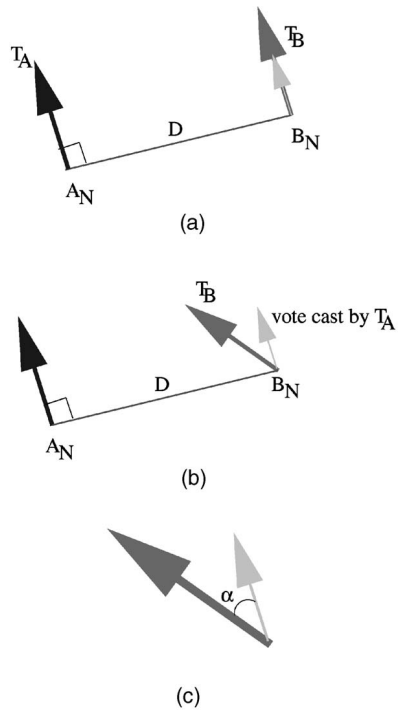


Fig. 10. Vote for colors. (a) The $N$D vectors at $A_N$ and $B_N$ are consistent, indicating that they are lying on the same $N$D straight line. (b) Inconsistent normals are indicated by vote inconsistency. (c) Vote inconsistency is measured by $\alpha$.

$$A_N = (\underbrace{0, \cdots, 0}_{N}).$$

2. Choose any unit direction $\vec{D}$ such that $\vec{\mathcal{T}_A} \cdot \vec{D} = 0$, that is, $\vec{\mathcal{T}_A}$ and $\vec{D}$ are perpendicular to each other. Then, convert $B$ into $N$D coordinates, by

$$B_N = A_N + \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}\vec{D}. \quad (5)$$

Therefore, $\vec{\mathcal{T}_A}$ is a normal to the $N$D straight line connecting $A_N$ and $B_N$. Now, $A_N$ casts a stick vote to $B_N$ in exactly the same way as described in Section 4, except that now a $N$D stick voting field[2] is used: In $N$D, an osculating circle becomes an osculating hypersphere. We can define a $N$D stick voting field by uniform sampling of normal directions in the $N$D space. The construction is exactly the same as the 2D stick voting field, but now in $N$D.

Refer to Fig. 10. When $B_N$ receives a stick vote from $A_N$, the vote will be matched with $\vec{\mathcal{T}_B}$. Vote consistency is indicated by $s \cos\alpha$, where $s = \lambda_1 - \lambda_2$ is the vote saliency given by the $N$D stick voting field, and $\alpha$ is the angle between $\vec{\mathcal{T}_B}$ and the received stick vote.

The total number of sample seeds casting votes to $B_N$ depends on the complete segmentation result, that is, the region size of the segment to which $B$ belongs. Among all sample seeds, let $\mathcal{A}$ be the 2D pixel (corresponding to the $A_N$) whose vote to $B_N$ gives the maximum $s \cos\alpha$ at $B_N$. To synthesize the color at $B$, we replace the zero components in $\vec{\mathcal{T}_B}$ by corresponding nonzero entries in $\vec{\mathcal{T}_A}$ (not $\vec{\mathcal{T}_A}$ where the corresponding entries are zero). In practice, not all zero components are replaced, only zero entries in a small window centered at $B$ are replaced.

The scale $N$, as depicted in Fig. 8, is the crucial parameter in our method. Its value depends on how complex the neighborhood structure is. If the scale is too large, the synthesis process is slow, which also makes the MRF assumption void. If the scale is too small, it is inadequate to capture the necessary structure or feature scale. Hence, we propose an automatic and adaptive scale selection method to determine the value of $N$.

### 5.3.2 Adaptive Scaling

Normally, texture inhomogeneity in images gives difficulty to assign only one global scale $N$. In other words, all sample seeds have their own smallest scale sizes that best capture their neighborhood information. The scale $N_i$ for different region $i$ should vary across the whole image in voting process, as described in the previous Section 5.3.1.

We observe that human eyes are more sensitive to edge discontinuity than to pure color distinctness when synthesis artifact exists. Accordingly, to select an appropriate scale for a sample seed, we compute its edge complexity by accumulating gradients $\nabla I$ within its neighborhood window. Simply summing them up will cancel opposite ones. Hence, the second order moment matrix for the vectors (tensor) within the window are used [4], [36]:

$$M_\sigma(x, y) = G_\sigma(x, y)((\nabla I)(\nabla I)^T), \quad (6)$$

where $G_\sigma$ denotes an Gaussian smoothing kernel with variance $\sigma^2$ centered at a pixel $(x, y)$. Since the tensor encoding process (Section 5.3.1) treats the window center and the boundary points equally, we set $\sigma = 0$ to make the Gaussian decay an averaging function to simplify the notation and computation:

$$M_N = AVG_N\{((\nabla I)(\nabla I)^T)\} \quad (7)$$

$$= \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix}, \quad (8)$$

where $M_N$ is a function of scale $N$. $trace(M_N) = q_{11} + q_{22}$ measures the average strength of the square of the gradient magnitude in the window of size $N$ [36]. By observation, inhomogeneity usually results in abrupt change in gradient magnitude. Therefore, we select the scale for each sample seed to be proportional to the local maxima threshold of $M_N$, as the value of $N$ increases. It gives good estimation in our examples. The detailed scheme is as follows:

1. For each sample seed $A$, increase its scale $N_A$ from the lower bound (typically set to 3, but it does no harm to start from 1) to the upper bound (depending on the image resolution).
2. For each scale $N_A$, compute $trace(M_{N_A})$.
3. If $trace(M_{N_A}) < trace(M_{N_A-1})$, set $N_A - 1 \rightarrow N_A$ and return (i.e., local maxima threshold is reached).
4. Otherwise, we continue the loop by incrementing $N_A$ until a maxima is found, or the upper bound has been reached.

## 6 Range and 3D Repairing

The same framework (complete segmentation and color synthesis) generalizes well into 3D, except that the synthesized color and texture are not necessarily pasted onto a

2. In implementation, we need not use an $N$D array to store the $N$D stick voting field due to its symmetry. It suffices to use a 2D array to store the 2D stick voting field and rotate it in $N$D space.

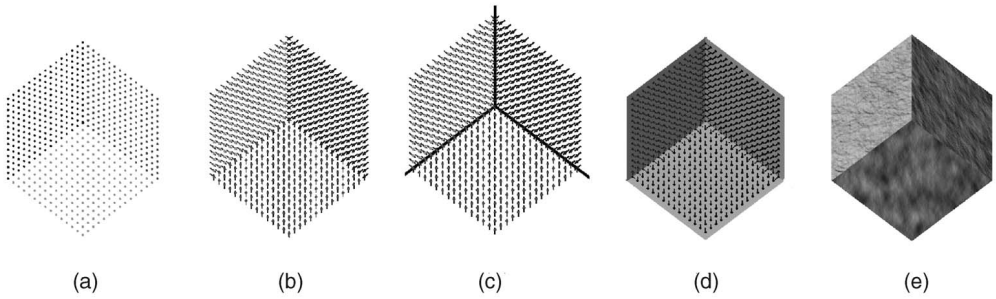


Fig. 11. Rundown of shape and texture syntheses.

planar surface. In this section, we show that the *same* $N$D color synthesis can be applied after performing complete segmentation accordingly. However, if there exist missing values in the given range or 3D data, we need to infer them first.

For range data where each pixel can record only one depth value, missing data are mostly caused by occlusion. We therefore need to fill in the holes in the depth map, by inferring discrete depth values. The same occlusion situation exists in 3D data. Moreover, missing data in 3D can also be caused by insufficient data resolution due to undersampling. In our method, we take a point cloud as 3D input and construct a mesh first. Noisy range and 3D data are regularized and processed by 3D tensor voting, which removes noise, infers missing data, and extracts a triangular surface mesh.

After all missing data have been filled, surface orientation discontinuities are detected as *curve junctions*, also by 3D tensor voting. The curves obtained serve the same purpose as those in 2D image repairing, to avoid mixing up textures with distinct statistical distribution. The 3D curve extraction algorithm can be found in [29], which is analogous to the modified marching cubes algorithm as in the case of 2D curve connection.

The extracted curves segment the range and 3D surface mesh into separate patches with their respective color and texture. Hence, 2D image repairing method can be applied to each patch for complete segmentation and color synthesis.

In summary, the main steps for range and 3D repairing method are:

1. missing range or missing 3D data inference (Sections 6.1 and 6.2),
2. 3D curve extraction [29],
3. mesh construction [29],
4. mesh partitioning or segmentation according to the extracted curves,
5. image repairing are performed on each surface patch.

We use Fig. 11 as a running example to illustrate the overall 3D data repairing process for shape and texture inference. Fig. 11a is obtained after missing data inference (Step 1), Normals shown in Fig. 11b and curve junctions shown as thick lines in Fig. 11c are obtained by 3D tensor voting in Step 2. A surface mesh can be extracted by a modified marching cubes algorithm, Fig. 11d (Step 3). The inferred surfaces and curves segment the scene into a set of piecewise smooth patches (Step 4). The uniform sampling criterion we enforce in Section 6.2 reduces distortion effect. To synthesize color and texture, 2D image repairing is applied separately to each patch (Step 5), where automatic scale adaptation is also performed, Fig. 11e.

In the following sections, we focus on the inference of missing data in complete segmentation for range and 3D data (Step 1). Other steps are either described elsewhere (Steps 2 and 3 in [29], and Step 5 in Section 5), or easy to implement (Step 4).

### 6.1 Inferring Missing Range Data

For range data, a pixel is defined as *defective* if it has no depth value, thus producing a hole in the range image. It is natural to extend the 2D tensor voting algorithm for hole filling we described in Section 5.2.2 to fill in missing depth information since the uniqueness constraint along the line of sight applies to depth map: Each pixel can contain only one depth value.

To achieve hole filling, we extend the voting space in Fig. 4 into 3D, where the $z$-axis represents the depth axis, and $(x, y)$ are the image coordinates. Depth values ($z$ values) vary from $Z_{min}$ to $Z_{max}$. Hence, each defective pixel in the image plane becomes an anchor. Instead of inferring a curve in Fig. 4, we infer a depth surface to cover up the anchors. Accordingly, we apply 3D tensor voting [29] to the *volume* of discrete imaginary points to vote for the true surface points: At each anchor $(x, y)$, a total of $Z_{max} - Z_{min} + 1$ votes are collected. The imaginary point $(x, y, d)$, $d \in [Z_{min}, Z_{max}]$, is selected as the desired surface point if its surface saliency is maximum along $z$ direction for each anchor $(x, y)$, analogous to the 2D situation (Fig. 4).

In order to accelerate the algorithm and not to oversmooth corners or surface orientation discontinuities in the voting process, our implementation has the following considerations:

- *Incremental hole filling*—If the hole size is small, we set the voting field size ($\sigma$) to be large enough so that the hole filling process can be accomplished in a single pass. However, it is not feasible for large holes since large voting field results in oversmoothing (and longer computing time as well). Therefore, we adopt an incremental approach to fill large holes, starting from hole boundary toward the center. The maximum number of passes is thus predetermined and is simply proportional to the size of the hole in pixels.
- *Discontinuity preservation*—Surface orientation discontinuities (e.g., corners) should not be smoothed out. We observe that along the hole boundary, high curvature points have fewer neighbors than low curvature ones. To preserve surface orientation discontinuity inside the hole, we assign higher priority to lower curvature points. They will vote first to incrementally fill the hole from the boundary toward the center, by propagating the smoothness constraint. High curvature points are less preferred since they do not propagate smoothness. Fig. 12c shows the hole filling result without curvature

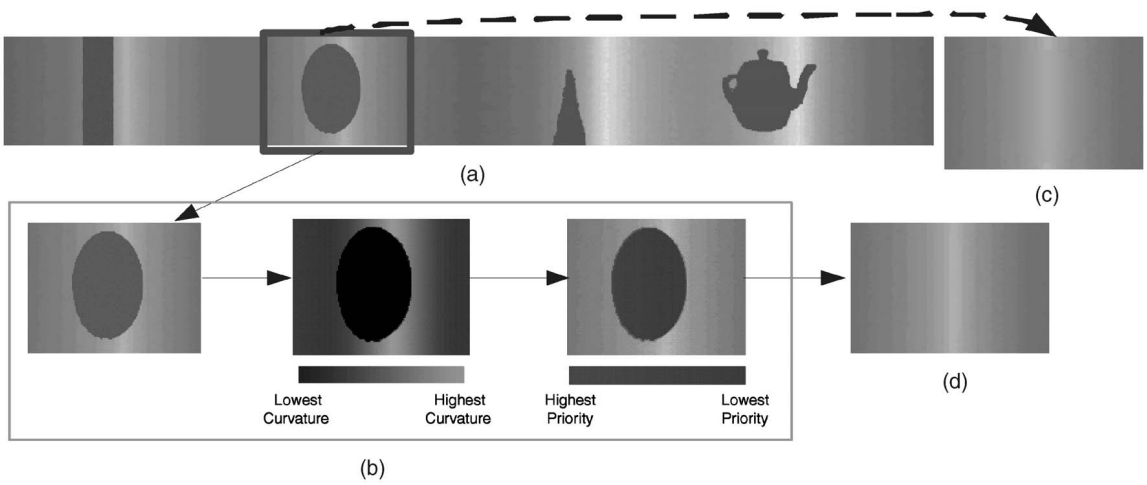


Fig. 12. Priority in range data filling for complete segmentation. Lighter points near high curvature area have lower priority than the darker points. (c) is the filling result without priority consideration, (d) shows better result with priority consideration, where high curvatures are not smoothed out.

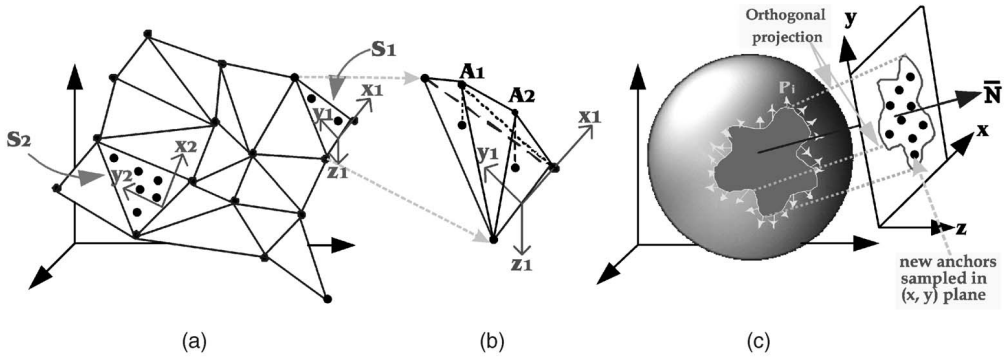


Fig. 13. Three-dimensional data filling illustration for complete segmentation. (a) A 3D mesh with triangles $S_1, S_2 \ldots S_n$, each of which is associated with a local coordinate system. $(\mathbf{x}_i, \mathbf{y}_i)$ defines a plane where the triangle $S_i$ is lying. The $\mathbf{z}_i$ axis is perpendicular to this plane. Black dots (nonmesh vertices) represent anchors. (b) Optimal depth for each anchor in $S_1$ is constructed along the $\mathbf{z}_1$ direction (e.g., $A_1$ and $A_2$). They are connected to form a finer mesh. (c) A more detailed illustration for constructing the initial local coordinate system. Local $z$-direction is the equal to the average of the hole boundary normal directions. The hole area is orthogonally projected onto $(\mathbf{x}, \mathbf{y})$ plane, where anchors are sampled.

priority consideration, while Fig. 12d has taken it into account. The result in Fig. 12d preserves more curvature details in the hole.

### 6.2 Inferring Missing 3D Data (Data Resampling)

Based on the method above, large holes (area with very low resolution) in 3D triangle meshes can be filled in the same way. However, we need to consider the sampling problem so that resulting points are roughly uniformly distributed on the object surface in 3D space. This reduces the distortion effect when surface textures are synthesized. In [26], an algorithm is described to fill complicated holes in meshes, which can be used for this purpose. Complex holes can be filled by [12] to produce a "watertight" surface mesh. Following, we present a simpler solution, and more discussion on this alternative is presented at the end of this section.

To satisfy the uniform sampling criterion, we adopt a divide-and-conquer method to locate anchors and reorganize them. Inspired by the previous section about the range data hole filling, we associate each surface patch with a local coordinate system and set anchors accordingly so that existing patches can be extrapolated to cover up a hole. Fig. 13a shows a triangulated mesh, $\cup_i S_i$, where $S_i$ is a triangle. Each triangle is associated with a local coordinate system, e.g., $S_i$ with $(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)$, where $(\mathbf{x}_i, \mathbf{y}_i)$ defines the same plane as triangle $S_i$. Therefore, we apply hole filling algorithm for range images in the previous section to each triangle and vote for more surface points to obtain a finer triangulation. These new points will be connected together to form a finer resulting surface to increase sampling density, Fig. 13b.

One minor modification is that instead of sampling depth candidates from the given $Z_{min}$ to $Z_{max}$, we constrain the depth range from $\max(Z_{min}, SUR)$ to $\min(Z_{max}, SUR)$ for each anchor $(x_i, y_i)$, where $SUR$ is the nearest surface point along $z$ or $-z$ direction in $(x_i, y_i)$ position. The detailed 3D hole filling algorithm is as follows:

1. Assign a coordinate system $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ to the entire hole, Fig. 13c. Sample anchors in the hole area and apply hole filling for range data to infer initial surface points, which will be triangulated to form finer triangles. The computation of $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ will be explained shortly.
2. If there is any triangle $S_i$ whose area size is larger than some threshold, i.e., insufficient sampling density, repeat the following steps.
3. Associate $S_i$ with a local coordinate system $(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)$ where the $(\mathbf{x}_i, \mathbf{y}_i)$ plane is the triangle plane $S_i$, and $z_i$ direction is perpendicular to it (e.g., $S_1$ in Fig. 13a).
4. Sample $S_i$ according to the assigned density value, and generate anchors in the $(\mathbf{x}_i, \mathbf{y}_i)$ plane (e.g., $S_1$ and $S_2$ in Fig. 13a).

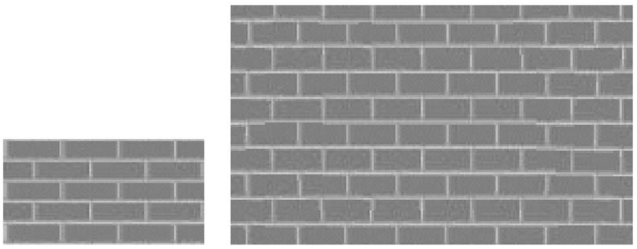

Fig. 14. Result on synthesizing structured textures. Left: input texture sample. Right: a larger synthesized image.

5. Hole filling algorithm for range data is applied to generate new surface points for all anchors. They are connected to form new triangles, Fig. 13b. To construct an integrated mesh given these new surface points, we can either apply 3D tensor voting to the subset of new points for inferring the mesh to connect them, or use the construction and optimization algorithm [19], [20] to generate the resulting mesh, Fig. 13b.

Here, we provide details for computing $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ in Step 1. We want to select an optimal $(\mathbf{x}, \mathbf{y})$ plane which maximizes the orthogonal projection area of the hole. However, it is expensive to accurately locate the maxima due to the varieties of the hole shape. Alternatively, a more intuitive but stable method is adopted to create the initial coordinate system $(\mathbf{x}, \mathbf{y}, \mathbf{z})$, shown in Fig. 13c, by first averaging the normal directions of all hole boundary points as $\overline{N}$, i.e., $\overline{N} = (\sum_{i=1}^{N_B} normal(P_i))/N_B$, where $N_B$ is the number of the boundary points $P_i$. The normals at these boundary points are either given (by the mesh) originally, or computed by tensor voting [29].

Then, we set the $\mathbf{z}$-direction to be equal to $\overline{N}$. The $(\mathbf{x}, \mathbf{y})$ plane is randomly selected as long as $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are perpendicular to each other. Orthogonally project the hole onto the $(\mathbf{x}, \mathbf{y})$ plane and compute anchors according to the sampling density (Fig. 13c).

### 6.3 Discussion

In this section, we have extended our methodology to repair a 3D mesh. Here, we qualify our proposed method, where the four results depicted in Figs. 23, 24, and 25 also show the limitation of our extension, and they can be improved in the future. Our 3D extension works in the presence of discontinuities ($C^0$ continuous but $C^1$ discontinuous), large amount of noise and missing data, as long as the local geometric scales remain approximately constant. When the hole to be repaired covers geometric features of multiple scales, the complete segmentation must be performed also in multiple scales, which is under investigation [39]. For example, because of self-occlusion, the hole may cover both the nose (small scale geometric feature) and cheek (large scale geometric feature) of a defective face mesh at the same time, when only a sparse set of scans is available. The output of the complete segmentation in multiscales is a segmented multiresolution mesh where different scale features are coherently integrated along scale transition (curves), in addition to discontinuity curves described above for restricting the color/texture synthesis, so that only relevant information in each segmented patch is used in the synthesis process.

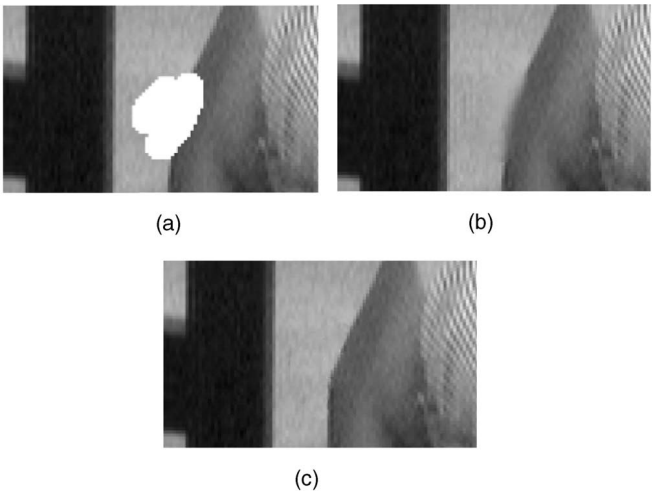


Fig. 15. Comparison with structure and texture image inpainting. (a) Original image extracted from [3]. (b) Structure and texture image inpainting result. (c) Our result.

## 7 Time Complexity Analysis

This section collectively analyzes the time complexity for our repairing methods. Let:

$D_2$ = number of anchors in 2D repairing.
$D_3$ = number of anchors in range/3D repairing.
$B$ = number of pixels along hole boundary ($B \ll D_2, D_3$).
$T$ = number of nondefective pixels in the image.
$R$ = number of regions ($R \ll T$).
$S$ = number of samples ($S = Z_{max} - Z_{min} + 1$).

There are two steps for 2D complete segmentation. In the texture-based region segmentation (Section 5.2.1), it takes $O(T)$ time to compute the joint intensity and gradient vector, and merge regions. It results in a total of $R$ regions. For curve connection, it takes $O(R^2)$ time to pair up regions, before putting them into a priority queue (very efficient

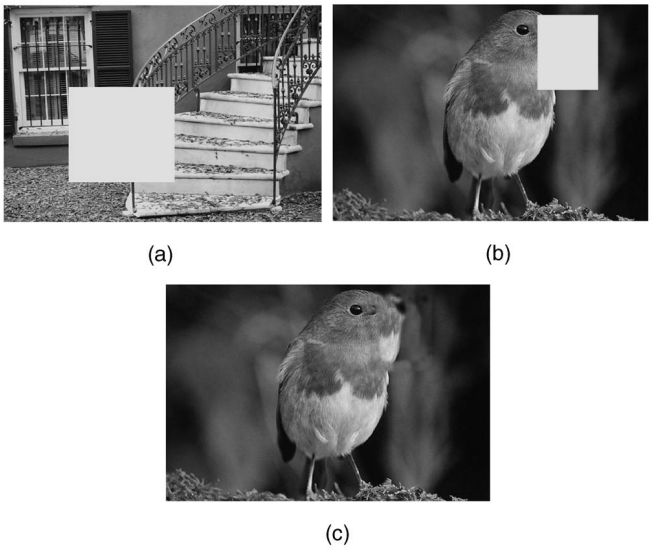


Fig. 16. Limitations of our method. Pixels inside the yellow or shaded rectangle cannot be well synthesized. (a) Handrail and stairs. (b) Bird beak. (c) Result on applying image repairing without any a priori knowledge.


Fig. 17. Sign post. Left: original image. Middle: masked image. Right: our result.


Fig. 18. Beach and Moor. Left: original images. Middle: damaged images by erasing some objects. Right: our image synthesis results. In Moor, we repair the full rainbow and hallucinate the upper, very faint rainbow "arc," which is largely occluded by the sign post.

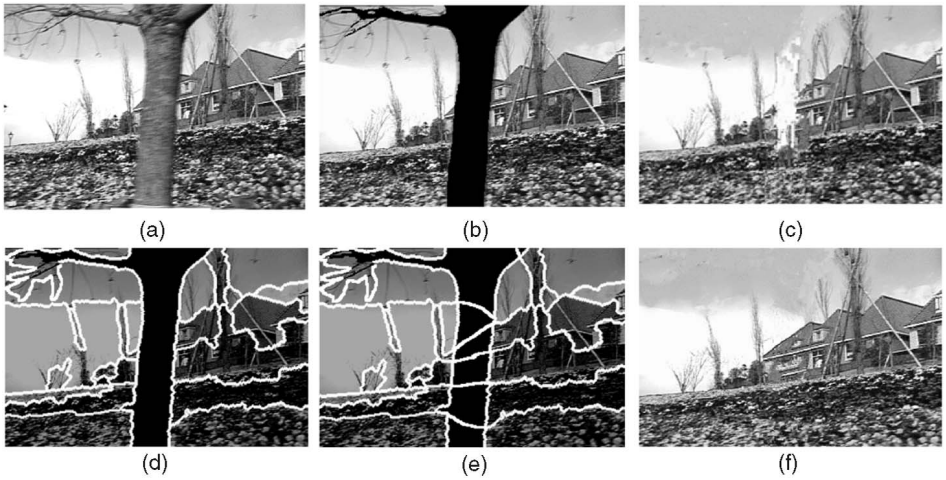


Fig. 19. Flower garden example. (a) Only a single image is used. (b) Damaged image by erasing the tree. (c) Result obtained by pure texture synthesis algorithm. (d) Image segmentation. (e) Curve connection. (f) Our image synthesis result.

Fibonacci heap can be used to implement the queue). $O(SD_2)$ time is spent for 2D tensor voting: $SD_2$ tensor votes are collected at the imaginary points inside the hole for computing the optimal curve points. There are at most $O(B)$ endpoints with their finite neighborhood where we sample the tensor votes for curve extrapolation. Summing up, the total time is $O(T + R^2 + SD_2 + B) \approx O(T + SD_2)$, that is, linear with the total number of pixels in the given image.

For complete segmentation in range and 3D data, after spending $O(B)$ time to fix the local coordinate system, 3D tensor voting is used to infer the (depth) surfaces and partitioning curves, which takes $O(SD_3)$ time.

For color and texture synthesis, once the scale for a region has been chosen (the time spent on scale iteration is related to image resolution and is linear with $T$, the total number of pixels), we perform $N$D tensor voting. To synthesize the color for each anchor, only a finite neighborhood need to be

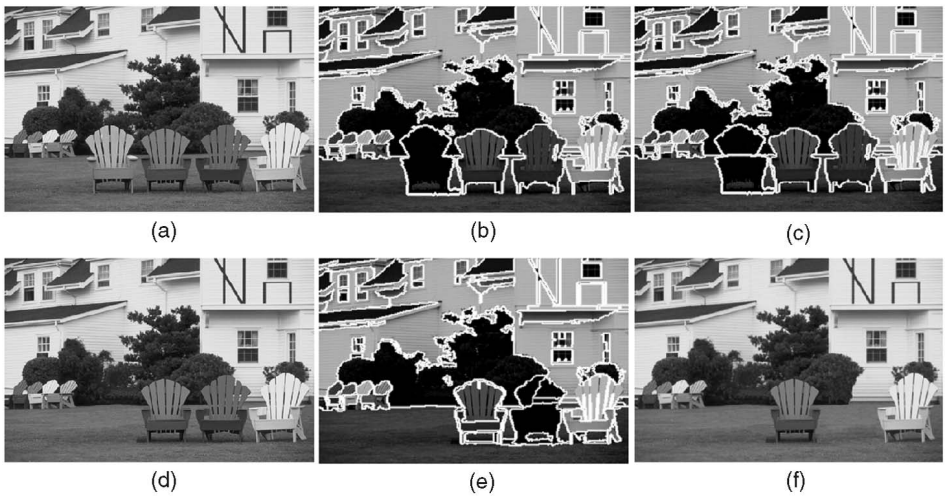


Fig. 20. Chair example. (a) Original image. (b) Image segmentation before curve connection, after removing the leftmost chair. (c) Complete segmentation after curve connection. (d) The repaired image. (e) Complete segmentation after removing the chair second from right. Note the complex shape of the curves inside the hole, which are inferred by our 2D hole filling algorithm using tensor voting. (f) Our result with two chairs left.

considered, by the MRF assumption. Therefore, at most $O(T + SD_3)$ time is spent.

The above analysis is valid when the size of the voting field is not large. Except for very sparse data, applications such as image/mesh restoration usually involves (quasi-)dense data, except at the hole. Therefore, $\sigma$, which used to define the voting field size, needs not be large. For large holes, incremental hole filling is performed to keep a small $\sigma$ in inference. Therefore, the size of the voting field can be treated as a constant in 2D, 3D, or $N$D. Tensor voting is thus a linear-time algorithm regardless of dimensionality. Typical range of $\sigma$ we use is 1 to 2.

## 8 Results

### 8.1 2D Image Results

We first show one texture synthesis result on regularly textured patterns (Fig. 14) where rigid structures are maintained.

One emphasis of this paper is repairing damaged images of real scenes. We have experimented a variety of such difficult natural images, most of them contain large and complex holes with difficult neighborhood topologies.

Figs. 17 and 18 show three examples. From a single damaged or masked image (middle) with a large amount of missing complex information, we are capable of synthesizing new pixels. By adaptively propagating neighborhood information, our method smoothly generates texture patterns without blurring important feature curves (right). The left images are original images with the occluding objects. There exist many methods to erase these objects, which are out of the scope of this paper. The original images (left) are provided for comparison.

Next, we also show some intermediate results for 2D image repairing, in addition to the final repaired images. From a single image of a flower garden with the occluding tree removed, we synthesize the result in Fig. 19f. The detailed process is: We first segment the damaged image. It is followed by a merging process described in Section 5.2.1 (Fig. 19d). Then, we complete our segmentation by curve connection using 2D tensor voting described in Section 5.2.2 (Fig. 19e). There are two key factors contributing to the success of this difficult example. First, thanks to the automatic color and texture segmentation [13] in Fig. 19d, a set of reliable partitioning curves incident at the hole boundaries is produced. Next, our complete segmentation infers the curves inside the hole, which provides sufficient information so that the color synthesis step is capable of distinguishing which pixels are relevant in the synthesis process. In this and other experiments, though there is a set of default thresholds or parameters in our implementation, they are either unchanged or not critical, making our method fully automatic. Fig. 20 shows another result, obtained using the same process. Some intermediate results are also shown.

Fig. 15 compares our method with image structure and texture image inpainting [3]. Since our method infers

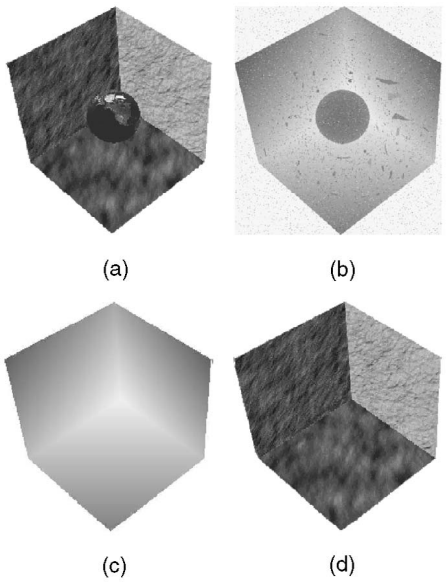


Fig. 21. Noisy corner example. (a) initial color image (the ball is an occluder), (b) initial noisy range data, (c) the recovered background layer (the occluded corner is not smoothed out), (d) the same view with automatically recovered background texture which fills the background hole and does not smooth out the corner.

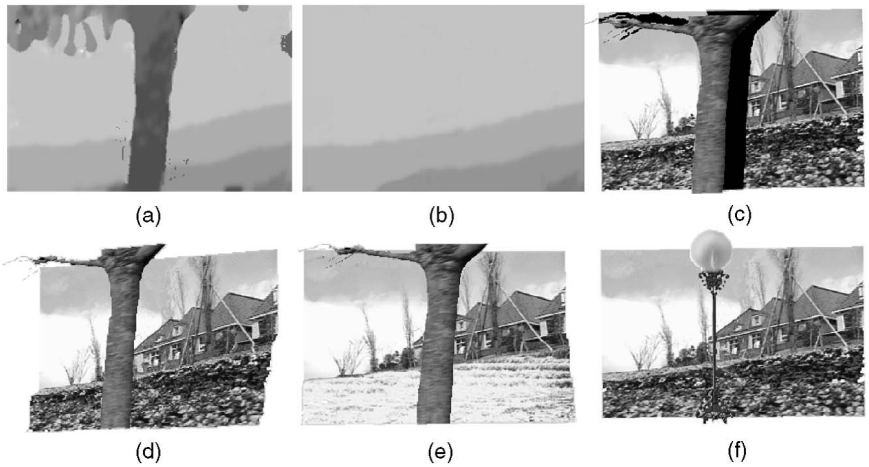


Fig. 22. Flower garden example with depth. (a) The input depth image after noise elimination and smoothing, (b) the background reconstruction result, in which depth discontinuities are preserved, (c) the result of traditional image warping, where holes due to occlusion are seen, (d) the same novel view we generated after running our system. In (e), we turn the flower garden into a water garden. (f) Alternatively, we can replace the tree truck by a lamp post without seeing the hole since a layered and textured description has been inferred.

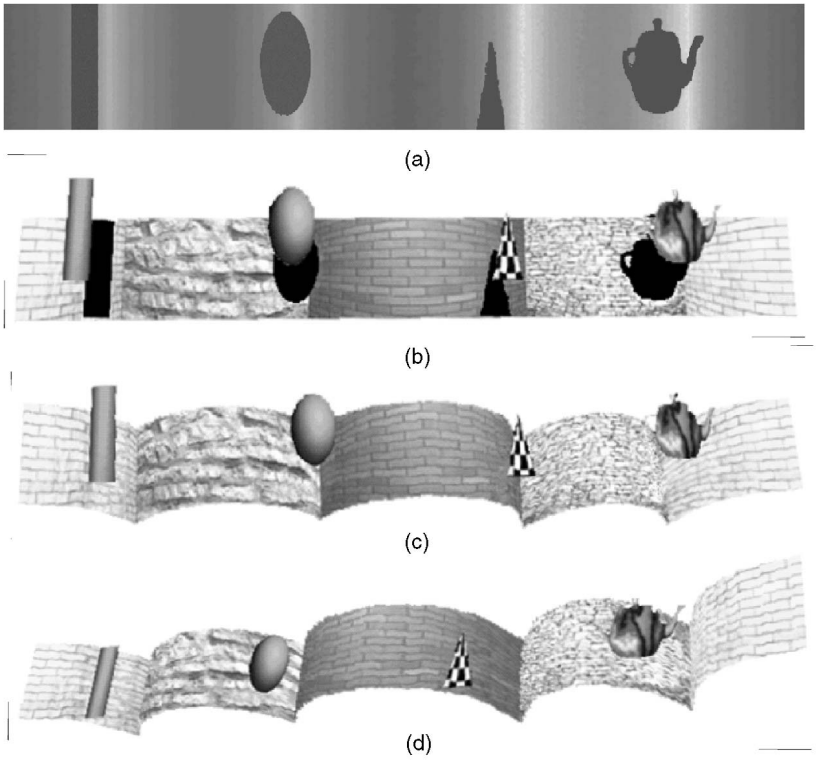


Fig. 23. Panorama. (a) Input noisy depth image. (b) Result of traditional image warping when an image at a novel view is rendered. (c) and (d) our results. Curve junctions and texture boundaries are preserved.

explicit segmentation information and separates different pattern information, it retains texture structures and discontinuities in regions, and does not introduce blurring. Another direct comparison is shown in Fig. 19c, which shows that texture synthesis technique such as [15] is not suitable in our task of repairing natural images. Without proper segmentation, pixels belonging to distinct statistical distributions are mixed together.

The actual running time of our method depends on the image texture and hole complexity. In our experiments, given a $400 \times 300$ pixels image and 10,000 pixels hole area, our method outputs the repaired images in less than 15 minutes on a Pentium III 1GHz PC.

### 8.2 Three-Dimensional and Range Data Results

Fig. 21a shows the input color image and Fig. 21b shows the corresponding depth map of the noisy corner example. Three-dimensional tensor voting is used to recover the background and fill the hole. Complete segmentation is represented by the inferred 3D surface, curve, and point junctions. As a result, the surface is optimally partitioned into three patches and textures are synthesized accordingly without undesirable mixture, Fig. 21d.

The flower garden example with depth is used in Fig. 22. The input depth map is shown in Fig. 22a, where noise has been removed. The background depth map is filled by 3D repairing after removing the tree trunk, Fig. 22b. A hole


Fig. 24. Mountain scene example. (a) is the original image with per-pixel depth, (b) is the hole filling result, after the trees and the hiker have been removed, (c) is the color and texture synthesis result from the original image without the foreground objects (the trees and the hiker), (d) note that using the original range data, holes are observed when viewpoint are changed, and (e) and (f) are rendered after overlapping textured layers are inferred by 3D and $N$D tensor voting, which perform complete segmentation and color/texture synthesis effectively.

is exposed when viewpoint changes, Fig. 22c. Note specifically that, after segmenting the background into three layers as shown in Fig. 22b, we input them to 2D image repairing separately (Section 6). The sky is distinguished from the house in 2D image repairing process that follows, which guarantees that different patterns will not be mixed up in the synthesis process. The result rendered at another viewpoint is shown in Fig. 22d. Figs. 22e and 22f indicate some potential applications with the segmented and textured description we inferred.

Fig. 23a shows a panoramic depth map we computed in [25] using multibaseline stereo. When viewpoint changes, the previously occluded background is exposed, Fig. 23b. Complete segmentation is performed by 3D tensor voting, which infers the missing depth values without smoothing out the junction curves, Fig. 23d. $N$D color and texture synthesis are performed to fill up the holes. Two views of the depth maps with color values are shown in Figs. 23c and 23d.

Another range data example is shown in Fig. 24. The input and the repaired depth map image are shown in the top of the figure. The hole in the depth map, resulted by removing the trees and the hiker, is filled by complete segmentation, implemented in 3D. $N$D tensor voting votes for the colors and textures inside the hole and the resulting image is shown in Fig. 24c. If we use the original input only, the image rendered at another viewpoint exposes a previously occluded background, Fig. 24d. By using our segmented and textured description, we render novel views in Figs. 24e and 24f with little artifact, except for the shadows.

Four results on 3D repairing are collectively shown in Fig. 25, which are generated using 3D data inference method (Section 6) in 3D repairing, followed by texture synthesis using our $N$D tensor voting methodology. Note the large holes in the teapot example (Fig. 25d). We show the closeup views in Fig. 25e from different angles to depict the nontextured meshes, before and after applying our 3D repairing method.

## 9 Discussion, Conclusion, and Future Work

Fig. 16 shows some limitations of our method if critical information is missing or insufficient. One example consists of various shapes in Fig. 16a, in which the missing portion of the handrail and stairs cannot be well synthesized due to

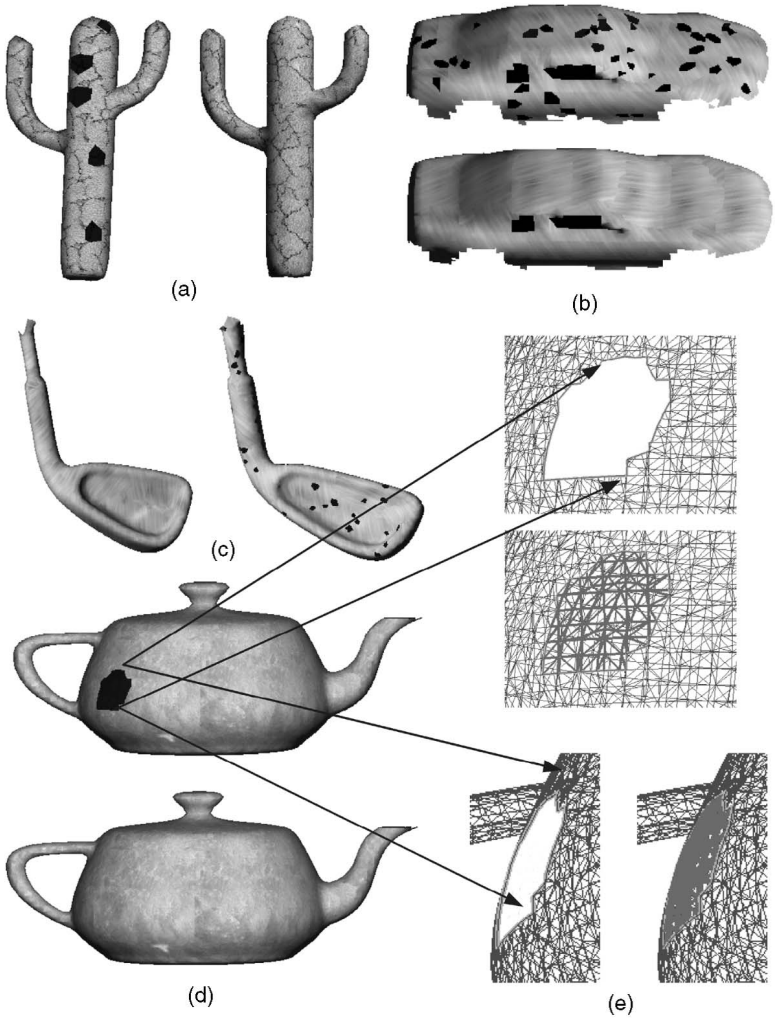


Fig. 25. Some results in 3D. (a) cactus part, (b) car, (c) golf club, and (d) teapot. Defective meshes with holes, repaired and textured meshes are shown. (e) is two closeup views on teapot example with holes from different viewing angles, where the generated meshes are not textured for display clarity.

a complex background and their irregular shapes. The other example consists of an entirely missing bird beak in Fig. 16b. Fig. 16c shows our 2D image repairing result, which is obviously incorrect without employing any knowledge. In both cases, additional knowledge on the scene is needed. Also, in the ambiguous situation where the missing area covers the intersection of two perpendicular regions with similar textures, our complete segmentation may pick the suboptimal interpretation in the absence of knowledge.

To conclude this paper, we have proposed a new method to automatically repair and restore damaged images. Our method is capable of dealing with large holes where missing details can be complex and inhomogeneous. They cannot be described by a small set of statistical parameters. Pure texture synthesis technique will fail on those input images. Real images of natural scenes are typical examples. We address this difficult problem by complete segmentation and robust curve connection. Adaptive $N$D tensor voting provides a unified basis to implement many of these tasks. Our image repairing can be generalized to range and 3D data as well: Complete segmentation is performed in 3D by 3D tensor voting to infer missing data and partition surface patches. The same adaptive $N$D tensor voting for color/texture synthesis is used in 2D, range, and 3D data. We have demonstrated very encouraging results on natural images using our method, and performed some comparison.

In the future, we propose to scale up the image repairing technique to process videos, and to repair damaged video sequence. It has important application to film restoration: Many films that are half a century old are in need of restoration. Image repairing produces spatially coherent and visually acceptable results for large image holes. The challenge for video repairing is the efficient exploitation of spatial and temporal coherence inherent in the large number of frames to achieve visually plausible restoration.

## ACKNOWLEDGMENTS

The authors would like to thank all the reviewers for their constructive comments to improve the final manuscript. This research is supported by the Research Grant Council of Hong Kong Special Administrative Region, China: HKUST6171/03E.

## REFERENCES

[1] C. Ballester, V. Caselles, J. Verdera, M. Bertalmio, and G. Sapiro, "A Variational Model for Filling-In," *Proc. Int'l Conf. Image Processing*, 2003.

[2] M. Bertalmio, G. Sapiro, C. Ballester, and V. Caselles, "Image Inpainting," *Proc. SIGGRAPH 2000 Conf.*, pp. 417-424, 2000.
[3] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous Structure and Texture Image Inpainting," *Proc. Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 707-712, 2003.
[4] J. Bigün, "Local Symmetry Features in Image Processing," PhD thesis, Linköping Univ., Sweden, 1988.
[5] J.S. De Bonet, "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images," *Proc. SIGGRAPH'97 Conf.*, pp. 361-368, 1997.
[6] R.A. Brooks, "Model-Based Three-Dimensional Interpretations of Two-Dimensional Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 140-150, 1983.
[7] J.C. Carr, W.R. Fright, and R.K. Beatson, "Surface Interpolation with Radial Basis Functions for Medical Imaging," *IEEE Trans. Medical Imaging*, vol. 16, no. 1, pp. 96-107, 1997.
[8] T. Chan and J. Shen, "Non-Texture Inpaintings by Curvature-Driven Diffusions," Technical Report 00-35, Dept. of Math., Univ. of Calif. at Los Angeles, 2000.
[9] C.-F. Chang, G. Bishop, and A. Lastra, "Ldi Tree: A Hierarchical Representation for Image-Based Rendering," *Proc. 26th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 291-298, 1999.
[10] A. Criminisi, P. Perez, and K. Toyama, "Object Removal by Exemplar-Based Inpainting," *Proc. Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 721-728, 2003.
[11] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," *Proc. 23rd Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 303-312, 1996.
[12] J. Davis, S. Marschner, M. Garr, and M. Levoy, "Filling Holes in Complex Surfaces Using Volumetric Diffusion," *Proc. First Int'l Symp. 3D Data Processing, Visualization, and Transmission*, 2002.
[13] Y. Deng and B.S. Manjunath, "Unsupervised Segmentation of Color-Texture Regions in Images and Video," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 800-810, Aug. 2001.
[14] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-Based Image Completion," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 303-312, 2003.
[15] A. Efros and T. K. Leung, "Texture Synthesis by Non-Parametric Sampling," *Proc. Seventh Int'l Conf. Computer Vision*, pp. 1033-1038, 1999.
[16] A.A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Proc. 28th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 341-346, 2001.
[17] I. Guskov and Z. Wood, "Topological Noise Removal," *Proc. Graphics Interface Conf.*, 2001.
[18] D.J. Heeger and J.R. Bergen, "Pyramid-Based Texture Analysis/Synthesis," *Proc. 22nd Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 229-238, 1995.
[19] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," *Proc. 19th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 71-78, 1992.
[20] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh Optimization," *Proc. 20th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 19-26, 1993.
[21] J. Jia and C.-K. Tang, "Image Repairing: Robust Image Synthesis by Adaptive ND Tensor Voting," *Proc. Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 643-650, 2003.
[22] S.B. Kang, R. Szeliski, and J. Chai, "Handling Occlusion in Dense Multi-View Stereo," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001.
[23] M. Kass, A. Witkin, and D. Terzopoulous, "Snakes: An Active Contour Models," *Int'l J. Computer Vision*, vol. 1, no. 4, pp. 321-331, 1987.
[24] A. Levin, A. Zomet, and Y. Weiss, "Learning How to Inpaint from Global Image Statistics," *Proc. Int'l Conf. Computer Vision*, pp. 305-312, 2003.
[25] Y. Li, C.-K. Tang, and H.-Y. Shum, "Efficient Dense Depth Estimation from Dense Multiperspective Panoramas," *Proc. Eighth Int'l Conf. Computer Vision*, pp. 119-126, 2001.
[26] P. Liepa, "Filling Holes in Meshes," *Proc. Eurographics Symp. Geometry Processing*, vol. 2, pp. 721-728, 2003.
[27] W.R. Mark, L. McMillan, and G. Bishop, "Post-Rendering 3D Warping," *Proc. 1997 Symp. Interactive 3D Graphics*, pp. 7-16, 1997.
[28] S. Masnou and J. Morel, "Level Line(s) Based Disocclusion," *Proc. Int'l Conf. Image Processing*, pp. 259-263, 1998.
[29] G. Medioni, M.S. Lee, and C.K. Tang, *A Computational Framework for Feature Extraction and Segmentation.* Elseviers Science, 2000.
[30] E.N. Mortensen and W.A. Barrett, "Intelligent Scissors for Image Composition," *Proc. 22nd Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 191-198, 1995.
[31] D.R. Peachey, "Solid Texturing of Complex Surfaces," *Proc. 12th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 279-286, 1985.
[32] P. Perez, A. Blake, and M. Gangnet, "Jetstream: Probabilistic Contour Extraction with Particles," *Proc. Int'l Conf. Computer Vision*, vol. 2, pp. 524-531, 2001.
[33] K. Perlin, "An Image Synthesizer," *Proc. 12th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 287-296, 1985.
[34] R. Pfeifle and H.-P. Seidel, "Triangular B-Splines for Blending and Filling of Polygonal Holes," *Proc. Graphics Interface Conf.*, pp. 186-193, 1996.
[35] J. Portilla and E.P. Simoncelli, "A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients," *Int'l J. Computer Vision*, vol. 40, no. 1, pp. 49-70, Oct. 2000.
[36] J. Gårding and T. Lindeberg, "Direct Computation of Shape Cues Using Scale-Adapted Spatial Derivative Operators," *Int'l J. Computer Vision*, vol. 17, no. 2, pp. 163-191, 1996.
[37] H.-Y. Shum and L.-W. He, "Rendering with Concentric Mosaics," *Proc. 26th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 299-306, 1999.
[38] J.M. Snyder and J.T. Kajiya, "Generative Modeling: A Symbolic System for Geometric Modeling," *Proc. 19th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 369-378, 1992.
[39] W.-S. Tong and C.-K. Tang, "Multiscale Surface Reconstruction by Tensor Voting," technical report, Dept. of Computer Science, Hong Kong Univ. of Science and Technology, 2004.
[40] G. Turk and J.F. O'Brien, "Variational Implicit Surfaces," Technical Report GIT-GVU-99-15, Graphics, Visualization, and Useability Center, Georgia Inst. of Technology, 1999.
[41] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro, "Inpainting Surface Holes," *Proc. Int'l Conf. Image Processing*, 2003.
[42] J. Wang and M.M. Oliveira, "A Hole Filling Strategy for Reconstruction of Smooth Surfaces in Range Images," *Proc. Brazilian Symp. Computer Graphics and Image Processing (SIBGRAPI03)*, 2003.
[43] L.-Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. 27th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 479-488, 2000.
[44] A. Witkin and M. Kass, "Reaction-Diffusion Textures," *Proc. 18th Ann. Conf. Computer Graphics and Interactive Techniques*, pp. 299-308, 1991.
[45] Y.Q. Xu, S.C. Zhu, B.N. Guo, and H.Y. Shum, "Asymptotically Admissible Texture Synthesis," *Proc. Second Int'l Workshop Statistical and Computational Theories of Vision*, 2001.
[46] H. Yamauchi, J. Haber, and H.-P. Seidel, "Image Restoration Using Multiresolution Texture Synthesis and Image Inpainting," *Proc. Computer Graphics International (CGI) Conf.*, pp. 120-125, 2003.

**Jiaya Jia** received the BSc degree in computer science from the Fudan University, ROC, in 2000. He is pursuing the PhD degree in the Department of Computer Science, Hong Kong University of Science and Technology. He is currently a visiting student at the Visual Computing Group of Microsoft Research Asia. His research interests include image and video processing, image-based rendering and pattern recognition. He is a student member of the IEEE Computer Society.

**Chi-Keung Tang** received the MS and PhD degrees in computer science from the University of Southern California, Los Angeles, in 1999 and 2000, respectively. He has been with the Computer Science Department at the Hong Kong University of Science and Technology since 2000, where he is currently an assistant professor. He is also an adjunct researcher at the Visual Computing Group of Microsoft Research, Asia, working on various exciting research topics in computer vision and graphics. His research interests include low to midlevel vision such as segmentation, correspondence, shape analysis, and vision and graphics topics such as image-based rendering and medical image analysis. He is a member of the IEEE Computer Society.