Representing Images Using Curvilinear Feature Driven Subdivision Surfaces

Hailing Zhou, Jianmin Zheng, and Lei Wei

***Abstract*— This paper presents a subdivision-based vector graphics for image representation and creation. The graphics representation is a subdivision surface defined by a triangular mesh augmented with color attribute at vertices and feature attribute at edges. Special cubic B-splines are proposed to describe curvilinear features of an image. New subdivision rules are then designed accordingly, which are applied to the mesh and the color attribute to define the spatial distribution and piecewise-smoothly varying colors of the image. A sharpness factor is introduced to control the color transition across the curvilinear edges. In addition, an automatic algorithm is developed to convert a raster image into such a vector graphics representation. The algorithm first detects the curvilinear features of the image, then constructs a triangulation based on the curvilinear edges and feature attributes, and finally iteratively optimizes the vertex color attributes and updates the triangulation. Compared with existing vector-based image representations, the proposed representation and algorithm have the following advantages in addition to the common merits (such as editability and scalability): 1) they allow flexible mesh topology and handle images or objects with complicated boundaries or features effectively; 2) they are able to faithfully reconstruct curvilinear features, especially in modeling subtle shading effects around feature curves; and 3) they offer a simple way for the user to create images in a freehand style. The effectiveness of the proposed method has been demonstrated in experiments.**

***Index Terms*— Image representation, subdivision surface, curvilinear feature, image reconstruction, image creation.**

I. Introduction

TWO typical representations for images are raster graphics and vector graphics. Raster graphics is a grid of pixels, with each pixel storing either a color/grey value or an index into a color palette. Pixel based image representation has revealed its drawbacks in many applications such as image magnification and printing. Vector graphics defines images using graphics objects like curves and surfaces, rather than discrete points of pixels. The vector representation is scalable, compact and resolution independent. It facilitates many image editing processes, for example, object-based image editing [1]. In recent years the research of exploring graphics based image representations and converting images from raster graphics into vector graphics has attracted increasing interest [2]–[12].

Various vector graphics primitives have been proposed to represent images, which include lines, curves and polygons supporting linear or radial color gradients of images [6], [13], [14], and surface patches such as quad-mesh based bicubic parametric surfaces describing smoothly varying colors [8]–[10]. Commercial software such as Adobe Illustrator and CorelDraw has succeeded in adopting vector graphics to represent and create image contents. However, existing graphics-based image representations still face technical challenges due to the complexity of color or intensity changes in many images. First, while graphics primitives are good for representing smooth geometry, an image usually contains both smoothly shaded regions and curvilinear features such as the boundaries of an object in the image, across which the color or intensity gradients are not continuous (see Figs. 12 and 13). Second, the curvilinear features may be in arbitrary shapes and orientations. The distribution of the features in an image could have complicated topology (see Figs. 12(right) and 13(c)). The topological constraints imposed by the structure of the existing graphics representations make them difficult to have a highly flexible spatial layout. Third, there may exist subtle shading effects around curvilinear features (see Fig. 13(b, c)). Many vector-based image representations are too rigid to simulate the subtle color changes properly due to the geometric continuity constraint. Fourth, it is also observed that most existing image representations do not facilitate the process that artists create images. To create an image (for example, Fig. 11(e)), the artists usually work by sketching some curves such as Fig. 11(a) with colors as the important visual cues.

Motivated by the above observations, we aim to design powerful vector representation that is able to effectively approximate raster images and their curvilinear features as well. The dedicated graphics primitives are special cubic B-splines for curvilinear features and a triangular mesh based subdivision surface for image representation. Both the B-spline curves and the subdivision surface are defined in R^5 with the first two channels for the location of the image and the last three channels for the color surface. Cubic B-splines are a popular representation for freeform smooth curves. We propose a specifically-designed cubic B-spline for an open curvilinear edge and use a periodical cubic B-spline for

Manuscript received April 25, 2013; revised December 27, 2013; accepted May 20, 2014. Date of current version June 23, 2014. This work was supported in part by the Multiplatform Game Innovation Centre, and in part by the Singapore National Research Foundation through the IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office, Media Development Authority. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Béatrice Pesquet-Popescu.

H. Zhou and L. Wei are with the Centre for Intelligent Systems Research, Deakin University, Burwood, VIC 3125, Australia (e-mail: hailing.zhou@hotmail.com; lei.wei@deakin.edu.au).

J. Zheng is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: asjmzheng@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2014.2327807

a closed curvilinear edge. These B-splines' refinement rules can be easily incorporated into the mesh subdivision scheme. A triangular mesh has flexible structure and is able to represent image contents of arbitrary topology. The subdivision scheme is an extension of conventional Loop subdivision. It is worth pointing out that directly using Loop subdivision for images may not be feasible for two main reasons: (1) Loop subdivision generates surfaces that are smooth everywhere while for images, there often exists discontinuity across the curvilinear edges; and (2) there is no apparent connection between Loop subdivision surfaces and curvilinear features. We design new mesh subdivision rules to allow the incorporation of sharp or semi-sharp features represented by the special cubic B-splines. As a result, we present a curvilinear feature driven subdivision surface representation that can effectively define images with piecewsie-smoothly varying intensity or color. Furthermore, we also develop an automatic algorithm to convert an image or an object in an image into the proposed vector representation.

Compared to previous work, our proposed representation and algorithm have several advantages in addition to the common merits (such as editability and scalability) of vector representations: (1) they support flexible mesh topology and thus are able to handle images or objects with complicated boundaries or features effectively; (2) they are able to faithfully reconstruct curvilinear features, especially in modeling subtle shading effects around feature curves; and (3) they offer a simple way for the user to create images in a freehand style. The experiments have confirmed these advantages.

The rest of the paper is organized as follows. Section II reviews some related work. Section III proposes a subdivision surface-based representation for images. Section IV presents an algorithm for image reconstruction using the proposed subdivision representation. Section V describes a way for curvilinear feature driven image creation. Section VI provides the experiments to demonstrate the effectiveness of the proposed representation. Section VII concludes the paper.

II. Related Work

Extensive research has been conducted to explore various methods for representing images. A class of methods is based on region decomposition. RaveGrid [13] extracts feature edges from an image, performs a constrained Delaunay triangulation on the vertices of the feature edges to yield triangles that tile the image, and merges the triangles into polygons according to some perceptual grouping criteria. Froumentin *et al.* [6] propose to decompose an image into regions that have relatively homogeneous colors. For each region, the boundary is fitted by a NURBS curve and the interior colors is specified using a constrained Delaunay triangulation. The curves associated with simple colors and the corresponding triangulations thus form a scale independent representation of the image. ARDECO [14] detects smooth regions and fits their boundaries by cubic spline curves. The interior colors are approximated using flat colors, or linear or circular gradients. In these approaches, the color variation in each region is generally assumed to be relatively plain. Recently, Xia *et al.* [12] propose to decompose an image into a set of non-overlapping regions, each of which is parameterized as a triangular Bézier patch with curved boundaries aligning with image features. The colors in the interior of the regions are approximated by a smooth function which is obtained by solving a non-linear optimization problem. This approach can efficiently represent an image with complicated region layout and piecewise smoothly varying color distribution, but the continuity between patches is difficult to control. In contrary, our approach can automatically handle the continuity among patches due to the merit of subdivision surfaces.

Due to flexible arrangement of triangles, a triangular mesh is a popular form for geometric representation and it is also found useful in image processing and computer vision, for example, in feature detection [15], image/video compression [16], [17], medical image analysis [18], [19], image registration [20], and motion track and compensation [20], [21]. Su *et al.* [22] propose a pixel-level triangulation method, which connects all the pixels by order, forming a quadrilateral mesh, and then splits each quad into two triangles by inserting one of two diagonals. The determination of inserted diagonal is based on the local image features. Yu *et al.* [7] propose a data dependent triangulation, which iteratively swaps an input grid of triangles to locally minimize the costs of involved triangle edges. The methods can reduce the visibility of artifacts and preserve edge features well. These two methods usually require a large number of triangles, which makes the process of further editing images inconvenient. Therefore methods have been developed to adaptively sample image at significant pixels that serve as mesh vertices. Yang *et al.* [3] propose a fast content-adaptive mesh generation method, which places mesh vertices according to the classical Floyd-Steinberg error diffusion. As a result, dense sampling is obtained in regions with high-frequency features and sparse sampling is in plain regions. In [4], a coarse to fine mesh generation method is proposed, which uses binary space partitions to refine triangles until the approximate error is smaller than a predefined threshold. Adams [5] proposes a flexible strategy for tradeoff between mesh quality and computational complexity in content-adaptive mesh generation. In these methods, colors at vertices are obtained from the original image directly or by fitting the original image. Within each triangle, color varies linearly. The linear interpolation of these methods actually lowers the capability of their representation.

To better represent smoothly varying colors, some formulations of bicubic patches commonly used in computer graphics are adapted for images. Price and Barrett [8] employ a collection of rectangular bicubic Bézier patches to represent an image. A local greedy strategy is used to refine patches to improve the representation accuracy, which however may result in many tiny patches. A gradient mesh defined by Ferguson patches is introduced by Sun *et al.* [9] and a semi-automatic algorithm is developed for creating an optimal gradient mesh. For images with complicated color variations, the user assistance is needed for initial mesh placement, which takes certain time and effort. Lai *et al.* [10] improve the gradient mesh method by developing an automatic algorithm that is able to generate mesh lines for objects of arbitrary topology. The underlying technical components are formulated

as linear problems. Thus the algorithm is fast and simple. In these methods, the surface patches have intrinsically the layout of rectangular structure, which makes them difficult in handling distinct image features and complicated topological distribution of regions in images. In comparison, our approach uses subdivision surfaces to represent images, which support arbitrary mesh structure and are able to conveniently model curvilinear features.

Besides regions, meshes or patches, other visually significant properties of the image can also be used represent images. For example, in [23], those high-curvature points on the image surface are used to represent the image in a compact way. This representation can preserve the essence of the original image. Edge-directed interpolation estimates significant edges in the image and the edges are used to guide the interpolation of image data to match arbitrarily oriented edges [24], [25]. The method can be extended to reconstruct all level-set contours in the image [26]. A powerful vector graphics called diffusion curves is proposed in [11] to emphasize meaningful features and the style of process that the artists create images. The diffusion curve is a geometric curve augmented with color and blur attributes. An image is represented by a set of these curves conforming to edges and reconstructed by diffusing colors from them. Diffusion curves are suitable and convenient for a user to create smooth-shaded images. However, diffusion curves have a few limitations. First, the image defined by diffusion curves is actually a solution to a Laplacian equation with Dirichlet boundary conditions. The process of solving the equation is not trivial. Some work has been developed to improve the robustness and speed of diffusion curves [27], [28]. Second, when diffusion curves detect edges in vectorizing raster images, they often fill smooth regions with overly dense edges and texture rich regions with insufficient number of edges. Jeschke et al [29] propose an automatic method to specify the position and color for diffusion curves. Third, compared to region, mesh or patch-based representations, a set of individual curves seem to lack strong structural connection. For example, they do not guarantee closed regions and thus may not be convenient for region-based editing. While our approach is similar to diffusion curves in supporting freehand drawing due to a curvilinear feature driven approach, it is based on subdivision surfaces, making it more structured and less dependent on edge detection.

III. Subdivision-Based Image Representation

We first introduce some notations that will be used in the paper for our descriptions. A point $v \in R^5$ is denoted by $v = (x, y, R, G, B)$ where the first two components (x, y) stand for the Cartesian coordinates in the $xy-$plane and the last three components (R, G, B) stand for RGB channels of color. Two projection operators LOC and $COLOR$ are defined. $LOC : (x, y, R, G, B) \rightarrow (x, y)$ projects each point in R^5 to its projection on the plane by extracting its first two components. $COLOR : (x, y, R, G, B) \rightarrow (R, G, B)$ takes the color value from a point in R^5 by extracting the last three components. We also use LOC and $COLOR$ for curves, surfaces and point sets in R^5 to extract their respect dimensional components unambiguously.

Now we present our subdivision based image representation. An image may contain curvilinear features. We propose to use a special B-spline curve in R^5 to represent the curvilinear features (see Section III-A) and a subdivision surface in R^5 to represent the image (see Section III-B). The first two dimensions of the B-spline curves and the subdivision surface describe the shape or location of features and the image on the 2D plane and the last three dimensions specify the color of features and the image. We devise new subdivision rules to model piecewise-smoothly varying images and their curvilinear features.

A. Curvilinear Feature Representation

Curvilinear feature is represented by a collection of curvilinear edges. Each curvilinear edge is a smooth curve. A closed curvilinear edge is defined by a uniform cubic periodic B-spline curve and thus it can be specified just by a polygon geometrically. Since the B-spline curve can be generated using subdivision of its control polygon, the use of B-spline representation makes it easy to be included in a subdivision surface. Particularly, if we refine such a B-spline curve representation by inserting new knots evenly into the uniform knot sequence, a new polygon consisting of new edge points and new vertex points is created [30]. Each edge point is the midpoint of the corresponding edge and each vertex point is the linear combination of the corresponding vertex and its two neighboring vertices with coefficients $6/8$, $1/8$ and $1/8$, respectively. This combination is also expressed by a mask: 1-6-1. By continuing this subdivision process, the initial polygon is gradually refined and the refined polygon converges to the B-spline curve.

If the curvilinear edge is a curve bounded by two endpoints, it is geometrically defined by a polyline denoted by $P_0P_1 \cdots P_n$, where P_i are the vertices of the polyline. The underlying representation of the curve is a special cubic B-spline curve with deBoor points $P_0, P_0, P_1, \ldots, P_n, P_n$ and knot sequence $\{0, 0, 0, 0, 1, 2, \ldots, n, n, n, n\}$. This B-spline curve is special because of the duplication of two endpoints P_0 and P_n. It is easy to check that the curve interpolates the two end points of the polyline. If we refine the knot sequence by inserting a knot at the midpoint of each nonzero knot interval, then the new knot sequence becomes $\{0, 0, 0, 0, 0.5, 1, 1.5, 2, \ldots, n-1, n-0.5, n, n, n, n\}$ and the deBoor points become $P_0, P_0, E_1, P_1', E_2, P_2', \ldots, P_{n-1}', E_n, P_n, P_n$ where

$$E_1 = \frac{3P_0 + P_1}{4}, \qquad E_n = \frac{P_{n-1} + 3P_n}{4},$$

$$E_i = \frac{P_{i-1} + P_i}{2} \quad \text{for } i = 2, \ldots, n-1,$$

$$P_1' = \frac{3P_0 + 11P_1 + 2P_2}{16}, \quad P_{n-1}' = \frac{2P_{n-2} + 11P_{n-1} + 3P_n}{16},$$

$$P_i' = \frac{P_{i-1} + 6P_i + P_{i+1}}{8} \quad \text{for } i = 2, \ldots, n-2.$$

The above formulae are derived using B-spline knot insertion algorithms [30]. They can also be explained using

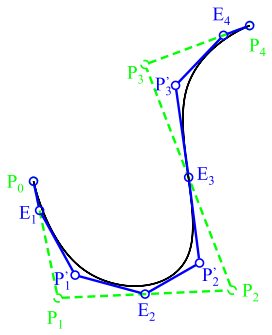


Fig. 1. A special cubic B-spline curve corresponds to an initial polyline (in green) and a refined polyline (in blue).

masks. That is, the masks for E_1, E_i, and E_n are 3-1, 1-1, and 1-3, respectively, and the masks for P'_1, P'_i, and P'_{n-1} are 3-11-2, 1-6-1, and 2-11-3, respectively. There are two exceptional cases that have special masks:

- $n = 1$: The curve is a straight line segment. The mask for E_1 is 1-1.
- $n = 2$: The masks for E_1 and E_2 are 3-1 and 1-3, and the mask for P'_1 is 3-10-3.

Thus the refinement of the special B-spline is equivalent to simply subdividing the initial polyline into the refined polyline $P_0E_1P'_1E_2\cdots E_nP_n$ (see Fig. 1 for an illustration). This subdivision has a pattern similar to the refinement of a uniform cubic B-spline curve. That is, each edge $P_{i-1}P_i$ corresponds to a new edge point E_i and each old vertex P_i corresponds to a new vertex point P'_i. However, the curve we define here is a non-uniform B-spline curve and it interpolates the two endpoints. Moreover, such specially designed curve formulation makes itself be easily integrated into the surface subdivision scheme introduced in the next subsection.

B. Subdivision Surfaces

Consider a triangular mesh M in R^5, denoted by (V, K) where $V = (v_1, v_2, \ldots, v_m), v_i \in R^5$ is a set of vertices and K is a simplicial complex specifying the connectivity of the vertices. Triangular meshes have flexible topology and thus are convenient for our purposes.

A subdivision surface is defined by repeatedly refining an initial control mesh. Loop subdivision is one subdivision scheme applied to triangular meshes, which generalizes C^2 quartic triangular B-splines to arbitrary topology [31]. The refinement step of Loop subdivision proceeds by splitting each triangle into four subtriangles. The vertices of the refined mesh are weighted averages of the vertices in the unrefined mesh. They are classified into two classes: edge points and vertex points corresponding to edges and vertices of the unrefined mesh, respectively. As the refinement goes to infinity, Loop subdivision results in a tangent plane continuous surface.

We aim to use a subdivision surface to define an image or an object in an image. The object could have an arbitrary boundary, holes, or many features which occur as sharp or semi-sharp curvilinear edges geometrically. An image can also be viewed as an object with a regular boundary and possibly many features. Since the image surface across the sharp or semi-sharp features is not smooth, we need to modify the Loop

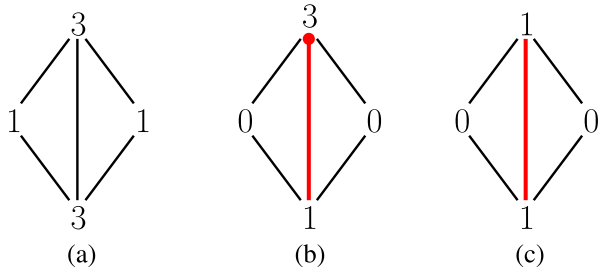


Fig. 2. Edge subdivision rules. Red lines denote crease edges and the disk denotes a corner vertex. (a) Smooth edge. (b) Crease edge with one corner endpoint. (c) Other crease edge.

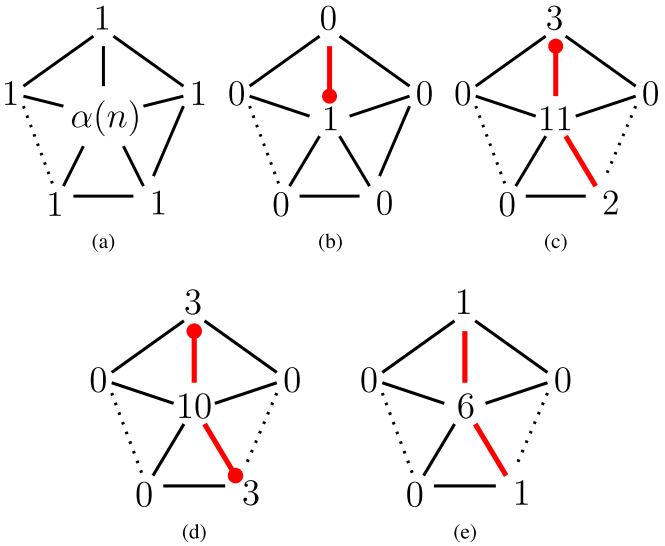


Fig. 3. Vertex subdivision rules. Red lines denote crease edges and disks denote corner vertices. (a) Smooth vertex. (b) Corner vertex. (c) Ccrease vertex with one corner neighbor. (d) Crease vertex with two corner neighbors. (e) Other crease vertex.

subdivision scheme in order to properly represent the image that contains curvilinear features.

Given a triangular mesh, some consecutive edges of the mesh form a polyline defining a B-spline curve for curvilinear feature. Such a polyline is called a *feature polyline* and the related edges and vertices are tagged as *crease*. In particular, the boundary of the mesh is viewed as feature polylines. Feature polylines may contain endpoints. The endpoints are tagged as *corner*. In addition, a vertex lying on two feature polylines is also tagged as corner. All the other vertices and edges are considered to be smooth vertices and edges. In this way, the pair (V, K) becomes a tagged simplicial complex. The Loop subdivision masks are modified so that the feature polylines define the demanded curvilinear features and the tangent plane continuity across crease edges is relaxed. In the subdivision process, new edges corresponding to an old crease edge are tagged as crease, a vertex corresponding to an old crease vertex is tagged as crease, and similarly a vertex corresponding to an old corner vertex is tagged as corner.

While subdivision at smooth edges and vertices can be performed using Loop subdivision algorithm, subdivision rules at crease edges and crease/corner vertices must be chosen carefully in order to match the rules for creating the curvilinear features given in Section III-A. Fig. 2 shows our edge

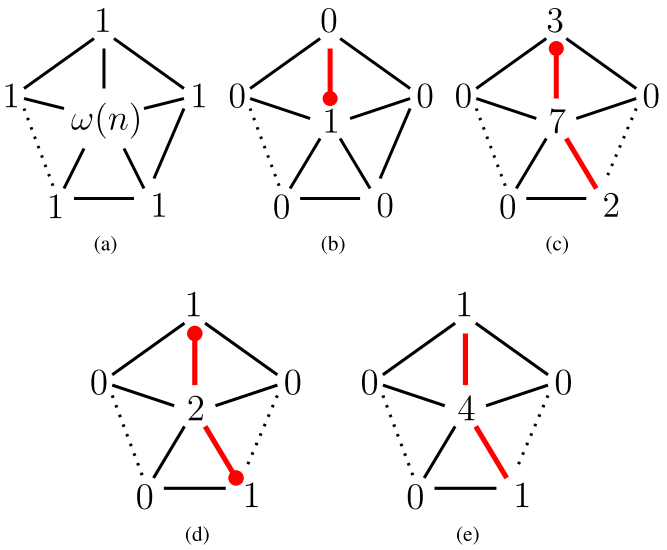


Fig. 4. Limit position masks for vertices. Red lines denote crease edges and disks denote corner vertices. (a) Smooth vertex. (b) Corner vertex. (c) Crease vertex with one corner neighbor. (d) Crease vertex with two corner neighbors. (e) Other crease vertex.

subdivision rules and Fig. 3 shows our vertex subdivision rules where

$$\alpha(n) = \frac{n(1-a(n))}{a(n)}$$

with

$$a(n) = \frac{5}{8} - \frac{(3+2\cos(2\pi/n))^2}{64}$$

and n is the valence of the vertex. The zeros in the crease subdivision masks completely decouple the behavior of the surface on one side of the crease from the behavior on the other side. Note that different subdivision rules have been presented in [32] and [33] in order to create piecewise smooth subdivision surfaces. Our subdivision rules are driven by the curvilinear features. Feature polylines without corner vertices generate uniform cubic B-spline curves and feature polylines with some corner vertices generate cubic B-spline curves with interpolating endpoints.

Computing limit points: We look at a vertex v^k of the mesh after k rounds of refinement and its 1-ring neighbors $v_1^k, \ldots, v_n^k$ where n is the valence of v^k. Let v^{k+1} be the vertex of the mesh after $k+1$ rounds of refinement, associated with v^k, and $v_1^{k+1}, \ldots, v_n^{k+1}$ be the 1-ring neighbors of v^{k+1}. Then we have

$$(v^{k+1}, v_1^{k+1}, \ldots, v_n^{k+1})^T = S_{n+1}(v^k, v_1^k, \ldots, v_n^k)^T \quad (1)$$

where S_{n+1} is a $(n+1) \times (n+1)$ matrix called the *local refinement matrix*. The local refinement matrix is important in analyzing point positions and smoothness of the limit surface. We perform eigenanalysis of matrix S_{n+1} as in [33] and [34] and derive the masks for computing the limit points. The masks depend on whether the vertex is a corner, crease, or smooth vertex and its neighboring vertices, and they are given in Fig. 4 where $\omega(n) = \frac{3n}{8a(n)}$. Note that for a crease vertex with one or two corner neighbors, after one round of refinement it will become a normal crease vertex, which is not connected to a corner vertex. Therefore the eigenanalysis should be performed after one round of refinement.

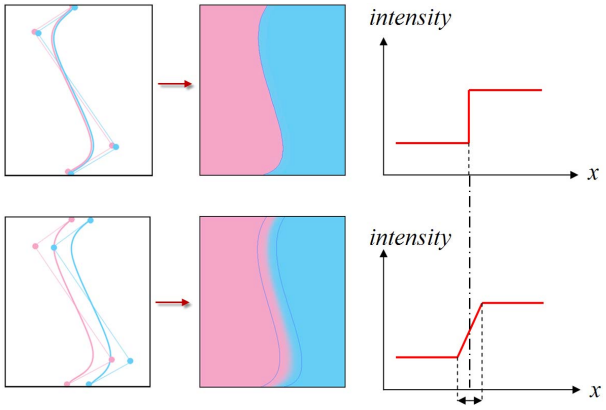


Fig. 5. Color discontinuity modeling. Top: a sharp feature is created with sharpness = 0.9; Bottom: a semi-sharp feature is created with sharpness = 0.1.

C. Color Discontinuity Modeling

Note that the proposed subdivision surface is piecewise smooth, divided by curvilinear features across which the surface is only C^0 continuous. C^0 continuity is necessary for the geometry part (i.e., the first two dimensions). It ensures that no gap occurs geometrically in the representation. Nevertheless, the property of C^0 continuity may be too strong for the color part (i.e., the last three dimensions) since it is quite often for an image to have color discontinuity across feature edges.

The introduction of crease edges into subdivision well solves the non-smoothness issue in geometric modeling, but it does not effectively solve the discontinuity problem for images. This motivates us to introduce *feature polyline pair* to model color discontinuity. That is, we design our triangular mesh such that feature polylines occur in pairs and the feature polylines in each pair are close geometrically but have different color attributes. Moreover, we further introduce a parameter "sharpness", which takes values from 0 to 1, to describe how much the color is discontinuous across an edge. The sharpness value of 1 means that the edge is sharp. The value falling in (0, 1) implies a semi-sharp edge. The value of sharpness is used to determine the geometric distance between two feature polylines in a pair: distance $= \rho(1 - \text{sharpness})$ where $\rho > 0$ is a constant. We set $\rho = 5$ in our experiment. When sharpness is close to 1, the distance is nearly zero. Then the color will have a sudden change across the curvilinear edge, which gives color discontinuity effect. Fig. 5 illustrates this idea.

Fig. 6 shows two images created using the proposed representation from triangular meshes. Tagged edges are marked by red and green colors for sharp and semi-sharp features.

IV. Image Reconstruction

This section presents our algorithm for automatically constructing subdivision-based representation for a given image. The algorithm consists of four phases: curvilinear curve construction, initial mesh construction, color optimization

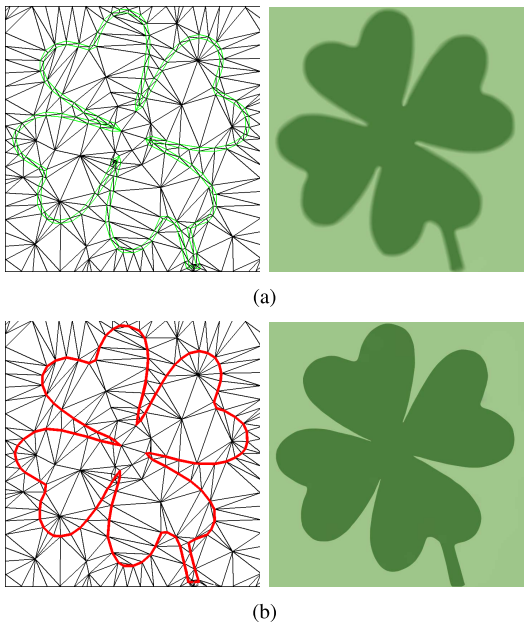


Fig. 6. Images with curvilinear edges and their meshes. (a) With semi-sharp edges. (b) With sharp edges.

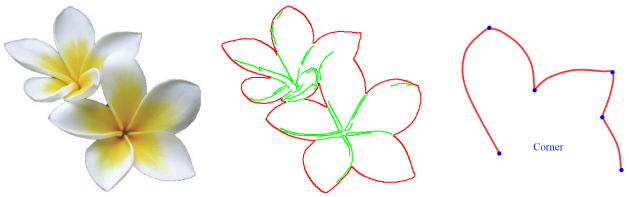


Fig. 7. Feature extraction. (a) Input objects; (b) Cleaned sharp (red) and semi-sharp (green) edges; (c) Some corners.

and mesh refinement. First, image analysis is performed to extract distinct curvilinear features and the feature curves are then approximated by cubic B-spline curves. Second, an initial triangular mesh with tagged information is constructed. Third, the color value for each vertex of the mesh is computed through an optimization procedure. Fourth, approximation error is checked. If the error is greater than a prescribed threshold, the mesh is refined and the algorithm goes back to phase 3. Phase 3 and phase 4 are repeated until a satisfactory approximation is reached. In this process, the mesh size and reconstruction quality should be traded off. The technical details of these phases are explained below.

A. Curvilinear Curve Construction

Curvilinear curve construction involves feature extraction and B-spline curve fitting. We identify the curvilinear features using the advanced contour detection technique proposed in [35]. For each extracted edge, we estimate the degree of blur for each pixel on the edge using the method of [36]. The blur scale is normalized to [0, 1]. After this, Otsu's thresholding method is adapted to minimize the within-group variances on the blur scales and the blue scales are quantized. Then each extracted curvilinear edge is associated with a value of sharpness that is (1 - blur-scale). Next, corners are to be identified. The algorithm in [37] is used to locate corners in each curvilinear edge. Also the endpoints of a curvilinear edge or the point where two curvilinear edges intersect are treated as corners. Fig. 7 gives an example of feature extraction.

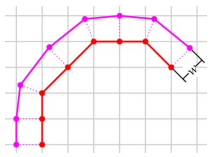


Fig. 8. The 2nd feature polyline (in pink) is created from the 1st one (in red).

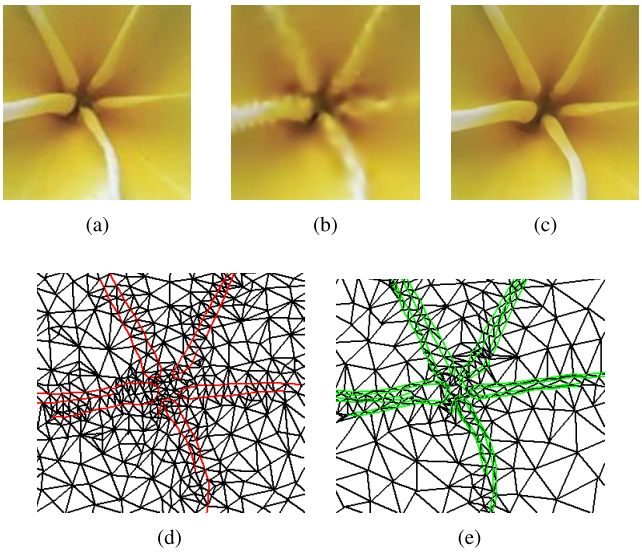


Fig. 9. Effects of image reconstruction using only single polylines marked in red in (d) and using feature polyline pairs marked in green in (e). (a) Input image. (b) Using single polylines. (c) Using polyline pairs. (d) Mesh for (b). (e) Mesh for (c).

For each curvilinear edge, we perform B-spline fitting. A cubic periodic B-spline is used for a closed curve and a special cubic B-spline is used for an open curve. If the length of the curvilinear curve is len, we heuristically set the number of deBoor points to be $\max\{4, len/8\}$. Once the type of the B-spline curve and the number of deBoor points are specified, what remain to be determined are only the deBoor points, which can be found by the conventional least squares fitting. If the approximation error is larger than a user-specified threshold, the number of deBoor points is increased and the least squares procedure is repeated.

As explained in Section III-C, we need to use feature polyline pairs to model (semi-)sharp edges. The above B-spline fitting has already produced one polyline for each feature curve. Therefore we can create another one by offsetting the first one along the "normal" direction by a certain distance (see Fig. 8). Specifically, for each deBoor point, we define the normal at this point to be the average of the normals of its two incident edges. Then each deBoor point is moved along its normal by a distance. The distance depends on the sharpness and is computed by the formula given in Section III-C. It is worth pointing out that using a pair of feature polylines is important for generating high quality reconstruction. Fig. 9 shows an example, where (a) is the input image, (b) is the reconstruction result using single curvilinear features as marked in red in (d) and (c) is the result using polyline pairs as marked in green in (e).

B. Initial Mesh Construction

Once the polylines are generated, a constrained Delaunay triangulation [38] is performed on them to create a triangular mesh in the image domain. The triangulation quality can be controlled by the minimal angles of triangles. The larger the minimal angles are set to be, the more regular the resulting triangles are, which results in more triangles. To balance the number and quality of the resulting triangles, we set the minimal angles to be 20 degrees in this paper, which works well in our experiments. In order to obtain a satisfied mesh, additional points are allowed to be inserted, but not on the polylines. The polylines have a tag showing the sharpness. As a result, an initial control mesh M is constructed augmented with tag information.

C. Color Optimization

Suppose the input image is I and the subdivision surface in R^5 is S. Now $(LOC(V), K)$, the projection of M onto the image plane, has been specified. What remains is to determine $COLOR(V)$, the color values for vertices. One may let $COLOR(V)$ be $I(LOC(V))$, but this simple color scheme does not necessarily offer a good solution. In the following we determine the vertex colors by solving the following minimization problem:

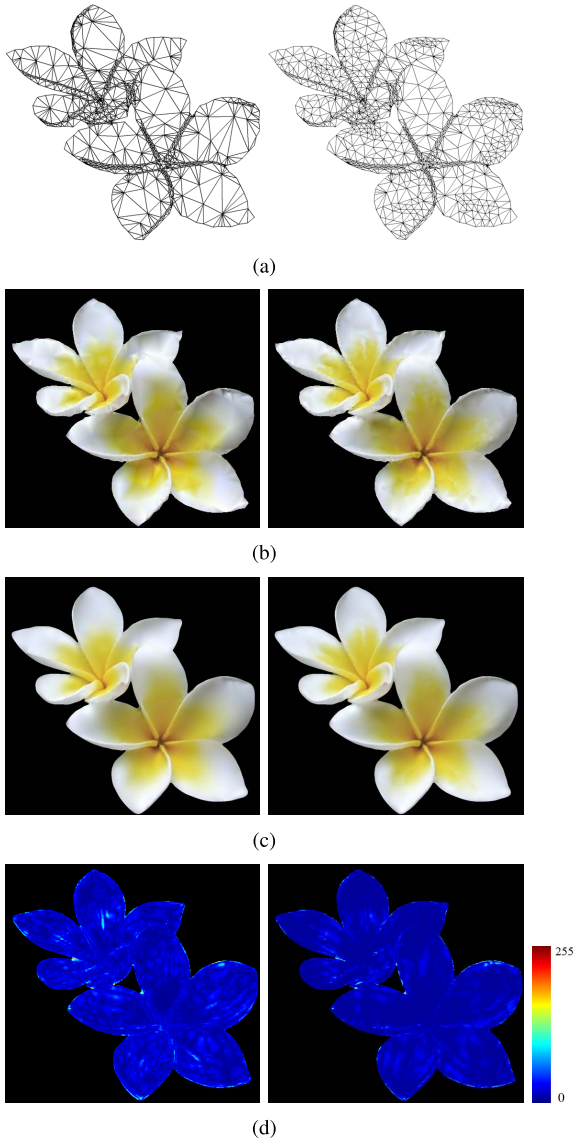
$$\min_{COLOR(V)} \sum_{(x,y)} \|I(x,y) - COLOR(H(x,y))\|^2 \qquad (2)$$

where $H(x,y) \in R^5$ is a point on S with $LOC(H) = (x,y)$.

In practice, however, computing $H(x,y)$ for given (x,y) is very complicated. Here we take an approach similar to the one used in [32]. We subdivide r times (typically $r = 2$) the original mesh M to generate a refined mesh M^r using the rules proposed in Section III-B and then push all the vertices of M^r to their limit positions using the position masks. In this way, we obtain a piecewise linear approximation $\tilde{S}$ to S. Since each vertex of M^r is a linear combination of the vertices of M and the limit position is also a linear combination of the vertices of M^r, each of the vertices of $\tilde{S}$ can be written as a linear combination of the vertices in V. Furthermore, since $\tilde{S}$ is piecewise linear, any point on $\tilde{S}$ can be linearly represented by at most three vertices of $\tilde{S}$. In particular, if a 2D point $q = (x,y)$ is within $LOC(\tilde{S})$, it must be contained in a triangle formed by vertices in $LOC(\tilde{S})$. Assume the three vertices are $LOC(\tilde{v_i})$, $LOC(\tilde{v_j})$, and $LOC(\tilde{v_k})$. Then

$$q = u_i LOC(\tilde{v_i}) + u_j LOC(\tilde{v_j}) + u_k LOC(\tilde{v_k})$$

where (u_i, u_j, u_k) the barycentric coordinates of q with respect to triangle $LOC(\tilde{v_i})LOC(\tilde{v_j})LOC(\tilde{v_k})$. If H is a point on $\tilde{S}$ satisfying $LOC(H) = q$, the linear relationship also holds for H in R^5, meaning $H = u_i\tilde{v_i} + u_j\tilde{v_j} + u_k\tilde{v_k}$. Thus each point on $\tilde{S}$—not just the vertices—can be written as a linear combination of the vertices in V. That is, $H = C_q V$ where C_q is a row vector whose entries is a combination of the effects of $r-$fold subdivision followed by applications of position masks and barycentric coordinates.



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

Fig. 10. An example of mesh refinement for the flower image in Fig. 7(a). Left column shows an initial mesh, the image defined by the initial mesh with the simple color scheme, the reconstruction result by the initial mesh with color optimization and the color map of the reconstruction error. Right column shows the counterparts when the initial mesh is refined. (a) Meshes before and after refinement. (b) The results by the simple color scheme. (c) The results by color optimization. (d) Reconstruction errors.

The minimization problem (2) now becomes

$$\min_{COLOR(V)} \sum_{(x,y)\in LOC(\tilde{S})} \|I(x,y) - C_{(x,y)} COLOR(V)\|^2 \qquad (3)$$

It is a typical least squares problem, which can be efficiently solved by linear solvers. This also explains why we choose to only optimize the color values of the vertices. Though optimizing both RGB values and the positions of the vertices simultaneously may give better reconstruction results, the coupling makes the optimization procedure highly nonlinear and difficult to solve.

To sample (x,y) within $LOC(\tilde{S})$, we use scan conversion to each triangle of $LOC(\tilde{S})$. The coherence of location enables us to compute the barycentric coordinates of a point from the barycentric coordinates of the previous point or the point on

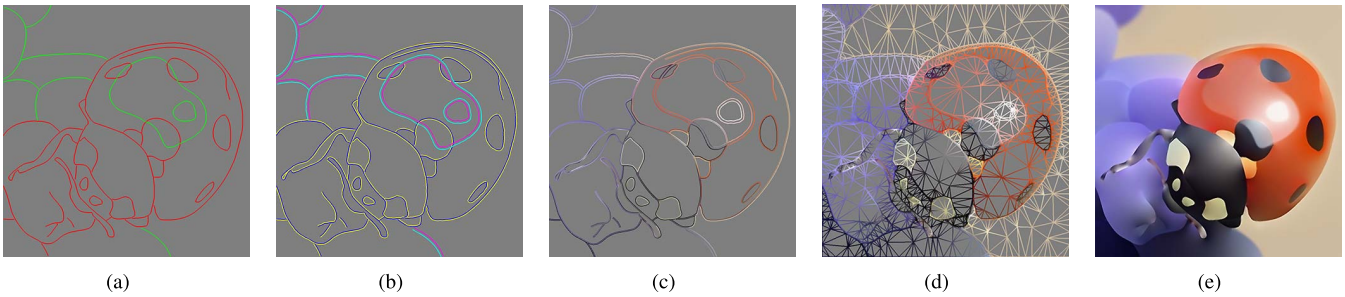


Fig. 11. Image creation from sketches: (a). The user sketches curvilinear edges (sharp and semi-sharp features are marked in red and green); (b) A pair of polylines is generated for each curvilinear curve (the pairs of sharp features are shown by yellow and blue colors, and the pairs of semi-sharp features are shown by pink and cyan colors); (c) The user paints color to the vertices of the curvilinear curves; (d) A tagged control mesh with color attributes is created; (e) An image is created using the subdivision surface.

the previous scan line by simple additions or subtractions. This speeds up the computation.

D. Mesh Refinement

After color optimization, we obtain a subdivision surface. The color fitting error is then evaluated and those pixels corresponding to a fitting error greater than a user-specified threshold are marked. To reduce the fitting error, we proceed to refine the mesh. We scan the marked pixels and iteratively select the pixels whose distances to the existing mesh vertices and already selected pixels are greater than a given threshold. Instead of refining the existing mesh by directly adding the new vertices corresponding to the selected pixels, we apply the constrained Delaunay triangulation again to the vertices of the existing mesh, the new vertices, and the feature polylines. Fig. 10 shows one example of such refinement. It can be seen that the refinement improves the results. The reconstruction errors are plotted using color in Fig. 10(d) (see the inset on the right for the color map). This color map is also used for plotting the errors in the rest of the paper.

V. Curvilinear Feature Driven Image Creation

The proposed subdivision-based image representation offers a simple way to create an image in a freehand style as in diffusion curves [11]. The user just sketches some curves as the curvilinear curves and specifies their color and sharpness attributes. Our algorithm can automatically create a tagged triangular mesh which defines the target image. The creation process basically involves the following steps:

- The user sketches curvilinear curves and assigns sharpness to them.
- The sketched curves are represented by cubic B-spline curves. As explained in Section IV, our algorithm automatically generates an adjacent curve for each sketched curvilinear curve. The user can adjust the distance between the curves of each pair to control the blur effect.
- The user specifies colors for each curvilinear curve.
- A tagged triangular mesh is automatically generated by applying constrained Delaunay triangulation to the control polygons of the generated cubic B-spline curves as explained in Section IV-B. The colors for the vertices of the mesh are then estimated using the method introduced in [28].
- The target image is created by the subdivision surface defined by the constructed mesh and color.

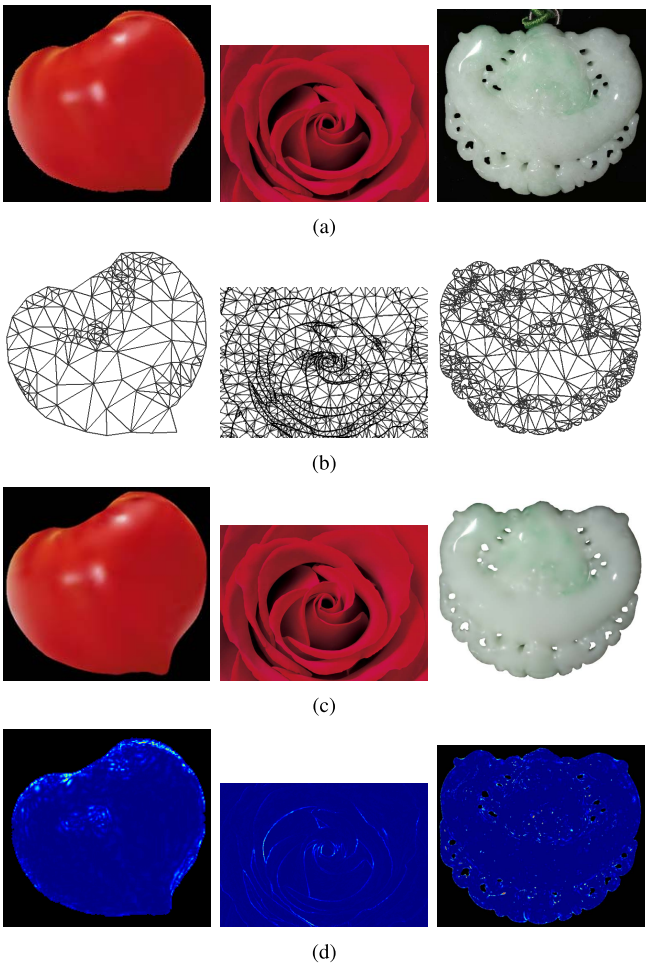


Fig. 12. Vectorization of 3 different images using our method. Left: image with smoothly varying color; Middle: image with curvilinear features; Right: image with complicated topology. (a) Input images. (b) Reconstructed meshes. (c) Reconstructed images. (d) Plotting of the reconstruction errors.

Fig. 11 illustrates this process. Our input sketches in Fig. 11(a) are very similar to those given in [11]. The generated polyline pairs allow us to conveniently specify different colors

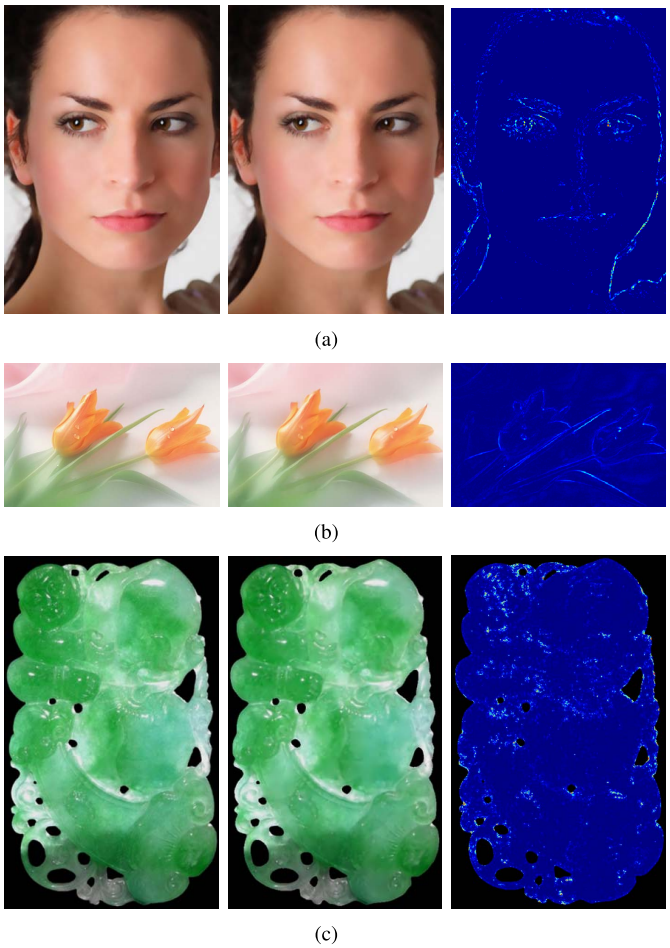


Fig. 13. More vectorization results: original images (left), reconstructed results (middle), and plotting of the reconstruction errors (right). (a) Face; (b) flowers; (c) green jade.

at two sides of the feature curves (see Fig. 11(c)), just as the method of diffusion curves. Moreover, our method further allows adjusting the distance between two feature curves in each pair. A tagged mesh associated with color attributes is automatically generated to define a subdivision surface and thus an image, which appears similar to the one produced in [11].

VI. Experiments

This section provides experimental results to validate our proposed method. The method is implemented using C++. The experiments are run on a PC with Intel Xeon 2.0 GHz CPU and 2GB RAM. The reconstruction quality is not only evaluated visually but also quantified by quality measurements.

A. Reconstruction Results

Our reconstruction algorithm handles a whole image and objects on an image in the same way, except for a preprocess step for objects, which is to cut out objects from images using some segmentation tool. For example, Fig. 12, Fig. 13(c), Fig. 17, and Fig. 18 are all objects.

Figs. 12–18 show images with various characteristics. Fig. 12(right) and Fig. 13(c) have complicated topology,

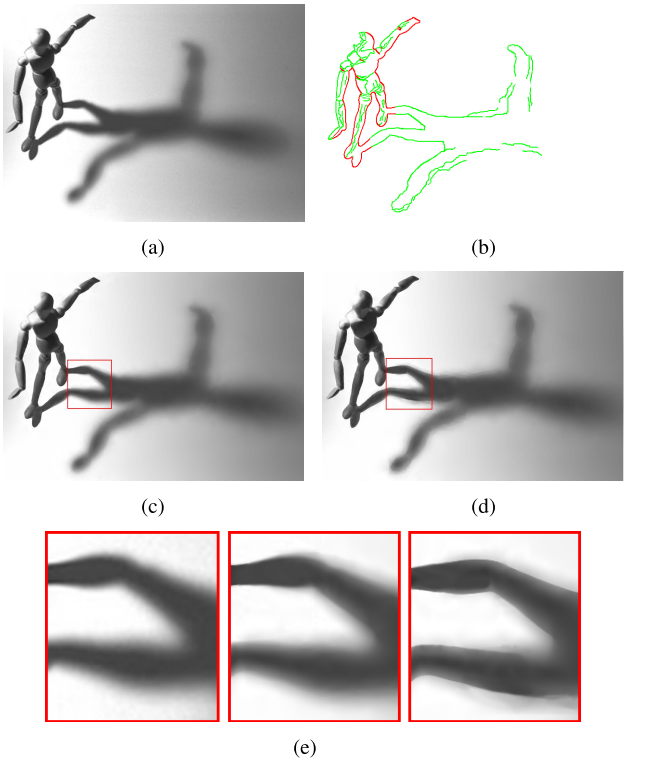


Fig. 14. Vectorization with/without use of sharpness. (a) Input image. (b) Curvilinear edges. (c) With sharpness. (d) Without sharpness. (e) Close-ups of (a), (c) and (d).

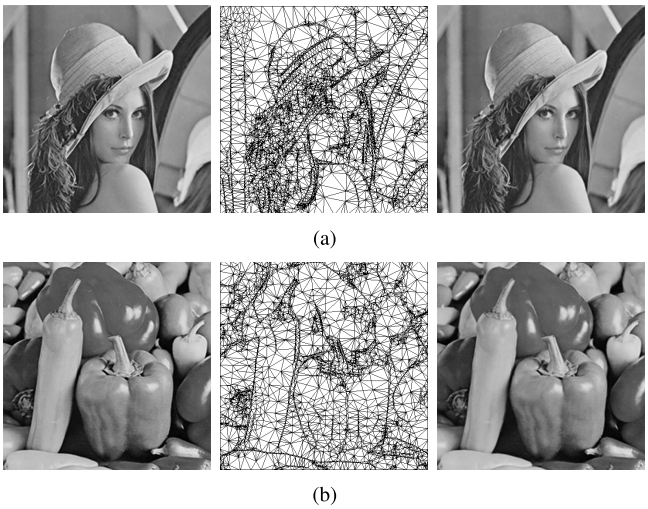
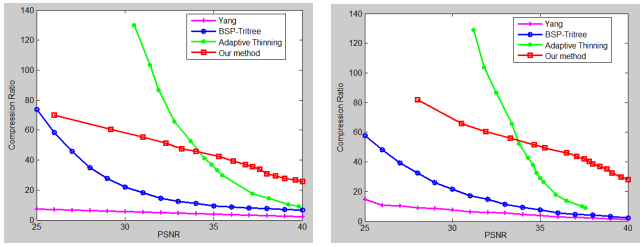


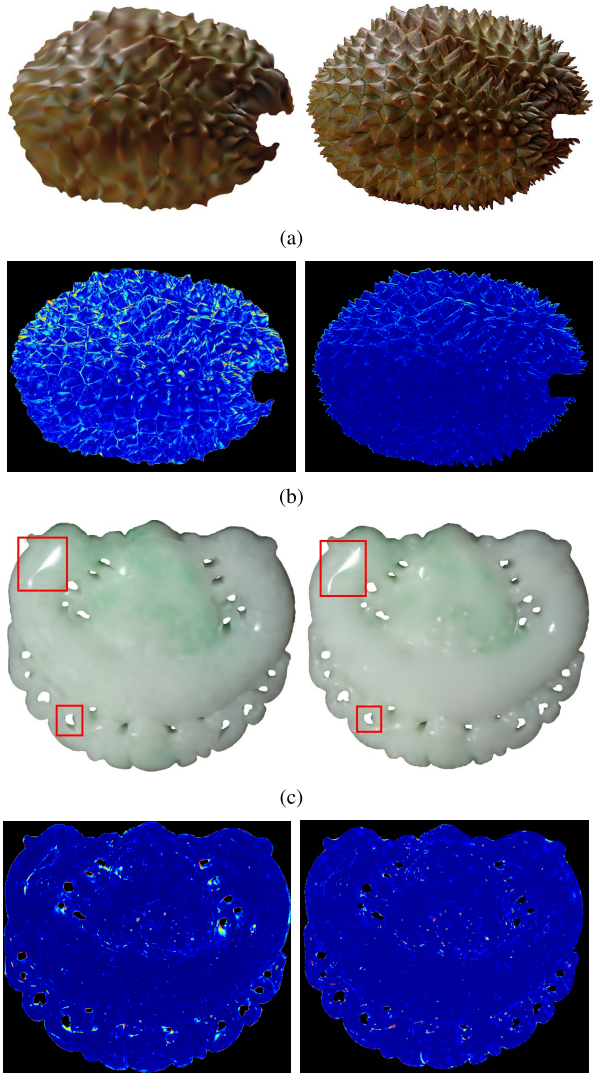
Fig. 15. Left: input images; Middle: reconstructed meshes; Right: reconstructed images. (a) Lena. (b) Pepper.

where the objects contain many holes. Fig. 12(middle) and Fig. 14 have complicated curvilinear features. Fig. 12(left) has smooth color transitions. Refer to Table I for the statistics of our experiment examples. In particular, columns 2–8 show the running time, the number of iterations, the number of resulting triangles, the size in Kbytes of our proposed representation without compression, the number of pixels of the input images, the size in Kbytes of the input image in PNG format, and the mean errors of the reconstruction, respectively. It can be seen that for all these examples,



(a) (b)

Fig. 16. Compression ratios at various PSNRs. (a) Lena. (b) Pepper.



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

Fig. 17. Comparison with [10]. Left: result of [10]; Right: ours. (a) Durian; (b) reconstruction errors for durian; (c) jade; (d) reconstruction errors for jade.

our vectorization results are very close to the original images, the reconstructed mesh is compact and the algorithm is efficient. It takes less than 2 minutes to reconstruct each of these examples except for the durian model that costs 5 minutes.

Our method associates curvilinear features with sharpness. Sharpness enables our method to effectively handle subtle color changes, which is important for sincere reconstruction. Fig. 14 shows how the puppet's shadow is reconstructed faithfully. The curvilinear edges of the image is detected as marked in Fig. 14(b) where red indicates a large sharpness value and green indicates a small sharpness value. Fig. 14(c) and (d) are the results of vectorization with and without using sharpness, respectively. The close-ups in Fig. 14(e) show the difference.

B. Quality Assessment

We evaluate our reconstruction quality using two popular measurements: the Peak Signal-to-Noise Ratio (PSNR) and the mean Structural SIMilarity index (MSSIM). Particularly, MSSIM is an excellent measurement relative to human subjective assessment [39]. Based on a curve reflecting the relationship between MSSIM/PSNR and Mean Opinion Scores (MOS) in [40], a test image with PNSR greater than 35 or MSSIM greater than 0.960 obtains an MOS higher than 60. The MOS is divided into five equal regions within [0, 100] and these five regions are marked with adjectives "Bad", "Poor", "Fair", "Good" and "Excellent". From Table I, we can see that all our testing results have "good" or "excellent" reconstructions.

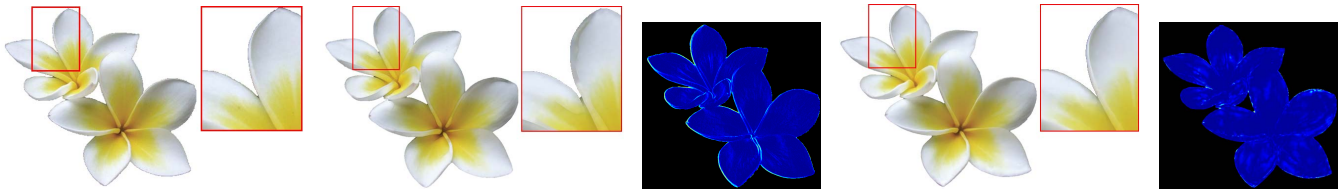
In [4], a compression ratio is introduced as the ratio of the number of the original image pixels to the number of vertices of the reconstructed mesh. Table I also reports the compression ratios of our experiment examples.

We use the Lena and pepper images (see Fig. 15) to evaluate the performance of our method and other mesh generation methods [3], [4], [41]. The performance curves for the compression ratios at various PSNRs are shown in Fig. 16, from which we can see that our method has a better compression ratio than Yang *et al.*'s [3] and BSP's [4] methods. Our method also outperforms the adaptive thinning scheme [41] for high image reconstruction quality.

C. Comparison With Previous Work

We compare our vectorization with two representative methods in literature. First let us examine how to estimate the bytes required to store the reconstructed vector graphics representations. In our representation, each vertex contains x, y and RGB values (in floating points) and thus requires $5 \times 4\text{bytes} = 20$ bytes, each face needs to store three indices of vertices, requiring $3 \times 2\text{bytes} = 6$ bytes. In addition, we need four integers (each 2bytes) to record the number of vertices, the number of triangles containing 3 tagged edges, the number of triangles containing 2 tagged edges and the number of triangles containing 1 tagged edge. In total we need $(\#vertices \times 20 + \#triangles \times 6 + 8)$bytes. The gradient mesh representation proposed in [10] needs to store corners of patches besides some boundary information. Each corner contains the position, RGB values and their partial derivatives, summing up to 15 scalar coefficients. Thus the gradient mesh representation requires at least $(\#corners \times 60)$bytes. In the patch-based representation proposed in [12], each patch has 66 scalar coefficients. Thus the representation of [12] requires $(\#patches \times 264)$bytes plus additional information for the connectivity of patches.

The objects in Fig. 17 have fine details. As reported in [10], the method of [10] has difficulty in recovering the durian



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

Fig. 18. Comparison with [12]. (a) Original image. (b) Result of [12]. (c) Plotting of errors of [12]. (d) Our result. (e) Plotting of errors of our method.

TABLE I

STATISTICS FOR THE EXAMPLES OF IMAGE VECTORIZATION USING OUR METHOD

Examples	time (seconds)	iterations	triangles	size of new representation	number of pixels	size of PNG files	mean errors	compression ratio	PSNR	MSSIM
Fig. 12(left)	10	2	349	5.8KB	225×214	42KB	1.28	250.8	41.21	0.9806
Fig. 12(middle)	58	2	6750	107.3KB	531×354	194KB	0.94	54.2	43.86	0.9972
Fig. 12(right)	61	2	2496	43.4KB	544×475	267KB	2.41	175.3	35.71	0.9638
Fig. 13(a)	71	2	6684	114.2KB	435×620	249KB	1.07	73.4	43.86	0.9892
Fig. 13(b)	62	2	5892	98.8KB	457×306	142KB	0.96	42.5	43.71	0.9905
Fig. 13(c)	113	3	11526	200.5KB	471×776	379KB	2.13	53.7	36.77	0.9727
Fig. 14(c)	61	2	2509	39.3KB	465×320	82 KB	1.38	118.2	40.53	0.9917
Fig. 14(d)	61	2	2509	39.3KB	465×320	82 KB	1.40	118.2	40.39	0.9898
Fig. 15(a)	109	3	10270	132.8KB	512×512	192KB	2.10	42.3	35.30	0.9603
Fig. 15(b)	75	2	9568	116.8KB	512×512	188KB	1.24	50.6	36.01	0.9656
Fig. 17(a)	317	1	30575	489.5KB	681×497	433KB	2.09	21.3	36.97	0.9770
Fig. 18	63	2	6257	103.4KB	530×500	276KB	0.85	77.6	44.74	0.9901

while our method can reconstruct the durian in quality with a reasonable mesh size. For the jade, our method produces a representation with 1474 vertices, 2496 triangles, and mean error of 2.41, while the method of [10] produces a gradient mesh with about 63 vertical lines and 56 horizontal lines and mean error of 2.76. The images in the figure show that our reconstruction result contains more details. The above estimation also indicates that the storage and mean error of our reconstruction results are 43.4KB and 2.41 while [10]'s are 206.7K and 2.76.

Fig. 18 is the comparison of our method with the method of [12]. Our method results in 3417 vertices, 6257 faces, and mean error of 0.85 while the method of [12] produces 380 patches with each patch containing 66 parameters and mean error of 0.98. The storage of our representation is thus at most 103.4KB and [12]'s is 98KB which does not include the connectivity information of patches.

D. Applications

With our vector graphics representation, standard graphics operations such as magnification, rotation and freeform deformation can be performed faithfully. Fig. 19 shows an example of image magnification based on the reconstruction image (middle) or the bicubic interpolation (right). It can be seen that with the proposed subdivision surface representation, the image features can be preserved very well for the process of magnification.

Vector-based editing. The image can be modified by changing deBoor points: 2D positions on the image plane or RGB color values. A tool has been implemented to allow the user to select individual points or edges, to sketch a closed curve to select a region, and to change them. Fig. 20(left) is a reconstructed image. The user selects some control points

Fig. 19. Image magnification ($\times 8$). Left: input image; Middle: using the reconstruction image; Right: using bicubic interpolation.

Fig. 20. Editing of image features.

that define the curvilinear features in the middle of the cup and move them to the new locations to warp the shape as shown in Fig. 20(middle), where the cup color is also retouched by changing the color values of the deBoor points. In addition, our vector graphics representation provides a convenient way to adjust sharpness of curvilinear features. It is achieved by changing the sharpness values of edges. Fig. 20(right) shows the result of changing part of a curvilinear edge from blur to sharp, which is highlighted by an ellipse.

VII. CONCLUSIONS

We have introduced subdivision surfaces in R^5 as a new vector graphics representation for raster images. The new subdivision surface scheme is an extension of Loop subdivision by incorporating new subdivision rules for modeling

curvilinear curves. The curvilinear curves are modeled by specially designed cubic B-spline curves. A curvilinear curve is also associated with a parameter controlling the sharpness. The proposed representation has flexibility in defining complicated contents of images, is capable of modeling various sharp or semi-sharp edges in a compact way, and provides a natural way for the user to create images in a freehand style. An automatic image reconstruction algorithm is presented to find a subdivision surface that best fits the input image. Experimental results have demonstrated the effectiveness of the proposed new image representation and its reconstruction algorithm.

REFERENCES

[1] W. A. Barrett and A. S. Cheney, "Object-based image editing," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 777–784, Jul. 2002.

[2] M. Kashimura, Y. Sato, and S. Ozawa, "Image description for coding using triangular patch structure," in *Proc. Int. Conf. Commun. Syst.*, Nov. 1992, pp. 330–334.

[3] Y. Yang, M. N. Wernick, and J. G. Brankov, "A fast approach for accurate content-adaptive mesh generation," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 886–881, Aug. 2003.

[4] M. Sarkis and K. Diepold, "Content adaptive mesh representation of images using binary space partionts," *IEEE Trans. Image Process.*, vol. 18, no. 5, pp. 1069–1079, May 2009.

[5] M. D. Adams, "A flexible content-adaptive mesh generation strategy for image representation," *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2414–2427, Sep. 2011.

[6] M. Froumentin, F. Labrosse, and P. Willis, "A vector-based representation for image warping," *Comput. Graph. Forum*, vol. 19, no. 3, pp. 419–425, Sep. 2000.

[7] X. Yu, B. S. Morse, and T. W. Sederberg, "Image reconstruction using data-dependent triangulation," *IEEE Comput. Graph. Appl.*, vol. 21, no. 3, pp. 62–68, May/Jun. 2001.

[8] B. Price and W. Barrett, "Object-based vectorization for interactive image editing," in *Proc. Pacific Graph.*, Sep. 2006, pp. 661–670.

[9] J. Sun, L. Liang, F. Wen, and H. Shum, "Image vectorization using optimized gradient meshes," in *Proc. SIGGRAPH*, 2007.

[10] Y.-K. Lai, S.-M. Hu, and R. R. Martin, "Automatic and topology-preserving gradient mesh generation for image vectorization," *ACM Trans. Graph.*, vol. 28, no. 3, p. 85, Aug. 2009.

[11] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin, "Diffusion curves: A vector representation for smooth-shaded images," *ACM Trans. Graph.*, vol. 27, no. 3, p. 92, Aug. 2008.

[12] T. Xia, B. Liao, and Y. Yu, "Patch-based image vectorization with automatic curvilinear feature alignment," *ACM Trans. Graph.*, vol. 28, no. 5, p. 115, Dec. 2009.

[13] S. Swaminarayan and L. Prasad, "Rapid automated polygonal image decomposition," in *Proc. 35th Appl. Imag. Pattern Recognit. Workshop*, Oct. 2006, p. 28.

[14] G. Lecot and B. Lévy, "ARDECO: Automatic region detection and conversion," in *Proc. Eurograph. Symp. Rendering*, 2006.

[15] S. A. Coleman, B. W. Scotney, and M. G. Herron, "Image feature detection on content-based meshes," in *Proc. Int. Conf. Image Process.*, 2002, vol. 1, pp. I-844–I-847.

[16] F. Davoine, M. Antonini, J.-M. Chassery, and M. Barlaud, "Fractal image compression based on Delaunay triangulation and vector quantization," *IEEE Trans. Image Process.*, vol. 5, no. 2, pp. 338–346, Feb. 1996.

[17] Y. Wang, O. Lee, and A. Vetro, "Use of two-dimensional deformable mesh structures for video coding. II. The analysis problem and a region-based coder employing an active mesh representation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 6, pp. 647–659, Dec. 1996.

[18] D. Hale, "Atomic images-a method for meshing digital images," in *Proc. 10th Int. Meshing Roundtable*, 2001, pp. 185–196.

[19] J. G. Brankov, Y. Yang, and M. N. Wernick, "Tomography image reconstruction using content-adaptive mesh modeling," in *Proc. Int. Conf. Image Process.*, 2001, vol. 1, pp. 690–693.

[20] Y. Wang and O. Lee, "Active mesh-a feature seeking and tracking image sequence representation scheme," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 610–624, Sep. 1994.

[21] Y. Altunbasak and A. M. Tekalp, "Closed-form connectivity-preserving solutions for motion compensation using 2-D meshes," *IEEE Trans. Image Process.*, vol. 6, no. 9, pp. 1255–1269, Sep. 1997.

[22] D. Su and P. Willis, "Image interpolation by pixel-level data-dependent triangulation," *Comput. Graph. Forum*, vol. 23, no. 2, pp. 189–201, Jul. 2004.

[23] S.-J. Wang, L.-C. Kuo, H.-W. Jong, and Z.-H. Wu, "Representing images using points on image surfaces," *IEEE Trans. Image Process.*, vol. 14, no. 8, pp. 1043–1056, Aug. 2005.

[24] J. Allebach and P. W. Wong, "Edge-directed interpolation," in *Proc. ICIP*, Sep. 1996, pp. 707–710.

[25] X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE Trans. Image Process.*, vol. 10, no. 10, pp. 1521–1527, Oct. 2001.

[26] B. S. Morse and D. Schwartzwald, "Isophote-based interpolation," in *Proc. ICIP*, Oct. 1998, pp. 227–231.

[27] S. Jeschke, D. Cline, and P. Wonka, "Rendering surface details with diffusion curves," *ACM Trans. Graph.*, vol. 28, no. 5, p. 117, Dec. 2009.

[28] W.-M. Pang, J. Qin, M. Cohen, P.-A. Heng, and K.-S. Choi, "Fast rendering of diffusion curves with triangles," *IEEE Comput. Graph. Appl.*, vol. 32, no. 4, pp. 68–78, Jul./Aug. 2012.

[29] S. Jeschke, D. Cline, and P. Wonka, "Estimating color and texture parameters for vector graphics," *Comput. Graph. Forum*, vol. 30, no. 2, pp. 523–532, Apr. 2011.

[30] E. Cohen, T. Tyche, and R. Riesenfeld, "Discrete *B*-splines and subdivision techniques in computer-aided geometric design and computer graphics," *Comput. Graph. Image Process.*, vol. 14, no. 2, pp. 87–111, Oct. 1980.

[31] C. Loop, "Smooth subdivision surfaces based on triangles," M.S. thesis, Dept. Math., Univ. Utah, Salt Lake City, UT, USA, 1987.

[32] H. Hoppe *et al.*, "Piecewise smooth surface reconstruction," in *Proc. ACM SIGGRAPH*, 1994, pp. 295–302.

[33] H. Hoppe, "Surface reconstruction from unorganized point," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. Washington, Seattle, WA, USA, 1994.

[34] J. E. Schweitzer, "Analysis and application of subdivision surfaces," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. Washington, Seattle, WA, USA, 1996.

[35] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[36] J. H. Elder and S. W. Zucker, "Local scale control for edge detection and blur estimation," *IEEE Trans. Pattern Anal. Mech. Intell.*, vol. 20, no. 7, pp. 699–716, Jul. 1998.

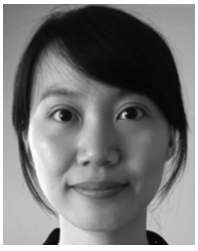
[37] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in *Proc. IEEE Conf. CVPR*, Jun. 2008, pp. 1–8.

[38] J. R. Shewchuk, "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator," in *Proc. Appl. Comput. Geometry, Towards Geometric Eng.*, May 1996, vol. 1148, pp. 203–222.

[39] K. Okarma, "Colour image quality assessment using structural similarity index and singular value decomposition," in *Proc. Int. Conf. Comput. Vis. Graph.*, Nov. 2008, pp. 55–65.

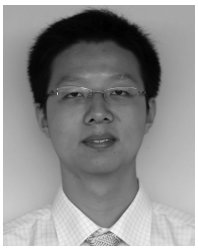
[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[41] L. Demaret, N. Dyn, and A. Iske, "Image compression by linear splines over adaptive triangulations," *Signal Process.*, vol. 86, no. 7, pp. 1604–1616, Jul. 2006.



Hailing Zhou (M'13) received the B.Eng. (Hons.) degree in computer science from Xidian University, Xi'an, China, in 2006, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2012, where she was a Project Officer in 2012. She is currently a Research Fellow with the Centre for Intelligent Systems Research, Deakin University, Burwood, VIC, Australia. Her main areas of research are computer vision, image processing, human–computer interaction, and computer graphics.

Jianmin Zheng received the B.S. and Ph.D. degrees from Zhejiang University, Hangzhou, China. He is currently an Associate Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His current research interests include computer-aided geometric design, computer graphics, animation, visualization, and interactive digital media. He has authored more than 100 technical papers in international conferences and journals. He is an Associate Editor of *The Visual Computer*.



Lei Wei (M'13) received the B.Eng. degree in computer science from Tianjin University, Tianjin, China, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2006 and 2011, respectively. He was a Research Associate at NTU before he was a Post-Doctoral Research Fellow at the Centre for Intelligent Systems Research, Deakin University, Burwood, VIC, Australia. His research interests include haptic rendering, haptic collision detection, networked haptics, medical haptics, human–computer interaction, and image processing.