A Subdivision-Based Representation for Vector Image Editing

Zicheng Liao, *Student Member*, *IEEE*, Hugues Hoppe, *Member*, *IEEE*, David Forsyth, *Fellow*, *IEEE*, and Yizhou Yu, *Senior Member*, *IEEE*

Abstract—Vector graphics has been employed in a wide variety of applications due to its scalability and editability. Editability is a high priority for artists and designers who wish to produce vector-based graphical content with user interaction. In this paper, we introduce a new vector image representation based on piecewise smooth subdivision surfaces, which is a simple, unified and flexible framework that supports a variety of operations, including shape editing, color editing, image stylization, and vector image processing. These operations effectively create novel vector graphics by reusing and altering existing image vectorization results. Because image vectorization yields an abstraction of the original raster image, controlling the level of detail of this abstraction is highly desirable. To this end, we design a feature-oriented vector image pyramid that offers multiple levels of abstraction simultaneously. Our new vector image representation can be rasterized efficiently using GPU-accelerated subdivision. Experiments indicate that our vector image representation achieves high visual quality and better supports editing operations than existing representations.

Index Terms—Vector graphics, subdivision surfaces, multiresolution representation, vector image editing

1 Introduction

There has been a recent resurgence of vector-based graphical content in personal computers and on the Internet. For example, major operating systems have increasingly adopted vector graphics in their user interface, and Adobe Flash has strengthened support for vector graphics in rich internet applications. Vector-based drawing tools, such as Adobe Illustrator and CorelDRAW, enjoy immense popularity among artists and designers. Such a wide range of applications is made possible by the fact that vector graphics is both editable and scalable. Editability is a high priority for artists and designers who wish to conveniently produce visual content with user interaction.

Since imaging devices typically produce raster images, image vectorization remains an important means for generating vector-based content. A recent trend in vector graphics research focuses on developing scalable (resolution-independent) representations of full-color raster images. One long-lasting challenge on this front is to make vectorized images easily editable so that artists and designers can incorporate them into their artwork. Since a full-color raster image typically has significant pixel-level detail and not all of this detail needs to be preserved in the abstracted version, a second challenge is to let users easily choose a desired level of detail for a vectorized image.

- *Z. Liao, and D. Forsyth, are with the Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N. Goodwin Ave., Urbana, IL 61801. E-mail: {liao17, daf}@illinois.edu.*
- *H. Hoppe is with Microsoft Research, One Microsoft Way, Redmond, WA 98052. E-mail: hhoppe@microsoft.com.*
- *Y. Yu is with the Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N. Goodwin Ave., Urbana, IL 61801 and the Department of Computer Science, University of Hong Kong. E-mail: yyz@illinois.edu, yizhouy@acm.org*

Manuscript received 3 Oct. 2011; revised 4 Jan. 2012; accepted 1 Feb. 2012; published online 29 Feb. 2012.
Recommended for acceptance by L. Kobbelt.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-10-0244.
Digital Object Identifier no. 10.1109/TVCG.2012.76.

In this paper, we introduce a vector image representation to meet the aforementioned challenges. In our representation, the image plane is decomposed into a set of triangular patches with potentially curved boundaries, and the color signals over the image plane are treated as height fields. A subset of the curved patch boundaries are automatically aligned with curvilinear features. The geometry of the patch boundaries as well as the color variations over the patches are represented using a piecewise smooth Loop subdivision scheme. Such a simplicial layout of patches avoids T-junctions and better supports feature-sensitive patch boundary alignment. With properly defined subdivision masks, the patch boundary curves are C^2 everywhere, and the color function is at least C^1 everywhere except across features where it is discontinuous.

To offer the flexibility of multiple levels of abstraction, we also design a multiresolution vector image representation. Different resolutions in this representation contain progressively coarser meshes, each one acting as the control mesh of a piecewise smooth subdivision surface. Because image features play a crucial role in vector image representations, our multiresolution representation is feature centric. Features are sorted and distributed to different resolutions according to their saliency scores. When switching between different resolutions, we "downsample" or "upsample" features rather than pixels. Multiple resolutions allow the user to choose a desired level of abstraction during image vectorization or vector image editing.

Using the piecewise smooth subdivision representation, we develop techniques to facilitate a variety of vector image editing operations, including shape editing, color editing, image stylization, and vector image processing. Such editing operations effectively create novel vector graphics by reusing and altering existing vectorized images. While shape editing, color editing, and image stylization can be applied to any single resolution, vector image processing involves inherently hierarchical operations that affect multiple levels simultaneously. Fig. 1 shows an example of vectorizing a raster image with our representation and shape/color editing results.

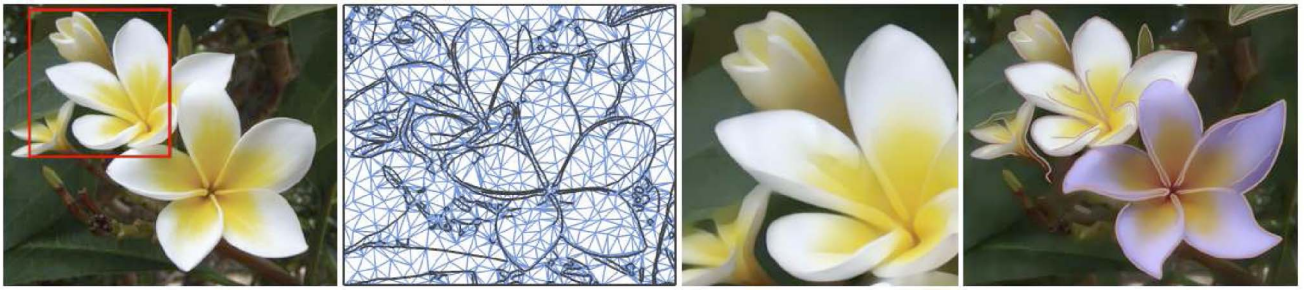


Fig. 1. A raster image converted to a piecewise smooth vector-based representation with curvilinear features. Guided by the feature curves, a multiresolution vector image pyramid enables intuitive editing of the resulting vector graphics. Left: Original image. Middle left: Subdivision surface control mesh for vectorization. Middle right: Magnified ($4\times$) local view of the vectorized image. Right: Combined effects of shape editing, color editing, and stylization on the vector representation.

We summarize our contributions in this paper as follows:

- We introduce a new vector image representation based on piecewise smooth subdivision surfaces. It is the first work that applies subdivision surfaces to modeling image with discontinuity curves. The fact that this representation automatically provides the desired continuity conditions is particularly useful for both vectorization and subsequent vector editing operations. Due to its simplicity and elegance, this representation is a unified and flexible framework that may find many other uses in vector image representations.
- This work supports a novel feature-oriented multiresolution vector image representation. Unlike traditional multiresolution mesh representations for shape editing, the most important motivation of our multiresolution representation is not facilitating vector image editing, but providing multiple levels of visual abstraction. According to their own preferences, users may choose different levels as the final vectorization result.
- This is also the first work that focuses extensively on vector image editing and processing. Our representation lets us process vector images directly, and achieves novel results different from such operations on raster images. Research in this direction is significant because it directly processes vector images without the need to go through any intermediate raster image representations. We expect this work to stimulate further research on processing of vector image representations.

2 Related Work

There exists extensive previous work on vectorization of non-photographic images [17], [18], [31], [32], which include fonts, clip arts, maps, and line drawings. However, in this paper we focus on photographic images. Existing vectorization techniques for full-color raster images fall roughly into three categories.

Triangulations. A few algorithms have been proposed based on constrained Delaunay triangulation [16], [22], [29]. In triangulation-based representations, each curvilinear feature needs to be approximated by many short line segments. Yet, these are still not resolution independent because the differences between a smoothly curved feature and a polyline with only C^0 continuity at the vertices become more obvious when magnified. Our technique overcomes this problem by fitting subdivision curves to patch boundaries. Such subdivision curves have C^2 continuity.

Parametric patches. Techniques involving higher order parametric functions, such as grids of Bézier patches [12], [25] or Ferguson patches [30], aim for a more editable and flexible vector representation. A vectorization technique based on optimized gradient meshes was introduced in [30], where manual mesh initialization is required to align mesh boundaries with salient image features. Such user-assisted mesh placement can be time consuming for an image with a large number of features. Gradient meshes are defined to be smooth everywhere, except at holes as introduced in [21]. While color discontinuities can be approximated by introducing degenerate quads or fold-overs, this is less convenient than a general network of tear curves. In addition, the rectangular arrangement of patches in gradient meshes hinders a highly adaptive spatial layout, making it challenging to align color discontinuities with image features. In comparison, our simplicial layout makes it easier to adaptively distribute patches and automatically align patch boundaries with all curvilinear features. Although the work in [12] uses triangular Bézier patches, it does not offer multiple levels of abstraction and its reconstructed color signals lack C^1 continuity across nonfeature region boundaries.

PDE solutions. A third category of techniques use a mesh-free representation. Diffusion curves [24], rely on curves with color and blur attributes as boundary conditions of a diffusion process. The final solution of this diffusion process defines the color variations of a vector image. This technique is particularly well suited for interactive authoring of vector graphics. However, it has a few limitations. First, diffusion curves are not coupled together by definition, which makes it hard to perform some vector image editing operations like region-based color or shape editing. In comparison, our technique builds a network of curved patches to better support vector image editing and signal processing. Second, diffusion curves focus primarily on discontinuity curves—they do not maintain detail between those sharp discontinuities. In contrast, our representation can approximate detail between the curves because we optimize the colors of interior vertices. Actually, our "unsimplified" mesh exactly reproduces the original image.

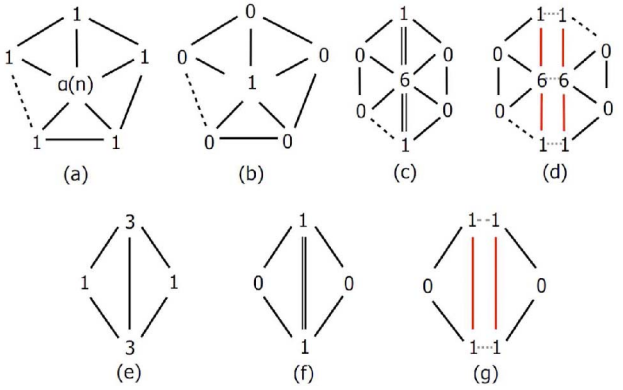


Fig. 2. Subdivision masks. (a-d) Subdivision vertex masks for smooth, corner, crease, and tear vertices. (e-g) Subdivision edge masks for smooth, crease and tear edges. In (d) and (g) the parallel vertex-pairs each connected by a gray dashed line are "split" vertices along the tear feature. In (a), $\alpha(n) = (\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n})^2 + \frac{3}{8}$ where n is the vertex valence.

3 Vector Image Representation

3.1 Subdivision Approach

We consider color variations in a raster image from a geometric perspective, treating each color channel as a height field over the 2D image domain. Thus, an image with three color channels is associated with a 2D surface in 5D space. Because an image has color discontinuities (i.e., features), we adopt a piecewise approximation. The image domain is partitioned into regions, each defining a locally smooth surface patch. Specifically, we define the complete piecewise smooth surface (spanning the full image domain) by adapting a piecewise smooth subdivision scheme [10], [11] as follows:

The subdivision scheme of Loop [11] defines a smooth (C^1) surface as the limit of a subdivision scheme applied to a control mesh $M = M^0$. The subdivision step $M^r \rightarrow M^{r+1}$ refines the mesh M^r by 1) replacing each triangle by four triangles and 2) computing vertex positions of M^{r+1} as affine combinations of nearby vertices in M^r, according to a set of subdivision *masks*. Each vertex in M^{r+1} is either a *vertex point* or *edge point*, depending on whether it corresponds to a vertex or edge in M^r, and the associated subdivision masks are shown in Figs. 2a and 2e.

In our setting, the control mesh is a 2D triangulation of the image domain, in which each vertex is a 5D vector (x, y, r, g, b). The effect of subdivision is to smooth both the 2D geometric positions and the 3D color coordinates. After subdivision, each triangle in the control mesh M^0 becomes a triangular region, generally with curved boundaries, and the color function is at least C^1 across all such boundaries.

The scheme of Hoppe et al. [10] extends subdivision to allow surface creases and corners, where the surface is continuous but not smooth. This is achieved by tagging control mesh edges as either *smooth* or *crease*.[1] However, for our purposes this is insufficient because the resulting surface is still everywhere continuous.

3.2 Discontinuous Subdivision

To model discontinuous functions, we further extend subdivision by introducing a third type of edge, a *tear*, which has the effect of splitting each adjacent vertex into two vertices (Figs. 2d and 2g). These two vertices share the same x, y spatial coordinates, so that the triangulation maintains a bijection onto the image domain. However, the two vertices may have different r, g, b color coordinates, so as to break color continuity.

A chain of tear edges is called a *tear feature*, and a chain of crease edges is called a *crease feature*. In this paper, we consider only tear features, because their associated discontinuities form the most prominent elements in vector graphics images. Vectorizing crease features, which are more subtle, is left as future work.

In our scheme, vertices have four types: *smooth*, *crease*, *tear*, and *corner*. A smooth vertex is a vertex incident only to smooth edges; crease and tear vertices are adjacent to exactly two crease and tear edges, respectively; corner vertices are located at all other configurations, including feature endpoints. To fix the rectangular image boundary, the four corners are marked as corner vertices, and all perimeter edges are marked as crease edges.

Fig. 2 shows the complete set of subdivision masks. The corner vertex mask ensures its position does not move after subdivision. The crease and tear masks both subdivide the feature curve to produce a cubic B-spline curve. The tear masks differ in that they act independently on the duplicated vertices across the tear.

In practice we apply two or three subdivision steps and then push the subdivided vertices to their limit positions (using a set of limit masks, not shown). Although the mesh could be further subdivided, we find that it already starts to form a sufficiently accurate piecewise linear approximation.

The goal of vectorization (Section 4) is to 1) optimize the vertex positions along this feature to align the resulting subdivided feature curve with the raster image discontinuities, and 2) optimize the vertex colors such that the piecewise smooth subdivided mesh best fits the image color function.

In this vector graphics setting, the piecewise smooth subdivision approach offers a number of benefits. First, it represents both the shapes of image features and the variations of color signals in a unified, resolution-independent representation. Second, it achieves the desired spatial and color continuity conditions by construction, without requiring constraints over the degrees of freedom, namely: 1) the subdivided feature curves are everywhere C^2, and 2) the color function is everywhere C^1, except across feature curves where it is C^{-1} (Actually, it is also C^2 away from extraordinary vertices, which are those with valence other than 6.). Because the vector image will be subject to interactive user manipulation, these properties guarantee that no matter how the user deforms the control meshes, feature curves will remain geometrically smooth, and the color field will remain smooth everywhere except across features. In comparison, the piecewise color representation in [12] may give rise to undesired visible seams across region boundaries.

3.3 Multiresolution Representation

While our vector image representation involves a sequence of progressively finer meshes, these meshes all share the same set of features—the same amount of detail. In Section 5, we extend this with a multiresolution structure,

1. We use the terminology "crease" rather than "sharp" to make clearer our further generalization.

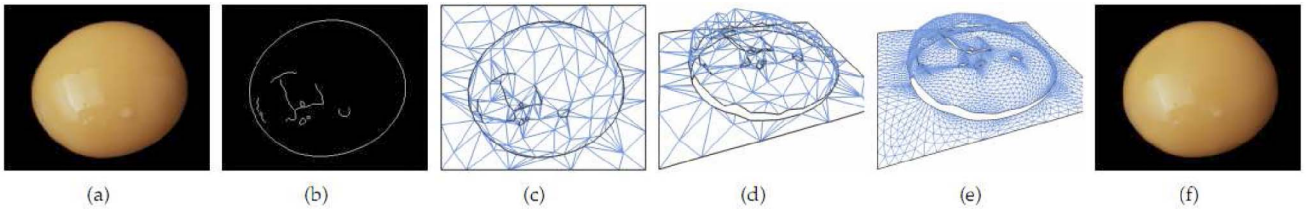


Fig. 3. Vectorization pipeline. (a) Original image. (b) Detected curvilinear features. (c) Control mesh of the reconstructed subdivision surface. (d) 3D view of the optimized control mesh. (e) Optimized control mesh subdivided twice. (f) Rasterization result of the reconstructed vector image (1.40/pixel mean reconstruction error using the control mesh of 303 (0.3 percent) vertices and 369 triangles).

in which each resolution level is itself a vector image, and contains a different level of detail.

4 Single-Level Image Vectorization

Our image vectorization pipeline consists of four major stages: feature detection, initial control mesh construction, mesh simplification, and color optimization.

Feature detection is performed through Canny edge detection and image segmentation. Detected Canny edges are thinned to 1-pixel wide, and broken pieces are linked together to form longer features [19]. If image segmentation (we use GrabCut [1] in our experiments) is performed to partition an image into regions, region boundaries are always closed and are also treated as features. The initial control mesh is created as a regular grid, with one vertex per image pixel. Additional vertices are introduced in the mesh at subpixel locations based on the detected features, and the mesh is locally retriangulated appropriately. All edges along features are marked as tear edges. Initial mesh construction and subsequent feature-preserving mesh simplification follow the work of Xia et al. [12] except that we use subdivided feature curves to fit image features. Note that more advanced feature detection method, such as the one in [28], could be adopted to achieve better subpixel accuracy without affecting our overall vectorization pipeline. We would like to leave this as a topic for future investigation.

Mesh simplification. Because the mesh is initially very dense, for efficiency we perform simplification using the quadric error metric of [4], treating the color channels as geometric height fields. To preserve the topology of feature tears, each tear vertex is only permitted to collapse with an adjacent vertex on the same tear. And to carefully preserve the geometric fidelity of the feature curves, after each collapse involving a tear vertex we solve an optimization to locally refit the subdivided feature curve to its associated feature in the raster image. This geometric optimization is formulated to minimize the summed squared distances between vertices of the densely subdivided mesh and their target positions:

$$E = \sum_{j=1}^{N_s} \|\mathbf{x}_j - \mathbf{V}\mathbf{y}_j\|^2, \tag{1}$$

where $\mathbf{V}$ is a $2 \times N_c$ matrix of the N_c unknown control vertices, and N_s is the number of affected vertices in the subdivided mesh. Vector $\mathbf{x}_j$ is the target position of the jth subdivided vertex, and $\mathbf{V}\mathbf{y}_j$ is the expression for the limit position of the jth subdivided vertex in terms of the control vertices. The target position $\mathbf{x}_j$ for a tear vertex is its projection onto the original feature curve; for all remaining vertices it is their current position.

Minimizing E is a sparse linear least-squares problem since the local nature of subdivision rules ensures that each $\mathbf{y}_j$ is a sparse vector. If the maximum fitting error along the new curve exceeds one pixel, the edge collapse is rolled back. Also, we prevent foldovers by disallowing edge collapses that result in flipped triangles. Once the number of vertices in the control mesh has been reduced to a predefined threshold, mesh simplification terminates and the structure of the control mesh becomes final.

Color optimization. Because the mesh simplification process is greedy and heuristic, the solution is far from optimal. In fact, the colors in the simplified mesh do not take into account subdivision at all. The final step globally optimizes the colors of the control mesh vertices. We use a formulation similar to (1), but this time over 3D colors rather than 2D positions. Thus, $\mathbf{V}$ becomes a $3 \times N_c$ matrix containing all control vertex colors, and N_s is the total number of vertices in the subdivided mesh. The target color $\mathbf{x}_j$ is the bilinearly filtered image color at the 2D location of the corresponding subdivided vertex.

Because color values may vary significantly across image features, missampling near features in the original raster image can result in disturbing results. Thanks to the one-pixel error bound in the earlier feature fitting, we need only pay special attention to vertices in a one-pixel band adjacent to the features. We obtain the target colors of these vertices as follows: For tear vertices themselves, the target color is assigned from the closest pixel on the feature. The remaining vertices that lie within one pixel from the features are referred to as *border vertices*. Their target colors are initially set to be undefined, and we perform hole filling to propagate correctly sampled colors from nearby interior vertices and tear vertices. Hole filling starts from the boundary of the holes and iteratively extends into the interior of the holes. The target color value of a vertex, whose color is previously undefined, is interpolated from the target values of its neighboring known vertices.

We solve the resulting large sparse linear system using TAUCS [3].

4.1 Results and Comparisons

Examples of vectorization and magnification can be found in Figs. 3 and 4. To demonstrate the quality and compactness of our vector image representation, we have compared our method with those in [12], [21]. As shown in Fig. 5, our result is at least C^1 across nonfeature patch boundaries whereas the result by Xia et al. [12] exhibits color discontinuities across such boundaries. Fig. 6 indicates that the amount of storage required by our method is comparable to gradient meshes.

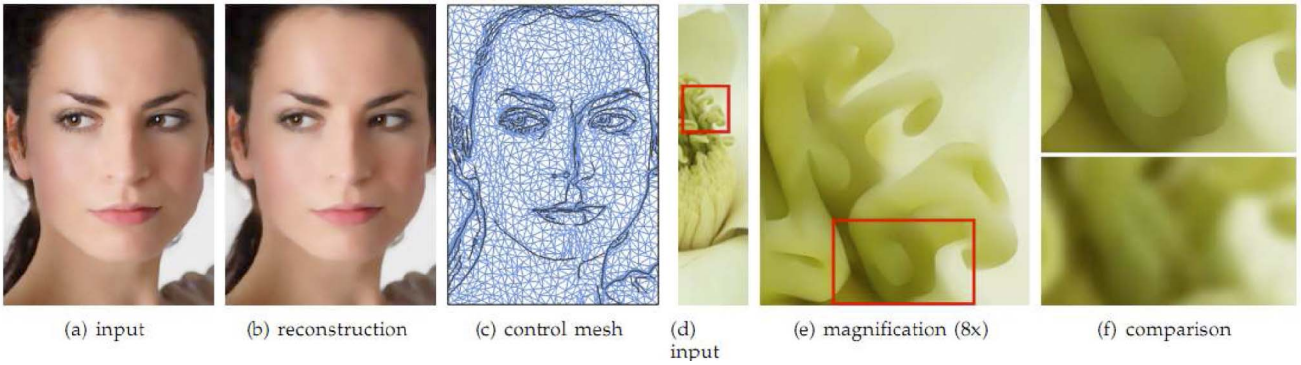


Fig. 4. Two vectorization examples. The left example uses a control mesh of 1,725 (1 percent) vertices and 2,470 triangles with mean reconstruction error 1.48; the right example shows a magnified view ($8\times$) of a local region of a flower pin using our vector representation and a comparison to the same scale magnification of the raster image using bicubic interpolation.

5 Multiresolution Vector Images

There are no universal criteria regarding the optimal density of features in a vectorized image. Denser features make the vectorized version more faithfully represent the original raster image while sparser features provide a higher level of abstraction, which could be more visually appealing. We introduce a feature-oriented multiresolution vector image representation to address this problem. Such a representation contains different levels of details at different resolutions, and thus provides vector-based approximations of a raster image over a spectrum of granularity and abstraction. It has the flexibility that users can choose their preferred level of abstraction in a vector image.

Our multiresolution vector images are based on features as basic building blocks due to their importance in vector representation. Thus, each resolution represents a distinct level of abstraction of the original raster image (Fig. 7). The multiresolution vector images are constructed as follows:

We first gather the set of features in the original image. Each feature f is assigned a *saliency score* that is a weighted summation of its length $P(f)$ and the average contrast (gradient magnitude) $C(f)$ across the feature

$$S(f) = P(f) + w\,C(f). \tag{2}$$

The user-configurable parameter w determines the relative importance of length and contrast. In addition, we allow users

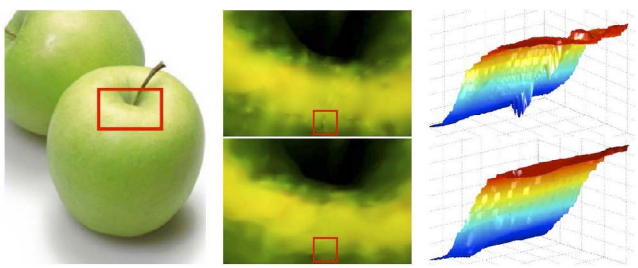


Fig. 5. Cross-boundary continuity: comparison with [12]. Left: Original image. Middle: Contrast-enhanced view of the vectorization of the local rectangular region by Xia et al. [12] (upper) and our method(bottom). Right: 3D reconstructed surface (gray scale as height) of the indicated local regions from the images in the middle. Note the color and geometric gradient discontinuities across patch boundaries from the result by Xia et al. [12] in the upper middle and upper right.

Fig. 6. Representation compactness compared to [21]. Left: Original image. Right: Vectorization result with mean error 2.13 using our method. Our subdivision-based representation takes up 14.0 KB of storage after zip compression; the gradient mesh representation [21] needs 9.4 KB storage with the same mean error, while JPEG compression with a comparable quality requires 20 KB.

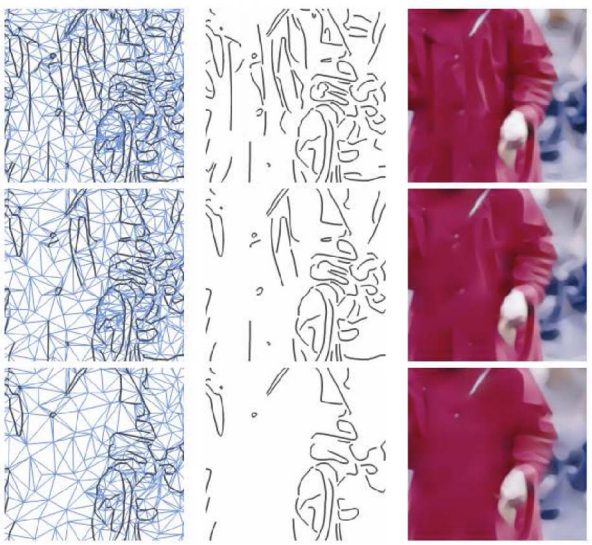


Fig. 7. Multiresolution abstraction. Top (a-d): Original raster image, the finest, intermediate, and coarsest levels of abstraction. Bottom (left to right): Cropped views of the control mesh, subdivided features, and vectorized image in three levels of abstraction.

to interactively adjust the saliency of semantically important features by interactively overriding their assigned resolution.

We uniformly group features into L subsets in descending order of saliency. This lets us define a sequence of *nested feature sets* $\{F_i\}_{i=0}^{L}$, where $F_j \subset F_i$ if $j > i$. In our multiresolution representation, we generate a single-level vector representation S_i for each feature set $F_0, \ldots, F_{L-1}$ such that S_i has C^1 continuity everywhere except for the subset of region boundaries aligned with F_i across which it has C^{-1} continuity.

Thus, we begin at level 0 with the finest control mesh, which contains all features. Level l is constructed from level $l-1$ by first removing the subset of features $F_{l-1} \setminus F_l$. Recall that every control vertex along a feature is paired with another vertex on the opposite side of the feature, and these vertices have the same x- and y-coordinates but different color coordinates. When a feature is eliminated, the open boundary it creates becomes sealed, and every pair of vertices on the boundary is merged into a single vertex with an averaged color. Second, mesh simplification is performed to eliminate a certain percentage of the vertices. In the current implementation, we remove 50 percent of the vertices between two consecutive levels by default. During this stage of simplification, each merged vertex on a just-eliminated feature is allowed to collapse with any other vertex, while a vertex on a remaining features is constrained to collapse only with other vertices on the same feature.

Note that w and L are heuristic parameters. In our experiments we always use default values, $w = 2$ and $L = 3$. However, users can choose to assign them alternative values through an interface.

6 Vector Image Editing

Editability is the main reason that vector graphics is widely used in content design. Traditional vector graphics is represented with high-level geometric primitives with adjustable parameters so that editing operations can be conveniently achieved. In this section, we demonstrate that our new vector representation for photographic images also exhibits such an advantage and supports a variety of editing operations. Note that even though most editing operations addressed here can already be performed on raster images, our goal is to perform direct vector image processing without going through any intermediate raster images.

6.1 Shape Editing

Shape editing of an image object is achieved by deforming a part of the control mesh corresponding to the image object at an appropriate level of the multiresolution vector images. We use the as-rigid-as-possible shape manipulation technique in [7] to solve for a new configuration of the control vertices given user-supplied deformation constraints. A deformation constraint is a pair of original and new control vertex positions. Given one or more deformation constraints, the technique in [7] is able to solve for new positions of the remaining control vertices by minimizing the overall mesh distortion.

We have implemented a simple shape editing interface. Users can provide deformation constraints by dragging a single vertex or a feature. When a feature is selected, the user can partially deform the feature or completely relocate the entire feature. In the former case, the mouse click position is the center of deformation and the closest feature is selected. The displacement of any vertex on the selected feature is based on its initial distance to the center of deformation using a Gaussian kernel. Users can specify the variance (σ^2) of the Gaussian kernel to adjust the region of influence. In the latter case, the user can translate and/or rotate a selected feature to define the new positions of the vertices on the feature.

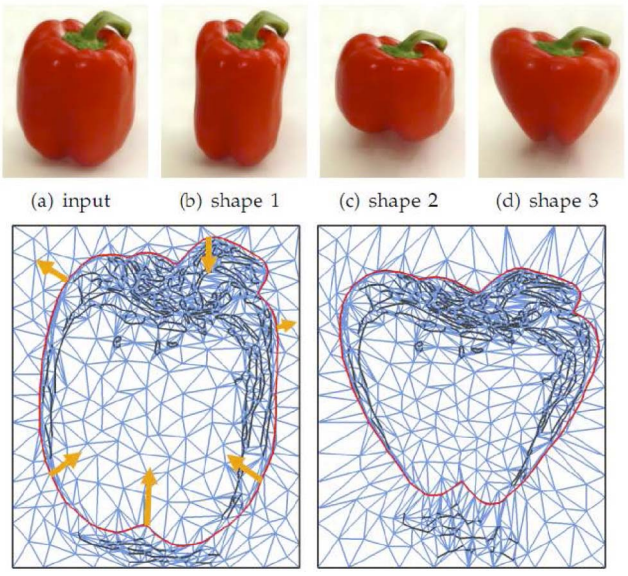


Fig. 8. Shape editing. Top row: Three shape editing results on a given vector image. Bottom row: Original control mesh for (a) and its deformation for (d) using six indicated mouse interactions.

Fig. 8 shows large-scale shape editing of an object silhouette to convincingly alter the perception of its 3D shape. Fig. 10 shows shape editing (and color editing, introduced in the next subsection) of multiple features in the same vector image to create a new facial expression. Note that such intuitive feature-oriented shape editing cannot be conveniently achieved with previous vector representations for photographic images. Diffusion curves [24] and individual gradient meshes in [30] are not coupled together by definition. Modern shape editing techniques, such as the one in [7] cannot be easily applied without significant enhancements to such representations. The automatic technique in [21] performs adaptive grid subdivision near image features but does not exactly align vertices with features, making high-precision feature selection and relocation hard to achieve. Although there is a base mesh holding all the Bézier patches together in [12], patch boundaries are individual Bézier curves. During shape deformation, the continuity between adjacent curve segments cannot be guaranteed without enforcing additional constraints among their control vertices.

6.2 Color Editing

With the mesh representation and explicit feature structures, color editing can be conveniently performed by defining region selection tools and then manipulating the color channels of the selected control vertices. Similar to the vertex and feature selection tool in shape editing, we support selecting a single vertex or an entire feature. Users can further specify a propagation radius to select a local region around the selected vertex or feature.

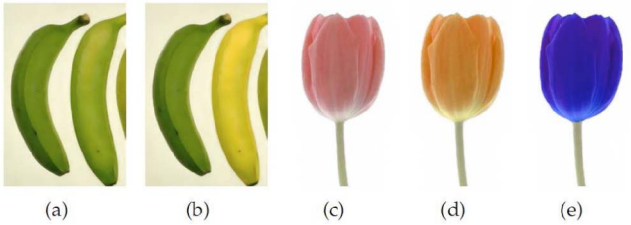


Fig. 9. Color editing. (a) Input vector image. (b) Color editing in the BLEND mode. (c) Input vector image 2. (d)-(e) Color editing in the TRANSFORM mode.

For color manipulation, users specify the rgb values of a new color that will affect the color of the selected vertex or feature. There are of course many transformation operators that one can apply. In our prototype we have explored two such operators, *BLEND* and *TRANSFORM*. In the BLEND mode, the final color is computed as a linear blend of the original and new colors. In the TRANSFORM mode, a seed vertex closest to the mouse click location is first chosen and a 3×3 diagonal color transform matrix is computed using the new color and the original color of the seed vertex. This transform matrix is then applied to all vertices within the selected local region. Both color editing modes preserve the original color variations in the selected region. Fig. 9 shows color editing results achieved with BLEND and TRANSFORM operators.

6.3 Abstraction and Stylization

Our multiresolution vector images provide a sequence of control meshes with progressive density. These control meshes generate subdivision surfaces that approximate the original raster image at different levels of details. Finer levels more faithfully represent the original raster image while coarser levels provide a higher level of abstraction with the removal of edges with low salience. This structure gives a natural solution to edge-aware multilevel image abstraction, which allows users to choose an appropriate abstraction level to display an image for various purposes.

We further generate stylized images from multiresolution vector images by drawing freestyle strokes along a subset of features (Fig. 11). Stylization requires a user-selected abstraction level and interactively selected regions of interest where features are going to be emphasized with strokes. In comparison with [8], where the input image is segmented into regions, each of which is filled with a constant color, our results put more emphasis on sharp image features, which are aligned with partial region boundaries, and preserve weakened color variations within local regions. Both methods show visually interesting results, but with different

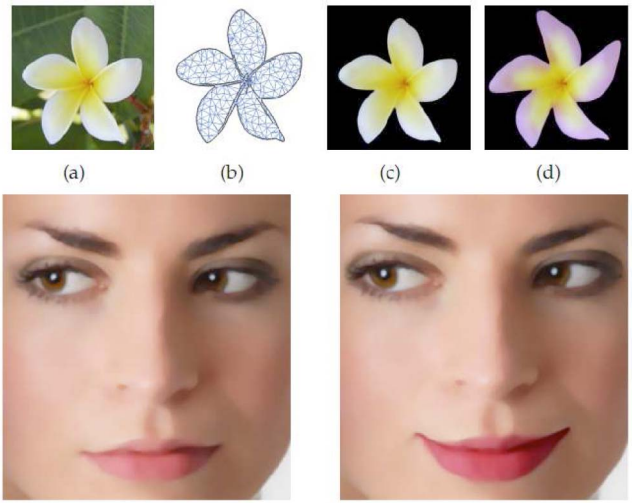


Fig. 10. Combined shape and color editing. Upper: (a) raster image; (b) control mesh of the extracted foreground layer; (c) foreground object vectorization; (d) shape and color editing to the object. Bottom left: Vector image input. Bottom right: Shape and color editing on the vectorized image. Shape editing includes deforming the mouth and eye brows, and enlarging the eyes. Color editing is performed on the lips.

stylization emphases. In our results, strokes are only used to enhance features within regions of interest. Within a region of interest, features with saliency scores higher than a threshold are always enhanced with strokes while features with saliency scores below the threshold are randomly chosen to be enhanced. The width of a stroke varies according to the length of the feature. Both ends of a stroke are linearly tapered.

Abstraction and stylization represent another novel application of our vector image representation. There have been no previous attempts to use vector image representations for such a purpose. Feature alignment and preservation as well as the removal of high-frequency details in our vector-based approximations are consistent with the goal of abstraction and stylization. Our results demonstrate that abstraction and stylization based on vectorization can be quite effective. Our technique also suggests a way to make smoother but edge-preserving base images for other methods that rely on a base- and detail-layer decomposition.

6.4 Signal Processing

It is desired to perform image processing tasks directly on a vector-based image representation, which eliminates the need to convert vector images back to raster images. Different levels of a multiresolution vector image, as introduced in Section 5, are mutually independent. Such a multilevel

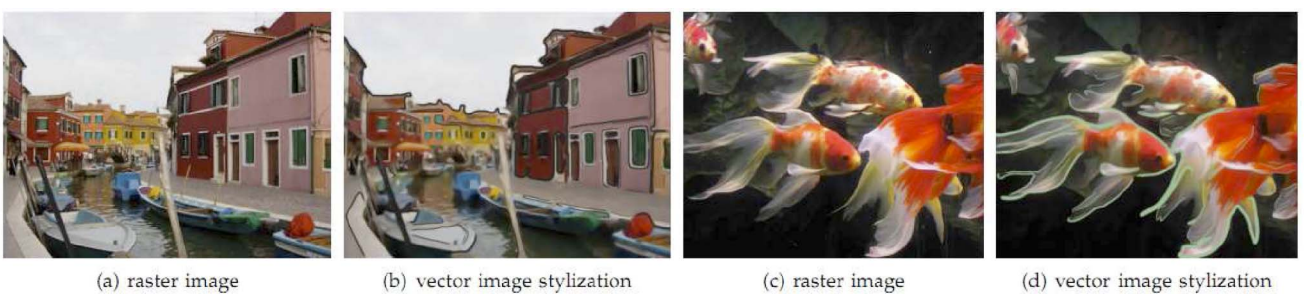


Fig. 11. Vector image stylization examples.

Fig. 12. Combined stylization and vector image processing results.

structure becomes inadequate for vector image processing tasks, such as filtering and enhancement, which need to work with all frequency bands simultaneously. We further enhance our multiresolution vector image representation by storing interlevel details. The resulting data structure is called a vector image pyramid.

Our representation for interlevel detail in the pyramid shares similarities with the multiresolution mesh hierarchy proposed in [9], which was in turn inspired by the Burt-Adelson image pyramid [5]. (Interestingly, a vector image representation combines characteristics of both meshes and images.) The idea is that during the simplification of the original control mesh using a sequence of elementary coarsening operations (i.e., edge collapses), we record for each operation a detail vector that expresses the position (or data) of the removed vertex relative to the resulting coarse neighborhood. Specifically, the removed vertex is predicted as a weighted combination (relaxation) of the coarse neighboring vertices, and the detail vector is the difference from this prediction. Interested readers are referred to [5], [9] for more details.

Some differences between our vector image pyramid and the multiresolution mesh hierarchy in [9] are summarized as follows:

- **Relaxation.** The relaxation operation R used to predict a vertex from its one-ring neighborhood has weights from [2]:

$$R(\mathbf{v}) = \sum_{i=1}^{n} w_i \mathbf{v}_i, \quad w_i \propto 1/\|\mathbf{v}' - \mathbf{v}'_i\| \text{ and } \sum_{i=1}^{n} w_i = 1, \tag{3}$$

where $\mathbf{v}'$ is the projection of $\mathbf{v}$ in the XY plane, which provides a perfect parameterization of our 2.5D color signal. The Fujiwara weights usually produce higher quality results in our experiments than the second-order divided differences in [9].

- **Local frames and detail vectors.** We store 2D position displacement vectors with respect to local frames in the simplified meshes. For color displacements, we simply use per-channel differences with respect to the global frame whose z-axis is perpendicular to the image plane.

  The detail vectors and scalars in the pyramid construction process store the differences between actual signals and their smoothly predicted version from the relaxation operation. Within a vector image pyramid, detail signals at finer levels accommodate relatively high-frequency details while those at coarser levels accommodate low-frequency details. As in [9], signal processing operations such as low-pass, high-pass, and band-pass filtering can be performed conveniently by appropriately editing such detail signals. Filtering and enhancement based on editing detail signals can be formulated as

$$\mathbf{v}' = R(\mathbf{v}) + \eta d(\mathbf{v}), \tag{4}$$

where the edited vertex $\mathbf{v}'$ is obtained from its relaxed prediction $\mathbf{v}$ and by scaling the precomputed detail signal in 2D geometric coordinates and/or 3D color coordinates. Smoothing is achieved by setting $0 < \eta < 1$, and enhancement is achieved by setting $\eta > 1$. Setting η as a function of pyramid level achieves filtering effects dependent on frequency bands.

Figs. 13 and 14 show two signal processing examples. Fig. 12d shows a combined effect of filtering and stylization. These results demonstrate that standard signal processing operations can be directly performed on a vector image without the need to convert it to a raster image first.

Note that signal processing operations have not been supported in previous vector image representations. Unlike

Fig. 13. Signal processing using our vector image representation. Upper left: Raster image. Bottom left: Vector approximation. Upper middle: Low-pass filtered vector approximation. Bottom middle: High-frequency enhanced vector approximation. Upper right: Difference map of the smoothed image and vector image. Bottom right: Difference map of the enhanced image and the vector image.

Fig. 14. Left: Original raster image. Middle: Vector image detail enhancement. Right: Difference map due to vector enhancement.

our multiresolution vector representation, they were not originally designed for signal processing tasks. Comparing to raster image processing, our cut-open mesh structure along sharp image features leads to perfect edge-preserving smoothing without the need of any extra treatment while the bilateral filter or other edge-preserving raster image filtering algorithms only partially preserve contrast across sharp edges.

7 Discussion

7.1 GPU-Based Rasterization

We rely on GPU-based rasterization of subdivision surfaces to achieve real-time vector image display. Recent work on real-time surface subdivision can be found in [33], [34]. In our experiment, we implemented rasterization using CUDA [15] on nVidia Geforce GTX275. For a display window with a moderate size (512×512) our GPU-based rasterization yields 60 frames per second. We do uniform subdivision on the control mesh and terminate when the total number of triangles exceeds the number of pixels in the display window. Note that a zoomed view only requires a portion of the control mesh to be subdivided. Thus, the rendering speed is determined by the display window size rather than the image size.

A triangle with its ordered one-ring neighborhood is the atomic unit in our parallel implementation. Multiple iterations of subdivision are performed on the initial control mesh. Each iteration subdivides each of the triangles from the previous iteration into four smaller triangles each associated with an ordered one-ring neighborhood itself. To avoid heavy data swapping between the CPU and the GPU, we allocate sufficient global memory on the GPU at the beginning and manage the memory layout to make only one data swap during the whole subdivision process. Shared memory is utilized to achieve high speed data access. The per-block shared memory size is the major hurdle to achieving a high level of parallelization. In our experimental configuration, 32 threads are created and executed simultaneously with synchronization per block and a total of 240 blocks are allocated.

7.2 Vector Representation Statistics

Table 1 summarizes the control mesh complexity of the vector images used in the paper except for the ones that have been mentioned in the context.

7.3 Limitations

There exist a few aspects about our algorithm and implementation that deserve further investigation. Our vector image representation currently only supports a single foreground layer. While this assumption does not negatively impact most of the operations, it does affect shape editing. Separating different objects in an image onto distinct layers enables a user to alter the shape of each object independently. Issues related to multiple layers include how to automatically or semiautomatically recognize layers in an image and how to fill gaps created when two overlapping layers are altered differently. Another limitation is that our current implementation does not support the insertion of new features into a vectorized image. We expect this can be accomplished in a straightforward way by first performing intersection tests between the new features and the triangles in the control mesh followed by retriangulation around the intersections.

TABLE 1
Complexity of Vector Image Control Meshes: Ratio of Vertices Relative to Original Unsimplified Mesh; Number of Vertices, Triangles, and Features; Optimization Time in Seconds

Image (Fig. No.)	Ratio	Vert	Tri	Feat	Opt time
flowers (1)	0.015	3000	3612	174	3.0
flower pin (4e)	0.01	207	239	17	0.37
peppers (6)	0.0125	2294	3731	48	9.8
girl (7)	0.05	6453	9356	306	5.9
pepper2 (8)	0.015	1960	2518	182	6.3
banana (9a)	0.01	883	1258	20	3.1
tulip (9c)	0.03	3313	5342	63	3.0
flower2 (10b)	0.0125	426	570	15	4.3
face (10 bot.)	0.05	3462	5998	54	2.9
Italy (11b)	0.04	5741	7807	303	5.1
goldfish (11d)	0.025	6766	8753	594	6.1
grapes (12)	0.08	9210	15325	153	8.2
apple (13))	0.08	11308	21267	24	11.7
horse (14)	0.08	12966	22366	113	13.1

Statistics for images girl, apple, horse, grapes *are at the finest level. The number of vertices is halved at each level from the preceding finer level.*

8 Conclusions

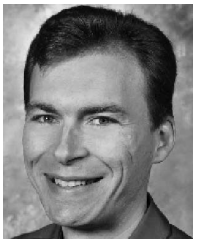
In this paper, we have introduced an effective vector-based representation and its associated vectorization algorithm for full-color raster images. Our representation is based on a triangular decomposition of the image plane and piecewise smooth Loop subdivision surfaces. We have also designed a feature-oriented vector image pyramid to support multiple levels of abstraction. Our multiresolution representation facilitates a variety of editing operations performed directly over a vector image. Experiments and comparisons have indicated that our representation and the associated vectorization algorithm can achieve high visual quality and better support editing operations than existing methods.

References

[1] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts," *ACM Trans. Graphics,* vol. 23, pp. 309-314, 2004.
[2] K. Fujiwara, "Eigenvalues of Laplacians on a Closed Riemannian Manifold and Its Nets," *Proc. Am. Math. Soc.,* vol. 123, pp. 2585-2594, 1995.
[3] S. Toledo, V. Rotkin, and D. Chen, "TAUCS: A Library of Sparse Linear Solvers. Version 2.2," Tel-Aviv Univ., 2003.
[4] M. Garland and P.S. Heckbert, "Surface Simplification Using Quadric Error Metrics," *Proc. ACM SIGGRAPH '97,* pp. 209-216, 1997.
[5] P.J. Burt and E.H. Adelson, "Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Comm.,* vol. C-31, no. 4, pp. 532-540, Apr. 1983.
[6] K.D. Cheng, W. Wang, H. Qin, K.K. Wong, H. Yang, and Y. Liu, "Design and Analysis of Optimization Methods for Subdivision Surface Fitting," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 5, pp. 878-890, Sept./Oct. 2007.
[7] T. Igarashi, T. Moscovich, and J.F. Hughes, "As-Rigid-As-Possible Shape Manipulation," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 1134-1141, 2005.
[8] D. DeCarlo and A. Santella, "Stylization and Abstraction of Photographs," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 769-776, 2002.

[9] I. Guskov and W. Sweldens, and P. Schröder, "Multiresolution Signal Processing for Meshes," *Proc. ACM SIGGRAPH '99,* pp. 325-334, 1999.
[10] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise Smooth Surface Reconstruction," *Proc. ACM SIGGRAPH,* pp. 295-302, 1994.
[11] C. Loop, "Smooth Subdivision Surfaces Based on Triangles," masters's thesis, Dept. of Math., Univ. of Utah, 1987.
[12] T. Xia, B. Liao, and Y. Yu, "Patch-Based Image Vectorization with Automatic Curvilinear Feature Alignment," *ACM Trans. Graphics,* vol. 28, no. 5, pp. 1-10, 2009.
[13] H. Chang and Y. Hong, "Vectorization of Hand-Drawn Image Using Piecewise Cubic Bézier Curves Fitting," *Pattern recognition,* vol. 31, no. 11, pp. 1747-1755, 1998.
[14] J.Y. Chiang, S.C. Tue, and Y.C. Leu, "A New Algorithm for Line Image Vectorization," *Pattern Recognition,* vol. 31, no. 10, pp. 1541-1549, 1998.
[15] NVidia, "NVidia CUDA Programming Guide 2.0," http://developer.nvidia.com/object/cuda.html, 2008.
[16] L. Demaret, N. Dyn, and A. Iske, "Image Compression by Linear Splines over Adaptive Triangulations," *Signal Processing,* vol. 86, no. 7, pp. 1604-1616, July 2006.
[17] X. Hilaire and K. Tombre, "Robust and Accurate Vectorization of Line Drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 6, pp. 890-904, June 2006.
[18] R.D.T. Janssen and A.M. Vossepoel, "Adaptive Vectorization of Line Drawing Images," *Computer Vision and Image Understanding,* vol. 65, no. 1, pp. 38-56, 1997.
[19] P.D. Kovesi, "MATLAB and Octave Functions for Computer Vision and Image Processing," http://www.csse.uwa.edu.au/~pk/research/matlabfns/, 2012.
[20] S. Lee, K. Chwa, and S.Y. Shin, "Image Metamorphosis Using Snakes and Free-Form Deformations," *Proc. ACM SIGGRAPH '05,* pp. 439-448, 1995.
[21] Y.-K. Lai, S.-M. Hu, and R.R. Martin, "Automatic and Topology-Preserving Gradient Mesh Generation for Image Vectorization," *ACM Trans. Graphics,* vol. 28, no. 3, article 85, 2009.
[22] G. Lecot and B. Levy, "Ardeco: Automatic Region DEtection and COnversion," *Proc. Eurographics Symp. Rendering (EGSR),* pp. 349-360, 2006.
[23] D. Nehab and H. Hoppe, "Random-Access Rendering of General Vector Graphics," *ACM Trans. Graphics,* vol. 27, no. 5, article 135, 2008.
[24] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin, "Diffusion Curves: A Vector Representation for Smooth-Shaded Images," *ACM Trans. Graphics,* vol. 27, no. 3, article 92, 2008.
[25] B. Price and W. Barrett, "Object-Based Vectorization for Interactive Image Editing," *The Visual Computer,* vol. 22, no. 9, pp. 661-670, Sept. 2006.
[26] A. Patney and J.D. Owens, "Real-Time Reyes-Style Adaptive Surface Subdivision," *ACM Trans. Graphics,* vol. 27, no. 5, article 143, 2008.
[27] K. Zhou, X. Huang, W. Xu, B. Guo, and H.Y. Shum, "Direct Manipulation of Subdivision Surfaces on GPUs," *ACM Trans. Graphics,* vol. 26, no. 3, article 91, 2007.
[28] C. Steger, "Subpixel-Precise Extraction of Lines and Edges," *Int'l Archives of Photogrammetry and Remote Sensing,* vol. 33, no. 3, pp. 141-156, 2000.
[29] S. Swaminarayan and L. Prasad, "Rapid Automated Polygonal Image Decomposition," *Proc. IEEE CS 35th Applied Imagery and Pattern Recognition Workshop (AIPR '06),* p. 28, 2006.
[30] J. Sun, L. Liang, F. Wen, and H.H. Shum, "Image Vectorization Using Optimized Gradient Meshes," *ACM Trans. Graphics,* vol. 26, no. 3, article 11, 2007, doi: http://doi.acm.org/10.1145/1276377.1276391.
[31] S.H. Zhang, T. Chen, Y.F. Zhang, S.M. Hu, and R.R. Martin, "Vectorizing Cartoon Animations," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 4, pp. 618-629, July/Aug. 2009.
[32] J. Jia and H. Yan, "Cartoon Image Vectorization Based on Shape Subdivision," *Proc. Computer Graphics Int'l (CGI),* pp. 225-231, 2001.
[33] A. Patney, M.S. Ebeida, and J.D. Owens, "Parallel View-Dependent Tessellation of Catmull-Clark Subdivision Surfaces," *Proc. Conf. High Performance Graphics,* pp. 99-108, Aug. 2009.
[34] C. Eisenacher, Q. Meyer, and C. Loop, "Real-Time View-Dependent Rendering of Parametric Surfaces," *Proc. Symp. Interactive 3D Graphics and Games,* pp. 137-143, 2009.

Zicheng Liao received the BE degree from Zhejiang University and the MS degree from the University of Illinois at Urbana-Champaign and in 2008 and 2010, respectively. He is working toward the PhD degree at the Department of Computer Science, University of Illinois at Urbana-Champaign. His research interests include computer graphics, computer vision, and machine learning. He is a student member of the IEEE and the IEEE Computer Society.



Hugues Hoppe is a principal researcher and manager of the Computer Graphics Group at Microsoft Research. His main interests lie in the multiresolution representation, parameterization, and synthesis of both geometry and images. He received the 2004 ACM SIGGRAPH Computer Graphics Achievement Award for pioneering work on surface reconstruction, progressive meshes, geometry texturing, and geometry images. His publications include 30 SIGGRAPH/TOG papers. He is a fellow of the ACM, and recently served as papers chair for SIGGRAPH 2011 and an editor-in-chief of the *ACM Transactions on Graphics*. He is a member of the IEEE and the IEEE Computer Society.

David Forsyth received the BSc and MSc degrees from the University of the Witwatersrand, Johannesburg and the MA and DPhil degrees from Oxford University. He is currently a professor of computer science at the University of Illinois. His interests include computer vision, computer graphics and machine learning. He is a fellow of the IEEE and the IEEE Computer Society.

Yizhou Yu received the BS and MS degrees from Zhejiang University in 1992 and 1994, respectively, and the PhD degree from the University of California at Berkeley in 2000. He is currently an associate professor in the Department of Computer Science at University of Illinois at Urbana-Champaign and The University of Hong Kong. He is a recipient of the best paper award at 2005 and 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2002 National Science Foundation CAREER Award and 1998 Microsoft Graduate Fellowship. He is on the editorial board of Computer Graphics Forum, The Visual Computer, *and International Journal of Software and Informatics*. His current research interests include computational photography, computer animation, digital geometry processing, and video analytics. He is a senior member of the IEEE.