SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2017, Vol. 60 032104:1–032104:10
doi: 10.1007/s11432-016-0130-4

Patch-based variational image approximation

Hao XIE[1,2] & Ruofeng TONG[1*]

[1]*Artificial Intelligence Institute, Zhejiang University, Hangzhou* 310027*, China;*
[2]*College of New Media, Zhejiang University of Media and Communications, Hangzhou* 310018*, China*

Received February 24, 2016; accepted April 22, 2016; published online October 13, 2016

Abstract Vector graphic gives us a new solution to the representation of raster images. Among many types of vectorized representations, the most popular is mesh representation, which inherits the benefits of vector graphics. Inspired by mesh, we propose a novel patch-based representation for raster images, in which pixels are partitioned into regions, and pixels belonging to the same region are converted into a 3D point cloud and approximated by a 3D planar patch with proper boundaries in a variational way. The resulting patches are then encoded via a half-edge structure for storage. The key point is that the vertices of boundaries are not located on the very positions of sample points, i.e. converted pixels, but dependent on the optimal position of the patch, which theoretically reduces the fitting errors. Experiments show that our algorithm produces better results.

Keywords image approximation, vector graphics, image representation, plane optimization, boundary extraction

Citation Xie H, Tong R F. Patch-based variational image approximation. Sci China Inf Sci, 2017, 60(3): 032104, doi: 10.1007/s11432-016-0130-4

1 Introduction

Image representation is a large topic in the field of image processing. Mesh-based representation, as the most popular type of vector graphics, which is a better choice among various representations, inherits many advantages from vector graphics, i.e., easiness of editing, independence of resolution, and reduction of storage.

In most mesh representations, the color value attached to each vertex is always determined by the intensity value of the corresponding pixel in the input image. However, such kind of settings might not be the optimal for image approximation. Take a 1D case (see Figure 1) for instance.

In traditional methods (Figure 1(a)), the end points of the line segment (the vertices in 2D case) are usually located at the very positions where the sample data are, which might not be the optimal choice; while in Figure 1(b), an optimal line segment is obtained at a free position, rather than that in the traditional ones, which might be less reasonable. Note that the metric of fitting error is in a least squares sense.

To this end, we propose a novel kind of patch-based representation for raster images, in which the vertices of planar patches are independent on the intensity of pixels at the same positions. To model

* Corresponding author (email: trf@zju.edu.cn)

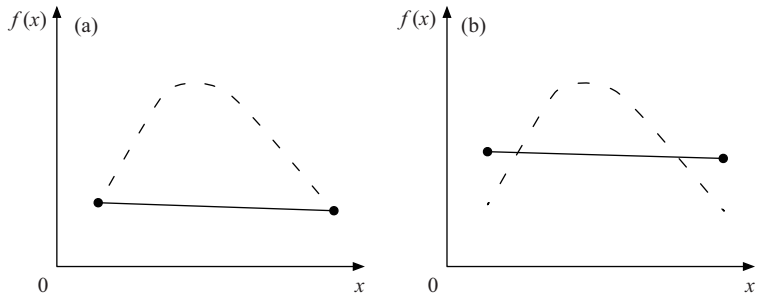

Figure 1 Comparison of fitting schemes. Traditional (a) and improved (b) fitting schemes for 1D case are illustrated. The dot curve denotes the sample points, and the straight line is the line segment to fit the sample points. As shown in the figure, (b) has less fitting errors than (a).

the problem, the pixels are considered as a height field in the form of point cloud, with the height value denoting the intensity of pixel, and the whole image is approximated by a given number of planar patches, which are then trimmed to form closed shapes, with or without holes in them. In this way, the pixels of an input image are partitioned into regions, and the pixels in each region are represented by a planar patch with one or more closed boundaries. Many of such patches thus cover and represent the input image.

Given an input raster image, our goal is to generate such a set of patches which could represent the input image as faithfully as possible. In general, the key idea of our algorithm is to minimize the gap between the point cloud and the patches in a variational sense. Specifically, given a set of initial patches, the process is performed in an iterative way. In each pass, the first thing is to partition all points, such that each point belongs to one patch; then the patches are optimized to fit the points, according to some metric. Once the gap between the patches and points is small enough, or the number of iterations is large enough, the process terminates. After this, proper boundaries are extracted for each patch, and organized as a polygon mesh or a mesh with spline curved edges for the sake of storing and displaying rendered images. This process is similar to the one in [1], and the difference is how to determine patch boundaries, which will be discussed later in details.

The contributions of our work are listed as follows:

• A novel planar patch-based image representation is proposed, in which the vertex colors are determined by optimization, rather than by a single pixel color, thus improves the quality of rendered results.

• The resulting patches are encoded via a half-edge structure for storage, which brings a neat image representation.

This paper is organized as follows: in Section 2, some related algorithms are reviewed. After a brief analysis of the original problem in Section 3, our main algorithm is delivered in Section 4. Some experimental results are shown in Section 5, and finally the conclusion is presented in Section 6.

2 Related work

To the best of our knowledge, few algorithms employ such patches as a vectorized representation of images, thus some similar image vectorization algorithms and their applications are reviewed in this section.

Frankly speaking, the most similar representation is the triangular-mesh-based representation, involving [2–5]. In [2], given an initial dense triangular mesh, the idea of this greedy algorithm is to iteratively remove vertices, until the vertices are sparse enough, that is, the number of vertices is reduced to a user specific value. At each time, a vertex with least influence is removed and the edges around are rearranged to satisfy a Delaunay triangulation. To save time, an Error Diffusion scheme [6] is employed to generate a sparser initial mesh, rather than the dense one. The algorithm in [2] is fast, yet the quality of rendered

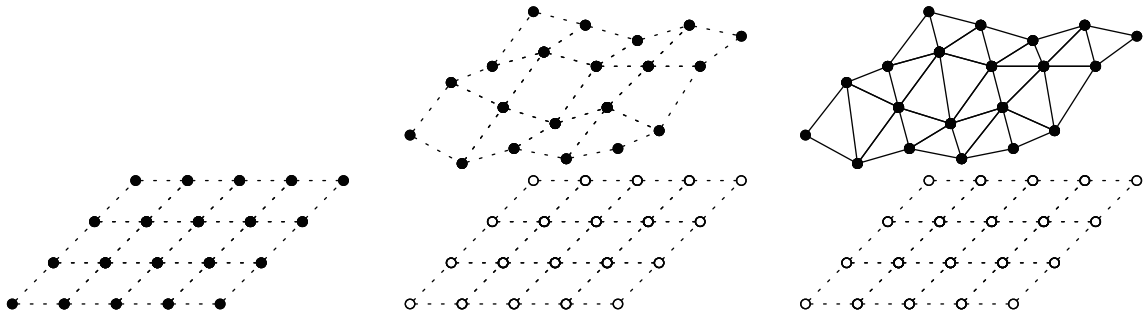

Figure 2 Conversion form image to mesh. Connectivity of pixels (black dots) in the image space (a), quadratic mesh (b), and the final triangular mesh (c) (white dots denote the previous positions of black dots).

images is not quite great, since the Delaunay triangulation criterion does not fit the requirement of data approximation.

Another type of algorithms [7] uses surface mesh to represent images. In [7], the authors first construct an initial dense surface mesh, with some crack holes corresponding to edge features of the input image; then perform a mesh simplification, keeping the main edge features of the image unchanged; and finally convert it into a Bezier triangular mesh. This algorithm could generate preferable results, yet the representation is somewhat complex.

There are still other types of image vectorization algorithms, such as gradient mesh based algorithms [8, 9], diffusion curve based algorithms [10–12], and partition of unity (PU) based algorithms [13]. They all have their advantages in representing raster images, and could generate different beautiful results.

Some applications of such structural image representation involve texture inpainting [14], image editing [15], image segmentation [16], and image extrapolation [17]. Most of them employ patch-based styles to represent images, which are entitled some semantic information, and such patch-based representation thus facilitates their own applications.

3 Problem modeling

The original task of image vectorization is to generate a kind of geometrical representation, e.g., mesh or something else, to minimize the gap between the geometrical primitives and the input image. This problem could be considered as: to find a proper set of geometrical primitives $\mathcal{G}$ minimizing the following L^p metric energy:

$$E(\mathcal{G}) = L^p(\mathcal{X}, \mathcal{G}) = \left(\frac{1}{|\mathcal{X}|} \int_{x \in \mathcal{X}} \|d(x, \mathcal{G})\|^p \mathrm{d}x \right)^{\frac{1}{p}}, \tag{1}$$

where $d(x, \mathcal{G}) = \inf_{g \in \mathcal{G}} \|T_I(x) - T_G(g)\|$, $\mathcal{G}$ is the set of geometrical primitives, $\mathcal{X}$ denotes the whole image domain, i.e., the set of all pixels, $T_I(\cdot)$ represents some feature operator of pixel x, e.g., color, $T_G(\cdot)$ is another feature operator which maps a geometrical primitive g into the same domain as $T_I(\cdot)$, $p > 0$ determines the type of metric, and $\|\cdot\|$ is ℓ^2 norm.

3.1 Problem conversion

The problem description above is abstract, and far from operable. In order to make it concrete, a conversion should be taken. Basically, for a raster image, the set of pixels belongs to a type of discrete domain, while the situation in Eq. (1) is a continuous one. Thus, our problem should be changed to be more continuous. As illustrated in Figure 2, such conversion is simply described below:

- All pixels are 4-connected to form a planar quadratic mesh with boundaries;
- Pixels are mapped from the image domain into a new 3D Euclidean space, with two dimensions unchanged and an extra dimension being the corresponding intensity value of pixels;

- Each unit of quadratic mesh is split into two triangles to form the final triangular mesh.

In such context, all pixels of the input image are converted from the image domain into a height field, with each height value being the intensity value. In order to improve the quality of rendering results, the mesh is divided into several regions. For each region, an optimal plane is obtained to fit the points in it. As a result, the input image is represented as several planar patches with region boundaries. Essentially, such patch-based representation actually employs a kind of linear interpolation to approximate the input image.

3.2 Objective function

Formally, this problem is specialized as follows: given an input raster image $I(x)$, where $x \in \mathcal{X} \subset \mathbb{R}^2$ with $\mathcal{X}$ denoting the set of all pixel points at the image domain, and also given the number of regions K, each pixel x of $I(x)$ is mapped to a point of the height field, $T_I(x) = (x_0, x_1, I(x))^{\mathrm{T}}$, where x_0 and x_1 denote two coordinates of pixel in the image domain, respectively, thus the whole image is converted into a height field. In addition, for each region, the geometrical primitive g is the plane (q, n), where q is a point on the plane and n is its normal. In this setting, operator $T_G(g) \in \{t|(t-q)\cdot n = 0\}$ and $|\mathcal{G}| = K$. Therefore, Eq. (1) is then specified as follows:

$$E(\mathcal{G}) = \sum_{k=1}^{K} \int_{x\in\mathcal{X}_k} \|T_I(x) - \Pi_k(x)\|^2 \,\mathrm{d}x, \tag{2}$$

where $\Pi_k(x)$ is the projected point of $T_I(x)$ onto the plane (q_k, n_k), which is supposed to approximate the k-th region $\mathcal{X}_k$.

Considering a measurement based on each triangular face, Eq. (2) is further embodied. Namely, by some simple calculus, the fitting energy for each triangle face T_i, relative to the k-th planar patch, has this form:

$$E_i(g_k) = \frac{|\widehat{T}_i|}{6} \sum_{p,q\in\{a_i,b_i,c_i\}} d_p d_q, \tag{3}$$

where $|\widehat{T}_i|$ is the area of the projection of T_i onto its belonging plane, d_p and d_q are the distance to the planar patch g_k from the p-th and q-th vertices v_p and v_q, respectively, and $\{a_i, b_i, c_i\}$ is the index set of the vertices of triangle T_i. Therefore, the original problem becomes to minimize the following energy:

$$E(\mathcal{G}) = \sum_{i\in\mathcal{I}} \inf_{g\in\mathcal{G}} E_i(g), \tag{4}$$

where $\mathcal{I}$ is the set of the whole triangle indices.

4 Patch-based representation

In this section, our patch-based representation of the input image is delivered in details. Our solution is inspired by geometric modeling for shape approximation [1], a similar idea is also proposed in [18]. To be clear, a brief structure of the solution is listed in Algorithm 1, in which two main steps are involved, namely plane optimization and boundary extraction. The first step aims to find the proper planes to approximate the mesh; while the second one is to find and express the boundaries of planes to form patches. Both steps are described below.

4.1 Plane optimization

Through the brief analysis above, the original problem is converted into an optimization one. To solve this problem, a variational method similar to [1] is employed, which is briefly reviewed in this section. Like many other solutions, this one also takes an iterative way to refine the result progressively, until some convergence is reached. In each pass, two steps are performed, namely triangle clustering and plane estimation.

Algorithm 1 Overview of our algorithm

Require: $N > 0$: number of iterations;
1: $i \Leftarrow N - 1$
2: PlaneOptimization:
3: **while** $i \geqslant 0$ **do**
4: TriangleClustering;
5: PlaneEstimation;
6: $i \Leftarrow i - 1$;
7: **end while**
8: BoundaryExtraction

4.1.1 *Triangle clustering*

Given an existing set of planes, the task of triangle clustering is to partition the mesh, and label each triangular face a tag, corresponding to a particular plane, such that the whole energy is minimized. Intuitively, each triangular face should be assigned the most nearest plane to it, in which the energy $E_i(g_k)$ is used as a measurement. Specifically, this could be considered as a mesh coloring problem. First, K seeds are chosen and each is assigned a different plane; then the labels are diffused away to cover the whole mesh. To accelerate this process, a priority queue is employed. At the beginning, K seeds and their adjacent triangles are pushed into the queue, and a tag is attached with each pushed triangle, indicating to which plane the triangle is being tested; then triangles in the queue are repeatedly popped up with least energy until the queue is empty. For each popped triangle, if it is not labeled, it will then be labeled as the tag it has, and its unlabeled adjacent triangles will be pushed into the queue with the same tag attached. When the queue is empty, all the triangles are labeled, and the partition is done.

4.1.2 *Plane estimation*

When the partition is done, the plane (q_k, n_k) of region $\mathcal{X}_k$ is updated accordingly by minimizing the energy

$$E(g_k, \mathcal{X}_k) = \sum_{i \in \mathcal{I}_k} E_i(g_k), \tag{5}$$

where $\mathcal{I}_k$ is the set of triangle indices belonging to the k-th region $\mathcal{X}_k$. The solution of this least squares optimization problem could be obtained by principal component analysis (PCA). The main purpose of PCA is to find main directions from a given group of points. In this context, all (infinite) points in region $\mathcal{X}_k$ is concerned. Therefore, according to the properties of PCA, q_k is simply the barycenter of region $\mathcal{X}_k$, and n_k is indicated by the eigenvector of the smallest eigenvalue of the covariance matrix of region $\mathcal{X}_k$, while eigenvectors of the other two eigenvalues are orthogonal with each other and both are parallel to the plane[1)]. That is,

$$q_k = \frac{\sum_{i \in \mathcal{I}_k} \bar{c}_i |T_i|}{\sum_{i \in \mathcal{I}_k} |T_i|}, \tag{6}$$

where $\bar{c}_i = \frac{1}{3} \sum_{p \in \{a_i, b_i, c_i\}} v_p$ is the barycenter of triangle T_i, and the covariance matrix[2)] is

$$\mathrm{COV}(\mathcal{X}_k) = \sum_{i \in \mathcal{I}_k} \left(M_i A M_i^{\mathrm{T}} + \bar{c}_i \bar{c}_i^{\mathrm{T}} - q_k q_k^{\mathrm{T}} \right) |T_i|, \tag{7}$$

where $A = \frac{1}{36} \begin{pmatrix} 7 & 10 & 0 \\ 10 & 7 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ and $M_i = (v_{c_i} - v_{a_i}, v_{b_i} - v_{a_i}, \mathbf{0})^{\mathrm{T}}$. For more details please refer to [1].

1) Since sample points are supposed to be distributed within a 2-manifold, and a plane is selected to fit these points, two main directions are parallel to the plane, and the last direction is then orthogonal to the plane, i.e. the normal.

2) According to the definition of covariance matrix, Eq. (7) could be inferred from $\sum_{i \in \mathcal{I}_k} \int_{T_i} (x - q_k)(x - q_k)^{\mathrm{T}} \mathrm{d}x$.

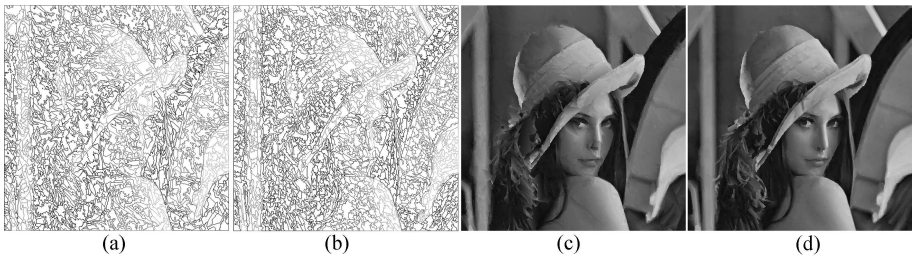

Figure 3 Example of boundaries and rendering results. Boundaries of Lena (a), (b) and corresponding rendered results (c), (d) with 0.5% sample ratio are shown respectively. Note that (a) and (c) are intermediate results with only 1 iteration; while (b) and (d) are final results after 2 iterations.

4.2 Boundary extraction

Now that the set of fitting planes, which are the essential components of the final planar patches, is obtained, the boundaries of these planes are also necessary; otherwise, there is no way to distinguish various regions. As a new form of representation, the boundaries of a plane should be easily expressed and stored, and this representation should also be robust enough to cope with various types of situations.

To achieve this, one could follow these steps:

(1) Find all vertices located at the boundaries of regions, the vertices of which usually have adjacent triangles belonging to more than one region.

(2) Record vertices with more than two adjacent regions from boundary vertices, which are supposed to be anchors of regions. As for the boundary with less than three such vertices, drop extra anchors to make sure that the boundary has at least three anchors. Four corner vertices, which correspond to four image corners, are also considered as anchors.

(3) Traverse each boundary, define an edge for the boundary vertices between each two neighboring anchors, and record the topological connectivity of anchors, i.e., the relationship between the edges and their adjacent regions.

(4) These edges are approximated by some polylines (via Ramer-Douglas-Peucker algorithm [19]) or splines, e.g., Bézier or B-spline curves (via least squares fitting [20]); the examples in this paper are all generated by polylines for simplicity.

After the above procedures, all data to be stored, involving both parameters and boundary information of planes, are available. The last thing to do is to organize and store them into a file with a special format, like the `*.off` file format for mesh. Topologically, such structure is much similar to a polygon, except that the patches (the polygonal faces) are not guaranteed to be connected geometrically. Thus, the well-known half-edge data structure is also suitable in this situation. Each edge is split into two half-edges, and each anchor corresponds to a vertex, so forms the half-edge structure.

When rendering, the boundaries of patches are discretized and connected to form polygons, then these polygons are depicted on the canvas with color being the height value of each vertex, and the inner region of polygons are filled via linear interpolation of their vertices. Note that the order of patches to be saved is closely related to the rendering result. For patches with two or more boundaries (one may consider the concentric circles for instance), only the outer boundary is stored, since the inner boundaries could be stored by other patches, for which these inner boundaries become the outer boundaries. Thus, patches containing inner boundaries should be stored (and accordingly rendered) before other patches; otherwise, patches located at the "holes" of other patches might be occluded. Figure 3 shows an example of the intermediate and final results of both wireframes and rendered images, in which boundaries are shown in intensities they should have in rendered images, leaving inner areas white.

5 Experiments

In this section, some experimental results are shown. Since the chief criterion of image vectorization is the recovery quality of the original image, the well-known peak signal-and-noise ratio (PSNR) [21]

Table 1 Quality comparisons. PSNR and SSIM values from [2] (with subscript a) and our algorithm (with subscript b) are listed in the last 4 columns, respectively. Larger PSNR and SSIM values are shown in bold.

Sampling ratio (%)	Image	Resolution	PSNR_a(dB)	PSNR_b(dB)	SSIM_a	SSIM_b
1			26.20	**29.46**	0.7784	**0.8684**
2	Lena	512×512	28.24	**30.39**	0.8180	**0.8938**
3			29.21	**31.01**	0.8368	**0.9060**
1			27.00	**30.04**	0.7191	**0.8346**
2	fruits	512×480	29.19	**31.47**	0.7847	**0.8722**
3			30.69	**32.01**	0.8190	**0.8938**
1			27.00	**28.95**	0.8244	**0.8639**
2	peppers	512×512	28.82	**30.02**	0.8538	**0.8867**
3			29.04	**30.31**	0.8655	**0.8990**
1			29.27	**30.19**	0.8444	**0.9069**
2	flower	512×480	**32.46**	32.28	0.8840	**0.9333**
3			**33.92**	32.95	0.8975	**0.9434**

and structural similarity (SSIM) index [14] are employed to evaluate the performance of algorithms. The definition of PSNR is simple: $\text{PSNR} = 20\log_{10}\frac{2^\rho-1}{\sqrt{\text{MSE}}}$, where ρ is the number of bits or samples in source or target images, and MSE stands for Mean Squared Error, which has the form $\text{MSE} = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\|I_r(x)-I(x)\|^2$ in this context, where $I_r(x)$ is the intensity value of rendered image at pixel x. It is obvious that the larger the PSNR value is, the better the performance of the algorithm is. While PSNR measures the per-pixel difference between two images with the same size, SSIM is a more complicated and overall image quality metric, which assesses the visual impact of an image, involving luminance, contrast and structure. In general, PSNR is not always a good measurement for visual quality of images. Thus, SSIM is employed as another perspective. Unlike PSNR, whose range is $[0,+\infty)$, SSIM has a range of $(-1,1]$, and a larger value also means a better quality. For more details of SSIM, please refer to [22].

In our experiments, several images are tested, most of which are common nature and human pictures. A set of PSNR and SSIM comparisons between [2] and ours is given in Table 1.

Note that our algorithm employs a different type of geometrical primitives, i.e., planar patches, to represent images. As far as we know, there is no exactly equivalent algorithm to compare with, thus, to make this comparison as fair as possible. The sampling ratio for [2] is the ratio of vertex number to pixel number; while for our algorithm, this ratio is that of patch number to the number of quads consisting of four adjacent pixels. As shown in Table 1, for most test images, our algorithm could generate preferable results. In our experiments, the initial partition of triangles has little influence on the final results. Therefore, a random initialization is employed. Moreover, we also find that the convergence is very fast. After only 2 iterations[3)], the results converge, and it takes no more than 3 s for all test images and all sampling ratios.

Figure 4 shows a comparison of rendered images between [2] (with 2% sample ratio) and our algorithm (with 1% sample ratio).

As shown in Figure 4, even if the sampling ratio of our algorithm is lower than that of [2], the rendering result of our algorithm is better. To see clearly, some local details are enlarged. In addition, some artificial effects generated by triangular mesh also have an influence on the result of [2]; while in our algorithm, these artificial effects are much weaker. The main reason for this difference is that the geometrical primitives of our algorithm are not triangles but polygons with more than three vertices when rendering. More comparisons are shown in Figure 5.

Our algorithm is implemented by C++ on an Arch Linux operating system, and the `Surface_mesh` library [23] is employed as the data structure of mesh. In addition, all test images are processed on a PC with an Intel(R) Core(TM) i7-4790K CPU and two 8 GB RAMs.

3) The intermediate results after only 1 iteration are shown in Figure 3.

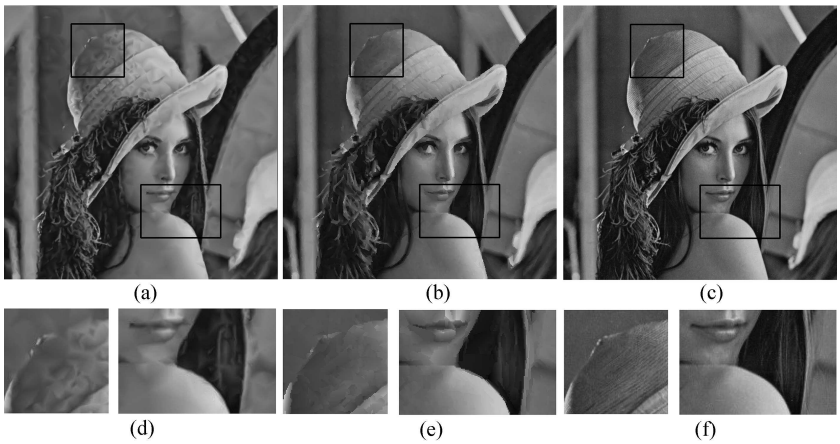

Figure 4 Detail comparison of rendered results. (a) and (b) are the rendering results of [2] and ours; (c) is the original image; (d), (e), and (f) are local details of the upper images, marked with rectangles.

Figure 5 More comparisons. Several images are tested, with 1% sample ratio. From left to right: results of [2], ours and original images. From top to bottom: fruits, peppers, flower and Elaine.

6 Conclusion

In this paper, we proposed a novel type of vectorized representation for raster images, namely the patch-based representation, in which the pixels of an input image are converted into a height field and then approximated by a specific number of planar patches with proper boundaries. Compared with traditional mesh representations whose vertices are located at the very sample points, ours could find an optimal position for the patches and thus could produce less fitting errors.

Some limitations also exist. First, while the quality of rendered images increases, the easiness of editing decreases, to some extent. Second, our patch-based representation is essentially a type of linear interpolation, which theoretically has less expressing ability than quadratic or higher interpolation, yet those schemes might be somehow time-consuming. Third, since the number of vertices cannot be well controlled in our current implementation, the fairness of comparison is not quite guarenteed.

In the future, we will try to explore other representations based on some non-linear interpolation, and we will take the easiness of editing and other factors into consideration as much as possible. Besides, seeking a way of controlling the number of anchor vertices will also be our future work.

Acknowledgements This work was supported by National High Technology Research and Development Program of China (Grant No. 2013AA013903).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Cohen-Steiner D, Alliez P, Desbrun M. Variational shape approximation. ACM Trans Graphics, 2004, 23: 905–914
2 Adams M D. A flexible content-adaptive mesh-generation strategy for image representation. IEEE Trans Image Process, 2011, 20: 2414–2427
3 Demaret L, Dyn N, Iske A. Image compression by linear splines over adaptive triangulations. Signal Process, 2006, 86: 1604–1616
4 Demaret L, Iske A, Mathematik Z, et al. Advances in digital image compression by adaptive thinning. Ann MCFA, 2004, 3: 105–109
5 Li P, Adams M D. A tuned mesh-generation strategy for image representation based on data-dependent triangulation. IEEE Trans Image Process, 2013, 22: 2004–2018
6 Yang Y Y, Wernick M N, Brankov J G. A fast approach for accurate content-adaptive mesh generation. IEEE Trans Image Process, 2003, 12: 866–881
7 Xia T, Liao B B, Yu Y Z. Patch-based image vectorization with automatic curvilinear feature alignment. ACM Trans Graphics, 2009, 28: 115
8 Lai Y-K, Hu S-M, Martin R R. Automatic and topology-preserving gradient mesh generation for image vectorization. ACM Trans Graphics, 2009, 28: 85
9 Sun J, Liang L, Wen F, et al. Image vectorization using optimized gradient meshes. ACM Trans Graphics, 2007, 26: 11
10 Jeschke S, Cline D, Wonka P. Estimating color and texture parameters for vector graphics. Comput Graphics Forum, 2011, 30: 523–532
11 Orzan A, Bousseau A, Barla P, et al. Diffusion curves: a vector representation for smooth-shaded images. Commun ACM, 2013, 56: 101–108
12 Orzan A, Bousseau A, Winnemoeller H, et al. Diffusion curves: a vector representation for smooth-shaded images. ACM Trans Graphics, 2008, 27: 101–108
13 Nagai Y, Ohtake Y, Yokota H, et al. Boundary-representable partition of unity for image magnification. Sci China Inf Sci, 2013, 56: 112106
14 Wang M, Lai Y-K, Liang Y, et al. BiggerPicture: data-driven image extrapolation using graph matching. ACM Trans Graphics, 2014, 33: 173
15 Hu S-M, Zhang F-L, Wang M, et al. PatchNet: a patch-based image representation for interactive library-driven image editing. ACM Trans Graphics, 2013, 32: 196
16 Li H, Wu W, Wu E H. Robust interactive image segmentation via graph-based manifold ranking. Comput Visual

Media, 2015, 1: 183–195

17 Chen X W, Zhou B, Guo Y, et al. Structure guided texture inpainting through multi-scale patches and global optimization for image completion. Sci China Inf Sci, 2014, 57: 012102

18 Lai Y-K, Martin R R. Vertex location optimisation for improved remeshing. Graphical Models, 2012, 74: 233–243

19 Douglas D H, Peucker T K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Int J Geogr Inf Geovisualization, 1973, 10: 112–122

20 Zhang M, Yan W, Yuan C M, et al. Curve fitting and optimal interpolation on CNC machines based on quadratic B-splines. Sci China Inf Sci, 2011, 54: 1407–1418

21 Huynh-Thu Q, Ghanbari M. Scope of validity of PSNR in image/video quality assessment. Electron Lett, 2008, 44: 800–835

22 Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process, 2004, 13: 600–612

23 Sieger D, Botsch M. Design, implementation, and evaluation of the surface_mesh data structure. In: Proceedings of the 20th International Meshing Roundtable, Paris, 2012. 533–550