Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Image meshing via hierarchical optimization*

Hao XIE, Ruo-feng TONG‡

(*Institute of Artificial Intelligence, Zhejiang University, Hangzhou 310027, China*)

E-mail: xiehao@zju.edu.cn; trf@zju.edu.cn

Received May 27, 2015; Revision accepted Sept. 16, 2015; Crosschecked Oct. 21, 2015

Abstract: Vector graphic, as a kind of geometric representation of raster images, has many advantages, e.g., definition independence and editing facility. A popular way to convert raster images into vector graphics is image meshing, the aim of which is to find a mesh to represent an image as faithfully as possible. For traditional meshing algorithms, the crux of the problem resides mainly in the high non-linearity and non-smoothness of the objective, which makes it difficult to find a desirable optimal solution. To ameliorate this situation, we present a hierarchical optimization algorithm solving the problem from coarser levels to finer ones, providing initialization for each level with its coarser ascent. To further simplify the problem, the original non-convex problem is converted to a linear least squares one, and thus becomes convex, which makes the problem much easier to solve. A dictionary learning framework is used to combine geometry and topology elegantly. Then an alternating scheme is employed to solve both parts. Experiments show that our algorithm runs fast and achieves better results than existing ones for most images.

Key words: Image meshing, Hierarchical optimization, Convexification
http://dx.doi.org/10.1631/FITEE.1500171 **CLC number:** TP391.7

1 Introduction

How to represent an image is an old yet still open topic. Currently, two categories, namely raster image and vector graphic, serve in our daily life. A raster image basically stores color values for all pixels (Compressed formats such as JPG and PNG are not considered for simplicity), and vector graphic is a type of geometric representation. Generally, the predominance of vector graphics is in part due to facility for editing, independence on display resolution, and usually volume reduction of storage, etc. Thus, vector graphic is a potential tool for representing images.

Image meshing, as a class of methods for converting raster images into vector graphics, is a popular branch of 'image vectorization'. With image meshing, many further applications, e.g., image wrapping, image composition, and object editing, become facilitated. The main process of image meshing is to find a (usually triangular) mesh covering the whole input image (This means that no pixel in the image would be left), with each vertex given a color value defined as its located pixel in the input image. Then, color values of other positions on the mesh can be expressed by the vertex colors mentioned above (mostly through linear interpolation). In this way, the input image can be rendered totally by the mesh. Naturally, the goal of image meshing is to find such an optimal mesh, to minimize the difference between the input image and its rendered counterpart as much as possible.

To obtain this optimal mesh, various methods have been proposed, most of which employ the Euclidean distance in the color space to measure the difference. In this paper, a hierarchical optimization method is presented. An example of inputs and outputs is illustrated in Fig. 1. The most difficult

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61170141) and the National High-Tech R&D Program (863) of China (No. 2013AA013903)

ORCID: Hao XIE, http://orcid.org/0000-0003-0270-2703

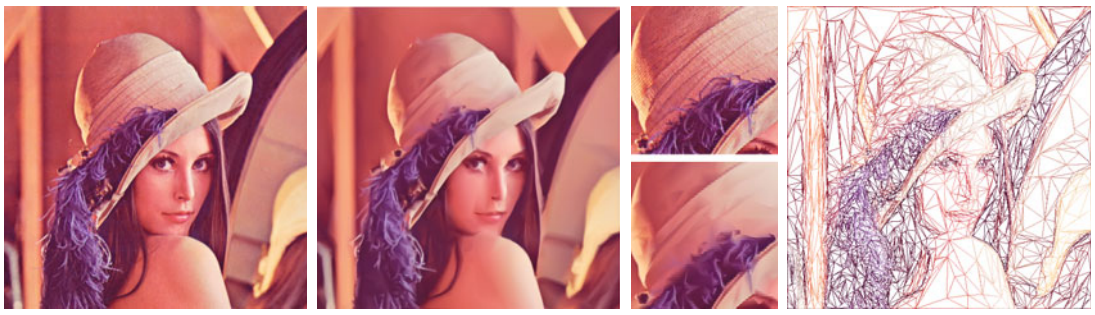

Fig. 1 Inputs and outputs. Given an image, 'Lena' (a), and the sampling ratio (1% in this example), our algorithm generates a mesh-based vector graphic (d) with its rendered result (b). Detail regions of both (a) and (b) are enlarged and shown in (c), in which upper and lower parts correspond to (a) and (b), respectively. References to color refer to the online version of this figure

point here is not the modeling of the problem, but the solution. As will be shown later, these objectives are usually highly non-linear and non-smooth, and direct use of conventional routines in numerical optimization methods may not achieve the desired results, or can hardly make the variables move even a tiny step in the solution space. In numerical optimization, a global optimal solution to a non-convex problem can rarely be obtained. Generally, initialization plays a key role in influencing the performance of non-convex optimization. Apart from the property of the problem itself, which can hardly be improved, a proper initialization of variables is much easier to achieve. To achieve a better initialization, a hierarchical manner is proposed in this paper. As a preprocessing step, the input image is filtered into several levels, ranging from coarse to fine, by, e.g., the bilateral filter, which preserves meaningful features. In this way, the coarsest level becomes much smoother, and thus may be easier to solve than the original one. Then, the result of a coarser level is recycled and employed as the initial value of its finer descendant. As a result, the performance of the optimizer for the finest level, i.e., the original image, is improved.

In addition to a hierarchical process, the objective function at the stage of geometry optimization is still non-convex. Generally, without using 'greedy' schemes, the remaining choices of global optimization are mostly quasi-Newton methods, e.g., L-BFGS (Liu and Nocedal, 1989), and the result is still highly dependent on the initial values. To further reduce the complexity of the problem, a conversion from a non-convex problem to a convex one is proposed. Since the color space is also a metric space, it can be considered together with the image space, which thus forms a new 5D metric space. In this way, a pixel in the image, as well as its attached color, is mapped into a 5D point in the newly constructed metric space. Therefore, the original non-convex problem is converted into a simple linear least squares one, and can be efficiently solved via some common routines, independent of the initialization. The running time can also be reduced.

The contributions of this paper are summarized as follows:

1. A hierarchical process is proposed, which provides an initial input for a finer level with the result of a coarser level to make the original problem smoother.

2. By combining the color space and image space, a conversion from a non-convex problem to a convex one is proposed to further reduce the complexity of the problem at the stage of geometry optimization.

2 Related work

Recently, many algorithms have been proposed on image meshing. The most representative one, proposed by Adams (2011), is a type of greedy algorithm. The key idea is to simply eliminate redundant vertices iteratively from an initial mesh with a denser distribution of vertices. In each loop, the vertex with the least influence on the result of the whole system was removed, and the edges were rearranged; i.e., vertices were re-connected, based on Delaunay triangulation. To accelerate the algorithm, Yang *et al.*

(2003) employed an error diffusion scheme to generate an initial mesh, instead of the initial mesh used in an adaptive thinning method (Demaret and Iske, 2004), consisting of vertices at all pixels. Although this algorithm is fast, it does not involve the quality of the recovered image when rearranging edges (simply because of using Delaunay triangulation), and thus the quality of the generated mesh is not sufficient.

As an improvement, Xie *et al.* (2014) took the movement of existing vertices into account, and got rid of the constraints of Delaunay triangulation. In Xie *et al.* (2014), an initial mesh was optimized by performing geometry and topology optimization alternately. Instead of processing a vertex only once as in Adams (2011), Xie *et al.* (2014) re-adjusted the position of a vertex by a reasonable number of times, so that the local optimality of that vertex would not be corrupted by later processing of other vertices.

Another type of algorithm involves Xia *et al.* (2009) and Liao *et al.* (2012): the authors considered the generated mesh as a surface mesh, with a third coordinate of a vertex, i.e., the color of its located pixel. From this viewpoint, an image can be processed by use of some geometry tools. Note that the three-channel colors are processed separately and consistently. As a result, three meshes with the same connectivity are generated for one given image. Although this conversion gives a good result, the reasons are not clearly stated in either study.

Xia *et al.* (2009) first constructed an initial dense surface mesh, with some crack holes corresponding to the edge features of the input image, then performed a mesh simplification while maintaining the main edge features of the image, and finally converted it into a Bézier triangular mesh. This algorithm employs a thin-plate spline (TPS) to interpolate colors on the mesh, generating a better result. The shortcomings of this algorithm involve the complex representation of the mesh, and the non-adaptive sampling when calculating the coefficients of the TPS.

Rather than a Bézier triangular mesh, Liao *et al.* (2012) employed piecewise smooth subdivision surfaces. The initial mesh was the same as the one in Xia *et al.* (2009), and in the stage of mesh simplification, edge features were maintained at different levels to form a multi-resolution mesh representation. Using these features, one can edit the generated mesh easily, with manipulations such as object deformation and feature enhancement. Although this representation is flexible, it is still somewhat complicated (edge features need to be stored). In addition, there is no standard criterion for the quality of the recovered image for either algorithm proposed by Xia *et al.* (2009) or Liao *et al.* (2012). The difference from our algorithm is that these two methods are based mainly on non-global mesh operations, while in ours, a global scheme is employed.

Still, there are many other image meshing methods, such as Demaret *et al.* (2006), Lecot and Levy (2006), and Swaminarayan and Prasad (2006). These methods assume that the color variation in each region is relatively plain, which can also be considered as being generated by linear interpolation. In another type of algorithm, Sun *et al.* (2007) and Lai *et al.* (2009) used gradient mesh to represent images. Their results are good, yet the algorithms depend heavily on image segmentation, as they can deal only with isolated objects with smooth color variance.

3 Objective

Given an input raster image $\boldsymbol{I}$, our goal is to generate a triangular mesh $\boldsymbol{M}=\{\boldsymbol{V},\boldsymbol{K}\}$ covering the image and representing it as faithfully as possible, where geometry $\boldsymbol{V}$ and topology $\boldsymbol{K}$ are described as follows.

For geometry information, let $\boldsymbol{v}_i\in\mathbb{R}^2$ $(i=1,2,\dots,n_{\mathrm{v}})$ denote a vertex of the triangular mesh, where n_{v} is the number of vertices, and $\boldsymbol{V}\in\mathbb{R}^{2\times n_{\mathrm{v}}}$, a matrix whose ith column represents vertex $\boldsymbol{v}_i$, denotes the set of all vertices on mesh $\boldsymbol{M}$. To recover image $\boldsymbol{I}$, each vertex is attached with a color value, and the colors of other positions on the mesh are linearly interpolated by neighboring vertices, namely all vertices of its belonging triangle. Inspired by Xiong *et al.* (2014), a dictionary learning framework is introduced for modeling both geometry and topology information.

A sparse matrix $\boldsymbol{K}\in\mathbb{R}^{n_{\mathrm{v}}\times n_{\mathrm{p}}}$ is constructed to denote the connectivity of vertices, whose column $\boldsymbol{\kappa}_j$ denotes the barycentric coordinates of pixel $\boldsymbol{p}_j\in\mathbb{R}^2$ $(j=1,2,\dots,n_{\mathrm{p}})$, where n_{p} is the number of sampling points, i.e., the number of pixels. Note that the sparse matrix $\boldsymbol{K}$ should be constructed such that the mesh can form a manifold. That is, each column of $\boldsymbol{K}$ has at most three non-zero elements, corresponding to up to three vertices of a triangular

face, and the summation of the magnitudes of these elements should be one. Thus, our goal is modeled as follows:

$$\begin{aligned}&(\boldsymbol{V}^*,\boldsymbol{K}^*)=\arg\min_{\boldsymbol{V},\boldsymbol{K}} E(\boldsymbol{V},\boldsymbol{K})\\&\text{s.t. } \|\boldsymbol{\kappa}_i\|_0\le 3,\ \|\boldsymbol{\kappa}_i\|_1=1,\ i=1,2,\ldots,n_{\mathrm{p}},\end{aligned}\tag{1}$$

where $E(\boldsymbol{V},\boldsymbol{K})$ measures the fitting error of the recovered image.

Let $\mathcal{I}\in\{f:\mathbb{R}^2\mapsto\mathbb{R}^3\}$. Then $\mathcal{I}(\boldsymbol{p})$ denotes the color value attached to pixel $\boldsymbol{p}$, and $E(\boldsymbol{V},\boldsymbol{K})$ is thus expressed as

$$\begin{aligned}E(\boldsymbol{V},\boldsymbol{K})&=\frac{1}{n_{\mathrm{p}}}\|\mathcal{I}(\boldsymbol{P})-\mathcal{I}(\boldsymbol{V})\boldsymbol{K}\|_{\mathrm{F}}^2\\&=\frac{1}{n_{\mathrm{p}}}\sum_i\|\mathcal{I}(\boldsymbol{p}_i)-\mathcal{I}(\boldsymbol{V})\boldsymbol{\kappa}_i\|_2^2,\end{aligned}\tag{2}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, $\boldsymbol{P}\in\mathbb{R}^{2\times n_{\mathrm{p}}}$ is the set of all pixels, whose jth column denotes the jth pixel $\boldsymbol{p}_j$, and $\mathcal{I}(\boldsymbol{P})=\{\mathcal{I}(\boldsymbol{p}_i)\}_{i=1}^{n_{\mathrm{p}}}\in\mathbb{R}^{3\times n_{\mathrm{p}}}$, $\mathcal{I}(\boldsymbol{V})=\{\mathcal{I}(\boldsymbol{v}_i)\}_{i=1}^{n_{\mathrm{v}}}\in\mathbb{R}^{3\times n_{\mathrm{v}}}$.

Note that the sparse matrix $\boldsymbol{K}$ mentioned above is constructed to elegantly combine the geometry and topology information in Eq. (2). In this way, the connectivity of vertices is encoded into a matrix, which makes the objective easy to express. Also, note that the model in this study is actually linear, while a more general higher order model might generate a better visual effect. However, a higher order model can make the whole process more complicated, and entails more running time. Thus, a simple linear model is employed.

4 Hierarchical optimization

4.1 Overview

Once the problem has been modeled, the next task is to solve it. As shown in Algorithm 1, the pipeline simply involves several steps. As a preprocessing step, a hierarchy with several levels is established for the input image. An initial mesh is then generated, followed by the whole iterative process for all levels. At each level, a geometry and topology alternating optimization is performed. To further simplify the problem, a convexification of the objective function is performed beforehand. Some details are discussed in the following paragraphs.

Algorithm 1 Pipeline of our algorithm

```
HierarchyEstablishment
MeshInitialization
i ⇐ N − 1
while Level i ≥ 0 do
    ObjectiveConvexification(i)
    AlternatingOptimization(i)
    i ⇐ i − 1
end while
```

4.2 Hierarchy establishment

As shown in Eq. (1), the objective function is highly non-linear and non-smooth, and thus direct use of common routines of optimization algorithms, e.g., L-BFGS, is not quite suitable in this context. Since the body of the algorithm itself can hardly be improved for such a non-convex problem, another path should be sought. For non-convex optimization, a proper initialization is significant. For this, a hierarchical process is proposed. Specifically, the input image is filtered into N levels beforehand. Intuitively, the coarser level is smoother than the finer one, and is supposed to have better performance through the same algorithm. Therefore, as illustrated in Fig. 2, the whole process starts from the coarsest level (level $N-1$), travels through all other levels of images, ranging from coarser to finer, and finally reaches the original input image (the finest level, level 0). When dealing with level i ($i<N-1$), the result from level $i+1$ is used as an initial value.

As for the choice of the filter, our goal is to make the problem smoother, while preserving some necessary features in the image. Thus, general smoothing filters, e.g., the Gaussian filter, are not quite suitable. Instead, filters preserving features are preferable. In our algorithm, the simple bilateral filter is chosen. In fact, other filters, such as the filter proposed in

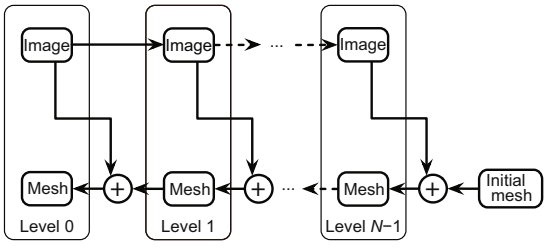

Fig. 2 Hierarchy establishment: a hierarchy with N levels, from coarser (higher level) to finer (lower level), is established. At each level, an image of this level and a mesh from the last level are employed to generate a mesh of this level. Note that the mesh of level 0 is the desired final mesh

Xu *et al.* (2011), might have a better result, yet are somewhat time-consuming compared with the bilateral filter.

4.3 Mesh initialization

Before the main process of various levels, an initial mesh for the coarsest level, i.e., level $N-1$, needs to be generated properly. Although our algorithm has convexified the problem at the stage of geometry optimization, the situation at the stage of topology optimization is not improved, and the whole problem is still somewhat complex. Thus, a totally random initial mesh might generate a undesirable result, and involve a longer running time. Therefore, to achieve better performance and effect, the results generated by Adams (2011) are employed as the very beginning initialization in our algorithm.

4.4 Objective convexification

To further simplify the problem, a convexification of the objective is performed. Specifically, the position of each vertex $\boldsymbol{v}_i \in \mathbb{R}^2$ on mesh $\boldsymbol{M}$, together with its attached color value $\mathcal{I}(\boldsymbol{v}_i) \in \mathbb{R}^3$, is mapped into a point $\hat{\boldsymbol{v}}_i = (\boldsymbol{v}_i^{\mathrm{T}}, \mathcal{I}(\boldsymbol{v}_i)^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^5$. Thus, the planar mesh becomes a 2-manifold in a 5-dimensional (5D) space. Similarly, all pixels of the input image $\boldsymbol{I}$ are converted into such 5D points. In this way, the original non-convex problem is converted into a convex one. Thus, Eq. (2) is converted as follows:

$$E(\hat{\boldsymbol{V}}) = \frac{1}{n_{\mathrm{p}}} \|\hat{\boldsymbol{P}} - \hat{\boldsymbol{V}}\boldsymbol{K}\|_{\mathrm{F}}^2 = \frac{1}{n_{\mathrm{p}}} \sum_i \|\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{V}}\boldsymbol{\kappa}_i\|_2^2, \tag{3}$$

where $\hat{\boldsymbol{p}}_i$ is the ith column of $\hat{\boldsymbol{P}} \in \mathbb{R}^{5\times n_{\mathrm{p}}}$, and $\hat{\boldsymbol{v}}_i$ is the ith column of $\hat{\boldsymbol{V}} \in \mathbb{R}^{5\times n_{\mathrm{v}}}$.

The rationale for this conversion is explained as follows. In the original problem, the metrics in the image space and color space are both strongly related to the Euclidian distances. Therefore, the combination of these two spaces is possible in form. On the other hand, it makes sense in geometry. In fact, this is a type of approximation to the original problem. Take a 1D image with one channel as an example (Fig. 3). Fig. 3a shows the color distance in the original problem, where the segments (red line) connecting the recovered image (slope straight line) and original one (dashed curve) are orthogonal to the image plane (the horizontal line below); in contrast, in Fig. 3b, this line is orthogonal to the triangular face of the mesh. When the distances become small enough, these two forms are supposed to generate similar effects. Based on the above explanation, this convexification is proposed and employed in our model.

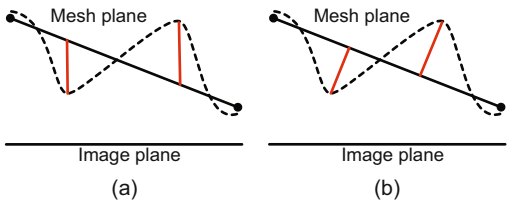

Fig. 3 Image meshing in a 1D image: the original problem (a) and that after convexification (b). References to color refer to the online version of this figure

4.5 Alternating optimization

After the preprocessing, the solution for each level can be solved via common routines. In general, to solve this problem, geometry and topology aspects are usually treated separately and alternately in each iteration, as in Xie *et al.* (2014). Both sub-optimization stages are discussed below.

4.5.1 Geometry optimization

In this stage, the topology $\boldsymbol{K}$ of the mesh, i.e., connection of vertices, is fixed, and the only variable concerned is the position matrix of vertices, $\hat{\boldsymbol{V}}$. Since the original problem has been converted into a simple linear least squares one (Section 4.4), the preconditioned linear conjugate gradient method is employed to solve $\hat{\boldsymbol{V}}$. Note that the rows of $\hat{\boldsymbol{V}}$ are solved separately using the same factorization of $\boldsymbol{K}\boldsymbol{K}^{\mathrm{T}}$, which can be done beforehand to speed up the process.

4.5.2 Topology optimization

Given a fixed geometry, i.e., positions of vertices $\hat{\boldsymbol{V}}$, the task of this stage is then to optimize the topology, i.e., the connectivity of vertices $\boldsymbol{K}$. As is known to all, topology operations involve edge flip, edge or face collapse, edge or face split, etc. Since the positions of vertices are constant in this context, which means the number of vertices is not changed, the only operator left is edge flip. To further reduce the total energy, the strategy in Xie *et al.* (2014) is employed, the process of which is described briefly below. For further details, please refer to Xie *et al.* (2014).

Before flipping the edges, a suspect set of edges

is defined, the element in which is an edge indicating that it might be flipped. At the beginning, all edges excluding boundary edges, i.e., interior edges, are inserted into the set. To sort these suspect edges, a fitting error defined as the summation of fitting errors of two adjacent faces is attached to each suspect as a criterion. Then the suspect set becomes a priority queue. Each time, a suspect is popped up and checked whether to flip or not, depending on whether the total energy declines. If a suspect is flipped, its neighbor edges are also labeled as suspects and pushed into the queue. The process ends when the queue is empty. To avoid possible endless loops, the times visited for each edge are recorded, and edges visited enough times are no longer concerned in this pass.

5 Experiments and discussion

In our experiments, all images are tested on a PC with AMD Phenom$^{\text{TM}}$ II X4 945 CPU and 8 GB RAM. All the algorithms are implemented in an Arch Linux operating system, using C++ language, with the Surface_mesh library (Sieger and Botsch, 2012) employed to represent the mesh data structure.

5.1 Measurements

The peak signal-to-noise ratio (PSNR) (Huynh-Thu and Ghanbari, 2008) and running time are used to evaluate our algorithm. Here,

$$\mathrm{PSNR} = 20 \log \frac{2^{\rho} - 1}{\sqrt{\mathrm{MSE}}},$$

where ρ is the number of bits or samples in source or target images, and MSE is the mean squared error between the original image and reconstructed image, which is the total fitting error in this context (Section 4.4). Generally, the larger the PSNR, the better the performance of fitting.

5.2 Parameters

The first parameter to be tuned is the number of hierarchy levels. In general, deeper levels might not produce better performance. To verify this, several images are tested via various levels. In our experiments, we find that using two or three levels usually generates better results. The reason for this is that after filtering enough times, the image is smooth enough, and the influence of initialization on the optimization fades.

Other parameters are described here. In bilateral filtering, the diameter of each pixel neighborhood that is used during filtering is set to 9, and the two parameters used to measure the size of convolution kernels in the color and coordinate space, σ_{Color} and σ_{Space}, are both set to 50. As for the number of iterations in each level, we find two iterations give a trade-off between performance and speed.

5.3 Comparisons

In this section, two algorithms given in Adams (2011) and Xie *et al.* (2014) are compared with ours. The images for evaluation are selected diversely, ranging from human to natural plants and animals. The results of PSNR are listed in Table 1. As shown, our algorithm performs better than the other two for most of the test images. Note that the PSNR value obtained here using the algorithm proposed by Xie *et al.* (2014) has some tiny difference from that given in Xie *et al.* (2014), due to the implementation. In Xie *et al.* (2014), the algorithm is for single-channel images, while in this study, it is implemented for color images, for convenience of comparison.

To compare the mesh generated, the wireframes of both initial and final results are shown in Fig. 4. The mesh from Adams (2011) is a Delaunay triangular mesh, which has less ability than ours to express detailed features. Note that both meshes shown are generated by a 1% sampling rate.

To further illustrate the effect of our algorithm, some images rendered by the three algorithms, and

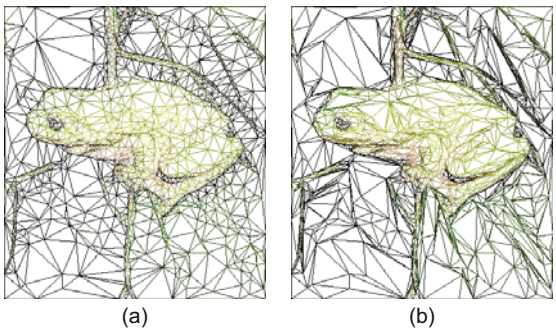

Fig. 4 Wireframes: (a) is the mesh generated from Adams (2011), employed as the initial mesh; (b) is ours. As shown, the triangles near edges are skinnier after optimization of our algorithm, to represent features better. References to color refer to the online version of this figure

Table 1 Peak signal-to-noise ratio (PSNR) comparison among our algorithm and the algorithm proposed by Adams (2011) and Xie *et al.* (2014)

Image	SR (%)	PSNR (dB)		
		Adams (2011)	Xie *et al.* (2014)	Ours
Lena	1	27.3759	28.8600	**29.7244**
	2	29.8263	30.9544	**31.3691**
	3	30.9801	31.9918	**32.1422**
Fruits	1	25.9269	26.9589	**27.7363**
	2	28.5327	29.4424	**29.7518**
	3	29.9629	**30.8024**	30.5960
Flower	1	25.8028	26.9718	**27.7477**
	2	27.9217	29.6071	**29.9863**
	3	28.6615	30.8009	**30.9495**
Falcon	1	27.3196	29.9666	**30.4886**
	2	28.5323	31.9729	**32.1160**
	3	28.8675	**32.7877**	32.7106
Car	1	23.1645	24.6478	**25.0299**
	2	25.6941	26.8857	**27.2032**
	3	26.8163	27.9490	**28.0324**
Fish	1	23.9873	25.2301	**25.9045**
	2	26.6296	27.7636	**28.1316**
	3	28.1074	29.2166	**29.2648**
Leaf	1	26.8061	27.8816	**28.6272**
	2	28.8395	29.9846	**30.3554**
	3	29.5924	31.0008	**31.1287**
Swan	1	24.4826	25.9614	**26.7547**
	2	26.9890	28.3783	**28.9821**
	3	28.3651	29.7019	**30.0601**
Frog	1	27.4404	28.9776	**29.8903**
	2	29.8848	31.0584	**31.5725**
	3	30.8791	31.9689	**32.2853**

SR: sampling rate

the original ones, are shown in Fig. 5 (see p.39), with some details highlighted and compared.

As demonstrated in Fig. 5, our algorithm has a better visual effect than the other two. For instance, the mouth of the frog in the first image is recovered much better, which can be seen in the enlarged regions in the second row. The competitiveness of our algorithm can also be seen in the edge of the leaf.

Among many factors, the main reason why our algorithm performs better is the choice of the initial value. A good initial value makes the problem much smoother, and thus makes it easier to reach a better local optimality. In addition, our algorithm converts a non-linear problem into a linear least squares one at the stage of geometry optimization, and such convexification of the original problem speeds up the whole process.

The running time of our algorithm is listed in Table 2. Although not in real time, it is already acceptable for daily use. Generally, a good initialization leads to a shorter running time, since it locates near the local optimum in the solution space.

Table 2 Runtime of our algorithm

Image	Resolution	SR (%)	Time (s)	Image	Resolution	SR (%)	Time (s)
Lena	512×512	1	2.03	Fish	320×400	1	0.66
		2	3.66			2	1.05
		3	6.23			3	1.66
Fruits	512×480	1	1.75	Leaf	373×400	1	0.83
		2	3.10			2	1.36
		3	5.19			3	2.20
Flower	400×300	1	0.66	Swan	286×400	1	0.60
		2	1.02			2	0.95
		3	1.61			3	1.48
Falcon	400×280	1	0.59	Frog	357×400	1	0.90
		2	0.86			2	1.44
		3	1.33			3	2.27
Car	400×300	1	0.64				
		2	1.04				
		3	1.61				

SR: sampling rate

5.4 Limitations

Our algorithm has some flaws. For instance, it is not quite suitable for images with complicated textures. Fig. 6 shows such an example of a failure case. The textures in the water and feather are not well recovered. Since the initial meshes are obtained by bilateral filtering, the results from our algorithm lack some details of complicated textures, while the main features are preserved.

Fig. 6 A failure case: (a) is the result of swan by our algorithm at a 1% sampling rate; (b) is the original image. References to color refer to the online version of this figure

6 Conclusions

We present a novel algorithm to convert a raster image into a vector graphic represented by a

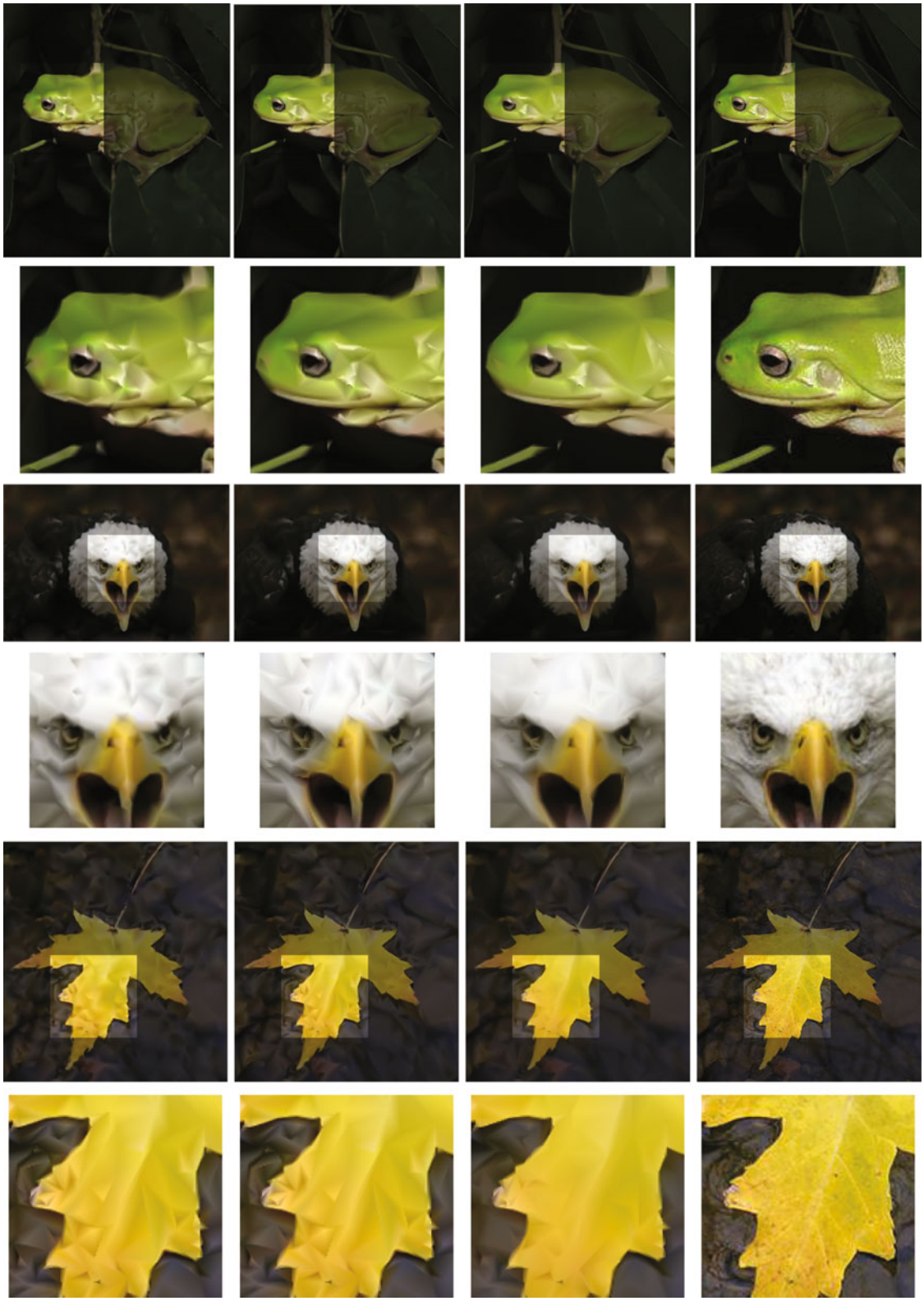

Fig. 5 Detailed comparison: the odd rows show the recovered images with original sizes, while the even rows enlarge their highlighted regions. From left to right: results obtained by the algorithms proposed by Adams (2011), Xie *et al.* (2014), and this study, and the original raster images. References to color refer to the online version of this figure

triangular mesh. Compared with other algorithms, the results are improved by establishing a hierarchy of several levels for the input image. In addition, a convexification of the problem in the stage of geometry optimization is performed to further simplify the non-linear problem into a linear least squares one, which can be solved easily via common optimizers.

In the future, we will seek a way to obtain a better initial value. For this, the PatchNet technique (Hu *et al.*, 2013) might be a favorable choice, for our algorithm to achieve a better local optimum. Meanwhile, we will consider using a more general non-linear model to represent images. We believe such a model would render results with better quality.

References

Adams, M.D., 2011. A flexible content-adaptive mesh-generation strategy for image representation. *IEEE Trans. Image Process.*, **20**(9):2414-2427. http://dx.doi.org/10.1109/TIP.2011.2128336

Demaret, L., Iske, A., 2004. Advances in digital image compression by adaptive thinning. *Ann. MCFA*, **3**:105-109.

Demaret, L., Dyn, N., Iske, A., 2006. Image compression by linear splines over adaptive triangulations. *Signal Process.*, **86**(7):1604-1616. http://dx.doi.org/10.1016/j.sigpro.2005.09.003

Hu, S.M., Zhang, F.L., Wang, M., *et al.*, 2013. PatchNet: a patch-based image representation for interactive library-driven image editing. *ACM Trans. Graph.*, **32**(6):196. http://dx.doi.org/10.1145/2508363.2508381

Huynh-Thu, Q., Ghanbari, M., 2008. Scope of validity of PSNR in image/video quality assessment. *Electron. Lett.*, **44**(13):800-801. http://dx.doi.org/10.1049/el:20080522

Lai, Y.K., Hu, S.M., Martin, R.R., 2009. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Trans. Graph.*, **28**(3):85. http://dx.doi.org/10.1145/1531326.1531391

Lecot, G., Levy, B., 2006. Ardeco: automatic region detection and conversion. 17th Eurographics Symp. on Rendering, p.349-360. http://dx.doi.org/10.2312/EGWR/EGSR06/349-360

Liao, Z.C., Hoppe, H., Forsyth, D., *et al.*, 2012. A subdivision-based representation for vector image editing. *IEEE Trans. Vis. Comput. Graph.*, **18**(11):1858-1867. http://dx.doi.org/10.1109/TVCG.2012.76

Liu, D.C., Nocedal, J., 1989. On the limited memory BFGS method for large-scale optimization. *Math. Program.*, **45**(3):503-528. http://dx.doi.org/10.1007/BF01589116

Sieger, D., Botsch, M., 2012. Design, implementation, and evaluation of the surface_mesh data structure. Proc. 20th Int. Meshing Roundtable, p.533-550. http://dx.doi.org/10.1007/978-3-642-24734-7_29

Sun, J., Liang, L., Wen, F., *et al.*, 2007. Image vectorization using optimized gradient meshes. *ACM Trans. Graph.*, **26**(3):11. http://dx.doi.org/10.1145/1239451.1239462

Swaminarayan, S., Prasad, L., 2006. Rapid automated polygonal image decomposition. 35th IEEE Applied Imagery and Pattern Recognition Workshop, p.28-33. http://dx.doi.org/10.1109/AIPR.2006.30

Xia, T., Liao, B.B., Yu, Y.Z., 2009. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans. Graph.*, **28**(5):115. http://dx.doi.org/10.1145/1618452.1618461

Xie, H., Tong, R.F., Zhang, Y., 2014. Image meshing via alternative optimization. *J. Comput. Inform. Syst.*, **10**(19):8209-8217. http://dx.doi.org/10.12733/jcis11723

Xiong, S.Y., Zhang, J.Y., Zheng, J.M., *et al.*, 2014. Robust surface reconstruction via dictionary learning. *ACM Trans. Graph.*, **33**(6). http://dx.doi.org/10.1145/2661229.2661263

Xu, L., Lu, C.W., Xu, Y., *et al.*, 2011. Image smoothing via L_0 gradient minimization. *ACM Trans. Graph.*, **30**(6):174. http://dx.doi.org/10.1145/2024156.2024208

Yang, Y.Y., Wernick, M.N., Brankov, J.G., 2003. A fast approach for accurate content-adaptive mesh generation. *IEEE Trans. Image Process.*, **12**(8):866-881. http://dx.doi.org/10.1109/TIP.2003.812757