Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

Field-aligned Quadrangulation for Image Vectorization

Guangshun Wei[1], Yuanfeng Zhou[†1], Xifeng Gao[2], Qian Ma[1], Shiqing Xin[1], Ying He[3]

[1]Shandong University [2]Florida State University [3]Nanyang Technological University

Abstract

Image vectorization is an important yet challenging problem, especially when the input image has rich content. In this paper, we develop a novel method for automatically vectorizing natural images with feature-aligned quad-dominant meshes. Inspired by the quadrangulation methods in 3D geometry processing, we propose a new directional field optimization technique by encoding the color gradients, sidestepping the explicit computing of salient image features. We further compute the anisotropic scales of the directional field by accommodating the distance among image features. Our method is fully automatic and efficient, which takes only a few seconds for a 400×400 image on a normal laptop. We demonstrate the effectiveness of the proposed method on various image editing applications.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Image processing

1. Introduction

Vector graphics, which is a collection of simple geometric elements with possibly color encoded, has several preferred benefits over a raster image, such as geometric editability, resolution-independence, compact representation, and ease of reuse. It is a default image representation in many applications such as color editing, image embedding, image compression and reconstruction.

Vector image can be represented as simple polygons and other shapes with points, lines, and curves which are easy for practitioners to create and modify, and can be also as complicated as gradient meshes [SLWS07, LHM09] and diffusion curves [STZ14, HSF*17]. Gradient meshes and diffusion curves gain some popularity in recent years due to their capability in achieving high-quality and photo-realistic rendering. However, they are either restricted to regular domains or inefficient to compute, limiting their usage for images with complex scenes. Mesh-based representations, such as triangle mesh [LHFY12] and quad mesh [LKSD17, XLM*18], have also been employed to represent a vector image. While the former is storage expensive, poor continuity and inconvenient for editing, the latter requires a robust quad-meshing approach that can capture the complex features in an image that is hard to achieve without introducing many irregular nodes. A shared difficulty for all the existing image vectorization methods is that they either require an image segmentation as a pre-processing step to extract features from the image or post-simplification of the geometric elements in the vector image to remove noise, where either one is a difficult problem to solve on its own.

In this paper, we present a new image vectorization method that overcomes the above limitations by developing a quad mesh generation method for images where neither feature extraction such as a full segmentation or post-processing is needed. Our method generates an image gradient-aware directional field and adapts the similar local parameterization of the instant meshing [JTPSH15] to guide the generation of quad-meshes for images. Different from state-of-the-art quadragulations [JFH*15, JTPSH15, HZN*18] that are specifically designed for geometries with features usually manually labeled, we consider an image with abundant implicit features and embed the gradient of pixels into the directional field optimization to bypass the challenges of feature extraction from images. Besides the inherited advantages from instant meshing [JTPSH15] which is fast and robust, the introduced approach produces quad meshes having high fidelity to 2D image because we utilize a new cross field optimization framework that works well for complex image features. Experiments demonstrate that our approach is effective for image vectorization and its related applications, such as vector image editing, embedding, and mosaic.

2. Related Work

Our method is based on quad mesh generation and surface parameterization methods. We review only the closely related work on quandrangulation, curve-based and mesh-based image vectorization.

Quandrangulation. A cross field is a powerful tool to guide the edge directions in quandrangulation. Specifically, [LJX*10, JTPSH15] use 4-way rotationally symmetric cross fields. The target directions are derived from the principal curvatures [CSM03, CP05], but the 2D image cannot get the principal curvatures. In this paper, the main direction of the cross field depends on the gradient of input image so as to preserve and align with the image feature. Smooth cross fields are generated by optimizing a nonlinear energy function based on periodic functions [HZ00, RVAL09] or a mixed-integer representation [RVLL08, BZK09]. However, those methods may be easily stuck in local minima. In our work, we alleviate this issue by alternating the direction field optimization and singularity cancellation.

† Corresponding author:yfzhou@sdu.edu.cn

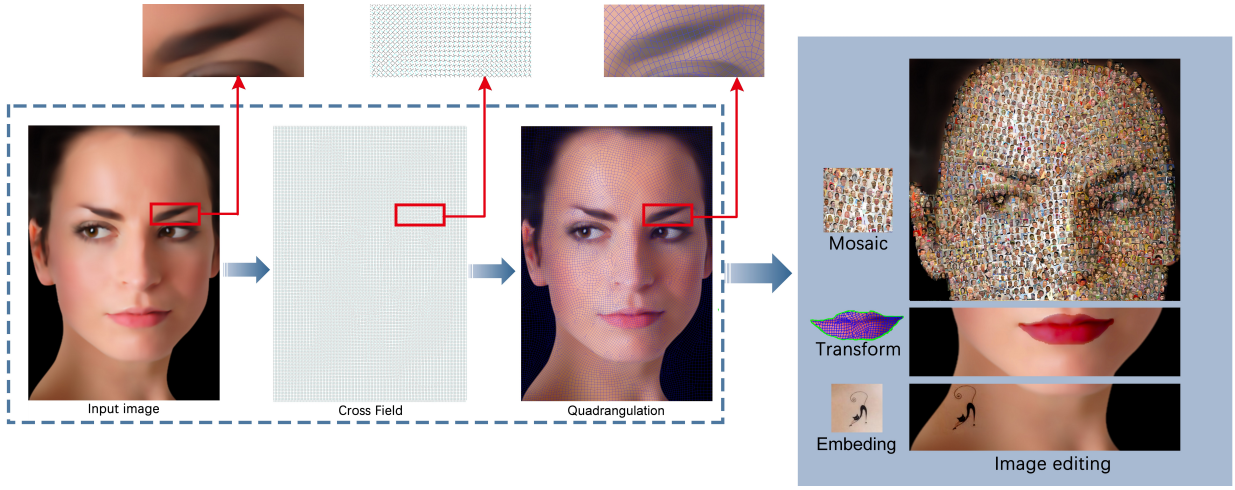


Figure 1: *The pipeline of our method.*

The parameterization can be extracted as quad-mesh using [EBCK13]. Ref. [LZ14] achieves minimum worst-case distortion and prioritizes higher distortion reduction. Global methods usually sacrifice most time to optimize the parametrization with integer constraints, which aims to enforce low distortion and reduce the number of singularities [BCE*13, MZ13, LZ14]. However, their implementations are complex, and usually not scalable. Quantized Global Optimization [CBK15] uses motorcycle graphs to quickly construct a valid quantization based on a seamless parametrization. In this paper, we extract quadrilateral vertices by optimizing the position of the vertex onto local coordinate system that can align with the image feature.

Image partition at pixel level requires each element is a small cluster of connected pixels with similar colors. The shape of elements should be regular, and sometimes be convex [DL15], which means that each element is a convex polygon. Unfortunately, partitioning an image into convex polygons is challenging to meet boundary adherence especially partitioning into quad-mesh. In particular, how to exploit geometric information disseminated into image partitioning, can be a valuable source of knowledge to analyze scenes and objects. Because images usually have complex features and boundary constraints, there is very little research work on partitioning image into quad elements.

On the other hand, many applications in computer graphics and shape modeling use models of surfaces that are composed of quadrilaterals, such as Catmull-Clark subdivision surfaces [WZHS15], mesh editing [LLL17] and physics-simulation [LHS*18]. Meshing algorithms can be classified into local and global methods. The former are usually simple, robust and scalable, but due to their locality, they tend to introduce many singularities. The most popular global approaches seamlessly parameterize the surface, regularly tessellate the parametric space, and then lift it back to 3D space. All of those methods can only process simple models without complex constraints.

Curve-based vectorization. Orzan et al. [OBW*08] use a rough version of the domain to efficiently solve the low-frequency components of the solution and use the fine version of the domain to refine the high-frequency components. But the solver is affected by flicker artifacts due to the rasterization of the curve on a discrete multi-scale pixel grid. Later, Prevost et al. [PJSH15] synthesize ray tracing methods by using intermediate triangle representations with cube patches to synthesize smooth images. Sun et al. [STZ14] proposed a fast multipole representation for diffusion curve images (DCI) random access evaluation. Their approach enables the real-time performance of rasterization and texture-mapped DCI for up to millions of curves. Hou et al. [HSF*17] present Poisson vector graphics (PVG), an extension of the popular diffusion curves, for generating smooth-shaded images. However, curve-based vectorization is complicated and inefficient for image editing.

Hybrid vectorial-rasterized image representation. Hybrid vectorial-rasterized plays a critical role in the maintenance of image representation. Pixels store a digital image as a grid of point samples [TC04] that can reconstruct a limited-bandwidth continuous 2-D source image. Ref. [HT04] processes a potentially non-planar quadrilateral directly without any splitting and interpolates attributes smoothly inside the quadrilateral on 3-D surface. Ref. [TC05] presents a fully automatic way to compute textures with customizable discontinuities and signal texture pair, starting from an original high resolution image. However, all of these methods have complex calculations and inefficient image representation.

Mesh-based image vectorization. Gradient mesh has more concise expression but more difficulty in quadrilateral mesh generation. In this paper, we will focus primarily on recent work on mesh-based image vectorization. Triangulation is the simplest representation structure and can be generated easily. So many image vectorization methods are based on triangulation. Swaminarayan and Prasad [SP06] presented an artistic abstraction method from the input image by generating triangle primitives along edges and colors are sampled from the input image. Demaret et al. [DDI06] presented an adaptive triangulation method for image and using the first-order spline to reconstruct input image details, specifically for image compression. Xia et al. [XLY09] proposed a triangle-based method with a patch-based image representation which has more color variations.

Besides triangulation, quadrilateralization is also a common representation structure for images. Lecot and Levy [LL06] divided images into a set of regions defined by cubic splines whose color gradients are given on vertices. Sun et al. [SLWS07] presented a quad mesh-based representation called gradient mesh for image vectorization. This method is semi-automatic and user interaction to generate the gradient mesh on the raster image. Lai et al. [LHM09] proposed

topology-preserving gradient meshes generated on arbitrary image regions with holes. The cubic spline surface fitting procedure is completely automatic. However, this method requires slit mapping for holes on images that would produce large parameterization distortion for images with complex features.

3. Algorithm

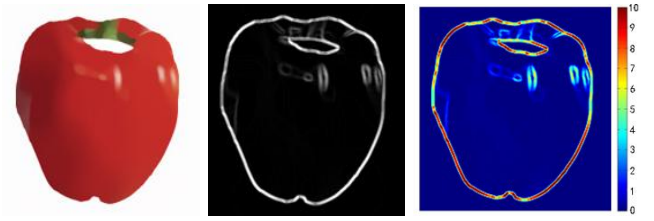
Given an image with the possibly complex scenes as the input, such as a natural image, we propose a new quadrilateral mesh generation method that can efficiently and robustly vectorize the image while capturing the features with high fidelity.

We represent the input image as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ is a pixel of the image and has a position $(i, j) \in \mathcal{R}^2$. Given two pixels adjacent to each other horizontally, vertically, or diagonally, we define an edge $e \in \mathcal{E}$ to connect them. The graph structure is used to smooth the scale, optimize the position field and extract the quadrilaterals.

As shown in Fig. 1, our method follows the standard field-aligned quad-meshing pipeline: generating an optimized directional field to guide the edge directions of the final mesh, computing a smooth parameterization to place the vertices of the mesh, and extracting the actual edge and face elements of the final mesh. However, the majority of methods following this pipeline are only for mesh inputs, rarely directly applicable to images. By representing an image with a graph, we make the technical contribution of obtaining a smoothed directional field that well captures the variation of colors of an image with as few as possible singularities in Section 3.1. For the purpose of completeness, we describe the parameterization in Section 3.2 and mesh extraction in Section 3.3.

3.1. Cross field optimization

We compute an anisotropy directional field for the graph $\mathcal{G}$. Our computation is decomposed into two steps: optimizing the directions first and then computing the scales along with the directions.



(a) Input image (b) Gradient image (c) Weight w

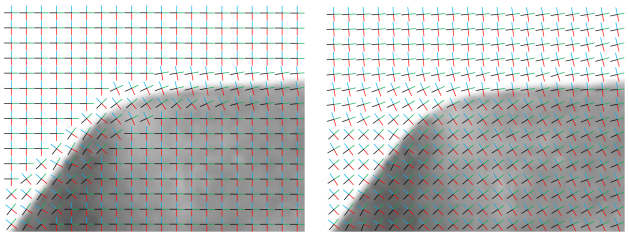
Figure 2: *The process of weights computing.*

The directions are aligned to image features without any parameter tuning by incorporating image gradient information into the optimization energy. The number of singularities is minimized through cancellation. Both the feature alignment and the singularity distributions are determined during the direction optimization, which we describe in the following.

Directions. We define a per-vertex four-way rotational symmetric cross field that has four directions with a unit length for each direction. Each cross can be represented by a unit vector up to a rotation by $\frac{k\pi}{2}$, where $k \in \{0,1,2,3\}$. We use the gradient direction computed for each vertex to initialize the cross field. For each vertex of the graph, its direction $(cos\theta, sin\theta)$ can be computed as follows:

$$\theta_{i,j} = arctan(g_{i+1,j} - g_{i-1,j}, g_{i,j+1} - g_{i,j-1}) \quad (1)$$

where g is the gray value from the input image and $\theta_{i,j} \in [-\pi, \pi]$.



(a) The initialized cross field (b) The smoothing of cross field

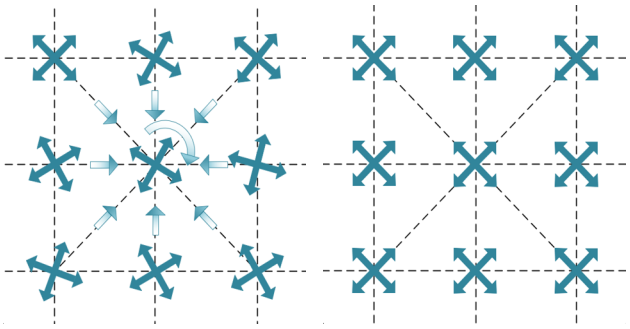
Figure 3: *Comparison before and after cross field smoothing.*

Since a natural image can have cluttered, discontinuous and numerous features, treating them as hard constraints will lead to a cross field not smooth in regions near the features and also containing too many singularities. We instead compute a weight through the image gradients and embed it into our energy:

$$\begin{aligned} p_{i,j} &= Round\left(\left(\theta_{i,j+1} - \theta_{i,j}\right)/\tfrac{\pi}{2}\right) \\ q_{i,j} &= Round\left(\left(\theta_{i+1,j} - \theta_{i,j}\right)/\tfrac{\pi}{2}\right) \end{aligned} \quad (2)$$

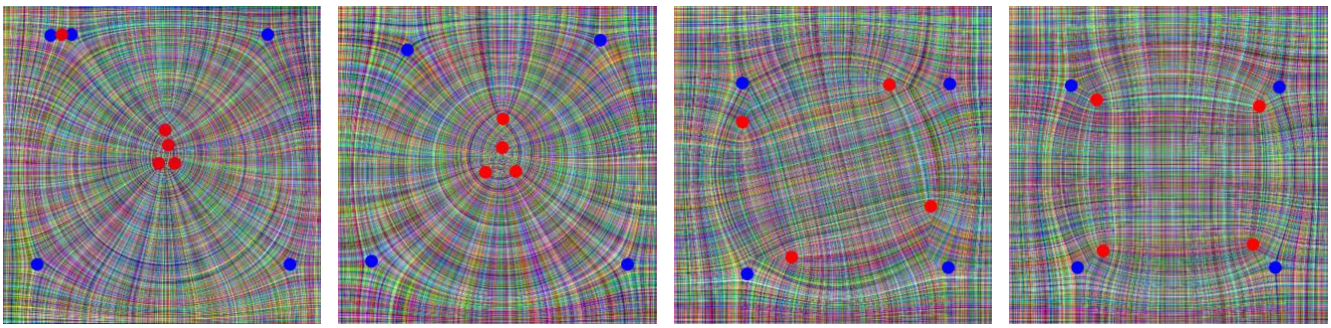
$$\begin{aligned} E_{\text{smooth}} = &\textstyle\sum_{i=1}^{m}\sum_{j=1}^{n-1} w_{i,j}\left(\theta_{i,j+1} - \theta_{i,j} - p_{i,j}\tfrac{\pi}{2}\right)^2 \\ &+\textstyle\sum_{i=1}^{m-1}\sum_{j=1}^{n} w_{i,j}\left(\theta_{i+1,j} - \theta_{i,j} - q_{i,j}\tfrac{\pi}{2}\right)^2 \end{aligned} \quad (3)$$

where the ambiguity induced by rotational symmetry is explicitly encoded using integer variables $p_{i,j}$ and $q_{i,j}$ are calculated by Eq. 2 and $p_{i,j}, q_{i,j} \in \mathcal{Z}$. The weight $w_{i,j} = e^{Gt_{i,j}}$ value is proportional to the length of the gradient as shown in Fig. 2, where $Gt_{i,j}$ is gradient value. From Fig. 3(a), we can see that the initialized cross field can align the boundary features of the image. Therefore, we set a large weight relative to the position where the gradient value is large to prevent the cross field deviating the image features during the smoothing process as shown in Fig. 3(b). What's more, a set of appropriate weights can make cross field align with features of the image and smooth cross field in the meantime. We use the cross field in the neighborhood to optimize the cross field at the center point by Eq. 3 as shown in Fig 4. By considering the cross field as a potential



(a) cross field before smoothing (b) cross field after smoothing

Figure 4: *The process of smoothing cross field.*



(a) 1 iterations (b) 3 iterations (c) 15 iterations (d) last iterations

Figure 5: *Singularity optimization. From (a) to (b), under the action of cross field, singularities move and a pair of positive and negative singularities offset each other. From (c) to (d), under the action of cross field, the singularity points continue to disperse until they reach a relatively stable position, at which time the cross field gradient around each singularity point is nearly symmetrical.*

field, the singularity point is equivalent to the charge in the potential field and the charge density can be simulated using the image gradient value. Therefore, we can decide a vertex of the graph is a positive or negative charge by:

$$I_{i,j} = p_{i,j} - p_{i+1,j} - q_{i,j} + q_{i,j+1} \tag{4}$$

If $I_{i,j} > 0$, the vertex $I_{i,j}$ is a positive charge, and if $I_{i,j} < 0$, it is negative. By calculating the potential field force at the charge, we can adjust its position. This can be achieved by modifying the integer offset between $p_{i,j}$ and $q_{i,j}$ of adjacent vertices. Fig. 5 illustrates examples of the movement of charges and the cancellation of a positive charge and a negative one by changing $p_{i,j}$ and $q_{i,j}$.

Our optimization of the direction and singularity is performed as follows. We employ an alternating optimization strategy to smooth cross field. Firstly, we compute p and q according to the initial cross field. Then we take w, p and q as constants to smooth cross field by solving the energy function defined by Eq. 3. Secondly, by fixing the directions, we remove nearby singularities by changing the values of $p_{i,j}$ and $q_{i,j}$. This process is repeated until the position and number of singular points no longer change. The pseudo-code is summarized in Algorithm. 1.

In the process of optimizing singular points, we choose to change the maximum absolute value of p and q around a singular point, which corresponds to the maximum difference between the two cross fields. In this case, the difference between the two cross field will be smaller by optimizing Eq. 3. Therefore, the singular point will move to a new location and the cross field will also be further smoothed. The singular points are characterized by their degrees (or equivalently, indices). The degree of positive and negative singular points are 3 and 5, respectively, so only singularities with opposite signs can be combined. When the gradient of cross field around all singular points is symmetrical, the singular points will be fixed. In Fig. 5, the singularities in blue and red that are separated by image boundary cannot be combined because the cross field around the boundary has a large weight. In this case, the conditions for merging the singular point are not met.

Scales. Given the smoothed cross field, we then compute the scale along each direction, so that small quadrilaterals are filled in regions with dense features and large elements are used for regions with sparse features. Firstly, we specify the number of grids m and calculate an initial length L based on m. Then we compute scales based

Algorithm 1 Smoothing cross field

Input:
The initial cross field $\theta_{i,j}$;
The integer variables $p_{i,j}$ and $q_{i,j}$;
The weight w of each $\theta_{i,j}$;
Output:
The smooth cross field $\theta_{i,j}$;
The set of singularities S;
repeat
  Updating the cross field by minimizing Eq. 3;
  Finding the singularities $S \leftarrow s$ by Eq. 4;
  while $S \neq \phi$ **do**
    Selecting $s \in S$;
    $S = S \setminus \{s\}$;
    Finding the absolute maximum p_{ij} or q_{ij} around the singular point s by calculating the Eq. 2;
    $p_{ij} \pm 1$ or $q_{ij} \pm 1$;
  end while
until The singularities (their positions and number) do not change

on a distance field where feature lines are treated as sources. The distance field function is defined as:

$$D(p) = e^{\frac{E^2(p)}{v^2}} \tag{5}$$

where $E(p)$ is the Euclidean distance from vertex p to the nearest feature edge that can be detected with Canny operator easily [Can87]. $v \in (0,2)$ is a control parameter. It determines the degree of change in the scale of the quadrilateral. The scale for the four directions at each vertex, α, can be computed by the following equation:

$$\alpha(p) = \frac{D(p) - \min_D}{\max_D - \min_D}. \tag{6}$$

We then update α by multiplying it with a target edge length for the final quad mesh. Up to now, we have computed an isotropic scale for a quadrilateral, which is not flexible enough. Imagine that, if the computed minimum scale is larger than the distance between two nearby feature lines as shown in Fig. 6, the isotropic scale will lead to a quad element not aligning well with features. We then compute an additional scale β.

Given the previously computed isotropic scale α, we then traverse all vertices on features to determine if β is necessary whether. For a

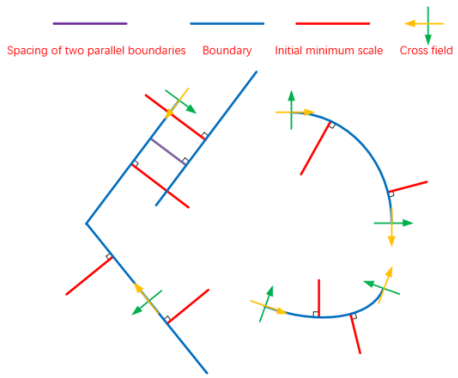


Figure 6: *Schematic diagram of optimizing scaled cross field.*

vertex on two parallel feature lines, we compute the distance between the two feature lines along the direction passing through the vertex. For example, the length of the purple line in Fig. 6. If the distance is less than $\alpha * L$ (e.g. the red line in Fig. 6), we compute scale β by Eq. 7, where D_{v_i} is the spacing of two parallel features.

$$\beta_i = D_{v_i} / (\alpha L) \quad v_i \in \mathcal{V}_{Boundary} \tag{7}$$

After re-computing the scales for all the vertices on features, we finally perform smoothing of the scales by:

$$\begin{aligned} \alpha_i &\leftarrow \sum_{j \in \mathcal{N}(i)} \alpha_j, \quad \alpha_i \leftarrow \alpha_i / N \\ \beta_i &\leftarrow \sum_{j \in \mathcal{N}(i)} \beta_j, \quad \beta_i \leftarrow \beta_i / N \end{aligned} \tag{8}$$

where N is the number of adjacent vertices.

After the smoothing of scales, $\alpha * L$ and $\beta * L$ are the final grid length for the two perpendicular directions, respectively. Smoothing of scale values can produce different values for the two directions, resulting in anisotropic elements (i.e. rectangular shaped rather than square-shaped quads). Anisotropy arises when it is required by feature alignment. In all other cases, the two scale values tend to be equal, resulting in isotropic (square-shaped) elements.

3.2. Vertex optimization

Given the optimized direction field and the scale field (α, β). We want to extract the final quad mesh whose edges align with the direction and vertex positions fall precisely on the feature lines. We modify the position optimization (local parameterization) proposed by [JTPSH15] to take into account the anisotropic scales.

Since each position p_i has two different scales(α_i, β_i), we design new energy function with scale constraints to optimize the position points along with the e_x and e_y directions respectively. As shown in Fig. 7, we establish a local coordinate system with p_i as the origin at vertex v_i. Here point p_0 and point p_1 are the neighbors of p_i, e_x and e_y represent two unit vectors perpendicular to each other, α_i and β_i are the scales of p_i. We define an integer translation of position point $u_{xj}, u_{yj} \in \mathcal{Z}$. Then our position optimization energy on the graph is as follows:

$$E_{position}(p,u) := \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}(i)} \left\| p_i - p_j - \left(u_{xj} \alpha_i L e_x + u_{yj} \beta_i L e_y\right) \right\|^2 \tag{9}$$

Given the above energy, the same optimization strategy to [JTPSH15] is employed. The optimization result is a position field that is smooth enough to obtain a high-quality quad mesh, where most of the edge lengths of the resulting mesh are close to α and β.

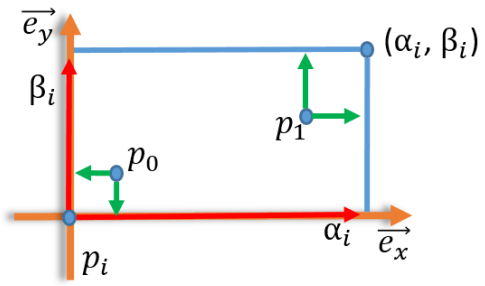
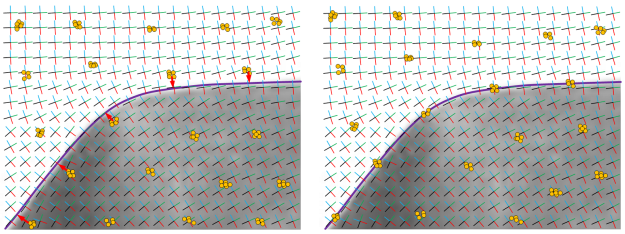


Figure 7: *The local coordinate system.*



(a) Before projection (b) After projection

Figure 8: *Projecting mesh vertices to feature curves.*

Our feature alignment is enforced by projecting vertices to points with the local maximal gradient during the optimization by

$$p_i = e_i \cdot (p_i - v_i) e_i + v_i \tag{10}$$

Fig. 8 demonstrates the effectiveness of our feature alignment, where the yellow dots represent the candidate position points, and the red arrow indicates the projecting direction of position point along the cross field's orientation.

3.3. Mesh extraction

Our quadrilateral mesh extracting process is similar to that of Jakob et al. [JTPSH15] by using a greedy searching strategy along orientations of cross field. Compare with [JTPSH15], our method can produce smoother cross field with image feature aligned. Moreover, the generated quadrilateral meshes are anisotropic under the field scales constrains, which requires considering the different scales of each vertex along with the four directions of the cross field, rather than only considering a fixed length. With these improvements, dense features in an image can be well captured (Fig. 10 right) by our method, while many of them are ignored by the original method.

4. Experimental Results and Applications

To evaluate the quality of the image vectorization by the proposed method, we show the experimental results and comparisons with the state-of-art methods. Furthermore, we give some application results. All experiments were performed on an Intel i5 core 2.50 GHz CPU laptop and 8GB RAM.

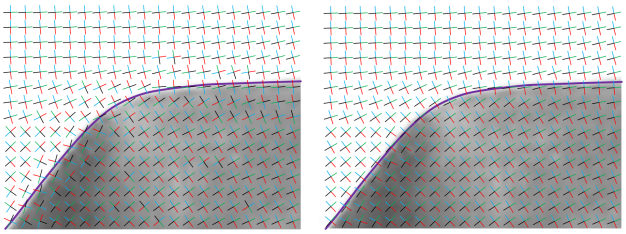
Comparisons with quadrangulation methods. There are many ways to generate meshes on the surface of a geometric model, but many methods have more or few requirements for input data. For

Table 1: *Comparison of the number of singularities.*

Image (Figure)	Instant cross field [JTPSH15]	Our method
Girl (Fig. 1)	53	36
Pepper (Fig. 2)	24	22
Plant (Fig. 10)	68	44
Amulet (Fig. 14)	331	203
Church (Fig. 14)	92	66
Boy (Fig. 14)	146	124

example, restrictions such as surface closure, smooth feature lines are required. So most methods can't be used directly to process 2D images. So we pre-extract the feature edges and consider the feature lines and the bounding box of the image as the input, and then smoothing cross field.

(a) Input image



(b) Instant cross field [JTPSH15] (c) Our cross field

Figure 9: *Comparisons of cross field after smoothing.*

From Fig. 9 and Table 1, we can see that our method can produce smoother cross field and fewer singularities than [JTPSH15] because we have designed a new method to optimize the cross field with soft constraints and merge singular points. On the other hand, [JTPSH15] propose a parameter-free alternative to the above process, which entirely sidesteps curvature-related heuristics, which directly optimizes for geometric approximation error by computing distances in the embedding space.

We compare several other state-of-the-art methods mainly in the aspect of quadrilateral mesh extracting based on 3D models [JTPSH15, JFH*15] with our quadrilateral mesh method based on 2D images. From Fig. 10, we can see that our method can produce quadrilateral with anisotropy and adaptability. So the quadrilateral generated by our method is more regular compared with [JFH*15]. Our quadrilateral has a better boundary recall compared with [JTPSH15].

Comparisons with convex image partitioning methods. Although our algorithm produces quadrilateral meshes by optimizing cross field and computing local parametrization different from the image partitioning, it can be evaluated using the standard quality criteria required for image partitioning methods. We assess the performance of our method by comparing it with CONPOLY [DL15] and N-Cuts [SM00] which are image over-segmentation methods as shown in Fig. 11. Image can be divided into a certain number of

Result of [JTPSH15] Result of [JFH*15] Our method

Figure 10: *Automatic quad-mesh generation.*

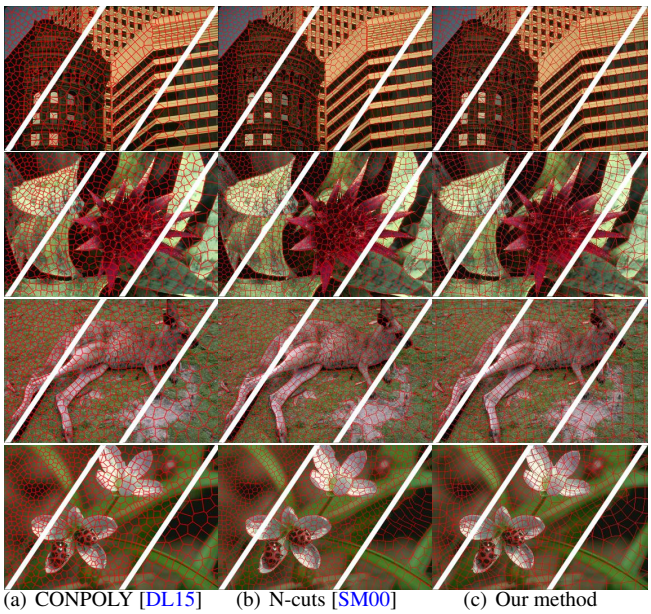
cells with the color similarity. We call these cells are superpixels. We perform our experiments on the Berkeley Segmentation database (BSDS500) [SM00], which contains 500 images with $481 * 321$ (or $321 * 481$) pixel resolution.

We compare our method with CONPOLY [DL15] and N-Cuts [SM00] in terms of boundary adherence quality criteria: boundary recall (*BR*) [LSK*09]. *BR* measures the consistency of superpixel boundaries and the ground truth, which is computed by calculating the fraction of the ground truth within a small disk-shaped neighborhood of superpixel boundaries. In our experiments, the radius of the disk is set to 2 pixels.

Fig. 12 shows the results on boundary recall statics. It is demonstrated that our method achieves better boundary recall when the number of quad-mesh are better than CONPOLY [DL15] and N-Cuts [SM00]. Because our algorithm produces quadrilateral meshes, which are closer to a squared shape than other methods. So that our method can improve boundary recall while maintaining the higher compactness.

4.1. Image vectorization

Given the quad-dominant mesh representation of an image, we treat each quad element as a Ferguson patch, where the Bessel interpolation method [FH00] is used to estimate the first partial derivatives. As adjacent patches share vertices and edge, patch curve boundaries have G^1 continuity and colors have C^1 continuity. Finally, given the values and derivatives at the patch corners, positions and colors can be readily evaluated inside the patch using bicubic interpolation. From Fig. 13(b), we can see that the quad mesh generated by our method can preserve image boundary and the most edge of quad-mesh can align the image gradient direction. Fig. 13(c) is the result



(a) CONPOLY [DL15] (b) N-cuts [SM00] (c) Our method

Figure 11: *Qualitative comparisons between our algorithm and CONPOLY [DL15], N-Cuts [SM00]. The number of superpixels in the three halves of each image are roughly 1500 (on the top left corner), 900(on the middle) and 300 (on the bottom right corner) respectively.*

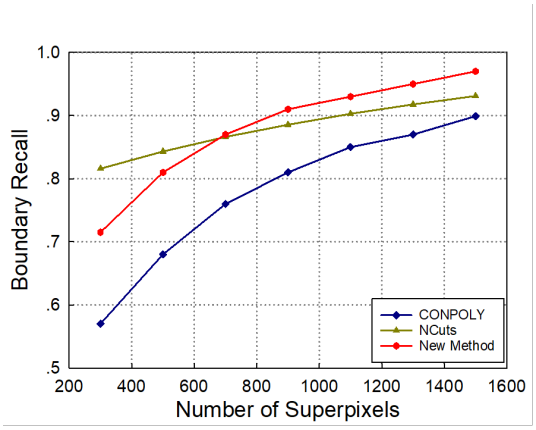
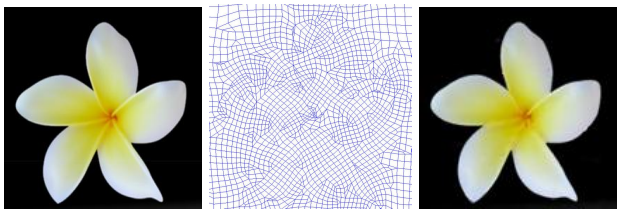


Figure 12: *Evaluating boundary adherence of two popular superpixel algorithms and our method on the BSDS500 dataset.*

of image reconstruction, and it takes 0.41 seconds for a typical mesh with 1378 patches in the resolution 258*262.

We compare our method with Lai et al. [LHM09] for image vectorization. Lai et al. [LHM09] proposed automatic gradient mesh generation for image vectorization with holes, but this method requires slit mapping for holes. Much more significant, our method performs more efficient and fully automatic. We can produce ideal representations both for simple images and natural images. Lai's method processed the Amulet image with 154.4 seconds and processed the Pepper image with 17 seconds. From the reconstruction color error comparison, Lai's method has computed average color error per pixel, in which the Amulet image is 2.76 and the Pepper image is 1.25. Our algorithm are 1.44 and 1.26, respectively. We can



(a) Input image (b) Quad mesh (c) Image reconstruction from quad mesh

Figure 13: *Automatic quad-mesh generation and image reconstruction.*

Table 2: *Timing statistics. Running times are measured in seconds on a laptop with an Intel i5 core 2.50 GHz CPU and 8GB RAM.*

Image (Figure)	Resolution	Quad-num	Time(s)	Error
Flower (Fig. 13)	258*262	1378	4.2	0.98
Amulet(our) (Fig. 14)	508*457	4673	18.6	1.44
Amulet(Lai) (Fig. 14)	508*457	4000	154.4	2.76
Pepper(our) (Fig.14)	256*270	956	4.4	1.26
Pepper(Lai) (Fig.14)	256*270	600	17	1.25
Church (Fig. 14)	481*321	2709	8.9	2.48
Boy (Fig. 14)	321*481	7635	9.2	1.7

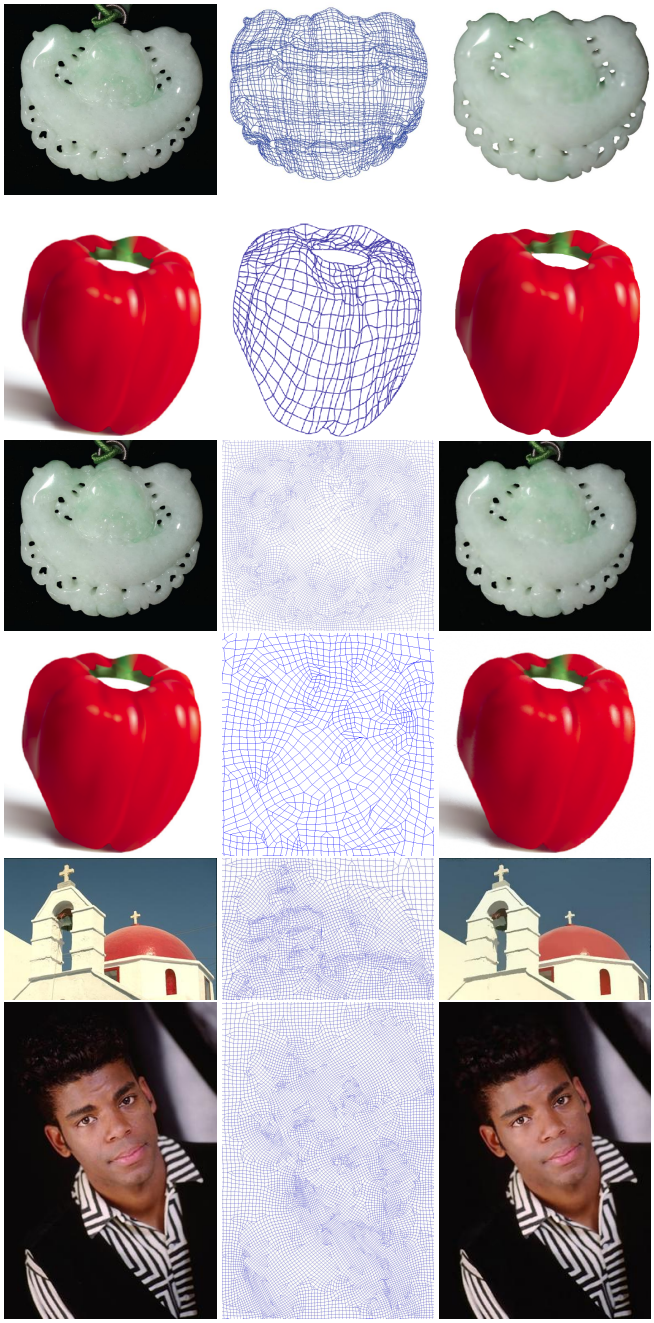
see that our results also achieve high visual quality compare with Lai's method under a similar number of quad elements.

Fig. 14 shows examples containing holes comparisons between our method and Lai et al. [LHM09], within which the Amulet image contains 21 holes. Since their method conformally parameterizes the object-of-interest into a rectangular domain with horizontal slips, it often requires the object to have a similar geometry as the parameterization domain to reduce the area distortion. Our method is more flexible and can deal with object with complex geometries. Our method also runs 5 times faster than Lai et al.'s method (see Table 2).

4.2. Vector image editing

Color editing. Color editing is usually used to manipulate the color of a specific object region while maintaining important boundary features. Many existing color editing methods are mainly aimed at raster images rather than vector images. Our quad meshes can restrict color operations to the desired object region. Firstly, we use an interactive image segmentation method [TGVB13] to let users select region of interest and chooses the color transform interval, our algorithm will automatically select control vertices and transform the color value of control vertices by establishing a mapping relationship between the target region and the color transform interval. In other words, we only edit the color value without change color gradient value of the vertex of quadrilateral by calculating the mean and variance of the user-specified color gradient interval. Finally, reconstruct the image based on the color information of the quadrilateral vertices as shown in Fig. 15.

Image embedding. Similar to color editing, the main function of image embedding is that the user selects the region of interest, and then uses our algorithm to partition the simple image into the quadrilateral mesh and obtain the control vertices. Then copy it onto a complex image. Finally, the image of the complex background is cloned onto the final target image as shown in Fig. 16.



Input images Quadrangulation Vectorization results

Figure 14: *Comparison with Lai et al.'s method [LHM09] (row 1-2). Our method (rows 3-6) is able to produce comparable results as their method with the similar number of quad elements, but it runs five times faster. Our method can also deal with image with complex content.*

Discernible image mosaic. We utilize quad meshes to produce discernible image mosaics, with relatively large image tiles replaced by images drawn from a database, to resemble a target image. Our method adopts a quad mesh based descriptor to encode the image in the target image and database: To keep enough information about the image, we split each mesh to the 8*8 grid in the direction of the cross field and calculate the mean of color of each grid. Then use these values to compute the description vector. Before calculating the description vectors, we apply edge-preserving smoothing to the image to remove the details, because they may cause the description vectors to produce noise. In order to directly measure the similarity between the current mesh and each image in the database, we can calculate the L_2 distance between the corresponding vectors. Compared with existing works on image mosaics, our method can process images of any topology, not limited to rectangular images and emphasizes the preservation of visual edges in the target image, which produces the mosaic image is more artistic and ornamental. We compared our method with other representative techniques that generate image mosaics with similar styles to ours. Ref. [PCK09] (GIzMOs) treats an image partitioning as a pre-processing step before replacing the tile, specifically, this method removes some images from their database to reduce the image similarity in favor of a stronger mosaic effect. We also compared with Picture Mosaics [LLC], well-known commercial software. From Fig. 17, where the pictures of Tai Chi and Mona Lisa are divided into 1000 meshes and 3000 meshes, respectively. We can see that the mosaic image produced by our method is closer to the original image and our method outperforms others on complex images.

5. Conclusions

We developed an efficient quadrangulation method tailored for image vectorization. Our method computes a smooth cross field by minimizing a new energy function with scaled constraints according to the direction of image gradient and distance to image boundary, thereby ensuring cross field can preserve the image features well. It then adjusts its singularities through cluttering and adopts a local parameterization to partition the image into a quad-dominant mesh. Our algorithm is fully automatic and it can process images with rich content. We demonstrated the advantages of our method through extensive experimental results and applications.

6. Acknowledgements

We are grateful to the anonymous reviewers for their helpful comments. We would also like to thank the authors of [JTPSH15, LHM09, JFH*15, PCK09] for sharing their programs and results for comparisons. The research of Yuanfeng Zhou was supported by the NSFC Fund (No.61772312), the key research and development project of Shandong province (2017GGX10110), the fundamental research funds of Shandong University (2018JC030). The research of Shiqing Xin was supported by the NSFC Fund (No.61772016). The research of Ying He was supported by the Singapore Ministry of Education Grant RG26/17.

References

[BCE*13] BOMMES D., CAMPEN M., EBKE H.-C., ALLIEZ P., KOBBELT L.: Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 98. 2

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. In *ACM Transactions On Graphics (TOG)* (2009), vol. 28, ACM, p. 77. 1

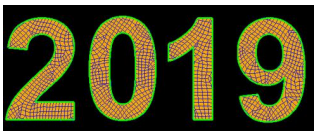
[Can87] CANNY J.: A computational approach to edge detection. In *Readings in computer vision*. Elsevier, 1987, pp. 184–203. 4

[CBK15] CAMPEN M., BOMMES D., KOBBELT L.: Quantized global parametrization. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 192. 2

(a) Input image (b) Region of interest (c) Selecting color gradient interval (d) Editing colors

(e) Input image (f) Editing colors (g) Input image (h) Editing colors

Figure 15: *Our color editing consists of the desired color selection, target color selection, and results.*



(a) Quadrangulation of simple shape

(b) Embedded image

(c) Quadrangulation of simple shape

(d) Embedded image

(e) The result of image embedding

Figure 16: *Image composition.*

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design 22*, 2 (2005), 121–146. 1

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry* (2003), ACM, pp. 312–321. 1

[DDI06] DEMARET L., DYN N., ISKE A.: Image compression by linear splines over adaptive triangulations. *Signal Processing 86*, 7 (2006), 1604–1616. 2

[DL15] DUAN L., LAFARGE F.: Image partitioning into convex polygons. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3119–3127. 2, 6, 7

[EBCK13] EBKE H.-C., BOMMES D., CAMPEN M., KOBBELT L.: Qex: robust quad mesh extraction. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 168. 2

[FH00] FARIN G., HANSFORD D.: *The essentials of CAGD*. AK Peters/CRC Press, 2000. 6

[HSF*17] HOU F., SUN Q., FANG Z., LIU Y.-J., HU S.-M., QIN H., HAO A., HE Y.: Poisson vector graphics (pvg) and its closed-form solver. *arXiv preprint arXiv:1701.04303* (2017). 1, 2

[HT04] HORMANN K., TARINI M.: A quadrilateral rendering primitive. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2004), ACM, pp. 7–14. 2

[HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 517–526. 1

[HZN*18] HUANG J., ZHOU Y., NIESSNER M., SHEWCHUK J. R., GUIBAS L. J.: Quadriflow: A scalable and robust method for quadrangulation. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 147–160. 1

[JFH*15] JIANG T., FANG X., HUANG J., BAO H., TONG Y., DESBRUN M.: Frame field generation through metric customization. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 40. 1, 6, 8

[JTPSH15] JAKOB W., TARINI M., PANOZZO D., SORKINE-HORNUNG O.: Instant field-aligned meshes. *ACM Trans. Graph. 34*, 6 (2015), 189–1. 1, 5, 6, 8

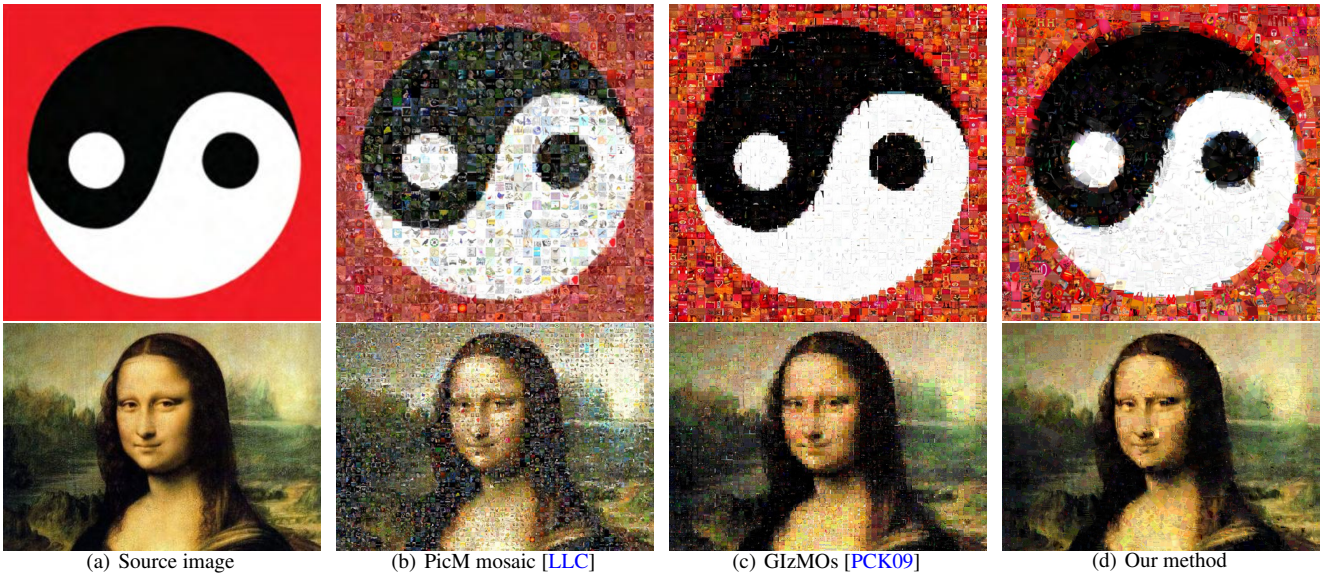
[LHFY12] LIAO Z., HOPPE H., FORSYTH D., YU Y.: A subdivision-based representation for vector image editing. *IEEE transactions on visualization and computer graphics 18*, 11 (2012), 1858–1867. 1

[LHM09] LAI Y.-K., HU S.-M., MARTIN R. R.: Automatic and topology-preserving gradient mesh generation for image vectorization. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 85. 1, 2, 7, 8

[LHS*18] LEE Y.-M., HU M.-H., SU C.-C., CHEN K.-L., TSENG M.-F., WU J.-S., CHENG G. C., BRANAM R.: Progress on developing a multi-physics simulation platform: Rigorous advanced plasma integration testbed (rapit). In *2018 Plasmadynamics and Lasers Conference* (2018), p. 2944. 2

[LJX*10] LAI Y.-K., JIN M., XIE X., HE Y., PALACIOS J., ZHANG E., HU S.-M., GU X.: Metric-driven rosy field design and remeshing. *IEEE Transactions on Visualization and Computer Graphics 16*, 1 (2010), 95–108. 1

[LKSD17] LIENG H., KOSINKA J., SHEN J., DODGSON N. A.: A colour interpolation scheme for topologically unrestricted gradient meshes. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 112–121. 1



(a) Source image (b) PicM mosaic [LLC] (c) GIzMOs [PCK09] (d) Our method

Figure 17: *Comparison with the existing methods. Thanks to the feature aligned cross fields, our method can preserve the features well.*

[LL06] LECOT G., LEVY B.: Ardeco: automatic region detection and conversion. In *17th Eurographics Symposium on Rendering-EGSR'06* (2006), pp. 349–360. 2

[LLC] LLC P. M.: Picture mosaics. https://www.picturemosaics.com/. Accessed April 4, 2019. 8, 10

[LLL17] LIAO W., LIU H., LI T.: Interactive shape editing for subdivision surfaces. In *Subdivision Surface Modeling Technology*. Springer, 2017, pp. 197–216. 2

[LSK*09] LEVINSHTEIN A., STERE A., KUTULAKOS K. N., FLEET D. J., DICKINSON S. J., SIDDIQI K.: Turbopixels: Fast superpixels using geometric flows. *IEEE transactions on pattern analysis and machine intelligence 31*, 12 (2009), 2290–2297. 6

[LZ14] LEVI Z., ZORIN D.: Strict minimizers for geometric optimization. *ACM Transactions on Graphics (TOG) 33*, 6 (2014), 185. 2

[MZ13] MYLES A., ZORIN D.: Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 105. 2

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: a vector representation for smooth-shaded images. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 92. 2

[PCK09] PAVIĆ D., CEUMERN U., KOBBELT L.: Gizmos: Genuine image mosaics with adaptive tiling. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 2244–2254. 8, 10

[PJSH15] PRÉVOST R., JAROSZ W., SORKINE-HORNUNG O.: A vectorial framework for ray traced diffusion curves. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 253–264. 2

[RVAL09] RAY N., VALLET B., ALONSO L., LEVY B.: Geometry-aware direction field processing. *ACM Transactions on Graphics (TOG) 29*, 1 (2009), 1. 1

[RVLL08] RAY N., VALLET B., LI W. C., LÉVY B.: N-symmetry direction field design. *ACM Transactions on Graphics (TOG) 27*, 2 (2008), 10. 1

[SLWS07] SUN J., LIANG L., WEN F., SHUM H.-Y.: Image vectorization using optimized gradient meshes. In *ACM Transactions on Graphics (TOG)* (2007), vol. 26, ACM, p. 11. 1, 2

[SM00] SHI J., MALIK J.: Normalized cuts and image segmentation. *Departmental Papers (CIS)* (2000), 107. 6, 7

[SP06] SWAMINARAYAN S., PRASAD L.: Rapid automated polygonal image decomposition. In *AIPR* (2006), IEEE Computer Society, p. 28. 2

[STZ14] SUN T., THAMJAROENPORN P., ZHENG C.: Fast multipole representation of diffusion curves and points. *ACM Trans. Graph. 33*, 4 (2014), 53–1. 1, 2

[TC04] TUMBLIN J., CHOUDHURY P.: Bixels: Picture samples with sharp embedded boundaries. In *Rendering Techniques* (2004), pp. 255–264. 2

[TC05] TARINI M., CIGNONI P.: Pinchmaps: Textures with customizable discontinuities. In *Computer Graphics Forum* (2005), vol. 24, Wiley Online Library, pp. 557–568. 2

[TGVB13] TANG M., GORELICK L., VEKSLER O., BOYKOV Y.: Grabcut in one cut. In *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 1769–1776. 7

[WZHS15] WEI X., ZHANG Y., HUGHES T. J., SCOTT M. A.: Truncated hierarchical catmull–clark subdivision with local refinement. *Computer Methods in Applied Mechanics and Engineering 291* (2015), 1–20. 2

[XLM*18] XU G., LI M., MOURRAIN B., RABCZUK T., XU J., BORDAS S. P.: Constructing iga-suitable planar parameterization from complex cad boundary by domain partition and global/local optimization. *Computer Methods in Applied Mechanics and Engineering 328* (2018), 175–200. 1

[XLY09] XIA T., LIAO B., YU Y.: Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 115. 2