TCB-spline-based Image Vectorization

HAIKUAN ZHU, JUAN CAO, YANYANG XIAO, and ZHONGGUI CHEN, Xiamen University, China
ZICHUN ZHONG, Wayne State University, USA
YONGJIE JESSICA ZHANG, Carnegie Mellon University, USA

Vector image representation methods that can faithfully reconstruct objects and color variations in a raster image are desired in many practical applications. This article presents triangular configuration B-spline (referred to as TCB-spline)-based vector graphics for raster image vectorization. Based on this new representation, an automatic raster image vectorization paradigm is proposed. The proposed framework first detects sharp curvilinear features in the image and constructs knot meshes based on the detected feature lines. It iteratively optimizes color and position of control points and updates the knot meshes. By using collinear knots at feature lines, both smooth and discontinuous color variations can be efficiently modeled by the same set of quadratic TCB-splines. A variational knot mesh generation method is designed to adaptively introduce knots and update their connectivity to satisfy the local reconstruction quality. Experiments and comparisons show that our framework outperforms the existing state-of-the-art methods in providing more faithful reconstruction results. In particular, our method is able to model undetected features and subtle or complicated color variations in-between features, which the previous methods cannot handle efficiently. Our vectorization representation also facilitates a variety of editing operations performed directly over vector images.

CCS Concepts: • **Computing methodologies** → **Image processing**; **Parametric curve and surface models;**

Additional Key Words and Phrases: Vector images, simplex splines, knot placement, mesh optimization

Juan Cao and Zhonggui Chen were supported by the NSFC (Grants No. 61872308 and No. 61972327), the Xiamen Youth Innovation Funds (Grant No. 3502Z20206029), and the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (Grant No. VRLAB2021B01). Yanyang Xiao was supported by the NSFC (Grant No. 62102174). Zichun Zhong was supported by the NSF under Grants No. OAC-1657364, No. IIS-1816511, No. OAC-1845962, and No. OAC-1910469. Yongjie Jessica Zhang was supported in part by NSF Grant No. CMMI-1953323 and a Honda grant.
Authors' addresses: H. Zhu and J. Cao (corresponding author), Xiamen University, School of Mathematical Sciences, Xiamen, Fujian 361000, China; emails: hkzhu@wayne.edu, juancao@xmu.edu.cn; Y. Xiao and Z. Chen, Xiamen University, School of Informatics, Xiamen, Fujian, 361000, China; emails: yanyangxiaoxyy@gmail.com, chenzhonggui@xmu.edu.cn; Z. Zhong, Wayne State University, Department of Computer Science, 5057 Woodward Ave., Suite 14109.2, Detroit, MI, 48202, USA; email: zichunzhong@wayne.edu; Y. J. Zhang, Carnegie Mellon University, Department of Mechanical Engineering, 1315 Wean Hall, 5000 Forbes Avenue, Pittsburgh, PA, 15213, USA; email: jessicaz@andrew.cmu.edu.

0730-0301/2022/6-ART34 $15.00
https://doi.org/10.1145/3513132

ACM Reference format:
Haikuan Zhu, Juan Cao, Yanyang Xiao, Zhonggui Chen, Zichun Zhong, and Yongjie Jessica Zhang. 2022. TCB-spline-based Image Vectorization. *ACM Trans. Graph.* 41, 3, Article 34 (June 2022), 17 pages.
https://doi.org/10.1145/3513132

1 INTRODUCTION

Raster images are the most common format for saving raw data acquired by imaging equipment. They use a grid of pixels to represent an image where each pixel has an individual color. While raster images have the power to depict photo-quality pictures, the manipulation of them without pixelating or losing image quality, such as magnifying and editing, is very difficult. Vector images provide an alternative representation to describe images. Unlike pixel-based raster images, vector images use geometry primitives, such as curves and patches, to mathematically define images. Owing to their merits such as editability and scalability, vector images have been increasingly adopted in major operating systems and multimedia frameworks. Since raster images are still the most prevalent image format with their simple and straightforward representation, raster image vectorization, which converts raster images into vector images with desired properties, is increasingly important.

In a full-color raster photograph, subtle details are present when the scene is complex. Many of the existing raster image vectorization methods are able to recover the specified semantically important image features, such as the silhouettes and contours of objects, e.g., Chen et al. [2020], Lai et al. [2009], Orzan et al. [2008], Sun et al. [2007], Xia et al. [2009], Xie et al. [2014], and Zhao et al. [2018]. However, the finer-level details in-between detected curvilinear features, which are also important visual cues for faithfully interpreting the original complex images, may not be well simulated. To model complex color variation, unintended discontinuity or artifacts may be introduced at regions away from the curvilinear features, and high memory footprint may be required in rasterization. Motivated by this observation, we aim to design a novel raster image vectorization method to accurately and compactly approximate both major curvilinear features and in-between color variations.

In this article, we introduce an effective and flexible representation for vector images and tailor a fully automatic raster image vectorization algorithm. Our image representation is inspired by the **triangular configuration B (TCB)**-splines [Liu 2007; Liu and Snoeyink 2007], which is a non-tensor product spline space defined over high-order triangulations. TCB-splines share excellent theoretic properties with classical univariate B-splines, such as partition of unity, local support, polynomial reproduction, and automatic built-in high-order smoothness. We explore these properties of TCB-splines and apply them to vector image representation. Compared to other existing methods, there are several

advantages of our TCB-spline-based representation and the proposed raster image vectorization algorithm in the following:

- Our TCB-spline-based representation is flexible and compatible in modeling both smooth and sharp color variations. Both the colors and the geometry of an object within a raster image are explicitly represented by the same set of quadratic TCB-splines. With a properly defined TCB-spline space, the color values of the resulting image are C^1 continuous everywhere except across sharp features where color discontinuities are desired. With an explicit expression, our vector representation can be rasterized efficiently with a significantly low memory footprint. Our representation also supports a variety of direct editing and interactive authoring.
- Our novel fitting-error-driven image vectorization algorithm is able to provide a faithful reconstruction from the original raster images. Benefiting from the proposed knot placement and triangulation method, the detected curvilinear features can be accurately recovered and complicated color variations between them can be well modeled. Our knot placement and triangulation adaptively introduce more basis functions in the under-fitted regions, hence the undetected finer-level features can also be faithfully and compactly simulated.

The remainder of this article is organized as follows. Section 2 reviews some related work of image vectorization. Section 3 introduces TCB-splines and their properties. Section 4 presents an algorithm overview for image vectorization using TCB-splines and Sections 5–8 describe details of the algorithm. Section 9 provides experiments to illustrate the efficacy and flexibility of the proposed representation and algorithm. Section 10 concludes this article.

2 RELATED WORK

With the increasing need for converting raster images into vectors, raster image vectorization has gained a great amount of research interest in recent years. Many methods have been proposed for vectorization of non-photographic images, such as fonts [ScanFont 2017], art-work [Kansal and Kumar 2015], line drawing [Bessmeltsev and Solomon 2019; Dori and Liu 1999; Guo et al. 2019; Hilaire and Tombre 2006], clip-art [Dominici et al. 2020; Hoshyari et al. 2018; Reddy et al. 2021], and bi-tonal images [Kuo-Chin Fan et al. 1995]. As non-photographic images contain relatively large areas of uniform colors and gradients, vectorization algorithms are mainly based on edge detection, corner detection, contour tracing, segmentation, line pattern recognition, and curves/patches fitting. Vectorization results are usually represented by vector primitives, e.g., curves and patches.

In this article, we focus on automatic vectorization of photographic quality images. One well-known challenge of photographic images is that within certain resolution they cannot well contain both inhomogeneous salient features and fine details and also suffer from complex color variations in-between these features. Hence, a powerful vector representation that is able to faithfully reconstruct color variations across the image space in addition to smooth or sharp features is desired. We discuss different image vectorization approaches below.

Approximation on partitions. There are several methods attempting to represent images using functions individually defined on partitions. The linear approximations on triangulations are typical instances of such representations [Dyn et al. 1990; Kreylos and Hamann 2001; Su and Willis 2004]. Some other methods approximate raster images using higher-order polynomials or generalized barycentric coordinates on optimized Voronoi tessellations [Armstrong 2006; Cao et al. 2018; Chen et al. 2014]. The subregions of a partition are allowed to be of arbitrary shapes and zigzag boundaries in Lecot and Lévy [2006]. However, due to the inherent piecewise structure, image patches on adjacent subregions are C^0 continuously or even discontinuously jointed in the partition-based representation. In addition, the curvilinear features in images are approximated by sequences of line segments in general. These artifacts become more obvious when the image is scaled up to larger sizes. Our method is able to represent the curvilinear feature using quadratic splines, where reconstructed feature curves joining C^1 almost everywhere except at joints, where the spline curves are G^1 continuous. The vector images are naturally C^1 continuous in color variation everywhere except at the curvilinear features, where we enforce the discontinuous color variation to match the original color variation of the input raster image. We can also obtain vector images with higher-order continuities by using higher-order splines.

Parametric patch-based methods. Gradient meshes are a drawing tool introduced in commercial software that is used to interactively create multicolored vector objects. They are another type of vector image representation that can efficiently model smooth color transitions. The image elements of a gradient mesh-based representation are planar rectangular Ferguson patches interpolating specified information, e.g., position, color, and gradient. A gradient mesh is typically used for representing a single object. The creation of gradient meshes for complex objects is labor-intensive. A semi-automatic method was provided to help convert images into gradient meshes [Sun et al. 2007]. However, this method relies on the user's assistances for guiding grid construction. Later, a fully automatic algorithm was proposed to generate a topology-preserving gradient mesh [Lai et al. 2009]. Due to the tensor-product restriction of Ferguson patches, it is challenging for a gradient mesh-based image vectorization method to automatically align patch boundaries and preserve detailed curvilinear features with a general spatial layout. Vector image representation based on triangular Bézier patches is proposed in Xia et al. [2009], where a network of triangular Bézier patches was constructed on a planar triangular mesh. This representation supports a flexible topology and facilitates adaptive patch distributions. However, the color is only C^0 continuous across the patch boundaries, even though the boundaries are not curvilinear features. A hybrid vector representation using parametric patches is proposed in Chen et al. [2020] for real-time editing. The post-processing of repeatedly computing the weighted averages across samples of multiple patches is applied, which could only reduce artifacts at the patch boundaries instead of eliminating them.

Subdivision surface-based methods. To achieve continuous color transitions with higher orders, subdivision-based vector image representations are independently presented in Zhou et al. [2014] and Liao et al. [2012], where a vector image is defined as a subdivision surface on a triangular mesh. In Zhou et al. [2014], triangular meshes used for defining subdivision surfaces are re-

quired to be free of triangles with small angles and vertices with large valences to guarantee the continuity of the final image representation. To obtain a satisfying mesh, the constrained Delaunay triangulation is used, and additional points should be inserted. Due to the restriction on triangulation, more triangles or vertices would be introduced in regions with minor color variations, increasing the difficulty of editing the image. In Liao et al. [2012], the triangular mesh for defining a subdivision surface is obtained by simplifying a dense mesh while preserving the detected features. As the mesh simplification process heavily relies on the feature detection results, without considering approximation errors between the vectorized output and the input raster image, complicated color variations in-between features would be ignored. Besides, artifacts may also occur around high/low valence vertices or slim triangles. As there is generally no specific requirement on the triangular mesh used for spline construction, TCB-spline-based representation can highly adaptive to complex geometry and color variations. Our method optimizes the triangular mesh driven by approximation errors, hence can faithfully and compactly recover more details. Moreover, owing to their explicit expression, the TCB-spline-based vector image can be efficiently rasterized, requiring a much lower footprint memory than the subdivision surface-based representation.

Diffusion curve-based methods. Diffusion curves are the fourth category of vector-based primitives used for smoothly shaded image representation [Orzan et al. 2008]. By diffusing a given set of automatically or manually selected curves in the image space, the final vector image contains sharp features along the selected curves with smoothly shaded regions between them. The diffusion curve-based image vectorization is well suited for interactive design. However, to rasterize a diffusion curve-based vector image, one needs to solve a partial differential equation defined on the entire image domain, which usually suffers from bad runtime performances and stability problems [Jeschke et al. 2009; Sun et al. 2012, 2014; Xie et al. 2014; Zhao et al. 2018]. Also, the diffusion curve methods may not faithfully represent the subtle color variations in natural images due to the limitations with Poisson equations. Although extensions of the diffusion curves framework were developed to improve the expression control (e.g., Finch et al. [2011], Hou et al. [2020], and Hu et al. [2019]), these methods mainly focus on applications such as image authoring and synthesis, rather than automatic vectorization of given images in our article.

3 TCB-SPLINES

In our vector image representation, the entire image is represented as a spline surface with control points in a 5D space (i.e., position + color). We here adopt the triangle configuration B-splines or TCB-splines to represent images. TCB-splines are linear combinations of simplex splines [De Boor 1976] defined over **triangle configurations (t-configs)** [Cao et al. 2019; Liu and Snoeyink 2007; Zhang et al. 2017]. Simplex splines and triangle configurations are the two main ingredients for TCB-spline construction. In the following, we will provide a brief introduction to simplex splines, t-configs, and TCB-splines. A detailed introduction to TCB-splines can be found in Cao et al. [2019], Liu and Snoeyink [2007], and Schmitt [2019].

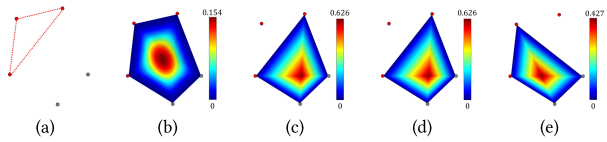


Fig. 1. Recursive evaluation of quadratic simplex splines. (a) Five knots for defining a quadratic simplex spline and the arbitrarily chosen non-degenerate triangle (red dot lines); (b) the quadratic simplex spline defined over panel (a), which can be computed as a linear combination of three linear simplex splines in panels (c–e).

3.1 Simplex Spline

A degree k simplex spline is a piecewise polynomial defined by a set of points (referred to as knots) $V = \{\boldsymbol{t}_0, \ldots, \boldsymbol{t}_{k+2}\} \subset \mathbb{R}^2$ that maps a point $\boldsymbol{u} \in \mathbb{R}^2$ to a real number as [De Boor 1976]

$$M(\boldsymbol{u}|V) = \sum_{j=0}^{2} \lambda_j(\boldsymbol{u}|\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\}) M(\boldsymbol{u}|V\backslash\{\boldsymbol{t}_{i_j}\}), \quad \boldsymbol{u} \in \mathbb{R}^2, \quad (1)$$

where $\lambda_j(\boldsymbol{u}|\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\})$ are barycentric coordinates of $\boldsymbol{u}$ with respect to a non-degenerated triangle $\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\} \subset V$, satisfying $\sum_{j=0}^{2} \lambda_j(\boldsymbol{u}|\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\}) = 1$ and $\sum_{j=0}^{2} \lambda_j(\boldsymbol{u}|\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\})\boldsymbol{t}_{i_j} = \boldsymbol{u}$. When $k = 0$,

$$M(\boldsymbol{u}|\{\boldsymbol{t}_0, \boldsymbol{t}_1, \boldsymbol{t}_2\}) = \begin{cases} 0, & \boldsymbol{u} \notin [V), \\ 1/|area(V)|, & \boldsymbol{u} \in [V), \end{cases} \quad (2)$$

is a normalized characteristic function over the triangle $V = \{\boldsymbol{t}_0, \boldsymbol{t}_1, \boldsymbol{t}_2\}$, where $[\cdots)$ is the half-open convex hull of a set of points [Franssen 1995]. We show an example for recursive evaluation of a quadratic simplex spline in Figure 1, where values of simplex splines are color-coded. Note that, the non-degenerated triangle $\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\}$ in Equation (1) can be chosen arbitrarily from V, i.e., the simplex spline is well-defined, independent of the specific choice of the non-degenerated triangle [Franssen 1995]. A degree k simplex spline defined in Equation (1) is automatically C^{k-1} smooth on the convex hull of V when knots in V are in general position (i.e., there are no duplicate knots and no three knots are collinear). Otherwise, if s knots in V are collinear, then the simplex spline is C^{k+1-s} continuous across this line. In particular, a degree k simplex spline with $k + 2$ collinear knots is C^{-1} continuous across this line; see a quadratic example in Figure S1 of the Supplementary Material.

3.2 T-configs and TCB-splines

Given a set K of n knots, a subset of $k + 3$ knots of K is referred to as a degree k configuration. Different methods have been proposed to select configurations such that simplex splines defined over these subsets span a bivariate spline space over K with desired properties [Neamtu 2001]. Configurations used in this article are t-configs [Liu 2007]. An algorithmic method, the so-called **link triangulation procedure (LTP)**, is proposed to recursively generate the family of t-configs [Liu 2007; Liu and Snoeyink 2007]. In the following, we briefly introduce the basic concept of t-configs, the LTP, and TCB-splines. We refer the reader to Cao et al. [2019], Liu [2007], Liu and Snoeyink [2007], and Schmitt [2019] for more details of the theory and algorithms.

A degree k t-config of K generated by the LTP is a pair of knot subsets (T, I) such that $T \bigcap I = \emptyset$, $\#I = k$ and $\#T = 3$. Denote

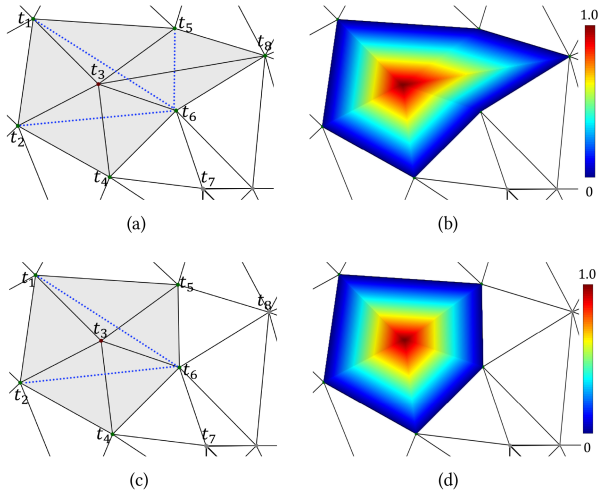


Fig. 2. Linear TCB-splines defined over different triangulations, where Γ_0 is shown in solid black lines and the polygons formed by the links of $\{\boldsymbol{t}_3\}$ are filled in gray and triangulated by dotted blue lines. (a) Degree one t-configs $\{(\{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_6\}, \{\boldsymbol{t}_3\}), (\{\boldsymbol{t}_2, \boldsymbol{t}_4, \boldsymbol{t}_6\}, \{\boldsymbol{t}_3\}), (\{\boldsymbol{t}_5, \boldsymbol{t}_1, \boldsymbol{t}_6\}, \{\boldsymbol{t}_3\}), (\{\boldsymbol{t}_5, \boldsymbol{t}_6, \boldsymbol{t}_8\}, \{\boldsymbol{t}_3\})\}$; (b) linear TCB-spline basis $B_{\{\boldsymbol{t}_3\}}$ defined over t-configs in panel (a); (c) degree one t-configs $\{(\{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_6\}, \{\boldsymbol{t}_3\}), (\{\boldsymbol{t}_2, \boldsymbol{t}_4, \boldsymbol{t}_6\}, \{\boldsymbol{t}_3\}), (\{\boldsymbol{t}_5, \boldsymbol{t}_1, \boldsymbol{t}_6\}, \{\boldsymbol{t}_3\})\}$; and (d) linear TCB-spline basis $B_{\{\boldsymbol{t}_3\}}$ defined over t-configs in panel (c).

ALGORITHM 1: Link triangulation procedure

Input: Γ_k.
Output: Γ_{k+1}.
$\Gamma_{k+1} \leftarrow \emptyset$;
for *each vertex* $I \bigcup \{\mathbf{t}_i\}$ *of* Γ_k **do**
  let L be the link of $I \bigcup \{\boldsymbol{t}_i\}$;
  if L *is a non-degenerated polygon;*
  then
    compute a constrained triangulation of L;
    for *each triangle* T^* *of this triangulation* **do**
      $\Gamma_{k+1} = \Gamma_{k+1} \bigcup (T^*, I \bigcup \{\boldsymbol{t}_i\})$;
    end
  end
end

a family of degree k t-configs of K as Γ_k. For a degree k t-config $(\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\}, I) \in \Gamma_k$, the subset $I \bigcup \{\boldsymbol{t}_{i_0}\}$ is called a vertex of the t-config, and is also called a vertex of Γ_k. The segment $[\boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}]$ oriented such that $\boldsymbol{t}_{i_0}$ is on the right-hand side is called an oriented edge of $I \bigcup \{\boldsymbol{t}_{i_0}\}$ in the configuration $(\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\}, I)$. Note that different degree k t-configs may have the same vertex. If we connect all the oriented edges of $I \bigcup \{\boldsymbol{t}_{i_0}\}$ in all t-configs of Γ_k, then we get a polygon or a degenerated polygon with zero area, and we call it the link of $I \bigcup \{\boldsymbol{t}_{i_0}\}$ [Schmitt 2019]. The LTP described in Algorithm 1 uses links to generate a family of degree $(k+1)$ t-configs, starting from a family of degree k t-configs. Note that, an arbitrary triangulation of the given knot set K corresponds to a family of degree zero t-configs Γ_0, as each face of the triangulation $\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\}$ corresponds to a degree zero t-config $(\{\boldsymbol{t}_{i_0}, \boldsymbol{t}_{i_1}, \boldsymbol{t}_{i_2}\}, \emptyset)$.

Assume that $X_I = \{(T, I^*) | I^* = I, (T, I^*) \in \Gamma_k\}$ is a set of t-configs sharing the common second knot subset I. $\Im_k$ is the collection of all the different second knot subsets I in Γ_k. A degree k TCB-spline basis function associated with a knot subset $I \in \Im_k$ is defined by a linear combination of simplex splines as in Liu [2007],

$$B_I(\boldsymbol{u}) = \sum_{(T,I)\in X_I} area(T) M(\boldsymbol{u} | T \cup I), \boldsymbol{u} \in \mathbb{R}^2. \tag{3}$$

The average of the knot subset I, denoted by $\mathbf{g}_I = \frac{\sum_{\boldsymbol{t}_i \in I} \boldsymbol{t}_i}{k}$, is called the Greville site of B_I. There is a one-to-one correspondence between the control points and TCB-spline basis functions (or Greville sites). T-configs give rise to a bivariate spline space that retains the fundamental properties of univariate B-splines. Hence, the splines defined over t-configs are referred to as TCB-splines. Figure S2 of the Supplementary Material shows examples of LTP up to degree 2 and the associated TCB-spline basis functions.

It should be pointed out that the LTP can yield different t-config families, then a different spline space for a given set of knots if it starts with a different triangulation of K; see Figure 2 for examples of linear basis functions defined over different triangulations. Since the computation of t-configs has freedom in the selection of triangulations in generating degree zero t-configs, TCB-splines also provide great flexibility for applications such as reconstructing surfaces [Zhang et al. 2017] and solving partial differential equations [Cao et al. 2019; Jia et al. 2013]. We will propose a triangulation method to generate spline basis functions that are suitable for modeling both smooth and sharp color changes in an image.

4 ALGORITHM OVERVIEW

In this article, a vector image is considered as a TCB-spline surface in a 5D space, which is represented by a linear combination of TCB-spline basis functions as

$$\boldsymbol{S}(\boldsymbol{u}) = \sum_{I \in \Im_k} B_I(\boldsymbol{u}) \boldsymbol{c}_I, \boldsymbol{u} \in \mathbb{R}^2, \tag{4}$$

where the control point $\boldsymbol{c}_I = (x_I, y_I, r_I, g_I, b_I)$ is a 5D vector with the first two and last three components representing geometry and color information, respectively. To ease the later discussion, we let $\boldsymbol{c}_I^{\text{geo}} = (x_I, y_I)$ and $\boldsymbol{c}_I^{\text{col}} = (r_I, g_I, b_I)$ denote the geometry part and the color part of $\boldsymbol{c}_I$, respectively; and call them the geometry control point and the color control point of $B_I(\boldsymbol{u})$, respectively. Similarly, we denote $\boldsymbol{S}^{\text{geo}}(\boldsymbol{u})$ and $\boldsymbol{S}^{\text{col}}(\boldsymbol{u})$ (respectively, $\boldsymbol{p}^{\text{geo}}$ and $\boldsymbol{p}^{\text{col}}$) as the geometry part and color part of the 5D surface Equation (4) (respectively, a pixel $\boldsymbol{p} = (x, y, r, g, b)$), respectively. Figure 3 shows a toy example of the image representation based on the quadratic TCB-splines. Note that, there are pairs of C^{-1} continuous basis functions along the feature lines (see Section 7.1), and only the ones above the feature lines are shown in the top row of Figure 3(b). The corresponding degree 2 t-configs are shown in the bottom row of Figure 3(b), where the polygons formed by the links of second knot subset (marked by red points) are filled in gray and triangulated by dotted blue lines. Figure 3(e) gives a close-up view of the top right corner of control net in Figure 3(c) with five locally labeled control points (marked in yellow), which are associated with the five basis functions in the top row of Figure 3(b) and one-to-one corresponding to the labeled Greville sites (marked by blue circles in the bottom row of Figure 3(b), respectively. Each pair of C^{-1} basis functions are associated with a pair of control points sharing the same positions but with different colors. The pairs of

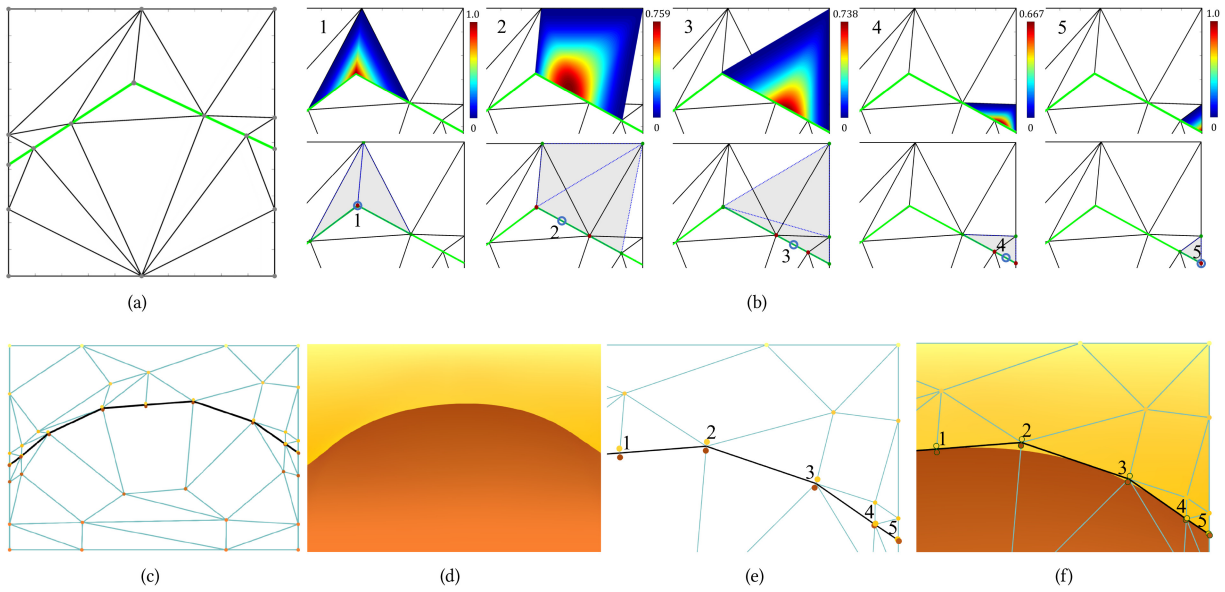


Fig. 3. A toy example of quadratic TCB-splines-based image representation. (a) A unit square domain with knots (marked in solid points), degree zero t-configs (corresponding to the triangulation of knots), and feature lines (marked in green); (b) five quadratic TCB-splines along the right segment of the feature lines (top row) and the corresponding degree 2 t-configs (bottom row); (c) the control points (net) in 5D space; (d) the image represented by quadratic TCB-splines with control net in panel (c) and quadratic TCB-spline basis functions partially shown in panel (b); (e) a close-up view of the top right corner of control net in panel (c) with five locally labeled control points (marked in yellow); and (f) a close-up view of the image and the control net, where the right segment of the feature lines in panel (a) are mapped to the feature curves in panel (f), represented by control points 1–5.

control points are slightly perturbed to ease the visualization in Figures 3(e) and 3(f).

Our goal is to create a TCB-spline surface that best approximates the input raster image, where each pixel has x and y coordinates and RGB color values. We divide each pixel into two subpixel sized triangles by connecting the diagonal. Then, we consider the input full-color raster image as a 5D triangular surface mesh (hereinafter referred to as image mesh) with each vertex associated with a 5D vector $\boldsymbol{p}$, where the components for geometry parts are normalized to $[0, 1]$ and color components for color parts range from 0 to 255. We work on this triangular mesh instead of the original pixel gird for image reconstruction with a sub-pixel accuracy. It also provides convenience for computations in the reconstruction framework, such as parametrization (Section 5.4).

A TCB-spline in Equation (4) is computed in a parametric domain (a unit square) to approximate the 5D triangular mesh. As shown in Figure 4, we first detect curvilinear features in an image that will be preserved in the later vectorization procedure; see Figure 4(b). Second, the image mesh is parameterized onto the parametric domain, where the detected curvilinear features are parameterized onto simplified polylines; see Figure 4(c). Third, a knot mesh is constructed by considering both sharp features and tone variations of the image; see Figure 4(d). Then, a TCB-spline surface is computed to approximate the 5D image mesh (see Figure 4(e)), where each control point at feature lines is split into a pair of points to achieve color discontinuities. Assume a pixel $\boldsymbol{p}$ of input raster image is mapped to point $\boldsymbol{u}$, then the pixelwise fitting errors between the raster image and the vector image Equation (4) in the color part and geometry part are defined as $||\boldsymbol{S}^{\text{geo}}(\boldsymbol{u}) - \boldsymbol{p}^{\text{geo}}||$ and $||\boldsymbol{S}^{\text{col}}(\boldsymbol{u}) - \boldsymbol{p}^{\text{col}}||$, respectively. Figure 4(f) shows the visualization of the pixel-wise fitting error of the color part. Finally, the approximation results are adaptively refined according to approximation errors; see Figures 4(g) and 4(h). The main procedure is shown in Figure 4 and details of each step are described in the following sections.

5 MESH PARAMETRIZATION

To parameterize the input 5D image mesh, we first detect features in the original raster image and map them onto polylines on the parametric domain. These polylines induce parametrization of the 5D image mesh.

5.1 Feature Detection

Image feature detection is a fundamental operation that has vast applications. There exist numerous algorithms in the literature for detecting and extracting features in images, such as the Canny edge detection [Canny 1986] and the edge drawing [Topal and Akinlar 2012]. For the application of image vectorization in this article, we revise the **edge drawing (ED)** method and apply it to extract features in the input image. The basic idea of the ED method is to connect anchors, i.e., pixels with local gradient extremum, by drawing edges between them under the guidance of gradient magnitude and edge direction map. The edge direction at a pixel is vertical if $|G_x| > |G_y|$; otherwise, it is horizontal. The outputs are one-pixel wide, contiguous chains of pixels. In the original ED method, gradient magnitudes are computed as $\sqrt{G_x^2 + G_y^2}$, where G_x and G_y represent horizontal and vertical gradients, respectively.

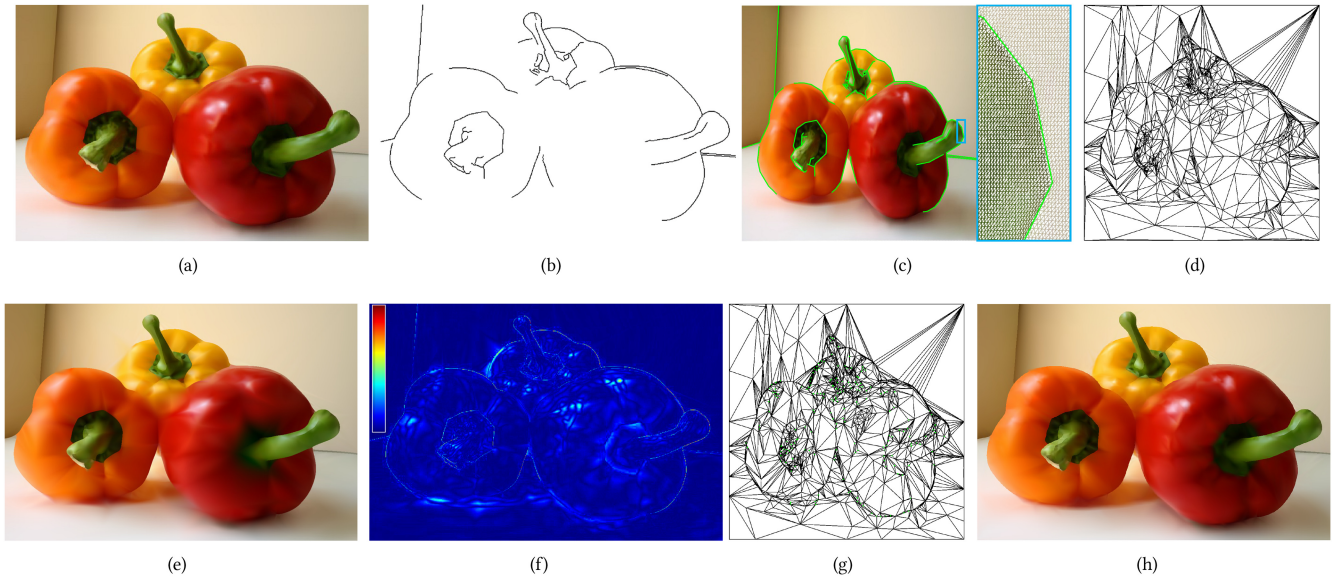


Fig. 4. TCB-spline-based image vectorization pipeline. (a) Input image, which is considered a 5D triangular mesh and referred to as an image mesh; (b) detected curvilinear features; (c) parametrization of image mesh (Section 5) onto the unit 2D domain (left) and a close-up of the parametrization in the blue rectangle box (right), where the detected features are mapped onto the feature lines marked by solid green lines; (d) initial knot mesh (Section 6); (e) vectorization result on panel (d), where signal-to-noise ratio (SNR) is 28.8; (f) the visualization of the pixel-wise fitting error of the color part of the result in panel (e), where red and blue stand for 255 and 0, respectively; (g) adaptively refined knot mesh (Section 8); and (h) final vectorization result.

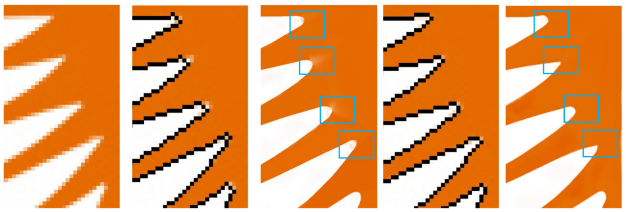


Fig. 5. Feature detection. (a) Input image; (b) feature detected by the ED method [Topal and Akinlar 2012]; (c) vectorization result with the ED method; (d) feature detected by our modified ED method; and (e) vectorization with the modified ED method.

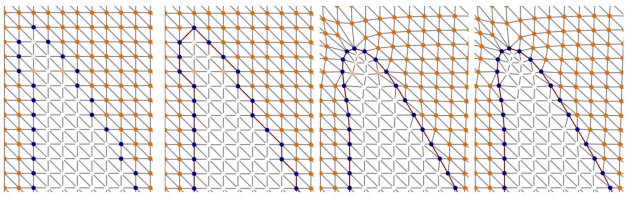


Fig. 6. Feature fairing and color correction. (a) Detected feature points marked in blue color; (b) connectivity update; (c) feature fairing; and (d) color correction.

However, the corners of feature lines are not well located when the value of $|G_x|$ is close to the value of $|G_y|$ (see Figure 5(b)), which may cause blurred colors nearby sharp features in the later color fitting procedure (see Figure 5(c)). Instead, we compute the gradient magnitude as $\max\{|G_x|, |G_y|\}$ at each pixel to obtain more accurate feature detection results; see Figures 5(d) and 5(e) for a comparison result. Experimentally, we find that the feature curves obtained from our modified ED method are within one pixel from the underlying "true" feature curves.

5.2 Feature Fairing

When feature detection is done, we obtain chains of pixels. We modify the image mesh connectivity using edge flipping. In particular, an edge flipping should be performed only if the new edge connects two contiguous points (pixels) of a detected feature chain; see Figure 6(b). Note that the features are zigzag chains of points (image pixels), which would lead to unpleasant volatility in the fitted feature results; see Figure 7(b). Thus, we smooth out the features using Laplacian smoothing [Sorkine et al. 2004]. In particular, a point $\boldsymbol{p}_i$ of the feature chain is replaced by

$$\boldsymbol{p}_i \leftarrow (1-\lambda)\boldsymbol{p}_i + \frac{\lambda}{4}(\boldsymbol{p}_{i-2} + \boldsymbol{p}_{i-1} + \boldsymbol{p}_{i+1} + \boldsymbol{p}_{i+2}), \tag{5}$$

with $\lambda = 0.5$. The positions of points that are not labelled as feature points are then slightly adjusted using the Tutte parameterization method [Tutte 1963] under the constraints of the fixed borders and feature points; see Figure 6(c).

5.3 Color Correction

We assume that the color varies smoothly within an image except for feature curves, where it changes discontinuously. However, the color change usually shows some smoothness around the feature curves for general images. As illustrated in Figure 6(c), the pixels inside and outside the feature curve are supposed to be pure orange and white, respectively. However, there are several pixels near the feature curve with colors in-between the orange and white.

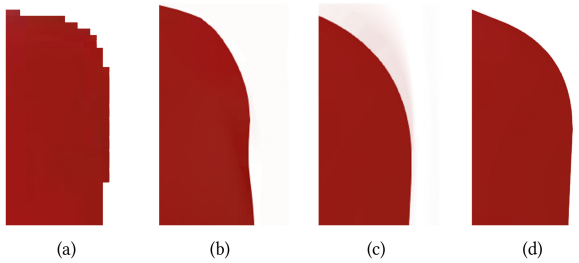


Fig. 7. The performance of our algorithm without feature fairing or color-correction. (a) Input image; (b) result without feature fairing; (c) result without color-correction; and (d) the result with feature fairing and color-correction.

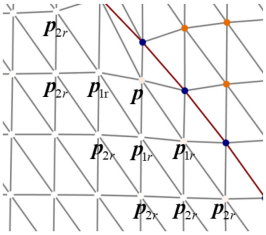


Fig. 8. One-ring non-feature neighbors $\boldsymbol{p}_{1r}$ and two-ring non-feature neighbors $\boldsymbol{p}_{2r}$ of point $\boldsymbol{p}$, where the feature curves is colored in red.

Besides, there may exist some pixels in the one-ring neighborhood of feature curves with quite different colors from other pixels on the same side of feature curves, due to the inaccuracy of the above feature detection method. The inconsistent colors of pixels on different sides of feature curves would lead to blurred reconstruction results; see Figure 7(c). We resolve this problem by correcting the color of pixels in the one-ring neighborhood of feature curves. In particular, the color of a pixel $\boldsymbol{p}$ is replaced by the weighted average color of its one-ring non-feature neighbors $\boldsymbol{p}_{1r}$ and two-ring non-feature neighbors $\boldsymbol{p}_{2r}$ as (see Figure 8)

$$\boldsymbol{p}^{\text{col}} = \omega_1 \bar{\boldsymbol{p}}_{1r}^{\text{col}} + \omega_2 \bar{\boldsymbol{p}}_{2r}^{\text{col}}, \tag{6}$$

where $\bar{\boldsymbol{p}}_{1r}^{\text{col}}$ and $\bar{\boldsymbol{p}}_{2r}^{\text{col}}$ are the average colors of $\boldsymbol{p}_{1r}$ and $\boldsymbol{p}_{2r}$, and weights ω_1 and ω_2 are set to 0.4 and 0.6, respectively. As shown in Figure 6(d), points near feature curves have a consistent color after color correction.

5.4 Image Mesh Parametrization

Parameterization is considered as a crucial step in the spline curve or surface fitting problem, since the quality of fitting or reconstruction results heavily relies on the parametrization. In our image fitting problem, a low-distortion bijective mapping between the input image mesh and a unit squared domain is required. To achieve this goal, we first map each original feature curve to a simplified polyline on the parametric domain such that the mapping is one-to-one. Then, we obtain the parametric coordinates of other points using the Tutte parameterization method [Tutte 1963].

Here, we adapt the framework of the classic Douglas-Peucker (D-P) algorithm [Douglas and Peucker 1973] to the parametrization of feature curves. With a pre-specified distance threshold ε_d (three-pixel width in this article), the basic process of the D-P algorithm for simplifying a given feature curve with vertices $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ is as follows:

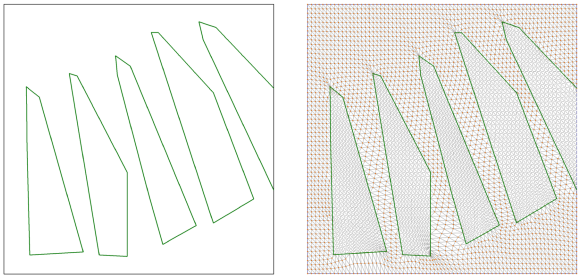


Fig. 9. Parametrization of the image in Figure 5. (a) Feature polylines; and (b) parametrization derived by using the Tutte parameterization method [Tutte 1963] with the constraints shown in panel (a).

Step 1. Find the farthest point $\boldsymbol{p}_i$ from the line segment formed by the first and the last points $\boldsymbol{p}_1, \boldsymbol{p}_n$.

Step 2. If the distance between $\boldsymbol{p}_i$ and $\boldsymbol{p}_1\boldsymbol{p}_n$ is smaller than ε_d, then all the points between $\boldsymbol{p}_1$ and $\boldsymbol{p}_n$ are discarded; otherwise, $\boldsymbol{p}_i$ is included in the simplified polyline.

Step 3. Two new segments $\boldsymbol{p}_1\boldsymbol{p}_i$ and $\boldsymbol{p}_i\boldsymbol{p}_n$ are then recursively handled by D-P algorithm.

The D-P algorithm repeats until no more points are needed to add to the resulting set. To guarantee a one-to-one mapping between the unit parametric-domain and triangular image mesh, we first have to ensure a one-to-one mapping between the feature curve and the simplified polyline. To this goal, we slightly modify the classic D-P algorithm by also including $\boldsymbol{p}_i$ in the resulting set in Step 2, if the projection of points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ to the segment $\boldsymbol{p}_1\boldsymbol{p}_n$ is not consecutively located.

Polylines generated by the D-P algorithm provide a close approximation to the original feature curves. We can obtain low distortions of the parameterization by directly projecting the feature points to the polylines. An example is shown in Figure 9. From these overlap-free constraints, a continuous mapping from the image mesh to the parametric domain is derived using the Tutte parameterization method [Tutte 1963]. Although there is no guarantee that the mapping is bijective with the feature curves and boundary constraints, numerically, we never find overlapping in our experiments. It is owing to the very low distortion introduced by feature curve parameterization. Note that we can also directly use the geometry part of the faired feature curve as the parameter values. However, this straightforward approach will introduce redundant knots around the curved features, as shown in Section 6.1 for the details of knot placement for modeling discontinuous color variations. Therefore, there are many more control points than the feature points around the curved features, leading to overfitting.

6 KNOT MESH GENERATION

In TCB-spline-based image vectorization, the quality of reconstructed image relies heavily on both the position and connectivity of knots. However, the approximation error function, depending on knot position and connectivity, is highly nonlinear and hard to be optimized efficiently. In fact, there is generally no effective and

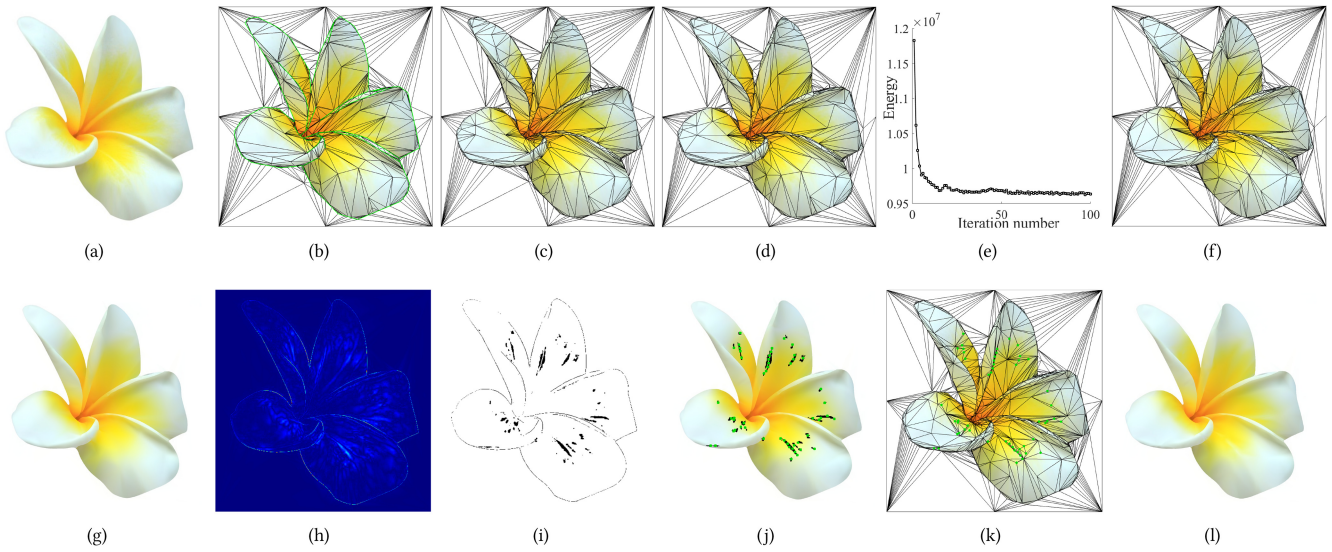


Fig. 10. Knot mesh optimization and adaptive knot mesh updating. (a) Input image; (b) initial knot placement and feature lines (marked in green); (c, d) results of knot position and connectivity optimization after the first iteration of alternating optimization, respectively; (e) plot of the energy versus the number of iterations for the alternating optimization; (f) knot mesh after the alternating optimization algorithm converges; (g) reconstruction result based on the knot mesh in panel (f); (h) fitting error of the reconstruction result in panel (g); (i) pixels with a large fitting error; (j) clusters of pixels in panel (i) and initial position of new knots (marked in green); (k) the optimized knot mesh with new knots inserted; and (l) the fitting result based on the knot mesh in panel (k).

intuitive method for instructing knot placement of spline-based reconstruction or approximation. Even for the simplest case, e.g., classic B-spline curve fitting, we need to solve a complex optimization problem to compute the optimal knot number and position, which is extremely time-consuming in general [Gálvez and Iglesias 2012; Xie et al. 2012; Yoshimoto et al. 2003]. In this section, we introduce a variational method to effectively generate knot meshes with optimal position and connectivity according to minimization of approximation errors. Instead of directly solving the original high nonlinear optimization problem, we solve a reasonably simplified optimization problem, which generate knot mesh close to optimal for TCB-spline approximations.

6.1 Initial Knot Placement

A natural raster image has noticeable curved features and exhibits color discontinuities across these features. We have to represent curved features and smooth/discontinuous color variations in the entire image using TCB-splines faithfully. Recall that TCB-splines possess built-in smoothness properties if all knots are in general position. We focus on quadratic TCB-splines, which are the lowest degree TCB-splines that automatically possess C^1 continuity.

To approximate discontinuity, we introduce collinear knots and multiple knots along the feature lines to locally reduce the continuity of TCB-spline bases. In particular, we evenly distribute one to three knots on each segment of feature lines, and increase the multiplicity of the knot at each vertex of the segment by two, i.e., each vertex is considered as a triple knot; see Figures 10(b) and 11. Besides, the four corners of the unit parametric domain are also considered as triple knots and included in the knot set. Note that, there is a one-to-one correspondence between the input image and the parametric domain by the parametrization method in Section 5.4.

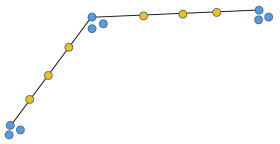


Fig. 11. Collinear knots along feature lines. The triple knots at corners are slightly perturbed to ease the visualization.

In other words, there is a color function defined over the parametric domain, denoted by $\phi(\boldsymbol{u})$. We then greedily insert knots between feature lines within the parametric domain according to the error of the piecewise linear approximation to the color function. More precisely, we triangulate the existing knots on the parametric domain by the Delaunay triangulation method; then, we compute the best linear approximation to the color data within each triangle face; a point randomly sampled from the triangle face with the maximum approximation error is added into the knot set of the triangulation; and the Delaunay connectivity is locally updated. The vertex insertion procedure is carried out until a specific threshold ε_c of the mean of the pixel-wise fitting error of the color part (MEC) is reached; see Figures 10(b)–10(d).

6.2 Knot Mesh Optimization

We denote the knot mesh obtained in the previous section by $T = (V, E, F)$, where $V = \{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_l\}$, $E = \{e_1, \ldots, e_m\}$, and $F = \{f_1, \ldots, f_n\}$ are the sets of knots, edges, and faces, respectively. Based on the knot mesh T, we obtain a piecewise linear approximation to the target color function $\phi(\boldsymbol{u})$ with an approximation error in the L_2 norm:

$$\varepsilon(T) = \sum_{k=1}^{n} \int_{f_k} |\phi(\boldsymbol{u}) - \phi_{f_k}(\boldsymbol{u})|^2 d\boldsymbol{u}, \tag{7}$$

where $f_k \in F$ is a face of the knot mesh T and $\phi_{f_k}(\boldsymbol{u})$ is the optimal linear approximation to $\phi(\boldsymbol{u})$ on f_k. Note that, TCB-splines defined over the knot mesh T are also piecewise polynomials over a partition induced by the associated knots, where the partition is much finer than the knot mesh T [Cao et al. 2019]. Hence, the piecewise linear functions on the knot mesh T as defined in Equation (7) can be viewed as a simplified model of our TCB-splines. The energy function in Equation (7) is possible to be further minimized with respect to the knot positions and connectivity of the knot mesh T. Although the minimizer of the above energy function only provides a knot distribution that is best for piecewise linear approximation to the given image, the proposed energy minimization still provides a heuristic to generate knots that are reasonable for our TCB-spline approximation. The proposed energy function is very difficult to be globally optimized due to its high non-convexity and non-linearity. In the following, we resort to finding a close to optimal minimizer of the energy function by alternating between two steps: knot position optimization and mesh connectivity optimization.

6.2.1 Knot Position Optimization. In the knot position optimization, we optimize the positions of knots while fixing the connectivity of the mesh. We propose to minimize the energy function by a gradient-based minimization algorithm. We first need to compute gradient of the energy function. Note that any changes to the knot positions not only affect the integral domain f_k in the energy function in Equation (7) but all the optimal linear approximation function $\phi_{f_k}(\boldsymbol{u})$ to $\phi(\boldsymbol{u})$ on f_k. To simplify the gradient computation, we assume that $\phi_{f_k}(\boldsymbol{u})$ is independent of knot positions. Therefore, by applying the generalized Leibniz rule [Flanders 1973], we obtain the gradient formula as

$$\frac{\partial \varepsilon(T)}{\partial \boldsymbol{t}_i} = \sum_{j \in N_i} \frac{\int_{\overrightarrow{\boldsymbol{t}_i \boldsymbol{t}_j}} (|\phi(\boldsymbol{u}) - \phi_{f_{ij}}(\boldsymbol{u})|^2 - |\phi(\boldsymbol{u}) - \phi_{f_{ji}}(\boldsymbol{u})|^2)(\boldsymbol{t}_j - \boldsymbol{u})^{\perp} d\boldsymbol{u}}{\|\boldsymbol{t}_j - \boldsymbol{t}_i\|}, \tag{8}$$

where N_i is the set of indices of one-ring neighboring vertices of $\boldsymbol{t}_i$, f_{ij} and f_{ji} are the triangles on the left- and right-hand sides of the directed edge $\overrightarrow{\boldsymbol{t}_i \boldsymbol{t}_j}$, respectively, and $(\cdot)^{\perp}$ means rotating a vector by 90° in the clockwise direction.

Now, we are left with choosing step size such that all variables move an appropriate amount in the steepest descent direction. Note that the energy function depends on the knot positions $\boldsymbol{t}_i$. In the classic gradient descent algorithm, we update all the variables simultaneously according to a certain step size δ in each iteration. In the context of optimizing our energy functions, a knot $\boldsymbol{t}_i$ with corresponding gradient components $\frac{\partial \varepsilon(T)}{\partial \boldsymbol{t}_i}$ is moved by a distance $\delta \| \frac{\partial \varepsilon(T)}{\partial \boldsymbol{t}_i} \|$. The moving distances may vary considerably from knot to knot and give rise to overlaps in the knot mesh, leading to a sharp increase in the energy function. Typically, the step size can be initiated from a unit length and then be reduced by half until the energy decreases. However, this is straightforward yet expensive way to decrease the energy function proposed in this article. Instead, we provide an efficient and intuitive choice of step size to guarantee the convergence of the gradient descent method. In particular, assume $I_{\max}$ is the maximal number of iterations of our gradient-based optimization algorithm, then each knot $\boldsymbol{t}_i$ is updated according to the position of its neighboring knots as

$$\boldsymbol{t}_i^{(k+1)} = \boldsymbol{t}_i^{(k)} - \delta_i^{(k)} \frac{\partial \varepsilon(T)}{\partial \boldsymbol{t}_i} \Big/ \left\| \frac{\partial \varepsilon(T)}{\partial \boldsymbol{t}_i} \right\|, \tag{9}$$

where k is the index of the current iteration and

$$\delta_i^{(k)} = \alpha \cdot \left(\frac{1}{2}\right)^{\frac{k}{I_{\max} - k}} \min_{j \in N_i} ||\boldsymbol{t}_i - \boldsymbol{t}_j|| \tag{10}$$

is the step length for $\boldsymbol{t}_i$ in iteration k with α a constant scaling factor. In our experiments, $\alpha \in [0.3, 0.5]$ usually gives a satisfactory result. The knot updating is terminated if the maximum number of iterations is exceeded. If any flipped faces appear at some iteration steps, then we reduce the step sizes of the corresponding knots by half until these situations are eliminated. The plot of the energy function with respect to the iteration number is presented in Figure 10(e), from which we can observe that our optimization algorithm achieves a fast convergence. Figure 10(c) shows the optimized knot mesh after an iteration of knot position optimization.

6.2.2 Knot Connectivity Updating. The connectivity of the knot mesh also plays an important role in TCB-spline-based approximation. To obtain the optimal connectivity of the knot mesh, we start from the constrained Delaunay triangulation of the obtained knot set, followed by an edge-flipping operation to progressively improve the connectivity. In particular, we create a Delaunay triangulation with constrained edges on feature lines obtained in Section 5.4. Then, an edge that is not constrained is flipped if this would decrease the sum of the energy on its incident triangles. We repeatedly perform the edge flipping until no further decrease of energy is possible; see Figure 10(d) for an example of the knot mesh after connectivity optimization. Note that our knot mesh generation method here may introduce long and skinny triangles to reduce the fitting errors. We observe no numerical trouble in our application with long and thin triangles. However, the extraordinarily long and skinny triangles make our adaptive TCB-spline representation redundant. To achieve more compact representations, we remove knots whose distances to the closest feature curves are smaller than one pixel-width, or adjacent triangles have inner angles smaller than 5 degrees or greater than 175 degrees. Then, local knot connectivity optimization will be re-applied instantly.

7 IMAGE APPROXIMATION

The goal of image vectorization in this article is to represent both the geometry information and the color variation using TCB-splines. In particular, we need to approximate the detected salient edges or contours and preserve sharp color variation across features while representing smooth color variation in-between feature lines in an image. To achieve this goal, we first enforce constraints on the knot mesh to introduce simplex splines that are discontinuous across feature lines. We then compute an approximation of the input image using the method of weighted least squares.

7.1 TCB-splines with Discontinuous Functions

A consequence of collinear knots at feature lines and the constraints of feature edges in the triangulation is the desired

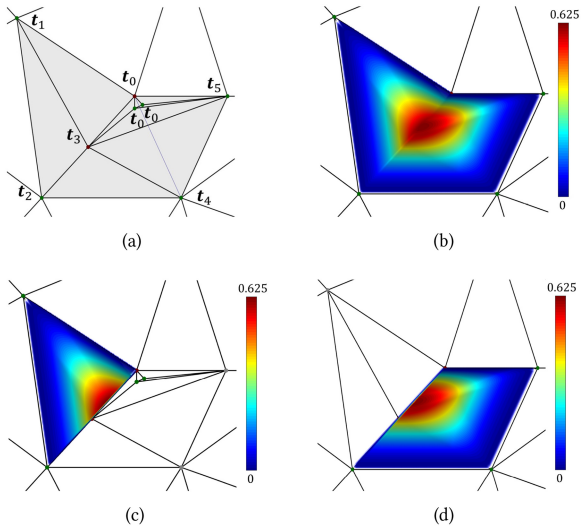


Fig. 12. Basis split. (a) Collinear knots t_0, t_2, and t_3, where the three-fold t_0 is pulled apart into three separate knots for t-config computation and illustration; (b) C^0 TCB-splines defined over T-configs $\{(\{t_0, t_1, t_2\}, I), (\{t_0, t_2, t_4\}, I), (\{t_0, t_4, t_5\}, I)\}$, where $I = \{t_0, t_3\}$; (c, d) C^{-1} basis functions obtained by the trimming TCB-spline in panel (b) along the feature line $t_0t_3t_2$.

discontinuity of a pair of simplex splines across feature lines. Note that a TCB-spline basis is a linear combination of simplex splines defined over t-configs sharing the common second knot subset; see Figure S3 of the Supplementary Material. Hence, each pair of discontinuous simplex splines contribute to the same TCB-spline and are equal when restricting to the correspondingly crossed feature lines. In other words, a quadratic TCB-spline is C^0 continuous across feature lines and can always split into the sum of a pair of C^{-1} functions; see Figure 12 as an example. Please also see Figure S3 of the Supplementary Material for a more detailed example. Instead of considering a TCB-spline B_I across feature lines as a single spline, we consider the corresponding pair of C^{-1} functions (denoted by $B_{I,1}$ and $B_{I,2}$) as separate splines and use them in later approximation. By this function substitution, we are able to represent both continuous (geometry) and discontinuous (color variation) functions across feature lines using the same set of spline functions. We let $\Im^d$ denote the collection of all the different second knot subsets of TCB-spline basis functions that cross feature lines, and let $\Im^c := \Im \setminus \Im^d$.

To find the optimal control points of TCB-splines, we consider the following problem: given a preprocessed image (generated from the feature fairing and color correction operations) with each pixel considered as a 5D point $\boldsymbol{p}_i = (x_i, y_i, r_i, g_i, b_i)$ and its parametric coordinates $\boldsymbol{u}_i$ obtained in Section 5.4, and a set of TCB-spline bases B_I, $I \in \Im = \Im^c \bigcup \Im^d$ with $\Im^d$ defined previously, the goal is to compute a TCB-spline surface

$$\boldsymbol{S}(\boldsymbol{u}) = \sum_{I \in \Im^c} \boldsymbol{c}_I B_I(\boldsymbol{u}) + \sum_{I \in \Im^d} (\boldsymbol{c}_{I,1} B_{I,1}(\boldsymbol{u}) + \boldsymbol{c}_{I,2} B_{I,2}(\boldsymbol{u})), \quad (11)$$

where $\boldsymbol{c}_I$ are control points in a 5D space to approximate the data $\boldsymbol{p}_i$. In the following, we compute the optimal geometry control points and color control points by approximating the geometry and the color variation using weighted least-squares.

7.2 Geometry Approximation

Note that the features convey the most important shape information of an image. Hence, it is desired to preserve the feature lines (e.g., contours). Given the data points $\boldsymbol{x}_i^{\text{geo}} = (x_i, y_i)$, $i = 1, 2, \ldots, M$, we compute the optimal geometry control points $\boldsymbol{c}_i^{\text{geo}}$ by solving the following weighted linear least-squares problem with linear equality constraints:

$$\min \sum_k \omega_k^{\text{geo}} \left\| \boldsymbol{x}_i^{\text{geo}} - \boldsymbol{S}^{\text{geo}}(\boldsymbol{u}_i) \right\|^2, \quad \boldsymbol{c}_{I,1}^{\text{geo}} = \boldsymbol{c}_{I,2}^{\text{geo}}, \quad I \in \Im^d, \quad (12)$$

where $\boldsymbol{S}^{\text{geo}}(\boldsymbol{u})$ is the geometry part of the 5D surface defined in. Equation (11) and ω_k^{geo} is the weight for magnifying the effects of data points at boundaries or features. In our implementation, ω_k^{geo} is set to be 1 by default and increased to 20 and 100 for data points at features or boundaries, respectively.

7.3 Color Approximation

The extracted one-pixel width features could have pixels with colors similar to pixels from both sides. The approximation results may come out blurry at features if we try to preserve the color information at features. Hence, different from previous geometry approximation, the color information at the feature lines is ignored in the color approximation. As the color changes sharply across features, we solve the following unrestricted weighted linear least-squares problem to achieve locally discontinuous color variation:

$$\min \sum_k \omega_k^{\text{col}} \left\| \boldsymbol{x}_i^{\text{col}} - \boldsymbol{S}^{\text{col}}(\boldsymbol{u}_i) \right\|^2, \quad (13)$$

where $\boldsymbol{S}^{\text{col}}(\boldsymbol{u})$ is the color part of the 5D surface defined in Equation (11), and ω_k^{geo} is used for eliminating the effects of colors at feature lines in the least-squares problem. In particular, ω_k^{col} is set to 0 if $\boldsymbol{x}_i$ is at feature lines; otherwise, it is set to 1. Figures 10(g) and 10(h) show approximation results obtained by computing the optimal geometry and color control points. We can observe that the features are basically preserved. To improve reconstruction accuracy, we resort to adaptively adding more knots at regions with a large fitting error and locally optimizing the mesh topologies.

8 ADAPTIVE REFINEMENT

In our adaptive refinement stage, the reconstruction quality of the vector image is progressively improved by adaptively refining the knot mesh according to approximation errors obtained in the previous reconstruction procedure. Note that the number, distribution, and connectivity of newly added knots are three major factors influencing the efficiency of our adaptive refinement algorithm. Unreasonable new knot insertion scheme may cause more iterations or introduce an excessive number of knots to achieve a satisfying result. In this section, we provide a new knot placement method. In each refinement iteration, we first add a moderate number of knots at parametric regions with a large fitting error. Then, we locally optimize the position of knots and their connectivity to other existing knots. Finally, we optimize the control points of TCB-splines defined over the updated knot mesh as we did in Section 7. The

adaptive refinement iteration is repeated until a required number of iterations is reached or both the MEC and the fitting error of the geometry part (MEG) are smaller than the prescribed threshold ε_c and ε_g, respectively. Details of the new knot placement scheme in the sth adaptive fitting iteration step are described in the following subsections.

8.1 New Knot Insertion

We decide the number of newly added knots according to the local fitting errors. We first introduce more knots at feature lines to preserve the salient features if the MEG fails to reach threshold ε_g. As the feature reconstruction can be considered as fitting a curve to the detected salient feature line, we straightforwardly refine the knot interval with a large fitting error at feature lines. In particular, two new knots are evenly inserted to the knot interval if there is a pixel with a fitting error larger than a unit (one-pixel width). To add new knots off feature lines, we first find out all the pixels with a fitting error larger than $\frac{MaxE_s}{2}$, where s is the fitting iteration index and $MaxE_s$ is the maximum fitting error of the color part of the s-th iteration; see Figure 10(i) for an example of $s = 1$. Second, the selected pixels are grouped into several clusters according to the distance between them. In particular, two selected pixels are in the same cluster if they are within two-ring neighborhood of each other; and clusters with less than 15 pixels are discarded; see Figure 10(j). Third, for each cluster, we compute the accumulated fitting error on the two-ring neighborhood of each pixel. The center of a pixel with the largest accumulated fitting error is selected as a new knot. Meanwhile, this pixel together with its two-ring neighbors are removed from the cluster. For each cluster, this procedure is repeated until at most 10 new knots have been selected or all the pixels have been processed. As an example shown in Figure 10(j), there are a total of 52 new knots inserted into the knot set, whose position and connectivity will be further optimized in Section 8.2.

8.2 Knot Mesh Local Optimization

For a knot mesh with new knots inserted, we apply the knot mesh optimization described in Section 6.2 to further locally adjust the position and connectivity of new knots. The original knots on the feature lines and outside the two-ring neighborhood of the new knots are fixed in the optimization; see Figure 10(k) for an example of the optimized knot mesh. The vectorization result after one iteration of adaptive knot insertion and optimization is shown in Figure 10(l), from which we can observe that the quality of color approximation in the petals is improved.

9 EXPERIMENTAL RESULTS

In this section, we present experimental results of our image vectorization framework. To demonstrate the quality of our vector image representation, we also compare our method with four classical methods from Lai et al. [2009], Liao et al. [2012], Orzan et al. [2008], and Xie et al. [2014] and two state-of-the-art methods from Chen et al. [2020] and Zhao et al. [2018]. We perform all our experiments on a laptop PC with a 3.1 GHz Intel Core i5 processor and 12 GB memory.

Table 1. Statistics for the Examples of Image Vectorization Using Our TCB-spline-based Representation

Examples	#Pixels	#Iter.	#Knots	#C.P.	MEC	Stor.	PSNR	Time
Figure 4	793 × 569	3	1,259	3,700	2.07	98	38.9	130.5
Figure 13	521 × 482	2	305	719	1.52	22	41.2	18.0
Figure 14	505 × 561	2	1,058	2,218	1.24	71	44.4	93.0
Figure 15	571 × 546	2	3,912	6,482	2.19	221	37.5	330.6
Figure 16(b)	225 × 214	1	176	382	1.14	11	42.4	6.7
Figure 16(c)	225 × 214	1	176	677	0.92	17	44.5	10.8
Figure 17(b)	544 × 475	1	912	1,305	2.64	42	33.4	31.6
Figure 17(c)	544 × 475	3	1,770	2,379	1.70	78	39.2	178.8
Figure 18	400 × 278	3	1,735	3,652	1.98	107	39.5	216.1
Figure 19	946 × 633	2	2,591	5,834	2.09	168	34.1	551
Figure 20	512 × 512	2	4,163	9,987	1.61	285	40.4	473.4
Figure 21	300 × 450	1	568	986	2.56	38	35.8	22.7
Figure 22	430 × 618	2	1214	2,605	1.29	76	41.8	109.2

Note: # denotes the number of elements; Iter. is short for iterations; C.P. is short for control points; MEC is short for the mean of the fitting error in color part; Stor. (KB) is short for storage, or the memory used to store the vector image without any compression; PSNR means the peak signal-to-noise ratio; Time (s) is the running time for entire adaptive refinement procedure.

Examples of vectorization and local magnification can be found in Figures 13–15. The resulting images are almost C^1 continuous everywhere except across features lines. We can also adopt higher-order TCB-splines in our image vectorization framework to achieve smoother color changes. Examples of quadratic and cubic TCB-splines-based representation obtained from the same set of knots are shown in Figure 16. We show examples of quadratic and cubic TCB-spline-based vector images with comparable MEC in Figure S6 of the Supplementary Material. We can observe that the cubic result has visually smoother color variation and lower MEC than the quadratic result while introducing more control points.

9.1 Comparison to Patch-based Methods

There are several techniques using higher order parametric functions for vector image representation. One of the classical approaches is the gradient mesh, which represents vector images by rectangular arranged Ferguson patches. The discontinuities at the feature curves are modelled using degenerated patches. Whereas, the unstructured nature of the TCB-splines makes it more flexible to adaptively distribute the knots or control points to the curvilinear features and complex color variations. As shown in Figure 17, our TCB-spline-based representation takes much less storage than the gradient mesh representation [Lai et al. 2009] to achieve results with comparable or higher quality. In Chen et al. [2020], a hybrid vector representation using Hermit patches and detailed features is proposed for localized thin-plate spline rasterization. However, this patch-based method cannot entirely remove the artifacts around patch boundaries. Whereas, we achieve visually smoother and more satisfying results, benefiting from the automatic smoothness of the TCB-splines; see the comparison in Figure 18.

9.2 Comparison to Diffusion Curve-based Methods

The diffusion curve-based vector images represent color discontinuities explicitly across the specified diffusion curves.

Fig. 13. Candle. (a) Input image; (b) magnified local view of the input; (c) vectorized image; (d) magnified local view of the vectorized image; and (e) control mesh with features (black lines) of panel (c).

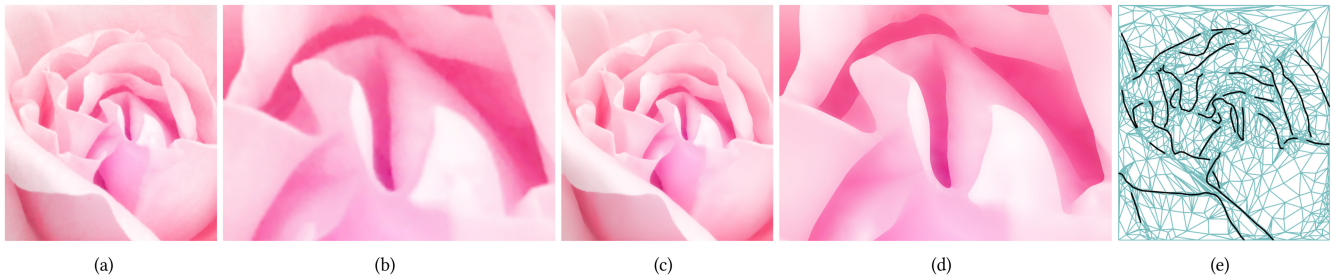


Fig. 14. Rose. (a) Input image; (b) magnified local view of the input; (c) vectorized image; (d) magnified local view of the vectorized image; and (e) control mesh with features (black lines) of panel (c).

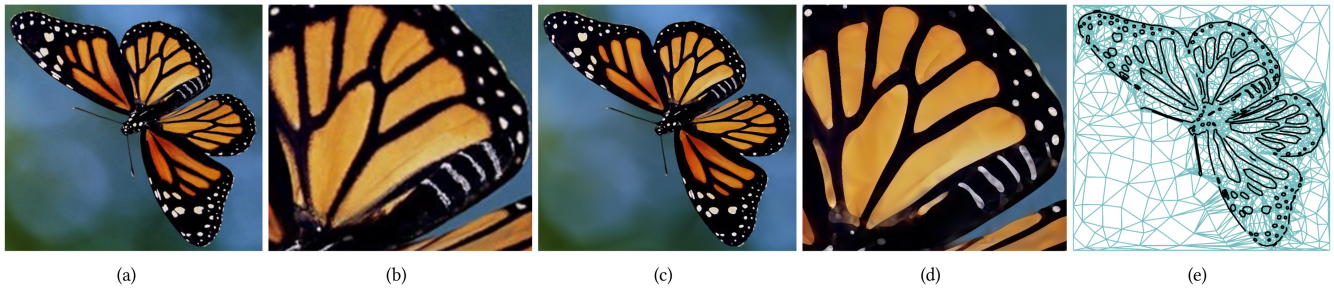


Fig. 15. Butterfly. (a) Input image; (b) magnified local view of the input; (c) vectorized image; (d) magnified local view of the vectorized image; and (e) control mesh with features (black lines) of panel (c).

Analogously, our vectorization results represented by quadratic TCB-splines are theoretically C^{-1} continuous across the detected curvilinear features and naturally C^1 at the remaining regions. There are usually some subtle or complicated color changes in-between the detected features within a photographic image. It is also hard to tell whether there is a color discontinuity in these regions. To represent these complex color variation, diffusion curve-based method may require dense curve networks, which may also introduce unintended discontinuities. Owing to the error-driven knot optimization method, our TCB-spline vectorization method can better capture and model the subtle or complicated color variations away from the detected features, hence achieve visually smoother results; see the comparison of our method with two classical diffusion curve-based method [Orzan et al. 2008; Xie et al. 2014] in Figure 19. In the state-of-the-art [Zhao et al. 2018], diffusion curves are optimized to be more compact, smoother and closed to achieve better approximation accuracy. However, there are still more or totally longer diffusion curves than the feature curves in our method, which introduce unintended discontinuity; see Figure 20.

9.3 Comparison to Subdivision Surface-based Methods

Similar to subdivision surface-based methods, our method can also achieve continuous color variation with higher orders. However, the subdivision surface desires regular control meshes free of high/low valence vertices (refer to as extra-ordinary vertices) and thin and long triangles. Otherwise, the limit surface may contain artifacts around extra-ordinary vertices or slim triangles; see high-resolution rasterization of subdivision-based vector image in Figure 21. Perhaps, for this reason, the control mesh is required to be Delaunay and the minimal angle of triangles no less than 20 degrees in Zhou et al. [2014]. In contrast, there is generally no specific requirement on either the topology of knot meshes or the control nets of TCB-splines. Hence, TCB-splines are relatively

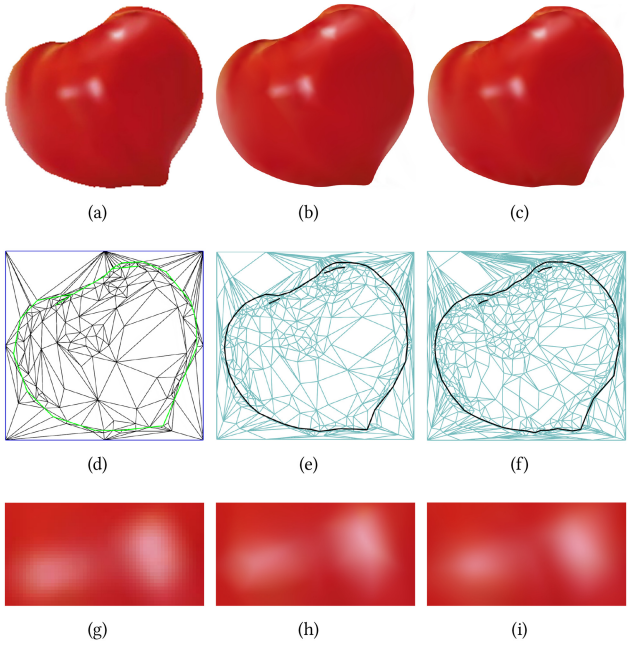


Fig. 16. Cubic and quadratic results on the same knot mesh. (a) Input image; (b, c) quadratic and cubic results on a same knot mesh in panel (d), respectively; (e, f) the control meshes of panels (b) and (c); (g–i) close-up views of the rectangular regions in panels (a–c), respectively.

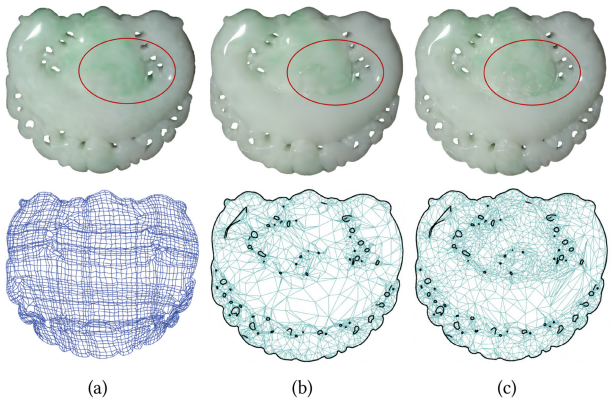


Fig. 17. Representation compactness and quality compared to gradient-mesh-based representation [Lai et al. 2009]. (a) vectorization result using the gradient mesh representation (MEC: 2.76, storage: 206 KB) and the gradient mesh; (b) vectorization result using our TCB-splines with comparable quality to panel (a) (MEC: 2.64, storage: 42 KB) and the control mesh (#C.P.: 1,305); and (c) high-quality vectorization result using TCB-splines (MEC: 1.70, storage: 78K) and the control mesh (#C.P.: 2,379). Panel (a) is taken from Lai et al. [2009]. ©2009 ACM. Included here by permission.

more flexible in representing complex geometry and color variations. With carefully designed knot optimization schemes, our TCB-spline-based vector representation is more adaptive to the complex color variations. As shown in Figures S8(c) and S8(d) of the Supplementary Material, our methods use fewer control points to achieve comparable quality vectorization to the method in Liao et al. [2012].

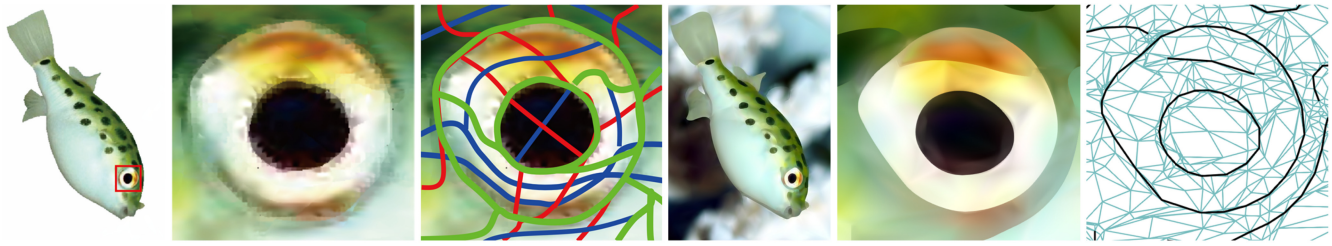
However, TCB-spline-based vector image representation provides explicit formulations, enabling efficient rasterization for vector image display. By directly evaluating the formulations, the memory footprint nearly changes in rasterization, regardless of the output resolution. Differently, the limit of a subdivision surface may be unable to be explicitly expressed. To achieve vector image display, one needs to apply explicit subdivision several times until the refined control mesh is dense enough [Liao et al. 2012]. Hence, the subdivision-surface-based method may incur higher memory and computation costs for generating high-resolution rasterizations. We compare the performance of the single-threaded version of our algorithm and the algorithm in Liao et al. [2012]. Our method requires 0.4 GB for rasterizing Figure 21 of 1.1×10^7 pixels (about the resolution for 4K displays), while [Liao et al. 2012] requires 3.4 GB, starting from an initial control mesh with 2,716 faces and applying six subdivisions. Note that both TCB-spline-based and subdivision surface-based representations have local support properties. Hence, the rasterization of TCB-spline-based vector image can also be accelerated by similar technologies, as in Liao et al. [2012].

9.4 Statistics

The statistics of the knot mesh and control mesh complexity, approximation errors of our vectorization results and the running time are summarized in Table 1. To compute the approximation error per pixel, we rasterize our vectorization output and compare it with the original image. For each tested example, the thresholds ε_c and ε_g for MEC and MEG are 2- and 1-pixel width, respectively. The entire adaptive refinement for image vectorization takes between 6.7 and 551 s, depending on the resolution and complexity of the input raster images. Each iteration of the adaptive refinement (Section 8) includes processes of knot mesh generation (Section 6) or updating (Sections 8.1 and 8.2) and image approximation (Section 7). The later process, which includes basis function computation/updating and control points optimization, is the most time-consuming step. It can usually finish in 294 s for all the test models. For example, the Rose model in Figure 14 with a moderate size takes a total of 93 s and two iterations to reach the threshold of MEC. In the first iteration, the knot mesh generation takes 1.91 s, and the image approximation takes 23.9 s, where the basis function computation and control points optimization take 19.9 s and 4 s, respectively. A quantitative comparison (including RMSE, length of curves, number of control points, and storage) with the above three types of methods can be found in Section 3.1 of the Supplementary Material.

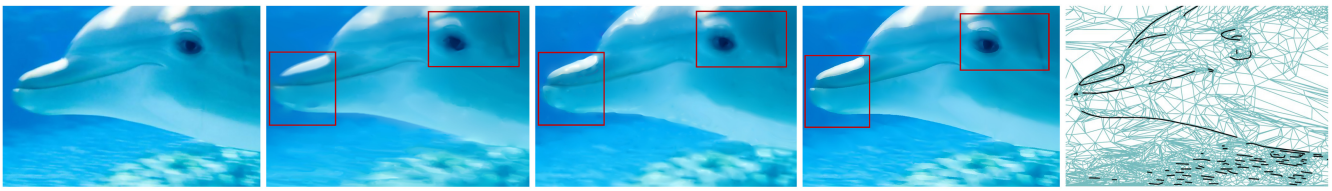
9.5 Editing

We also demonstrate that our vector representation supports a variety of editing operations; see Figure 22 for the shape and color editing results. By changing the color and position of the control points, we obtain new vertorization results. We construct a prototype interactive system for directly editing the TCB-spline-based vector image representation in real-time without resorting to intermediate raster representation. Although the number of control points in TCB-spline-based image representation generally increases with the complexity of input images, it is still much smaller



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

Fig. 18. Representation compactness and quality compared to the thin-plate-based method [Chen et al. 2020]. (a) Results from Chen et al. [2020]; (b) a close-up view of result in panel (a); (c) the patch boundaries of panel (b); (d) vectorization result using our TCB-splines (MEC:1.98, number of control points: 3,652); (e) a close-up view of panel (d); and (f) the control mesh of panel (e). Panels (a–c) are taken from Chen et al. [2020]. @2020 IEEE. Included here by permission.



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

Fig. 19. Comparison of vectorization results with [Orzan et al. 2008; Xie et al. 2014]. Here, we only show close-up views but report the original data in Table 1. (a) Input image from Orzan et al. [2008]; (b) diffusion curve-based representation (results from Orzan et al. [2008] without storage reported); (c) hierarchical diffusion curve-based representation (results from [Xie et al. 2014], storage: 298KB); (d) TCB-spline-based representation (Storage: 168 KB); and (e) control mesh of panel (d).

(a) (b) (c) (d) (e) (f) (g)

Fig. 20. Representation compactness and quality compared to the inverse diffusion curve-based method [Zhao et al. 2018]. (a) Input image; (b) vectorization result from Zhao et al. [2018] (RMSE: 0.0262; total diffusion curve length (normalized such that the length of the longest side of image is 1): 19.11); (c) a close-up view of panel (b); (d) diffusion curves of panel (b); (e) our result (RMSE: 0.0095; total feature curve length (normalized such that the length of the longest side of image is 1): 8.99); (f) a close-up view of panel (e); and (g) control mesh and control polygon of feature curves (marked in black lines) of panel (f). Panels (a–d) are taken from Zhao et al. [2018]. ©2018 IEEE. Included here by permission.

than the number of pixels of the input raster images. To avoid tediously individual control points editing, we provide different editing tools to achieve coarse-to-detailed levels of editing. Our real-time editing tools include a **region of interest (ROI)** selection operation, ROI-based, circle-based, and control-point-based shape/color editing operations; see the supplementary video. The user can use the brush tool and feature polygon selection tool to select control points of ROI and feature curves for editing, respectively. The ROI-based shape deformation is achieved by using the Mean Value Coordinates method [Hormann and Floater 2006]. We first specify a bounding cage that encloses ROI, and then continuously move the enclosed control points by adjusting the cage vertices. The circle-based tool lets us manipulate (e.g., twirl, push inward/outward, and transform) the circled control points, similar to the Liquify tool for raster images in Adobe Photoshop. We resort to the finest level of editing, i.e., individual control point editing only when the provided tools cannot achieve satisfactory editing results. Control points may also be edited by incorporating our interactive system with other more sophisticated manipulation methods [Jacobson et al. 2011; Xie et al. 2014].

9.6 Authoring

The example in Figure 23 shows an authoring result using quadratic TCB-splines-based vector image representation. We also include the interactive sequences in the accompanying video. We take as input the specified quadratic feature curves (see Figure 23(a)). Curve modeling methods (e.g., Chen et al. [2019] and Yan et al. [2017]) can be used to design the input curves

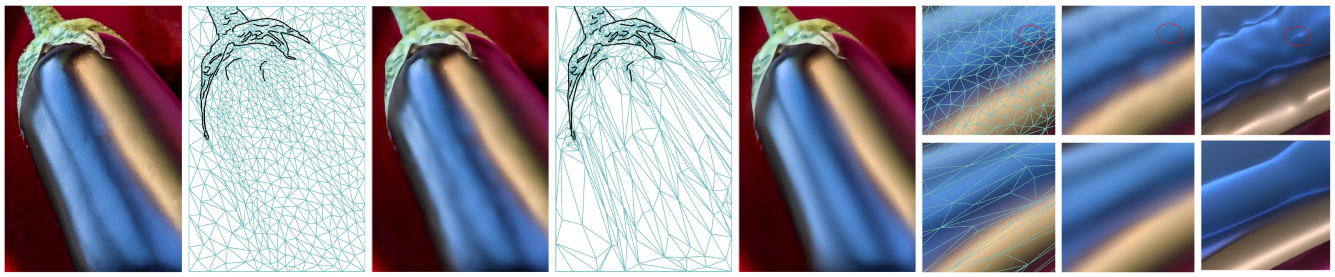


Fig. 21. Adaptive representations and artifacts at extra-ordinary vertices: comparison with subdivision-based methods. (a) Input; (b) control mesh and vector result by [Liao et al. 2012]; (c) result by our method; (d) close-up views of panel (b, top) and panel (c, bottom), where an extra-ordinary vertex of subdivision surface is marked by red ovals and the contrast is enhanced by the 3D reconstructed surfaces (gray-scale as height) in the last column.

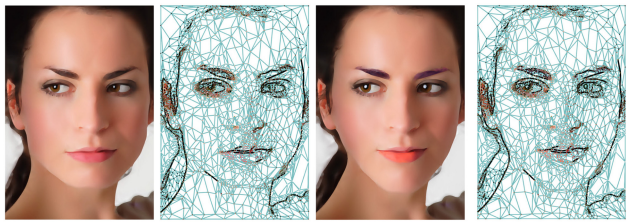


Fig. 22. Color editing and shape editing. (a, d) Input vector image and the corresponding control mesh; (c, d) color and shape editing result and the corresponding control mesh.

interactively. First, we sample the feature curves and generate feature polygons on the parametric domain using D-P algorithms described in Section 5.4. Second, collinear knots and multiple knots are introduced along the feature lines and at the parametric domain as in Section 6.1. Moreover, we uniformly distribute a given number of knots on the parametric domain; see Figure 23(b), where ten extra knots are uniformly placed at the domain boundary. Third, we compute the t-configs and the TCB-spline basis as in Section 3.2. For each TCB-spline basis $B_I(\boldsymbol{u})$ associated with knot subset I, we assign the average of knots in I to the geometry part of the control points; see Figure 23(c). The color part is specified as a default value. Last, we can further modify the vector images by adjusting the control points, as described above. Figure 23(d) shows the output vector images by only changing the colors using the brush tools in our editing system.

We can further insert more primitives by specifying a sequence of points with assigned colors in the image domain to indicate the rough position of control points. We also provide four options to achieve different color variations around the newly introduced control points, including C^1 continuous, C^1 continuous along polyline connecting specified points, C^0 and C^{-1} continuous across feature curves, as shown in Figures 23(e) and 23(h). Please refer to Figure S10 and Section 4 in the Supplementary Material for the details about primitive insertion and the supplementary video shows the interactive manipulations for authoring and editing. The diffusion curve-based vector images represent color discontinuities explicitly across all inserted diffusion curves. In contrast, our representation is more flexible to create both smooth and discontinuous color changes. As the examples demonstrate, our method provides controllable results from a compact user specification. We also report the numbers of feature curves and control points for this example to intuitively measure the input complexity and authoring feasibility. Note that the knot mesh in Figures 23(b) and 23(f) only provides the intermediate result for generating the final control meshes. Users do not need to worry about the knot mesh generation or editing in the application of authoring, as they can intuitively edit the vector image via control meshes.

Limitation. Existing research in vectorization is done with various objectives, such as reducing user intervention, minimizing the number of colors used, providing multiresolution abstraction and stylization, preserving editability, or matching appearance with the input. Since our goal is to provide a faithful reconstruction of the input raster image, our method may require a slightly more storage or introduce more control points for editing purposes. However, with the ability to reproduce more faithful results, such imperfection is usually ignorable. However, the resolution-independent nature of our TCB-spline-based vector format makes a direct compression of input images. However, as requiring many vector primitives to achieve a realistic and detailed look, our method may not outperform the state-of-the-art bitmap compression and super-resolution methods in terms of image compression rate.

10 CONCLUSIONS

In this article, we have proposed a novel TCB-spline-based vector image representation and its associated automatic vectorization framework for raster images. By inserting collinear knots at feature lines, both smooth and discontinuous color variations are faithfully modeled by the same set of quadratic TCB-splines. A variational knot mesh generation method is tailored and incorporated into our framework, which adaptively introduces more knots at regions with low reconstruction quality. Experiments and comparisons show that our framework performs better than other existing methods in modeling undetected features and complicated color variations in-between feature lines. Our vectorization representation also facilitates a variety of editing operations performed directly over vector images. Our vector representation for images can directly be generalized to the vector representation for videos using trivariate TCB-splines with control points in 6D (space-color-time) space. One interesting future work is to generalize the vector

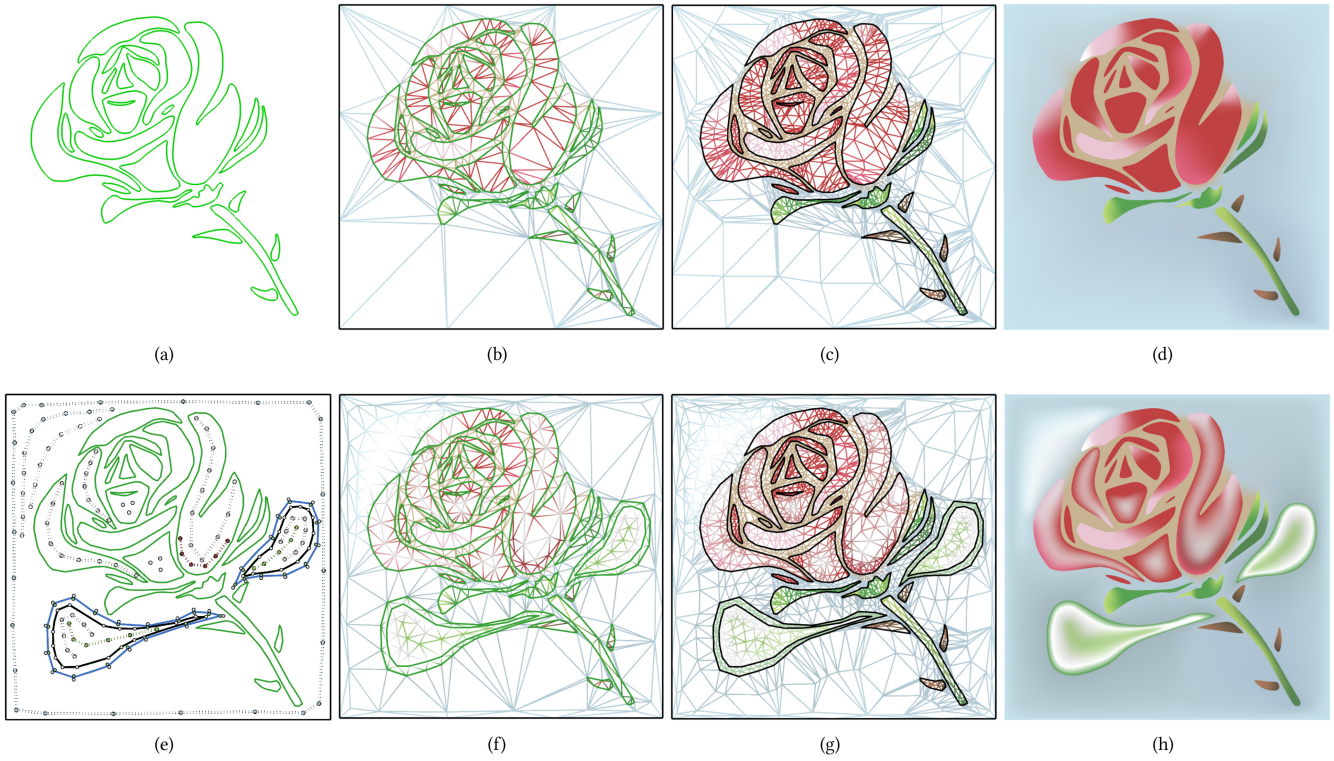


Fig. 23. Example of authoring using quadratic TCB-splines. (a) The input feature curves (26 curves); (b) initial knot meshes (482 knots) with colors; (c) control mesh of 1,133 control points for panel (b), with the color specified; (d) vector image for panel (c); (e) new primitives are added into panel (d), where the rough positions of primitives with different color variations are marked by circles (C^1 continuous), circles connected by dash lines (C^1 continuous along line segments), circles connected by solid black lines (C^0 continuous) and solid blue lines (C^{-1} continuous), respectively; (f) updated knot mesh (with 664 knots); (g) updated control mesh (1,636 control points) for panel (f); and (h) the final vector output for panel (g).

reconstruction algorithm from images to videos. The entire video, treated as 6D volume data, can be reconstructed directly instead of being vectorized on a frame-by-frame basis to achieve temporal consistency.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments. We are grateful to Shuang Zhao, Kuo-Wei Chen, and Hailing Zhou for sharing their data and image results for the comparison. We also thank Chih-Yuan Yao for granting permission to use materials on the website: http://graphics.csie.ntust.edu.tw/pub/RealTimeTPS/.

REFERENCES

Curtis A. Armstrong. 2006. *Vectorization of Raster Images Using B-Spline Surfaces.* Master's thesis. Brigham Young University-Provo.

Mikhail Bessmeltsev and Justin Solomon. 2019. Vectorization of line drawings via polyvector fields. *ACM Trans. Graph.* 38, 1, Article 9 (Jan. 2019), 12 pages.

John Canny. 1986. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (1986), 679–698.

Juan Cao, Zhonggui Chen, Xiaodong Wei, and Yongjie Jessica Zhang. 2019. A finite element framework based on bivariate simplex splines on triangle configurations. *Comput. Methods Appl. Mech. Eng.* 357 (2019), 112598.

Juan Cao, Yanyang Xiao, Zhonggui Chen, Wenping Wang, and Chandrajit Bajaj. 2018. Functional data approximation on bounded domains using polygonal finite elements. *Comput.-aided Geom. Design* 63 (2018), 149–163.

Kuowei Chen, Yingsheng Luo, Yuchi Lai, Yanlin Chen, Chihyuan Yao, Hungkuo Chu, and Tongyee Lee. 2020. Image vectorization with real-time thin-plate spline. *IEEE Trans. Multimedia* 22, 1 (2020), 15–29.

Zhonggui Chen, Jinxin Huang, Juan Cao, and Yongjie Jessica Zhang. 2019. Interpolatory curve modeling with feature points control. *Comput.-aided Design* 114 (2019), 155–163.

Zhonggui Chen, Yanyang Xiao, and Juan Cao. 2014. Approximation by piecewise polynomials on Voronoi tessellation. *Graph. Models* 76, 5 (2014), 522–531.

Carl De Boor. 1976. *Splines as linear combinations of B-splines. A survey.* Technical Report No. MRC-TSR-1667. Wisconsin University-Madison Mathematics Research Center.

Edoardo Alberto Dominici, Nico Schertler, Jonathan Griffin, Shayan Hoshyari, Leonid Sigal, and Alla Sheffer. 2020. PolyFit: Perception-Aligned vectorization of raster clip-art via intermediate polygonal fitting. *ACM Trans. Graph.* 39, 4, Article 77 (July 2020), 16 pages.

Dov Dori and Wenyin Liu. 1999. Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.* 21, 3 (1999), 202–215.

David H. Douglas and Thomas K. Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: Int. J. Geogr. Info. Geovisual.* 10, 2 (1973), 112–122.

Nira Dyn, David Levin, and Samuel Rippa. 1990. Data dependent triangulations for piecewise linear interpolation. *IMA J. Numer. Anal.* 10, 1 (1990), 137–154.

Mark Finch, John Snyder, and Hugues Hoppe. 2011. Freeform vector graphics with controlled thin-plate splines. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–0.

Harley Flanders. 1973. Differentiation under the integral sign. *Amer. Math. Month.* 80, 6 (1973), 615–627.

Michael Franssen. 1995. *Evaluation of DMS-splines.* Master's thesis. Eindhoven University of Technology.

Akemi Gálvez and Andrés Iglesias. 2012. Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points. *Info. Sci.* 192 (June 2012), 174–192.

Yi Guo, Zhuming Zhang, Chu Han, Wenbo Hu, Chengze Li, and Tien-Tsin Wong. 2019. Deep line drawing vectorization via line subdivision and topology reconstruction. *Comput. Graph. Forum* 38, 7 (2019), 81–90.

Xavier Hilaire and Karl Tombre. 2006. Robust and accurate vectorization of line drawings. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 6 (June 2006), 890–904.

Kai Hormann and Michael S. Floater. 2006. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.* 25, 4 (Oct. 2006), 1424–441.

Shayan Hoshyari, Edoardo Alberto Dominici, Alla Sheffer, Nathan Carr, Zhaowen Wang, Duygu Ceylan, and I.-Chao Shen. 2018. Perception-Driven semi-structured boundary vectorization. *ACM Trans. Graph.* 37, 4, Article 118 (July 2018), 14 pages.

Fei Hou, Qian Sun, Zheng Fang, Yong-Jin Liu, Shi-Min Hu, Hong Qin, Aimin Hao, and Ying He. 2020. Poisson vector graphics (PVG). *IEEE Trans. Visual. Comput. Graph.* 26, 2 (2020), 1361–1371.

Yixin Hu, Teseo Schneider, Xifeng Gao, Qingnan Zhou, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2019. TriWild: Robust triangulation with curve constraints. *ACM Trans. Graph.* 38, 4, Article 52 (July 2019), 15 pages.

Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH)* 30, 4 (2011), 78:1–78:8.

Stefan Jeschke, David Cline, and Peter Wonka. 2009. A GPU Laplacian solver for diffusion curves and Poisson image editing. *ACM Trans. Graph.* 28, 5, Article 116 (Dec. 2009), 8 pages.

Yue Jia, Yongjie Zhang, Gang Xu, Xiaoying Zhuang, and Timon Rabczuk. 2013. Reproducing kernel triangular B-spline-based FEM for solving PDEs. *Comput. Methods Appl. Mech. Eng.* 267 (2013), 342–358.

Ruchin Kansal and Subodh Kumar. 2015. A vectorization framework for constant and linear gradient filled regions. *Visual Comput.* 31, 5 (May 2015), 717–732.

Oliver Kreylos and Bernd Hamann. 2001. On simulated annealing and the construction of linear spline approximations for scattered data. *IEEE Trans. Visual. Comput. Graph.* 7, 1 (Jan 2001), 17–31.

Kuo-Chin Fan, Den-Fong Chen, and Ming-Gang Wen. 1995. A new vectorization-based approach to the skeletonization of binary images. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 2. 627–630.

Yu-Kun Lai, Shi-Min Hu, and Ralph R. Martin. 2009. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Trans. Graph.* 28, 3, Article 85 (July 2009), 8 pages.

Gregory Lecot and Bruno Lévy. 2006. Ardeco: Automatic region detection and conversion. In *Proceedings of the 17th Eurographics Symposium on Rendering (EGSR'06)*. Nicosia/Cyprus, 349–360.

Zicheng Liao, Hugues Hoppe, David Forsyth, and Yizhou Yu. 2012. A subdivision-based representation for vector image editing. *IEEE Trans. Visual. Comput. Graph.* 18, 11 (Nov 2012), 1858–1867.

Yuanxin Liu. 2007. *Computations of Delaunay and higher order triangulations, with applications to splines.* Ph.D. Dissertation. University of North Carolina at Chapel Hill.

Yuanxin Liu and Jack Snoeyink. 2007. Quadratic and cubic B-splines by generalizing higher-order Voronoi diagrams. In *Proceedings of the Symposium on Computational Geometry*. 150–157.

Marian Neamtu. 2001. What is the natural generalization of univariate splines to higher dimensions? In *Mathematical Methods for Curves and Surfaces*. Vanderbilt University, Nashville, TN, 355–392.

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: A vector representation for smooth-shaded images. *ACM Trans. Graph.* 27, 3, Article 92 (Aug. 2008), 8 pages.

Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J. Mitra. 2021. Im2Vec: Synthesizing Vector Graphics without Vector Supervision. Retrieved from https://arXiv:cs.CV/2102.02798.

ScanFont. 2017. Font Lab. Retrieved from http://old.fontlab.com/font-converter/scanfont//.

Dominique Schmitt. 2019. Bivariate B-splines from convex pseudo-circle configurations. In *Fundamentals of Computation Theory*, Leszek Antoni Gąsieniec, Jesper Jansson, and Christos Levcopoulos (Eds.). Springer International Publishing, Cham, 335–349.

Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 175–184.

Dan Su and Philip Willis. 2004. Image interpolation by pixel-level data-dependent triangulation. *Comput. Graph. Forum* 23, 2 (2004), 189–201.

Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. 2007. Image vectorization using optimized gradient meshes. *ACM Trans. Graph.* 26, 3, Article 11 (July 2007).

Timothy Sun, Papoj Thamjaroenporn, and Changxi Zheng. 2014. Fast multipole representation of diffusion curves and points. *ACM Trans. Graph.* 33, 4, Article 53 (July 2014), 12 pages.

Xin Sun, Guofu Xie, Yue Dong, Stephen Lin, Weiwei Xu, Wencheng Wang, Xin Tong, and Baining Guo. 2012. Diffusion curve textures for resolution independent texture mapping. *ACM Trans. Graph.* 31, 4, Article 74 (July 2012), 9 pages.

Cihan Topal and Cuneyt Akinlar. 2012. Edge drawing: A combined real-time edge and segment detector. *J. Visual Commun. Image Represent.* 23, 6 (2012), 862–872.

William Thomas Tutte. 1963. How to draw a graph. *Proc. London Math. Soc.* s3-13, 1 (1963), 743–767.

Tian Xia, Binbin Liao, and Yizhou Yu. 2009. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans. Graph.* 28, 5, Article 115 (Dec. 2009), 115:1–115:10 pages.

Guofu Xie, Xin Sun, Xin Tong, and Derek Nowrouzezahrai. 2014. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Trans. Graph.* 33, 6, Article 230 (Nov. 2014), 11 pages.

Weicheng Xie, Xiufen Zou, Jiandong Yang, and Jiebin Yang. 2012. Iteration and optimization scheme for the reconstruction of 3D surfaces based on non-uniform rational B-splines. *Comput.-aided Design* 44, 11 (Nov. 2012), 1127–1140.

Zhipei Yan, Stephen Schiller, Gregg Wilensky, Nathan Carr, and Scott Schaefer. 2017. K-Curves: Interpolation at local maximum curvature. *ACM Trans. Graph.* 36, 4, Article 129 (July 2017), 7 pages.

Fujiichi Yoshimoto, Toshinobu Harada, and Yoshihide Yoshimoto. 2003. Data fitting with a spline using a real-coded genetic algorithm. *Comput.-aided Design* 35, 8 (2003), 751–760.

Yuhua Zhang, Juan Cao, Zhonggui Chen, and Xiaoming Zeng. 2017. Surface reconstruction using simplex splines on feature-sensitive configurations. *Comput.-aided Geom. Design* 50 (2017), 14–28.

Shuang Zhao, Fredo Durand, and Changxi Zheng. 2018. Inverse diffusion curves using shape optimization. *IEEE Trans. Visual. Comput. Graph.* 24, 7 (2018), 2153–2166.

Hailing Zhou, Jianmin Zheng, and Lei Wei. 2014. Representing images using curvilinear feature driven subdivision surfaces. *IEEE Trans. Image Process.* 23, 8 (Aug 2014), 3268–3280.

Received April 2021; revised January 2022; accepted January 2022