DOI: 10.1111/cgf.14495

COMPUTER GRAPHICS *forum*
Volume 41 (2022), number 6 pp. 23–36

Image Representation on Curved Optimal Triangulation

Yanyang Xiao,[1] Juan Cao[2] and Zhonggui Chen[3]

[1]School of Information Engineering, Nanchang University, Nanchang, China
[2]School of Mathematical Sciences, Xiamen University, Xiamen, China
[3]School of Informatics, Xiamen University, Xiamen, China

Abstract

Image triangulation aims to generate an optimal partition with triangular elements to represent the given image. One bottleneck in ensuring approximation quality between the original image and a piecewise approximation over the triangulation is the inaccurate alignment of straight edges to the curved features. In this paper, we propose a novel variational method called curved optimal triangulation, where not all edges are straight segments, but may also be quadratic Bézier curves. The energy function is defined as the total approximation error determined by vertex locations, connectivity and bending of edges. The gradient formulas of this function are derived explicitly in closed form to optimize the energy function efficiently. We test our method on several models to demonstrate its efficacy and ability in preserving features. We also explore its applications in the automatic generation of stylization and Lowpoly images. With the same number of vertices, our curved optimal triangulation method generates more accurate and visually pleasing results compared with previous methods that only use straight segments.

Keywords: image processing, mesh generation, modelling, curves & surfaces

CCS Concepts: • Computing methodologies → Image processing; Geometry

1. Introduction

The geometric representation of bitmap images can be viewed as a piecewise approximation defined over geometric objects to represent the original image as faithfully as possible. Given its advantages, such as resolution independence, data compression and editing, geometric representation has attracted much attention in various fields, including image approximation (e.g. [LL06, LA13, CXC14, LG19]), vectorization (e.g. [XLY09, ZZW14, FLB17]) and the artist community [Bit14, Bry17]. Although many other geometric representations exist, e.g. the diffusion curve [OBW*08], the gradient mesh [LHM09] and image triangulations [LG19], each has its way and advantages in modelling the colour variation. Image triangulation gains popularity as it can easily be used to create different art effects, such as stylization [LG19] and Lowpoly [GW16, MC17]. This paper provides a new alternative for triangulating images by introducing curved edges to reconstruct colour variations more faithfully and better capture features across the image space.

An image not always only contains straight features. To reduce approximation error, triangulating an image usually requires dense vertices and short segments along its curved features, which can be easily observed from the results of existing methods (e.g. [LA13, LG19]). This requirement can be mainly attributed to the weak ability of straight segments to capture curved features. For instance, the zoomed-in detail in Figure 1(b) shows that some vertices are not located on the feature curves.

Curved edges are much more competitive than straight edges in preserving such features because of their flexible modelling ability (Figure 1(d)). In this paper, we aim to improve the alignment between triangulation edges and image features by replacing straight edges with quadratic Bézier curves. Specifically, each edge is associated with a quadratic Bézier curve whose first and last control points are coincident with the end points of edges. We achieve this curved feature alignment goal by adjusting the positions of vertices and control points. This configuration allows us to place fewer vertices along features to capture them efficiently, and we reconstruct images more accurately by further introducing vertices into other regions with high approximation error. Figure 1 shows the benefits of curved triangulation in preserving features and improving approximation quality.

Generating a curved triangulation that conforms to given mathematical curves has recently attracted much interest (e.g. [ADF14, FP16, HSG*19, MC20]). These methods are mostly proposed for

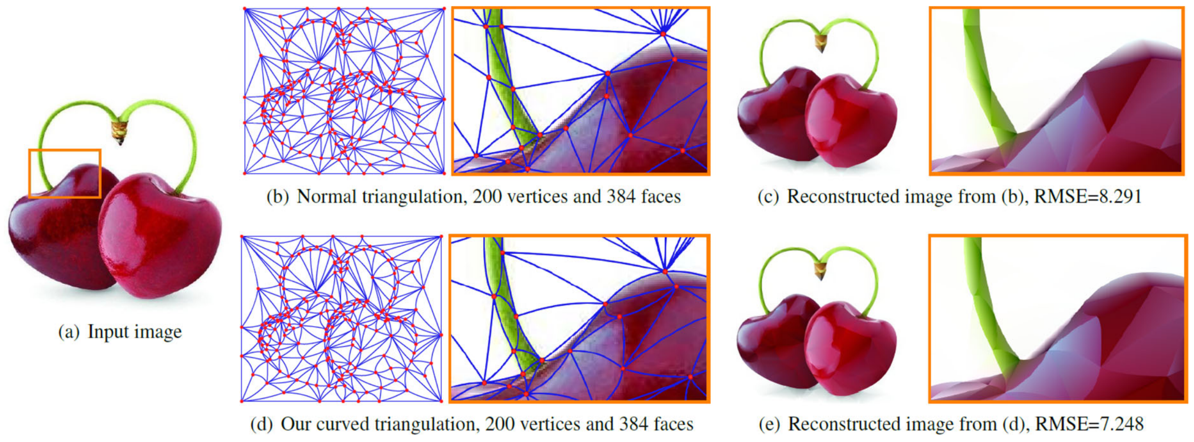

(a) Input image
(b) Normal triangulation, 200 vertices and 384 faces
(c) Reconstructed image from (b), RMSE=8.291
(d) Our curved triangulation, 200 vertices and 384 faces
(e) Reconstructed image from (d), RMSE=7.248

Figure 1: *An input raster image (a) is represented as the optimal triangulation with straight (b) and curved (d) edges. The reconstructed images are shown in (c) and (e). The zoomed-in details and RMSE values illustrate the benefits of curved triangulation in preserving features and improving approximation quality.*

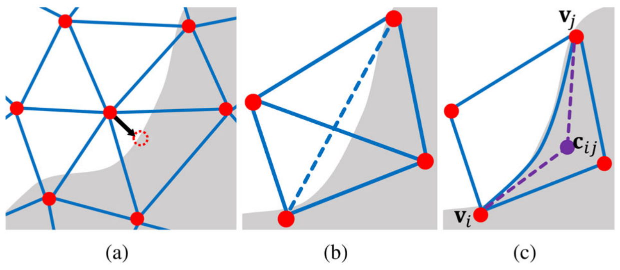


Figure 2: *Three types of elements to be optimized in this paper. The red solid circles and blue solid segments/curves denote the vertices and edges, respectively. (a) Vertex optimization; (b) edge flipping and (c) optimization of control point (purple solid circle).*

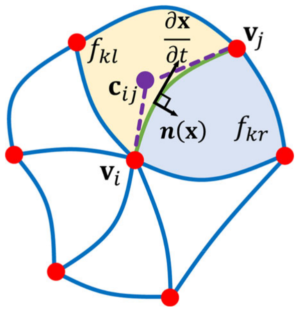

f_{kl}
$\frac{\partial \mathbf{x}}{\partial t}$
$\mathbf{v}_j$
$\mathbf{c}_{ij}$
$\boldsymbol{n}(\mathbf{x})$
f_{kr}
$\mathbf{v}_i$

Figure 3: *Notations used in Section 3.2. f_{kl}: light yellow region, f_{kr}: light blue region, $\widetilde{\mathbf{v}_i\mathbf{v}_j}$: green solid curve and $\mathbf{c}_{ij}$: purple solid circle.*

finite element methods (FEMs) and mainly focus on mesh quality and preservation of given curves. By contrast, we focus on improving image approximation quality by introducing curved triangulation. Our goal is to minimize the total approximation error between the original image and the piecewise approximation over the triangulation. We formulate this goal as an optimization task determined by the vertex locations, connectivity and control point positions of the triangulation in order for the curved optimal triangulation to correspond to the minimum approximation error. Our contributions can be summarized as follows:

1. By introducing curved edges, we propose the energy function of curved optimal triangulation whose minimal result outperforms those triangulations with only straight segments in capturing curved features and improving approximation quality.
2. We deduce explicitly the gradient formulas of the proposed energy function with respect to the positions of vertices and control points, thereby allowing us to optimize the energy function efficiently.
3. We propose an optimization algorithm for searching the curved optimal triangulation of given images, and for generating more accurate and visually pleasing results compared with extant methods. We also explore its applications in the automatic generation of stylization and hybrid Lowpoly images.

   The paper is organized as follows: Section 2 briefly reviews several related works. Section 3 describes the curved optimal triangulation. Section 4 presents the generation algorithm and Section 5 presents the experimental results and comparisons. Section 6 concludes the paper.

2. Related Work

With a given number of vertices, our method generates curved triangulation with high approximation quality, which is mainly related to image approximation, image triangulation and curved meshing.

2.1. Image approximation

While many studies have examined image approximation, we only focus on those methods that are based on geometric representations. These methods usually find an optimal distribution of some geometric objects to partition the original image based on the approximation error. Triangulation [BE92] and Voronoi diagram [Aur91]

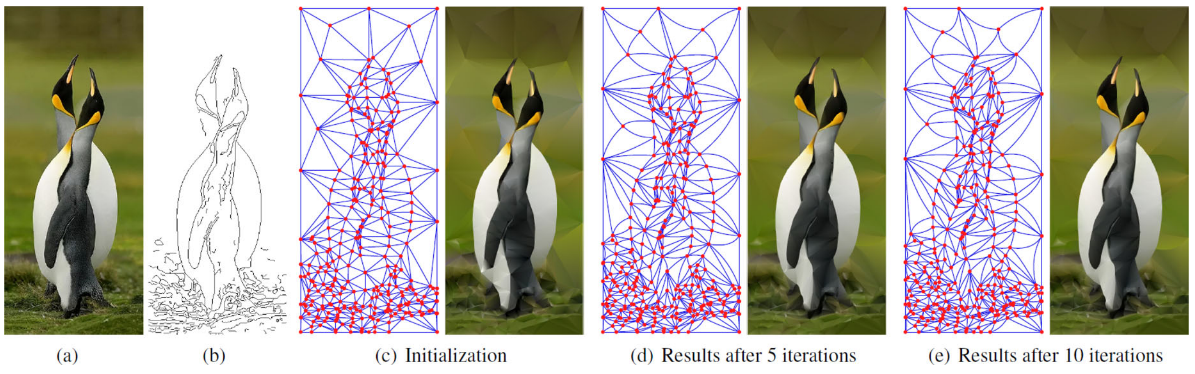

(a)
(b)
(c) Initialization
(d) Results after 5 iterations
(e) Results after 10 iterations

Figure 4: *Algorithm pipeline. (a) Input image; (b) detected features; (c–e) left: triangulation, right: reconstructed image from piecewise approximations over respective triangulation. (c) RMSE = 13.604; (d) RMSE = 11.658 and (e) RMSE = 11.368.*

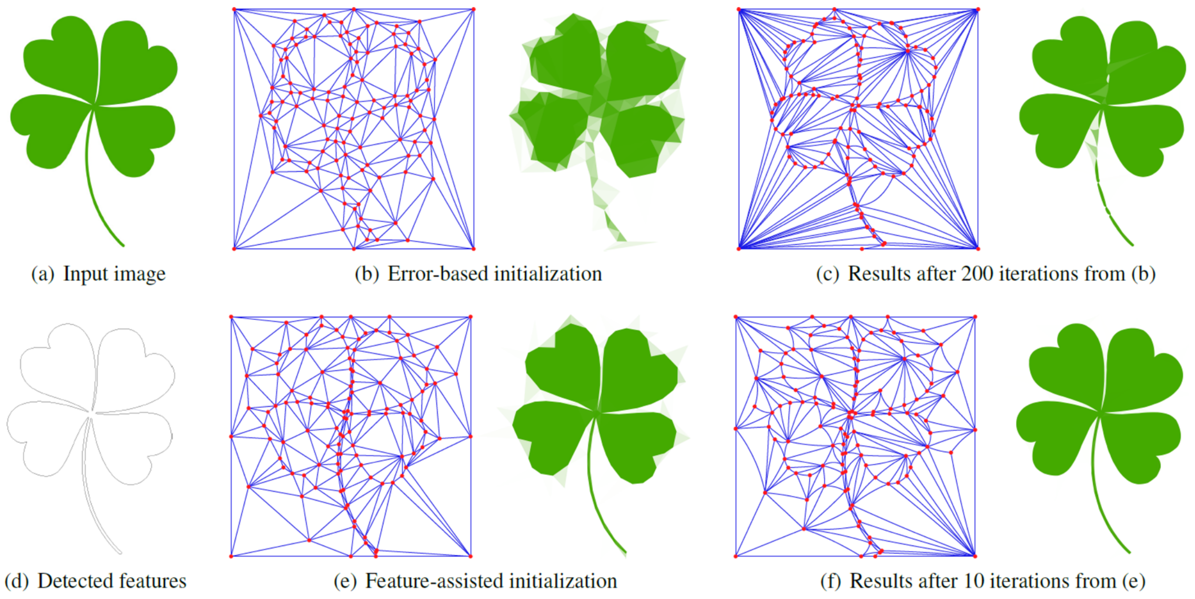

(a) Input image
(b) Error-based initialization
(c) Results after 200 iterations from (b)
(d) Detected features
(e) Feature-assisted initialization
(f) Results after 10 iterations from (e)

Figure 5: *Initialization comparison. Meshes have 120 vertices. (b) 232 faces, RMSE = 25.979; (c) RMSE = 7.781; (e) 227 faces, RMSE = 11.776; (f) RMSE = 3.369.*

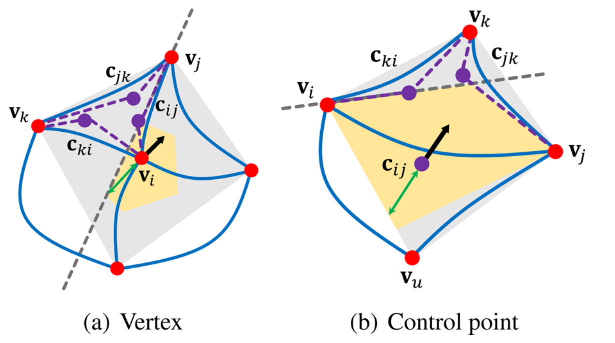
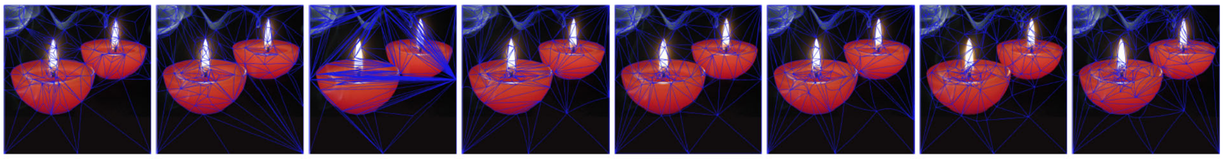


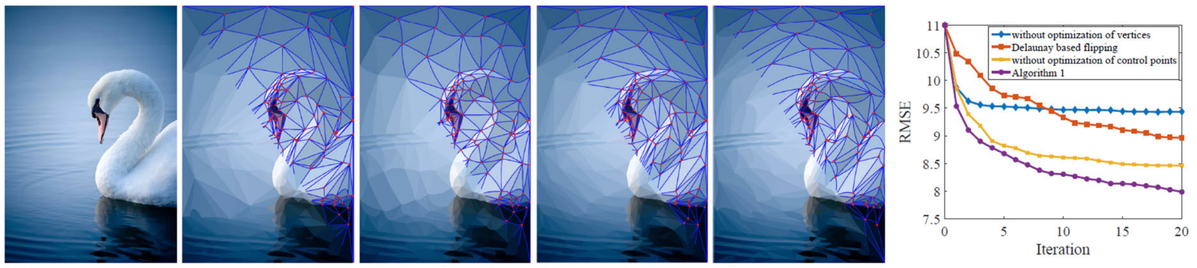
Figure 6: *Constraint polygons (light yellow) to ensure non-overlapping. Black arrows: gradient direction; green arrows: safety step length.*

are two commonly used structures for dividing the image domain. We defer our discussion of image triangulation to Section 2.2. Martinez *et al*. [MMPRQ07] and Nivoliers and Lévy [NL13] proposed a piecewise constant approximation of a given image over a Voronoi diagram, but their methods differ in terms of optimization strategies. The method proposed in Nivoliers and Lévy[NL13] was further extended by Chen *et al*. [CXC14] to piecewise polynomial approximation with arbitrary degrees, while Cao *et al*. [CXC*18] adopted barycentric coordinates to construct the approximation. They identified the optimal solution when the image features coincide with the edges of Voronoi cells, forming polylines along these features, which also inevitably occurs in other geometric partitions with straight boundaries. Another example is that Bauchet and Lafarge [BL18] generated polygons matching with image features by progressively extending pre-detected line-segments until they meet each other. However, most vectorization methods prefer



(a) RMSE=11.602 (b) RMSE=8.179 (c) RMSE=28.293 (d) RMSE=8.866 (e) RMSE=14.636 (f) RMSE=8.559 (g) RMSE=21.79 (h) RMSE=10.33

Figure 7: *Initialization with noises on vertex positions and connectivity. The underlying image in (a–h) is the original image. (a) Normal initialization; (c) initialization with random connectivity; (e/g) initializations with random movements of 10- and 50-pixel width on vertices, respectively; (b/d/f) the optimized meshes after 20 iterations of (a/c/e), respectively; (h) the optimized mesh after 50 iterations of (g).*



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

Figure 8: *The effect of different types of optimization on the results. (a) Input image; (b) the result generated by Algorithm 1 without vertex optimization; (c) the result generated by replacing error-based edge flipping with Delaunay rule-based edge flipping in Algorithm 1; (d) the result generated by Algorithm 1 without control point optimization; (e) the result generated by Algorithm 1; (f) the graphs of RMSE values of optimizations (b–e), respectively. The initial meshes are all the same with 300 vertices, and polynomial degree = 0.*

using curved geometries to align these features accurately (e.g. the curves in Refs.[LL06, OBW*08, FLB17], gradient mesh in Refs.[SLWS07, LHM09] and Bézier patches in Ref.[XLY09]). Precise alignment enables geometric objects to cover pixels without jumped colours. Therefore, we replace the straight edges of triangulation with quadratic Bézier curves to improve approximation quality.

2.2. Image triangulation

Decomposing a raster image into triangles is critical to image approximation, extraction, compression, vectorization and the community of designers, such as Favreau *et al*. [FLBA20] proposed a framework of Delaunay point processes to extract geometric structures from images. Data-dependent triangulation (DDT) [LUH07] is one of the most important methods for triangulating images. Most conventional DDT algorithms, including refinement [LA13], decimation [DDFI05] and modification [LI06], assign a colour to each vertex and reconstruct an image through interpolation over triangles. These algorithms are piecewise linear approximations with C^0 continuity at vertices and edges, hence exposing their weaknesses in expressing image discontinuities. As a result, most of the vertices are placed near both sides of the image features. To address this drawback, Tu and Adams [TA13] used a wedge structure to explicitly represent the discontinuities, whereas Lawonn and Günther [LG19] defined approximating polynomial dependently over each triangle, hence allowing multiple values on a single vertex. We extend the work of Lawonn and Günther[LG19] and similarly transform the generation of image triangulation into a minimization problem. The key difference between our work and theirs is that we equip the edges of triangulation with control points to make them curved and then deduce the explicit gradient formulas of vertices and control points instead of following the approximate computation in Lawonn and Günther[LG19]. To the best of our knowledge, only few studies have investigated curved triangulation for images. Xia *et al*. [XLY09] produced intermediate Bézier patches served for image vectorization, where they optimized Bézier edges for a positive Jacobian determinant of straight-to-curve mapping (i.e. non-overlapping) and fitting to traced pixel-level paths. In this paper, we provide a simpler and more flexible alternative for generating curved triangulation while focusing on improving reconstruction quality with a given number of vertices.

2.3. Curved meshing

Generating meshes that conform to given curves or surfaces has attracted much research attention given that curvilinear meshes can provide better numerical accuracy and efficiency in FEMs [GR09, ADF14, FP16, FAB*18, HSG*19, MC20, MC21]. For example, Feng *et al*. [FAB*18] extended optimal Delaunay triangulation [CX04] to curved meshing for higher-order basis functions. Most of these methods care more about exact expression of given curves and resultant mesh quality (e.g. large minimal angle), while we seek a low approximation error between the piecewise approximation over the triangulation and the original image. A basic requirement in curved meshing is a positive Jacobian determinant, that is a regular geometric map from a parametric triangle to physical ones, thereby suggesting that no overlapping should be observed among curved

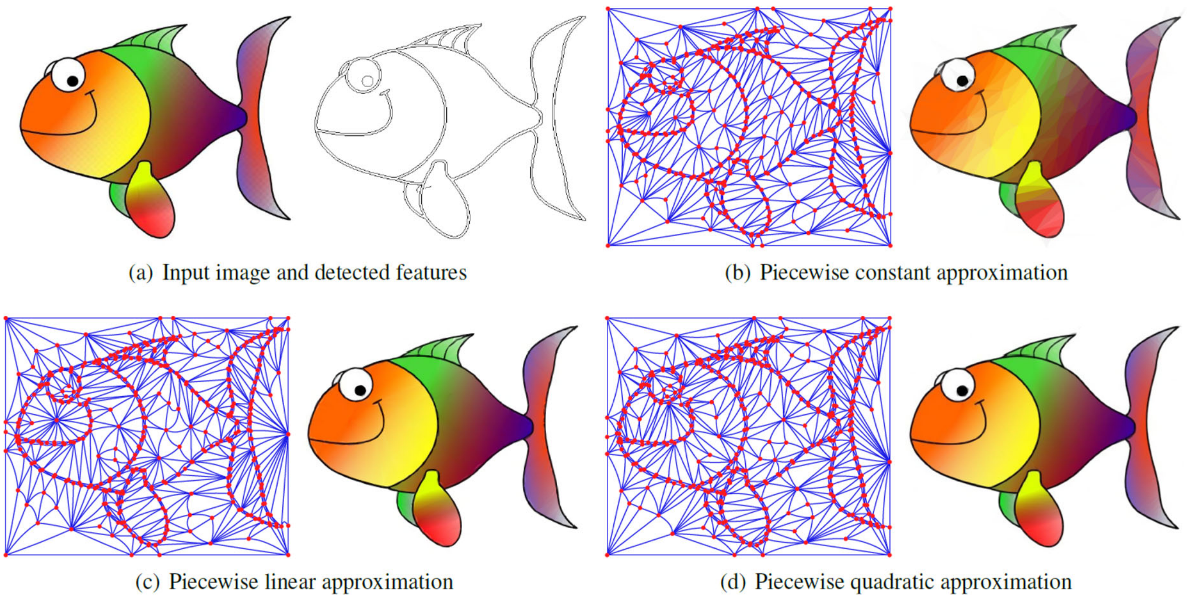

(a) Input image and detected features
(b) Piecewise constant approximation
(c) Piecewise linear approximation
(d) Piecewise quadratic approximation

Figure 9: *Approximation by piecewise polynomials with different degrees. Meshes are with 400 vertices and 783 faces. (b) RMSE = 13.125; (c) RMSE = 9.178 and (d) RMSE = 6.155.*

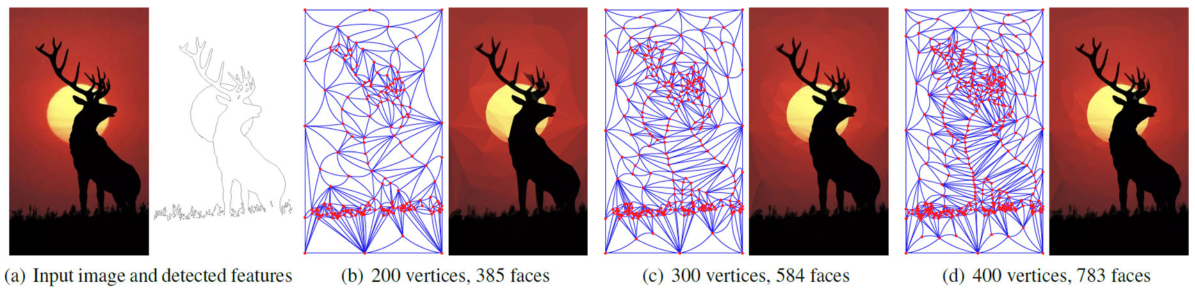

(a) Input image and detected features
(b) 200 vertices, 385 faces
(c) 300 vertices, 584 faces
(d) 400 vertices, 783 faces

Figure 10: *Approximation quality against vertex number (polynomial degree = 0). (b) RMSE = 6.998; (c) RMSE = 5.710 and (d) RMSE = 5.314.*

triangles. To ensure the intersection between curved triangles remains empty in our meshes, we constrain the movements of vertices and control points. We also perform a violation test for edge flipping, similar to Mandad and Campen[MC20].

3. Curved Optimal Triangulation

In this section, we describe the core idea of our curved optimal triangulation in detail. We initially define its energy function in Section 3.1 and then deduce the derivatives with respect to the positions of vertices and control points in Section 3.2. Given that they are presented for greyscale images, we extend all formulas of the curved optimal triangulation to colour images in Section 3.3.

3.1. Energy function

Suppose that $h : \Omega \to \mathrm{R}$ is a given function defined over the 2D domain Ω. We adopt triangulation $T = (V, F)$ to partition the domain, where $V = \{\mathbf{v}_i\}_{i=1}^{n}$, $F = \{f_k\}_{k=1}^{m}$ denote sets of n vertices and m faces, respectively. We construct piecewise approximation by computing a best-fitting polynomial P_k^* with arbitrary degree by applying the least squares method over each triangular region f_k, $k = 1, \ldots, m$, in order for the total approximation error between the given function h and the piecewise approximation to be measurable by

$$\mathcal{E}(T) = \sum_{k=1}^{m} \int_{f_k} \left| h(\mathbf{x}) - P_k^*(\mathbf{x}) \right|^2 d\mathbf{x}. \tag{1}$$

The goal here is to minimize the error function and obtain an optimal triangulation. Given that triangulation T is determined by vertices V and their connectivity E, optimizing V and E through moving vertices and flipping edges (Figures 2(a) and 2(b)) are the main operations of existing triangulation generation methods.

We replace the straight edges of the triangulation with quadratic Bézier curves. Imagine that each edge $\mathbf{v}_i\mathbf{v}_j$ of the triangulation is associated with a control point $\mathbf{c}_{ij}$ (Figure 2(c)). With the two end vertices $\mathbf{v}_i$, $\mathbf{v}_j$ and the control point $\mathbf{c}_{ij}$, the straight edge is curved to

$$\mathbf{x}_{ij}(t) = (1-t)^2\mathbf{v}_i + 2t(1-t)\mathbf{c}_{ij} + t^2\mathbf{v}_j, t \in [0, 1], \tag{2}$$

where t is the parameter.

The curved triangulation divides the domain Ω into a set of regions, with each region bounded by three curves. We still notate them as F. The partition is decided by three elements, namely, the vertex locations, connectivity and positions of control points. We then define our energy function as

$$\mathcal{E}(V, E, C) = \sum_{k=1}^{m} \int_{f_k} \left|h(\mathbf{x}) - P_k^*(\mathbf{x})\right|^2 d\mathbf{x}, \tag{3}$$

where C is the set of control points of all edges in the triangulation. C also represents the key difference between our method and those proposed in the literature. The curved optimal triangulation can be obtained by minimizing the energy function through optimizing the positions of vertices and control points, and flipping the edges iteratively (Figure 2).

Curved optimal triangulation is the global minimizer of the proposed energy. However, the energy function determined by the vertex positions and connectivity and control point positions is highly non-linear and non-convex. In practice, one usually can only compute a local minimizer of such a function. For simplicity, we will still use the term 'curved optimal triangulation' to refer to a curved triangulation given by a local minimizer of the proposed energy.

3.2. Gradient formulas

Simultaneously optimizing three types of elements is impractical. A common solution is to update one type while fixing the others in each iteration during the optimization. The gradient formulas for the positions of vertices and control points are given as follows.

3.2.1. *Gradient of vertex position*

When the connectivity and control points of edges are kept unchanged, the partition to domain Ω is affected by the positions of vertices. To avoid degenerate triangulation, each vertex can only move inside its one-ring neighbouring regions. Therefore, by applying the general Leibniz rule [Fla73], the gradient of Equation (3) with respect to vertex $\mathbf{v}_i$ can be derived as

$$\frac{\partial \mathcal{E}}{\partial \mathbf{v}_i} = \sum_{k \in F_i} \int_{f_k} \frac{\partial \left|h(\mathbf{x}) - P_k^*(\mathbf{x})\right|^2}{\partial \mathbf{v}_i} d\mathbf{x} + \sum_{j \in N_i} \int_{\widetilde{\mathbf{v}_i\mathbf{v}_j}} \Delta(\mathbf{x}) \frac{\partial \mathbf{x}}{\partial \mathbf{v}_i} \mathbf{n}(\mathbf{x}) ds, \tag{4}$$

where F_i is the indices set of the neighbouring faces of $\mathbf{v}_i$, and N_i denotes the indices set of its neighbouring vertices. Along the curved edge from $\mathbf{v}_i$ to $\mathbf{v}_j$, the regions on the left- and right-hand sides are denoted by f_{kl} and f_{kr}, respectively. Furthermore, $\Delta(\mathbf{x})$ represents the difference of approximation errors at point $\mathbf{x}$ over regions f_{kl} and f_{kr}, i.e.

$$\Delta(\mathbf{x}) = \left|h(\mathbf{x}) - P_{kl}^*(\mathbf{x})\right|^2 - \left|h(\mathbf{x}) - P_{kr}^*(\mathbf{x})\right|^2.$$

For boundary vertex $\mathbf{v}_i$, f_{kl} or f_{kr} may be null. Therefore, the corresponding approximation error is set to 0. $\mathbf{n}(\mathbf{x})$ represents the unit normal vector at point $\mathbf{x}$ outward f_{kl} (see the notations in Figure 3).

The best-fitting polynomials $\{P_k^*(\mathbf{x})\}_{k=1}^m$ can be calculated immediately by the least squares method after the triangulation is given. According to the envelope theorem [Sil99], we have

$$\int_{f_k} \frac{\partial \left|h(\mathbf{x}) - P_k^*(\mathbf{x})\right|^2}{\partial \mathbf{v}_i} d\mathbf{x} = \mathbf{0}.$$

Let us focus on the second item of the right-hand side in Equation (4). The point $\mathbf{x}$ on the curved edge $\mathbf{v}_i\mathbf{v}_j$ satisfies Equation (2). By differentiating $\mathbf{x}$ with respect to $\mathbf{v}_i$ and t, we obtain

$$\frac{\partial \mathbf{x}_{ij}(t)}{\partial \mathbf{v}_i} = (1-t)^p \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

where $p = 2$, and

$$\frac{\partial \mathbf{x}_{ij}(t)}{\partial t} = -2(1-t)\mathbf{v}_i + (2-4t)\mathbf{c}_{ij} + 2t\mathbf{v}_j.$$

Moreover, the relationship between $\mathbf{n}(\mathbf{x})$ and $\partial \mathbf{x}_{ij}(t)/\partial t$ satisfies

$$\mathbf{n}(\mathbf{x}) = \frac{(\partial \mathbf{x}_{ij}(t)/\partial t)^\perp}{\left\|\partial \mathbf{x}_{ij}(t)/\partial t\right\|},$$

where $(\cdot)^\perp$ denotes a 90° rotation in the clockwise direction. By integrating the above equations into Equation (4), the derivative of the energy function with respect to $\mathbf{v}_i$ is simplified as

$$\frac{\partial \mathcal{E}}{\partial \mathbf{v}_i} = \sum_{j \in N_i} \int_0^1 \Delta(\mathbf{x}_{ij}(t))(1-t)^p (\partial \mathbf{x}_{ij}(t)/\partial t)^\perp dt. \tag{5}$$

This equation is a general form that is suitable for vertices connected by Bézier edges in any degree p, including straight edges ($p = 1$), because we can simply replace Equation (2) with other Bézier equations.

3.2.2. *Gradient of control point position*

Since there can be no overlap between the curved triangles, the movement of each control point should be restricted, too, the details will be discussed in Section 4.3. Each control point only affects two adjacent triangles of its associated edge, and has a local influence on updating the triangulation. Similar to the above derivation, we formulate the energy function with respect to control point $\mathbf{c}_{ij}$ of edge $\mathbf{v}_i\mathbf{v}_j$ as

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial \mathbf{c}_{ij}} &= \int_{\widetilde{\mathbf{v}_i\mathbf{v}_j}} \Delta(\mathbf{x}) \frac{\partial \mathbf{x}}{\partial \mathbf{c}_{ij}} \mathbf{n}(\mathbf{x}) ds \\ &= \int_0^1 \Delta(\mathbf{x}_{ij}(t)) 2t(1-t)(\partial \mathbf{x}_{ij}(t)/\partial t)^\perp dt. \end{aligned} \tag{6}$$

The notations here are similarly defined as those in Equation (5).

3.3. Extension to colour images

Given that we aim to generate curved optimal triangulation for images, we reformulate the above equations with minor modifications.

Equation (3) can be applied directly to greyscale images. For a colour image, each of its colour channels is treated as a function to be approximated simultaneously over the curved triangulation. We take RGB images as an example. The other modes can be handled analogically in the same way. First, the energy function is modified to

$$\mathcal{E}(V,E,C) = \sum_{k=1}^{m} \int_{f_k} (|r(\mathbf{x}) - R_k^*(\mathbf{x})|^2 + |g(\mathbf{x}) - G_k^*(\mathbf{x})|^2 + |b(\mathbf{x}) - B_k^*(\mathbf{x})|^2) d\mathbf{x}, \quad (7)$$

where $r(\mathbf{x})$, $g(\mathbf{x})$ and $b(\mathbf{x})$ denote the red, green and blue colours in the input image, respectively, whereas $R_k^*(\mathbf{x})$, $G_k^*(\mathbf{x})$ and $B_k^*(\mathbf{x})$ denote the corresponding best-fitting polynomials over the curved triangle region f_k. Second, the gradient formulas can be derived in the same fashion as that described in Section 3.2. Actually, the only variable that needs to be modified here is $\Delta(\mathbf{x})$, that is

$$\begin{aligned}\Delta(\mathbf{x}) &= (|r(\mathbf{x}) - R_{kl}^*(\mathbf{x})|^2 + |g(\mathbf{x}) - G_{kl}^*(\mathbf{x})|^2 + |b(\mathbf{x}) - B_{kl}^*(\mathbf{x})|^2) \\ &\quad -(|r(\mathbf{x}) - R_{kr}^*(\mathbf{x})|^2 + |g(\mathbf{x}) - G_{kr}^*(\mathbf{x})|^2 + |b(\mathbf{x}) - B_{kr}^*(\mathbf{x})|^2).\end{aligned}$$

4. Algorithm

Although the energy function of curved optimal triangulation is well-defined, how to quickly find its minimum presents a challenge because of its non-linear property, especially for input images with complicated contents. In this section, we develop an optimization framework to generate satisfying results. The algorithm overview is initially presented, then the details of initialization and optimization are explained later.

4.1. Overview

We iteratively optimize the vertices, connectivity and control points of the triangulation in sequence. We update the vertices and control points using a gradient-based method and optimize connectivity by performing edge flipping tests. Both the gradient-based method and edge flipping tests are guided by the rules of energy decrease and overlapping rejection. Additional details on the optimization can be found in Section 4.3. The optimization of control points should be performed after optimizing connectivity in each iteration, given that aligning the edge to the feature is more important, otherwise optimizing its control point is meaningless.

The optimization requires an initial triangulation, which can be generated in various ways, such as the Delaunay triangulation of random samples. A good initial guess can accelerate the convergence of a non-linear optimization problem. In our case, the optimal solution is that a subset of vertices and curved edges converge to image features in the results, given that such triangulation divides the image into several regions with similar colours, thereby leading to a lower approximation error. Therefore, a better choice is to initialize triangulation based on the detected features of the input image, which places part of the vertices along each feature curve. This feature-assisted initialization has great advantage in quickly searching for the minimum of the energy function. Additional details can be found in Section 4.2.

With a good initialization, the optimization can generate results with high approximation quality after a few iterations. The pseudo-code of the algorithm is listed in Algorithm 1, and Figure 4 shows an example of the pipeline.

Algorithm 1 Generation of Curved Optimal Triangulation

```
Input: image, number of vertices $n$, and terminating conditions;
Output: curved optimal triangulation $T$;
 1: features detection;
 2: feature-assisted initialization of triangulation $T$;
 3: compute best approximating polynomials for all faces;
 4: for each internal edge $\mathbf{v}_i\mathbf{v}_j$ do
 5:     $\mathbf{c}_{ij} \leftarrow (\mathbf{v}_i + \mathbf{v}_j)/2$;
 6: end for
 7: while not meet terminating conditions do
 8:     for each vertex $\mathbf{v}_i$ do
 9:         $\mathbf{v}_i \leftarrow \mathbf{v}_i - \alpha_i^* \frac{\partial\mathcal{E}/\partial\mathbf{v}_i}{\|\partial\mathcal{E}/\partial\mathbf{v}_i\|}$;
10:     end for
11:     for each internal edge $\mathbf{v}_a\mathbf{v}_c$ do
12:         if energy decreases after flipping $\mathbf{v}_a\mathbf{v}_c$ then
13:             delete $\mathbf{c}_{ac}$;
14:             flip $\mathbf{v}_a\mathbf{v}_c$ to $\mathbf{v}_b\mathbf{v}_d$;
15:             $\mathbf{c}_{bd} \leftarrow (\mathbf{v}_b + \mathbf{v}_d)/2$;
16:         end if
17:     end for
18:     for each internal edge $\mathbf{v}_i\mathbf{v}_j$ do
19:         $\mathbf{c}_{ij} \leftarrow \mathbf{c}_{ij} - \beta_{ij}^* \frac{\partial\mathcal{E}/\partial\mathbf{c}_{ij}}{\|\partial\mathcal{E}/\partial\mathbf{c}_{ij}\|}$;
20:     end for
21:     update approximating polynomials;
22: end while
```

4.2. Initialization

Error-based initialization is a common strategy, which starts from a coarse triangulation and fine-tunes the results by iteratively inserting a new vertex into the triangle with maximum approximation error [CXC14, XCC*18]. In this case, regardless of its number of iterations, the optimization can easily get stuck in the local minima for most testing images, especially for those features that are located close to one another (Figures 5(b) and 5(c)). Therefore, this initialization does not help much achieving minimum, given that each vertex is not inserted under an optimal triangulation, thereby gathering an excessive number of vertices around the same regions. In addition, most vertices adhere to their nearby features in the local optimum, and only few of them move to important features located elsewhere.

Therefore, distributing a moderate number of vertices along each feature curve is more reasonable. Although there have been many triangulation generation methods, their results are not suitable as initialization for our optimization. As mentioned, using meshes with only straight edges to approximate images usually requires dense vertices along curved features. However, the triangulation with curved edges can represent those features by using fewer vertices. Hence, more vertices can be placed elsewhere with high

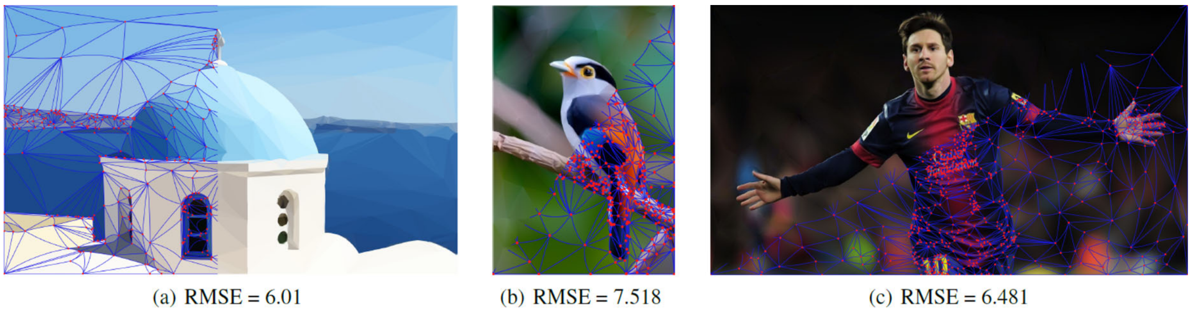


Figure 11: *More results (meshes and reconstruction results) of our method. (a) 500 vertices, polynomial degree = 0; (b) 500 vertices, polynomial degree = 1; (c) 1200 vertices, polynomial degree = 2.*

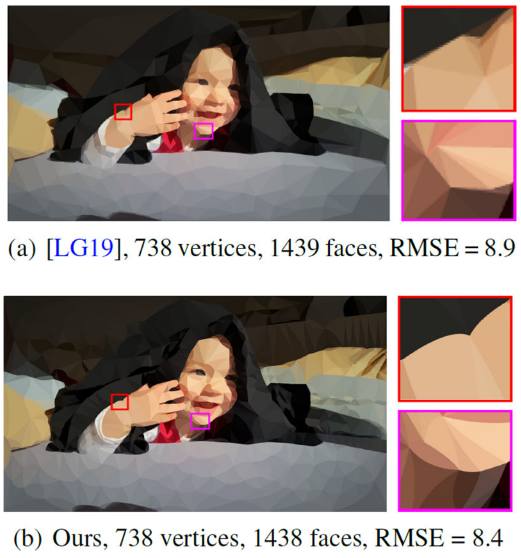

(a) [LG19], 738 vertices, 1439 faces, RMSE = 8.9
(b) Ours, 738 vertices, 1438 faces, RMSE = 8.4

Figure 12: *Comparison with Lawonn and [LG19], polynomial degree = 0.*

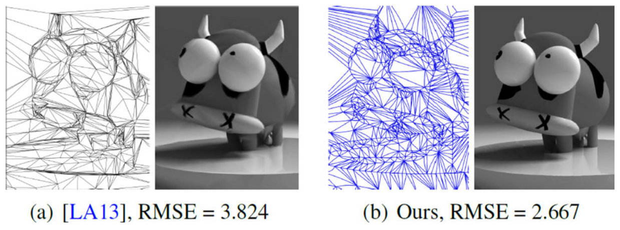


Figure 13: *Comparison with Li and Adams[LA13], 980 vertices, polynomial degree = 1.*

approximation errors, significantly increasing the approximation quality of final results.

In this paper, we prefer to initialize triangulation with the assistance of detecting features of the input image by the following four steps:

Step 1 (Feature extraction).We extract image features (see Figure 5(d)) by using the edge drawing (ED) method [TA12]. Unlike other feature detection methods (e.g. the Canny method [Can86]), which may yield feature lines with non-uniform thickness, the ED method produces one-pixel-wide pixel paths, simplifying the following process.

Step 2 (Feature simplification).We simplify each pixel path to a polyline with fewer vertices using the Douglas–Peucker algorithm [DP73]. In particular, to simply a pixel path from p_i to p_j, we start with a line segment $p_i p_j$. If the distance of the farthest pixel on the path from this segment is smaller than a threshold (5-pixel width), then we accept this simplification.

Step 3 (Constrained Delaunay triangulation).We generate a Delaunay triangulation by taking all the line segments generated in Step 2 as constraints.

Step 4 (Triangulation updating).We alternately insert a vertex at the circumcentre of the triangle with the largest area and accumulate approximation error until the vertex number budget is reached.

Figure 5(e) illustrates a feature-assisted initialization, which significantly improves approximation quality compared with error-based initialization (Figure 5(b)). After the optimization with only 10 iterations, the final curved triangulation can precisely recover the input image (Figure 5(f)).

4.3. Optimization

We construct the best approximating polynomials for all faces of the triangulation by using the least squares method and update them iteratively during the optimization. We also initialize control points as middle points of all internal edges. Afterward, the triangulation is ready to be optimized.

Vertices. The traditional gradient descent method updates all variables with same step length. We choose to update vertices one by one instead of moving them simultaneously to prevent degeneration. In other words, each vertex cannot be moved outside its one-ring neighbouring regions. In this case, we give the step length individually. Specifically, except for the four corners, each vertex $\mathbf{v}_i$ is updated as

$$\mathbf{v}_i = \mathbf{v}_i - \alpha_i^* \frac{\partial \mathcal{E}/\partial \mathbf{v}_i}{\|\partial \mathcal{E}/\partial \mathbf{v}_i\|},$$

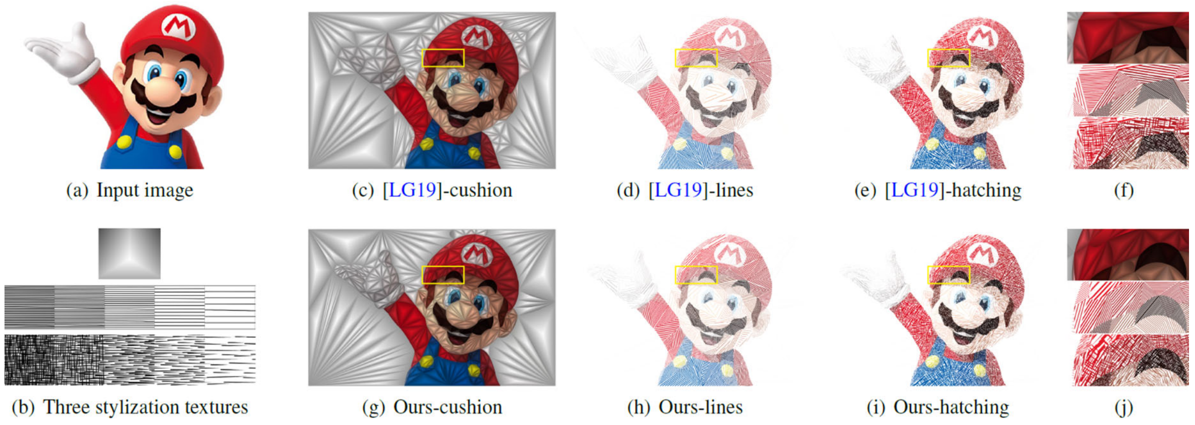


Figure 14: *Comparison of image stylizations with Lawonn and Günther [LG19]. Both triangulations have 300 vertices. (b) Textures from top to bottom: cushion, lines and hatching; (f) zoomed-in details of (c–e) and (j) zoomed-in details of (g–i).*

where α_i^* is the step length decided by the following two steps:

1. To avoid self-intersection in the triangulation, we introduce a constraint polygon (the light yellow region in Figure 6(a)) for $\mathbf{v}_i$ to restrict its movement. As shown in Figure 6(a), the constraint polygon of $\mathbf{v}_i$ is first set as the 1-ring neighbourhood of $\mathbf{v}_i$ (the grey region in Figure 6(a)), then in each adjacent triangle $\triangle\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k$, the constraint polygon is clipped by six lines $\mathbf{v}_j\mathbf{c}_{ij}$, $\mathbf{v}_j\mathbf{c}_{jk}$, $\mathbf{v}_j\mathbf{c}_{ki}$, $\mathbf{v}_k\mathbf{c}_{ij}$, $\mathbf{v}_k\mathbf{c}_{jk}$ and $\mathbf{v}_k\mathbf{c}_{ki}$, and the part on the same side of the lines as $\mathbf{v}_i$ is retained. From the construction of the constraint polygon, we know that the vertex $\mathbf{v}_i$ stays on the same side of the six clipping lines after the optimization, which prevents overlaps between the control triangles $\triangle\mathbf{v}_i\mathbf{v}_j\mathbf{c}_{ij}$, $\triangle\mathbf{v}_j\mathbf{v}_k\mathbf{c}_{jk}$ and $\triangle\mathbf{v}_k\mathbf{v}_i\mathbf{c}_{ki}$. Thus we guarantee that there is no intersection between all adjacent curved edges of $\mathbf{v}_i$ except the end points.
2. Along the direction $-(\partial\mathcal{E}/\partial\mathbf{v}_i)/\|\partial\mathcal{E}/\partial\mathbf{v}_i\|$, the safety length α_i (green arrow in Figure 6(a)) is calculated. We then try to update $\mathbf{v}_i$ with $\alpha_i^* = 0.2\alpha_i$. If the energy decreases, then we accept the step length. Otherwise, we scale α_i^* by 0.2 and try again. For better efficiency, we limit the maximum search count to 5 by default. Due to approximate integration involved in our computation, we consider the failure of the line search as a numerical issue. And we heuristically set α_i^* to 0.2-pixel width when line search fails to prevent getting stuck at a bad local minimum, which enables us to obtain a better result in the next iteration.

For boundary vertices, we project their gradients to boundaries to ensure that the triangulation completely covers the image domain after moving them. And for an interior vertex, if there exists an adjacent triangle, one of whose interior angles is opposite to a boundary edge and is greater than 175°, then it will be handled as a new boundary vertex.

Connectivity. We take the commonly used operation (i.e. edge flipping) to optimize connectivity based on error decrease. Each internal edge $\mathbf{v}_a\mathbf{v}_c$ is adjacent to two curved triangles $\triangle\mathbf{v}_a\mathbf{v}_b\mathbf{v}_c$ and $\triangle\mathbf{v}_a\mathbf{v}_c\mathbf{v}_d$. We flip the curved edge $\mathbf{v}_a\mathbf{v}_c$ to the straight edge $\mathbf{v}_b\mathbf{v}_d$. To prevent overlapping, segment $\mathbf{v}_b\mathbf{v}_d$ cannot intersect with any control segment of the neighbouring edges of $\mathbf{v}_a\mathbf{v}_c$. If so, and when the approximation error is reduced after the flipping, i.e. $\mathcal{E}(\{\triangle\mathbf{v}_a\mathbf{v}_b\mathbf{v}_c, \triangle\mathbf{v}_a\mathbf{v}_c\mathbf{v}_d\}) > \mathcal{E}(\{\triangle\mathbf{v}_b\mathbf{v}_c\mathbf{v}_d, \triangle\mathbf{v}_b\mathbf{v}_d\mathbf{v}_a\})$, then the flipping is accepted, and the control point of the new edge is reset to the middle point of its two end vertices.

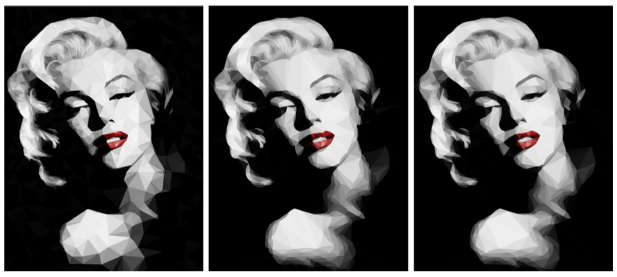


Figure 15: *Comparison of Lowpoly generation with Gai and Wang [GW16]. (a) Result of Gai and Wang [GW16]; (b) our result generated by Algorithm 1; (c) our result generated by Algorithm 1 without control points optimization of normal edges.*

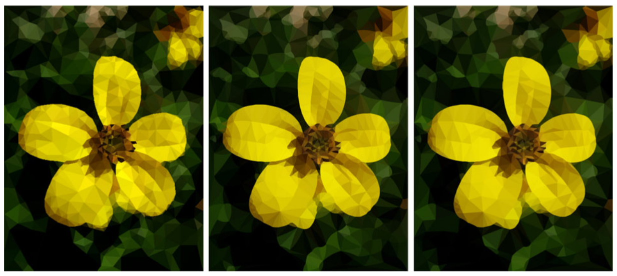


Figure 16: *Comparison of Lowpoly generation with Ma and Chen [MC17]. (a) Result of Ma and Chen [MC17]; (b) our result generated by Algorithm 1; (c) our result generated by Algorithm 1 without control points optimization of normal edges.*

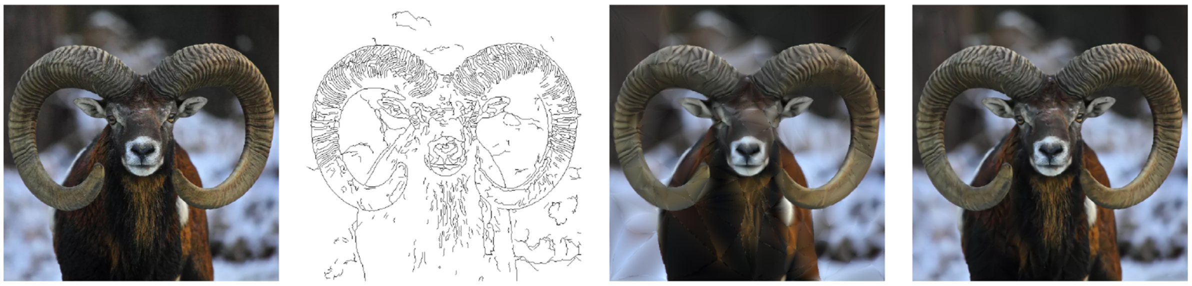

(a) Input image
(b) Detected features
(c) 300 vertices, RMSE = 10.239
(d) 3000 vertices, RMSE = 6.185

Figure 17: *Piecewise quadratic approximation results of an image with many subtle features. The eyes of argali in (c) are extremely distorted due to the insufficient number of samples.*

Control points. Similar to vertex updating, each control point $\mathbf{c}_{ij}$ associated with internal edge $\mathbf{v}_i\mathbf{v}_j$ is updated as

$$\mathbf{c}_{ij} = \mathbf{c}_{ij} - \beta^*_{ij}\frac{\partial\mathcal{E}/\partial\mathbf{c}_{ij}}{\|\partial\mathcal{E}/\partial\mathbf{c}_{ij}\|},$$

where β^*_{ij} means its step length that is searched in the same way as that in the vertices optimization. Differently, its constraint polygon is obtained by clipping the quadrilateral $\diamond\mathbf{v}_i\mathbf{v}_u\mathbf{v}_j\mathbf{v}_k$ with two lines $\mathbf{v}_j\mathbf{c}_{jk}$ and $\mathbf{v}_i\mathbf{c}_{ki}$ in each adjacent triangle $\triangle\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k$ (see Figure 6(b)).

Terminating conditions. Users can provide two terminating conditions for the optimization, one of which is the maximum iteration number $Iter$, and the other is an approximation quality metric, such as root mean squared error (RMSE). These conditions are set to 10 and 1.0 by default, respectively. The algorithm terminates when the iteration exceeds $Iter$ or when the RMSE is below a given threshold. Benefiting from the feature-assisted initialization, the optimization can generate satisfactory results in a few iterations.

5. Results and Applications

In this section, we demonstrate the effectiveness of our curved optimal triangulation method through several experiments and comparisons with other methods, and then explore its applications in the automatic generation of stylization and hybrid Lowpoly images. The polynomial bases $\{1, x, y, xy, x^2, y^2\}$, $\{1, x, y\}$ and $\{1\}$ are used for piecewise quadratic, linear and constant approximations, respectively. We implement the algorithm using C++ and discretize each curved edge to at least eight segments for integral computations, such as approximating polynomials and gradients. We use the Triangle library [She96] to generate constrained Delaunay triangulation in the initialization. All results are obtained using a laptop with a 1.6-GHz Intel Core I5 processor and 16-GB RAM. We tested our method for all examples in both Lab and RGB colour spaces and obtained very similar results. For simplicity, we only show the results obtained in the RGB colour space.

5.1. Results and comparisons

We use RMSE to quantify approximation quality. RMSE measures the distance between the input image and piecewise approximations over the triangulation and can be calculated as $\sqrt{\mathcal{E}(T)/(W \times H)}$, where W and H are image width and height, respectively.

The curved edges of each final triangulation appear anywhere in most examples because we do not distinguish whether or not these edges are related to the features. The control points of normal edges can be optimized earlier than those of edges relating to the features. However, according to user requirements, we can tag edges if they are related to the features in the initialization (i.e. feature edges and normal edges) and either skip or continue optimizing the control points of normal edges.

We first show the importance of the initial triangulation by introducing some noises on vertex positions and connectivity in the initialization stage, see Figure 7. Compared to Figure 7(a,b), the results of Figure 7(c,d) indicate that random connectivity changes on the initial mesh has a slight influence on the final result, since we adopt the error-based edge flipping during the optimization. In contrast, the initial vertex positions have a greater impact on the final approximation error. Our optimization is capable to deal with slight deviations of initial vertex positions from image features, see Figure 7(e,f), but fails to obtain a satisfactory triangulation even after many iterations when initial vertices deviate heavily from the features, see Figure 7(g,h).

We also test the influences of the three operations, i.e. vertex move, control point move and edge flipping, on optimization results. From the same initial triangulation, Figure 8(b,d) is generated by Algorithm 1 without optimization of vertices and control points, respectively, while Figure 8(c) is obtained by replacing the error-based flipping with Delaunay-based flipping. All the three operations have a positive effect on the improvement of approximation quality, which can be shown by RMSE during the iterations. The vertex move plays a significant role for the optimization. As shown in Figure 8(f), RMSE drops slowest after disabling the optimization of vertices.

Figure 9 shows an example where the original image is approximated using piecewise polynomials with different degrees. The output triangulations have 400 vertices (about 0.15% sample rate) and 783 faces. From the visual effect and RMSE values of the reconstruction results, the approximation quality can be significantly improved by using a high-order polynomial basis. Although the colour of the original image changes dramatically, our linear approximation recovers the colour variations well enough (Figure 9(c)).

Figure 10 shows the influence of vertex number on the results. Apparently, the triangulation with more vertices increases the approximation quality, because the image can be divided into more regions and approximated more accurately over each region. However, the speed of such increase gradually decreases after all image features are captured well enough by the curved triangulation (Figures 10(c) and 10(d)), where the contribution of 100 new vertices is less than 0.4 RMSE. More of our results can be found in Figure 11.

Figures 1, 12 and 13 compare the triangulations with curved and straight edges. The triangulation in Figure 1(b) is generated by the proposed algorithm without control points optimization, hence resulting in several vertices deviating from the features, even though they are initially sampled from the detected features. By contrast, the optimizations of vertices and control points positively affect one another in our curved triangulation (the zoomed-in details in the figure). We also compare our approach with other state-of-the-art triangulation generation methods. For instance, Figure 12 compares our approach with Lawonn and Günther[LG19]. The main difference between their energy function and ours is the control points, and their optimization framework iteratively inserts one vertex and optimizes all of them, which requires many iterations. Our result looks more pleasant than that of Lawonn and Günther[LG19], because we recover the curved features in large quantities, such as the wrist and chin of the baby as shown in the figure. Meanwhile, Figure 13 compares our approach with Li and Adams[LA13], and also demonstrates the great advantage of our curved optimal triangulation in image representation.

5.2. Stylization

The demand for triangulating images is increasing in designer and artist communities [Bit14, Bry17]. We demonstrate the applications of our method on image stylization in this section and on hybrid Lowpoly generation in the following section. For the former, we directly follow the concept of Lawonn and Günther[LG19] and apply stylization texture on each curved triangular region. We assign the texture coordinates of the vertices and control points of a target region in two ways. First, we set a base hexagon in the texture domain and copy its coordinates to the vertices and control points of the target region, which is used for the top texture shown in Figure 14(b). Second, we scale the bounding box of the region to the texture domain, which is used for middle and bottom textures shown in Figure 14(b). We also allow users to provide multiple patterns in a single texture to simulate changes in colour brightness, which is achieved via a proper selection of texture coordinates.

In Figure 14, we generate three stylization images using the given textures (Figure 14(b)), cushion, lines and hatching [PHWF01], respectively, and compare our results (Figure 14(g–i)) with those of Lawonn and Günther[LG19] (Figure 14(c–e)). Their zoomed-in details can be found in Figures 14(f) and 14(j), respectively. We set the same texture coordinates for the results using the same texture to eliminate the influence of texture coordinates. Therefore, the results in Figure 14(c–e) may differ from that of the original version of Lawonn and Günther[LG19]. Our results are more faithful to the curved objects in original image, see the eyebrow and hat for instance.

5.3. Hybrid lowpoly

Several methods focus on the automatic generation of Lowpoly images, such as Gai and Wang[GW16] and Ma and Chen [MC17]. A conventional Lowpoly image triangulate with only straight edges and fills constant colour in each triangle for the given image. However, artifacts may be generated due to the inaccurate representation of the boundaries of curved objects (see the petals in Figure 16(a) for example).

We provide users an alternative approach for generating hybird Lowpoly images represented by curved triangulation. The degree of approximating polynomials is set to 0 in this application. We generate two versions of our results shown in Figsures 15 and 16, one generated by Algorithm 1 and the other generated by Algorithm 1 without control points optimization of normal edges.

Figure 15 compares our results with that of Gai and Wang [GW16], who initialized constrained Delaunay triangulation with the aid of features detection, too, but optimized non-critical vertices using the centroidal Voronoi tessellation method [DFG99, LWL*09], which leads to more uniform triangles in their Lowpoly result (Figure 15(a)). By contrast, we focus on approximation quality, and our resultant Lowpoly images are closer to the input (Figure 15(b,c)). Ma and Chen [MC17] developed an interactive system for Lowpoly rendering, and one of their results is shown in Figure 16(a). They achieved triangulation by using an adaptive thinning method, which tends to generate small thin and long triangles around the image features. Our results shown in Figure 16(b,c) have clearer and more natural image features, such as petal boundaries, compared with those of Ma and Chen[MC17]. As both Gai and Wang [GW16] and Ma and Chen [MC17] did not report the vertex number of their resultant triangulations, we only present here the similar results to theirs for comparison.

5.4. Timing statistics

The proposed algorithm usually runs in a few seconds, depending on the image size, amount of vertices and iteration number. Having more image pixels, vertices and iterations usually requires a longer running time. A high-order polynomial basis also significantly affects the efficiency, given that a higher-dimensional linear system needs to be solved in each region of the triangulation if the polynomial degree is high. The running times of most examples presented in this paper are listed in Table 1, where the columns represent the figure index, image size, degree of approximating polynomials, vertex number, iteration number and time in seconds, respectively.

Our curved triangulation-based vector graphics can be efficiently rendered using a GPU. In particular, we create a shader programme for all curved triangles and upload the coefficients of their approximating polynomials to the fragment shader. The rendering can reach above 52 frames per second for all examples in the paper, making the rendering real-time.

Table 1: *Statistics of running time.*

Fig.	$W \times H$	Degree	#Vertices	#Iter	Time(s)
1	500×415	1	200	10	2.9
4	300×710	1	300	10	3.4
5	1024×1024	0	120	10	3.7
7(b/d/f)	800×800	0	200	20	4.1
7(h)	800×800	0	200	50	8.8
8	367×550	0	300	20	3.1
9(b)	560×470	0	400	15	2.8
9(c)	560×470	1	400	15	5.5
9(d)	560×470	2	400	15	9.6
10(b)	640×1136	0	200	12	3.5
10(c)	640×1136	0	300	12	3.9
10(d)	640×1136	0	400	12	4.1
11(a)	2025×1195	0	500	15	9.2
11(b)	530×780	1	500	15	7.5
11(c)	2048×1152	2	1200	15	42.6
12(b)	1280×720	0	738	10	4.4
13(b)	1024×768	1	980	10	6.8
14	512×325	0	300	20	3.8
15(b)	600×800	0	600	10	4.1
16(b)	492×656	0	600	10	2.6
17(c)	640×640	2	300	15	12.6
17(d)	640×640	2	3000	15	38.2

6. Conclusions and future work

We propose a curved optimal triangulation corresponding to the minimal of the energy function (Equation (3)). We introduce control points for triangulation edges as an important factor for partitioning a given image. We also deduce explicit gradient formulas for the positions of vertices (Equation (5)) and control points (Equation (6)) that enable us to search for the minimum of the energy function efficiently. We develop a generation algorithm of curved optimal triangulation under the guidance of energy decrease and non-overlapping. To make the optimization start from a good point, we use the results of features detection for the initialization. We demonstrate the effectiveness of our proposed algorithm through several experiments and through comparisons with other methods and applications.

Limitations and future work. The curved optimal triangulation captures image features piece by piece, hence the continuity at the joint point (i.e. vertex) between curved edges is only C^0, which may be unacceptable for some applications that require high continuities. A promising solution to this problem is to add more constraints on the control points, such as the collineation of a vertex and the two control points of its neighbouring feature edges, which we will leave for a future work. Furthermore, although our method is robust, if the input image contains many subtle features (e.g. grassland and feather), then a larger number of vertices is needed to achieve a satisfactory reconstruction (see Figure 17 for an example). How to initialize triangulation based on the priority of detected features can be considered in future studies. It should be pointed out that the curved triangulation results are obtained with the measure of colour difference between pixels in this paper, and we are also interested in investigating the proposed method with other perceptual metrics. Finally, another exploration direction is to improve the regularity of curved triangles, which directly correlates to the aesthetic beauty of the final result.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. The research of Yanyang Xiao was supported by the National Natural Science Foundation of China (No. 62102174). The research of Juan Cao and Zhonggui Chen was supported by the National Natural Science Foundation of China (Nos. 61872308 and 61972327), the Natural Science Foundation of Fujian Province of China (No. 2019J01026), the Fundamental Research Funds for the Central Universities (Nos. 20720190011 and 20720190063), the Xiamen Youth Innovation Funds (No. 3502Z20206029) and the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No. VRLAB2021B01).

References

[ADF14] ABGRALL R., DOBRZYNSKI C., FROEHLY A.: A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems. *International Journal for Numerical Methods in Fluids 76*, 4 (2014), 246–266.

[Aur91] AURENHAMMER F.: Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys 23*, 3 (Sept. 1991), 345–405.

[BE92] BERN M., EPPSTEIN D.: Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry*. World Scientific, Singapore (1992), pp. 23–90.

[Bit14] BITENCOURT B.: https://dribbble.com/Bitencourt (2014). Accessed: 2014-10-05

[BL18] BAUCHET J.-P., LAFARGE F.: KIPPI: KInetic polygonal partitioning of images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, USA, June 2018).

[Bry17] BRYAN A.: Hand-drawn image triangulations. https://www.joshbryanart.com/triangulations (2017). Accessed: 2017-12-23

[Can86] CANNY J.: A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8, 6* (1986) 679–698.

[CX04] CHEN L., XU J.: Optimal Delaunay triangulations. *Journal of Computational Mathematics 22*, 2 (2004), 299–308.

[CXC14] CHEN Z., XIAO Y., CAO J.: Approximation by piecewise polynomials on Voronoi tessellation. *Graphical Models 76*, 5 (2014), 522–531.

[CXC*18] CAO J., XIAO Y., CHEN Z., WANG W., BAJAJE C.: Functional data approximation on bounded domains using polygonal finite elements. *Computer Aided Geometric Design 63* (2018), 149–163.

[DDFI05] Demaret L., Dyn N., Floater M. S., Iske A.: Adaptive thinning for terrain modelling and image compression. In *Advances in Multiresolution for Geometric Modelling*. N. A. Dodgson, M. S. Floater and M. A. Sabin (Eds.). Springer, Berlin, Heidelberg (2005), pp. 319–338.

[DFG99] Du Q., Faber V., Gunzburger M.: Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review 41*, 4 (1999), 637–676.

[DP73] Douglas D. H., Peucker T. K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*. (1973) *10*(2), 112–122.

[FAB*18] Feng L., Alliez P., Busé L., Delingette H., Desbrun M.: Curved optimal Delaunay triangulation. *ACM Transactions on Graphics 37*, 4 (July 2018), 1–16.

[Fla73] Flanders H.: Differentiation under the integral sign. *American Mathematical Monthly 80*, 6 (1973), 615–627.

[FLB17] Favreau J.-D., Lafarge F., Bousseau A.: Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM Transactions on Graphics 36*, 6 (Nov. 2017), 1–11.

[FLBA20] Favreau J.-D., Lafarge F., Bousseau A., Auvolat A.: Extracting geometric structures in images with Delaunay point processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 42*, 4 (2020), 837–850.

[FP16] Fortunato M., Persson P.-O.: High-order unstructured curved mesh generation using the winslow equations. *Journal of Computational Physics 307* (2016), 1–14.

[GR09] Geuzaine C., Remacle J.-F.: Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering 79*, 11 (2009), 1309–1331.

[GW16] Gai M., Wang G.: Artistic low poly rendering for images. *The Visual Computer 32* (2016), 491–500.

[HSG*19] Hu Y., Schneider T., Gao X., Zhou Q., Jacobson A., Zorin D., Panozzo D.: Triwild: Robust triangulation with curve constraints. *ACM Transactions on Graphics 38*, 4 (July 2019), 1–15.

[LA13] Li P., Adams M. D.: A tuned mesh-generation strategy for image representation based on data-dependent triangulation. *IEEE Transactions on Image Processing 22*, 5 (2013), 2004–2018.

[LG19] Lawonn K., Günther T.: Stylized image triangulation. *Computer Graphics Forum 38*, 1 (2019), 221–234.

[LHM09] Lai Y.-K., Hu S.-M., Martin R. R.: Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transactions on Graphics 28*, 3 (July 2009), 1–8.

[LI06] Laurent D., Iske A.: Adaptive image approximation by linear splines over locally optimal Delaunay triangulations. *IEEE Signal Processing Letters 13* (June 2006), 281–284.

[LL06] Lecot G., Lévy B.: Ardeco: Automatic region detection and conversion. In *Proceedings of the EGSR'06*, Eurographics Association, pp. 349–360.

[LUH07] Lehner B., Umlauf G. & Hamann B.: Survey of techniques for data-dependent triangulations. Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI) (Kaiserslautern, Germany, 2007), pp.178–187.

[LWL*09] Liu Y., Wang W., Lévy B., Sun F., Yan D.-M., Lu L., Yang C.: On centroidal Voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics* (2009). Presented at SIGGRAPH 2010.

[MC17] Ma Y., Chen X.: An interactive system for Low-poly illustration generation from images using adaptive thinning. In proceedings of the IEEE International Conference on Multimedia and Expo (Hong Kong, China, 2017), pp. 1033–1038.

[MC20] Mandad M., Campen M.: Bézier guarding: Precise higher-order meshing of curved 2D domains. *ACM Transactions on Graphics 39*, 4 (July 2020), 103:1–103:15.

[MC21] Mandad M., Campen M.: Guaranteed-quality higher-order triangular meshing of 2D domains. *ACM Transactions on Graphics 40*, 4 (July 2021), 1–14.

[MMPRQ07] Martinez A., Martinez J., Pérez-Rosés H., Quirós R.: Image processing using Voronoi diagrams. In *Proceedings of the Conference on Image Processing, Computer Vision, and Pattern Recognition*, pp. 485–491.

[NL13] Nivoliers V., Lévy B.: Approximating functions on a mesh with restricted Voronoi diagrams. *Computer Graphics Forum 32*, 5 (2013), 83–92.

[OBW*08] Orzan A., Bousseau A., Winnemöller H., Barla P., Thollot J. & Salesin D.: Diffusion curves: A vector representation for smooth-shaded images. ACM Transactions on Graphics 27, 3 (2008), 1–8.

[PHWF01] Praun E., Hoppe H., Webb M., Finkelstein A.: Real-time hatching. In *SIGGRAPH'01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), Association for Computing Machinery, pp. 581.

[She96] Shewchuk J. R.: Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering, Lecture Notes in Computer Science* (vol. 1148). M. C. Lin and D. Manocha (Eds.). Springer-Verlag (May 1996), pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[Sil99] Silberberg E.: The Viner–Wong envelope theorem. *The Journal of Economic Education 30*, 1 (1999), 75–79.

[SLWS07] Sun J., Liang L., Wen F., Shum H.-Y.: Image vectorization using optimized gradient meshes. In *Proceedings of the SIGGRAPH '07*, Association for Computing Machinery, pp. 11.

[TA12] Topal C., Akinlar C.: Edge drawing: A combined real-time edge and segment detector. *Journal of Visual Communication and Image Representation 23*, 6 (2012), 862–872.

[TA13] Tu X., Adams M.: Improved mesh models of images through the explicit representation of discontinuities. *Canadian Journal of Electrical and Computer Engineering 36*, 2 (2013), 78–86.

[XCC*18] Xiao Y., Chen Z., Cao J., Zhang Y. J., Wang C.: Optimal power diagrams via function approximation. *Computer-Aided Design 102* (2018), 52–60.

[XLY09] Xia T., Liao B., Yu Y.: Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 1–10.

[ZZW14] Zhou H., Zheng J., Wei L.: Representing images using curvilinear feature driven subdivision surfaces. *IEEE Transactions on Image Processing 23*, 8 (2014), 3268–3280.