Hindawi
Security and Communication Networks
Volume 2020, Article ID 8871588, 13 pages
https://doi.org/10.1155/2020/8871588

Research Article

A Novel Approach to Transform Bitmap to Vector Image for Data Reliable Visualization considering the Image Triangulation and Color Selection

Zhike Han,[1] **Xiuchao Wu**,[1] **Meng Chi**,[1] **Jun Tang**,[2] **and Lijing Yang**[2]

[1]*College of Computer Science and Technology, Zhejiang University, 38 Zheda Road, Hangzhou, China*
[2]*Hangzhou Health Information Center, Hangzhou, China*

Correspondence should be addressed to Meng Chi; chimeng089@gmail.com

Received 5 July 2020; Revised 22 September 2020; Accepted 21 October 2020; Published 2 November 2020

Academic Editor: Honghao Gao

Vector image is a type of image composed of many geometric primitives. Compared with bitmaps, vector images have the ability to save memory as well as to enlarge without distortion. Meanwhile, it has been commonly adopted in data visualization (image data) because it can be scaled to multiple sizes to fit different scenes. For instance, it can be applied for the illustrations in newspapers and magazines, the logo on the web, the background for poster, the design of text, and traffic signs. However, transforming a bitmap to vector image is still a challenging problem because of the complicated content of a bitmap, which tends to consist of more than just simple geometry. Aiming at this issue, there is a new approach proposed to transform from bitmaps to vector images, which is based on triangle units and consists of three steps. In detail, firstly, there is an initial mesh constructed for one image in pixel level after detecting features. Then, the initial mesh will be simplified by collapsing two vertices as the initial mesh is too dense to represent one image. Specifically, there are two main parts in this step, which are collapse conditions and collapse influences. In the step of collapsing, issues such as overlap and sharp triangles can be conquered by a sort-edge method (which will be illustrated specifically later). The final step is to select one color for each triangle, since it is helpful to save the memory and speed up the process of this method. In addition, one color will represent one triangle; hence, in the final step, the four-triangle sample method will be applied in order to prevent a vector image from generating too large color discontinuity. Once the pretest proceeds without mistake, the method above is able to work for the general bitmaps. Note that our method can be applied to information security and privacy, since one image can be encoded to some triangles and colors.

1. Introduction

Visualization of image data tends to be spread into many aspects in today's industries such as the illustrations in magazines, the logo on the web, and the images of posters. To take a poster as an example (especially a big size of poster), usually, there might be a large number of scales needed for various scenes if there are more than one image in a poster. Then generally there are two options for poster production. One is to prepare bitmaps of multiple scales to meet diverse needs, while it costs large storage space and larger bitmaps might be distorted. Therefore, in order to prevent visual distortion in enlarging, the second method of applying vector images to a poster tends to be preferred instead. In the field of information security and privacy, image vectorization can be used to represent one image by a lot of triangles, which can encrypt the image. In the edge computing mode, some images may need to be processed into a vector, which can ensure the reliability of data.

To transform bitmaps into vector images, some methods have been proposed [1–3]. Xia et al. [1] introduced a vectorization algorithm, in which they constructed the initial mesh in the subpixel level and used TPS [4] to do the color fitting. In this way, the representing feature in subpixel can pinpoint the location of features; meanwhile, Thin-Plate Splines (TPS) seem to perform well on the surface smooth. However, critically it requires large memory and computation and a huge number of parameters for one triangle.

Instead of using curved triangles like [1], Liao et al. [2] provided an iterated method for image triangulation. They subdivided each triangle into four small triangles to make the surface smooth. With the increasing number of iterations, the surface becomes smoother. However, it is time-consuming and might generate a large number of small triangles. Therefore, both the TPS and subdivided method will result in vector images with large memory.

In order to avoid potential problems in previous experience, there is a new method about transforming bitmap to vector image proposed in this paper. The main difference of this current method is in the aspects of pixel-level image triangulation and color selection (i.e., allocating one color for one triangle). The new method can be divided into the three following steps: initial mesh construction, mesh simplification, and color selection. Different from [1] whose initial mesh is constructed in the subpixel level, the current method's initial mesh is created in pixel level. In this way, less cost of memory is required and the process is able to be run faster.

Before constructing the mesh, feature detection is necessary, since color discontinuity is significant in these features' area, which should be treated specifically. The initial mesh is very dense after creation; thus, a simplification of initial mesh tends to be essential. In particular, collapse is a way of mesh simplification, which can be defined as a merge of one vertex and another, followed by some influence on each of the vertices' neighborhood's changing (Figure 1). In this step, two aspects need to be considered, collapse condition and collapse influence. The former refers to the requirement to perform different operations on the collapse in different situations. The latter refers to the effect on the structure of the vertices after the collapse is completed. At the same time, in order to prevent overlap (Figure 2) and sharp angle in the results of collapse, the sort-edge method is applied. Once the mesh is simplified, one image can be represented by a certain number of triangles, which is called image triangulation. Image triangulation has extensive usages such as image vectorization [1] and image editing [2]. As for the last step of color selection, there are many interpolation functions such as nearest interpolation, linear interpolation, and Thin-Plate Splines [4]. This paper will use one color as an example to fill a triangle. Even though it is challenging to fill a triangle with just one color considering it is a discrete fitting, using one color fitting still takes advantage of reducing memory costs and speeding up the program process. Furthermore, beneficially the four-triangle sample method used for color selection is able to prevent vector from generating overdiscontinuity. After color selection for each triangle, every triangle's coordinates of three vertices and RGB colors will be recorded, which can be saved as an SVG format vector image. Our contributions are summarized as follows:

(i) We propose a method to convert bitmaps into vector figures. This method includes three steps: initial mesh construction, mesh simplification, and color selection.

(ii) In initial mesh construction, we first constructed a square mesh at the pixel level instead of the subpixel level to improve efficiency. Then we used feature detection and its results to divide the square mesh into the triangular mesh. The use of feature detection allows the segmentation process to consider the significance of color discontinuity.

(iii) In initial mesh construction, we mainly simplify the generated triangular mesh through the collapse operation. In this process, various situations are regulated by constructing collapse conditions, and the occurrence of overlap and the sharp triangle is avoided through the sort-edge method.

(iv) In color selection, to improve efficiency, one color is used for one triangle, and a quantitative method is used to measure the quality of the color selection. At the same time, four-triangle sampling is used to reduce color discontinuity.

The rest of this paper is organized as the following sections. Section 2 will elaborate related work about image vectorization. In Section 3, the method of image vectorization will be introduced, which includes three steps of initial mesh construction, mesh simplification, and color selection. Section 4 is going to illustrate the results of vector images, the discussion of the current method, and future work.

2. Related Work

The vector image is represented by geometric base primitives. It has wide applications such as [5] and may apply to some other domains like image captioning [6–8]. Recently, some new methods for image vectorization are proposed [9–12]. In general, there are three geometric base primitives for transforming a bitmap into a vector image, which are diffusion curve, gradient mesh, and triangles. The diffusion curve [13, 14] is the base unit for a smooth vector image creation. For example, Xie et al. [3] and their team have suggested a method that can automatically generate sparse diffusion curve vectorizations of raster images by fitting curves in the Laplacian domain. Besides, there are also some studies about the diffusion curve [15, 16]. Image vectorization based on gradient mesh [17, 18] is also available. To prove it, there are some studies that discovered color operations by gradient mesh [19–22].

Triangle is used as based primitives for image vectorization in this paper. Image triangulation has wide applications such as image vectorization [1], image editing [2], stylized image [23], image compression [24, 25], and some other cases like [26–28]. Image triangulation usually has two steps, initial mesh construction and mesh simplification. As has been stated previously, the initial mesh construction of Xia et al. [1] is overcomplicated and memory-consuming, since they use subpixel features to construct the initial mesh. As for mesh simplification, a method of [29, 30] using quadric error as a metric can determine the order of simplification. In the current method, there tends to be no particular order for collapse; meanwhile, the collapse is only limited by collapse conditions. For each iteration, the passive vertices of one collapse are randomly chosen. Besides, Liao

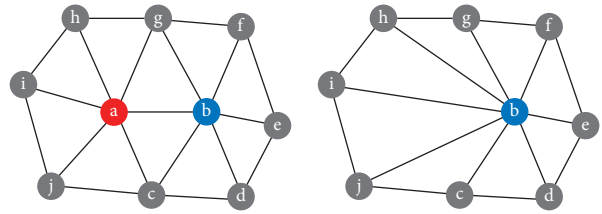


Figure 1: (a) Before collapse. (b) After collapse. We can see that there are 5 types of vertices $\{a, b, C_{ab}, O_a, O_b\}$.

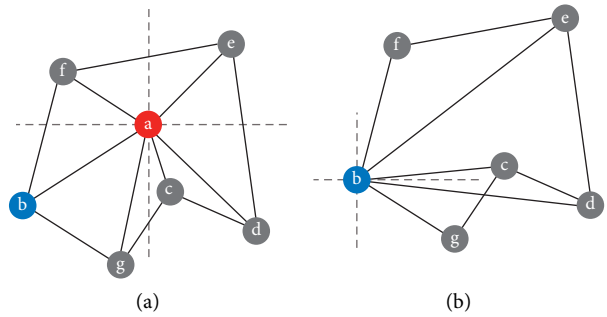


Figure 2: (a) Before collapse. (b) Overlap after collapse. We can see that before collapse the sequence of neighbors of a except b is $\{g, c, d, e, f\}$ but after collapse the sequence becomes $\{g, d, c, e, f\}$ which means overlap happens.

et al. [2] and their team provide a different way for image triangulation. They add an additional step that subdivides the simplified mesh to smooth the surface in a further way. Specifically, one triangle can be subdivided into four small triangles for a smoother surface based on certain algorithms for triangulation such as classical triangulation algorithms like Delaunay [31]. Recently, TriWild [32] is suggested as a preferred one, which is a robust 2D meshing algorithm and generates curved triangles reproducing smooth feature curves.

Color selection is a suitable method. In the case of [1], it uses TPS [4] to fit color. Thin-Plate Splines (TPS) are a spline-based technique for data interpolation and smoothing. Recently, Chen et al. [33, 34] discovered image vectorization by adopting TPS. Applying TPS to color fitting requires saving a huge number of parameters for one triangle, though it can achieve a smooth surface. In [2], the color can be smooth and natural by subdividing one triangle into four small triangles. This paper's method may generate too large color discontinuity, since one color for one triangle is a discrete fitting. Fortunately, conducting a four-triangle sample method is able to address this problem.

In conclusion, "one color for one triangle" can maintain its advantages of saving storage and easy calculation; on the other hand, it can avoid disadvantages of generating large color discontinuity by using the four-triangle sample method.

3. Image Vectorization

In this section, the process of constructing the initial triangular mesh and simplifying the mesh and color fitting (or color selection) will be illustrated specifically. The first two steps are renamed as image triangulation. The third step achieves image vectorization by color selection. The total steps of image vectorization are summarized as follows:

(1) The first step is the initial mesh construction. It is about how to construct the mesh based on the original pixels after feature detection. Feature pixels are treated specifically, since color discontinuity is much larger compared with general pixels.

(2) The second step is mesh simplification. It is necessary to simplify the mesh to get a smaller number of triangles because the initial mesh tends to be too dense to represent one image. In particular, collapse is used to simplify the mesh, which contains collapse conditions and collapse influences. Problems like overlap and sharp triangles are able to be addressed by the sort-edge method.

(3) The final step is color selection, which is about how to determine the color for the triangle. The error function is defined in this step to get the best color for each triangle. It is recommended to adopt a four-triangle sample method to prevent vector image from generating color overdiscontinuity.

3.1. Initial Mesh Construction. The bitmap is composed of pixels with colors (gray or RGB). Image vectorization begins with constructing the initial base units from bitmap because image vectorization needs to use geometric units to represent one image. Triangles are used as base units in this method; thus, this beginning step is about constructing

initial triangular mesh on a bitmap. It can be subdivided into three steps, which are square mesh construction, feature detection, and transferring square mesh to triangular mesh.

Firstly, for each pixel in an image, the nearest four pixels (up, down, left, and right) are linked, which will result in a square mesh. In this step, for general pixels, they have four neighbor pixels. For boundary pixels, they have only three neighbor pixels. In addition, for four vertexes of one image, they have only two neighbor pixels.

Secondly, neighbor feature pixels belonging to the same feature should have one edge to link them together. Canny [35] is used to detect features, which consists of feature pixels. Detected features are thinned to 1-pixel wide and linked by small gaps. Features shorter than 10 pixels should be discarded. Different features are marked for different flags. Once the feature pixels in the image are identified, they will be divided into different types. Then, the square mesh can be transferred to triangular mesh.

Finally, the square mesh is transferred to a triangular mesh. For each square mesh, there are four pixels. If the diagonal of square mesh has two feature pixels, the two feature pixels must be linked by one edge as neighbor pixels. Otherwise, either side of diagonal can be chosen as an edge (Figure 3), which is not significantly influenced by the features. All square meshes should be operated referring to the above statements for generating triangular mesh in the final. Initial mesh construction tends to be completed here.

3.2. Simplify Mesh. A large number of triangles are generated by initial mesh construction. Mesh simplification can reduce the number of triangles by collapse. As noted in this paper, the two vertices related to one collapse should be distinguished. As is shown in Figure 1, vertices a collapse into vertices b. a is called active vertices in this collapse; and b is called passive vertices (i.e., the position of passive vertices in one collapse is fixed). In the process of collapse, specifically, there are two points that need to be concerned. One is the condition of collapse, and the other is the influence of collapse. Garland and Heckbert [29] and their team simplify mesh by quadric error metrics. Differently, collapse conditions are used to control collapse in the current method. The collapse condition is composed of some rules to limit the collapse. For example, collapse conditions should prevent collapse from generating overlap and sharp triangles. Moreover, collapse conditions can only be conducted when one vertex collapses into another, which needs the user's personal judgement. Collapse influence is about the changes in the topology of the triangular mesh when collapse happens. There are five kinds of vertices type. The influence of each type of vertices will be explained in detail later.

3.2.1. Collapse Conditions. The whole image has four vertices that should be fixed in the process of collapse. In the same way, one image has four boundaries. The vertices in the same boundary can collapse to each other. In one collapse, if one vertex is in boundary and another is not, then it can be passive vertex but cannot be active vertex, since boundary

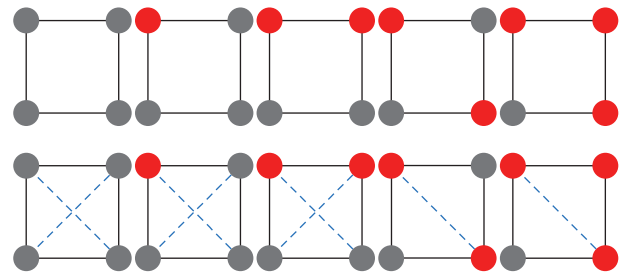


Figure 3: Initial mesh construction. Red point represents feature pixel, and gray point represents general pixel.

vertices collapsing into nonboundary vertices will destroy the shape of the image. The rest of the vertex is divided into feature vertices and general vertices (nonfeature vertices). Feature vertices are detected by Canny. Two feature vertices with the same feature can collapse to each other (in the condition that is not concerned about the shape of the feature, which will be talked about below). However, if collapse happens between feature vertices and general vertices, feature vertices cannot be active vertices. As for general vertices, they can be either active vertices or passive vertices.

For one feature (here feature refers to the set of feature vertices that belong to the same feature), the end of it should be fixed; otherwise, this feature may disappear. Suppose one feature $f = \{v_1, v_2, \ldots, v_n\}$, where $v_i, i \in n$ denotes the vertices in order which belong to feature f, which has n vertices. If v_1 and v_n are not fixed, then v_1 can collapse to v_2 when meeting the collapse conditions. Additionally, it may appear that v_{n-1} collapses to v_n and, finally, only v_n remains. It may show that the feature $f = \{v_n\}$ if the end vertices of f are not fixed. There is another constriction of collapse between the same feature vertices, which is the shape of one feature. One collapse should not be allowed if it influences the shape of the feature a lot. As shown in Figure 4, a collapses to b (same as a is active vertices and b is passive vertices in this collapse), and angle(cab) is used to measure the extent of change to the shape of the feature, which can be computed as

$$\theta = \arccos\left(\frac{L(a,c)^2 + L(a,b)^2 - L(b,c)^2}{2L(a,c)L(a,b)}\right), \quad (1)$$

where $L(\cdot)$ denotes the Euclidean distance of two vertices and θ represents angle(cab).

Collapse does not allow two-edge cross which is called overlap; hence, the collapse will be rolled back once overlap happens. The way of judging whether there is overlap or not is based on the following criteria (sort-edge method). As shown in Figure 4, the authors suppose that vertex a collapses to vertex b. Before the collapse, a coordinated system is created with a as the origin. The anticlockwise order of the neighbors of a (except passive vertices) is $\{g, c, d, e, f\}$. Note that only the order is concerned (i.e., the first one of this order is arbitrary); $\{g, c, d, e, f\}$ is equivalent to $\{f, g, c, d, e\}$. After the collapse, the authors use vertex b as the origin to create a coordinate system and to judge if the anticlockwise

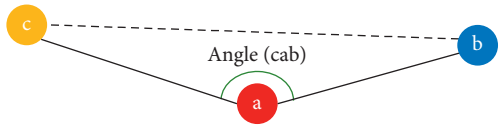


Figure 4: Keep feature shape. If vertices a want to collapse into vertices b, then angle(cab) should be larger than threshold.

order of the neighbors of b is the same as that of a. If the order before the collapse is different from that after the collapse, then roll back this collapse; otherwise allow this collapse.

The sharp triangle is defined as the triangle that has a very small angle. A stricter limit for the collapse is to avoid sharp triangles due to their many disadvantages. For example, compared with the equilateral triangle, the distribution of sampling points for a sharp triangle is more uneven in the process of color selection, which may cause a large error. Besides, the sharp triangle is not friendly to subsequent collapse. Fortunately, the sort-edge method can be adopted to avoid sharp triangles. In order to avoid sharp triangles, it is allowed to directly calculate the angle of two adjacent edges and then roll back the collapse that generates a small angle because the sequence of edges has already been known.

3.2.2. Collapse Influences. A collapse will be accepted if it meets collapse conditions. The vertices in one collapse can be identified as five types. It is supposed that vertex a collapses to vertex b (i.e., a is the active vertex and b is the passive vertex). Then five types of vertices can be defined as $\{a, b, C_{ab}, O_a, O_b\}$, where C_{ab} denotes the vertices that are neighbors of both a and b. O_a denotes the vertices that only link with a and O_b denotes the vertices that only link with b. Take Figure 3 for an example; $C_{ab} = \{c, g\}$, $O_a = \{h, i, j\}$, $O_b = \{d, e, f\}$. In actual implementation, the authors create a struct for vertices that have a neighbor list, also called the edge list. Specifically, the changes of five types of vertices were actually accompanied by the changes of neighbor list, which are explained as follows.

For vertex a, it is the active vertices in this collapse, which just requires clear identification of all neighbors in the neighbor list, since vertices a will disappear after the collapse. For passive vertices b, they delete the neighbor a and add all vertices in O_a as new neighbors. When a disappears after the collapse, all vertices in O_a should delete a in their neighbor list and include b, since, with a collapsing to b, the vertices linked to a will now link to b. As for vertices in C_{ab}, they delete a in their neighbor list because vertices a collapsing to vertices b can be described as that a and b are now the same. Vertex in O_b is kept unchanged, which means the collapse has no influence on O_b. As shown in Figure 1, the collapse can change the triangular mesh structure while making the number of triangles less, which is the aim of mesh simplification.

3.2.3. Algorithm Design. In code implementation, a structure named *Vertices* for vertices is designed. In the structure *Vertices*, there are some members such as *position* (the position of this vertex), *Neighbors* (the neighbor vertices of this vertex, it is a list), and *color* (the color of these vertices). Therefore, the list of *Vertices* can be used to represent the triangular mesh. Actually, the step of mesh simplification is also operating *Neighbors* in *Vertices*.

Algorithm 1 shows the symbolic description of the algorithm based on the assumption that vertices a collapse to vertices b. The structure *Vertices* of a is as v_a, and similarly, v_b denotes the structure *Vertices* of b. N_a, N_b denote the *Neighbors* of a, b, and n_a, n_b denote the items of N_a, N_b, which are also the structure. The function **remove (x, y)** means deleting its neighbor **y** by removing **y** from its *Neighbors* (remove **y** in **x**'s neighbor list) in the structure *Vertices* of **x**. Function **create (x, y)** means adding **y** into **x**'s neighbor list and function **clear (x)** means clearing all neighbors of vertices **x**.

As is discussed in collapse influences, the vertices in one collapse can be divided into five types. For each type, the impact tends to be different. Specifically, Algorithm 1 starts from the loop through the neighbor vertices of active vertices in one collapse. As for each of the neighbor vertices, their neighbor list tends to be updated according to their own type. The time complexity of Algorithm 1 is $O(1)$ because of the constant number of neighbors of one vertex.

Given that the triangular mesh can be represented by the list of *Vertices*. The steps of algorithm for mesh simplification are shown in Algorithm 2. Function **random (x)** means to randomly choose one neighbor of vertices **x**; meanwhile, function **collapse_condition (x, y)** returns true if this collapse is allowed, and function **collapse (x, y)** in this algorithm refers to the functionality of Algorithm 1.

As is shown in Algorithm 2, it loops through the entire vertices list. As for each vertex, one of its neighbor vertices is randomly chosen. Then this vertex is collapsed to its neighbor vertices. Algorithm 2 will be iterated T times. In most of authors' experiments, T is set to 40. Therefore, the time complexity of Algorithm 2 is $O(M)$, where M denotes the total number of vertices in one image.

3.3. Color Selection. After mesh simplification, image triangulation is completed, which means the bitmap is represented by many triangles. In order to distinguish image vectorization, each triangle should be colored. For triangle coloring, Xia et al. [1] introduced TPS [4] to do color fitting, but TPS [4] has high computation and storage consumption. A different way of color fitting is implemented in this current study. Specifically, only one color will be used to represent one triangle, which needs more accurate color selection as it is a discrete fitting.

Each triangle should be subdivided into n^2 equal parts, which means dividing each side length of the triangle into n equal parts (in this paper n is set to 3). Then the center of each subtriangle is taken as a sample (Figure 5). Since only one color is needed, an error function is defined for each sample like the following formula:

Input: the structure *Vertices* of active vertices a, the structure *Vertices* of passive vertices b.
Output: the structure *Vertices* of $\{a, b, C_{ab}, O_a, O_b\}$

```
(1) for n_a in N_a do
(2)   if n_a == v_b then
(3)     continue
(4)   isCommon == false//judge if n_a is in C_ab
(5)   for n_b in N_b do
(6)     if n_a == n_b then
(7)     isCommon = true
(8)     break
(9)   for n_{n_a} in N_{n_a} do
(10)    if n_{n_a} == v_a then
(11)      remove (n_a, a)//operation for both O_a and C_ab
(12)      if isCommon == false then
(13)        create (v_b, n_a)//operation for b
(14)        create (n_a, v_b)//operation for O_a
(15) clear (v_a)//operation for a
```

Algorithm 1: Collapse algorithm.

Input: the list of Vertices
Output: the list of Vertices (simplified)

```
(1) for v in Vertices do
(2)   n = random (v)
(3)   if collapse_condition (v, n) == true then
(4)     collapse (v, n)
```

Algorithm 2: Mesh simplification algorithm.

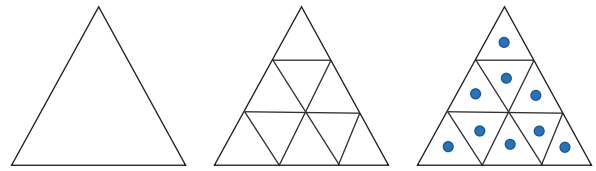


Figure 5: (a) The original triangle. (b) Subtriangles. (c) Samples of subtriangles.

$$\text{error}_i = \sum_{j}^{N} \left\| c_i - c_j \right\|, \tag{2}$$

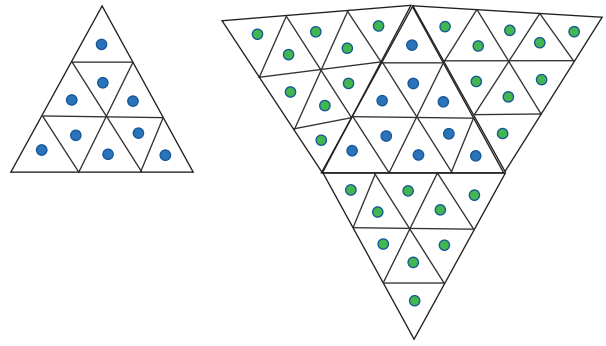
where c_i, c_j denote the colors of the ith and jth samples. N is the number of samples. error_i denotes the error of ith sample.

The best color of the triangle is the ith sample with minimum error_i for $i = 1, 2, \ldots, N$.

$$c_{\text{best}} = \text{color}\left(\arg\min_{i \in N} \text{error}_i \right), \tag{3}$$

where $\text{color}(\cdot)$ denotes the function that gets color by using sampling point as input parameter.

However, sampling from only one triangle may cause large color continuity; hence, each triangle is too independent. Therefore, to address this problem, this paper provides an approach of four-triangle sample (Figure 6) method. Conducting this method can start from defining the triangle that is doing color selection as a reference triangle. For each reference triangle, there are three near triangles and each of them shares one edge with a reference triangle. Sampling points from these four triangles get the best color for the reference triangle using formulas (2) and (3). Nevertheless, there is a consideration that needs to be concerned which is if one side of the reference triangle is feature edge, the triangle related to this feature edge should not be sampled. Two triangles sharing one feature edge must have a



(a) (b)

Figure 6: (a) One-triangle sample. (b) Four-triangle sample.

very different color because color discontinuity is significant in the feature area.

4. Results and Discussion

The proposed method about image vectorization consists of three steps, which are initial mesh construction, mesh simplification, and color selection (Figure 7). Some results about our method are shown in Figure 8. All of these steps are described, respectively, as follows:

Step 1: initial mesh construction

(i) *Input Images*. The input images are bitmaps with formats of *png*, *jpg*, etc. The size of input images is arbitrary.

(ii) *Canny Detection*. Using Canny to detect features in the input image, the thresholds of Canny are set to 0.05 and 0.125, respectively. The features shorter than 10 pixels are removed.

(iii) *Square Mesh Construction*. For each pixel in the input image, connect it to its neighbor s with four pixels (up, down, left, and right), which will result in a dense square mesh.

(iv) *Triangular Mesh Construction*. Every square mesh consists of 4 pixels. If both the two pixels on its diagonal are feature pixels (detected by Canny), just connect this diagonal and it will make this square change to 2 triangles. Or either diagonal can be connected to generate 2 triangles. Thus, a dense triangular mesh is got.

Step 2: mesh simplification

(i) *Collapse Vertices*. Each vertex (same as the pixel, also named vertices in this section) randomly chooses one neighbor vertex of it (two vertices with an edge between them are called neighbor vertices) and then collapses itself into its neighbor vertices if this collapse meets collapse conditions. The iterations for all vertices are set to 40. After that, the simplified triangular mesh can be created.

(ii) *Extract Triangles*. BFS algorithm is used to extract all triangles on the triangular mesh. Each triangle is saved as 3 vertices.

Step 3: color selection

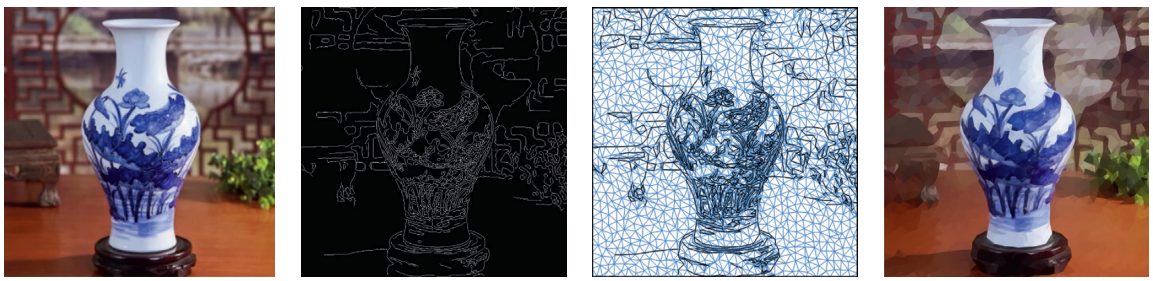
(i) *Point Sampling*. The side of every extracted triangle needs to be equally subdivided into n parts. As shown in Figure 5, linking the points on the side will generate n^2 subtriangles. The center of every subtriangle will be extracted as a sampling point. Thus, n^2 sampling points can be gained for each triangle. In this implementation, n is set to 3.

(ii) *Select Colors for Triangles*. This method uses formulas (2) and (3) to get the best color from the sampling points for each triangle.

(iii) *Saved as SVG Format*. Once the three vertices of each triangle and their corresponding color are finally settled, all the information can be saved as SVG format, which is also the format of the vector image.

4.1. Details and Memory Trade-Off Experiment. This experiment is about adjusting parameters to do details and memory trade-off. The current method about image vectorization starts from Canny detection whose threshold is set to 0.05 and 0.125, respectively. Features shorter than 10 pixels are dropped, since they are considered unimportant. In the step of mesh simplification, for each iteration, all pixels in one image will be enumerated to collapse and the total number of iterations is set to 40. In color selection, each triangle will be subdivided into 9 small triangles for sampling.

The compared parameters here are cos(angle), feature length, and side length. The parameter cos(angle) controls the smallest angle of each triangle; feature length limits the length of feature; side length prevents the side length of each triangle from over length. To test the influences of these parameters, the authors set cos(angle) in $\{-0.8, -0.5, -0.2\}$, feature length in $\{10, 30, 50\}$, and side length = 100. The result is shown in Figure 9. In addition, the authors set side length in $\{10, 30, 50, 100\}$, cos(angle)) = −0.5, and feature length = 30. The result is shown in Figure 10.



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

Figure 7: The pipeline of image vectorization. From left to right are input image, Canny detection, simplified triangular mesh, and vector image.

Note that side length is different from feature length, though both of them are related to the side length of the triangle. The feature length just controls the edge whose two vertices are both features, while side length controls the rest of the edge length. The reason for choosing two parameters to control is that it makes the method more flexible. In this way, more details of the foreground of the image can be preserved; meanwhile, more details of the image background can be discarded.

As shown in Figure 9, the petal of the flower of the top-right image is more simplified than the others. This is because the increase of cos(angle(cab)) and feature length will make the feature vertices collapse more freely. Specifically, the increase of cos(angle(cab)) makes the restriction on the sharp triangle in the collapse conditions relax so that the collapse that has a greater impact on the shape of the feature can also occur; and a larger *feature length* parameter value leads to the fact that, after more collapses occur, the *feature length* can still be within the limit. These two changes will make the final number of triangles less, so more details of the picture will be lost. But more triangles and better details will take up more memory. As shown in Table 1, with the decrease of cos(angle(cab)) and *feature length*, the number of triangles increased. Although the details of the picture are better, the memory increases. Therefore, when cos(angle(cab)) is −0.8 and *feature length* is 10, the feature vertices can hardly collapse and may generate many tiny triangles to be close to the curve. So, as can be seen in Figure 9, the details of the image under this setting are better, but its memory is the largest in Table 1.

In Figure 10, when the side length is 10, the simplified triangular mesh is very dense, and the details of the output image are well preserved. However, when the side length becomes 100, a smaller number of triangles can represent this image; meanwhile, the details are not preserved well. Thus, if the *side length* becomes bigger, the number of triangles will decline. Therefore, this parameter can determine the density of triangles. At the same time, it can be seen in Table 2 that as the *side length* increases, the memory also increases. It also shows that better preservation of picture details comes at the cost of memory increase.

The three parameters actually can impact the details of one output vector image by controlling the triangles' number and shape (angle, side length). Thus, details and memory trade-off can be done for different needs.

4.2. K-Medoids for Color Simplification Experiment. This experiment is about the use of K-medoids for reducing the number of colors and about the difference in changing the value K in K-medoids. Usually, after getting the vector image, many colors may not be important to the contribution of image features. If the number of useless colors in one image can be reduced with the most details of the image maintained, the storage cost of this image will be reduced. Therefore, an algorithm for simplifying the number of colors is needed. Fortunately, in data science, K-medoids is an unsupervised method for clustering data and it is robust to noise. It can be exactly used to cluster the colors in images to achieve the purpose of reducing the number of useless colors.

For each triangle, one color is saved as an RGB format. Therefore, each color in RGB format can be regarded as a 3D space point, which can then be clustered by K-medoids. For each cluster, there is one central point making the equal distance between the center points and keeping all member points in this cluster smallest. After the termination of the K-medoids algorithm, all member points in one cluster will be replaced by center points, which means all colors in this cluster are replaced by center color. K-medoids is sensitive to the initial parameters so that sometimes initial colors can be selected manually for further preciseness.

It is set that cos(angle) = −0.5, feature length = 30, side length = 30, and K is in {15, 10, 5}, where K denotes the number of clusters in K-medoids (i.e., the number of colors for the vector image). There are different values of K in {15, 10, 5} tested whose outputs are illustrated in Figure 11.

As shown in Figure 11, with the number of colors descending, the apple just keeps the most important colors (i.e., red and green). This is because K-medoids can find a good color to represent a group of colors. The last apple shows that only 5 colors can represent this apple. It proves that K-medoids can efficiently cluster image colors and keep the most important color for apples. Furthermore, it is obvious that when the number of colors descends, the vector image will become less detailed. For example, compared with the first apple in Figure 11, the last apple loses minor texture significantly. If one image with many similar colors is unable to be distinguished, the adoption of K-medoids can decrease the number of colors and also reduce the storage of this image.

(a) (b)

Figure 8: (a) Input image. (b) Vector image.

4.3. The Advantages of Four-Triangle Sample Experiment. There is an approach called the four-triangle sample method in the process of color selection, which will be described below. One triangle is generally marked as a reference triangle; the remaining three triangles sharing an edge with the reference triangle are also taken into account. Finally, the color fitting will be arranged for sampling points from these four triangles. In this experiment, a comparison of the difference between using and not using this method will be conducted.

It is set that cos(angle) = −0.5, feature length = 30, and side length = 30. The compared parameter here is the number of sampling triangles, which tends to be one and four, respectively.

In Figure 12, the magnification part shows that the four-triangle sample (Figure 12(c)) is more robust than the one-triangle sample (Figure 12(b)). The original image (Figure 12(a)) has many noisy colors such as the ground color in the original image. The one-triangle sample may

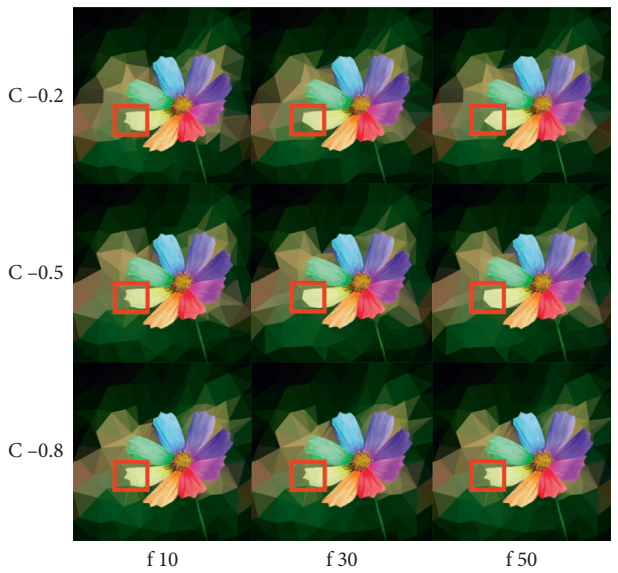


Figure 9: The influence of cos (angle (cab)) and feature length. We can see that the increase of cos (angle (cab)) and feature length will lead to less limitation for feature collapse.

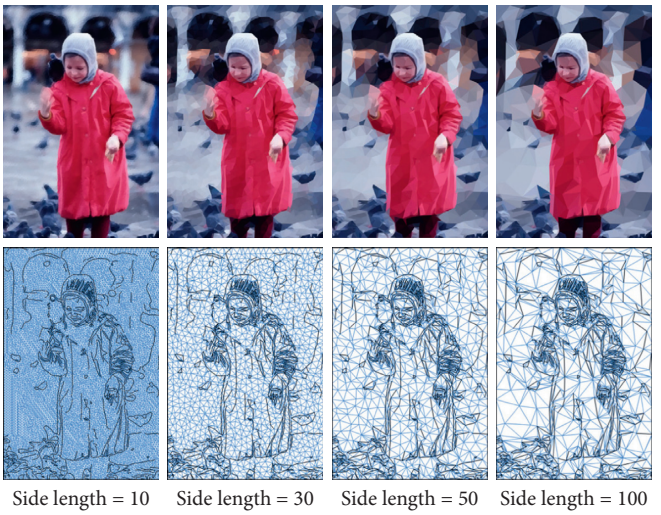


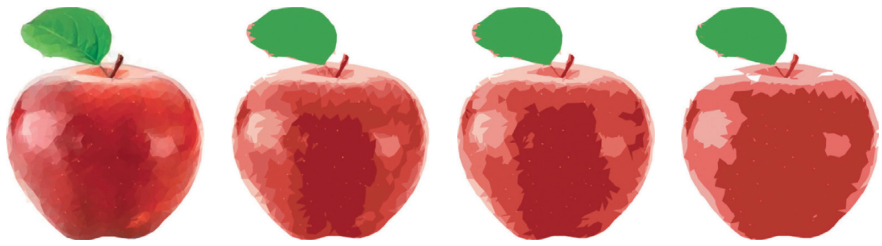
Figure 10: Vector image with different side length.

Table 1: The size of experimental results images of the source image with a size of 32 kB under different parameters.

Image size (kB)	$f = 10$	$f = 30$	$f = 50$
$c = -0.2$	18.1	15.3	12.2
$c = -0.5$	19.8	17.6	14.7
$c = -0.8$	23.7	20.3	17.4

Table 2: The size of experimental results images of the source image with a size of 758 kB under different parameters.

Side length	10	30	50	100
Image size (kB)	636.2	531.6	425.4	239.7



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

Figure 11: Simplified color. From left to right, the number of colors is getting less. (a) Stylized image. (b) Stylized image number of color 15. (c) Stylized image number of color 10. (d) Stylized image number of color 5.

(a) (b) (c)

Figure 12: Comparison of one-triangle sample with four-triangle sample. (a) Original image. (b) One-triangle color selection. (c) Four-triangle color selection.

increase the proportion of these noisy colors, which may cause many small discontinuity triangles. Differently, the four-triangle sample takes the neighbor triangles into account, which is able to reduce the proportion of noisy colors. To expand this method, a larger number of triangles can be sampled for a better result. However, computing costs should be concerned in that case. Therefore, users should consider the balance of cost and result when using the four-triangle sample method. Any reference triangle of the four-triangle sample method is unlikely to be influenced by feature triangles which are sharing the same feature edge with reference triangle but on different sides.

5. Conclusions and Future Work

Due to its simplicity and effectiveness, this paper's method has the ability to do image triangulation quickly. In addition, some parameters are provided to adjust the density of triangles or the extent of the collapse. Problems of collapse such as overlap and sharp triangles are fixed by the sort-edge method. Using one color to represent one triangle makes less computation and storage, while it may cause a challengeable task since one color may lead to large error. Thanks to the four-triangle sample method, this problem can be conquered and color selection tends to be more robust. Our method outputs each triangle's three vertices and RGB color, which then can be saved as an SVG format image to get a vector image.

There are still some limitations to this method and recommendations for future work. For example, in textureless areas like the cloud in Figure 12, the performance of this method still has the potential to improve. To be more specific, the number of features is not large enough, and so they are not representative enough in textureless areas. In order to acquire more features, adjusting parameters of Canny is suggested. However, increasing features may cause additional unexpected limitations, which might result in dense triangles. Conclusively, the result of feature detection is a big consideration for the authors. Therefore, it tends to be recommended to explore a method for facilitating feature detection in the future. For the consideration of time and space efficiency, the conversion method proposed in this paper is relatively simple, but the conversion result of some specific types of pictures is completely acceptable. In the future, we plan to develop a quantitative measurement method to measure the quality of the conversion result or even directly measure the original image to determine whether this method is suitable for it.

Data Availability

The software code and samples used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This paper is supported by Medical Health Science and Technology Project of Zhejiang Provincial Health Commission (2018KY645) and Zhejiang Science and Technology Plan Project of China (2020C03091).

References

[1] T. Xia, B. Liao, and Y. Yu, "Patch-based image vectorization with automatic curvilinear feature alignment," in *Proceedings of the ACM SIGGRAPH Asia 2009 papers on—SIGGRAPH Asia*, Yokohama, Japan, December 2009.

[2] Z. Liao, H. Hoppe, D. Forsyth, and Y. Yu, "A subdivision-based representation for vector image editing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 11, pp. 1858–1867, 2012.

[3] G. Xie, X. Sun, X. Tong, and D. Nowrouzezahrai, "Hierarchical diffusion curves for accurate automatic image vectorization," *ACM Transactions on Graphics*, vol. 33, no. 6, pp. 1–11, 2014.

[4] F. L. Bookstein, "Principal warps: Thin-Plate splines and the decomposition of deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 6, pp. 567–585, 1989.

[5] Q. Fu, Y. He, F. Hou, J. Zhang, A. Zeng, and Y.-J. Liu, "Vectorization based color transfer for portrait images," *Computer-Aided Design*, vol. 115, pp. 111–121, 2019.

[6] J. Yu, J. Li, Z. Yu, and Q. Huang, "Multimodal transformer with multi-view visual representation for image captioning," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[7] J. Yu, M. Tan, H. Zhang, D. Tao, and Y. Rui, "Hierarchical deep click feature prediction for fine-grained image recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[8] X. Yang, S. Zhou, and M. Cao, "An approach to alleviate the sparsity problem of hybrid collaborative filtering based recommendations: the product-attribute perspective from user reviews," *Mobile Networks and Applications*, vol. 25, no. 2, pp. 376–390, 2019.

[9] S. Lu, W. Jiang, X. Ding et al., "Depth-aware image vectorization and editing," *The Visual Computer*, vol. 35, no. 6–8, pp. 1027–1039, 2019.

[10] J. Shan and W. Jiang, "Research on vectorization method of complex linear image data," in *Proceedings of the International Conference on Advanced Hybrid Information Processing*, pp. 315–321, Springer, Harbin, China, July 2018.

[11] B. Kerautret, P. Ngo, Y. Kenmochi, and A. Vacavant, "Greyscale image vectorization from geometric digital contour representations," in *Proceedings of the International Conference on Discrete Geometry for Computer Imagery*, pp. 319–331, Springer, Vienna, Austria, September 2017.

[12] M. Yang, H. Chao, C. Zhang et al., "Effective clipart image vectorization through direct optimization of bezigons," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 2, pp. 1063–1075, 2015.

[13] A. Orzan, "Diffusion curves: a vector representation for smooth-shaded images," in *Proceedings of the SIGGRAPH'08*, Los Angeles, CA, USA, August 2008.

[14] S. Jeschke, "Generalized diffusion curves: an improved vector representation for smooth-shaded images," *Computer Graphics Forum*, vol. 35, no. 2, pp. 71–79, 2016.

[15] H. Lin, J. Zhang, and C. Xu, "Diffusion curves with diffusion coefficients," *Computational Visual Media*, vol. 4, no. 2, pp. 149–160, 2018.

[16] S. Zhao, F. Durand, and C. Zheng, "Inverse diffusion curves using shape optimization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 7, pp. 2153–2166, 2017.

[17] B. Price and W. Barrett, "Object-based vectorization for interactive image editing," *Visual Computer*, vol. 22, no. 9–11, pp. 661–670, 2006.

[18] J. Sun, L. Liang, F. Wen, and H.-Y. Shum, "Image vectorization using optimized gradient meshes," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 11, 2007.

[19] Y. Xiao, L. Wan, C. S. Leung, Y.-K. Lai, and T.-T. Wong, "Optimization-based gradient mesh colour transfer," *Computer Graphics Forum*, vol. 34, no. 6, pp. 123–134, 2015.

[20] L. Wan, Y. Xiao, N. Dou, C.-S. Leung, and Y.-K. Lai, "Scribble-based gradient mesh recoloring," *Multimedia Tools and Applications*, vol. 77, no. 11, pp. 13753–13771, 2018.

[21] G. J. Hettinga, R. Brals, and J. Kosinka, "Colour interpolants for polygonal gradient meshes," *Computer Aided Geometric Design*, vol. 74, Article ID 101769, 2019.

[22] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[23] K. Lawonn and T. Günther, "Stylized image triangulation," *Computer Graphics Forum*, vol. 38, no. 1, pp. 221–234, 2019.

[24] D. Marwood, P. Massimino, M. Covell et al., "Representing images in 200 bytes: compression via triangulation," in *Proceedings of the 25th IEEE international conference on image processing (ICIP)*, pp. 405–409, IEEE, Athens, Greece, October 2018.

[25] R.-I. Chang and C.-Y. Su, "Color gradient vectorization for SVG compression of comic image," *Journal of Visual Communication and Image Representation*, vol. 33, pp. 235–246, 2015.

[26] S. Roy, P. Shivakumara, U. Pal, T. Lu, and G. H. Kumar, "Delaunay triangulation based text detection from multi-view images of natural scene," *Pattern Recognition Letters*, vol. 129, pp. 92–100, 2020.

[27] X. Zhang, F. Fan, M. Gheisari, and G. Srivastava, "A novel auto-focus method for image processing using laser triangulation," *IEEE Access*, vol. 7, pp. 64837–64843, 2019.

[28] H. Gao, W. Huang, and Y. Duan, "The cloud-edge based dynamic reconfiguration to service workflow for mobile ecommerce environments: a QoS prediction perspective," *ACM Transactions on Internet Technology*, 2020.

[29] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 209–216, ACM Press/Addison-Wesley Publishing Co., Los Angeles, CA, USA, August 1997.

[30] H. Gao, L. Kuang, Y. Yin, B. Guo, and K. Dou, "Mining consuming behaviors with temporal evolution for personalized recommendation in mobile marketing apps," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1233–1248, 2020.

[31] D. T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International Journal of*

Computer & Information Sciences, vol. 9, no. 3, pp. 219–242, 1980.

[32] Y. Hu, T. Schneider, X. Gao et al., "TriWild: robust triangulation with curve constraints," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 52, 2019.

[33] K. W. Chen, Y. S. Luo, Y. C. Lai et al., "Image vectorization with real-time thin-plate spline," *IEEE Transactions on Multimedia*, vol. 22, no. 1, pp. 15–29, 2019.

[34] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4532–4542, 2019.

[35] J. F. Canny, "A computation approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.