Image Triangulation Using the Sobel Operator for Vertex Selection

Olivia X. Laske
Department of Mathematics, Statistics, and Computer Science,
Macalester College, St. Paul, MN, USA

Lori Ziegelmeier
Department of Mathematics, Statistics, and Computer Science,
Macalester College, St. Paul, MN, USA

Abstract

Image triangulation, the practice of decomposing images into triangles, deliberately employs simplification to create an abstracted representation. While triangulating an image is a relatively simple process, difficulties arise when determining which vertices produce recognizable and visually pleasing output images. With the goal of producing art, we discuss an image triangulation algorithm in Python that utilizes Sobel edge detection and point cloud sparsification to determine final vertices for a triangulation, resulting in the creation of artistic triangulated compositions.

2012 ACM Subject Classification Computing methodologies → Image processing

Keywords and phrases Image Triangulation, Sharpening, Sobel Edge Detection, Delaunay Triangulation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2024.91

Category Media Exposition

Supplementary Material *Software*: `https://github.com/OliviaLaske/ImageTriangulation` [1] archived at `swh:1:dir:9ffab16fab1c54439e4641ae8167740d984993b8`

Funding *Lori Ziegelmeier*: Support through NSF:CDS&E-MSS-1854703 and NSF:BCS-2318171.

1 Introduction

Image triangulation creates an abstract representation of an image, which can be defined by four primary principles: the output image (a) divides the original image into a set of non-overlapping triangles, (b) simplifies the original image, (c) approximates original image features as triangles, and (d) retains the integrity of the original image. Several algorithms exist to perform image triangulation, such as [3], [4], [5], and [7]. While this may be done for a variety of purposes, our goal is to use image triangulation as a form of art.

Here, we implement an image triangulation algorithm to create visual works of art; see GitHub repository [2]. To simplify processing, the algorithm converts an image to grayscale. Next, it sharpens the image and applies the Sobel operator to detect edges. After identifying key pixels along the edges, it triangulates them and fills the resulting triangles with color to create the triangulated image. We discuss each step of our approach in the following section.

2 Image Triangulation Algorithm

2.1 Convert to Grayscale

We illustrate our algorithm with an image taken by the first author, Figure 1 (Left). We convert the original image to grayscale using the linear combination for the red, green, and blue (RGB) pixel values: $c = 0.299R + 0.587G + 0.114B$, where c is rounded to the nearest integer [6]. After iterating over each pixel, Figure 1 (Right) displays the grayscale image.

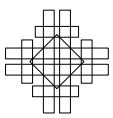


Figure 1 (Left) Original image before processing. (Right) Image after converting to grayscale.

2.2 Sharpen Image

Next, we use a Laplacian sharpening kernel L to more clearly define the edges [6]:

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{1}$$

We use L to perform image convolution, demonstrated in Figure 2. For each interior pixel I_{xy} of the $n \times n$ image I, we calculate the sharpened grayscale value H_{xy} as the weighted sum of I_{xy} and the surrounding pixels (assuming 0-indexing),

$$H_{xy} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} I_{x+i,y+i} L_{i+1,j+1}. \tag{2}$$

Repeating the process for each interior pixel, we create a new sharpened image of size $(n-2) \times (n-2)$ since boundary pixels are not mapped to the new image. Figure 3 (Left) displays the image after using the sharpening procedure.

2.3 Apply Sobel Operator

We then apply the Sobel operator to extract the edges in the image [8]. The process requires running the image convolution process twice using the following x and y kernels,

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad G = \sqrt{G_x^2 + G_y^2}. \tag{3}$$

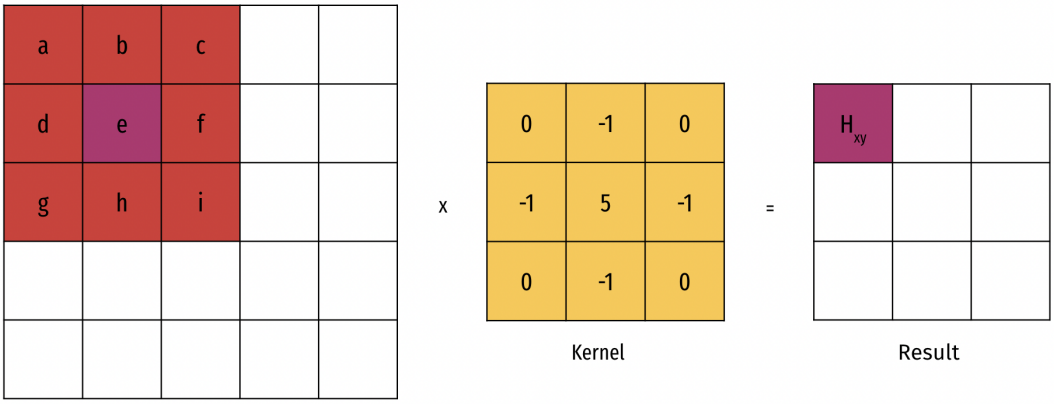


Figure 2 The image convolution process, which produces a new image by taking a weighted sum of each original pixel and its surrounding pixels. For the pink pixel I_{xy}, $H_{xy} = -b - d + 5e - f - h$.

Figure 3 Image after (Left) sharpening with Eq. 1 kernel, (Right) applying the Sobel operator.

The x-kernel whitens pixels to the left and right of pixel I_{xy}, while the y-kernel whitens pixels above and below pixel I_{xy}. Pixels that are part of a well-defined edge in the resulting $(n-4) \times (n-4)$ image[1] have grayscale values closer to white (255) than to black (0); see Figure 3 (Right). We reject pixels below a threshold grayscale value t, selecting $t = 50$.

2.4 Triangulate Points

Let S be the set of pixels above the Sobel threshold; these are used as the vertices to create the triangulation. Even with a threshold value, the pixels are too densely packed to create a visually appealing image. We reduce the density by uniformly sampling $|S|/d$ points, where $|S|$ is the number of pixels in S and d is a density parameter specified by the user. We select $d = 60$. From these sampled points as vertices, we then compute the Delaunay triangulation. This choice can be modified by the user if there is a preference for a particular triangulation behavior, such as anisotropy. Figure 4 (Left) shows this triangulation.

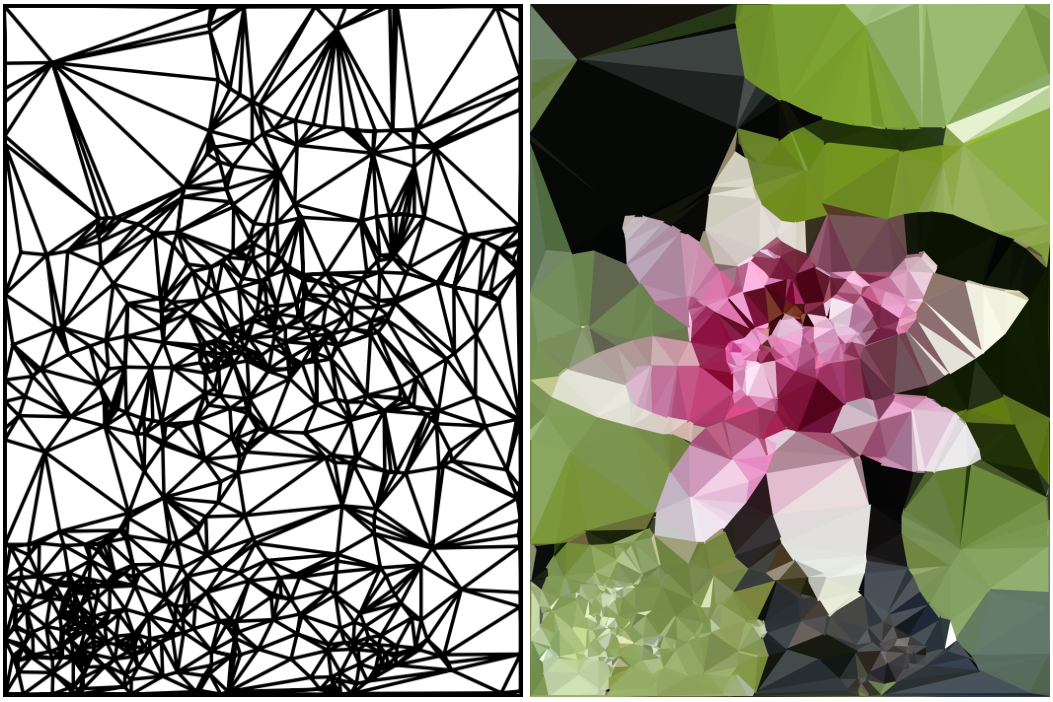


Figure 4 (Left) Triangulation of image. (Right) Final colored image triangulation.

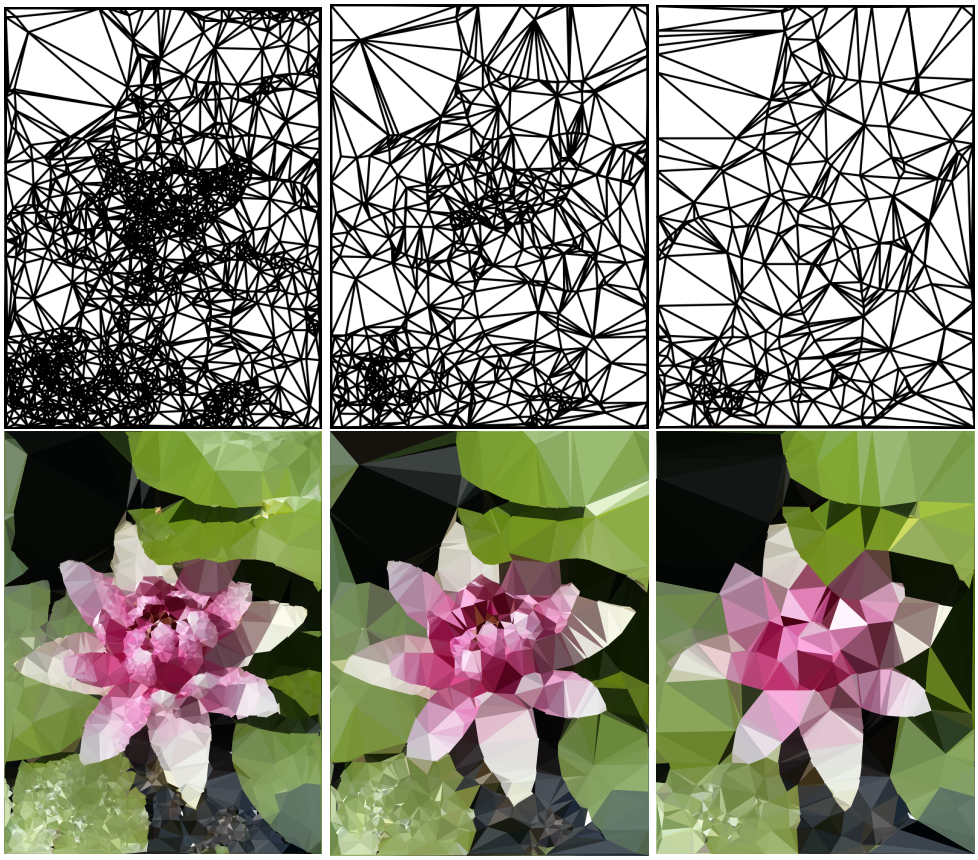
2.5 Color in Triangles

Finally, we calculate the centroid of each triangle with the average of its vertex coordinates, rounded to the nearest integer, to obtain indices (x, y). The final color of the triangle is equal to the RGB value at pixel (x, y) in the original image. Figure 4 (Right) depicts the final image triangulation.

[1] Alternatively, one could adjust the indexing of both the sharpening kernel and Sobel operator to include boundary pixels.

3 Varying Triangulation Parameters

We investigate the effect of varying two parameters selected by the user: the threshold value t to select edges after applying the Sobel operation and the density parameter d to subsample pixels above this threshold. Both are directly related to the number of vertices in

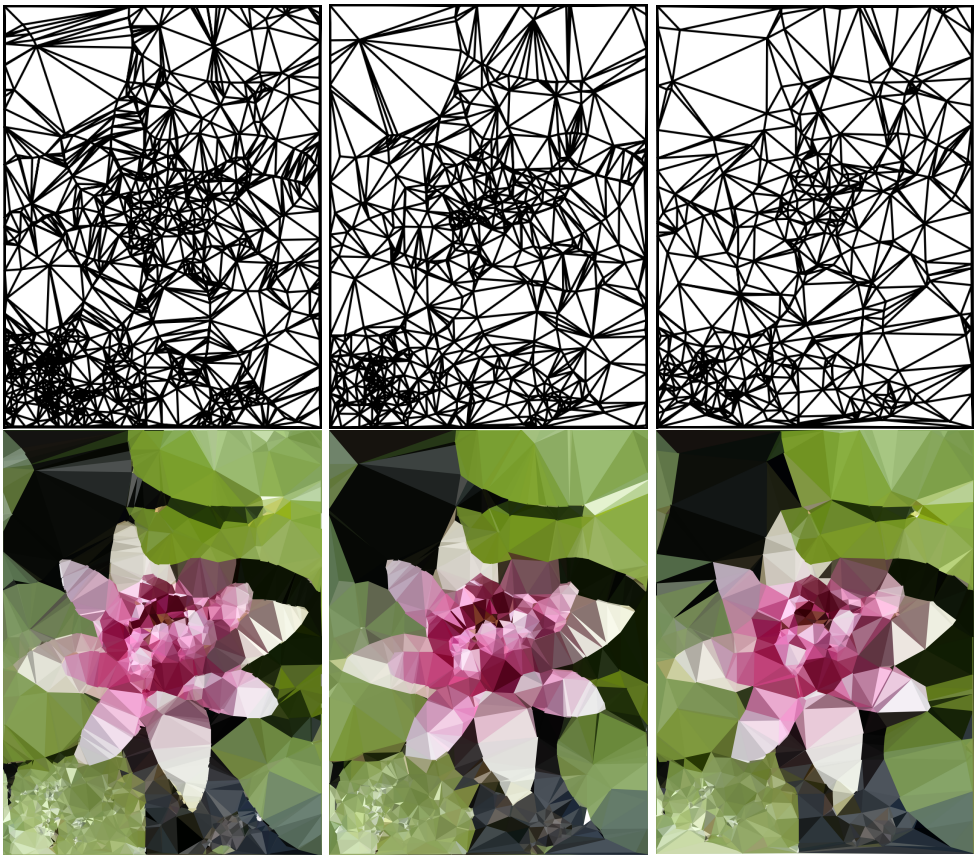


(a) $t = 25$ **(b)** $t = 50$ **(c)** $t = 75$

Figure 5 Differences in final image triangulation depending on threshold to select pixels.

the triangulation. As Figure 5 shows, the number of vertices and triangles decreases as the threshold increases for fixed d. Parameters t and d have similar effects. That is, increasing d with fixed t reduces the number of triangles and vertices, as seen in Figure 6. The image in the leftmost panel in Figure 6, though, demonstrates the extent to which d declutters the points as opposed to t. As a result, the main purpose of density reduction is to decrease run time and avoid clusters of very small triangles.

Finally, we consider random points as triangulation vertices as opposed to following the algorithm discussed here. We see in Figure 7 the image triangulation using 1000 randomly generated points, which is slightly more than the number of vertices that our algorithm finds (with $t = 50$ and $d = 60$). The integrity of the image is destroyed, losing many of the features. By using edge detection, though, we can reduce the number of points needed to make a recognizable image and preserve the underlying skeleton.



(a) $d = 35$ **(b)** $d = 60$ **(c)** $d = 85$

Figure 6 Differences in final image triangulation depending on density parameter to subsample pixels above the threshold.

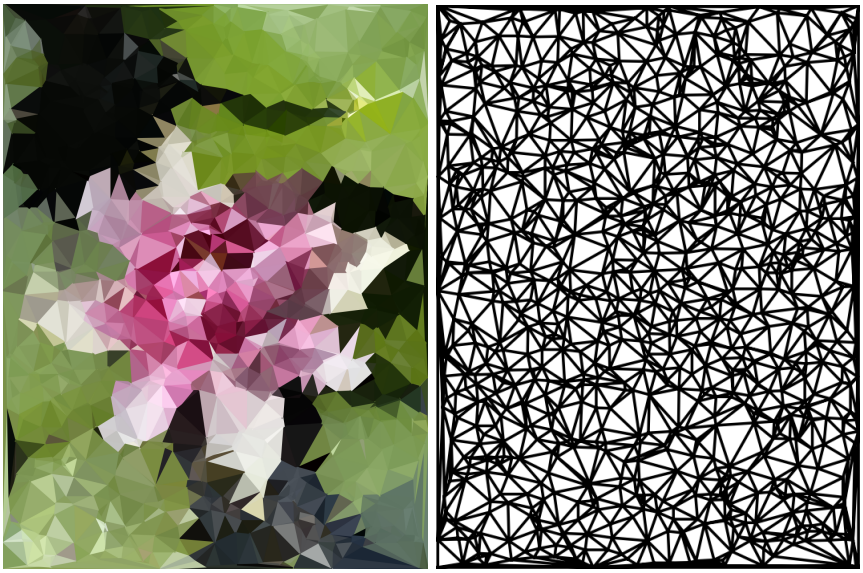


Figure 7 Triangulation using randomly selected pixels as vertices.

4 Conclusion

In this brief article, we describe an algorithm to triangulate an image, detailed in the Github repository [2]. This repository also contains a video of several examples of resulting artistic triangulated images. While the algorithm we outline successfully triangulates any image, the ideal threshold value and density reduction parameter are subjective. If a user desires a more abstract image, a higher threshold value, higher density reduction parameter, or randomized point cloud is suitable. However, if a user is aspiring for a triangulated image closer to the original image, the opposite holds true. Alternatively to an image triangulation, a user may instead wish to consider the dual graph of the Delaunay triangulation. Each Voronoi region would then be colored accordingly instead of each triangle. The output would be more similar to a mosaic and could be considered a separate form of art.

References

1 Olivia X. Laske and Lori Ziegelmeier. OliviaLaske/ImageTriangulation. Software, swhId: swh:1:dir:9ffab16fab1c54439e4641ae8167740d984993b8, (visited on 21/05/2024). URL: https://github.com/OliviaLaske/ImageTriangulation.

2 Laske, Olivia. Source code, *Image Triangulation*. https://github.com/OliviaLaske/ImageTriangulation, 2024. Accessed March 2024.

3 Kai Lawonn and Tobias Günther. Stylized image triangulation. *Computer Graphics Forum*, 38(1):221–234, 2019. URL: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13526.

4 David Marwood, Pascal Massimino, Michele Covell, and Shumeet Baluja. Representing images in 200 bytes: Compression via triangulation, 2018. arXiv:1809.02257.

5 Godwin Onoja and Terhemen Aboiyar. Digital image segmentation using delaunay triangulation algorithm. *Nigerian Annals of Pure and Applied Sciences*, 3:268–283, July 2020. doi:10.46912/napas.83.

6 Tuan D. Pham. Kriging-weighted laplacian kernels for grayscale image sharpening. *IEEE Access*, 10:57094–57106, 2022. doi:10.1109/ACCESS.2022.3178607.

7 Simo, Endre. Delaunay Image Triangulation. https://www.esimov.com/2019/04/image-triangulation-in-go, 2019. Accessed March 2024.

8 Run Tian, Guiling Sun, Xiaochao Liu, and Bowen Zheng. Sobel edge detection based on weighted nuclear norm minimization image denoising. *Electronics*, 10:655, March 2021. doi:10.3390/electronics10060655.