Artistic Enhancement of Image Edges using Directional Pseudo-coloring

Shouvik Mani
C3, Inc.
Redwood City, CA 94063
shouvik.mani@c3iot.ai

Abstract

Computing the gradient of an image is a common step in computer vision pipelines. The image gradient is used to quantify the magnitude and direction of edges in an image and often serve as features for downstream machine learning tasks. Typically, the image gradient is represented as a grayscale image. This paper presents an approach to color the image gradient in a deliberate, coherent, and artistic manner. By using the direction of the image gradient to pseudo-color the magnitude of the image gradient, we can enhance the visual quality of image edges. The result is an artistic representation of the original image, with edges colored consistently according to their direction.

1 Introduction

Edge detection is a fundamental image processing technique that involves computing an image gradient to quantify the magnitude and direction of edges in an image. In this paper, we introduce a technique to color the edges of an image in a deliberate, coherent, and artistic manner. Examples of the resulting colored image edges are provided in the appendix, Section 4.1.

Image gradients are important because they serve as components for various downstream tasks in computer vision such as edge detection and image classification. However, the image gradients themselves are dull. Image gradients are typically represented as grayscale or black-and-white images showing the change in intensity at each pixel of the original image [4].

The goal of this paper is to make image gradients more appealing and informative by enhancing them with color. The human eye can discern only two-dozen shades of gray, but thousands of shades of color [5]. A richer, colored image gradient will reveal details that are not visible in the grayscale gradient. Further, we hope that a colored image gradient will create an artistic rendering of the original image that is aesthetically pleasing.

2 Methodology

The proposed pipeline for coloring image edges has two key parts: edge detection and edge coloring.

2.1 Edge detection

The procedure for edge detection is described in detail in the appendix, Section 4.2. From the edge detection process, we obtain an image gradient with magnitude G and direction θ. An example of the gradient magnitude G is shown in the middle panel of Figure 1. G and θ serve as inputs to edge coloring.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2.2 Edge coloring

Once the edges have been detected, there are several ways to color them. One possible approach is to color the edges using the same colors from the original image, masking the thresholded gradient magnitude image over the original image. However, this does not add any novelty to the final image.

Instead, we choose to pseudo-color the gradient magnitude image using values from the gradient direction image. Pseudo-coloring is the process of mapping the shades of gray in a grayscale image to colors using a color map [2]. A color map, defined in line (1), is a function that maps normalized numerical values to RGB pixel values along some color spectrum [6].

$$color\_map : [0,1] \rightarrow (R \in [0,1], G \in [0,1], B \in [0,1]) \tag{1}$$

After selecting a color map, we transform our gradient magnitude and direction to prepare them for coloring. First, in line (2), we threshold the gradient magnitude image, converting pixel $G_{x,y}$ to 1 if it has intensity above a threshold value t, and 0 otherwise. This thresholding turns the gradient magnitude into a black-and-white image T. Then, in line (3), we mask the thresholded gradient magnitude over the gradient direction to create the colored image C. If $T_{x,y}$ is 0, then the pixel $C_{x,y}$ becomes a black pixel (R=0, G=0, B=0). Otherwise, if $T_{x,y}$ is 1, we normalize the pixel's gradient direction $\theta_{x,y}$ between 0 and 1 and apply the color map to color pixel $C_{x,y}$. As a result, C is an image of the thresholded gradient magnitude pseudo-colored by the normalized gradient direction.

$$T_{x,y} = \begin{cases} 1 & \text{if } G_{x,y} \geq t \\ 0 & \text{otherwise} \end{cases} \quad , \forall x \in X, \forall y \in Y \tag{2}$$

$$C_{x,y} = \begin{cases} color\_map\left(\frac{\theta_{x,y} - min(\theta)}{max(\theta)}\right) & \text{if } T_{x,y} = 1 \\ (R = 0, G = 0, B = 0) & \text{otherwise} \end{cases} \quad , \forall x \in X, \forall y \in Y \tag{3}$$

Because of this directional pseudo-coloring technique, edges along the same direction are assigned the same color, adding consistency and coherency to the image. The resulting image is also surprisingly novel, with edges of multiple colors contrasted against a black background. Examples of the colored images are shown in the right-hand panel of Figure 1 and in Section 4.2.

Figure 1: A demonstration of the approach. Left: the original image. Middle: the image gradient magnitude. Right: the image gradient magnitude pseudo-colored by the image gradient direction.

3 Conclusion

We have presented an algorithm to color the edges of an image artistically. This approach leverages both the gradient image magnitude and direction to color image edges. By enhancing image gradients with color, we can better visualize edges and create new artistic renderings of images.

Acknowledgments

I would like to thank my mentors at Carnegie Mellon University for their support on this project. Specifically, I am thankful for Dr. Ioannis Gkioulekas' insightful and passionate instruction in his Computer Vision class. I am also grateful to Dr. Eunsu Kang and Dr. Barnabas Poczos, whose Art and Machine Learning class provided a venue for arts-computing projects such as this one.

References

[1] Gkioulekas, I. (2018). Image filtering. `http://www.cs.cmu.edu/~16385/lectures/lecture2.pdf`.

[2] Radewan, C. (1975). Digital image processing with pseudo-color. *Acquisition and Analysis of Pictorial Data*. `https://doi.org/10.1117/12.954071`.

[3] Sobel, I. (1968). An isotropic 3x3 image gradient operator.

[4] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*.

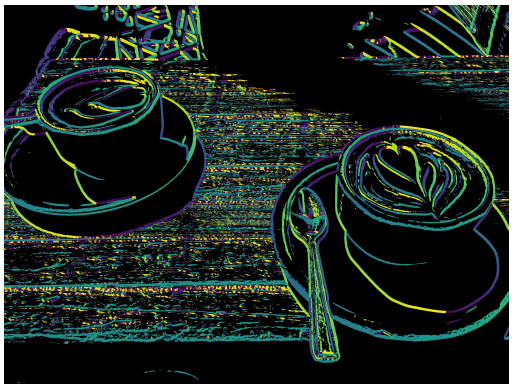
[5] Valentinuzzi, M. (2004). *Understanding the Human Machine: A Primer for Bioengineering*.

[6] Wang, R. (2016). Color and pseudo-color images. `http://fourier.eng.hmc.edu/e161/lectures/digital_image/node4.html`.

4 Appendix

4.1 Examples of image edge coloring

Here, we present a few examples of results from the edge coloring pipeline. The original images are shown on the left along with their colored image edges on the right.

4.2 Edge detection

The standard approach for edge detection is to convolve an image with a derivative filter like the Sobel filter to differentiate the intensity of the image pixels with respect to the horizontal and vertical directions of the image. However, since differentiation is very sensitive to noise, it is useful to blur the image first [1]. This can be achieved by convolving the original image f with a blur filter, such as the Gaussian filter, to produce a blurred image f_b. For the purposes of edge detection, we will first convert f to a grayscale image. Figure 2 shows the blur transformation.

$$f_b = f * \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4}$$

Figure 2: Blurring denoises the image and makes it easier to detect edges. The original image (left) is converted to grayscale (middle) and then blurred (right) using a blur filter.

Once the image has been blurred, we can perform edge detection using Sobel filters [3]. We convolve the blurred image f_b with the horizontal and vertical Sobel filters, S_x and S_y respectively, to approximate the image derivatives in the horizontal and vertical directions, $\frac{\partial f_b}{\partial x}$ and $\frac{\partial f_b}{\partial y}$ respectively. $\frac{\partial f_b}{\partial x}$ has a strong response to vertical edges, and $\frac{\partial f_b}{\partial y}$ has a strong response to horizontal edges.

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{5}$$

$$\frac{\partial f_b}{\partial x} = f_b * S_x \tag{6}$$

$$\frac{\partial f_b}{\partial y} = f_b * S_y \tag{7}$$

$\frac{\partial f_b}{\partial x}$ and $\frac{\partial f_b}{\partial y}$ can be combined to form the image gradient ∇f_b. The following equations show how to compute the magnitude G and direction θ of the image gradient. The resulting gradient magnitude represents the intensity of edges in the image and is displayed in Figure 3.

$$\nabla f_b = \left[\frac{\partial f_b}{\partial x}, \frac{\partial f_b}{\partial y}\right] \tag{8}$$

$$G = ||\nabla f_b|| = \sqrt{\left(\frac{\partial f_b}{\partial x}\right)^2 + \left(\frac{\partial f_b}{\partial y}\right)^2} \tag{9}$$

$$\theta = tan^{-1}\left(\frac{\partial f_b}{\partial x} / \frac{\partial f_b}{\partial y}\right) \tag{10}$$

Figure 3: Edge detection using Sobel filters. We convolve the blurred image (left) with the Sobel filters and compute the magnitude (middle) and direction (right) of the image gradient.

4.3 Code

The code to reproduce these experiments is available in the following Jupyter Notebook: `https://github.com/shouvikmani/edge-colorizer/blob/master/edge_colorizer.ipynb`.