15-463 Final Project: Non-Photorealistic Rendering

Lily Chen
Carnegie Mellon University
lyc1@andrew.cmu.edu
https://lilyychen96.github.io/

Tony Lu
Carnegie Mellon University
tianyua1@andrew.cmu.edu

Abstract

Non-photorealistic rendering (NPR) combines computer graphics with artistic techniques to generate non-photorealistic images. We were interested in taking a more fine arts approach to computational photography through non-photorealistic rendering, or NPR. In this paper, we examine several NPR methods for processing images in accordance to art movements like impressionism, expressionism, and cubism, as well as contemporary applications such as cel-shading. This paper is split into three sections and proceeds through the simple but novel processes for the impressionism, Cubism, and cel-shading rendering methods with sample images and results.

1. Impressionism

Impressionism is a late 19th-century art movement characterized by small brush strokes that, from up close looked like a series of dabs of colors but from far away, provided an "impression" of everyday objects and scenes. This section of the paper was inspired by Litwinowicz to render individual images for a hand-painted look [7].

1.1. Background

While impressionism, pointillism, and expressionism all have their own distinguishing styles, each art movement could be characterized by the same base implementation. For instance, pointillism features very short brush strokes with a greater variation in color in adjacent areas; impressionism and expressionism tend to have rich, vibrant colors, but expressionism is darker with more agitated brush strokes.

1.2. Stroke parameters

On a scale of 1-10, with 1-3 as differing "degrees" of pointillism, 4-7 for impressionism, and 8-10 for expressionism, the definition of the brush strokes could be calculated for each painting style. After finding the edges in the input image, these brush stroke parameters were calculated according to the desired style and image size. Litwinowicz suggested detecting edges in an image with the Sobel operator; however, the Sobel operator does not give well-defined edges and may introduce "noise" in the resulting edge map. Instead, we opted for the Canny edge detector, which gives well-defined edges.

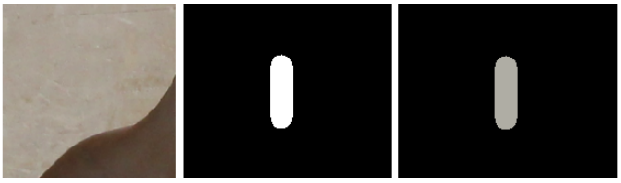

Figure 1. A section of the image along with the brush stroke mask and resulting brush stroke with color.

Brush stroke lengths depended on the input "degree" of each style, save for pointillism, which had a general range of 1 pixel to a length of 3 pixels.

Brush radii between the styles had a range between 1 pixel and either 5 pixels or, depending on the size of the input image, image height divided by 150. For pointillism, brush stroke radii had a greater range due to its short brush stroke lengths.

To increase the painterly appearance, random variations and perturbations were added. A range for the color perturbation was calculated according to the style, which differed between and within styles, and applied to each color channel for each brush stroke.

For the impressionism and expressionism styles, the image was converted to the HSV color space to increase the saturation of the image to allow for a more vibrant rendered image. For impressionism, the saturation was scaled by 1.2; for expressionism, the saturation was scaled by 1.4 while the overall values were scaled by 0.95 to results in a darker but still more vibrant rendered output.

Lastly, a threshold was determined for each style that

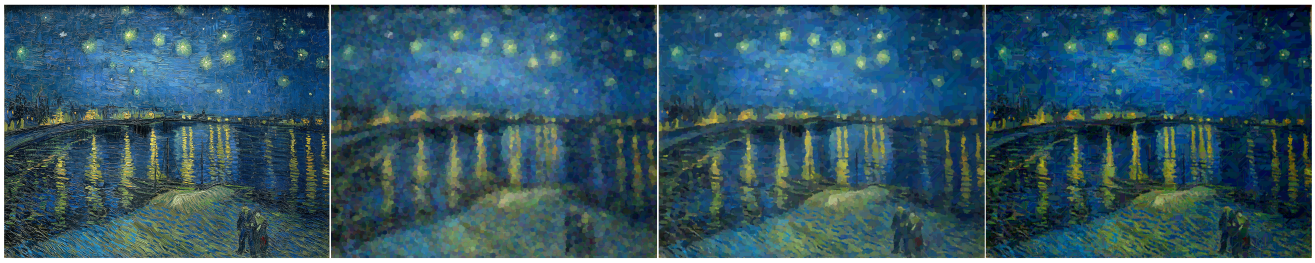

Figure 2. (From left to right): The input image, "Starry Night on the Rhone" by Van Gogh; a pointillist rendering; an impressionist rendering; an expressionist rendering. Note that Van Gogh was a late-Impressionist early-Expressionist painter.

was used to determine the clipping of a brush stroke when it reaches an edge.

1.3. Stroke orientation and magnitude

The Prewitt operator was used to compute an approximation of the gradient of the image intensity in magnitude and direction. Because the gradient directions are calculated to be perpendicular to edges, 90°was added to the directions and clipped to range from -180°to 180°. Since the output of the Prewitt operator gave small magnitudes for areas away from edges, the gradient magnitude was subtracted the maximum of the gradient magnitudes and added a perturbation of 0.0001 to ensure non-zero magnitudes and to "reverse" the magnitudes for the brush strokes, which was used to calculate the brush stroke lengths [7].

1.4. Rendering strokes

By picking a random row and column in the image, the rendering of strokes was rather efficient and often took a long time to render the entire image. However, due to the nature of the "painting", in which brush stroke often overlapped one another, this allowed for a greater likeliness to a real painting, in which painters will often paint over the same area several times with different colors, length, size, and other parameters.

For each stroke mask, a random pixel of the image was selected as the brush stroke starting point center. From that pixel, the gradient orientation was found and convert to radians. For an angle of 0°, a random angle was chosen. The brush stroke length was found by scaling the gradient magnitude at that pixel, clipped if it exceeded the predetermined minimum and maximum lengths found when calculating brush stroke parameters. If the brush stroke starting point center was near an edge (according to the threshold determined by the brush stroke parameters section), the brush stroke radius was small, between 1 and the style number in pixels. Conversely, brush stroke starting point centers that were further away from edges allowed for a larger brush stroke radius.

A circular mask is constructed according to the brush stroke radius. Then the mask is lengthened by traveling in the gradient direction, clipping the length if the mask crosses an edge or exceeds the image size.

After the brush stroke mask is created, the average value for each color channel of each pixel in the input image that falls within the mask is found and added with a random color perturbation, as described in the stroke parameters section [7]. Then the brush stroke is "painted" onto the output image before repeating the entire processes again until the entire output image is painted.

1.5. Future work

Currently, the algorithm is excessively slow due to the random picking of pixels for the brush stroke starting points. To increase the speed of the rendering, an array of pixel locations at a specified sampling rate across the image would be beneficial to speeding up the process. When attempting this, it resulted in actual rows and columns of brush strokes in the image, leading to an undesired result.

Since the rendering was extremely slow, the input images were often downsampled for faster rendering times. Consequently, the detailing in the images were often obscured away. A method for remedying this issue would be to construct an image pyramid of differing sampling rates with which the image could be rendered at each level and superpositioned such that areas with lots of detailing will not be blurred away.

In addition, adding texture to the brush stroke mask could have increased the painterly appearance and possibly increased the speed of the rendering.

2. Cubism

Cubism is an art movement dating back to the early 20th century. It is distinguished by geometric forms and sharp angles. Objects are broken up and reassembled in an abstract manner. The most well-known Cubist artist was arguably Pablo Picasso. Most of our algorithm is adapted

Figure 3. A portion of an input image along with its rectangular distortion.

from Collomosse, although many parts were left out or simplified [4].

2.1. Image splitting

To get the angular forms in Cubist artwork, the input image is first split into 36 circles. The centers are spaced equally in a 6 by 6 grid in the image, and the radii are randomly sampled to be slightly greater than one-sixth of the minimum image dimension. Pixels in these circles are mapped to an enclosing concentric superquadric region. Superquadrics are of the form

$$\left(\frac{x}{a}\right)^{\frac{2}{\alpha}} + \left(\frac{y}{b}\right)^{\frac{2}{\alpha}} = r^{\frac{2}{\alpha}}$$

As α approaches zero, the shape tends toward a rectangle, which is how angular forms are achieved. a was randomly sampled to be between 0.2 and 0.8, and b was set as $1 - a$. I used $\alpha = 0.01$, and I set r equal to two times the radius of the corresponding circle, which prevented the superquadric from being too skinny for small values of a or b.

2.2. Pixel mapping

An angle in $[-\frac{\pi}{2}, \frac{\pi}{2}]$ is sampled and the superquadric is rotated according to this angle. The colors are blurred to give a painted texture. The pixels are then mapped from circles to their corresponding superquadrics. This mapping is done in random order to prevent overlaid superquadrics from looking as if they are patterned. An example mapping for a given circle is shown in Figure 3. After all circles have been mapped to their corresponding superquadrics, regions that have not been filled are filled with a blurred version of their corresponding pixels in the input image. Some examples of images and their outputs are shown in Figure 4.

2.3. Future work

The algorithm is currently very rudimentary, and many improvements can be made. In Picasso's works, facial features are still very distinguishable despite the angularity.

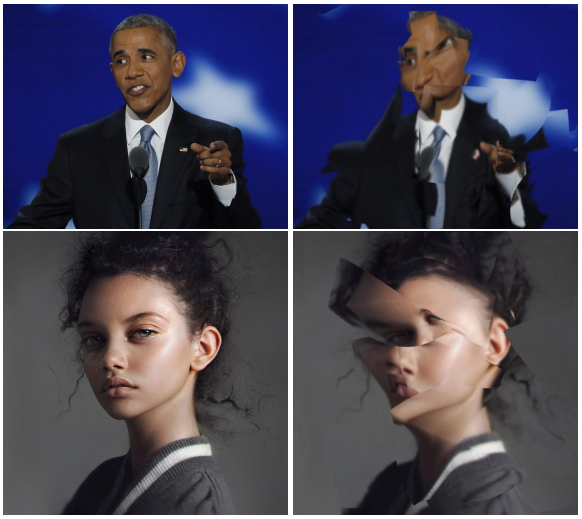

Figure 4. Some examples of Cubist renderings. The left image is the source image, while the right image is the output.

Collomosse accounts for this by identifying salient features and then fitting superquadrics to those regions of the image rather than placing circles at fixed points. This prevents facial features from being split apart or being distorted too heavily.

For a more painterly rendering, Collomosse uses a gradient across background sections of the image. They also use a region adjacency graph to apply different gradient directions to adjacent sections. To enhance the texture, they quantize colors to mimic how an artist paints with a restricted palette. Brush strokes are also implemented in 3D as inverted cones and then project to 2D based on z-buffering.

3. Cel-Shading

Cel-shading, or toon shading, is a type of rendering that makes 3D objects appear 2D by lessening shade gradients and explicitly define edges, creating a cartoon or comic book-like effect. While typically used on 3D models, we wanted to see how well this type of rendering could be applied to a 2D image.

3.1. Bilateral filtering

The bilateral filter was used to preserve edges while provide a degree of smoothing over all other areas. While most filters may process color channels separately, the bilateral filter can operate on the three channels at one and determine which are perceptually similar or not [8]. It replaces each pixel intensity with a weighted average across the color channels of nearby pixels.

For MATLAB versions 2018a and 2018b, there is the *im-*

Figure 5. The input and resulting output images for the cel-shading script. Notice the extra edges detected in the sky and the jagged boundaries between segments.

bilatfilt function that implements Tomasi's paper by finding the variance of the pixel values in accordance to Euclidean distance and filtering with the bilateral filter [1]. For MATLAB versions earlier than the 2018 series, the script *bfilter2.m* is a compatible implementation of Tomasi's same paper [6].

3.2. Color segmentation

With the bilateral filtered RGB image, we used K-means clustering to determine the segments of color within the image. For this, we wanted a maximum of 25 colors in the image so we performed k-means clustering with a k value of 25.

For MATLAB versions 2018a and 2018b, there is the *imsegkmeans* function that will perform the k-means clustering segmentation [3]. For MATLAB versions earlier than the 2018 series, a different process is described in MATLAB documentation [2].

Then, for each cluster, the average intensity of each color channel is averaged according to the mask for each cluster [2]. Finally, each mask is combined to form the color segmented image.

Figure 6. Cel-shading render of "Amore e Psiche" by Antonio Canova.

3.3. Edge detection

Like the impressionism rendering process, the edge map was determined with the Canny edge detector with sigma value of 4. After finding the bilateral filtered and color segmented image, the edge map was overlaid on top to explicitly define the edges of the object in the image.

3.4. Future work

Because of a combination of the Canny edge detector and the downsizing of the images, some details of the edges map were lost while others were augmented. Creating an edge map with a Derivative of Gaussian (DoG) or Laplacian of Gaussian (LoG) filter would have allowed for thicker edges but would also add an increased but less stark depth to the edges, allowing for a smoother cel-shading effect. Alternatively, using the Sobel operator as suggested by Litwinowicz would have provided the same type of fuzzy edge detection as the DoG filter.

Furthermore, the color segmentation edges were rather rough and serrated. To remedy this, constructing an image pyramid of set sampling rates between each level and finding the segmentation for each level would allow for smoother regions of segmentation [5]. However, this does not necessarily output less rough boundaries between the segments.

4. Code

Our code is currently open-source on Github and may be subject to being made private. Our project can be found with the repository name **NPR-Art-Filters**.

References

[1] Bilateral filtering of images with gaussian kernels. Accessed: 2018-12-15.

[2] Color-based segmentation using k-means clustering. Accessed: 2018-12-15.

[3] K-means clustering based image segmentation. Accessed: 2018-12-15.

[4] J. Collomosse and P. Hall. Cubist style rendering from photographs. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):443–453, 2003.

[5] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. 21(3):769–776, 2002.

[6] D. Lanman. Bilateral filtering, 2006. Accessed: 2018-12-15.

[7] P. Litwinowicz. Processing images and video for an impressionist effect. pages 407–414, 1997.

[8] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. pages 839–846, 1998.

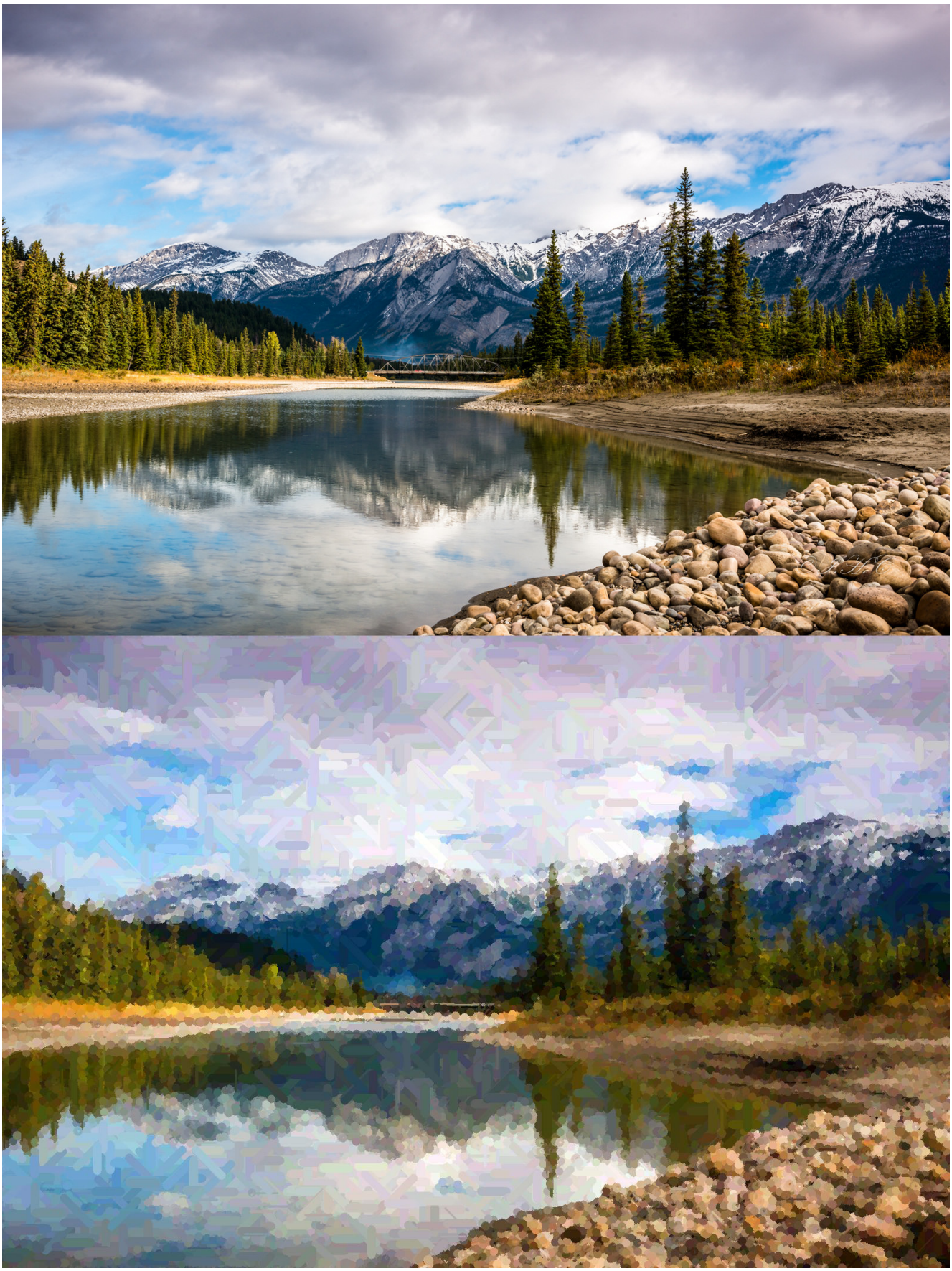

Figure 7. Impressionist rendering of Jasper National Park, Canada.