**Painterly Rendering**

SeungJun Lee

Portland State University

**Abstract**

I will be presenting a method, based on the paper *Painterly Rendering with Curved Brush Strokes of Multiple Sizes* by Aaron Hertzmann [1], that renders a given image into a hand-painted processed image. The output image will be drawn in layers with different brush stroke sizes to mimic the hand painting skills of layering. Layering with only adding brush stroke to the required area will create different texture through multiple brush sizes displaying. The difference between the canvas, the output image, and the reference, blurred image of the input, will guide the algorithm to add specific layers. In this process, I will be focusing on the pointillism technique, which is done through dotted points to show the acrylic paint and oil paint methods.

Throughout the paper, I will be describing my algorithm while showing the results and examples. Furthermore, I will be discussing the limitations and future works.

1. **Introduction**

Technological advancement has brought high pixelated and high resolution photography accessible to many people. Even with the accessibility of high quality photos, people still remained yearning for hand created art works. With the development and computer technology, developers were able to bring different methods of altering and modifying a raw image. Even with the given multitudes of image processing methods, it was still different with the hand processed art, such as drawing or paintings.

Currently, there are multiple different accessible rendering algorithms that could alter a given image into a hand painted or hand sketched output image. Although the art may be same and the methodology may be same, each artists display different feelings from the final output. This was the reason behind implementing and modifying the pre-existing algorithm [1], to potentially have a different flare and taste to the painterly rendering in pointillism method.

* 1. **Related Works**

Overall algorithm implemented throughout the rendering process is solely based on the paper, *Painterly Rendering with Curved Brush Strokes of Multiple Sizes* by Aaron Hertzmann [1], and I do not take any credit towards the creation of the algorithms.

* 1. **Overview**

In the following section, I will be showcasing the method behind the painterly rendering and follow with section 3, which will be providing with examples and results of the implementation.

1. **Method**

Common and base method for creating a painting is to start with a layering. The layering process is done through different brush sizes, which begins with the largest then ends with the smallest. Largest brush will start structuring the painting while smallest brush will be refining the details of the image. Following the traditional method, there are clear difference between hand paintings and image processing, which are the image size. The artist can have small or large canvas but will have to work in finer detail with the same small brush due to the physicality of the canvas, but in computer image standpoint, the viewing port’s size is same while the image size is different. The difference can be managed by creating different sets of brush sizes. A separate algorithm will take in the image size and determine the minimum brush size to maximize the granulate texture while keeping the finer details. If the brush is too small for the image size, then it will produce a slight blurred version of the image because it will refine to finely. On the other hand, if the brush is too large for the image size, then it creates a blob texture without any details remaining. By creating the correct brush size, it stays between the blur and the blob. After creating the minimum brush size, create a brush queue, which iterates for creating layering technique. The brush queue will contain different brush sizes ranging from **R1 … Rn**, with **R1** being the largest.

For each iteration of the brush queue, a reference image is created via applying the source image with Gaussian blur with **Rcurrent** from the OpenCV library. By applying the Gaussian blur, the image will have less sharp edging and allows the brush to be applied in the general area. The reference image is then subtracted with the canvas, the output image, to create a difference map. The difference map will be displaying the difference in color, black being no difference, and the brush stroke will be added to the canvas where the difference map indicates. By utilizing the difference map, we can assure to not overlay any unnecessary areas and retain the layering effect of different brush sizes.

A picture containing indoor, dark, sitting, person

Description automatically generated

**Figure 1.** Difference map indicating the pixels that have the color difference. The colored region indicates the RGB values subtracted by the canvas image from the reference image.

For the last iteration of the brush size, **Rn** the smallest brush size, we change the difference map creating. Throughout the iteration, we’ve created the difference map by subtracting the canvas from the reference. This allows the difference map to not catch darker region of the source image, such as eye retina, shadows, and subtle shading. Due to difference map’s indication of change, which is color and not black, the algorithm ignores the darkened region of the difference map. This problem leads to not painting the eye or shades on the contouring. By subtracting the canvas with the reference, it shows the darkened area for the smallest brush to go over and does the final touch ups needed.

The iteration of the brush is processed with few other iteration to step through the difference map. The brush size will be increased to act as the stepping size of the difference map to minimize the overlapping of the same brush strokes. The traverse through the difference map will start from the top left corner then finish with the bottom right corner of the image. When the difference is found, through difference map’s color indication, same point of the reference color will be extracted then painted onto the canvas.

Unlike the original paper [1], this algorithm does not iterate through until no difference has been detected. In my belief, the errors and the rigidness of the output image shows human-like errors that creates art-like artifacts.

1. **Experiment**

**A picture containing person, indoor, food, man

Description automatically generated**

**Figure 2.** Example used from the original paper

A close up of a fruit

Description automatically generated

**Figure 3.** Original paper’s output image

**A picture containing indoor, food, cake, table

Description automatically generated**

**Figure 4.** Output image from this algorithm

As shown above, the original paper [1], resulted in a much smoother textured painted version while current paper’s algorithm has displayed much rigid and rough textured image. This difference comes from the original paper’s iteration until no difference. Both result displays a paint-like texture and look while avoiding the general blurring of an image.

A person posing for the camera

Description automatically generated

**Figure 5.** Portrait image with a detailed hair display

A close up of a mans face

Description automatically generated

**Figure 6.** Output of the figure 5

Testing the algorithm on the finer detail, such as hair strands and subtle skin tone changes resulted in positive. Although the hair strands aren’t perfectly detailed, it showed that the algorithm saw the difference in the background and the hair strands and tried to add the trends to it. The edges between the portrait and the background has a harsh transition but within the facial skin tone transition looks promising.

A picture containing person, wall, sitting, phone

Description automatically generated

**Figure 7.** Portrait image that has similar background and skin color tone

A picture containing person, man, outdoor, sitting

Description automatically generated

**Figure 8.** Output of the figure 7

Testing onto an image that contains a slight similarity between the background and the skin tone showed that the algorithm is capable of drawing a line between the two. Although the algorithm was able to keep the separation between the background and the human and the pants, this image particularly resulted in a blurred-like image. It looks like the finishing brush size was too small and resulted in a too detail-oriented output that made it look like a blurred image of the source.

A picture containing person, wall, indoor, clothing

Description automatically generated

**Figure 9.** Image with the distinct detail

A person wearing a white shirt and black hair

Description automatically generated

**Figure 10.** Output of the figure 9

This was to test the algorithm on how well the distinct features of the source image it can retain. From the output image, it showed that the algorithm had no problem in keeping the detailed feature from the source image. The color gradient in the upper right lips displays that algorithm’s ability to keep a fine detail.

A person looking at the camera

Description automatically generated

**Figure 11.** Image with small and distinct detailed features

A person looking at the camera

Description automatically generated

**Figure 12.** Output of the figure 11

This image was to test the algorithm’s ability to retain a small and fine details of the image. In figure 11, the subject contains a small and fine details with similar color tone on his face. The output tried to mimic the color tone and the details but due to the brush size, it was not able to follow through. Although the algorithm was able to detect and tried, it lacks in detail especially on the subject’s forehead area.

A person posing for the camera

Description automatically generated

**Figure 13.** Image in black and white

A person posing for the camera

Description automatically generated

**Figure 14.** Output of the figure 13

The image was to test the algorithm’s ability to translate black and white image into painterly rendering. Painterly rendering heavily rely on the mesh and overlaying of colors because black and white has less gradient capability. Due to losing the capabilities of layering color palette and having a subtle color gradients, black and white picture did not do well.

1. **Discussion**

I have implemented the painterly rendering [1] with the pointillism brush strokes to create an image into a hand-painted output. Although it showed lacking in finer details, such as small-in-size color difference and edge trimming, it showed great promise in color gradation and authentic human-like feeling of rough and rigid edges. The issues in this algorithm is to create a perfectly fitting minimum brush size to accommodate the finer details while avoid being too detailed. The image size and the brush size was not linearly related and figuring out which polynomial relation is also the key issue.

Black and white images also has issue with the algorithm. Due to losing the ability to utilize color, loses the paint-like feeling and brings the blurred-like image output. This issue is mostly likely due to the personal translation of a painting style but black and white would fit more into sketch-based rendering.

**4.1. Future works**

I personally loved doing this project due to my personal interest in art and painting. For future works, I will be working on finer detailing with different brush stroke implementation. Implementing long brush stroke for areas of similar color tone and object edging will bring much life into the painting. Also implementing paint type might be a difficult but promising task, such as implementing certain level of thickness of paint or glistening of the oil.

1. **References**

[1] Hertzmann, Aaron. *Painterly Rendering with Curved Brush Strokes of Multiple Sizes*. Media Research Laboratory Department of Computer Science New York University, www.mrl.nyu.edu/publications/painterly98/hertzmann-siggraph98.pdf.