Create Pointillism Art from Digital Images
Analysis of pointillism art algorithm

Yanshu Hong
Department of Philosophy
Stanford University
Stanford, CA
yanshuh@stanford.edu

Tiffany Liu
Department of Electrical Engineering
Stanford University
Stanford, CA
tiffliu@stanford.edu

II. RELATED WORK

The field of non-photorealistic rendering has developed various methods to create paintings with computers. One of the first papers on rendering paintings with computers is Hertzman’s work on using different sized curved brush strokes for painting rendering [3]. Others have expanded his research to include painting with brushstrokes of various shapes and sizes and optimization of brushstroke orientation and placement depending on edge features in an image. For example, Chi and Liu presented a paper that describes the use of ellipsoid dots that are oriented according to a direction map to enhance details in a pointillism image [1]. They also use the intensity map of the image to determine the size of each ellipsoid dot. We also incorporate the intensity map in our algorithm, which is described in section III.C. Another paper by Krüger and Wörgötter expands from using dots to using multicolored icons to reproduce pointillism-like artwork. Their technique produces images with more discrete patterns of color [5].

Other methods in NPR involve experimentation of colors in an image. Analysis of paintings from particular artists or scenes and application of modern color theory can be used to determine what colors should be in a painting. For example, Wu presents a paper that uses machine learning techniques to analyze Seurat’s paintings to learn the color statistics in his pointillism art to generate pointillism paintings that match Seurat’s style [10]. Yang presents a paper that uses a combination of modern color theory and incorporation of the halo effect found in Seurat’s paintings to generate images in Seurat’s pointillism style [11]. Sugita and Takahashi describe a method of mixing colors by half-toning dots together to mimic the optical mixture of dots present in Seurat’s art [9]. We draw ideas from these various papers to create our own method of creating pointillism art. While these papers focus on mimicking features of famous artists’ pointillism style, this paper describes an algorithm for generating pointillism art that has the flexibility to fit the user’s artistic taste. Furthermore, our algorithm stands out from others because we are recreating a new image according to the original image. The computer is practically painting a new image with various pointillism features “in mind”.

Abstract—Pointillism is a branch of impressionism that can be dated back to the late 19th century. It is a painting technique that only uses tiny, distinct dots to form patterns of color. It enjoys a duality in being both discrete up close as dots and continuous from distance as patterns. Combining this artistic inspiration with the techniques of digital image processing, we developed an image-processing algorithm that creates pointillism art from ordinary digital images. We evaluated various aspects of the algorithm and analyzed their effect on the aesthetics of the resulting image. We also identified specific steps that affect the aesthetics the most. Finally, since art is subjective, we proposed one possible set of parameters for the algorithm that we deem to work the best.

Keywords—pointillism; image processing; non-photorealistic rendering; color theory; K-means clustering

I. INTRODUCTION

As computer graphics research develops new methods to mimicking reality, there has also been a movement to develop methods to abstract reality, known as non-photorealistic rendering, or NPR. One non-photorealistic rendering method we are interested in investigating is reconstructing natural images in impressionism styles, specifically pointillism. Pointillism is an artistic method that uses small distinct dots of primary colors to paint an image [8]. Pointillism artists take advantage of how the human visual system blends small discrete dots of color into other colors. Therefore, if a viewer looks at a pointillism painting up close, the painting looks like a bunch of randomly colored dots, but if he or she steps back, the dots blend together to form colors, which make up an image. In this paper, we describe our methodology to design and implement an algorithm to create pointillism art from natural images. The rest of the paper is organized as the following. Section II reviews previous and existing work related to developing pointillism graphics. Section III describes our methodology and algorithm design. Section IV shows one resulting image with our optimal parameters with respects to our subjective evaluation of aesthetics. It also includes suggestions for future work. Additional resulting images created from our optimal parameters are included in the Appendix.
III. ALGORITHM

Our pointillism algorithm can be broken into multiple steps as shown in Fig. 1. The general workflow is to first create a color palette, then convert every pixel in the original image into a cluster of dots with these specific colors and finally “paint” these dots onto the canvas. The orange colored boxes in Fig. 1 denote steps with “artistic potential”, where users can make the most noticeable changes to the final image to fit their aesthetic preference.

A. Preprocessing

Input images tend to be fairly large and the output image can be tens of times larger than the input image. To reduce the amount of computations in our pointillism algorithm, the image will be resized by subsampling. Prior to subsampling, the image is filtered with a low pass filter to reduce aliasing that may result from the subsequent subsampling. The low pass filter is a normalized Gaussian filter. After the image is subsampled, the resulting image still retains its details. The downsampled factor is determined by the size of the dot cluster upon image generation (discussed in section III.D).

We didn’t add an additional filter after downsampling as outlined in Fig. 1, but such possibilities are discussed in Section IV.B.

B. Color Selection

After the image is reduced in size, we identify the colors we want to use to create our pointillism image. Seurat, the father of pointillism, was known to use at most 11 colors in his paintings. We also use a limited number of colors as our primary colors, 16 to be exact. We first determine our initial 8 primary colors by applying K-means clustering (an unsupervised machine learning technique) to the colors in the input image. In the flow diagram, the use of K-means clustering is also considered an artistic choice because the primary colors can be chosen in many different ways. For example, a user may want to use a different set of primary colors that he or she predetermines before analyzing an image. We chose to use K-means clustering to pick colors because we want the final image to look similar in color to the original image. Fig. 2 demonstrates the progression of how we determined the optimal color palette.

1) K-means Clustering Algorithm

K-means algorithm begins by picking 8 random locations in the image as the initial centroids. Then for each pixel in the image, the algorithm assigns it to the centroid with the shortest Euclidean distance to the RGB values of that pixel. New clusters are then formed by classifying each pixel with its assigned centroid. Within each cluster, a new mean of the RGB values of all pixels in that cluster is calculated and the existing centroid is updated to be the new mean [4]. We define a cost function that sums the distances between each pixel color values and its centroid color values. The algorithm repeats until the cost function converges. As a result, the K-means clustering algorithm finds the 8 most popular colors in the image.

2) RGB to HSV Color Space

After the primary colors of an image are determined, we found that our K-means algorithm tends to find much darker colors than what we like, as shown in Fig. 2(b). In the RGB color space, brightness and saturation of a color are difficult to manipulate because these characteristics are embedded in all three color channels. The HSV color space separates color type (hue), saturation and brightness (value) into three separate components. Therefore, the RGB values of the 8 primary colors are converted to values in the HSV color space.

In the HSV color space, hue (H) is represented by angles in a color wheel (360 degrees of rotation) as opposed to R, G, B coordinates in an RGB cube. Saturation (S), the vibrancy of the color, is represented by values from 0 to 255. Lower saturation values have more gray in the color, making the color seem more faded. Value (V), the brightness of the color, is also represented by values from 0 to 255, with 0 being dark and 255 being bright [7].
3) Color Saturation and Brightness Boosting

In the HSV color space, we experimented with different ways to improve the brightness and saturation of the K-means primary colors to produce a more vivid final image. Equations 1-5 describe the steps we used to determine brighter colors for the resulting pointillism image. The resulting palette of brighter colors creates a brighter image as shown in Fig. 2(c).

\[ K_{\text{color}} = \{R, G, B\} \rightarrow \{H, S, V\} \]  
\[ H_{\text{new}} = H \]  
\[ S_{\text{new}} = (S)^{0.75} + 0.05 \]  
\[ V_{\text{new}} = (V)^{0.75} + 0.05 \]  
\[ \text{primaryColors} = \{H_{\text{new}}, S_{\text{new}}, V_{\text{new}}\} \rightarrow \{R_{\text{new}}, G_{\text{new}}, B_{\text{new}}\} \]  

C. Color Transformation

In the final image, clusters of dots are used to represent a single pixel of the downsampled image. Each cluster consists of dots of three different colors from the color palette. Two colors are chosen in respect to minimum distance to the original pixel’s color in the RGB color space. Then the third color is a randomly chosen color from the remaining 14 colors. We add a random factor to mimic the scenario when an artist mixes a color and the brush might not be clean enough so a random color may be introduced.

Once each pixel’s colors are matched to three colors from the color palate, the number of dots for each of the three colors needs to be determined. Equation 6 determines the ratio of dots per color according to the RGB values of the three primary colors. Equation 7 determines the actual number of dots based on the abovementioned ratio and the total number of dots per cluster. This total number is defined by \( a \), the maximum number of dots in one cluster, and \( l \), the pixel’s intensity value from the downsampled image’s inverse grayscale map. When the grayscale map is created, the contrast of the intensity values is enhanced by applying gamma-distortion of 1.8.

\[
\begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3
\end{bmatrix} = \begin{bmatrix}
C_1(R) & C_1(G) & C_1(B) \\
C_2(R) & C_2(G) & C_2(B) \\
C_3(R) & C_3(G) & C_3(B)
\end{bmatrix}^{-1} \cdot \begin{bmatrix}
P(R) \\
P(G) \\
P(B)
\end{bmatrix}
\]

\[
\begin{bmatrix}
n_1 \\
n_2 \\
n_3
\end{bmatrix} = \begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3
\end{bmatrix} \cdot \frac{al}{Q_1+Q_2+Q_3}
\]

The intensity map ensures that the lighter regions of an image have significantly fewer dots than darker regions of an image. Otherwise all the shading in the image will be derived predominantly by the color and would result in a flat image as shown in Fig. 4. The use of dot concentration to represent intensity in an image is known as stippling.

4) Complement Colors

While Seurat used a limited color palate, analysis of his paintings demonstrated that he used complement colors to possibly enhance features as shown in Fig. 3. In modern color theory, the use of complement colors is known as color juxtaposition. Color juxtaposition is described to be when two colored areas are observed to be quite close to each other, no matter in space or in time, each of the colors will shift its hue and lightness [6]. As described in Yang’s paper, “if a dark red and a light yellow are put together side by side or one after the other, then the red will shift as if it is mixed with the complement color of light yellow, i.e. dark blue, while the yellow will shift as if it is mixed with the complement color of the dark red, i.e. light cyan” [11]. Therefore, when two complement colors are placed side by side, they will have maximized the color juxtaposition shift, increasing the contrast between the two colors. Based on this aspect of color theory, our algorithm also incorporates the use of complement colors in addition to our primary colors to produce a 16-color color palette. The larger color palette produces images with higher contrast. The color complements are calculated in the HSV color space by shifting the hue of a primary color by a random degree uniformly distributed from 0° to 180°. The image generated from a palette of brighter colors and their random complements is shown in Fig. 2(d).
To more accurately represent the color of the original pixel, we want to blend the dot colors instead of letting them overlapping each other. Therefore, we set the opacity of the dots to be less than 100%.

In addition, the opacity of the dots and the maximum number of dots in a dot cluster, \( \alpha \), affect saturation and brightness of the final image. Fig. 5 compares final images with different \( \alpha \) at the same opacity and with different opacity at the same \( \alpha \). The images use the same color palette for consistency between comparisons. Higher \( \alpha \) produces darker and more saturated images as seen in the first row of Fig. 5 because an image with higher \( \alpha \) has more dots per cluster than one with lower \( \alpha \). Images with higher \( \alpha \) will appear noisier because more dots are subject to the random factor in our color selection method. The level of opacity also affects the brightness of the image. Images with lower level of opacity are brighter than ones with higher level of opacity, as seen in the second row of Fig. 5. These details are particularly noticeable in the flag in the background and in the president’s face.

\[
\text{Opacity} = 50\% , \quad \alpha = 50 \quad \alpha = 150 \quad \alpha = 200
\]

\[
\alpha = 100 , \quad 25\% \text{opacity} \quad 50\% \text{opacity} \quad 75\% \text{opacity}
\]

**Figure 5 Comparison of dot density and dot opacity. Top row shows images at 50\% opacity with varying \( \alpha \). Bottom row shows images at \( \alpha = 100 \) with varying opacities.**

**D. Image Generation**

1) Dot Distribution

To create the final image, the algorithm "paints" dots onto a blank canvas. Each dot on the final image is part of a dot cluster, which represents a single pixel from the original image. The center of each dot cluster is "d" pixels away from each other dot cluster center in the resulting image as shown in Fig. 6(a). The dots within one cluster (to be precise, distances from dots to their cluster center) are distributed following a Gaussian distribution parameterized by mean \( \mu \) and variance \( \sigma \). Therefore, 68% of the dots in each dot cluster are within 1\( \sigma \) pixels from the center. The area 1\( \sigma \) within the center of a distribution spans an area of 2\( \sigma \times 2\sigma \) as shown in Fig. 6(b). Each dot cluster is separated by a distance of \( d \), so the amount of blending between dot clusters is dependent on \( d \) and \( \sigma \).

\[
\text{d} = 0.5\sigma \quad \text{d} = 1\sigma \quad \text{d} = 2\sigma
\]

**Figure 6 (a) Pictorial demonstration of how each original pixel is treated in resulting image. (b) Original pixel is represented by Gaussian dot distribution. (c) Graphical depiction of the relationship of pixel distance \( d \) between pixels and size of Gaussian distribution \( \sigma \), with corresponding resulting images below.**

2) Brushstroke Size and Orientation

Each dot is made up of a kernel that maps out the shape of a brushstroke. For our image, the dot is a diagonal rounded brushstroke that spans an 11x11 area. The orientation of each dot is derived from local gradients. For each pixel, we applied four Kirsch operators to find the direction that has the strongest gradient, to which the brushstroke is aligned. Left image of Fig. 7(a) shows when all the brushstrokes are oriented according to the strongest gradients found by the Kirsch operators. Rotating brushstrokes help outline edges, but it also makes the flat regions in the image have a swirling effect. For example, some areas on the president’s suit jacket appear to be swirling when it should be flat. Some people may find this effect more artistic. We decided to apply a threshold to the gradients to limit the appearance of swirls in flat areas. The threshold can be changed
depending on artistic preference. The image to the right in Fig. 7(a) shows an image that has a gradient threshold applied, so brushstrokes in the flat regions are the same standard orientation (45 degrees to the right) and only edges with gradients above the threshold have the brushstroke rotated to the direction of the gradient.

(a) Comparison of brushstroke orientation in images.

(b) Local details.

Figure 7 (a) Two images that compare the effects of orienting the brushstrokes to match the direction of the strongest gradients. (b) Local details in the same area of both images.

IV. RESULTS

A. Optimal Parameter Values

As we discussed in section III, different parameters configured in our algorithm yield largely different resulting images. The effect of each individual parameter has been thoroughly analyzed in section III. After multiple attempts and fine-tunes, we came up with a set of parameters that yield pointillism art with high contrast, high vibrancy, correct color representation and a balance between structure and randomness.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>intensity alpha (α)</td>
<td>100.0</td>
</tr>
<tr>
<td>cluster distance (d)</td>
<td>25.0</td>
</tr>
<tr>
<td>scatter distribution std. (σ)</td>
<td>25.0</td>
</tr>
<tr>
<td>scatter distribution mean (μ)</td>
<td>10.0</td>
</tr>
<tr>
<td>brushstroke radius (r)</td>
<td>6.0</td>
</tr>
<tr>
<td>brushstroke transparency</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 1 One optimal set of parameters.

With this set of parameters, we generate pointillism arts from original images of different kinds. The downsample factor is chosen separately for each input image based on its specific size. Fig. 8 shows one resulting image. For more images, see the Appendix.

Figure 8 Original input image and the resulting pointillism art for the portrait of President Barack Obama.

B. Future Considerations

There are various areas where more work can be done to create aesthetically pleasing art. In the preprocessing step, additional filters can be applied to the downsampled image to produce a final image with even more impressionistic abstraction. One of the easiest methods is to add more blur to the image with another Gaussian filter. The blurring can also be localized, such as blurring the background only to create depth of focus in an image. Another idea is to incorporate morphological image processing with a unique structuring element to dilate or erode features in an image. These are various ideas that can be incorporated before the color selection and image generation step.

Another area that can be further explored is in the methodology for selecting and using colors to enhance features. As mentioned previously, complement colors add contrast to an
image. When the edges are found in the image, complement colors can be strategically placed to outline edges in an image. This may make edges appear sharper because contrasting colors are easier to perceive than brushstroke orientation. The selection of the third color can also be more systematic to ensure more predictable outcomes.

We can also experiment with different brushstroke sizes and shapes. The algorithm can incorporate changing the brushstroke size dependent on gradients and intensities of an image, similar to the method proposed by Chi and Liu’s paper [1]. The runtime of the algorithm can also be improved. Instead of painting each dot individually, each distribution of a specific colored dot can be done in batch method. Overall there are a variety of different techniques that can be incorporated into the structure of this algorithm to produce different aesthetic effects in pointillism art. Our algorithm lays the initial framework for generating pointillism art with flexibility that can take account for user desired features.

ACKNOWLEDGMENT

We acknowledge Professor Gordon Wetzstein for providing initial direction for this project.

WORK DIVISION

We worked together in person on the design and development of the algorithm, paper writing, and poster creation. Yanshu did most of the coding, while Tiffany did most of the paper writing and poster layout. Images were generated by both.

REFERENCES


IMAGE SOURCES


