6.815: Painterly Rendering

Megan Wei

December 9, 2020

1 Motivations

As an art history enthusiast, I was super excited when I came across the non-photorealistic rendering assignment since it reminded me of some of my favorite art eras: impressionism, post-impressionism, expressionism, pointillism, and more. With different textures and brush styles, I'll be able to imitate these styles and become a modern-day Monet.
In this assignment, I implemented brush, single scale paint, painterly, compute angles, single scale oriented paint, and oriented paint.

Chihuly Garden and Glass. Seattle, WA.

2 Background

There are various techniques when it comes to painting. For example, one might start off by creating a rough sketch with a large brush and then using smaller brushes to highlight details. Typically, fine strokes are used to bring attention to fine detail, while large strokes are used to paint areas with less detail. Not only do we vary brush sizes, but we also vary the brush strokes (ex. dots to create a pointillist effect or long, curved brush strokes to create

an expressionist look). Varying brush strokes can bring in a 3D element to the painting, visualizing contours of images better. Also, changing up the color can bring a new look to a painting. In pointillist and colorist wash art styles, there is more randomness in color, whereas for impressionist art, there isn't much deviation from the original color. In this assignment, we will explore painting with different brush sizes, strokes, random noise in color, as well as orienting our brush strokes to understand these painting techniques.

There are many applications and improvements to this algorithm. Applications of non-photorealistic rendering include animations, real-time processing of videos, and concise images for graphic design. There have been improvements in the animation space, where Walt Disney Feature Animation researchers have found a solution that eliminates the shower door"effect and creates coherent images that don't exhibit random frame-to-frame changes. In addition, there is work being done to better represent wet media, such as watercolor and oil paint. There is also research on conveying not just physical but semantic elements, such as emphasis, mood, and emotions, in paintings, as well as conveying important features of an image. Bouncing off of that idea, there are also areas of research that explore splitting a scene into certain regions. While it would be convenient to have a fully automated algorithm that identifies these components and paints them accordingly, incorporating user interaction would also be beneficial as users will be able to define critical regions and play a role in how they want certain regions to be painted without having to do the technical work of actually painting it.

Papers:

- Aaron Hertzmann. 1998. Painterly rendering with curved brush strokes of multiple sizes. In Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH '98). Association for Computing Machinery, New York, NY, USA, 453–460.
- Barbara J. Meier. 1996. Painterly rendering for animation. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96). Association for Computing Machinery, New York, NY, USA, 477–484.
- J. Lansdown and S. Schofield, "Expressive rendering: a review of nonphotorealistic techniques," in IEEE Computer Graphics and Applications, vol. 15, no. 3, pp. 29-37, May 1995, doi: 10.1109/38.376610.

3 Algorithm

Overview

The Non-Photorealistic Rendering assignment consists of these components:

(1) Brush

(2) Painterly Rendering

   (a) Single-Scale

   (b) Two-Scale

(3) Oriented Painterly Rendering

   (a) Compute Angles

   (b) Single-Scale

   (c) Two-Scale

Implementation

Helper function:
`inBounds(out, y, x, texture)`
Description: Checks if point `y, x` is not within half the size of `texture` from the boundaries of `out`.

Main functions:
`brush(out, y, x, color, texture)`
Description: Draws a brush stroke of the opacity `texture` and color `color` centered at `y, x` on `out`, a mutable image. Disregard points that are closer than half the size of `texture` away from the image boundaries. If `y, x` are in bounds, linearly interpolate between color in `out` and `color` based on `texture` at each pixel.

`singleScalePaint(im, out, importance, texture, size=10, N=1000, noise=0.3)`
Description: Single-Scale Paint. Scale `texture` to size `size`. Splat brush with color `(1-noise/2+noise*np.random.rand(3))*im[y,x]` onto `out` for `N` random `y, x` points, which are in bounds and accepted according to `importance`.

`painterly(im, texture, N=10000, size=50, noise=0.3)`
Description: Two-Scale Paint. First layer: coarse strokes of size `size` with constant `importance`. Second layer: fine strokes of size `size/4` with greater `importance` for higher frequencies in the image. Compute a sharpness map on `im` to determine these higher frequencies.

`computeAngles(im)`
Description: Compute tensor from `im`. For each pixel, compute the eigenvalues from the structure tensor matrix and find the eigenvector that corresponds to the minimum eigenvalue. Find the angle between that eigenvector and the horizontal line and set all 3 channels to that angle.

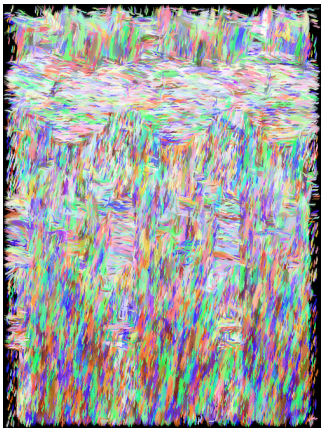
`singleScaleOrientedPaint(im, out, thetas, importance, texture, size, N, noise, nAngles=36)`
Description: Single-Scale Oriented Paint. Scale `texture` to `size`. Splat brush with color `(1-noise/2+noise*np.random.rand(3))*im[y,x]` onto `out` for `N` random `y, x` points, which are in bounds and accepted according to `importance`. Rotate brushes, creating a list of `nAngles` angles. Find theta at point `y, x` from `thetas` and use the `texture` from the list of rotated brushes that matches theta and brush.

`orientedPaint(im, texture, N=7000, size=50, noise=0.3)`
Description: Two-Scale Oriented Paint. Similar to `painterly` with two layers. Compute angles of image and pass those into each `singleScaleOrientedPaint` call.

See code in `/asst/NPR_Starter/npr.py` for more details.

Challenges

One challenge was that I misinterpreted the `size` term as a scale factor and made my `texture` much larger than my output image, `out`. This led to a dark `out` image. It doesn't quite make sense to have a brush size larger than the canvas, unless you're spilling a bucket of paint over a canvas, so I realized that `size` represented the maximum side length for `texture`.
One minor issue was including the alpha (transparency channel) of my image. While this wasn't result I was hoping for, this did produce interesting results though.



4 Results

Statistics

Here, I'll vary the `size`, `N`, `noise`, and `texture` parameters and use Two-Scale Paint as well as Two-Scale Oriented Paint.

This is the original image:

Stray Cat in Petra, Jordan.

Using the default parameters of `N = 10000`, `size = 50`, and `noise = 0.3` and using `brush1`, these were my results for Single-Scale Paint (left) and Two-Scale Paint (right):

Since there are still a lot of spots not covered on the canvas, I can improve this by increasing `N`, the number of splats, and also increasing `size`, the size of each brush stroke. It would look much better if I just increased `N` and maintained the `size`; however, this would take a long time to ensure a large majority of the canvas is covered. Varying `noise` wouldn't improve the problem of not having most spots of the canvas covered, but it would add interesting colors to the mix.

Changing `N` to 40000 and keeping everything else constant, when I call Two-Scale Paint, I get a slightly better result:

Meanwhile, changing `size` to 100 and keeping everything else constant, when I call Two-Scale Paint, this is the resulting image. This filled up a few more spaces in the canvas compared to the default, but there is still room for improvement.

Just for fun, when I set `N = 40000`, `size = 100`, and `noise = 0.7`, I get the following result. This is nice in that it covers a good majority of the canvas, but the noise level is a bit high, which is why we see streaks of other colors, such as red or green, in many areas in the image. This would be good if we're going for a colorist wash style.

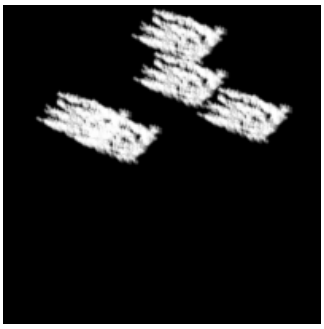
Changing the `texture` to `longBrush` and setting `N = 40000` with everything else set to default, I get the following. There are still some areas of the canvas that are left empty, and there are noticeable horizontal black stripes that really bring this out.

To improve this further, I used Two-Scale Oriented Paint with `brush1` and `N = 50000` with all other parameters set to their default values. This led to a much better result. This could still be improved by increasing `N`.

Here, I scaled my original image down to improve run-time. I used `N = 25000` and `longBrush` with all other parameters default. The cat looks great with this `texture` and the hair seems really defined. Perhaps, the background could use a different `texture`.

Images

Result from brush test with `brush1`:



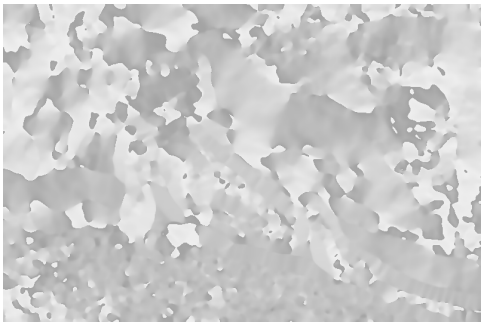
Varying brushes on the same image and default parameters calling Single-Scale Paint: standard brush vs long brush

Long Brush Two-Scale Paint with default parameters:

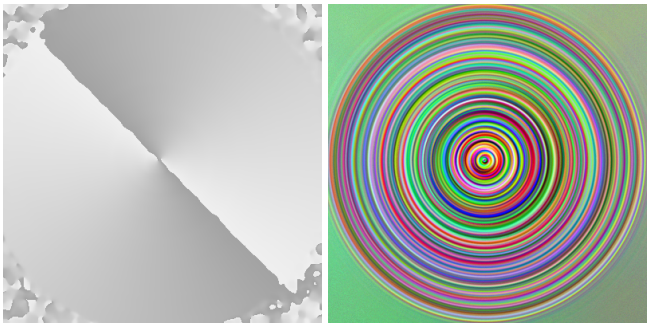
Long Brush Single-Scale Oriented Paint with default parameters:

Long Brush Two-Scale Oriented Paint with default parameters:

Test Angle on Lizard:



Test Angle on round image (left) with round image (right):



Long Brush Two-Scale Oriented Paint on Mount Rainier: