CS 445 Painterly Rendering Report

Motivation:

Painterly rendering is a technique of representing images in the style of a painting. The idea behind this project is to understand rendering realistic pictures into non-realistic pictures. We would like to understand the techniques for painting an image with multiple brush sizes, and for painting with long, curved brush strokes.
This project is based on the paper “Painterly Rendering with Curved Brush Strokes of Multiple Sizes” https://www.mrl.nyu.edu/publications/painterly98/hertzmann-siggraph98.pdf.

Approach:

We have used two different algorithms to render different styles of painting. One is the normal strokes algorithm and the other is curved brush strokes algorithm. Both these styles of rendering take an image as input and a list of brush sizes.

The initial canvas is a black image. Starting with the largest brush size, the canvas is painted using colors from the reference image. Reference image is the gaussian filtered source image. Layers of images are generated from the largest to smallest brush size.

For all the layers, first a difference image between the reference image and the canvas is computed. A region from this difference image is selected such that its area error is greater than threshold value. That selected region is then used to generate strokes. All the strokes are generated and then used to paint the canvas in random order.
The difference in the styles is in the generation and painting of strokes.

Approach used in Normal strokes algorithm:
From the selected region, the color at that particular region is taken and used to paint circles with brush size as radius on the canvas.
def make_stroke(self, canvas, r, x, y, reference_img) in painterly_rendering.py implements this algorithm

Approach used in Curved strokes algorithm:
Stroke direction and points are computed using gradient values. These strokes are used to draw lines with brush size as thickness on the canvas.
def makeSplineStroke(self, canvas, r, x0, y0, refImage) in painterly_rendering.py implements this algorithm

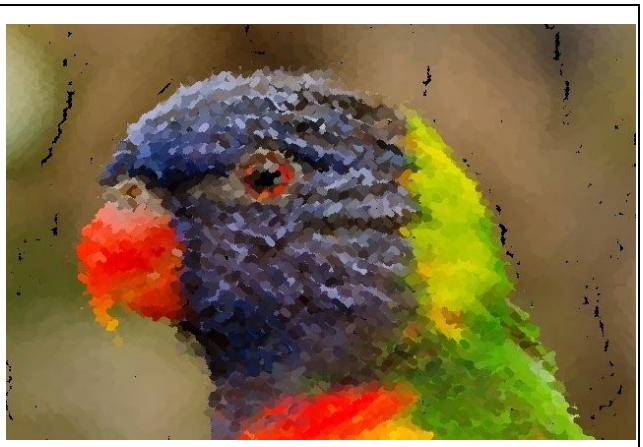
Results

- Test Image 1 - obtained from the source paper

Input Image: 	
Image rendering using normal strokes algorithm: 	Image rendering using curved strokes algorithm: 

- Test Image 2 - Our selection for the report

Input Image: 	
Image rendering using normal strokes algorithm:	Image rendering using curved strokes algorithm:

Implementation Details:

This project is developed using python. Python libraries used are:

- opencv-python
- numpy
- matplotlib

Predefined functions used:

- cv2.GaussianBlur() - to blur an image using a Gaussian filter.
- cv2.Sobel() - to get the unit vector of gradient.
- cv2.line(), cv2.circle() - to produce strokes on the image.

User Defined class and functions used:
We created a class *Painter* that implements the apis Paint() as PaintLayer() as described in the paper. With respect to Stroke selections we have defined two methods, *MakeStroke()* to draw normal strokes and *MakeSplineStroke()* to draw curved strokes as described in the paper.

Challenges:

The major challenge faced in implementation is in the curved strokes algorithm to get the stroke direction. We faced issues in getting the unit vector of gradients. We resolved it by converting the image to grayscale to obtain the correct x and y gradients.

Group:

Bollam Raja Shekar (**bollamr2**) - Normal brush strokes algorithm
Hanisha Nunna (**hnunna2**) - Curved brush strokes algorithm