Instagram_filters

September 24, 2019

1 Introduction

Instagram is a photo-enhancing and sharing app for mobile phones. The name came from "Instant Camera and TeleGram". It allows users to apply different filters to their pictures and share it. It was launched in September 2010 and was an instant hit among its users with 1 Million registered users within 2 months and 10 million in a year. It was acquired by Facebook in 2012 for $1 Billion. **You know what is cool? A billion dollars!**

There are many photographic filters present in the app like Clarendon, Gingham, Lark, Moon, X-Pro II, Nashville, Sutro, Kelvin and many more. Most filters can be designed using the method described in color enhancement using curves section.

In this part of the project, You will design 2 instagram-like filter on your own.

1. Pencil Sketch Filter - This will generate a sketch of the given image as shown in the output below.
2. Cartoon Filter using - This should produce a cartoonified output of the input image.

Given below is the expected output for the two filters. Looking at the output, you should be able to figure out what processing can produce those results.

You can also come up with your own filter and show them in the discussion forum.

|

|

|

|

Original Image

```
| <center>Pencil Sketch Result</center>      | <center>Cartoon Filter Result</center>      |
```

```
In [1]: #include "includeLibraries.h"
        #include <opencv2/opencv.hpp>
        #include "matplotlibcpp.h"
        #include "displayImages.h"
```

```
In [2]: using namespace std;
```

```
In [3]: using namespace cv;
In [4]: using namespace matplotlibcpp;

In [5]: Mat cartoonify(Mat image, int arguments = 0)
        {

            Mat cartoonImage;

            cv::Mat grayImage;
            cv::cvtColor(image, grayImage, cv::COLOR_BGR2GRAY);

            //apply gaussian blur
            cv::GaussianBlur(grayImage, grayImage, Size(3, 3), 0);

            //find edges using Laplaican
            cv::Mat edgeImage;
            cv::Laplacian(grayImage, edgeImage, -1, 5);
            cv::convertScaleAbs(edgeImage, edgeImage);

            //invert the image
            edgeImage = 255 - edgeImage;

            //apply thresholding
            cv::threshold(edgeImage, edgeImage, 150, 255, THRESH_BINARY);

            //blur images heavily using edgePreservingFilter
            cv::Mat edgePreservingImage;
            cv::edgePreservingFilter(image, edgePreservingImage, 2, 50, 0.4);

            cartoonImage = cv::Scalar::all(0);

            // Combine the cartoon and edges
            cv::bitwise_and(edgePreservingImage, edgePreservingImage, cartoonImage, edgeImage)

            return cartoonImage;
        }

In [6]: Mat pencilSketch(cv::Mat image, int arguments = 0)
        {

            Mat pencilSketchImage;

            //Convert color image to gray image since background is white and sketch is black
            cv::cvtColor(image, pencilSketchImage, cv::COLOR_BGR2GRAY);

            //Apply gaussian blur
            cv::GaussianBlur(pencilSketchImage, pencilSketchImage, cv::Size(3, 3), 0);

             //Detect edges using Laplacian
            cv::Laplacian(pencilSketchImage, pencilSketchImage, -1, 5);

            //Invert the binary image
            pencilSketchImage = 255 - pencilSketchImage;

            //binary thresholding
            cv::threshold(pencilSketchImage, pencilSketchImage, 150, 255, cv::THRESH_BINARY);

            return pencilSketchImage;
        }
```

```
In [7]: string imagePath = DATA_PATH + "images/trump.jpg";
        Mat image = imread(imagePath);

        Mat cartoonImage = cartoonify(image);
        Mat pencilSketchImage = pencilSketch(image);
```

```
In [8]: plt::figure_size(1200,500);
        plt::subplot(1,3,1);
        plt::imshow(image);
        auto pltImg = displayImage(image);
        plt::subplot(1,3,2);
        plt::imshow(cartoonImage);
        pltImg = displayImage(cartoonImage);
        plt::subplot(1,3,3);
        plt::imshow(pencilSketchImage);
        pltImg = displayImage(pencilSketchImage);
        pltImg
```

Out[8]: