CS 4475 Project 2 Report

Sanghyun Chun

Shc1991@gatech.edu

902759995 (schun41)

Introduction

This time, I decided to apply Gaussian Pyramid into my work. I researched on how to do a cartoon filter, and came across http://www.learnopencv.com/non-photorealistic-rendering-using-opencv-python-c/.

It seemed interesting and applicable to what we learned in class, so I decided to do my own version of a cartoon filter.

Workflow

Basic Overview

How to run the code (you can also refer to README.txt):

1. Download any image and name it "source.jpg" in the same folder as the Python code. Do not worry About the size; it will be downscaled automatically if needed.

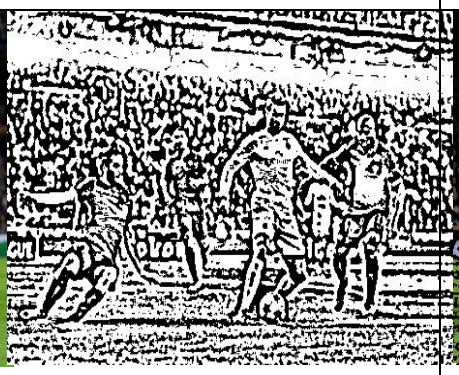
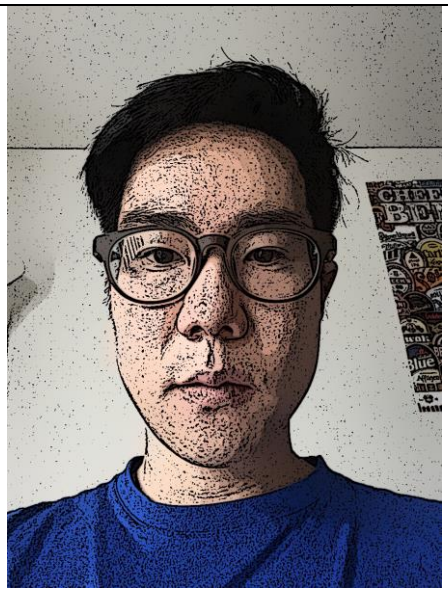
2. Run the code (if using PyCharm like myself, Alt+Shift+X)

3. The code has succeeded if the final output image pops up. Also, check the folder to see if "mask.jpg" and "cartoon.jpg" are created. The image output may take a while, depending on the source image size.

In the same folder as your input image, you should see 2 other images created:

1. mask.jpg
2. cartoon.jpg

Each image represents the mask image and the final output image. Below is an example of what you should get:

Original	Mask	Cartoon
		
		
		

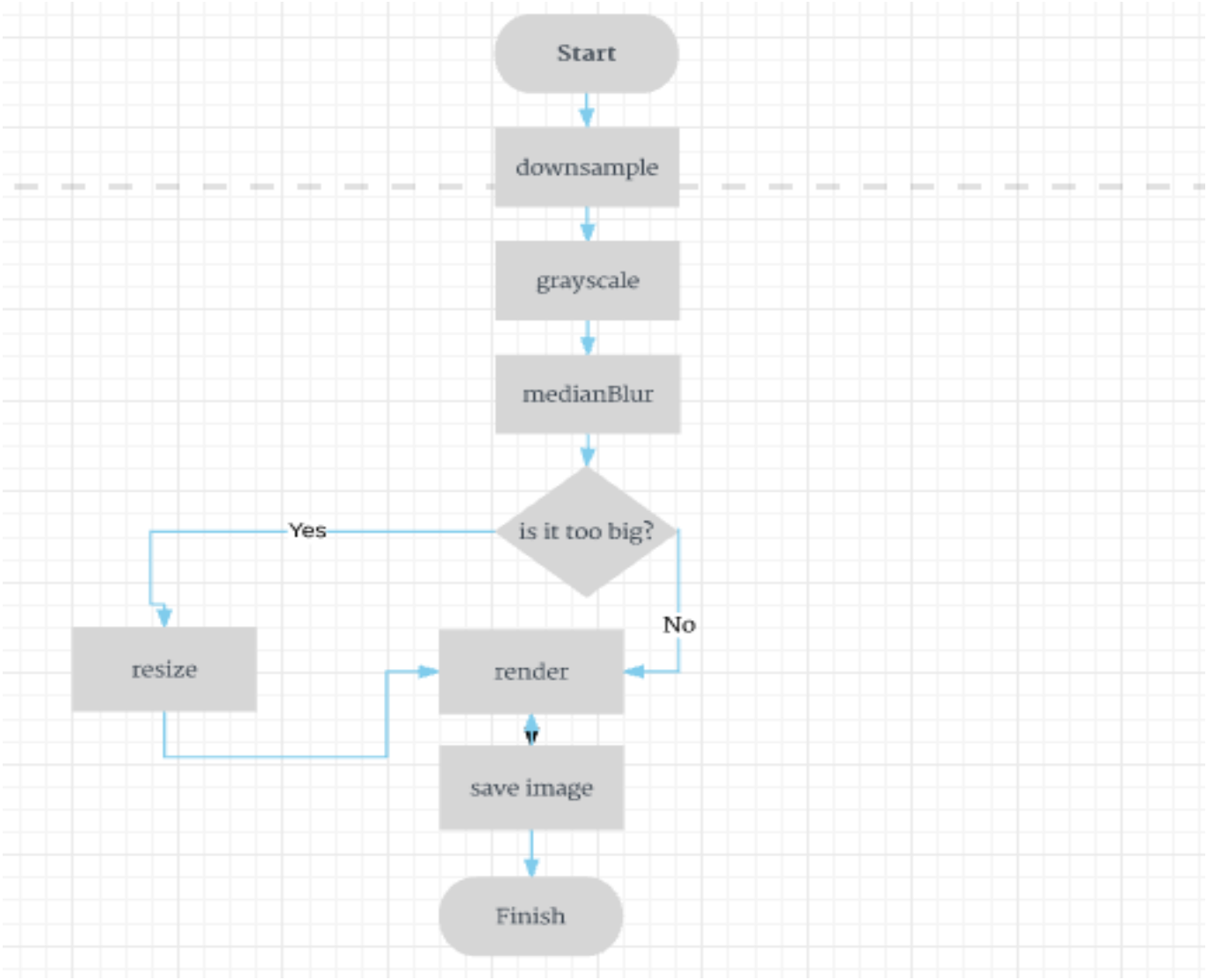
(Note: the images above were shrunk to fit the table. Refer to the images in the folder to see in detail.)

In-Depth Analysis of Algorithms

There are in total 4 main parts to the codes:

1. downSample
    a. This part is needed to apply bilateral filter to the image, using Gaussian Pyramid.
    b. First, downscale the image based on the parameter given (for this assignment, I found that downscaling 1 step was enough). Used cv2.pyrDown)
    c. Apply cv2.bilateralFilter on the image as many times as the given parameter (in this case, 50 steps)
    d. Upscale the image to its original size (cv2.pyrUp)
2. grayscale
    a. Turn the image gray to ease the rendering process. Used cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
3. medianBlur
    a. Apply median blur on the image to get the smoothness effect on the final output
    b. Used cv2.medianBlur
4. Render
    a. This part incorporates the functions above to create the mask, then apply it to the image and return the final output.
    b. First, determine if the image is too big to fit on the screen. If either the number of pixels in the rows or columns exceed 700, shrink the image to half its size, then proceed.
    c. Apply downscale, grayscale, and medianBlur to the image
    d. Create mask. The OpenCV online documentation page has a section where it describes adaptive thresholding. Cv2.adaptiveThreshold allows us to get the edge details on images that have different lightings. I used 11 and 2 as its parameters.
    e. Once mask is created, save the mask image and return the bitwise AND product of the mask and image.
5. Image Output
    a. Save the image generated by render() as “cartoon.jpg”, and cv2.imshow to display it.

Here is the flowchart of the algorithm:



Credits

For this assignment, I used 3 stock images from Google. Here are the links:

http://www.travelandleisure.com/sites/default/files/styles/1600x1000/public/1454344088/Paris-France-Stowe-Writer-Guide-QUOTES0116.jpg?itok=qQQ2dyuw

http://specials-images.forbesimg.com/imageserve/573c6b334bbe6f63618536ea/0x0.jpg?fit=scale&background=000000

https://pmcvariety.files.wordpress.com/2016/04/captain-america-civil-war.jpg?w=670&h=377&crop=1

These were used for non-profit purposes.

These are the resources that I used for this assignment:

http://www.learnopencv.com/non-photorealistic-rendering-using-opencv-python-c

http://www.shellandslate.com/fastmedian.html

http://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html#gsc.tab=0

Artistic / Technical Emphasis

I believe my assignment demonstrates adequate technical emphasis. I incorporated many of the theories learned in class. Not only basic manipulations such as grayscaling image and blurring present, the assignment also uses downsampling an image using Gaussian Pyramid and adaptive thresholding, which are more advanced. I believe the project has 25% technicality.

On the other hand, the assignment produces an output that is artistic enough to justify the allocation of the other 25%. The resulting image is significantly stylized from the original image, and gives a new aesthetic life to its predecessor. I found that the cartoon rendering works best on portraits and images where a human face is present. This may be because the face contains many ripples and lines that the algorithm perceives as edges. I allocate 25% to the artistic merit for this assignment.