Alexandria University
Faculty of Engineering
Computer and Systems Department

CSE: Computer Vision
Assigned: Sat, Feb 22, 2025
Due: Tue, Mar 11, 2025

Assignment 1

1 Part I: Applying Image Processing Filters For Image Cartoonifying

In this part of the assignment we want to make the real-world images look like they are genuinely from a cartoon. The basic idea is to fill the flat parts with some color and then draw thick lines on the strong edges. In other words, the flat areas should become much more flat and the edges should become much more distinct. We will detect edges and smooth the flat areas, then draw enhanced edges back on top to produce a cartoon or comic book effect.

Figure 1: Final output for image cartoonifier program

(a) Original image

(b) Image after cartoonifying

1.1 Generating a black-and-white sketch

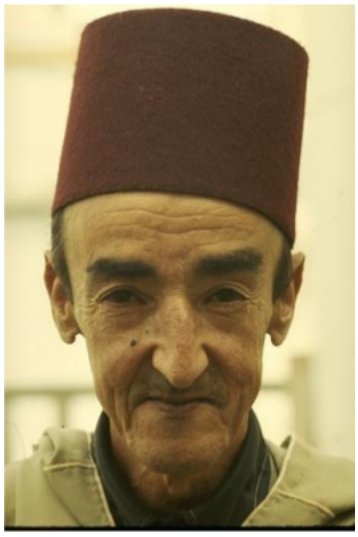
To obtain a sketch (black-and-white drawing) of the image, we will use an edge-detection filter, whereas to obtain a color painting, we will use an edge-preserving filter (bilateral filter) to further smooth the flat regions while keeping the edges intact. By overlaying the sketch drawing on top of the color painting, we obtain a cartoon effect as shown earlier in the screenshot of the final program.

There are many different edge detection filters, such as Sobel, Scharr, Laplacian filters, or Canny-edge detector. We will use a Laplacian edge filter since it produces edges that look most similar to hand sketches compared to Sobel or Scharr.

Good Luck!

Dr. Marwan Torki
Eng. Shereen Elkordi
Eng. Mohamed Badran
Eng. Heba Elwazzan

Figure 2: Converting RGB Image to Grayscale



(a) Original RGB image

(b) Grayscale Image

1.1.1 Noise Reduction Using Median Filter

We need to reduce the noise in the image before we use a Laplacian edge filter. We will use a Median filter because it is good at removing noise while keeping edges sharp. Since Laplacian filters use grayscale images, we must convert from OpenCV's default BGR format to Grayscale. Then for noise reduction, we apply a Median filter.

1.1.2 Edge Detection Using Laplacian Filter

After noise reduction, a Laplacian filter is used for edge detection. The Laplacian filter produces edges with varying brightness, so to make the edges look more like a sketch we apply a binary threshold to make the edges either white or black.

1.2 Generating a color painting and a cartoon

A strong bilateral filter smoothes flat regions while keeping edges sharp, and is therefore great as an automatic cartoonifier or painting filter, except that it is extremely slow (that is, measured in seconds or even minutes rather than milliseconds!). We will therefore use some tricks to obtain a nice cartoonifier that still runs at an acceptable speed. The most important trick we can use is to perform bilateral filtering at a lower resolution. It will have a similar effect as at full resolution.

Figure 3: Noise reduction using Median filter

(a) Original Grayscale Image

(b) Smoothed Grayscale Image

Figure 4: Edge detection using Laplacian Filter

(a) Smoothed Grayscale Image

(b) Edge Detection

Figure 5: Edges Thresholding

(a) Output from Laplacian Filter

(b) Output after image thresholding

Rather than applying a large bilateral filter, we will apply many small bilateral filters to produce a strong cartoon effect in less time. We will truncate the filter so that instead of performing a whole filter (for example, a filter size of 21 x 21), it just uses the minimum filter size needed for a convincing result (for example, with a filter size of just 9 x 9).

We have four parameters that control the bilateral filter: color strength, positional strength, size, and repetition count.

Figure 6: Applying Bilateral Filter

(a) Original Image

(b) Output from bilateral filter

Then we can overlay the edge mask that we found earlier. To overlay the edge mask "sketch" onto the bilateral filter "painting", we can start with a black background and copy the "painting" pixels that aren't edges in the "sketch" mask.

Figure 7: Creating Cartoon Effect

(a) Output from bilateral filter

(b) Final Output