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

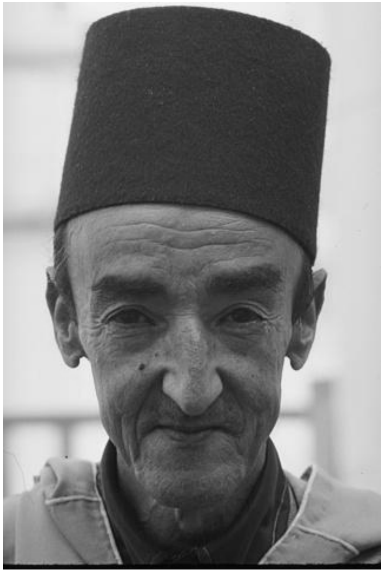
1.1 Generating a black-and-white sketch

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

Figure 2: Converting RGB Image to Grayscale

(a) Original RGB Image



(b) Grayscale Image

similar to hand sketches compared to Sobel or Scharr.

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

Figure 3: Noise reduction using Median filter

(a) Original Graysclae Image (b) Smoothed Grayscale Image

Figure 4: Edge detection using Laplacian Filter

(a) Smoothed Graysclae Image (b) Edge Detection

Figure 5: Edges Thresholding

(a) Output from Laplacian Filter (b) Output after image thresholding

can use is to perform bilateral filtering at a lower resolution. It will have a similar effect as at full resolution.

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

2 Coins Detection Using Hough Transform

The objective of this part of the assignmentt is the detection of circular coins in an image using Hough Transform.

Figure 8: Final output after coin detection

(a) Original image

(b) Image after coin detection

2.1 Circle Hough Transform (CHT)

The Hough transform can be used to determine the parameters of a circle when a number of points that fall on the perimeter are known. A circle with radius R and center (a, b) can be described with the following equations: $(xa)^2 + (xb)^2 = R^2$. Where the locus of (a, b) points in the parameter space fall on a circle of radius R centered at (x, y). The true center point will be common to all parameter circles, and can be found with a Hough accumulation array.

2.2 Dataset

Your program shall be tested on several images for Egyptian coins. The data set has been pre-processed and most of the noise has been removed. You could assume the following about the data set:

- Images include the following coins only: one Egyptian pound (135 pixels radius), 50 piastres (120 pixels radius) and 25 piastres (107 pixels radius).
- There are at most 20 coins per image.
- Each image has at least one coin of each type.
- For simplicity, all the coins of the same type have the same radius.

2.3 Implementation Details

2.3.1 Smoothing the image

Smoothing the image is accomplished using a 2-dimensional median smoothing filter.

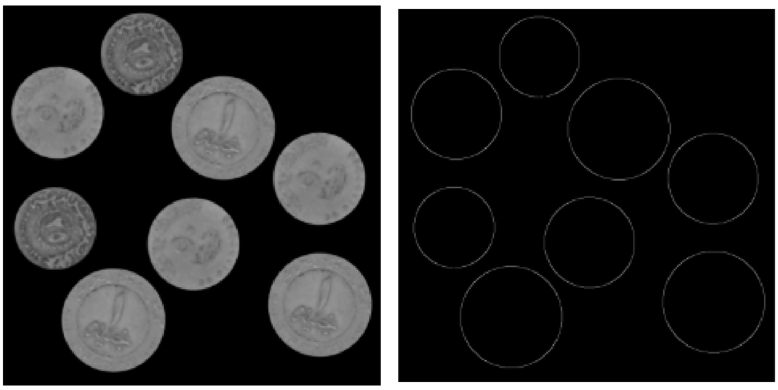
Figure 9: Smoothing the image with median filter

(a) Gray Scale Image (b) Smoothed Image

2.3.2 Edge Detection

After smoothing the image, Cannys algorithm has to be used for edge detection. We will use relatively high values for thresholding to remove most of the noise.

Figure 10: Edge Detection Using Cannys Algorithm



(a) Smoothed Image (b) Detected Edges

2.3.3 Accumulation into (a, b)-space using circular Hough transform

```
procedure CHT(Edges, R)
    for each edge point ∈ Edges do
        for each possible value of b do
            Substitute in (1) and solve for a
            Increment H(a, b)
        end for
    end for
end procedure
```

2.3.4 Refining Coordinates and CHT Post-Processing

During the whole process of finding the centers, some inaccuracies could occur. This could be due to choosing a large bin size for CHT or due to noise in the detected edges. Therefore, after finding the coordinates, a search in the (a, b)-space is executed. We look for the highest peak of the accumulator function and perform non-maximum suppression for lower values.

3 Notes

You are required to deliver the following:

- Your code.
- Output for some test images.
- Report including explanation of your code and representative results on sample test images.

You can work in groups of 2 or 3