Image processing

Assignment 2

Pouria Malek Khayat

Exercise 1

Introduction

In this part, we are given a photo of a camera. It is required to convert the photo to a cartoonish one. This is the original photo.

How to convert the photo

The general idea is to extract the edges, flatten it and then gather the edges with black.

We follow these steps to achieve this goal.

- Applying cvtColor function

  The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on.

  This is required since we use the Canny function which needs the gray image.

- Applying the Canny function

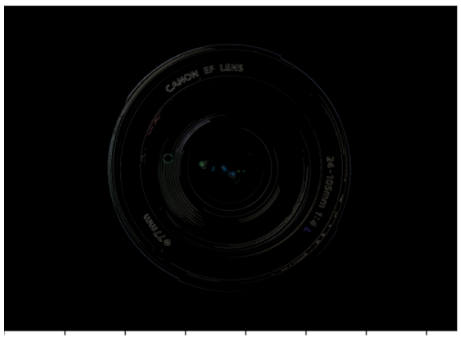
  The function finds edges in the input image and marks them in the output map edges using the Canny algorithm. The smallest value between threshold1 and threshold2 is used for edge linking. The largest value is used to find initial segments of strong edges.

- Applying the bilateralFilter function

The function applies bilateral filtering to the input image. bilateralFilter can reduce unwanted noise very well while keeping edges fairly sharp. However, it is very slow compared to most filters.

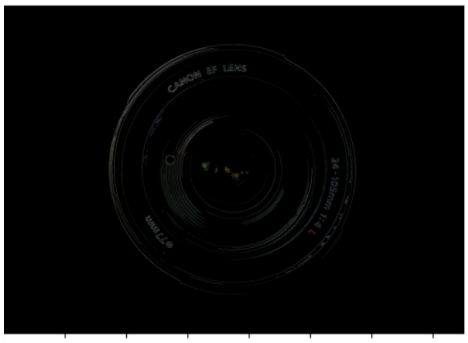
- Applying bitwise_and function

  computes bitwise conjunction of the two arrays (dst = src1 & src2) Calculates the per-element bit-wise conjunction of two arrays or an array and a scalar.



- Applying cvtColor function

  This function was explained before. We apply this in order to get the RGB format.



Exercise 2

In this part, we are required to apply gaussian noise on each photo and then apply a gaussian filter with different intensities.

It is required to calculate the MSE and find the best gaussian filter for each photo.

We take the following steps to achieve this:

First, we initiate noise_vector as a random vector. Then we add it to the image to make it noisy.

Then, we apply the Clip function to limit the values in the image. After this, we have to follow these steps for each image - filter size - noise:

- Apply medianBlur function

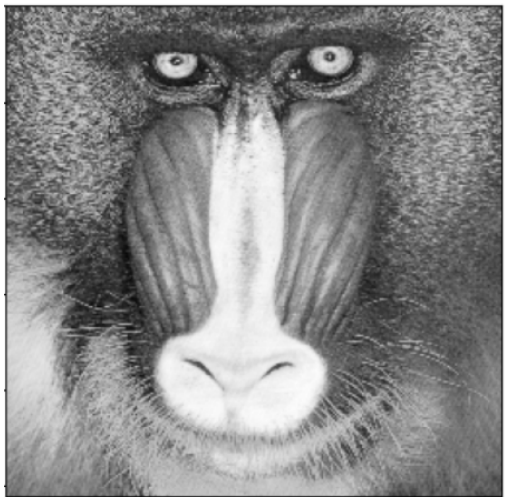
  The function smoothes an image using the median filter. Each channel of a multi-channel image is processed independently.

- Calculating the MSE. it is then added to a list.
- Plotting the images.

This is the result for best filters:

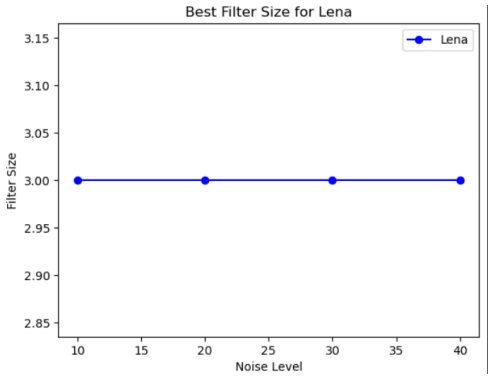
```
lena - Noise: 10, Best Filter Size: 3, MSE: 142.54
lena - Noise: 20, Best Filter Size: 3, MSE: 437.20
lena - Noise: 30, Best Filter Size: 3, MSE: 931.86
lena - Noise: 40, Best Filter Size: 3, MSE: 1625.85
caman - Noise: 10, Best Filter Size: 3, MSE: 209.17
caman - Noise: 20, Best Filter Size: 3, MSE: 501.03
caman - Noise: 30, Best Filter Size: 3, MSE: 991.02
caman - Noise: 40, Best Filter Size: 3, MSE: 1677.44
baboon - Noise: 10, Best Filter Size: 3, MSE: 552.14
baboon - Noise: 20, Best Filter Size: 3, MSE: 827.66
baboon - Noise: 30, Best Filter Size: 3, MSE: 1275.48
baboon - Noise: 40, Best Filter Size: 3, MSE: 1891.02
```

These are the original images:

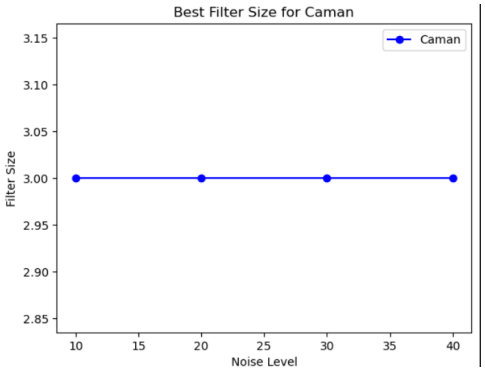


It looks like the best filter is 3 for all of the images, regardless of the images. Although we can not make assumptions just by 3 images. Here is the plot for each photo with noise and filter size as axis:

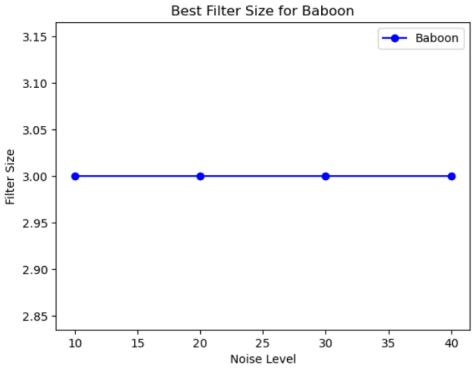
- Lena:



- Camera man:



- Baboon:



As it appears, all best filters are 3 for each photo and noise.

In B part, we calculate PSNR using the calculated MSE. the values can be seen in the jupyter notebook. PSNR is calculated using this formula:

$$PSNR = 10\log_{10}\left(L^2/MSE\right)$$

Conclusion

It looks like the best gaussian filter for each photo regardless of the noise is 3. But we can not be sure and it needs to be tested on other images as well.