CS 484 - CS 555: Introduction to Computer Vision
Spring 2020

Homework 3: Image Classification

Instructor: Dr. Sedat Ozer
TA: Mr. Aydamir Mirzayev

Due: April 15, 2020.

Instructions

- Please first read this entire document from beginning till the end.
- You will **submit a single Python script, or Jupyter Notbeook** including all the methods for each step below (a single <YourName>_<YourID>_PA3.m file where you will replace <YourName> with your own particular full name and <YourID> with your current student ID). For each step, you will write a different function within the same Python script.
- In addition to that, you will also **submit a single report** in PDF format and save it as: <YourName>_<YourID>_PA3_Report.pdf. Your report will include all the figures and/or the output values of your individual Python functions as asked in each part of this assignment below.
- If you use the same code section/snippet in multiple sections, then copy and paste that code in each method individually so that that method can run by itself when copied into another file.
- Each method should be able to run by itself and be able to produce the asked results. Error producing codes may not receive credit.
- For this assignment, you should compress all your files: .py/.ipynb file and your report file (.pdf) into a single <YourName>_<YourID>_PA3.zip file. Email your zip file to your TA (Mr. Mirzayev) with the subject: "CS484_CS555_HW3_Submission".
- Please pay attention to the format before submitting your report. Not only your results, but the quality and aesthetics of your report will also affect your final grade.
- Contact the TA: *aydamir.mirzayev@bilkent.edu.tr*, if you have any question.
- Not complying with the submission rules will result in 10 points penalty.

Loading Images

Note that in this assignment, you will be performing operations on single channel (gray-scale) images, however some of the images sent to might be multi channel, so you will first need to convert them to single-channel gray-scale images. Please use the code snippet below.

```
import matplotlib.image as mpimg
from skimage.color import rgb2gray
image = rgb2gray( mpimg.imread( 'image.png'))
```

1 Part I: Edge Detection

In this question, we will study various spatial filtering operations.

In this part of the assignment, you will perform edge detection on the provided three images: *edge*1.*png*, *edge*2.*png*, *edge*3.*png* (see Fig. 1). You will use different edge detection methods including Sobel, Prewitt and Canny edge detector. You will use gray-scale, single channel version of the provided images.

(a)

(b)

(c)

Figure 1: Input images to be used for edge detection in Part I.

1.1 Edge detection with Sobel and Prewitt operators

On each of the given three images, you will detect edges by using Sobel and Prewitt operators separately.

1. 3x3 Sobel Operators: For each given image above, detect horizontal and vertical edges by using Sobel operators and plot them. Include the results in your report. You will perform both horizontal and vertical edge detection on each image separately by using the corresponding 3x3 Sobel operator.

2. 3x3 Prewitt Operators: For each given image above, detect horizontal and vertical edges by using Prewitt operators and plot them. Include the results in your report. You will perform both horizontal and vertical edge detection on each image separately by using the corresponding 3x3 Prewitt operator.

3. Comment on your results: Show your results in your report and comment on them: are all the edges successfully detected? Are there any difference between using Sobel and Prewitt filters? If so, report which one yielded better result.

1.2 Edge detection with Canny edge detector

Another method that you will use for edge detection is Canny edge detector.

For this part, you have to find optimal parameters of Canny edge detector including the Gaussian smoothing parameter σ and the hysteresis thresholding parameters: T_low and T_high (in total three parameters) for each image.

1. Report the best values for each of those three parameters that worked best for you on each image. Comment on why you think that those parameters are the best parameters.

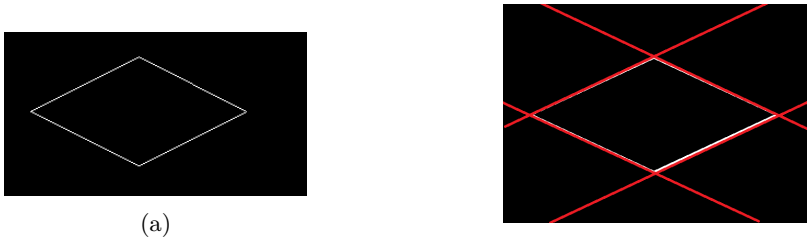
2. In your report, for comparison, include at least four different sets of parameters (i.e., you need to show different σ, T_low and T_high values and their corresponding result for four times).

3. How do your results compare to the results obtained from Sobel and Prewitt operators? If you see an improvement, comment on the potential reasons for that.

2 Part II: Edge Linking with Hough Transform

For this part of the assignment, you will implement Hough transform and visualize the quality of your Hough implementation by drawing the detected lines on the input grayscale image. Hough transform operates on the detected edges to fit line(s) on the edge pixels. Please refer to the slides on Edge Detection. You should also check other online resources such as: PDF tutorial, and PDF tutorial for various presentations of Hough transform. In this section, you have three parts to work on. See below:

1. Implement Hough transform to predict the line parameters: ρ and θ for the lines that fit to the edges as discussed in the class. Use your implementation to find the lines shown on *hough.png*. To check your results, the expected output is shown in Fig. 2a. See Fig. 3a for the expected detected lines (shown in red).



(a) (b)

Figure 2: Detected lines with Hough transform are shown on a basic sample image. Note: due to the nature of this particular sample image, the (grayscale) input image and the detected edges on that input image are considered the same.

2. You will use the images: *edge*1.*png* and *hough*2.*png* for the rest of this question (Image *hough*2.*png* is shown in Fig. 3). Apply Hough transform to find the lines in the images: *edge*1.*png* and *hough*2.*png* and highlight the found lines on each of those images. Remember: Typically, Hough Transform is used after applying an edge detection operator on the input image. First apply the edge detection algorithm of your choosing (from the ones used in the previous part) and then apply Hough Transform to find the lines on the detected edges. Plot both Hough Transforms (in ρ and θ space) and detected lines in the image space for each image. Did you need to use thresholding in Hough Space? If so, what threshold value worked best for you to find meaningful lines on each image?

3. For the image: *edge.png*, use Hough Transform and particularly find the line of the horizon that is clearly visible in the image. Report your results. Comment on the performance of your implementation, how well you were able to find the horizon line.

(a)

Figure 3: Image: *hough*2.*png* is shown.