Programming Assignment-I

Robot Vision

Coding Standard and General Requirements

Code for all programming assignments should be **well documented**. A working program with no comments will receive **only partial credit**. Documentation entails writing a description of each function/method, class/structure, as well as comments throughout the code to explain the program flow. Programming language for the assignment is **Python**. You can use standard python built-in IDLE, or other IDLEs such as CANOPY, PyCharm Community Edition, PyScripter, CodeSculptor, Eric Python, Eclipse plus PyDev, etc.

Following libraries can be used when necessary:

- PIL (The Python Imaging Library), Matplotlib, NumPy, SciPy, LibSVM, OpenCV, VLFeat, python-graph.

If you are asked to implement "Gaussian Filtering", you are not allowed to use a Gaussian function from a known library, you need to implement it from scratch.

Submit by **26th Feb 2021**, 11.59pm. PA1 is 10% of your total grade.

Question 1: Box Filtering [1 pt]

Implement two box filters (one with 3 by 3, the other one is 5 by 5 kernel size), and apply them to the given images (i.e., image1.png and image2.png) separately. Show the resulting images, and explain the resulting images, and their differences. (For convolution operation, you can use built-in function. Do not use built-in function for box filtering.)

Question 2: Median Filtering [1 pt]

Implement three median filters (3 by 3, 5 by 5, and 7 by 7 kernel size), and apply these filters to image1.png and image2.png separately. Show (and discuss as comments) the resulting differences for each kernel on the screen, explain where median filters are most effective. (For convolution operation, you can use built-in function. Do not use built-in function for Median filtering.)

Question 3: Gaussian Filtering [1 pt]

Implement a two-dimensional Gaussian kernel with a variation (sigma) equal to 3, 5, and 10. Apply these three Gaussian kernels to image1.png and image2.png separately, show them on the screen, discuss the differences of Gaussian operations with different sigmas (as comments on the code). Also, compare your results with question 1 and question 2: what are the differences between these three filters, what do you observe (as comments on the code)? Which filtering is the most effective in which images ? Why ? (For convolution operation, you can use built-in function. Do not use built-in function for Gaussian filtering.)

Question 4: Gradient Operations [1 pt]

Write a derivative operations (forward, backward, and central difference) that can be applied to any given image (f) and produces two images: gradient x, and gradient y images (i.e., f_x, f_y). Also, calculate gradient magnitude as $\sqrt{(f_x^2 + f_y^2)}$. The image that you should use for this problem is called image3.png. Show the results on the screen (Do not use built-in gradient functions, create difference operator yourself as stated clearly in the question.)

Question 5: Sobel Filtering [1 pt]

Implement Sobel filtering with 3 by 3 kernel size. Apply it to image1.png and image2.png. Show the results on the screen, and discuss the resulting images (as comments on the code). (For convolution operation, you can use built-in function. Do not use built-in function for Sobel filtering.)

Question 6: Fast Gaussian Filtering [1 pt]

Implement question 3, but this time, use only 1D Gaussian to do filtering in 2D. The trick is the following: since Gaussians are separable, you can use 1D Gaussian to filter the image in one direction first, and then the other direction can be filtered. Show the results on the screen and compare the efficiency of both methods (question 3 and question 6) as comments on the code. Use image1.png and image2.png for smoothing and show results too. (For convolution operation, you can use built-in function. Do not use built-in function for fast Gaussian filtering.)

Question 7: Histogram [2 pt]

Implement histogram function from scratch, and show the resulting bar-graph (histogram). Use 256, 128, and 64 bins to visualize histograms. Comment on the resulting differences with respect to bins. Use image4.png to conduct this experiment. (Do not use built-in histogram function, create it yourself.)

Question 8: Canny Edge Detection [7 pt]

Your tasks:

0 pt Use two different images (canny1.jpg and canny2.jpg) to perform the following steps for getting Canny edges of the input image.

1 pts Use 1-dimensional Gaussians to implement 2D Gaussian filtering yourself (do not use built-in functions).

1 pts Obtain gradient images (x-dim, y-dim, gradient magnitude, and gradient orientation) by following the Canny algorithm that we have seen in the class. Show resulting gradient images on screen and in the report.

2 pts Implement non-max suppression algorithm to reduce some of the falsely detected edges in the gradient images (from the previous step). Show the improved edge map on the screen and in the report.

1 pts Implement hysteresis thresholding algorithm and use it to further enhance the edge map obtained from the previous step. Show the final Canny edge map on the screen and in the report.

1 pt Show the effect of σ in edge detection by choosing three different σ values when smoothing. Note that you need to indicate which σ works best as a comment in your assignment.

1 pt Discuss about the different filtering approaches you took for four pictures. Since pictures are the same scene but different noise and smoothing patterns, you need to adjust your Canny edge filtering parameters to show similar results to Canny edges of the output-canny1.png and output-canny2.png.

Question 9: Image segmentation [5 pts]

In this question you goal is to implement Otsu thresholding to perform image segmentation. The algorithm was discussed during a class lecture.

Your tasks:

- First implement a simple thresholding based image binarization algorithm. Plot the histogram for three different input image. Now based on the plot, perform binarization at three different threshold levels.
- Implement a Otsu thresholding. Use the determined threshold to perform segmentation on the three input image.

NOTE: You are free to choose any 3 images. If the images are colored, you can convert them to greyscale by averaging the RGB values at each pixel. You can also use any library function to convert it to greyscale.

What to submit:

- Code
- A short write-up about your implementation with results (as indicated for each variation) and your observations from each results.