HW2 - Blurring and Edge Detection
Due 10/17/2024

1 Introduction

This Homework assignment is designed so that you can get an understanding why blurring an image first can help with getting better results. There will be two parts to this assignment.

You CAN use the built in cv2.GaussianBlur() for blurring your image throughout this assignment. You do not need to implement your own blur filter.

2 Part 1 - Visualizing Blurring in 3D: 20 points

- Download the dog.jpeg from canvas and convert it to a grey scale image.
- Plot the original dog photo in 3D like we did in class.
- Run a 5x5 Gaussian Blur on the dog photo and plot the blurred image in 3D.
- Run a 11x11 Gaussian Blur on the dog photo and plot the blurred image in 3D.
- Display Your 3 (original, 5x5, 11x11) graphs at the same time to make it easier for the TAs to grade.
- Answer the following question by adding it to the end of your code as a comment: What do you notice about the 3D graphs as the filter size increases?

3 Part 2 - Edge Detection: 60 points

In this part you must implement your own Sobel edge detection. There are two Sobel Edge detection filters. One for detecting if there are edges horizontally and another for detecting edges vertically. You must implement both of them and combine the result of them to get one final output. You will than compare your results back to a more popular edge detection algorithm called, Canny Edge Detection.

Canny uses the same steps as Sobel but applies a Hysteresis Thresholding after determining the derivative to help decide which edge should be kept.

Hysteresis Thresholding uses a min and max value to determine which edges to keep.

More information can be found here:

https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

Note: In class I showed that the Sobel filter was multiplied by 1/8 after summation. In my implementation I did not find it useful to multiply by 1/8. You might get different result if you do.

Sobel x filter

1	0	-1
2	0	-2
1	0	-1

Sobel y filter

1	2	1
0	0	0
-1	-2	-1

Steps to complete:

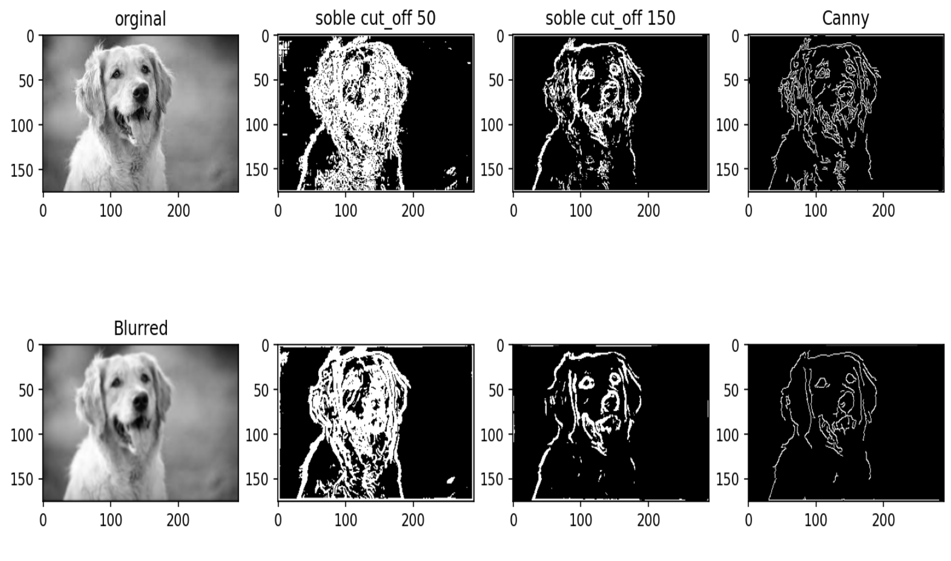
- Download the dog.jpeg from canvas and convert that to a grey scale image.
- Run the dog photo through both Sobel filters (x and y) and combine the results using a **smaller** cut off value (I used a cut off of 50).
- Run the dog photo through both Sobel filters (x and y) and combine the results using a **larger** cut off value (I used a cut off of 150).
- Run the dog photo using the built in cv2.Canny(imgage, 100, 100). (I used a min and max both of 100).
- Run a 5x5 Gaussian Blur on the dog photo and run the sobel filter and canny filter on the blurred image.
- Use a 2x4 grid to show all your results.
- Answer the following questions at the end of your code by leaving a comment:

  - What did you notice when you went from a lower threshold value to a higher one?
  - What did you notice before and after applying a Gaussian Blur to the image?

4 Grading (Out of 100 points)

- 10 Points : Add your name and date to the beginning of your code.
- 10 Points : Use appropriate comments throughout your code.
- 20 Points : Your output in part 1 shows all three graphs showing the differences between applying larger bluring filters to the image.
- 20 Points : Sobel Edge detection is working properly.
- 20 Points : Canny Edge detection is working properly.
- 10 Points : Results are shown in a grid with matplotlib with titles for each sub graph (see example output).
- 10 Points: You answered the questions from both part 1 and part 2.

Example output



5 How to turn in

You must submit a zip folder that includes all files, folders, and images that are required to run your program. Name the zip folder "HW2-Lastname.zip" and upload it to canvas.

Good Luck Cat!