Computer Vision *ECS 174, Spring 2022*

HW1: Filters and edge detection

Please note that only PDF submissions are accepted. We encourage using LaTeX to produce your writeups. You'll need *mydefs.sty* and *notes.sty* which can be downloaded from the course page.

Please turn in the written assignment added to this latex file (in PDF format) along with the complete code and its results.

Section 1: Coding Assignment

Before you start, please download the project's files from the course webpage.

For this assignment, you'll be implementing functions for filtering and edge detection. Please note that Matlab has built-in functions for most things that you need to do here, so for educational purposes, you should not use them here. Particularly, you should not use filtering commands and so on in your final code. However, you should feel free to use them to verify your results. You may use Python or Matlab.

1. (10 points) Implement the function GaussianBlurImage(image, sigma) to Gaussian blur an image. "sigma" is the standard deviation of the Gaussian. Implement the Gaussian blur using a 2D filter. You may use Matlab's "normpdf".

   Required: Gaussian blur the image "Seattle.jpg" with a sigma of 4.0, and save as "1.png".

2. (10 points) Implement the function SeparableGaussianBlurImage (image, sigma) to Gaussian blur an image using separable filters. "sigma" is the standard deviation of the Gaussian. The separable filter should first Gaussian blur the image horizontally, followed by blurring the image vertically. The final image should look the same as when blurring the image with GaussianBlurImage.

   Required: Gaussian blur the image "Seattle.jpg" with a sigma of 4.0, and save as "2.png".

3. (10 points) Implement the function SharpenImage(image, sigma, alpha) to sharpen an image. "sigma" is the Gaussian standard deviation and alpha is the scale factor (see Slide 16 in Edges.) You do not need to use the second derivative for this.

   Required: Sharpen "Yosemite.png" with a sigma of 1.0 and alpha of 5.0 and save as "4.png".

4. (10 points) Implement SobelImage(image) to compute edge magnitude and orientation information. SobelImage should display the magnitude and orientation of the edges in an image. You may use "rgb2gray" command in Matlab to convert color image into a gray scale image and then run the filter on that.

   Required: Compute Sobel edge filter on "LadyBug.jpg" and save as "5a.png" and "5b.png".

5. (10 points) Implement BilinearInterpolation(image, x, y) to compute the linearly interpolated pixel value at (x, y). Both x and y are continuous values.

   Required: Upsample image "Moire_small.jpg" to be 4 times larger in each direction (16 times more image area) once with nearest neighbor interpolation and save as "6a.png" and once with bilinear interpolation and save as "6b.png".

6. (10 points) Implement FindPeaksImage(image, thres) to find the peak edge responses perpendicular to the edges. The edge magnitude and orientations can be computed using the Sobel filter you just implemented. A peak response is found by comparing a pixel's edge magnitude to two samples perpendicular to an edge at a distance of one pixel (slide "Non-maximum suppression" in EdgeDetection), call these two samples e0 and e1. Compute e0 and e1 using BilinearInterpolation. A pixel is a peak response if it is larger than the following three : threshold ("thres"), e0, and e1. Assign the peak responses a value of 255 and everything else 0.

   Required: Find the peak responses in "Circle.png" with thres = 40.0 and save as "7.png".

7. **(10 extra points)** Implement BilateralImage(image, sigmaS, sigmaI) to bilaterally blur an image. "sigmaS" is the spatial standard deviation and "sigmaI" is the standard deviation in intensity. Hint: The code should be very similar to GaussianBlurImage.

8. **(10 extra points)** Implement the Hough transform to find lines using the peaks found from FindPeaksImage.

9. **(10 extra points)** Create a movie from progressively applying filters. For example, try iteratively applying GaussianBlurImage then FindPeaksImage, fun things happen after several iterations. The more creative the better. You may just output a bunch of images, i.e., 1.jpg, 2.jpg, 3.jpg, etc. or make a gif animation, or use "movie" command in Matlab.

Section 2: Written Assignment

1. (10 points) Run GaussianBlurImage and SeparableGaussianBlurImage with sigma = 2, 4, 8 on "Seattle.jpg". How many seconds does it take to run each function? How long do you think it would take to run each with sigma = 32?

2. (10 points) What is the best amount of blur to apply when down-sampling Moire.jpg by 8x (pressing "Half Size" 3 times)? Does down-sampling "Seattle.jpg" require the same amount of blur?

3. (10 points) Can you find an edge in "TightRope.png" that is visible to the human eye, but does not have a strong response from the Sobel edge detector?

4. (10 points) If you rotate the image 20 times by 2 degrees, does it produce the same result as rotating the image by 40 degrees? If not, why? You may use "imrotate" command in Matlab.