Due: 3^{rd} **March, 2021 at 2:00 PM**

1 Introduction

This project is designed to test your understanding of filtering and edge detection. You will be developing your own Canny edge detection algorithm from scratch. This project has far more coding than Homework 1 so *please start early.*

ELMS has a template "Project1_template.py" made up of functions that I would like you to complete and test. You will also turn in a pdf report describing your work and illustrating the output of your algorithm.

2 Restricted functions

Do not import any modules not included in the template. Do not use np.convolve, skimage.feature.canny, cv2.canny, or any other similar functions that directly solve parts of the assignment. You are permitted to use np.fft2 and related functions if you so choose – they are not required to complete the first two parts of the assignment. If in doubt about whether a function is OK to use, please message Piazza with a public post.

3 Part 1: Filtering

3.1 Gaussian Kernel

10 points

Write a function that forms a `3sigma`×`3sigma` Gaussian kernel with standard deviation `sigma`.

```
gausskernel(sigma)
```

You can assume `sigma` is a positive integer.
Hint: `np.meshgrid` is useful here.

3.2 Convolution

20 points

Write a function that convolves an image with a given filter.

```
myImageFilter(I, h)
```

As input, the function takes a grayscale image (`I`) and a convolution filter stored as a 2D numpy array `h`. The output of the function should be an image of the same size as `I` which results from convolving `I` with `h`. You can assume that the filter `h` is odd sized along both dimensions. You will need to handle boundary cases on the edges of the image. For example, when you place a convolution mask on the top left corner of the image, most of the filter mask will lie outside the image. One solution is to output a zero value at all these locations.

Test your function by filtering the included "Iribe.jpg" image using Gaussian kernels with standard deviations 3, 5, and 10. Please include the results in your report.

3.3 Filters

10 points

Apply the filters `h1`, `h2`, and `h3` from the template file to the "Iribe.jpg" image and add these results to your report. Describe and explain what effect each of them has on the image.

4 Part 2: Edge Detection

You will now implement and test the canny edge detection algorithm. You will write a function that applies the canny edge detection algorithm to an image.

```
    myCanny(I, sigma, t_low, t_high)
```

As input, the function takes a grayscale image (`I`), a smoothing parameter (`sigma`), a lower threshold (`t_low`), and an upper threshold (`t_high`). The function should return a binary image made up of the edges of `I`.

Each of the following steps take place within the `myCanny` function.

4.1 Noise reduction with Gaussian filtering

5 points

Using the functions you developed in the previous section, apply a Gaussian filter to the image with standard deviation `sigma`.

4.2 Finding image gradients

15 points

- Compute the derivatives ($D_x(x,y)$ and $D_y(x,y)$) using following filters respectively

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad , \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

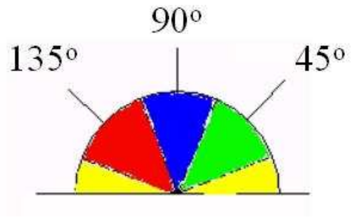
- Compute the gradient magnitude

$$D = \sqrt{D_x^2(x,y) + D_y^2(x,y)}$$

and the angle of the gradient

$$\theta = \arctan\left(\frac{D_y(x,y)}{D_x(x,y)}\right)$$

Compute θ' by rounding the angle θ to one of four directions $0°$, $45°$, $90°$, or $135°$. For edges, $180° = 0°$, $225° = 45°$, etc. This means θ in the ranges $[-22.5°, \ldots\ 22.5°]$ and $[157.5°, \ldots\ 202.5°]$ would "round" to $\theta' = 0°$. For a pictorial representation, each edge takes on one of four colors:



Here, the colors would repeat on the lower half of the circle (green around $225°$, blue around $270°$, and red around $315°$)

4.3 Edge thinning / Non-maximum suppression

15 points

Three pixels in a 3×3 around pixel (x, y) are examined:

- If $\theta'(x, y) = 0°$, then the pixels $(x + 1, y), (x, y)$, and $(x - 1, y)$ are examined.
- If $\theta'(x, y) = 90°$, then the pixels $(x, y + 1), (x, y)$, and $(x, y - 1)$ are examined.
- If $\theta'(x, y) = 45°$, then the pixels $(x + 1, y + 1), (x, y)$, and $(x - 1, y - 1)$ are examined.
- If $\theta'(x, y) = 135°$, then the pixels $(x + 1, y - 1), (x, y)$, and $(x - 1, y + 1)$ are examined.

If pixel (x, y) has the highest gradient magnitude of the three pixels examined, it is kept as an edge. If one of the other two pixels has a higher gradient magnitude, then pixel (x, y) is not on the "center" of the edge and should not be classified as an edge pixel.
At the end of this process, you should achieve a one pixel wide edge.

4.4 Hysterisis Thresholding

15 points

Some of the edges detected by the above steps will not actually be valid, but will just be noise. We would like to filter this noise out. Eliminating pixels whose gradient magnitude D falls below some threshold removes the worst of this problem, but it introduces a new problem.

A simple threshold may actually remove valid parts of a connected edge, leaving a disconnected final edge image. This happens in regions where the edge's gradient magnitude fluctuates between just above and just below the threshold. *Hysteresis* is one way of solving this problem. Instead of choosing a single threshold, two thresholds t_{high} and t_{low} are used. Pixels with a gradient magnitude $D < t_{low}$ are discarded immediately. However, pixels with $t_{low} \leq D < t_{high}$ are only kept if they form a continuous edge line with pixels with high gradient magnitude (i.e., above t_{high}).

- If pixel (x, y) has gradient magnitude less than t_{low} discard the edge (write out black).
- If pixel (x, y) has gradient magnitude greater than t_{high} keep the edge (write out white).
- If pixel (x, y) has gradient magnitude between t_{low} and t_{high}:
  - Identify all the pixels connected to pixel (x, y) via a path through pixels with gradient magnitudes greater than t_{low}. (See the hint.)
  - If any pixel in that set has a gradient magnitude greater than t_{high} keep the edge (write out white).
  - Else, discard the edge (write out black).

Hint: `scipy.ndimage.measurements.label` finds and labels all connected (as defined by the structure argument) components in a binary image. Using this function, you can assign a label to every pixel with a gradient magnitude greater than t_{low} and then decide if it should be kept by determining if any pixels with gradient magnitudes over t_{high} share the same label.

4.5 Testing

10 points

Perform edge detection on "Iribe.jpg" with different values of `sigma`, `t_low`, and `t_high`. Save and discuss the results for different values of each parameter.

5 *Extra Credit*: Hybrid Images

15 points

Combine the low frequencies of one image with the high frequencies of another to form a hybrid image, preferably with your selfies.
Below are some references on hybrid images.

1. https://en.wikipedia.org/wiki/Hybrid_image
2. http://cvcl.mit.edu/hybrid/OlivaTorralb_Hybrid_Siggraph06.pdf

Hint: Combining images in the Fourier domain is an easy way to form a hybrid image. Lecture 8 includes most of the code required to complete this task.

Submission Instructions

Your canvas submission should consist of a zip file named **YourDirectoryID_Project1.zip**, for example xyz123_Project1.zip. The file must contain the following:

- Iribe.jpg
- Project1.py or .ipynb
- report.pdf

If performing the extra credit portion of the assignment, also include the images you used to form the hybrid image:

- Img1.jpg
- Img2.jpg

Collaboration Policy

You are encouraged to discuss ideas with your peers. However, the code should be your own and should represent your understanding of the assignment. Code should not be shared or copied. If you reference anyone else's code in writing your project, you must properly cite it in your code (in comments) and in your report.

Please list any individuals you collaborated with at the end of your report.

Plagiarism

Plagiarism of any form will not be tolerated. You are expected to credit all sources explicitly. If you have any doubts regarding what is and is not plagiarism, talk to me.

Credit

Thanks to Ashok Veeraraghavan, Ioannis Gkioulekas, and Mohammad Teli for sharing their course resources.