COMP 4102A: Assignment 1

Posted: January 26,2020 Due by 23:59 Sunday, February 9

Instructions for submission: Please write a document (pdf or doc) with your solutions on theory questions. Include your codes for edge detection and hybrid images separately in two folders. Submit a zipped folder contains: theory questions solutions, edge detection folder and hybrid images folder. Please submit through cuLearn. You are expected to work on the assignment **individually**.

1 (30 points) Theory questions

1. (5 mark) What is the time complexity of a convolution with an N by N sized kernel when using a direct convolution with a square 2d mask, and when using a separable kernel? Each of the answers is in the form of O(?) where ? is an expression in N. If you do not know what big O notation means, it is an expression which is proportional to the running time of a given algorithm.

2. (5 mark) Mark each of the following as true or false. A) When we smooth an image with a Gaussian filter we are performing a high-pass filter operation which is accentuating the high frequencies in the image. B) All possible image filters are also linear image filters. C) A median filter is a linear image filter. D) A Gaussian filter is not a linear image filter.

3. (5 mark) Assume we have a 3 by 3 box filter or averaging filter as we called it in our lectures on filtering. If this convolution filter is applied repeatedly (over and over again) to an image then in the limit what does the image look like? That is what are the characteristics of that image if this filter were applied an infinite number of times?

4. (5 points) You convolve an image I with a filter $f_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, then take the output and convolve it with another filter $f_2 = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$. Is it possible to get the same final result by just performing one convolution? If so, what is the filter to do this?

5. (5 points) Scale a vector $[x \;\; y]^T$ in the plane can be achieved by $x' = sx$ and $y' = sy$ where s is a scalar.
   (a) Write out the matrix form of this transformation.
   (b) Write out the transformation matrix for homogeneous coordinates.
   (c) If the transformation also includes a translation $x' = sx + t_x$ and $y' = sy + t_y$
   Write out the transformation matrix for homogeneous coordinates.
   (d) What is the equivalent of the above matrix for three-dimensional vectors?

6. (5 points) Find the least square solution $\overline{\mathrm{x}}$ for $Ax = b$ if

$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

Verify that the error vector b -$A\bar{x}$ is orthogonal to the columns of A.

2 (35 points) Edge detection

Write a function that finds edge intensity and orientation in an image. Display the output of your function for one of the given images in the handout.

$$function[img1] = myEdgeFilter(img0, sigma) \tag{1}$$

The function will input a greyscale image (img0) and scalar (sigma). sigma is the standard deviation of the Gaussian smoothing kernel to be used before edge detection. The function will output img1, the edge *magnitude* image.

First, use your convolution function to smooth out the image with the specified Gaussian kernel. This helps reduce noise and spurious fine edges in the image. The size of the Gaussian filter should depend on sigma (e.g., hsize = 2 * ceil(3 * sigma) + 1).

The edge magnitude image img1 can be calculated from image gradients in the x direction and y direction. To find the image gradient imgx in the x direction, convolve the smoothed image with the x-oriented Sobel filter. Similarly, find image gradient imgy in the y direction by convolving the smoothed image with the y-oriented Sobel filter. You can also output imgx and imgy if needed.

In many cases, the high gradient magnitude region along an edge will be quite thick. For finding lines its best to have edges that are a single pixel wide. Towards this end, make your edge filter implement non-maximum suppression, that is for each pixel look at the two neighboring pixels along the gradient direction and if either of those pixels has a larger gradient magnitude then set the edge magnitude at the center pixel to zero. Map the gradient angle to the closest of 4 cases, where the line is sloped at almost $0^o, 45^o, 90^o$, and 135^o. For example, 30^o would map to 45^o. Please refer to the slides on non-maximum suppression for more details. You do not need to do hysteresis thresholding.

Your code cannot call on OpenCV edge function, or any other similar functions. You may use the edge detection from library just for comparison and debugging.

Submission

All results should be in a folder called **Edge detection**. You can use your own images and you should include the original image in the folder if you use your own. Submit your code with the **edge detection** result after non-maximum suppression, the **gradient magnitude** image and the **gradient orientation** image. Your code should have comments for each function and unclear variables.

3 (35 points) Hybrid Images

The goal of this assignment is to write an image filtering function and use it to create hybrid images using a simplified version of the SIGGRAPH 2006 **paper** by Oliva, Torralba, and Schyns. Hybrid images are static images that change in interpretation as a function of the viewing distance. The basic idea is that high frequency tends to dominate perception when it is available, but, at a distance, only the low frequency (smooth) part of the signal can be seen. By blending the high frequency portion of one image with the low-frequency portion of another, you get a hybrid image that leads to different interpretations at different distances.

This project is intended to familiarize you with image filtering. Once you have created an image filtering function, it is relatively straightforward to construct hybrid images.

This project requires you to implement 5 functions each of which builds onto a previous function:

- cross-correlation-2d
- convolve-2d
- gaussian-blur-kernel-2d
- low-pass
- high-pass

Image Filtering. Image filtering (or convolution) is a fundamental image processing tool. See chapter 3.2 of Szeliski and the lecture materials to learn about image filtering (specifically linear filtering). Numpy has numerous built in and efficient functions to perform image filtering, but you will be writing your own such function from scratch for this assignment. More specifically, you will implement **cross-correlation-2d**, followed by **convolve-2d** which would use cross-correlation-2d.

Gaussian Blur. As you have seen in the lectures, there are a few different way to blur an image, for example taking an unweighted average of the neighboring pixels. Gaussian blur is a special kind of weighted averaging of neighboring pixels, and is described in the lecture slides. To implement Gaussian blur, you will implement a function gaussian-blur-kernel-2d that produces a kernel of a given height and width which can then be passed to convolve-2d from above, along with an image, to produce a blurred version of the image.

High and Low Pass Filters. Recall that a low pass filter is one that removes the fine details from an image (or, really, any signal), whereas a high pass filter only retails the fine details, and gets rid of the coarse details from an image. Thus, using Gaussian blurring as described above, implement high-pass and low-pass functions.

Hybrid Images. A hybrid image is the sum of a low-pass filtered version of the one image and a high-pass filtered version of a second image. There is a free parameter, which can be tuned for each image pair, which controls how much high frequency to remove from the first image and how much low frequency to leave in the second image. This is called the "cutoff-frequency". In the paper it is suggested to use two cutoff frequencies (one tuned for each image) and you are free to try that, as well. In the starter code, the cutoff frequency is controlled by changing the standard deviation (sigma) of the Gaussian filter used in constructing the hybrid images. We provide you with the code for creating a hybrid image, using the functions described above.

Forbidden functions. For just this assignment, you are forbidden from using any Numpy, Scipy, OpenCV, or other preimplemented functions for filtering. This limitation will be lifted in future assignments, but for now, you should use for loops or Numpy vectorization to apply a kernel to each pixel in the image. The bulk of your code will be in cross-correlation-2d, and gaussian-blur-kernel-2d with the other

functions using these functions either directly or through one of the other functions you implement.

Figure 1: Left: Two original images. Right: The low-pass (blurred) and high-pass versions of these images.

Adding the high and low frequencies together gives you the image at the top of this page. If you're having trouble seeing the multiple interpretations of the image, a useful way to visualize the effect is to progressively downsample the hybrid image as is done below:

Figure 2: Hybrid images in multiscales

Submission

The sample images of cat and dog are included in folder data. However, **you should chose a different cat or dog image from front view** for your results. You can also use any other images. Fore example you can use your own face images in two different extreme expressions as input. Since we don not apply alignments for this assignment it's better to chose images that already have good alignments.

You can use the same Gaussian function you implemented for edge detection. You should include all results in a folder named **Hybrid images**. Submit all five functions implemented in **hybrid.py** or **hybrid.cpp**. Submit the left and right images you used to create hybrid image (name them left.png, right.png). Submit the hybrid image (**hybrid.png**) produced by using your implementation and left,right images. Include a read-me file that contains high pass and low pass filter parameters(kernel size and kernel sigma) and Mix-in ratio. It should also contain which image's higher/lower frequencies are used. Optionally you can add comments on something interesting or different you did in the project.

Acknowledgements. Hybrid image assignment is based on versions developed by James Hays, Derek Hoiem and Noah Snavely.