CS5290: COMPUTER VISION
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, IIT-HYDERABAD

Assignment 2

Max Points: 120
This assignment is to be done individually.

Submission Due: 4 Sep 2015 11:59 pm

INSTRUCTIONS

- Your submission, named Your_Roll_No_Assign2 should comprise of a ZIP file with a DOC/DOCX/PDF file with your theory/descriptive answers, and a .PY file for each programming exercise. Please name the .PY file with the problem number (e.g. Prob2a.py). Make sure your code actually works (in an empty workspace) and that all your results are displayed properly.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the "Grace Days" section of the course portal.
- Your evaluation will of course depend on the correctness of your answer, but also on a clear presentation of your results and good writing style. Note that you will get marks for the solution, not for the result. If you get stuck, try to explain why and describe the problems you encountered – you can get partial credit even if you did not complete the task. So please hand in enough information for us to understand what you did, what things you tried, and how it worked!
- Please read the department plagiarism policy available under the course site. Do not engage in any form of cheating – please talk to instructor or TA if you have concerns.

1 THEORY

1. **Linear Filters** *(3 + 4 + 4 + 4 + 5 = 20 points)*: In class, we introduced 2D discrete space convolution. Consider an input image $I[i, j]$ and a filter $F[i, j]$. The 2D convolution $F * I$ is defined as

$$(F * I)[i, j] = \sum_{k,l} I[i-k, j-l]F[k, l] \tag{1.1}$$

   a) Convolve the following I and F (using pen and paper). Assume we use zero-padding where necessary.

$$I = \begin{bmatrix} 2 & 0 & 1 \\ 1 & -1 & 2 \end{bmatrix} F = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \tag{1.2}$$

   Please DO NOT write programs. You must show all necessary steps here. It will also be helpful for answering question (d).

   b) Note that the F given in Equation 1.2 is separable, that is, it can be written as a product of two filters: $F = F_1 F_2$. Find F_1 and F_2. Then, compute $(F_1 * I)$ and $F_2 * (F_1 * I)$, i.e., first perform 1D convolution on each column, followed by another 1D convolution on each row. (Please DO NOT write programs. You must show all necessary steps here.)

   c) Prove that for any separable filter $F = F_1 F_2$
   $F * I = F_2 * (F_1 * I)$
   Hint: Expand equation 1.1 directly.

   d) Carefully count the exact number of multiplications (multiplications only, including those multiplications due to zero-padding) involved in part (a) and part (b). which one of these requires fewer operations? *Note:* We are asking for two exact numerical values here (no approximations). You may find the computation steps you wrote down for (a) and (b) helpful here.

   e) Consider a more general case: I is an $M_1 \times N_1$ image , and F is an $M_2 \times N_2$ separable filter.

      i. How many multiplications do you need to do a direct 2D convolution?

      ii. How many multiplications do you need to do 1D convolution on rows and columns?
      Hint: For (i) and (ii), we are asking for two functions of M_1, N_1, M_2 and N_2 here, no approximations.

      iii. Use Big-O notation to argue which one is more efficient in general: direct 2D convolution or two successive 1D convolutions?

2. **Canny Edge Detector** *(5 + 5 = 10 points)*: Suppose the Canny edge detector successfully detects an edge. The detected edge (shown as the red horizontal line in Figure 2a) is then rotated by θ, where the relationship between a point on the original edge (x, y) and a point on the rotated edge (x', y') is defined as

$$x' = xcos\theta \; ; \; y' = ysin\theta$$

a) Will the rotated edge be detected using the same Canny edge detector? Provide either a mathematical proof or a counter example. *Hint:* The detection of an edge by the Canny edge detector depends only on the magnitude of its derivative. The derivative at point (x,y) is determined by its components along the x and y directions. Think about how these magnitudes have changed because of the rotation.

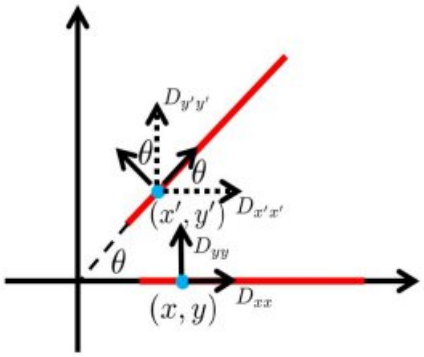

Figure 1.1: Canny Edge Detector

b) After running the Canny edge detector on an image, you notice that long edges are broken into short segments separated by gaps. In addition, some spurious edges appear. For each of the two thresholds (low and high) used in hysteresis thresholding, state how you would adjust the threshold (up and down) to address both problems. Assume that a setting exists for the two thresholds that produce the desired result. Explain your answer briefly.

2 PROGRAMMING

1. **Cross-correlation** *(5 + 5 + 10 = 20 points)***:** In class, we covered cross-correlation, in which a template image is multiplied with sections of a larger image to measure how similar each section is to the template. Normalized cross-correlation is a small refinement to this process. More specifically, before multiplying the template with each small section of the image, the section is scaled and offset so it has zero mean and variance of 1. This increases accuracy by penalizing image sections which have high intensity but do not match the pattern of the template.

   a) Write your own code to perform normalized cross-correlation (NCC) given an image and a template. You can use the provided image ("u2cuba.jpg") and template ("trailer.png") to test your code.

b) In the result from the previous question, describe why the peak occurs where it does. Also, explain the straight-line artifacts you observe in the cross-correlation (Hint: look at the template and the original image).

c) Now, perform cross-correlation using the larger template ("trailerSlightlyBigger.png"). Note that the larger template does not exactly match the image. Describe your results, and why they are different from part (a). What does this tell you about the limitations of cross-correlation for identifying objects in real-world photos? (If required, scale the image intensity in the correlation image so that it covers the full range. If you do this, please remember that the color white will not denote the same value in the two cross-correlation images.)

d) Above, we saw that cross-correlation can be fragile. One way to make it less fragile is to perform cross-correlation using many templates to cover the different ways an object may appear in an image. Suppose we wish to search for N_R possible rotations of an object at N_S possible sizes. Assume the image is size $n \times n$ and the template is roughly size $m \times m$. How many mathematical operations will the entire search require? Here, we're looking for a "Big-O Notation" estimate. In other words, you may neglect constant factors, such as the effects of image edge padding and smaller terms.

2. **Convolution** *(10 points)*: Write your own Python function that implements the discrete 2D convolution operator, i.e., given an intensity image and a filter mask (a matrix with coefficients) your function should convolve the source image with the filter mask and return the resulting output image. You may assume that the filter mask is a square matrix with odd size (e.g. 3 x 3, 5 x 5, 7 x 7). You will need to decide on a sensible strategy for dealing with the image borders. An example skeleton for your function is given as follows.

```
def my_conv2(im_in, kernel):
...
return (im_in*kernel)
```

To execute your operator, you should call your implemented function as:

```
» my_conv2(inputimage, mask)
```

Please note: your implementation will be slow, depending on the input image size and kernel size. So you may want to test your code with smaller, "thumbnail" images during debugging.

3. **Edge Detection** *(15 + 5 = 20 points)*:

a) Using your implementation of the convolution operator, try out and compare the following edge filters on the image "clown.tif" (provided with this assignment). The filters are described in the lecture slides.

- Sobel edge detector: size 3×3, vertical G_x and horizontal G_y. Compute the approximate magnitude $|G|$ of the filter responses: $|G| = |G_x| + |G_y|$.
- Laplacian for edge detection: size 3×3.

Show the original image and all result images (labeled with the used filter kernel and filter kernel size). Compare the results and give a qualitative comment.

b) Compare your results with the 2-D convolution from the SCIPY package function (`scipy.signal.convolve2d`). Compare the outputs and speed issues that you observed between your code and the inbuilt function.

4. **Fourier transform** *(10 + 10 = 20 points)***:** In this exercise, we take a look at the discrete 2D Fourier transform in Python (Numpy). The function `fft2` computes the 2D fast Fourier transform (FFT) of an input matrix (the output is complex-valued!); `ifft2` computes the inverse FFT of a complex valued input matrix (the output might be complex-valued; therefore use the function `real` to get the real part only, or `absolute` for the magnitude); and `fftshift` shifts the zero-frequency component to the center of spectrum by swapping quadrants (this is useful for visualization, as mentioned in class).

   a) *Phase and Magnitude:* Write your own code that computes and displays the magnitude and phase of an intensity image (use the provided image file "mandrill.tif" for this exercise). Check the lecture slides for the mathematical formulae for the magnitude and phase. Include your magnitude and phase plots of the mandrill image in your submission.

   b) *Phase vs Magnitude:* Investigate the "information content" of phase and magnitude. Swap the phase of the clown image (file "clown.tif") and the phase of the mandrill image (previous task), i.e. replace the mandrill's phase information with the phase information from the clown image and vice versa, and combine them into new matrices. Then compute the inverse Fourier transform of the created complex matrices. Your solution should include magnitude and phase plots of the images before and after the transformation. Now, comment on the following: What do the magnitude plots tell you? Can you recognize the clown or the mandrill? Are the phase plots more informative than the magnitude plots? Can you still recognize the mandrill's fur, for example? Take a look at the generated images: which information is more important — phase or magnitude?

   NOTE: Other useful functions are: `angle`, `log`. Convert the input image from RGB to grayscale beforehand.

5. **Hybrid Images** *(20 points)***:** This exercise is based on the paper on this topic in SIGGRAPH 2006 by Oliva, Torralba, and Schyns. A hybrid image is the sum of a low-pass filtered version of one image and a high-pass filtered version of a second image. 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. Here is an example (Figure 2.1):

   If you're having trouble seeing the multiple interpretations of the hybrid image, a useful way to visualize the effect is by progressively downsampling the hybrid image as is done here (Figure 2.2).

Figure 2.1: Example of a hybrid image

Figure 2.2: Visualizing a hybrid image at different resolutions

We provide you with 5 pairs of aligned images which can be merged reasonably well into hybrid images. The hybrid images will differ depending on which image you assign as image1 (which will provide the low frequencies) and which image you assign as image2 (which will provide the high frequencies). You need to submit your code to generate a hybrid image, as well as any one interesting hybrid image example generated using your code. (You can use other images too, but you'll need to align the images in a photo editor such as Photoshop).

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". The cutoff frequency can be controlled by changing the standard deviation of the Gaussian filter used in constructing the hybrid images. You will want to tune this for every image pair to get the best results.

NOTE: The high frequency image may be zero-mean with negative values; you can visualize it by adding 0.5. In the resulting visualization, bright values are positive and dark values are negative.