Submission details

- This assignment should be done in pairs (contact the TA if this is a problem).
- The topics of this assignment are filtering, K-Means and Edge Detection.
- The submission date is **2/24/2016**. Please pay attention to the late submission policy.
- Coding should be done in Matlab or Python. We prefer Python 2, but Python 3 is possible.
- You are not required to use any specific function or library, unless stated otherwise. If in doubt, please contact the TA via email or the piazza website.
- To install (email TA if you run into trouble)
    - Ubuntu Linux:
      sudo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-pandas python-sympy python-nose
    - On Windows or Mac, you may have to install a Python distribution. We recommend the Anaconda distribution, but other distributions are also possible.
- For submission, package up your code as a zip file. Include your written answers as a pdf file named writeup.pdf. Include graphs and images in your writeup, and please label them so we know which figures go with which sub-problem.
- Send the final zip file to the TA. Add the course number to the subject of the email.
- If you have any questions about this assignment, please contact the TA stinger@tx.technion.ac.il.

Task 1: Image filtering and enhancement

1. Convolve the following sub-image with a normalized 3×3 mean filter. What is the output?

4	1	6	1	3
3	2	7	7	2
2	5	7	3	7
1	4	7	1	3
0	1	6	4	4

2. Redo the convolution with a median filter. What is the difference from item 1?
3. For the same sub-image, find the gradient magnitude and gradient direction at the center entry using the Sobel operator.
4. Design three different filters of the following types:
    a. A filter kernel which takes into account only the distance between pixels.
    b. A filter kernel which takes into account only the distance between pixel values.
    c. A filter kernel which takes into account both distances.

   Use the provided three images and apply the three filters to them. Then add some noticeable noise to the images, and apply the filters again. What can you conclude from the filter operation? Show your results.
5. Implement the following Unsharp Masking: $f_{unsharp}(x,y)=2\cdot f(x,y)-g(x,y)*f(x,y)$, where $f(x,y)$ is the input image and $g(x,y)$ is a smoothing operator. For $g(x,y)$ we will use two Gaussian filters (from scipy): one with $\sigma=0.75$ and with $\sigma=2.5$. Use the provided three images and blur them with each of the smoothing Gaussian filters. Then apply the Unsharp Masking scheme on the blurred images. What are the results? Explain the change in the images, and the difference between the smoothing operators.

Task 2: Color quantization with k-means

In this exercise you will write a code that quantizes the color space of an image. We will explore two color spaces, RGB and LAB.

1. Write a function that gets as input an RGB image, quantizes the colors in the image via k-means, and then replaces the color at each pixel with the color of its quantized version. The number of clusters k is a variable of your function.
2. Write a function that gets an RGB image as input and converts it to LAB. The function then quantizes only the L channel and replaces every pixel in it with the intensity of its quantized version. Next, take the quantized L channel and the original A,B channels and convert the image back into RGB.
3. Write a function that computes SSD (Sum of Squared Distances) between two RGB images.
4. Write a function that gets as input an RGB image and draws the histogram of the L channel in two ways. First, when the histogram bins are spaced regularly. Second, when the bins are determined according to the colors you get when quantizing the L channel.
5. Write a function that calls all the functions above and presents the histograms of the images before and after quantization. The function should display the images and report the SSD between the original image and its quantized version. It should do this for two values of k (you can choose the values). Please make sure each result has a clear title and explanation.

Please run the function you wrote on the two provided images. Explain the results you get. What causes the differences before and after quantization? What is the effect of changing k? Do you get the same results for every execution of your program? Why?

Task 3: Edge detection

In this question you will gain some experience with edge detection and with performance measurement of it. You should demonstrate your results on the three provided images.

A. Given an image, an edge detection process should ideally detect the boundaries of all important objects. To see how edge detectors function in practice, we ask you to apply three edge detectors, Sobel, Gaussian-Laplace and Canny, to the gray level images.
How to implement the detectors:
- Canny is fully implemented in skimage.feature library.
- Sobel filtering is implemented in skimage.feature, but global thresholding is then needed.
- Gaussian-Laplace filtering is implemented in scipy.ndimage.filters, but zero crossing and local thresholding are then needed. You should implement both for each pixel in a 3x3 neighborhood.

Apply the detectors to the images and tune the parameters of each detector manually such that they give the best edge image according to your subjective opinion.

1. Explain in a few sentences the way of operation of each of the three detectors.

2. Explain the threshold parameter and the sigma parameter of the detectors. What happens when you increase/decrease each of the parameters? Which parameter values work best?

3. Run the Gaussian-Laplace method with threshold = 0. The edges are supposed to be closed curves. Explain why this happens.

4. Invent your own edge detection method. It can be a variation of one of the tested ones or some other detector you came up with. Please do not copy something from the literature. We would like you to identify a limitation of the detectors you tested above and design your detector such that it aims to solve that limitation. A good idea with a short clear explanation is what would please us most. Demonstrate the results visually.

B. To measure the performance of an edge-detector, it is common to compare the detection results with those chosen manually by a person. The comparison is between two sets, the set E of pixels detected as edges and the ground truth (GT) set of pixels selected manually. To compare the sets, it is common to use two measures, Precision (P) and Recall (R):

$$P = \frac{|E \cap GT|}{|E|} \quad R = \frac{|E \cap GT|}{|GT|}$$

An edge detector usually depends on a parameter. Usually, the edge detector process calculates some measure of spatial change (e.g. gradient magnitude) and then accepts the pixel as an edge if the measure exceeds a threshold. Note that for very small thresholds almost all pixels are detected, including most GT edges but also many other pixels (small P, large R). Conversely, for large thresholds, many of the pixels detected are true edges but also many of the true edges are missed (large P, small R).

For good performance, both precision and recall are important. Therefore, it is common to express performance by a precision-recall curve, parameterized by some parameter (e.g. the threshold). Another way is to take an average of both. The common average measure is called the F-measure:

$$F = 2\frac{PR}{P+R}.$$

1. Explain in a couple of sentences why both Precision and Recall are important.

2. Using the GT pictures provided, plot a precision-recall curve averaged over all images. Plot the curve for each method while taking the threshold as the varying parameter (i.e., one plot, with three curves for every image). Choose the other parameters in any reasonable way (no need to optimize them).

3. The edge detection process often makes spatial errors. Change your code to consider the detection as true even if it is 3 pixels far from a GT edge, and re-plot the curves. Use an efficient method to test whether a pixel in E is 3 pixels close to a pixel in GT. Describe this method in a few sentences. Is the difference significant? Hint: make the GT edges wider just for the calculation of $|E \cap GT|$.

4. Show F-measure curves for each image. Which method achieves the best F-measure for every image? Write your conclusion regarding all three edge detectors.