Submission details

- This assignment should be done in pairs (contact the TA if this is a problem).
- The topics of this assignment are Image Pyramids, Hough Transform and Segmentation.
- The submission date is **3/09/2016**. 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.
- 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 Pyramids

In this exercise you will use the Gaussian and Laplacian pyramids for image enhancement. The image pyramid was originally created by Peter J. Burt and Edward H. Adelson for image coding, but it was soon surpassed by JPEG. The interested reader can find the paper describing their work using the following citation.

Burt, P.J. and Adelson, E.H., "The Laplacian pyramid as a compact image code", *IEEE Trans. on Communications*, Vol. 31, No. 4, pp. 532-540, 1983.

1. Implement a function for creating Gaussian and Laplacian Pyramids with five levels for a given input image. Implement also a function for reconstructing an image from a given Laplacian Pyramid. Please do not use library functions in this item.
2. Build Gaussian and Laplacian Pyramids for the images mandril.tif and toucan.tif. What are the differences between the Laplacian Pyramids of the two images? Explain.
3. Reconstruct the original images from their Laplacian Pyramids. Show in a simple way that the original images and the reconstructed ones are identical.
4. Implement an image enhancement method for the images focus1.tif and focus2.tif. The method should create a single entirely-focused image from the two partially-focused ones. Use a pyramid with four levels. Show the results and discuss their quality.

Task 2: Finding circles using Hough Transform

In this exercise you will implement a system that can find circles in images. The system will get an image and a radius as input and will return the coordinates of all the centers of the circles with that radius. Please do not use library functions and implement the Hough Transform on your own.

The main function will have the following interface:

[centers] = detectCircles(im, radius, usegradient)

im - input image
radius - the desired radius
usegradient - a flag stating whether the edge gradient should be used or not in the vote counting
centers - an array of size Nx2, where N is the number of detected circles, and each row contains the x and y coordinates of a circle.

Don't forget to write a short documentation for each function you write! We will look for that.

You should perform the following experiments on the provided images:

1. Explain your algorithm and how it works with and without using the image gradients.
2. Draw the circles you found on top of the provided images for a radius of your choice.
3. What are the advantages of using the gradient? What are the disadvantages?
4. For one of the images, show the Hough space you got. Please explain what you see.
5. For one of the images, try three different quantizations of the Hough space. What is the effect of changing the quantization?
6. Can your system be extended to finding circles of any radius? i.e., can we remove the "radius" variable and instead return all circles in the image. If you can think of a practical solution, implement it and show your results. The output array now should be Nx3, where the 3rd column contains the circle radius.
7. BONUS: Can you think of a method for using Hough transform to discover rectangles? If so, please describe it shortly (No need to code this one).

Task 3: Segmentation with Superpixels

In this exercise you will get to know a simple method for creating Superpixels, called Simple Linear Iterative Clustering (SLIC). Creating Superpixels is an important segmentation technique, used for many Computer Vision applications, such as Object Detection, Depth Estimation and Human Pose Estimation. It allows us to perform many processes on perceptually meaningful regions rather than on the pixels themselves, which enables reduced complexity.

The algorithm:

a. Create K region centers on an equally sampled regular grid on the input $M \times N$ image.
b. For each center, create a $2S \times 2S$ window centered at the center, where $S = \sqrt{\frac{M \cdot N}{K}}$.
c. For each pixel inside the window, extract a feature vector and check the distance to the center's feature vector using some distance metric.
d. Decide whether to allocate the pixel to the center or not using some threshold value.
e. After going over all the region centers, recalculate the center's position for each cluster of associated pixels by averaging over the cluster members.
f. Go back to stage b. and repeat b.-f. until convergence or a predefined number of iterations. The resulting clusters are the superpixels.

For more details, see the paper:

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods”, *IEEE Trans. on PAMI,* Vol. 34, No. 11, pp. 2274 - 2282, 2012.

1. Implement the SLIC method for a general feature vector and general distance metric. Please write documentation for your functions. Do not use library functions in this item.
2. Our initial tests will incorporate a simple feature vector, similar to the Mean-Shift method: $\underline{f} = \begin{bmatrix} r & g & b & x & y \end{bmatrix}^T$. What will be a good distance metric for measuring similarity between a pixel and the center in this 5D space? Explain your choice.
3. Take the castle images and try three different weights for the color values and the spatial coordinates in your distance metric. Show the results and explain them.
4. For the same images, show a heat map of the distance between the pixel and the center in the 5D space, both at initialization and after convergence (with the same scale). Explain the result.
5. Take the castle images and try four different K values: 16, 64, 256, 1024. Measure the time it takes to create each of the results with the Python *timeit* library (in Matlab use Tic Toc). Explain the results and the times you get.
6. Implement the Boundary Recall measure for superpixel evaluation. This is exactly the Recall measure from HW1, measured on edge maps produced from the segmentations. Consider the detection as true even if it is 3 pixels far from a groundtruth edge, similarly to HW1, item 3.B.3.
7. Implement the Under-Segmentation measure for superpixel evaluation:

   $$U = \frac{1}{N} \sum_{S \in GT} \sum_{P \in SP} \min\left(|P \cap S|, |P| - |P \cap S|\right)$$

   Where S is a segment in the ground-truth segmentation GT, P is a superpixel in your segmentation SP and N is the number of pixels in the image.
8. Select four K values and calculate the above two measures and the running time for running your SLIC implementation on the given BSDS images, using their groundtruth segmentations. Use the given groundtruth edge maps for the Boundary Recall measure instead of calculating them from the groundtruth segmentations. Average both measures' results over the 5 groundtruth segmentations and edge maps for each image, and then over all the BSDS images to achieve a working point. Plot 3 graphs of K vs. the relevant measure and explain them.
9. Try to improve your results in item 8 using the following suggestions:
   a. A more complex distance measure.
   b. A different color space.
   c. Richer feature vectors (e.g. using gradients, etc.).
   d. A modification to the algorithm stages.

   Implement your suggestion and demonstrate its advantages on at least three different images, comparing to results from item 8. Explain what you changed and why, and also why do you think it worked (or not). Highly successful suggestions will receive a rewarding bonus.