Submission details

- This assignment can be done individually or in pairs (though we strongly encourage you to work in pairs).
- Programs should be in Python. You can use either Python 2 or Python 3.
- The example Python code uses the SciPy library (http://www.scipy.org), which provides Matlab-like arrays with similar functionality. It also has some handy functions for plotting graphs of signals, which the Python standard library lacks. These operations tend to make signal and image processing algorithms much easier to code. You are not required to use this library, but it could make your life easier.
- 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 containing SciPy. 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 or word file named writeup.[pdf|docx]. 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 (see next bullet). Add the course name to the subject of the mail.
- If you have any questions about this assignment, please contact the TA (stinger@tx.technion.ac.il).

Task 1: 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.

The main function will have the following interface:

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

im = input image
radius = the desired radius
usegradient = a binary variable 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.

Images for this task: Planets, ladybug, MoonCraters, colorful2, colorful3

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 we see.
5. For one of the images, try 3 different quantization 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 yes, please describe it shortly. (No need to code this one.)

Task 2: Color quantization with k-means

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

Images for this task: colorful1, colorful2, colorful3

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 HSV color space. The function then quantizes only the H channel and replaces every pixel in it with the value of its quantized version. Next, take the quantized H channel and the original S,V channels and convert the image back into RGB.
3. Write a function that computes the 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 H 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 H 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.
6. Please run the function you wrote on the three 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?