Assignment 1
Getting Familiar with Basic Image Operations in MATLAB

Please remember:

1. What you must hand in includes the assignment report (.pdf), source codes (.m) and output files (.png). Please insert each part in a different folder, and zip them all together into an archive file named according to the following template: HW1_XXXXXXXX.zip
   Where XXXXXXXX must be replaced with your student ID.
2. Your work will be evaluated mostly by the quality of your report. Don't forget to explain what you have done, and provide enough discussions which proves you have realized the subject.
3. Your codes must be separated for each question, and for each part. For example, you have to create a separate .m file for part b. of question 3. Please name it like p3b.m.
4. Some problems include some "Keywords" in order to help you find useful information about that problem.
5. Using MATLAB image processing built-in functions is not allowed, except for simple operations like reading, displaying, converting and saving images, or in cases it is clearly allowed to.
6. **Please upload your work in Moodle, before the end of March 10th.**
7. If there is ***any*** question, please don't hesitate to contact me through the following email address: **ali.the.special@gmail.com**
   I'd be glad to help.
8. Unfortunately, it is quite easy to detect copy-pasted or even structurally similar works, no matter being copied from another student or internet sources. Try to send me your own work, without being worried about the grade! ;)

1. Understanding and Working with Images as Matrices

In MATLAB, images are stored as matrices, with intensity values as their elements. Here, you are going to do some basic matrix operations to create and modify images. Default image size in this problem is 400x400.

a. Create a random 8-bit image. Display it using MATLAB function `imshow()`. Use `imwrite()` to save this output and all the remaining problems results later on.
   Hint: Beware of datatype issues. In MATLAB, 8-bit images are stored as 1-byte unsigned integers, i.e. "uint8".

b. Create and display a black horizontal line with the width of 100 pixels, centred in a white background.

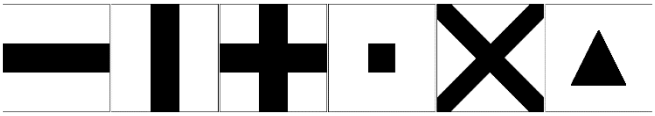

Figure 1 From left to right, expected outputs of part b. to part g.

c. Create and display a black vertical line with the width of 100 pixels, centred in a white background.

d. Create and display a black cross of the width 100 pixels in each direction, centred in a white background.
e. Create and display a black 100x100 rectangle, centred in a white background.
f. Create and display a black X mark of the width about 100 pixels in each diagonal line, centred in a white background.
g. Create and display a black equilateral triangle with the base of 100 pixels, centred in a white background.
h. Implement a function to get binary images (like the above results) as input and change white pixels to black pixels and vice versa. Apply this function to part b-g outputs and display the results.
i. Some of the above outputs can be obtained by applying simple matrix operations on the others. Which outputs are they, and what are those operations? Obtain them using the new way.

2. Fundamental Image Operations in MATLAB

In this problem, you are going to get familiar with basic functions of MATLAB image processing toolbox.

a. Read the file "trump.jpg" from P2 directory in inputs folder, using `imread()` function. Let's call it T1. Display it.
b. Convert T1 to grayscale by `rgb2gray()` function. Let's call it T2. Display the result.
c. Extract a 50x50 rectangle of T2, starting in position (170, 370), using `imcrop()` function. Display the result.
   Hint: Note the difference between Cartesian coordinate system and image coordinate system.
d. Rotate T2 by 90, 180 and 270 degrees, using `imrotate()` function. Display the results.
e. Resize T2 by scaling factor of 0.5 and 2, using `imresize()` function. Display the results.
f. Find the horizontal and vertical mirror version of T2 using `flipdim()` function. Let's call the horizontal flipped version T3.
g. Create a new image, consisting of T2 and T3, side by side each other ([T2,T3]).
h. Implement your own version of `rgb2gray()`. Apply it on T1 and compare the result with part b.
i. Implement your own version of `imcrop()`. Apply it on T2 to crop a 50x50 rectangle, starting in position (170, 370), and compare the result with part c.

Figure 2 Horizontal flipped version of T1, named T3

j. Implement your own simplified version of `imrotate()`, which can rotate input image by 90, 180 and 270 degrees. Apply it on T2 and compare the results with part d.
k. Implement your own simplified version of `imresize()`, which can resize input image by the scaling factor of 0.5 and 2. Apply it on T2 and compare the results with part e.
l. Implement your own version of `flipdim()`, which can mirror image both horizontally and vertically. Apply it on T2 and compare the results with part f.
   Hint: One way to compare two images is to use MATLAB function `isequal()`. You are free to use your own method as well.

3. Working with Intensity Values by Image Thresholding

Keywords: *Image Thresholding*, *Image Histogram*

Image Thresholding is a simple way of partitioning image into two parts or more. It is usually done by replacing each pixel in an image with a black pixel if the pixel intensity value is less than some fixed constant, i.e. threshold, or a white pixel if the pixel intensity value is greater than that constant.

a. Implement a function which takes an image and a constant as the input, and applies image thresholding using that constant as the threshold value. Use "kitten.jpg" in P3 directory as its input, and display the results using three arbitrary values for threshold.
b. Display "kitten.jpg" histogram using MATLAB function `imhist()`.
c. Can you suggest a proper way for determining a good threshold value using image histogram? If yes, apply image thresholding on the input image with the new threshold value, and display the result.
d. Implement your own version of `imhist()`. Apply it on the input image and compare the result with part b.
e. (Additional point) **Balanced histogram thresholding** is a very simple, yet effective histogram-based thresholding method. Find out how it works, and implement it. Apply it on the input image and compare the result with part a. and part c.

Figure 3 An example of image thresholding

4. Image Decryption by Image Arithmetic and Logical Operations

Keywords: *Image Cryptography, Steganography, Image Arithmetic Operations, Image Logical Operations*

Image Cryptography is a technique which allows information to be encrypted in an image. If this information is some kind of secret messages, it is called **Steganography**. Ideally, anyone who scan the steganographed data will fail to know it contains a hidden message.

In this problem, you are going to decrypt very simple steganographed images.

a. Read the images "source.png" and "encrypted.png" from P4 directory in inputs folder. Display them side by side each other in a single figure. Can you notice any difference?
b. Compare the source and encrypted image using MATLAB function `isequal()`. Are they the same?
c. Using basic image operations, reveal the hidden message in a clear binary image. Display the result.
d. Let's try a more complicated scenario of steganography. Read images "salam1.png" and "salam2.png". Can you reveal the message hidden in these two images?

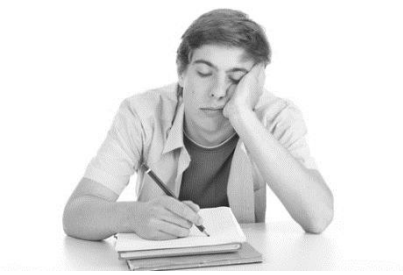

Figure 4 This image contains a big, yet invisible, secret message!

e. Do the same for “leo1.png” and “leo2.png”.
Note: Your results in part d. and part e. should be as clear as possible. Noisy results are not acceptable, so try to find the best operation.
Hint: You can use logical operations as well.

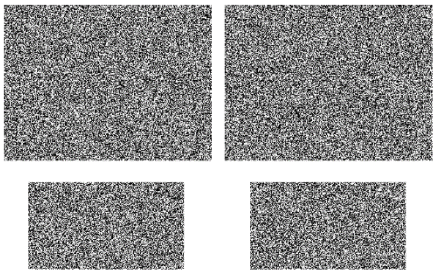

Figure 5 These images might look meaningless, but there are some secret messages hidden in them!

5. Tampering Detection using Image Subtraction

Keywords: *Image Tampering, Image Subtraction*, *Image Alignment*

Image Tampering is the process of manipulating images without leaving obvious visual traces of having been tampered. Modern digital technology and the availability of increasingly powerful image processing tools has made it so easy to change an image details without noticeable trace, making **Tampering Detection** one of the most active areas in the field of Image Processing.

When the reference image is available, one simple way to detect a tampered image is by subtracting the tampered image from the reference image. Image alignment may be required before the subtraction.

Figure 6 Example of a tampered image

Figure 7 Reference (left) and tampered (right) image of Mona Lisa

a. Read the file “monalisa-fake.png” from P5 directory of the inputs folder. Using its reference image, “monalisa.png” from the same folder, discover the name of the forger! Display it in a binary image.

b. Read the following images from P5 directory:
“irises-reference.jpg”
“irises-tampered.jpg”
“the_starry_night-reference.jpg”
“the_starry_night-tampered.jpg”

For each painting 1 and 2, submit a binary image where the tampered regions are marked by white pixels and the non-tampered regions are marked by black pixels. Note that this problem is a little more complicated than the prior one, as it needs some preprocessing steps.

Figure 8 Reference (on the left side of each) and tampered (on the right side of each) paintings of part b.

6. Basic Matrix Modifications for Image Splicing using MATLAB

Keywords: *Image Thresholding, Image Arithmetic operations, Image Logical Operations*

Image Splicing is the process of cropping a region from an image, and pasting it to the same or another image. Powerful image processing tools like Photoshop has made it so hard to detect whether an image is modified or not. **Splicing Detection** is an area in image processing which deals with these problems.

Although Photoshop is very popular for doing it, one can also does simple image splicing using MATLAB. Please read "surfer.jpg" from P6 directory. As can be seen, the background is white, making it very easy to detect boundary areas of the lying man image. Please paste it onto the image "sea.jpg", so that it looks real. You may need some preprocessing operations like scaling beforehand.

Note that your output must be in RGB (colourful) format.

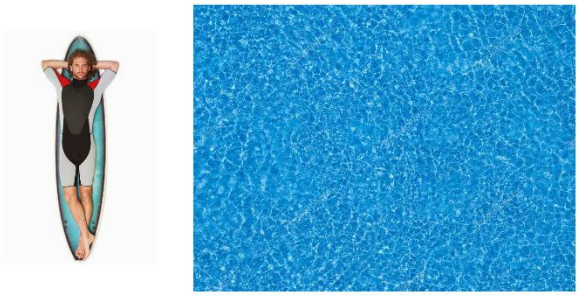

Figure 9 Surfer and the Sea, the input images of this problem

7. Extracting Relevant Colors from an Image

Keywords: *Color Quantization, Color Histogram, Image Thresholding, Clustering*

Image Segmentation is the process of partitioning an image into several segments, i.e. sets of pixels known as super-pixels. Segmented image is the simplified version of the image and could be more meaningful and easier to analyse, as can be seen in figure 10.

Figure 10 Example of Image Segmentation

Most image segmentation methods are based on color information of image pixels. One primary step of these methods is to find a set of most relevant colors of an image, or an image "palette", and assign each area of the image to a color from this set. Finding this set is also called **Color Quantization**. Figure 11. Shows an example of an image and its dominant color and palette colors set.

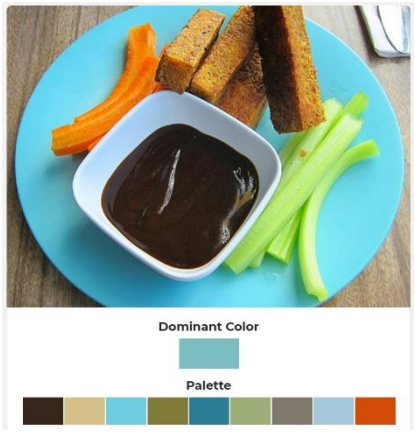

Figure 11 Finding dominant color and palette sets of an example image

Your task is to do something similar in MATLAB, i.e. to write a function which finds the most frequent colors of an input image. Your function input is the image and a p_size parameter which determines the size of the palette, and its output is a set of 100x100 boxes, each with a specific color.

a. Read the image "nature.jpg" from P7 directory in inputs folder, and find its dominant color and palette set. Set the palette size to the following values: 3, 5. 7.

b. Read the image "noucamp.jpg" from the mentioned directory, and find its dominant color and palette set. Set the palette size to the following values: 5, 7, 9.

c. (Additional point) Apply image segmentation to both images using part a. and part b. results.

8. Some Explanatory Questions

Please answer the following questions as clear as possible:

a. In at least 10 and at most 20 sentences, explain the procedure of taking a picture in a digital camera.
b. What is “Image quality”? What do we mean when there is two images of the same scene, and we say “Image 1 is better than Image 2”? Is there any numerical measurements, or it is all about qualitative measurements? Please explain.
c. We have two images, Image 1 and Image 2, of the same scene. Image 1 is of the size 400x300, while Image 2 is of the size 1600x1200. What can you say about their quality? Explain your answer.
d. We have two images, Image 1 and Image 2, of the same scene. Image 1 color depth is 8-bit, while Image 2 color depth is 24-bit. What can you say about their quality? Explain your answer.

Good Luck!
Ali Abbasi