EE371Q Digital Image Processing

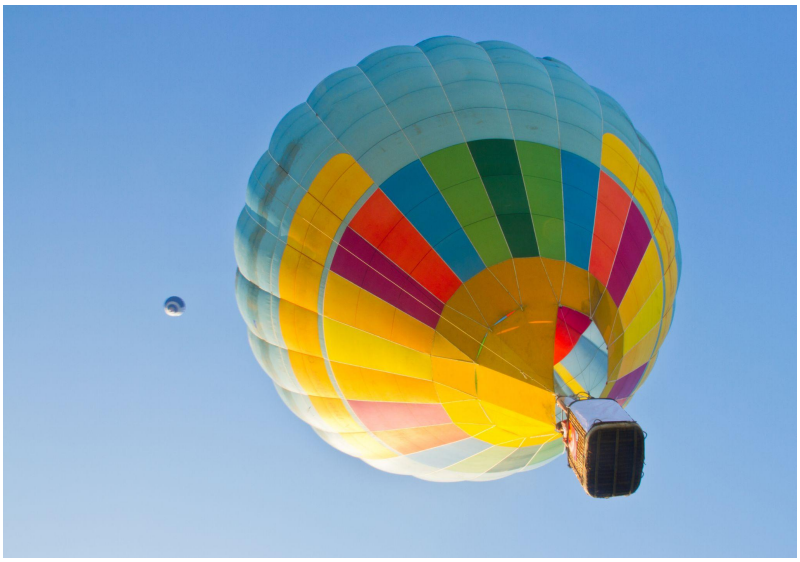
Homework #1

- This homework is due two weeks from now, on **September 18th at 11:59pm**.
- You are encouraged to work with other students to understand the course material. However, sharing source code, images, or other answers from any homework is strictly forbidden. You are to submit your own work.
- Please include your written answer and/or output images for each problem.
- The instructions are written with Matlab and Python in mind, but feel free to use any programming language for this assignment.
- Please show all steps for full credit and attach the code to each problem.
- **All the images needed for this homework can be downloaded as .jpg files from Canvas under the HW1 folder in the Files tab.**
- The homework should be formatted as a PDF file. While submitting on Canvas, zip the PDF file with the images you generated, and submit as a single zipped file.

Problem 1 - Basic Image Processing (30 points)

In this problem, you will be performing a few basic operations on images. You will be working on the image given below, which can be downloaded from Files -> HW1 -> Images tab in Canvas.

For this problem, use a new color image named "balloons.jpg," which can be downloaded from the Files tab on Canvas.

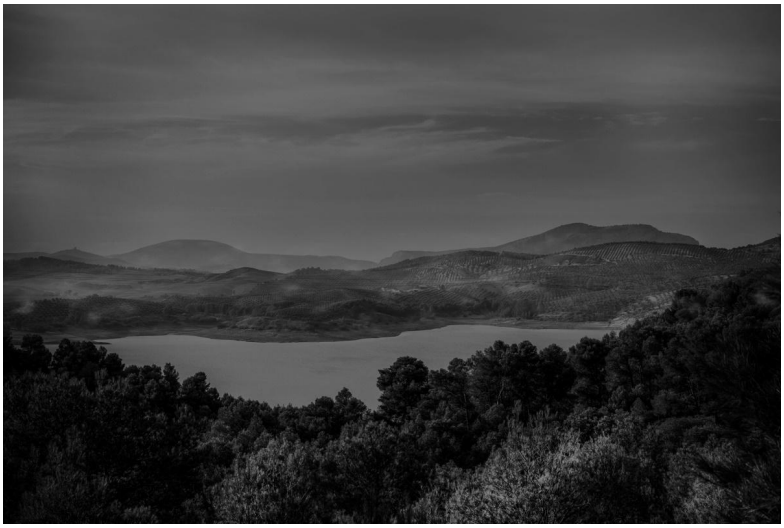


balloons.jpg

a. (1 point) Read the "balloons.jpg" image and display it in a new window.
b. (3 points) Extract the Red, Green, and Blue color channels from the image and show these three color component images along with the original image in a 2x2 grid with labels.
c. (5 points) YUV is an older color space described in Module - 1, with the equations to compute Y, U and V from RGB. Convert the original image to YUV format, i.e. generate three images of Y, U and V components separately by using the converting equation. Display all three of these images side-by-side.
d. (5 points) Convert the original image to grayscale image. Write your own function to create the grayscale from the color image. Create 4 different images each with different color intensity weightings, one of which should have equal color intensity weighting for R, G and B. Display these 4 images in a 2x2 grid with labels.
e. (4 points) Rotate the **top left quarter** of the image by 180 degrees. You cannot use the built-in function ***imrotate*** or equivalents***.*** Display the updated image in a new window.
f. (4 points) Flip the original image horizontally (along the central vertical axis) and vertically. Display the original image and the two flipped images in a 1x3 grid with labels. You cannot use the built-in functions ***fliplr and flipud*** or equivalents.
g. (4 points) Implement a color filter that emphasizes a specific color (e.g., blue, red, or green) in the image. Apply the filter and display the result.
h. (4 points) Apply a custom color enhancement technique of your choice to enhance the visual appeal of the image. Explain the technique and its effects. Display the enhanced color image and provide a brief description of the enhancement.

Problem 2 - Histogram Equalization and Contrast Enhancement (20 points)

In this problem, you will work on generating the histogram of an image, and performing operations on the histogram to enhance the contrast of the image. Do not use the built-in functions imhist, imadjust, or histeq functions. You should create your own histogram manipulation functions.



q2.jpg

a. (2 points) Read the image in grayscale. Compute and plot the histogram of the image and display it in a new window with a label. You are not allowed to use the imhist inbuilt function. Is the image underexposed or overexposed?
b. (5 points) Implement the full scale contrast stretch (FSCS) algorithm. Write a function called "contrast_stretching(image)" that takes the original grayscale image as input and returns the enhanced image. Apply this function to the "q2.jpg" image
c. (5 points) You need to now implement the logarithmic contrast compression. Write a function “log_contrast_compression(image)”, which takes the input as the original image and gives output as the logarithmic contrast compressed image. Take q2.jpg, pass it through “log_contrast_compression(image)”, and pass the output of this function through “contrast_stretching(image)” (FSCS algorithm). Display the logarithmic contrast compressed image, its histogram, the final image after passing through both the functions and its histogram in a 2x2 grid with labels.
d. (5 points) Implement the Gamma Correction algorithm. Write a function called "gamma_correction(image, gamma)" that takes the original grayscale image and a gamma value as inputs and returns the gamma-corrected image. Experiment with different gamma values to find one that works well for the image. Display the processed image and mention the value of 'gamma' used.
e. (3 points) Display the original "q2.jpg" image and the three enhanced images obtained from Contrast Stretching, Logarithmic Range Compression, and Gamma Correction in a 2x2 grid with labels. Analyze the visual quality of the images and determine which

enhancement technique performs best for the "q2.jpg" image. Provide a brief explanation of your choice.

Problem 3 - Binary Image Morphology (30 points)

We are going to apply morphological operators on the two images given below, and analyze the effects of these operations.


stars.jpg and *fingerprint.jpg*

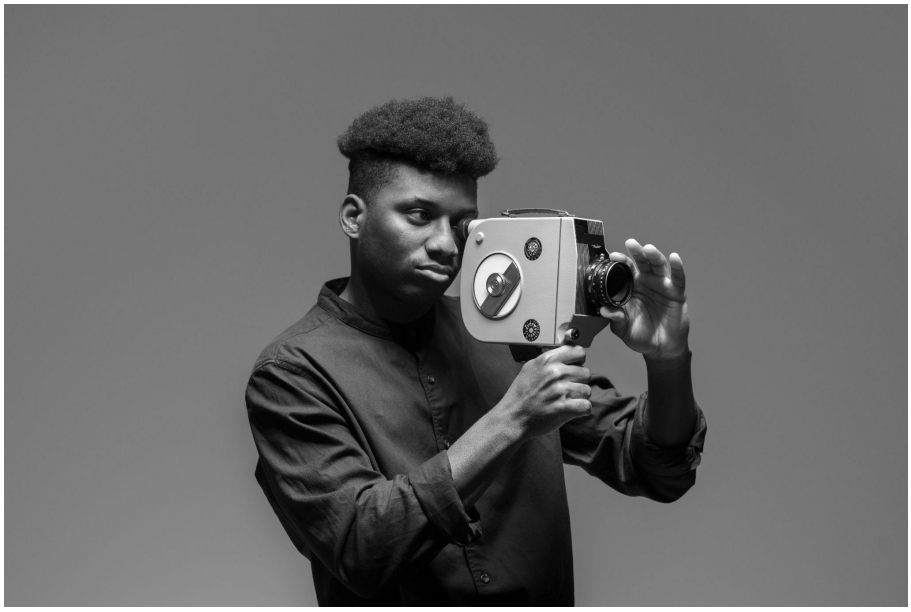
a. (5 points) Read both the images in grayscale. Before running morphological operations, binarize the images by applying thresholding with an appropriate threshold value. Display the two thresholded images side by side. Explain which image thresholding is more effective and why. For the subsequent parts of this question, you will be working with these thresholded images, not the original images.
b. (10 points) Implement two functions: "custom_dilate()" and "custom_erode()." These functions should take the binary image and the size of a square structuring element as inputs and return the dilated and eroded images, respectively. You are not allowed to use loops in your implementations. Helpful functions: pad, im2col, reshape, numpy.lib.stride_tricks.sliding_window_view(), Display the dilated and eroded images of both "stars.jpg" and "fingerprint.jpg" (thresholded images) in a 2x2 grid with labels.
c. (5 points) Modify your "custom_dilate()" and "custom_erode()" functions to implement five additional morphological operations: opening, closing, open-close, close-open, and median filtering. Run "fingerprint.jpg" (thresholded) through each of these five functions and display the five output images in a 1x5 grid with labels.
d. (5 points) Generate a clean binary image from the thresholded "fingerprint.jpg" image by removing small isolated regions and filling holes. Use a combination of morphological operations such as erosion, dilation, opening, or closing. Create a boundary image of the

cleaned object. Display the boundary image and the thresholded "fingerprint.jpg" image side by side.

e. (5 points) Write a function called "boundary_length()" to count the white pixels in the boundary image in "fingerprint.jpg." Report the calculated boundary length.

Problem 4 - Bit-Plane Manipulation (15 points)

In this problem, you will explore bit-plane manipulation and the addition of hidden information to an image. Download the image "q4.jpg" from the Files tab on Canvas for this problem.



q4.jpg

a. (5 points) Read the image "q4.jpg." Modify it using a 3-bit quantization bar. Essentially, represent the intensity of each pixel using only 3 bits instead of the original 8 bits. Display the 3-bit quantized image and the original image side by side.

b. (5 points) Use the bitget function to extract each of the 8 bit-planes from the 3-bit quantized image. Display the 8 bit-planes in a 2x4 grid with appropriate labels. Helpful functions in python: numpy.unpackbits(), numpy.packbits()

c. (5 points) Experiment with hiding information in the bit-planes. Start by adding hidden information in bit-plane 1 and reconstructing the image. Then, generate seven more images by adding hidden information in bit-plane 2, bit-plane 3, and so on, up to bit-plane 8. Display all eight images with "hidden information" in a 2x4 grid with proper labels.

After generating these images, answer the following questions:

- Are any of these images perceptually distorted compared to the original 3-bit quantized image?
- If any are distorted, which ones, and what kind of distortion do you observe?

Problem 5 - Perceptron in Image Processing (5 points)

Consider a simple binary image processing task where you want to classify whether an image contains a specific object (e.g., a circle) or not. You decide to use a single-layer perceptron for this task.

a. (3 points) Briefly explain what a perceptron is and how it can be applied to image processing tasks like the one described above.
b. (2 points) What are the key components and parameters you would need to define when implementing a perceptron for this image classification task? Provide a concise list of these components and parameters.