MACM 203 Assignment 9
Spring 2025

This assignment is due Tuesday April 1st at 10pm. Upload your solutions to Crowdmark. Write your solutions as a single Matlab Live Script and export the script to PDF. Write the course number and assignment number as the title of the Matlab Live Script, followed by the table of contents, and then create a section for each part of the question.

Keep in mind that your assignment, including the source code, is a document that will be read in order to be marked. It has to be very clear and properly formatted.

Your Matlab code must be general enough to solve any other instance of the same problem *without modification.*

Assignments must be written individually. You can discuss in groups, but you have to write your assignment yourself. In case of academic dishonesty SFU policies will be applied.

Preamble

This week's assignment focuses on Information Hiding. Read the course notes for this week in detail before you start working on the assignment.

Question 1 (15 marks)

Part (a)

This part reviews indexing of two-dimensional matrices *by just one index.* Create a matrix `T` that has two rows, three columns and its entries are random numbers. Display the entire matrix. Then display the values `T(1)`, `T(2)`, `T(3)`, `T(4)`, `T(5)`, `T(6)`.

You will note that in this indexing, the linear ordering of the matrix entries is by columns from left to right, and within each column it is from top to bottom. In fact the actual linear ordering of the matrix entries is immaterial; the important part is that the ordering is uniquely defined for each matrix, and it can be obtained easily in Matlab.

Part (b)

Implement message extraction as described in section 2.2 of this week's lecture notes. Write a Matlab function `extract` that accepts on input a grayscale image with uint8 entries and it returns the extracted message consisting of LSBs of the image pixels, under the ordering of pixels practised in part (a) above. That is, if the image has m rows and n columns, then the function returns a vector with mn entries, each of which is 0 or 1.

Part (c)

Implement secret message embedding as described in section 2.3 of this week's lecture notes. Write a Matlab function `embed` that accepts on input a grayscale image with uint8 entries (which will serve as the cover object), and a secret message which is a vector whose entries are integers 0 or 1. The length of the secret message is the same as the number of pixels in the image. The function returns a distorted image such that the message extracted from the distorted image is the same as the secret message that is passed as the second argument to your `embed` function.

Part (d)

Use Google to find some grayscale JPEG image that may work well as a cover object, and download the image. Use the word grayscale in the search. Randomize your search by using for example geographic names (cities, countries) etc. It is not expected that the image that you end up using would be the same as the image used by anyone else on this assignment. The image should be at least 500×500 pixels; it need not be too big or else your computations may be slow. Google allows you to narrow down your search by the approximate size of the image.

Use `imread` to load the image into Matlab and confirm that it is grayscale and the bit depth is 8. If this is not the case, find another image.

Part (e)

Generate a "secret message" as a vector whose entries are integers 0, 1 (generated uniformly at random) and whose length is the same as the number of pixels in the image that you obtained in part (d). You can think that this secret message is a ciphertext obtained by some block cipher as discussed last week in the Cryptography lecture. Ciphertexts do look like random bit vectors, hence this is a plausible setup. (In fact ciphertexts are *required* to be statistically indistinguishable from random bit vectors.)

Avoid hardcoding the image dimensions; write a general code. Use your function `embed` written in part (c) to embed this secret message into the image. Display the original image and the distorted image carrying the secret message. Remember to give titles to the images.

Do you feel that the distortion caused by embedding the secret message is visually perceptible by a casual human observer? Remember that only the distorted image is actually communicated; the original image (before distortion) is known to the sender but to nobody else. If you feel that the distorted image looks too strange, then try to find a better cover object in part (d) and repeat the process.

Part (f)

Use your function `extract` written in part (b) to extract the secret message from the distorted cover object obtained in part (e). Check that the extracted message indeed is

the same as the secret message that you embedded into the cover object in part (e).

Throughout the assignment *avoid* printing any large matrices or vectors; instead use more clever methods to find the answers. For example, to check if all entries of some 0,1-vector are 1s, add all entries and compare the sum to the length of the vector.

Comment on all parts of your work. Explain your code. If your code is not sufficiently explained, a dishonesty case may be opened with you to determine whether you wrote the solution yourself.