MACM 203 Assignment 4
Spring 2025

This assignment is due Tuesday February 25th 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*. That is, if you are asked to run your code on some image that is specified in the assignment, then your code must run on any other image without modification.

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

Preamble

This week's assignment focuses on image processing and writing Matlab functions.

Question 1 (20 marks)

Part (a)

A vignette image is an image with an artistic effect where the brightness decreases gradually from the centre towards all four edges, creating a subtle darkening or fading effect. This technique draws focus to the centre of the image, enhancing its visual appeal and depth. Vignetting can be done in many ways by varying the intensities of the darkening (fading) effect. Below is an example of an image (on the left) and a vignette version of this image (on the right).

Download the image `vase.jpg` from Canvas. Implement your own Matlab algorithm to construct a vignette image from the downloaded image. You should avoid those for loops

that can be avoided easily, but you can use for loops if you believe that avoiding them results in code that is too hard to read. You can comment on this issue in your code.

Display both the original and vignette images for comparison. Give an explanation of your approach and the observations that you made during the process.

Part (b)

Write Matlab function `Rot90` that rotates an image by 90° clockwise, and write Matlab function `Rot180` that rotates an image by 180°. The image may be colour **or** grayscale. Your functions should **not** display anything; rather they should return the arrays containing the rotated images.

Your code must be written from scratch; the only allowed operations to perform the rotation are array indexing, array transposition, and the built-in function `flip` (see documentation). You must not use any for loops or while loops. You are **not** required to pre-allocate the array that will store the rotated image before you start the computation.

Hint: Draw a rectangle on a piece of paper and label its four corners A,B,C,D. Observe how the operations of rotation, transposition, and `flip` (two cases) operate on the four corners of the rectangle. If you get the four corners positioned correctly, then the rest of the image will fall into place too.

Part (c)

Read the image `sherlock.jpg` from Matlab's bank of images. Display the image. Apply your functions `Rot90` and `Rot180` to this image and display the output of your functions, that is, the two rotated images.

Part (d)

Read the image `logo.tif` from Matlab's bank of images. Display the image. Apply your functions `Rot90` and `Rot180` to this image and display the output of your functions, that is, the two rotated images.