ECE462 Multimedia Systems

Laboratory Assignment 1

Preparation Questions

Name: Mathura Shivakaran

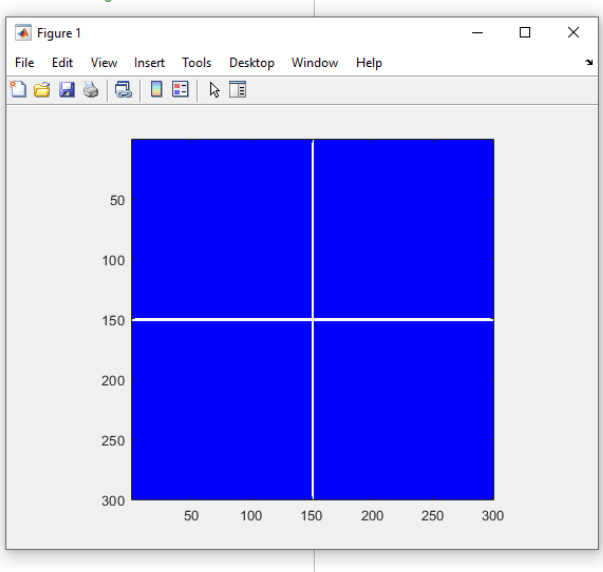
Student Number: 1006879184

(Q1) 2 Marks

Write MATLAB code to create a 300×300 pure blue image (i.e. all pixels have the RGB values (0,0,255)) and draw a white cross (+) in the middle of the image, i.e. two perpendicular lines running the entire length of the image (300 pixels), each 2 pixels wide. You do not need to display the image. Your code shouldn't be more than *four lines* and *you shouldn't use any loops*. (Hint: look at the ":" operator explained in Section 6 of the reading material.)

```
% Creating blue image with white cross
imageBlue = zeros(300, 300, 3);          % Initializing a 300 x 300 matrix for RGB image
imageBlue(:, :, 3) = 255;                % Setting the B channel only to 255 throughout whole image
imageBlue(:, 150:151, 1:3) = 255;        % Setting RGB channels all to 255 for two columns across all rows for white
imageBlue(150:151, :, 1:3) = 255;        % Setting RGB channels all to 255 for two rows across all columns for white

% Displaying the created image
imageBlue = uint8(imageBlue);
image(imageBlue);
axis image;
```



We start by initializing a 300x300 matrix full of zeroes. We then only set the third dimension (blue colour channel) to maximum intensity. This causes the colour channel to be (0, 0, 255), a pure blue image. In the third line, we are drawing the vertical white line in the middle (column pixels 150 and 151) across all rows. In the fourth line, we are drawing the horizontal white line to finish the cross in the middle (row pixels 150 and 151) across all columns. To draw the white lines, we use the ":" operator to manipulate all three dimensions, as white has a colour code of (255, 255, 255). This is completed in four lines. The rest of lines are for the purpose of displaying.

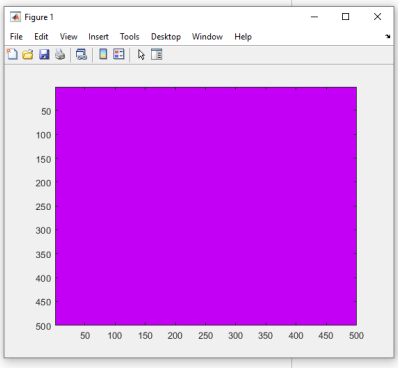
(Q2) 2 Marks

An image in the RGB color space is created and stored in the MATLAB variable `A` of type `double`. Using two lines of code, set the green intensity of `A` to zero and display the image.

```
% Initializing a 500x500 matrix for the image
A = zeros(500, 500, 3);

% Creating a random colour (pastel purple)
A(:, :, 1) = 195;
A(:, :, 2) = 177;
A(:, :, 3) = 245;

% Setting the green intensity to 0 and displaying in two lines of code
A(:, :, 2) = 0;
image(uint8(A));
```



We start by creating a random matrix A that displays a pastel purple image. In the first line of answer code, we set only dimension 2 to have all of its element equal to zero. This causes the second channel (green colour channel) to have an intensity of 0. In the second line of answer code, we convert the matrix into the uint8 data type for accuracy and display it in one line.

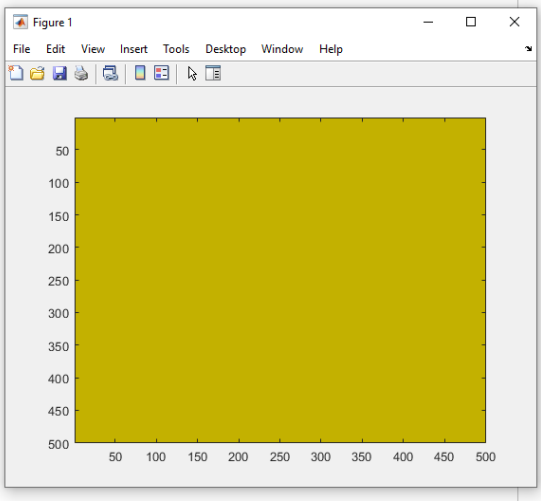
(Q3) 2 Marks

An image in the RGB color space is created and stored in the MATLAB variable `A` of type `double`. Using two lines of code, subtract from the blue component its mean (of the blue-component), and display the image.

```
% Initializing a 500x500 matrix for the image
A = zeros(500, 500, 3);

% Creating a random colour (pastel purple)
A(:, :, 1) = 195;
A(:, :, 2) = 177;
A(:, :, 3) = 245;

% Subtracting average of blue channel from itself and displaying in two lines of code
A(:, :, 3) = A(:, :, 3) - mean(A(:, :, 3), 'all');
image(uint8(A));
```



We start by creating a random matrix A that displays a pastel purple image. In the first line of answer code, we use the "mean" operator in MATLAB which in our case, calculates the average of all the elements across dimension 3 (aka, the blue colour channel). We also use the keyword "all" to ensure it is applied to all elements in the dimension. We set the third channel (blue colour channel) to be equivalent to the CURRENT value of the blue channel subtract the average calculated. In the second line of answer code, we convert the matrix into the uint8 data type for accuracy and display it in one line.

(Q4) 2 Marks

Write the MATLAB code that draws a 1000×1000 image `A` like in Figure 3 and then displays then image. You should make an estimate of what the grey value should be.

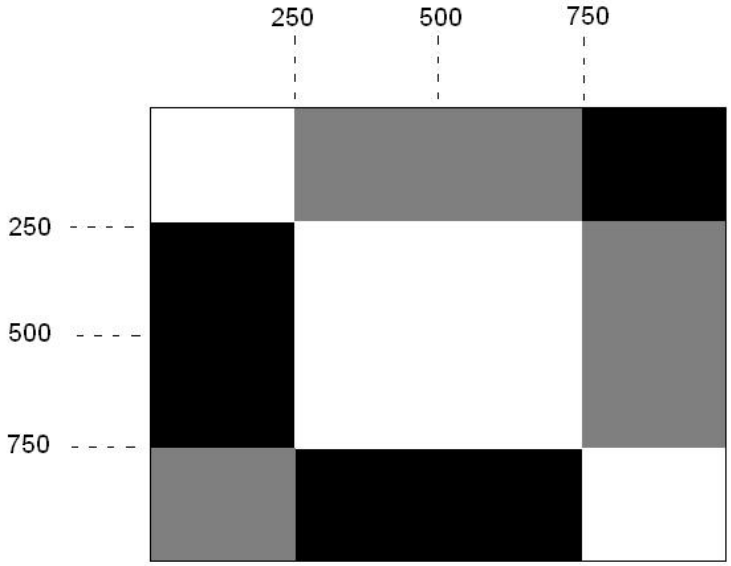


Figure 3: Image with squares and rectangles

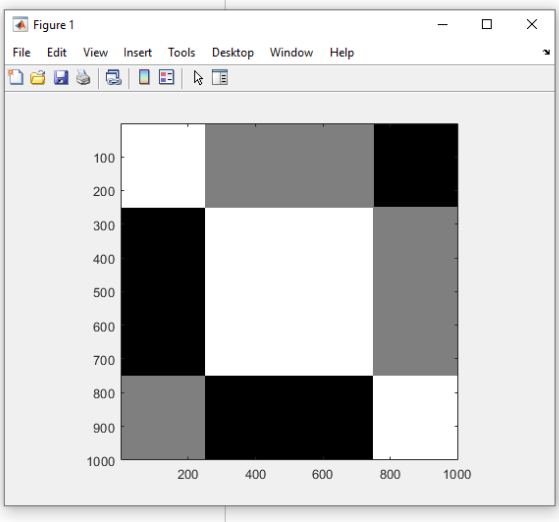
```
% Creating the monochrome image

% Initializing 1000x1000 image (already black)
imageGrey = zeros(1000,1000,3);

% Creating the blocks that are white according to pixel values
imageGrey(1:250, 1:250, 1:3) = 255;
imageGrey(250:750, 250:750, 1:3) = 255;
imageGrey(750:1000, 750:1000, 1:3) = 255;

% Creating the blocks that are grey (127, 127, 127) according to pixel
% values
imageGrey(1:250, 250:750, 1:3) = 127;
imageGrey(250:750, 750:1000, 1:3) = 127;
imageGrey(750:1000, 1:250, 1:3) = 127;

% Displaying the created image
imageGrey = uint8(imageGrey);
image(imageGrey);
axis image;
```



First, we initialize a 1000x1000 matrix with zeroes. This makes the whole image black already, so the black shapes are essentially already done. Then, for each colour, we identify the pixel coordinates and set the colour channels of the specified area accordingly (White = 255, 255, 255, Grey = 127, 127, 127). We then display the image.

(Q5) 2 Marks You are given a vector `S` of size N. Split `S` into two vectors `S1` and `S2` such that `S1` contains those elements of `S` that correspond to *even* indices i.e. `S`(2),`S`(4),`S`(6),... and `S2` all the *odd* ones. You should do that *without using any loops*.

```
% Vector S with N elements (Here, N = 6)
S = [11, 22, 33, 44, 55, 66];

% To find the even indices, we save every second element of the original
% vector S into S1, starting from the SECOND vector element to the last
S1 = S(2:2:end);       % Even Indices

% To find the odd indices, we save every second element of the original
% vector S into S2, starting from the FIRST vector element to the last
S2 = S(1:2:end);       % Odd Indices

% Printing the result
disp(S1);
disp(S2);
```

Command Window

New to MATLAB? See resources for Getting Started.

```
    22    44    66

    11    33    55
```

We first initialize a vector S and include some element values. To find the even indices, we use S1 = S(2:2:end). This essentially indirectly loops through the original vector S and saves every second element into S1. We start from the SECOND element and go till the last to ensure we are saving the even indices. To find the odd indices, we use S2 = S(1:2:end). This indirectly loops through S and saves every second element into S2. We start from the FIRST element and go till the last to ensure we are saving the odd indices. We then print S1 and S2 to the terminal.