INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, BHOPAL (IIIT-BHOPAL)

Department of Information Technology

Digital Image Processing (IT - 312)
Vth – Semester

Submitted to:
Dr. Bhupendra Singh Kirar
Assistant Professor, IT

Submitted By:
Aakash Borse
22U03028

INDEX

S. No.	Name of Assignment	Date	Submission Date	Page No.	Remarks
1.	Introduction to MATLAB Commands and Functions in Digital Image Processing	12 Aug 2024	12 Sep 2024	1	
2.	Writing Programs to Read and Display Images Objective	12 Aug 2024	12 Sep 2024	2	
3.	Writing Programs to Convert to Grayscale and Display Using Subplot	03 Sep 2024	12 Sep 2024	3	
4.	To write a program for histogram calculation and equalization	03 Sep 2024	12 Sep 2024	4-5	
5.	To write and execute programs for image arithmetic operations	12 Sep 2024	10 Oct 2024	6-8	
6.	To write and execute programs for image logical operations	12 Sep 2024	10 Oct 2024	9-10	

Experiment 1:
Introduction to MATLAB Commands and Functions in Digital Image Processing

MATLAB is a high-level programming environment commonly used for numerical computing and image processing. Here's a brief overview of some essential commands and functions used in MATLAB for image processing:

Reading an Image

- Command: imread
- Description: Reads an image file and stores it in a variable.
- Example:
  ```
  img = imread('abcd.jpg');
  ```

Displaying an Image

- Command: imshow
- Description: Displays an image in a new figure window.
- Example:
  ```
  imshow(img);
  ```

Converting to Grayscale

- Command: rgb2gray
- Description: Converts a color image to a grayscale image.
- Example:
  ```
  grayImg = rgb2gray(img);
  ```

Creating Subplots

- Command: subplot
- Description: Divides the figure window into a grid and allows you to place multiple plots in different sections.

- Example:
  ```
  subplot(1, 2, 1); % Creates a subplot in a 1x2 grid, first position
  imshow(img); title('Original Image');
  subplot(1, 2, 2); % Creates a subplot in a 1x2 grid, second position
  imshow(grayImg); title('Grayscale Image');
  ```

Experiment 2:
Writing Programs to Read and Display Images Objective

Reading an Image

```
img = imread('filename.jpg');
```

-imread: Reads an image from a file.

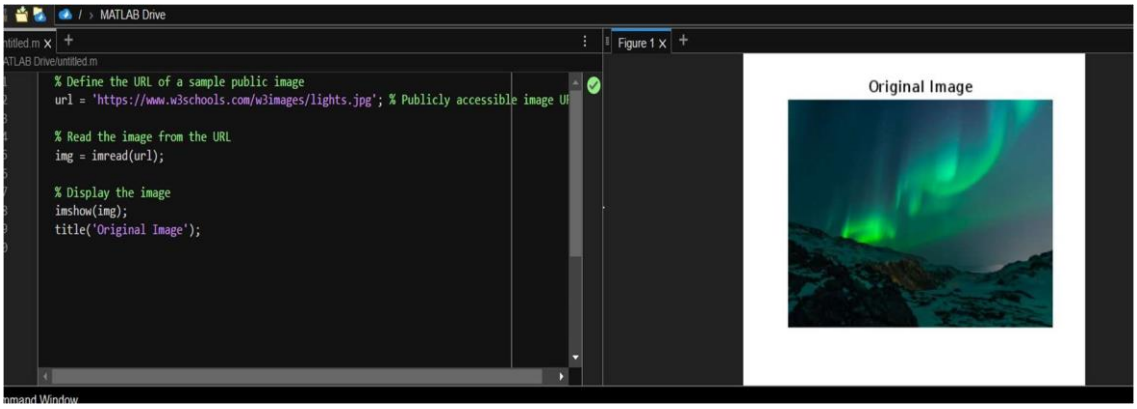
Displaying an Image

```
imshow(img);
```

-imshow: Displays the image in a window.

Example Program

```
img = imread('example.jpg');
imshow(img);
title('Original Image');
```



Experiment 3:
Writing Programs to Convert to Grayscale and Display Using Subplot

MATLAB Commands

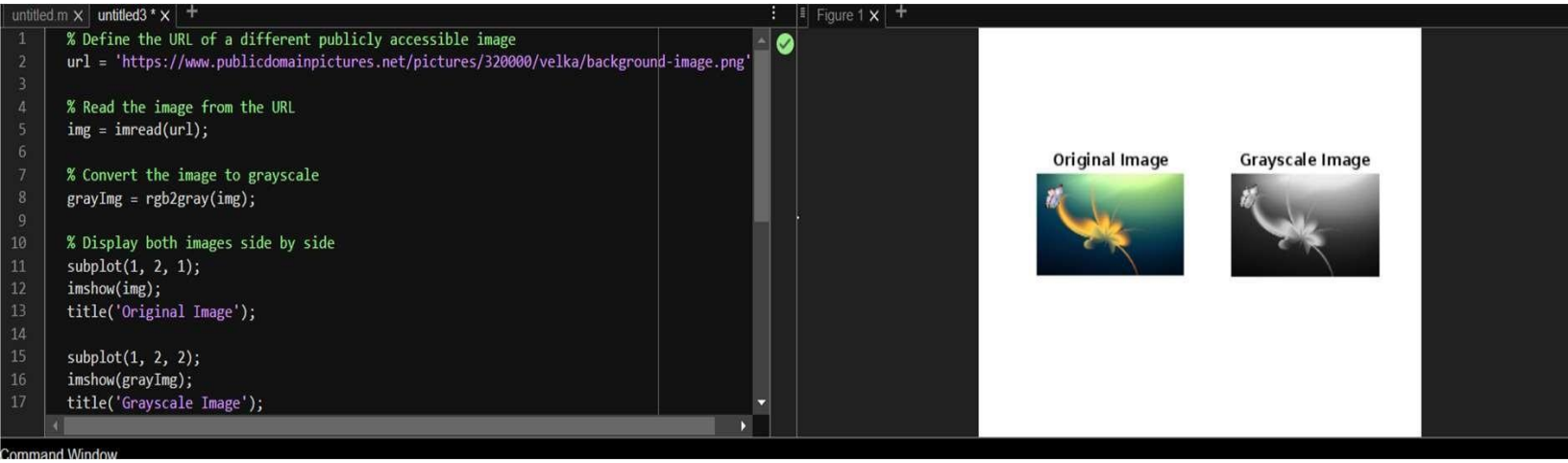
Convert to Grayscale grayImg
`rgb2gray(img);`
- rgb2gray: Converts a color image to grayscale.

Create Subplots
`subplot(m, n, p);`
- subplot: Divides the figure into a grid of subplots.

Displaying Images in Subplots
```
subplot(1, 2, 1); imshow(img); title('Original Image');
subplot(1, 2, 2); imshow(grayImg); title('Grayscale Image');
```

Example Program
```
url = 'https://www.publicdomainpictures.net/pictures/320000/velka/background-image.png';
img = imread(url); grayImg = rgb2gray(img);
subplot(1, 2, 1);
imshow(img); title('Original Image');
subplot(1, 2, 2); imshow(grayImg); title('Grayscale Image');
```

Conclusion
In this experiment, you have learned to:
Use MATLAB commands to read and display images.
Convert images to grayscale.
Utilize the subplot function to display multiple images in a single figure.
These basic skills form the foundation for more advanced image processing tasks in MATLAB.

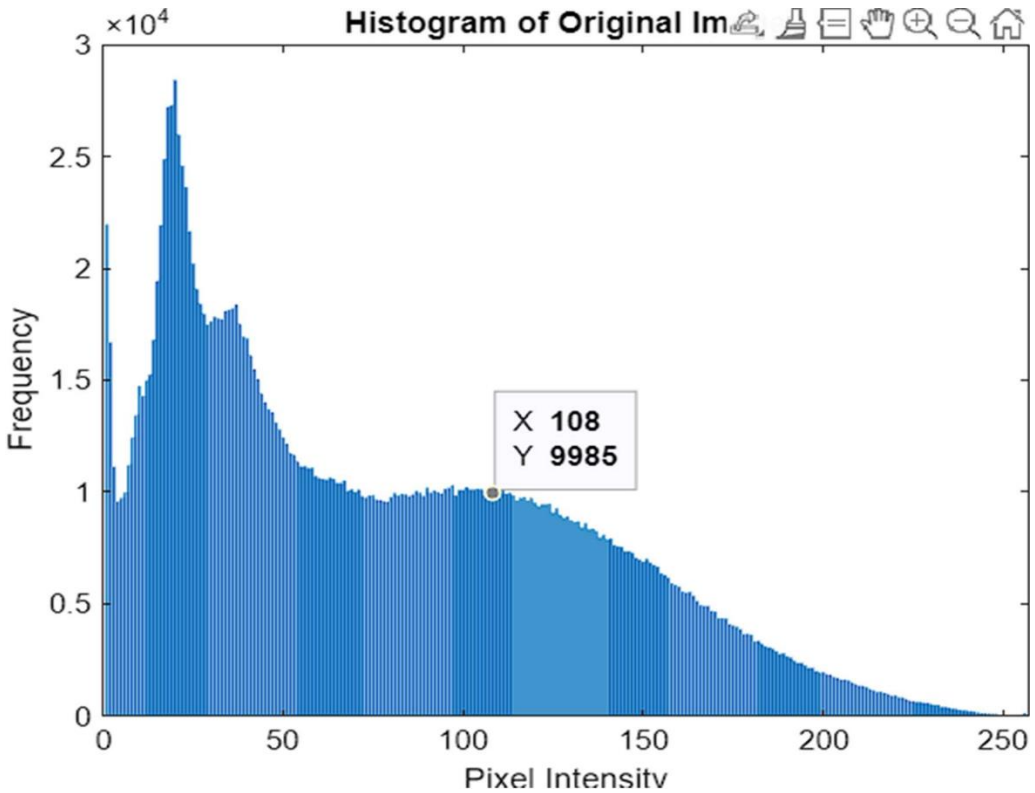


Experiment 4:

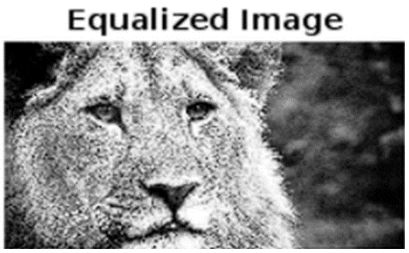
To write a program for histogram calculation and equilization

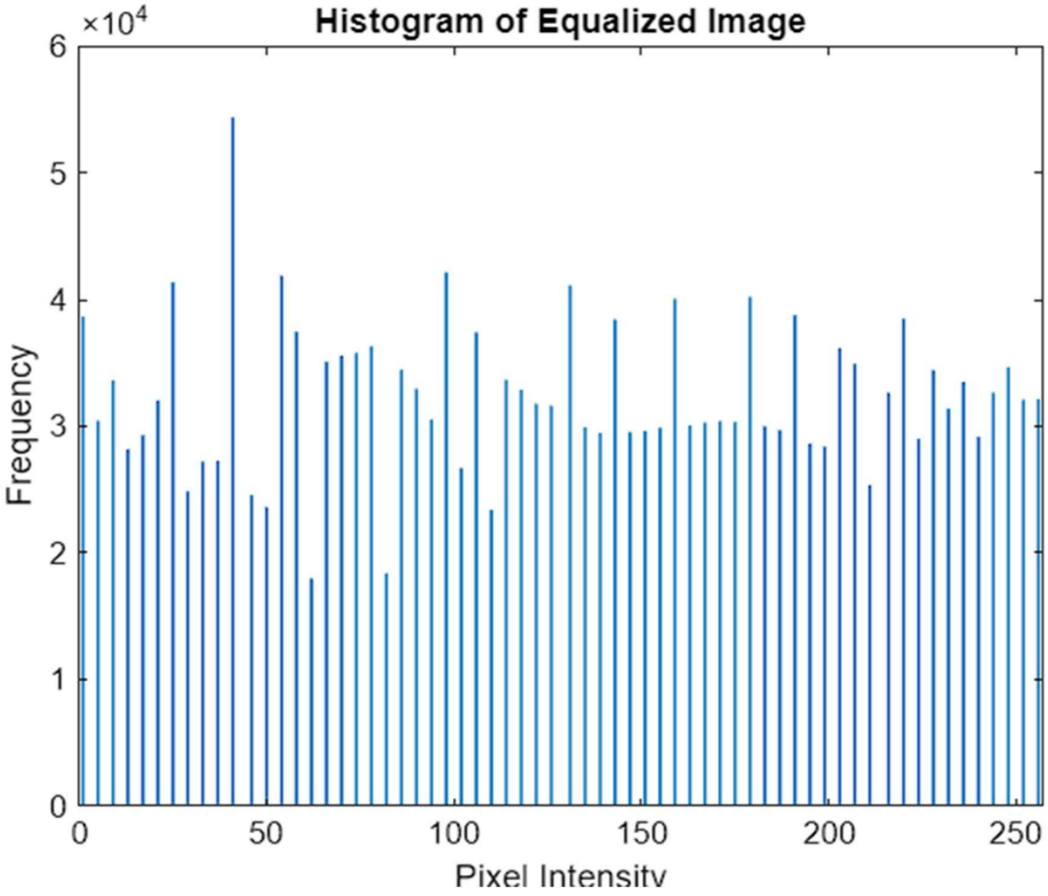
MATLAB code to calculate the histogram and perform histogram equalization on a grayscale image.

```
image = imread('https://wallpapercave.com/wp/wp3028175.jpg');
if size(image, 3) == 3
image = rgb2gray(image); end
figure; imshow(image);
title('Original Image');
hist_orig = imhist(image); figure;
bar(hist_orig);
title('Histogram of Original Image');
xlabel('Pixel Intensity');
ylabel('Frequency');
equalized_image = histeq(image);
figure; imshow(equalized_image);
title('Equalized Image');
hist_eq = imhist(equalized_image);
figure;
bar(hist_eq);
title('Histogram of Equalized Image');
xlabel('Pixel Intensity');
ylabel('Frequency');
figure; subplot(1, 2, 1);
imshow(image);
title('Original Image');
subplot(1, 2, 2);
imshow(equalized_image);
title('Equalized Image');
```


Histogram of Original Im
×10^4
Frequency
Pixel Intensity
X 108
Y 9985


Original Image


Equalized Image


Histogram of Equalized Image
×10^4
Frequency
Pixel Intensity

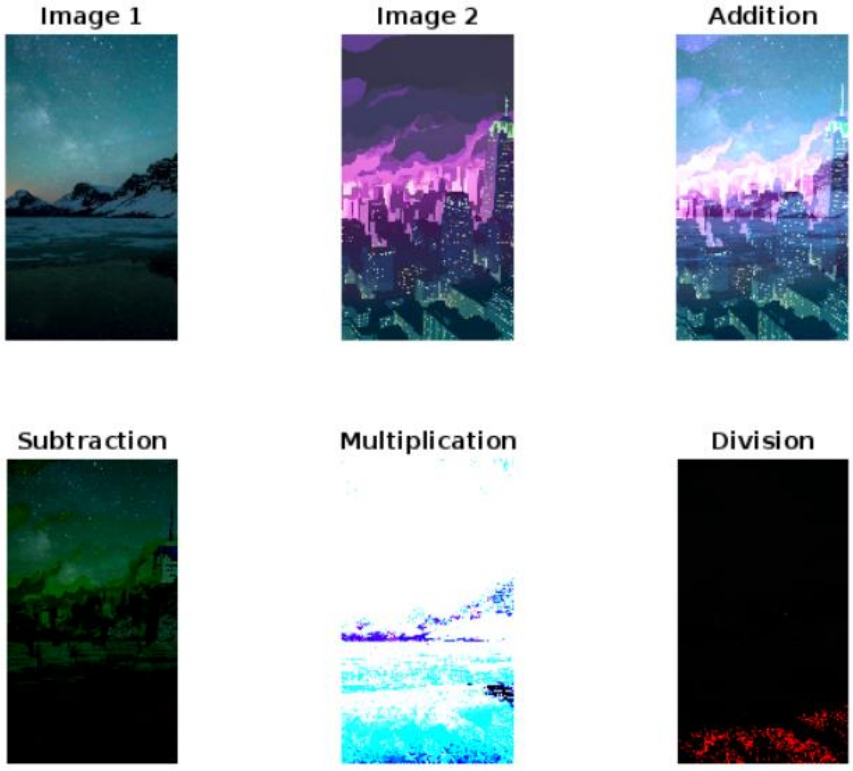
Experiment 5:

To write and execute programs for image arithmetic operations

MATLAB code for addition, subtraction, multiplication and division of two images.

```
img1 = imread('image1.jpg');
img2 = imread('image2.jpg');

img1 = double(img1);
img2 = double(img2);

%Addition
img_add = img1 + img2;
img_add = uint8(img_add);

% Subtraction
img_sub = img1 - img2;
img_sub = uint8(img_sub);

% Multiplication
img_mult = img1 .* img2;
img_mult = uint8(img_mult);

% Division
epsilon = 1e-6;
img_div = img1 ./ (img2 + epsilon);
img_div = uint8(img_div);

% Display the original and resulting images
figure;
subplot(2,3,1), imshow(uint8(img1)), title('Image 1');
subplot(2,3,2), imshow(uint8(img2)), title('Image 2');
subplot(2,3,3), imshow(img_add), title('Addition');
subplot(2,3,4), imshow(img_sub), title('Subtraction');
subplot(2,3,5), imshow(img_mult), title('Multiplication');
subplot(2,3,6), imshow(img_div), title('Division');
```

Output:

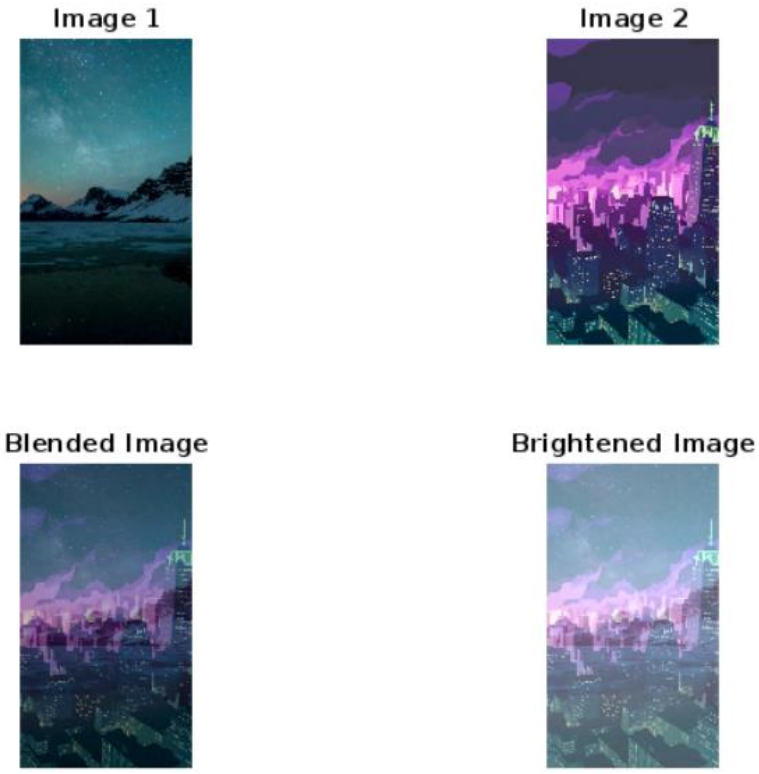


MATLAB code for executing image blending, calculating mean value and adjusting the brightness of the image by mean value.

```
img1 = imread('image1.jpg'); % Replace with your image file
img2 = imread('image2.jpg'); % Replace with your image file

img1 = double(img1);
img2 = double(img2);

% 1. Image Blending
alpha = 0.5;
blended_img = alpha * img1 + (1 - alpha) * img2;

blended_img = uint8(blended_img);

% 2. Calculating Mean Value of the Blended Image
mean_value = mean(blended_img(:));
fprintf('Mean pixel value of the blended image: %.2f\n',
mean_value);

% 3. Adjust brightness by changing the mean value
target_mean_value = 150;

current_mean_value = mean(blended_img(:));
```

```
brightness_adjustment_factor = target_mean_value -
current_mean_value;

brightened_img = blended_img + brightness_adjustment_factor;

brightened_img = min(max(brightened_img, 0), 255);

brightened_img = uint8(brightened_img);

% Display the original images, blended image, and brightened image
figure;
subplot(2,2,1), imshow(uint8(img1)), title('Image 1');
subplot(2,2,2), imshow(uint8(img2)), title('Image 2');
subplot(2,2,3), imshow(blended_img), title('Blended Image');
subplot(2,2,4), imshow(brightened_img), title('Brightened Image');
```

Output:



Experiment 6:

To write and execute programs for image logical operations

MATLAB code for executing AND, OR, NAND, NOR, EXOR, EXNOR and NOT operation on two images and calculating intersection of two images.

```
img1 = imread('image1.jpg');
img2 = imread('image2.jpg');

if size(img1, 3) == 3
    img1 = rgb2gray(img1);
end
if size(img2, 3) == 3
    img2 = rgb2gray(img2);
end

threshold = 128;
binary_img1 = imbinarize(img1, double(threshold) / 255);
binary_img2 = imbinarize(img2, double(threshold) / 255);

% 1. AND operation
and_img = binary_img1 & binary_img2;

% 2. OR operation
or_img = binary_img1 | binary_img2;

% 3. NAND operation
nand_img = ~(binary_img1 & binary_img2);

% 4. NOR operation
nor_img = ~(binary_img1 | binary_img2);

% 5. XOR (EXOR) operation
xor_img = xor(binary_img1, binary_img2);

% 6. XNOR (EXNOR) operation
xnor_img = ~(xor(binary_img1, binary_img2));

% 7. NOT operation (for img1)
not_img1 = ~binary_img1;

% 8. Intersection of two images (AND is often used to calculate
intersection)
intersection_img = and_img;

% Display the results
figure;
subplot(3, 3, 1), imshow(binary_img1), title('Binary Image 1');
subplot(3, 3, 2), imshow(binary_img2), title('Binary Image 2');
subplot(3, 3, 3), imshow(and_img), title('AND Operation');
```

```
subplot(3, 3, 4), imshow(or_img), title('OR Operation');
subplot(3, 3, 5), imshow(nand_img), title('NAND Operation');
subplot(3, 3, 6), imshow(nor_img), title('NOR Operation');
subplot(3, 3, 7), imshow(xor_img), title('XOR Operation');
subplot(3, 3, 8), imshow(xnor_img), title('XNOR Operation');
subplot(3, 3, 9), imshow(not_img1), title('NOT Operation (Image
1)');

figure;
imshow(intersection_img);
title('Intersection of Two Images (AND Operation)');
```

Output: