

DEPARTMENT OF COMPUTER ENGINEERING & APPLICATIONS

DIP LAB FILE

**Name: Pavan Kumar Sharma**

**University Rollno. : 2115000702**

**Subject Name: Digital Image Processing Lab**

**Subject Code: BCSE 0101**

**Course: B.Tech.**

**Year: III**

**Semester: V**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Exp.**  **No.** | **Program** | **Page**  **No.** |
| 1 | Introduction with the MATLAB Command. | 1 |
| 2 | Read the image and display it. | 1 |
| 3 | Create the horizontal and vertical strips. | 1 |
| 4 | Create chessboard. | 2 |
| 5 | Read the colored image and check attributes. | 2 |
| 6 | Perform Log Transformation on Image. | 3 |
| 7 | Perform Negative Transformation on Image. | 3 |
| 8 | Perform Power Log Transformation on Image. | 4 |
| 9 | Perform Gray Level Slicing with and without background. | 4-5 |
| 10 | Perform Histogram Equalization on image. | 5-6 |
| 11 | Find Four, Diagonal and Eight Neighbours of image. | 6-7 |
| 12 | Perform Dilation on Image. | 7-8 |
| 13 | Perform Erosion on Image. | 8-9 |
| 14 | Perform Perforn Closing(Dilation Followed by Erosion) on Image. | 9-10 |

**Experiment -1**

**Objective:-** Introduction with the MATLAB Command.

**Commands:-**

1. clc - Clear Command window

2. clearall - clear the workspace

3. date - displays current date

4. linspace - creates regularly spaced rectory

5. Mat – returns largest element

6. Min - returns smallest element

7. size – computes array size

8. ones – creates an array of ones

9. zeros – creates an array of zeros

10. subplot – creates plot in subwindows.

**Experiment -2**

**Objective:-** Read the image and display it.

**Implement:-**

Img = imread(’cameraman.tif’);

imshow(Img);

**Output:-**



**Experiment -3**

**Objective:-** Create the horizontal and vertical strips.

**Implement:-**

zero\_arr = zeros(10,20); %m\*n zeros matrix

%horizontal strips

hor\_strip = uint8(zero\_arr);

hor\_strip(2:2:10,:) = 255;

subplot(1,2,1),imshow(hor\_strip)

%vertical strip

ver\_strip = uint8(zero\_arr);

ver\_strip(:,2:2:20) = 255; % array(st\_row:diff:end\_row,st\_col:diff:end\_col)

subplot(1,2,2),imshow(ver\_strip)

**Output:-**



**Experiment -4**

**Objective:-** Create chessboard.

**Implement:-**

zero\_arr = zeros(8,8); %m\*n zeros matrix

%chess

hor\_strip = uint8(zero\_arr);

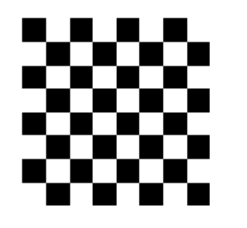
hor\_strip(2:2:8,:) = 255;

hor\_strip(:,2:2:8) = 255;

hor\_strip(2:2:8,2:2:8) = 0;

subplot(1,2,1),imshow(hor\_strip)

**Output:-**



**Experiment -5**

**Objective:-** Read the colored image and check attributes.

**Implement:-**

i1 = imread ("peppers.png");

r = i1 (:,:,1);

g = i1(:,:,2);

b = i1(:,:,3);

subplot(2,2,1);

imshow (i1);

subplot(2,2,2);

imshow (r);

subplot(2,2,3);

imshow (g);

subplot(2,2,4);

imshow (b);

**Output:-**

****

**Experiment -6**

**Objective:-** Perform Log Transformation on Image.

**Implement:-**

img = imread('cameraman.tif');

subplot(1,2,1), imshow(img)

% Convert datatype to Double

% (for allowing fractional values)

r = double(img);

% Constant determining the

% nature of the log curve

C = 40;

% Performing log transformation

S = C \* log(1 + r);

%T = 255/(C \* log(256));

% Converting the datatype back

% to integer for displaying

Log = uint8(S);

subplot(1,2,2), imshow(Log)

**Output:-**



**Experiment -7**

**Objective:-** Perform Negative Transformation on Image.

**Implement:-**

img = imread('cameraman.tif');

%Negative Transformation - 1

neg=255-img;

subplot(1,2,1),imshow(img)

%Negative Transformation - 2

neg=img;

for row=1:size(img,1)

for col=1:size(img,2)

neg(row,col,:)=255-img(row,col,:);

end

end

subplot(1,2,2),imshow(neg)

**Output:-**



**Experiment -8**

**Objective:-** Perform Power Log Transformation on Image.

**Implement:-**

img = imread('cameraman.tif');

subplot(1,2,1), imshow(img)

% Convert datatype to Double

% (for allowing fractional values)

r = double(img);

% Constant determining the

% nature of the log curve

C = 100;

gamma = 0.5;

% Performing power log transformation

S = C \* log(1 + r)^gamma;

T = 255/(C \* log(256)^gamma);

% Converting the datatype back

% to integer for displaying

Log = uint8(T \* S);

subplot(1,2,2), imshow(Log)

**Output:-**



**Experiment -9**

**Objective:-** Perform Gray Level Slicing with and without background.

**Implement:-**

% Read the input image

grayImage = imread('cameraman.tif');

% Set the threshold values (adjust as needed)

lowThreshold = 50;

highThreshold = 150;

% Perform gray level slicing without background reference

outputImage = grayImage;

outputImage(grayImage > lowThreshold & grayImage < highThreshold) = 255;

% Perform gray level slicing with background reference

whback= grayImage;

whback(grayImage < lowThreshold | grayImage > highThreshold) = 255;

subplot(2,2,3), imshow(whback), title('With Background')

% Display the original and processed images

figure;

subplot(2, 2, 1), imshow(grayImage), title('Original Image');

subplot(2, 2, 2), imshow(outputImage), title('Gray Level Slicing without Background Reference');

**Output:-**



**Experiment -10**

**Objective:-** Perform Histogram Equalization on image.

**Implement:-**

img=imread('cameraman.tif');

[n,m]=size(img);

%n-->row

%m-->col

% First Column --> rk

for i=1:256

t(i,1)=i-1;

t(i,2)=0;

end

% Second Column --> nk

%Frequency Count

for i=1:n

for j=1:m

pix=img(i,j)+1;

t(pix,2)=t(pix,2)+1;

end

end

for i=1:256

t(i,3)=t(i,2)/(256\*256);

end

t(1,4)=t(1,3)

for i=2:256

t(i,4)=t(i-1,4)+t(i,3);

end

for i=1:256

t(i,5)=t(i,4)\*255;

end

for i=1:256

t(i,6)=round(t(i,5));

end

for i=1:256s(i,1)=i-1;

s(i,2)=0;

end

for i=1:256

pix=t(i,6)+1;

s(pix,2)=s(pix,2)+t(i,2);

end

img\_2=img;

for i=1:n

for j=1:m

img\_2(i,j)=t(img(i,j)+1,6);

end

end

subplot(3,2,1),imshow(img),title('Original');

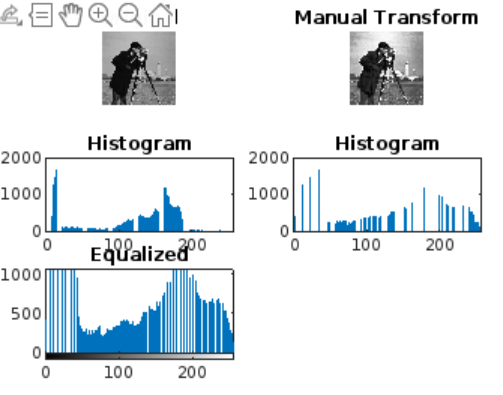
subplot(3,2,2),imshow(img\_2),title('Manual Transform');

subplot(3,2,3),bar(t(:,1),t(:,2)),title('Histogram');

subplot(3,2,4),bar(s(:,1),s(:,2)),title('Histogram');

subplot(3,2,5),imhist(img\_2),title("Equalized");

**Output:-**

****

**Experiment -11**

**Objective:-** Find Four, Diagonal and Eight Neighbours of image.

**Implement:-**

% Define the image matrix

img = [1, 2, 3;

4, 5, 6;

7, 8, 9];

% Input values for x and y

x = input('X-axis: ');

y = input('Y-axis: ');

% Define height and width

[height, width] = size(img);

fprintf('Height: %d, Width: %d\n', height, width);

% Initialize arrays for neighbors

fourN = NaN(1, 4);

eightN = NaN(1, 8);

diagonalN = NaN(1, 4);

% Four Neighbour

if x > 1

fourN(1) = img(x - 1, y);

end

if y > 1

fourN(2) = img(x, y - 1);

end

if x < height

fourN(3) = img(x + 1, y);

end

if y < width

fourN(4) = img(x, y + 1);

end

% Diagonal Neighbour

if x > 1 && y > 1

diagonalN(1) = img(x - 1, y - 1);

end

if x < height && y < width

diagonalN(2) = img(x + 1, y + 1);

end

if x < height && y > 1

diagonalN(3) = img(x + 1, y - 1);

end

if x > 1 && y < width

diagonalN(4) = img(x - 1, y + 1);

end

% Display the results

fprintf('Four Neighbors: %s\n', mat2str(fourN));

fprintf('Diagonal Neighbors: %s\n', mat2str(diagonalN));

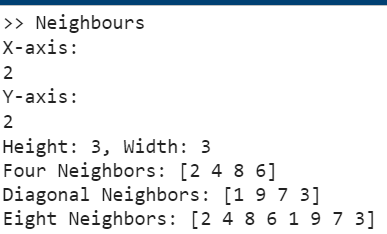
% Combine eight neighbors

eightN(1:4) = fourN(1:4);

eightN(5:8) = diagonalN(1:4);

fprintf('Eight Neighbors: %s\n', mat2str(eightN));

**Output:-**



**Experiment -12**

**Objective:-** Perform Dilation on Image.

**Implement:-**

% Create a binary image

binaryImage = imread('cameraman.tif')

binaryImage = im2bw(binaryImage);

% Create a structuring element (a simple 3x3 square)

se = logical(ones(3, 3));

% Get the size of the binary image

[rows, cols] = size(binaryImage);

% Initialize the result image

dilatedBinaryImage = false(rows, cols);

% Iterate through the image and perform dilation

for r = 2:rows-1

for c = 2:cols-1

% Check if any of the structuring element pixels overlap with the image

if any(any(binaryImage(r-1:r+1, c-1:c+1) & se))

dilatedBinaryImage(r, c) = true;

end

end

end

% Display the original and manually dilated binary images

figure;

subplot(2, 1, 1),imshow(binaryImage);

title('Original Binary Image');

subplot(2, 1, 2),imshow(dilatedBinaryImage);

title('Manually Dilated Binary Image');

**Output:-**



**Experiment -13**

**Objective:-** Perform Erosion on Image.

**Implement:-**

% Create a binary image

binaryImage = imread('cameraman.tif');

binaryImage = im2bw(binaryImage);

% Create a structuring element (a simple 3x3 square)

se = logical(ones(3, 3));

[rows, cols] = size(binaryImage);

erodedBinaryImage = true(rows, cols);

% Iterate through the image and perform erosion

for r = 2:rows-1

for c = 2:cols-1

% Check if all structuring element pixels overlap with the image

if all(all(binaryImage(r-1:r+1, c-1:c+1) & se))

erodedBinaryImage(r, c) = true;

else

erodedBinaryImage(r, c) = false;

end

end

end

% Display the original and manually eroded binary images

figure;

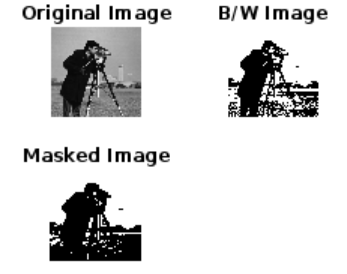
subplot(1, 2, 1),imshow(binaryImage);

title('Original Binary Image');

subplot(1, 2, 2),imshow(erodedBinaryImage);

title('Manually Eroded Binary Image');

**Output:-**



**Experiment -14**

**Objective:-** Perform Perforn Closing(Dilation Followed by Erosion) on Image.

**Implement:-**

%Closing

% Create a binary image

binaryImage = imread('cameraman.tif')

binaryImage = im2bw(binaryImage);

% Create a structuring element (a simple 3x3 square)

se = logical(ones(3, 3));

% Get the size of the binary image

[rows, cols] = size(binaryImage);

% Initialize the dilatedresult image

dilatedBinaryImage = false(rows, cols);

% Display the original image

figure;

subplot(2, 2, 1), imshow(binaryImage), title('Original Image');

% Iterate through the image and perform dilation

for r = 2:rows-1

for c = 2:cols-1

% Check if any of the structuring element pixels overlap with the image

if any(any(binaryImage(r-1:r+1, c-1:c+1) & se))

dilatedBinaryImage(r, c) = true;

end

end

end

% Display the dilated image

figure;

subplot(2, 2, 2), imshow(binaryImage), title('Dilated Image');

% Initialize the result image

erodedBinaryImage = true(rows, cols);

% Iterate through the image and perform erosion

for r = 2:rows-1

for c = 2:cols-1

% Check if all structuring element pixels overlap with the image ­­­­

if all(all(dilatedBinaryImage(r-1:r+1, c-1:c+1) & se))

erodedBinaryImage(r, c) = true;

else

erodedBinaryImage(r, c) = false;

end

end

end

subplot(2, 2, 3), imshow(erodedBinaryImage), title('Erosion Binary Image');

**Output:-**

