Srishyla Educational Trust (R), Bheemasamudra

GM INSTITUTE OF TECHNOLOGY, DAVANGERE

Approved by AICTE | Affiliated by V.T.U Belagavi | Recognized by Govt. of Karnataka

NATIONAL BOARD of ACCREDITATION Accredited Programs CSE/ECE/CE/ME/ISE/BT

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Digital Image Processing Laboratory

(Subject Code: 18AIL67)

Lab Manual

For VI Semester

VISION & MISSION OF THE AI&ML DEPARTMENT

VISION

To develop AIML engineers with **technical competencies** and able to solve **social issues** in real **world.**

MISSION

1. To provide **academic environment** for **training students** with necessary **infrastructure through pedagogical learning.**

2. To prepare students with **technical and professional skills** of **industry** through **interdisciplinary** approach.

3. To **motivate** and **support students** in developing **entrepreneurial skills** through **innovation.**

4. To encourage students for **solving research** problems, in **collaboration** with **other disciplines** and support of **funding agencies.**

Staff Incharge **HOD**

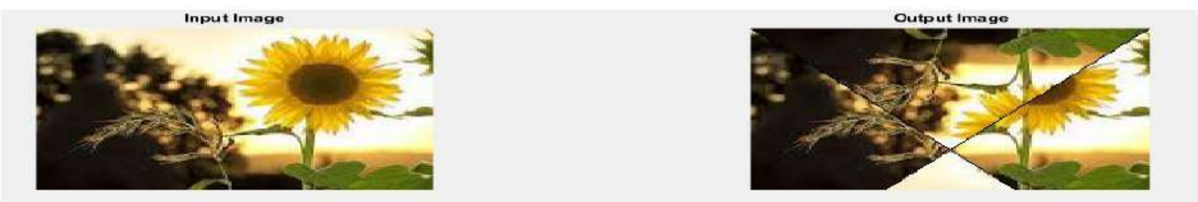
Mr. Rangaswamy H
Asst. Professor, Dept. of AIML

Keerthi Prasad G
HEAD, Dept. of AIML

1. Write a Program to read a digital image. Split and display image into 4 quadrants, up, down, right and left

```
% Read the Input Image
OI = imread('C:\Users\OMLINE CENTER\Desktop\images.jpg');
figure
subplot(2,2,1);
imshow(OI);
title('Input Image');
%%
% Identify the rows and columns
[rows, columns, numberOfColorChannels] = size(OI);
%%
% ndgrid Rectangular grid in N-D space
[Y,X]=ndgrid(1:rows,1:columns);
%% Identify the four quadrants
Jf=columns+1-Y;
Q1=(Y<X & X<Jf);
Q2=(Y<X & X>Jf);
Q3=(Y>X & X>Jf);
Q4=(Y>X & X<Jf);
%%
% flipdim - Flip array along specified dimension
flippedImage = flipud(OI);
%%  Mask out Q1 | Q3
% Mask the image using bsxfun() function to multiply the mask by each channel individually.
% bsxfun - Apply element-by-element binary operation to two arrays with singleton expansion enabled
mask = Q1 | Q3;
maskedImage1 = bsxfun(@times, flippedImage, cast(mask, 'like', OI));
mask = Q2 | Q4;
maskedImage2 = bsxfun(@times, OI, cast(mask, 'like', OI));
%% Mask1 and Mask2 combined for quadrants
finalImage = maskedImage1 + maskedImage2;
subplot(2,2,2);
imshow(finalImage);
title('Output Image')
```

OUTPUT

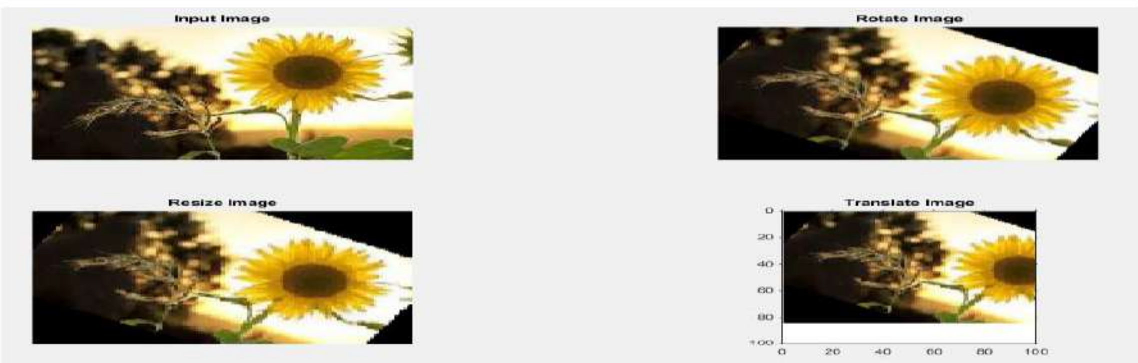


GMIT

2. Write a program to show rotation, scaling, and translation of an image
% reading the image

```
I=imread('C:\Users\OMLINE CENTER\Desktop\images.jpg');
figure
subplot(2,2,1);
imshow(I);
title('Input Image');

%%
% rotate the image
deg = -30;
I2 = imrotate(I, deg, 'bilinear', 'crop');
subplot(2,2,2);
imshow(I2)
title('Rotate Image');
%%
% resize the image(scale)
I3=imresize(I2,0.7,'nearest');
subplot(2,2,3);
 imshow(I3)
 title('Resize Image');

%%
% translate the image
 T = [1 0 0; 0 1 0;0 0 1];
tform_translate = maketform('affine',T);
[I4 xdata ydata] = imtransform(I3,tform_translate);
subplot(2,2,4);
imshow(I4)
title('Translate Image');
axis on
axis([0 100 0 100])
%%
```

OUTPUT

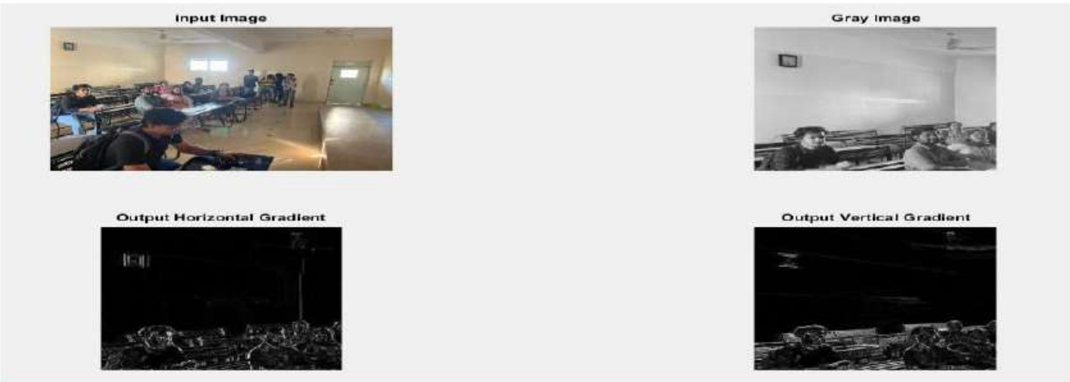


GMIT

3. Read an image, first apply erosion to the image and then subtract the result from the original. Demonstrate the difference in the edge image if you use dilation instead of erosion.

```
% Read input image
rgb = imread('C:\Users\OMLINE CENTER\Desktop\13.jpg');
figure;
subplot(2,2,1);
imshow(rgb);
title('input Image');
I = rgb2gray(rgb(1:256,1:256,:));
%% Structure elements

se = strel(ones(3,3));
%% Substract the gradient from dialation to erosion

basic_gradient = imdilate(I, se) - imerode(I, se);

subplot(2,2,2)
imshow(I);
title('Gray Image');

internal_gradient = I - imerode(I, se);
external_gradient = imdilate(I, se) - I;
%%

seh = strel([1 1 1]);
sev = strel([1;1;1]);

horizontal_gradient = imdilate(I,seh) - imerode(I,seh);
vertical_gradient = imdilate(I,sev) - imerode(I,sev);
subplot(2,2,3)
imshow(horizontal_gradient, []), title('Output Horizontal Gradient')
subplot(2,2,4)
imshow(vertical_gradient, []), title('Output Vertical Gradient')
```

OUTPUT

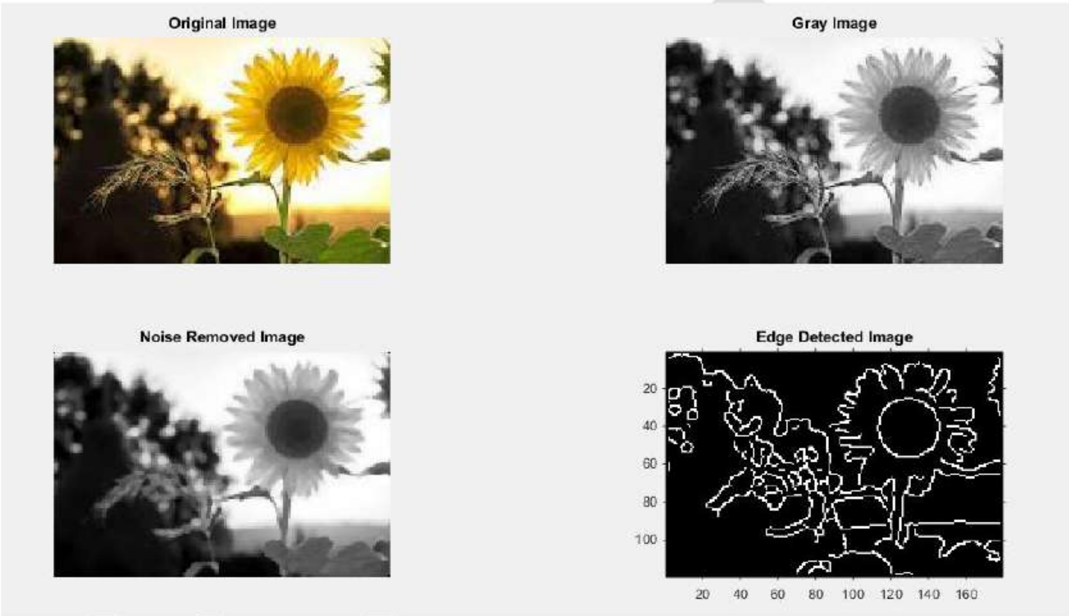


4. Read an image and extract and display low-level features such as edges, textures using filtering techniques

```
OI= imread('C:\Users\OMLINE CENTER\Desktop\images.jpg');
figure
subplot(2,2,1);
imshow(OI)
title('Original Image');

%%
GI=rgb2gray(OI);
% figure
 subplot(2,2,2);
imshow(GI)
title('Gray Image');
%%
% Find Limits to Stretch Contrast in Grayscale Image
SI= imadjust(GI,stretchlim(GI),[]);
% figure
subplot(2,2,3);
imshow(SI)
title('Stretched Image');

%%
% Now use a median filter to filter the noisy image, J. The example also uses
% a 3-by-3 neighborhood. Display the two filtered images side-by-side for comparison.
% Notice that medfilt2 does a better job of removing noise, with less blurring of edges
of the coins.
K = medfilt2(SI);
% figure
 subplot(2,2,3);
imshow(K)
title('Noise Removed Image');
%
%% Edge detection
% Get edges
Canny_img = edge(K, 'Canny');
subplot(2, 2, 4);
imshow(Canny_img, [])
axis('on', 'image');
title('Edge Detected Image')
% Enlarge figure to full screen.
set(gcf, 'Units', 'Normalized', 'Outerposition', [0, 0.05, 1, 0.95]);

%%
%(i)Entropy
%finding entropy value from gray image
 e=entropy(K);%entropy syntax
 fprintf(1,'entropy of image =%f\n',e);
 %%
```

```
%(ii)Mean
%finding mean value from gray image
m = mean2(K);
fprintf(1,'mean of image =%f\n',m);
%%

%(iii)Variance
%finding variance value from gray image
%To calculate the variance
v = var(double(K(:)));
fprintf(1,'Variance of image =%f\n',v);
```

OUTPUT



5. Demonstrate enhancing and segmenting low contrast 2D images.

```
I = imread('C:\Users\OMLINE CENTER\Desktop\13.jpg');
I=imresize(I,0.5);
figure
subplot(2,2,1)
imshow(I)
title('Original Image')
K = imadjust(I,[0.3 0.7],[]);

subplot(2,2,2)
imshow(K)
title('Enhanced Image')
%% separte channel
redchannel=K(:,:,1);
greenchannel=K(:,:,2);
bluechannel=K(:,:,3);
data=double([redchannel(:),greenchannel(:),bluechannel(:)]);
% irerartion of number of classes
for i=1:10
numberofClasses=i;
% using K means color image segmentation
[m,n]=kmeans(data,numberofClasses);
m=reshape(m,size(K,1),size(K,2));
n=n/255;
clusteredImage=label2rgb(m,n);
subplot(2,2,3);
imshow(clusteredImage);
title('Segmented image')
end
```

OUTPUT

GMIT