**NAME - ANKIT MISHRA**

**REG NO. - 20168047**

**MULTIMEDIA TECHNOLOGY**

**B.Tech 3rd Year, IT**

**Assignment – 1**

1. **Rotate an image by 90°.**

**Code :**

x = imread('download.jpg');

x = rgb2gray(x);

[r,c] = size(x);

m = zeros(c, r, 'uint8');

for i=1:r

for j=1:c

m(j, i) = x(r-i+1, j);

end

end

imshow(m),

title('90degree');

1. **Rotate an image by 180°.**

**Code :**

x = imread('download.jpg');

x = rgb2gray(x);

[r,c] = size(x);

m = zeros(r, c, 'uint8');

for i=1:r

for j=1:c

m(i, j) = x(r-i+1, c-j+1);

end

end

imshow(m),

title('180degree');

1. **Create mirror image of a given image.**

**Code :**

x = imread('download.jpg');

x = rgb2gray(x);

[r,c] = size(x);

m = zeros(r, c, 'uint8');

for i=1:r

for j=1:c

m(i, j) = x(i, c-j+1);

end

end

imshow(x),title('Original Image');

imshow(m),title('mirror image');

**Assignment -2**

1. **Zoom an image using pixel replication method.**

**Code :**

x = imread('download.jpg');

x = rgb2gray(x);

[r,c] = size(x);

m = zeros(2\*r,2\*c,'uint8');

for i=1:r

for j=1:c

m(2\*i-1,2\*j-1) = x(i,j);

m(2\*i,2\*j) = x(i,j);

m(2\*i-1,2\*j) = x(i,j);

m(2\*i,2\*j-1) = x(i,j);

end

end

figure,imshow(m),title('Pixel Replication method for zooming');

1. **Zoom an image using zero order hold method.**

**Code :**

x = imread('download.jpg');

x = rgb2gray(x);

[r,c] = size(x);

m = zeros(2\*r-1,2\*c-1,'uint16');

x = uint16(x);

for i=1:r

for j=1:c

m(2\*i-1,2\*j-1) = x(i,j);

if (j~=c)

m(2\*i-1,2\*j) = (x(i,j)+x(i,j+1))/2;

end

if (i~=r)

m(2\*i,2\*j-1) = (x(i,j)+x(i+1,j))/2;

end

if (i~=r && j~=c)

m(2\*i,2\*j) = (x(i,j+1)+x(i+1,j))/2;

end

end

end

n = uint8(m);

figure,imshow(n),title('zero order hold method for zooming');

1. **Zoom an image using k-times zooming method.**

**Code :**

x = imread('download.jpg');

x = rgb2gray(x);

[r,c] = size(x);

m = zeros(4\*(r-1), 4\*(c-1),'uint16');

n = zeros(r,4\*(c-1), 'uint16');

x = uint16(x)

for i=1:r

for j=1:c-1

op = (x(i,j+1)-x(i,j))/4;

n(i,4\*j) = x(i,j);

n(i,4\*j+1) = n(i,4\*j) + op;

n(i,4\*j+2) = n(i,4\*j+1) + op;

n(i,4\*j+3) = n(i,4\*j+2) + op;

end

end

for i=1:r-1

for j=1:4\*(c-1)

op = (n(i+1,j) - n(i,j))/4;

m(4\*i,j) = n(i,j);

m(4\*i+1,j) = m(4\*i,j) + op;

m(4\*i+2,j) = m(4\*i+1,j) + op;

m(4\*i+3, j) = m(4\*i+2,j) + op;

end

end

m = uint8(m);

figure,imshow(m),title('k times zoom method');

**Assignment-3**

1. **Zoom out a given image.**

**Code :**

I = imread('download.jpg');

temp = imread('download.jpg');

b=temp;

[m,n,colormap]=size(b);

%If RGB Image is given at Input

if colormap==3

x=b(:,:,1);

y=b(:,:,2);

z=b(:,:,3);

end

new\_mat=zeros(m\*2,n\*2,3,'uint8');

a = 1;

b = 1;

c = 0;

for i=1:m

for j=1:n

new\_mat(a,b,1) = x(i,j);

new\_mat(a,b+1,1) =x(i,j);

new\_mat(a+1,b,1) =x(i,j);

new\_mat(a+1,b+1,1) =x(i,j);

% c = c + 1;

new\_mat(a,b,2) = y(i,j);

new\_mat(a,b+1,2) =y(i,j);

new\_mat(a+1,b,2) = y(i,j);

new\_mat(a+1,b+1,2) =y(i,j);

new\_mat(a,b,3) = z(i,j);

new\_mat(a,b+1,3) =z(i,j);

new\_mat(a+1,b,3) = z(i,j);

new\_mat(a+1,b+1,3) = z(i,j);

b = b + 2;

end

b = 1;

a = a + 2;

end

%imshow(new\_mat);

% zoom out

tmp = zeros(m,n,3,'uint8');

a = 1

b = 1

for i = 1:2:2\*m

for j = 1:2:2\*n

tmp(a,b,1) = new\_mat(i,j,1);

tmp(a,b,2) = new\_mat(i,j,2);

tmp(a,b,3) = new\_mat(i,j,3);

b = b + 1;

end

b = 1;

a = a + 1;

end

imshow(tmp),title('zoom out image');

1. **Draw Border around an image.**

**Code :**

Img = imread('download.jpg');

temp = imread('download.jpg');

b=temp;

[m,n,colormap]=size(b);

%If RGB Image is given at Input

if colormap==3

x=b(:,:,1);

y=b(:,:,2);

z=b(:,:,3);

end

mat = zeros(m+40,n+40,3,'uint8');

for i=1:(m+20)

for j=1:20

mat(i,j,1) = 0;

mat(i,j,2) = 0;

mat(i,j,3) = 0;

end

end

for i=1:20

for j=1:n+20

mat(i,j,1) = 0;

mat(i,j,2) = 0;

mat(i,j,3) = 0;

end

end

for i=m+20:-1:m

for j=1:n+20

mat(i,j,1) = 0;

mat(i,j,2) = 0;

mat(i,j,3) = 0;

end

end

for i=1:m+20

for j=n:n+20

mat(i,j,1) = 0;

mat(i,j,2) = 0;

mat(i,j,3) = 0;

end

end

for i=1:m

for j=1:n

mat(i+20,j+20,1) = x(i,j);

mat(i+20,j+20,2) = y(i,j);

mat(i+20,j+20,3) = z(i,j);

end

end

imshow(mat),title('Border around image');

1. **Divide an image into 4 parts.**

**Code :**

Img = imread('download.jpg');

temp = imread('download.jpg');

b=temp;

[m,n,colormap]=size(b);

%If RGB Image is given at Input

if colormap==3

x=b(:,:,1);

y=b(:,:,2);

z=b(:,:,3);

end

mm = m/2;

nn = n/2;

m1 = zeros(mm,nn,3,'uint8');

m2 = zeros(mm,nn,3,'uint8');

m3 = zeros(mm,nn,3,'uint8');

m4 = zeros(mm,nn,3,'uint8');

for i = 1:mm

for j = 1:nn

m1(i,j,1) = x(i,j);

m1(i,j,2)= y(i,j);

m1(i,j,3) = z(i,j);

end

end

for i = mm+1:m

for j = 1:nn

m2(i-mm,j,1) = x(i,j);

m2(i-mm,j,2)= y(i,j);

m2(i-mm,j,3) = z(i,j);

end

end

for i = 1:mm

for j = nn+1:n

m3(i,j-nn,1) = x(i,j);

m3(i,j-nn,2)= y(i,j);

m3(i,j-nn,3) = z(i,j);

end

end

for i = mm+1:m

for j = nn+1:n

m4(i-mm,j-nn,1) = x(i,j);

m4(i-mm,j-nn,2)= y(i,j);

m4(i-mm,j-nn,3) = z(i,j);

end

end

figure,title('Image in 4 parts');

subplot(2,2,1), imshow(m1);

subplot(2,2,2), imshow(m3);

subplot(2,2,3), imshow(m2);

subplot(2,2,4), imshow(m4);

**Assignment-4**

1. **Read and display an image. Check how the image appears in workspace. Plot histogram of this image and then improve the contrast of the image by histogram equalization technique. Write the adjusted image to a disk file.**

**Code :**

tmp = imread('download.jpg');

I = tmp;

% If the image is a color image, change it to a gray image

[height,width, colormap]=size(tmp);

I = rgb2gray(tmp);

% show the original image

subplot(241)

imshow(I)

title('Original Image');

% no\_of\_pixels is the number of occurrences of each gray level

no\_of\_pixels = com\_hist(I,'Histogram of Original Image',2);

% prob\_of\_pixel is the probability of an occurrence of each gray level

prob\_of\_pixel = com\_normalized\_hist(no\_of\_pixels,I);

% cum\_pixel is the cumulative distribution function

cum\_pixel = comp\_cumul\_hist(prob\_of\_pixel);

% cum\_pixel convert to new gray levels

mapping = zeros(1,256);

for i = 1:256

mapping(i) = uint8(255\*cum\_pixel(i) + 0.5);

end

for i = 1:height

for j = 1:width

I(i,j)=mapping(I(i,j) + 1);

end

end

% show the new image

subplot(245)

imshow(I)

title('New Image');

%imsave

function f = com\_hist(I,title1,position)

%compute\_histogram

[height,width]=size(I);

% Compute the number of occurrences of each gray level

no\_of\_pixels = zeros(1,256); % saare pixels ke occurance nikal lo

for i = 1:height

for j = 1:width

no\_of\_pixels( I(i,j) + 1 ) = no\_of\_pixels( I(i,j) + 1 ) + 1;

end

end

subplot(2,4,position)

bar(no\_of\_pixels)

title(title1);

f = no\_of\_pixels;

end

function f = com\_normalized\_hist(no\_of\_pixels,I)

[height,width]=size(I);

% Compute the probability of an occurrence of each gray level

prob\_of\_pixel = zeros(1,256);

for i = 1:256

prob\_of\_pixel(i) = no\_of\_pixels(i) / (height \* width \* 1.0);

end

f = prob\_of\_pixel;

end

function f = comp\_cumul\_hist(prob\_of\_pixel)

% Compute the cumulative distribution function

cum\_pixel = zeros(1,256);

for i = 1:256

if (i == 1)

cum\_pixel(i) = prob\_of\_pixel(i);

else

cum\_pixel(i) = cum\_pixel(i-1) + prob\_of\_pixel(i);

end

end

f = cum\_pixel;

end

**Assignment-5**

1. **Input image is composed of eight 1 bit planes ranging from bit plane 0 for the least significant bit to bit plane 7 for the most significant bit. Plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image & plane 7 contains all the high order bits. Extract each bit plain from gray scale image. Merge all these extracted planes to get original image.**

**Code :**

tmp = imread('download.jpg');

A = rgb2gray(tmp);

[height,width]=size(A);

figure, imshow(A);title('Original Image');

B1=bitget(A,1); figure, imshow(logical(B1));title('Bit plane 1');

B2=bitget(A,2); figure, imshow(logical(B2));title('Bit plane 2');

B3=bitget(A,3); figure, imshow(logical(B3));title('Bit plane 3');

B4=bitget(A,4); figure, imshow(logical(B4));title('Bit plane 4');

B5=bitget(A,5); figure, imshow(logical(B5));title('Bit plane 5');

B6=bitget(A,6); figure, imshow(logical(B6));title('Bit plane 6');

B7=bitget(A,7); figure, imshow(logical(B7));title('Bit plane 7');

B8=bitget(A,8); figure, imshow(logical(B8));title('Bit plane 8');

BB = (2\*(2\*(2\*(2\*(2\*(2\*(2\*B8+B7)+ B6) + B5) + B4) + B3) + B2) + B1);

figure, imshow(BB);title('Recombined Image from extracted planes');

1. **Hide some information in LSB plane and then show the hidden content by extracting the targeted plane.**

**Code :**

tmp = imread('download.jpg');

A = rgb2gray(tmp);

tmp2 = imread('Krishna.jpg');

tmp2 = rgb2gray(tmp2);

[h,w] = size(tmp2);

[height,width]=size(A);

%figure, imshow(A);title('Original Image');

mb8=bitget(tmp2,8);

mb7=bitget(tmp2,7);

mb6=bitget(tmp2,6);

new\_mat = zeros(h,w,'uint8');

for i=1:height

for j=1:width

new\_mat(i,j) = A(i,j);

end

end

B1=bitget(new\_mat,1);

B2=bitget(new\_mat,2);

B3=bitget(new\_mat,3);

B4=bitget(new\_mat,4);

B5=bitget(new\_mat,5);

B6=bitget(new\_mat,6);

B7=bitget(new\_mat,7);

B8=bitget(new\_mat,8);

B1 = mb8;

B2 = mb7;

B3 = mb6;

BB = (2\*(2\*(2\*(2\*(2\*(2\*(2\*B8+B7)+ B6)+B5)+B4)+B3)+B2)+B1);

xx = zeros(height,width,'uint8');

for i=1:height

for j=1:width

xx(i,j) = BB(i,j);

end

end

figure,imshow(xx),title('original Image');

BB2 = (2^7\*B1) + (2^6\*B2) + (2^5\*B3);

figure,imshow(BB2),title('hidden image');

1. **After zooming using pixel replication method write your own function to adjust brightness of image.**

**Code :**

I = imread('download.jpg');

temp = imread('download.jpg');

b=temp;

tt = I;

[m,n,colormap]=size(b);

%imshow(I);

%If RGB Image is given at Input

if colormap==3

x=b(:,:,1);

y=b(:,:,2);

z=b(:,:,3);

end

new\_mat=zeros(m\*2,n\*2,3,'uint8');

a = 1;

b = 1;

c = 0;

for i=1:m

for j=1:n

new\_mat(a,b,1) = x(i,j);

new\_mat(a,b+1,1) =x(i,j);

new\_mat(a+1,b,1) =x(i,j);

new\_mat(a+1,b+1,1) =x(i,j);

new\_mat(a,b,2) = y(i,j);

new\_mat(a,b+1,2) =y(i,j);

new\_mat(a+1,b,2) = y(i,j);

new\_mat(a+1,b+1,2) =y(i,j);

new\_mat(a,b,3) = z(i,j);

new\_mat(a,b+1,3) =z(i,j);

new\_mat(a+1,b,3) = z(i,j);

new\_mat(a+1,b+1,3) = z(i,j);

b = b + 2;

end

b = 1;

a = a + 2;

end

%imshow(new\_mat);

% brightness

II = brightness(tt);

figure,imshow(tt),title('original Image');

figure,imshow(II),title('Brighter Image');

function f = brightness(I)

[w,h,colormap] = size(I);

for i=1:w

for j=1:h

if(I(i,j,1)<=215)

I(i,j,1) = I(i,j,1) + 35;

end

if(I(i,j,2)<=215)

I(i,j,2) = I(i,j,2) + 35;

end

if(I(i,j,3)<=215)

I(i,j,3) = I(i,j,3) + 35;

end

end

end

f = I;

end

**Assignment-6**

1. **Implement a MeanFilter function to perform a filtering operation on the input image.**

**Call this function to process the noisy image by using a weighted 3-by-3 averaging filter and a standard 5-by-5 averaging filter, respectively. Display original image and both filtered image.(Don’t use inbuilt function for filtering.). Plot Histogram of filtered image and compare it with original image**.

**Code :**

Img = imread('download.jpg');

temp = imread('download.jpg');

b=temp;

[m,n,colormap]=size(b);

%If RGB Image is given at Input

if colormap==3

x=b(:,:,1);

y=b(:,:,2);

z=b(:,:,3);

x1=b(:,:,1);

y1=b(:,:,2);

z1=b(:,:,3);

end

% for 3 \* 3

for i=1:m-2

sum1 = 0;

sum2 = 0;

sum3=0;

for j=1:n-2

for r=i:(i+2)

for s=j:(j+2)

sum1 = int16(sum1) + int16(x(r,s));

sum2 = int16(sum2) + int16(y(r,s));

sum3 = int16(sum3) + int16(z(r,s));

end

end

avg1 = int16(sum1 / 9);

avg2 = int16(sum2 / 9);

avg3 = int16(sum3 / 9);

x(i+1,j+1) = avg1;

y(i+1,j+1) = avg2;

z(i+1,j+1) = avg3;

sum1=0;sum2=0; sum3 = 0;

end

end

c(:,:,1)=x;

c(:,:,2)=y;

c(:,:,3)=z;

subplot(4,4, 1);

imshow(Img);

title('Original Image');

subplot(4, 4, 2);

imshow(c);

title('3\*3 mean filtering');

%for 5\*5

for i=1:m-4

sum1 = 0;

sum2 = 0;

sum3=0;

for j=1:n-4

for r=i:(i+4)

for s=j:(j+4)

sum1 = int16(sum1) + int16(x1(r,s));

sum2 = int16(sum2) + int16(y1(r,s));

sum3 = int16(sum3) + int16(z1(r,s));

end

end

avg1 = int16(sum1 / 25);

avg2 = int16(sum2 / 25);

avg3 = int16(sum3 / 25);

x1(i+2,j+2) = avg1;

y1(i+2,j+2) = avg2;

z1(i+2,j+2) = avg3;

sum1=0;sum2=0; sum3 = 0;

end

end

cc(:,:,1)=x1;

cc(:,:,2)=y1;

cc(:,:,3)=z1;

subplot(4, 4, 3);

imshow(Img);

title('Original Image');

subplot(4, 4, 4);

imshow(cc);

title('5\*5 mean filtering');

Img = rgb2gray(Img);

c = rgb2gray(c);

cc = rgb2gray(cc);

subplot(4,4,5);

imhist(Img);

title('Original Image Histogram');

subplot(4, 4, 6);

imhist(c);

title('3\*3 histogram');

subplot(4,4,7);

imhist(Img);

title('Original Image Histogram');

subplot(4, 4, 8);

imhist(cc);

title('5\*5 histogram');

1. **Implement a MedianFilter function to perform a filtering operation on the input image.**

**Call this function to process the same noisy image Circuit by using a standard 3-by-3 median filter and a standard 5-by-5 median filter, respectively. Display original image and both filtered images. (Don’t use inbuilt function for filtering.). Plot Histogram of filtered image and compare it with original image.**

**Code :**

Img = imread('download.jpg');

temp = imread('download.jpg');

b=temp;

[m,n,colormap]=size(b);

%If RGB Image is given at Input

if colormap==3

x=b(:,:,1);

y=b(:,:,2);

z=b(:,:,3);

x1=b(:,:,1);

y1=b(:,:,2);

z1=b(:,:,3);

end

% for 3 \* 3

for i=1:m-2

for j=1:n-2

tt1 = zeros(1, 9);

tt2 = zeros(1, 9);

tt3 = zeros(1, 9);

kk = 1;

for r=i:(i+2)

for s=j:(j+2)

tt1(kk) = x(r,s);

tt2(kk) = y(r,s);

tt3(kk) = z(r,s);

kk = kk + 1;

end

end

t1 = sort(tt1);

t2 = sort(tt2);

t3 = sort(tt3);

x(i+1,j+1) = t1(5);

y(i+1,j+1) = t2(5);

z(i+1,j+1) = t3(5);

end

end

c(:,:,1)=x;

c(:,:,2)=y;

c(:,:,3)=z;

subplot(4,4, 1);

imshow(Img);

title('Original Image');

subplot(4, 4, 2);

imshow(c);

title('3\*3 median\_filtering');

%for 5\*5

for i=1:m-4

for j=1:n-4

tt1 = zeros(1, 25);

tt2 = zeros(1, 25);

tt3 = zeros(1, 25);

kk = 1;

for r=i:(i+4)

for s=j:(j+4)

tt1(kk) = x1(r,s);

tt2(kk) = y1(r,s);

tt3(kk) = z1(r,s);

kk = kk + 1;

end

end

t1 = sort(tt1);

t2 = sort(tt2);

t3 = sort(tt3);

x1(i+2,j+2) = t1(13);

y1(i+2,j+2) = t2(13);

z1(i+2,j+2) = t3(13);

end

end

cc(:,:,1)=x1;

cc(:,:,2)=y1;

cc(:,:,3)=z1;

subplot(4,4, 3);

imshow(Img);

title('Original Image');

subplot(4, 4, 4);

imshow(cc);

title('5\*5 median\_filtering');

Img = rgb2gray(Img);

c = rgb2gray(c);

cc = rgb2gray(cc);

subplot(4,4,5);

imhist(Img);

title('Original Image Histogram');

subplot(4, 4, 6);

imhist(c);

title('3\*3 histogram');

subplot(4,4,7);

imhist(Img);

title('Original Image Histogram');

subplot(4, 4, 8);

imhist(cc);

title('5\*5 histogram');

1. **Use appropriate Matlab function calls to repeat the steps a) and b). Use if/else statements to compare each of your processing results with the corresponding result obtained by calling a Matlab function and display a message to indicate whether your processing result is the same as the one got from Matlab. If the comparison shows the difference between your implementation results and the Matlab’s results, display the difference image(s) with the appropriate titles.**

**Code :**

Img = imread('download.jpg');

Img = rgb2gray(Img);

temp = imread('download.jpg');

b=temp;

prompt = 'Enter your choice :\n1.Mean Filter \n2.Median Filter \n';

xx = input(prompt);

if(xx==1)

% mean filter using builtin function

Kaverage = filter2(fspecial('average',3),Img)/255;

figure,imshow(Kaverage),title('Mean filer from builtin function');

mean(b);

disp('Filtering is almost same from both functions');

else

%median filer using builtin function

Kmedian = medfilt2(Img);

imshow(Kmedian),title('Median filer from builtin function');

median(b);

disp('Filtering is almost similar from both functions');

end

function mean(b)

[m,n,colormap]=size(b);

x=b(:,:,1);

y=b(:,:,2);

z=b(:,:,3);

% for 3 \* 3

for i=1:m-2

sum1 = 0;

sum2 = 0;

sum3=0;

for j=1:n-2

for r=i:(i+2)

for s=j:(j+2)

sum1 = int16(sum1) + int16(x(r,s));

sum2 = int16(sum2) + int16(y(r,s));

sum3 = int16(sum3) + int16(z(r,s));

end

end

avg1 = int16(sum1 / 9);

avg2 = int16(sum2 / 9);

avg3 = int16(sum3 / 9);

x(i+1,j+1) = avg1;

y(i+1,j+1) = avg2;

z(i+1,j+1) = avg3;

sum1=0;sum2=0; sum3 = 0;

end

end

c(:,:,1)=x;

c(:,:,2)=y;

c(:,:,3)=z;

figure,imshow(rgb2gray(c)),title('Mean filer from self written function');

end

function median(b)

[m,n,colormap]=size(b);

x=b(:,:,1);

y=b(:,:,2);

z=b(:,:,3);

for i=1:m-2

for j=1:n-2

tt1 = zeros(1, 9);

tt2 = zeros(1, 9);

tt3 = zeros(1, 9);

kk = 1;

for r=i:(i+2)

for s=j:(j+2)

tt1(kk) = x(r,s);

tt2(kk) = y(r,s);

tt3(kk) = z(r,s);

kk = kk + 1;

end

end

t1 = sort(tt1);

t2 = sort(tt2);

t3 = sort(tt3);

x(i+1,j+1) = t1(5);

y(i+1,j+1) = t2(5);

z(i+1,j+1) = t3(5);

end

end

c(:,:,1)=x;

c(:,:,2)=y;

c(:,:,3)=z;

figure,imshow(rgb2gray(c)),title('Median filer from self written function');

end

**Assignment-7**

1. **Take an input image combining any one alphabet try to extract some basic features and match with some other images containing alphabet (same/different).**

**Code :**

BW = imread('download.png');

figure,imshow(BW),title('Original Image');

%Remove interior pixels to leave an outline of the shapes.

BW2 = bwmorph(BW,'remove');

figure,imshow(BW2),title('Remove interior pixels');

%Get the image skeleton

BW3 = bwmorph(BW,'skel',Inf);

figure,imshow(BW3),title('image skelton');

%erode image

se = offsetstrel('ball',5,5);

erodedI = imerode(BW,se);

figure,imshow(erodedI),title('erode image');

%dilate image

se = strel('line',11,90);

BW5 = imdilate(BW,se);

figure,imshow(BW5),title('dilate image');

% gradient of the image

[Gmag, Gdir] = imgradient(BW,'prewitt');

figure,imshowpair(Gmag, Gdir, 'montage');

title('Gradient Magnitude, Gmag (left), and Gradient Direction, Gdir (right), using Prewitt method');

**Assignment-8**

Write a program in MATLAB for detecting Edges using following operators.

1. **Prewitt Operator**

**Code :**

openImage = rgb2gray(imread('apple.jpg'));

openImage = im2double(openImage); %// Convert to double

%// Corrected masks

b=[-1 -1 -1;0 0 0;1 1 1]/6;

c=[-1 0 1; -1 0 1; -1 0 1]/6;

Gx=abs(conv2(openImage,c,'same'));

Gy=abs(conv2(openImage,b,'same'));

G = sqrt( Gx.^2 + Gy.^2);

out = G > 0.08995; %// Threshold image

figure;

imshow(out),title('prewit operator');

1. **Sobel Operator**

**Code :**

A=imread('apple.jpg');

B=rgb2gray(A);

C=double(B);

for i=1:size(C,1)-2

for j=1:size(C,2)-2

%Sobel mask for x-direction:

Gx=((2\*C(i+2,j+1)+C(i+2,j)+C(i+2,j+2))-(2\*C(i,j+1)+C(i,j)+C(i,j+2)));

%Sobel mask for y-direction:

Gy=((2\*C(i+1,j+2)+C(i,j+2)+C(i+2,j+2))-(2\*C(i+1,j)+C(i,j)+C(i+2,j)));

B(i,j)=sqrt(Gx.^2+Gy.^2);

end

end

figure,imshow(B); title('Sobel gradient');

1. **Roberts Operator**

**Code :**

img = rgb2gray(imread('apple.jpg'));

edge(img,'Roberts'),title('roberts operator');

s