CSE 5524 - Homework #2
09/09/2013
Manjari Akella

1) Write a MATLAB function to compute and display the 2D Gaussian derivative masks Gx and Gy for a given sigma (see class notes). Note: each mask is a square 2D matrix.

- Used the formula for first derivative Gaussians in notes
- Removed –ve sign from the formulas to orient the system similar to Cartesian system
- Output surface plots are at sigma = 6

Output

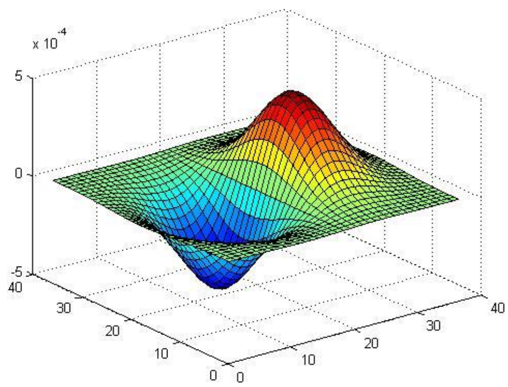


Figure 1: x-direction

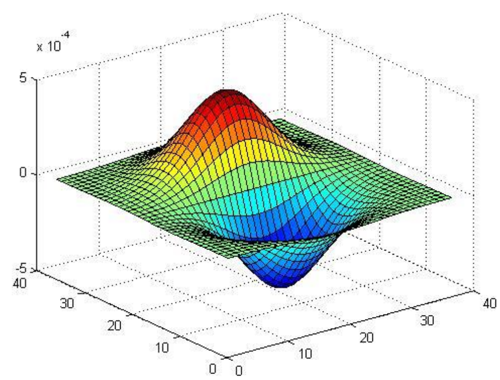
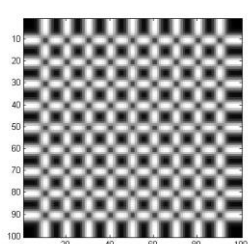


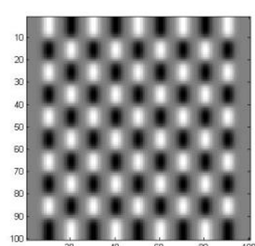
Figure 2: y-direction

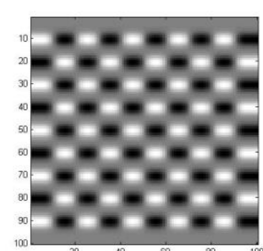
2) Compute and display the gradient magnitude of an image (search the web for an interesting image; convert to grayscale if necessary; make sure to upload image with code).

- Tested on a 2 images
- Checkerboard image showed horizontal and vertical band for y and x direction respectively

Output







- Chose another photo (a group photo) because it had a lot of detail
- The gradient magnitude caught not only the edges of all people in the picture but also small artifacts like ear rings and hairclips
- Strong and bright edges were observed in the group photo

Output

Figure 3: Original Image

Figure 4: Gradient Magnitude

3) Threshold and display the magnitude image with different threshold levels.

- Did thresholding only for group photo
- Checked for threshold values of >10%,30%,50%,70%,90% of the maximum intensity in image and displayed results for each
- Increasing value of the threshold leads to fewer edges as expected
- Output here shows only at 10% of maximum value, program shows the entire output

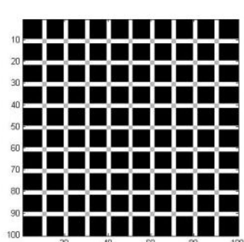
<u>Output</u>

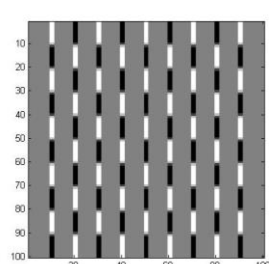
Figure 5: Threshold = 10% of Max value

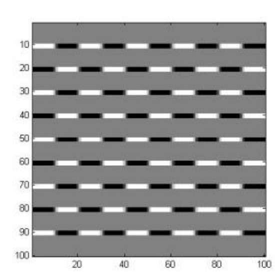
4) Compare the above results with the Sobel masks.

- Tried searching for direct functions. Found imgradient() in 2013's documentation of MATLAB which returns both magnitude and direction
- For the checker board image, thinner and sharper strips were observed
- Didn't check checkerboard for threshold values

<u>Output</u>







- For the group photo, edges obtained were fainter and grainer than the previous method
- Printed only at threshold value of 10% of maximum intensity, rest displayed in code
- Threshold at 90% of maximum value didn’t preserve any pixel values

Output

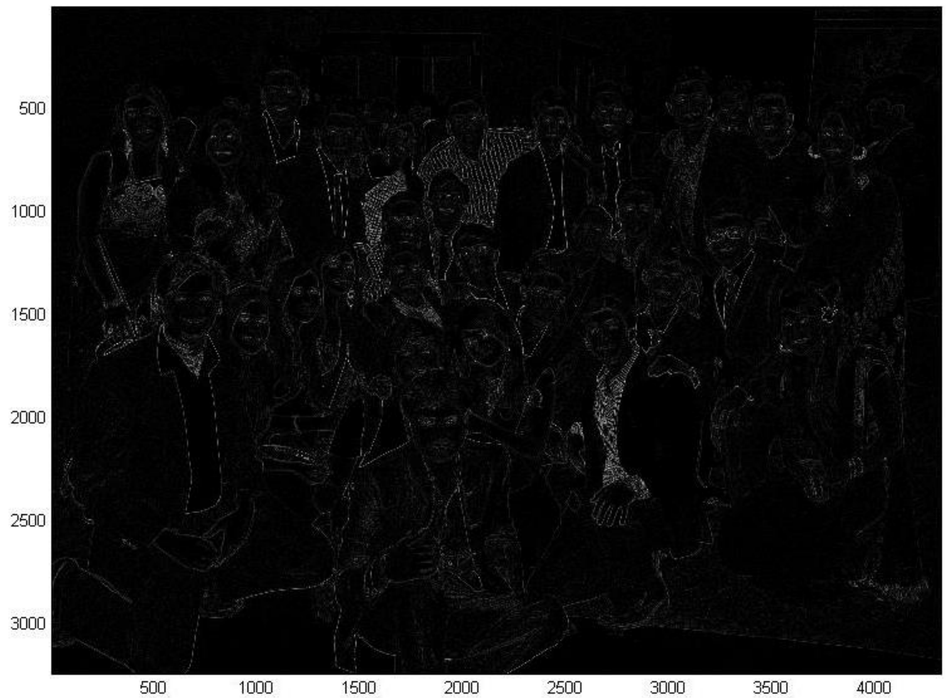


Figure 6: Gradient Magnitude using Sobel

Figure 7: Threshold = 10% of max value

5) Run the MATLAB canny edge detector, edge(Im,'canny'), on your image and display the default results.

- Default setting of canny detector on group photo doesn’t provide with much information
- Can’t immediately make out what the picture is representing

Output

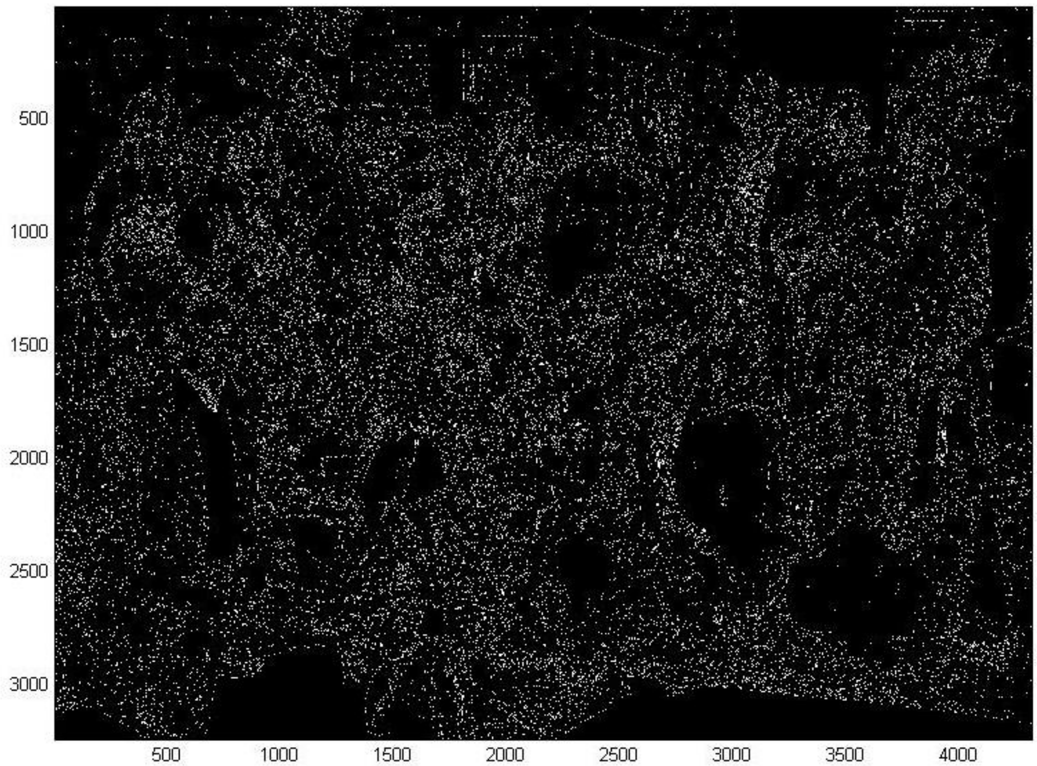


Figure 8: Canny Detector default settings - Group photo

- Also tried with a flower image having less detail
- Can identify easily what the picture is representing

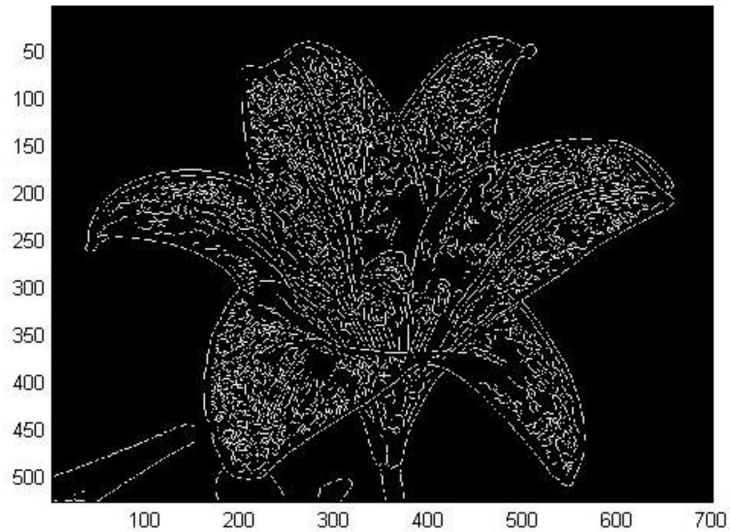


Figure 9: Canny default settings

6) Generate a 4-level Gaussian pyramid (original image is level-1) and the corresponding Laplacian pyramid of an image. Use the formula in the notes to first determine a viable image size, and create an image (e.g., crop if needed) to test the pyramid code. Use a=0.4 for the Gaussian mask – use separable masks.

- Started with a 385x385 image (M_r, M_c = 96, N=2 for the formula to determine a viable image size)
- Tested 2 images – lena and a peacock

Output

Lena

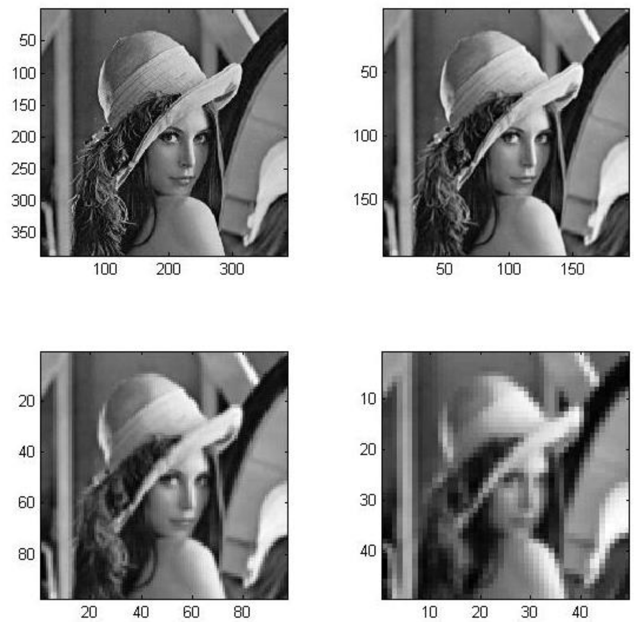


Figure 10: Gaussian Pyramid

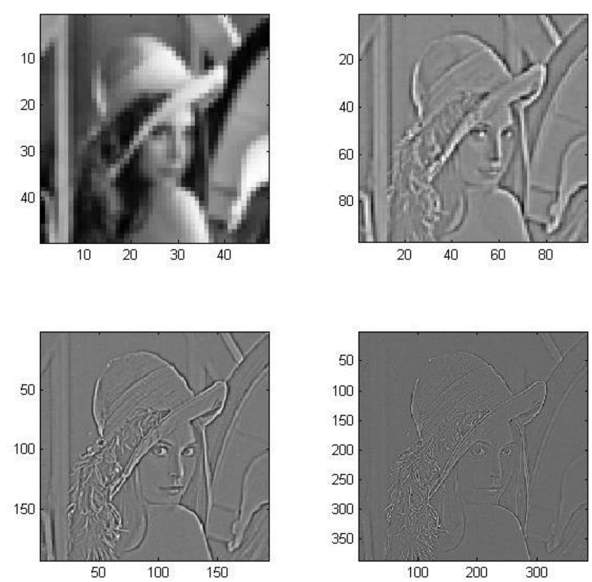


Figure 11: Laplacian Pyramid

Peacock

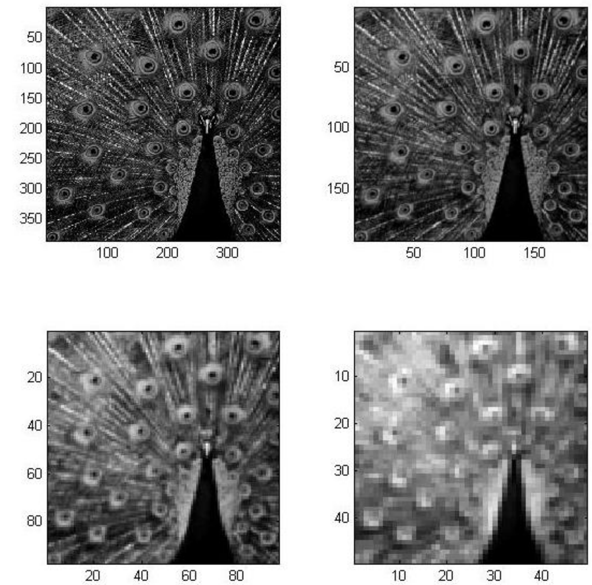


Figure 12: Gaussian Pyramid

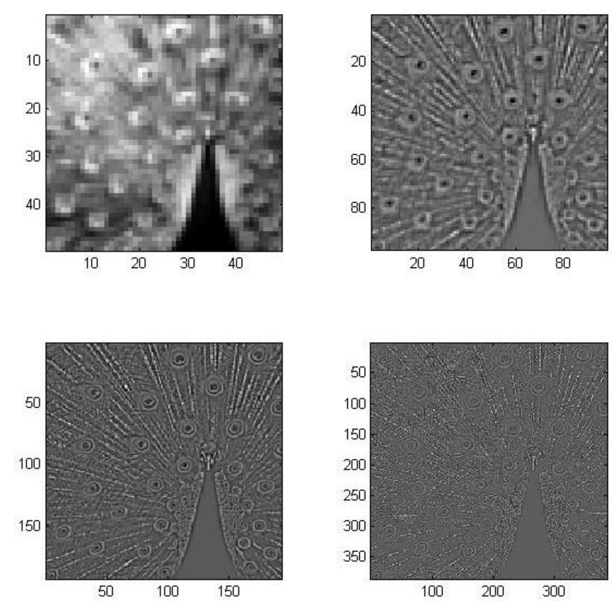


Figure 13: Laplacian Pyramid

CODE

1). gaussDeriv2D function

```
function [ Gx,Gy ] = gaussDeriv2D( sigma )
% Creates a 2D Gaussian Derivative mask in x and y directions
% Returns and displays them
range = ceil(3*sigma);
hsize = 2*ceil(3*sigma)+1;
for i = 1:hsize
    for j = 1:hsize
        x = i-range-1;
        y = j-range-1;
        t_1=x/(2*pi*(sigma^4));
        t_2=exp(-(x^2+y^2)/(2*sigma*sigma));
        Gx(j,i) = t_1*t_2;
    end
end

%generate a 2-D Gaussian kernel along y direction
Gy=Gx';

figure('Name','Q1: Gx (x-direction)','NumberTitle','off'),surf(Gx);
figure('Name','Q1: Gy (y-direction)','NumberTitle','off'),surf(Gy);
end
```

2). reduce function for Gaussian pyramid

```
function [ I2 ] = reduce_G( I1,w1,w2 )
%Reduction to lower levels (Gaussian Pyramid)
I = imfilter(I1,w1,'replicate');
I2full = imfilter(I,w2,'replicate');
I2 = I2full(1:2:size(I2full,1),1:2:size(I2full,2));
end
```

3). expand function for Laplacian pyramid

```
function [ I ] = expand( E2 )
% Interpolate and expand (Laplaican Pyramid)
I = zeros(2*size(E2,1)-1,2*size(E2,2)-1,'double');
% Fill odd rows and columns in new image using old one
x=0;
for i=1:2:size(I,1)
y=0;
    for j=1:2:size(I,2)
        I(i,j)= E2(i-x,j-y);
        y=y+1;
    end
    x=x+1;
end

% Interpolate even columns
for i=1:size(I,1)
    for j=1:size(I,2)
        if ((I(i,j)==0)&&(mod(i,2)~=0))
```

```
            a = I(i,j-1);
            b = I(i,j+1);
            I(i,j)= (a+b)/2;
        end
    end
end

% Interpolate even rows
for i=1:size(I,1)
    for j=1:size(I,2)
        if (I(i,j)==0)
            a = I(i-1,j);
            b = I(i+1,j);
            I(i,j)= (a+b)/2;
        end
    end
end

end
```

4). HW2.m script

```
% Manjari Akella
% CSE5524 - HW2
% 09/09/2013

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 1

refresh();
sigma = input('Enter the value of sigma\n');
[Gx,Gy] = gaussDeriv2D(sigma);
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 2

close all;
%  For checkerboard image and a group photograph
pics = {'my_pictures/1.bmp','my_pictures/2.jpg'};
for i=1:2
    close all;
%     Group photograph is not grayscale
    if (i==2)
        GIm = rgb2gray(imread(pics{i}));
        Im = double(GIm);
    else
        GIm = imread(pics{i});
        Im = double(GIm);
    end
    figure('Name','Q2:Original
Image','NumberTitle','off'),imagesc(uint8(Im));
    axis('image');
    colormap('gray');
    gxIm = imfilter(Im,Gx,'replicate');
```

```
    figure('Name','Q2:x-gradient','NumberTitle','off'),imagesc(gxIm);
    axis('image');
    colormap('gray');
    gyIm = imfilter(Im,Gy,'replicate');
    figure('Name','Q2:y-gradient','NumberTitle','off'),imagesc(gyIm);
    axis('image');
    colormap('gray');
    magIm = sqrt(gxIm.^2+gyIm.^2);
    figure('Name','Q2:Gradient
Magnitude','NumberTitle','off'),imagesc(magIm);
    axis('image');
    colormap('gray');
    pause;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 3

close all;
max_val = max(max(magIm));
% Greater than percentage of max value
for i=0.1:0.2:0.9
    tIm = magIm>(i*max_val);
    figure('Name',strcat('Q3:Threshold ',num2str(i*100),'
percent'),'NumberTitle','off'),imagesc(uint8(tIm));
    colormap('gray');
    axis('image');
end
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 4

refresh();
%  For checkerboard image and a group photograph
pics = {'my_pictures/1.bmp','my_pictures/2.jpg'};
for i=1:2
    close all;
%     Group photograph is not grayscale
    if (i==2)
        GIm = rgb2gray(imread(pics{i}));
        Im = double(GIm);
    else
        GIm = imread(pics{i});
        Im = double(GIm);
    end
    fx = -fspecial('sobel')';
    fxIm = imfilter(Im,fx,'replicate');
    figure('Name','Q4:Sobel x-direction','NumberTitle','off'),imagesc(fxIm);
    colormap('gray');
    axis('image');
    fy = fx';
    fyIm = imfilter(Im,fy,'replicate');
    figure('Name','Q4:Sobel y-direction','NumberTitle','off'),imagesc(fyIm);
    colormap('gray');
    axis('image');
```

```
    magIm = sqrt(fxIm.^2+fyIm.^2);
    figure('Name','Q4:Sobel Gradient
Magnitude','NumberTitle','off'),imagesc(magIm);
    colormap('gray');
    axis('image');
    pause;
    close all;
end
    max_val = max(max(magIm));
    % Greater than percentage of max value
    for i=0.1:0.2:0.9
        tIm = magIm>(i*max_val);
        figure('Name',strcat('Q4:Threshold ',num2str(i*100),'
percent'),'NumberTitle','off'),imagesc(tIm);
        colormap('gray');
        axis('image');
    end
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 5

refresh();

% Group photo (more detail in image)
% Im = double(rgb2gray(imread('my_pictures/2.jpg')));

% Less detail in image
Im = double(rgb2gray(imread('my_pictures/3.jpg')));
EGIm = edge(Im,'canny');
figure('Name','Q5_Canny default result','NumberTitle','off'),imagesc(EGIm);
colormap('gray');
axis('image');
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 6

refresh();
a=0.4;
w1=[0.25-0.5*a,0.25,a,0.25,0.25-0.5*a];
w2 = w1';

%Reduce operation
% Level 1
Im=double(rgb2gray(imread('my_pictures/4.jpg')));
% Im=double(rgb2gray(imread('my_pictures/lena.jpg')));
figure('Name','Q6_Gaussian_Pyramid','NumberTitle','off'),subplot(2,2,1),image
sc(Im);
colormap('gray');
axis('image');

% Level 2
I2 = reduce_G(Im,w1,w2);
subplot(2,2,2),imagesc(I2);
```

```
colormap('gray');
axis('image');

% Level 3
I3 = reduce_G(I2,w1,w2);
subplot(2,2,3),imagesc(I3);
colormap('gray');
axis('image');

% Level 4
I4 = reduce_G(I3,w1,w2);
subplot(2,2,4),imagesc(I4);
colormap('gray');
axis('image');

% Expand operation
% Level 4
E4 = I4;
figure('Name','Q6_Laplacian_Pyramid','NumberTitle','off'),subplot(2,2,1),imag
esc(E4);
colormap('gray');
axis('image');

% Level 3
I = expand(I4);
E3 = I3-I;
subplot(2,2,2),imagesc(E3);
colormap('gray');
axis('image');

% Level 2
I = expand(I3);
E2 = I2-I;
subplot(2,2,3),imagesc(E2);
colormap('gray');
axis('image');

% Level 1
I = expand(I2);
E1 = Im-I;
subplot(2,2,4),imagesc(E1);
colormap('gray');
axis('image');

% Reconstruction
L3 = expand(E4);
temp = L3+E3;
L2 = expand(temp);
temp = L2+E2;
L1 = expand(temp);
result = L1+E1;

figure('Name','Q6_Reconstructed','NumberTitle','off'),imagesc(result);
colormap('gray');
axis('image');
```

```
figure('Name','Q6_Original','NumberTitle','off'),imagesc(Im);
colormap('gray');
axis('image');

%%%%%%%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%
```