Homework Assignment #2

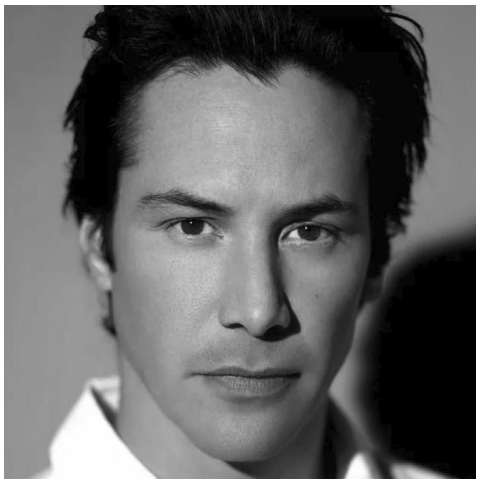
Computer Vision for HCI - CSE 5524 Shailesh Tripathi (tripathi.52)

Q1.

```
sigma=1.0;
faceIm=double(imread('keanu.jpg'));

% MATLAB does not recommends 'imfilter' for gaussian filter as
% 'imgaussfilt' is more optimized than the other. The whole operation can
% be performed in one step instead of two.
%G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
%gIm = imfilter(faceIm, G, 'replicate');

gIm = imgaussfilt(faceIm,sigma,'FilterSize', 2*ceil(3*sigma)+1, 'Padding',
'replicate');
% NOTE: Using imgaussfilt instead of fspecial gives very close yet unequal
% results . Relative error = 1.7053e-15. This much error is acceptable.

% Display the results
imshow(gIm/255);  % double images need range of 0-1

imwrite(uint8(gIm), 'gIm.bmp');
```

Gaussian filter was applied to the image with varying sigma values. Most of the people were able to recognize Keanu Reeves even when the image was blur.
When tried with different people the different sigma values for which people were able to identify were 20, 11, 16 and 10.

Original Image



Gaussian filter with sigma = 5.0

Q2. Gaussian derivative

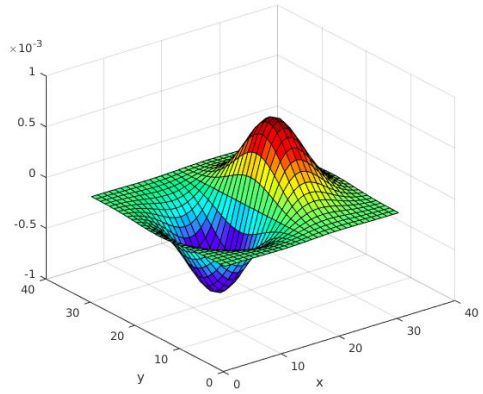
```
% Use gaussDeriv2D to get the Gaussian Derivative Mask
[Gx, Gy] = gaussDeriv2D(sigma); % Defined at the end

% Plot the surface graph
surf(Gx);
colormap(jet);
xlabel('x')
ylabel('y')
title('Gx');
pause;

% Plot 2D image
imagesc(Gx); colormap('gray');
pause;

% Plot the surface graph
surf(Gy);
colormap(jet);
xlabel('x')
ylabel('y')
title('Gy');
pause;

% Plot 2D image
imagesc(Gy); colormap('gray');
pause;


function [Gx, Gy] = gaussDeriv2D(sigma)
      % Calculate appropriate length of the square mask.
      % Mask will have length (3*sigma) points on both sides of origin
      length = ceil(3*sigma);

      % Initialize size of mask
      Gx = zeros(2*length +1, 2*length +1);
      Gy = zeros(2*length +1, 2*length +1);

      for x = -length : length
      for y = - length : length
            % Assume x0 = 0 and y0 = 0 (both mean x and mean y are zero)
            % Calculate the component common to both Gx and Gy
            e = exp(-(x*x + y*y)/(2*sigma*sigma)) ;
            e = e / (2 * pi * (sigma^4));
```

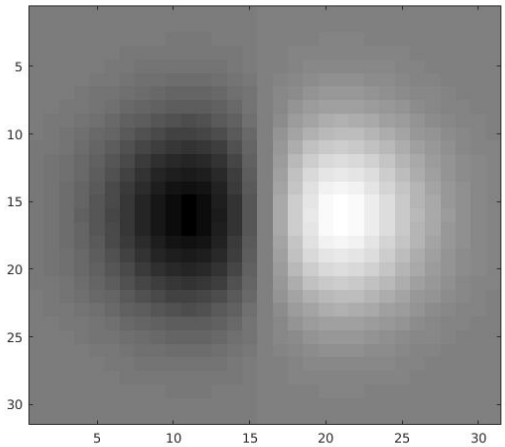
```
            % Multiply x and e for Gx
            % Note that x-coordinate would be the 2nd dimension in MATLAB
            % and y-coordinate would the first coordinate.
            Gx(y + length +1, x + length +1) = x * e;

            % Multiply y and e for Gy
            Gy(y + length +1, x + length +1) = y * e;
        end
        end
end
```

The gaussian derivative plotted are:

Gx

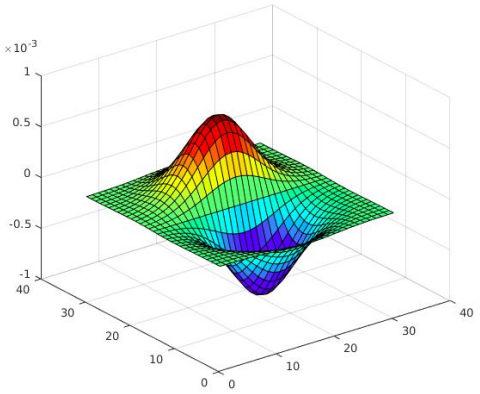
$$\frac{\partial g(x,y;\sigma)}{\partial x} = \frac{+(x-x_o)}{2\pi\sigma^4} e^{-\frac{(x-x_o)^2+(y-y_o)^2}{2\sigma^2}}$$

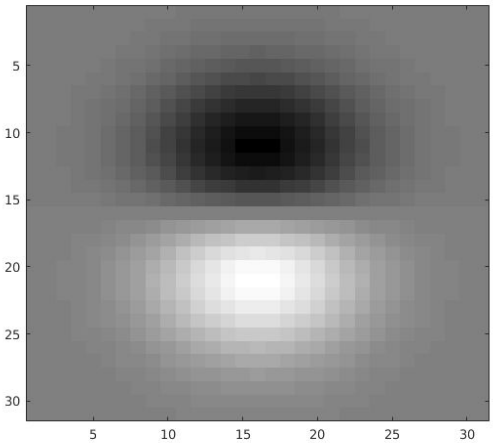




Gy

$$\frac{\partial g(x,y;\sigma)}{\partial y} = \frac{+(y-y_o)}{2\pi\sigma^4} e^{-\frac{(x-x_o)^2+(y-y_o)^2}{2\sigma^2}}$$





*The Gy image looks inverted in y-axis but is actually correct because the direction in which y is increasing in 3D plot vs 2D plot is different. Note the y axis marking.

Q3.

Gaussian Derivative in image of traffic on road. (Sigma = 1.0)

```
% Gaussian derivative is used in autonomous vehicles to detect other
% vehicles. Here is an example.
rgbIm=(imread('car.jpg'));

% Use gaussian derivative with smaller sigma value to get better results
sigma = 1.0;
[Gx, Gy] = gaussDeriv2D(sigma);

% Convert RGB image to grayscale
Im = double(rgb2gray(rgbIm));

% Apply Gx filter
gxIm = imfilter(Im, Gx, 'replicate');
% Apply Gy filter
gyIm = imfilter(Im, Gy, 'replicate');

% Compute magnitude to get the resultant image.
magIm = sqrt(gxIm.^2 + gyIm.^2);

% Display the image. Double images need range 0-1
imagesc(magIm/255);
axis('image');
colormap('gray');
```

Original Image:

Gaussian Derivative Filter applied:

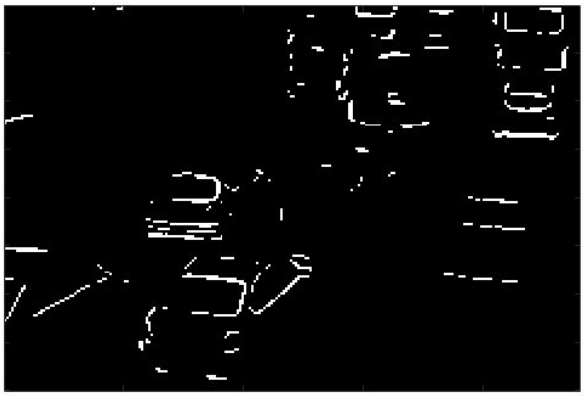
Q4. Threshold applied to image after Gaussian Derivative Filter: (Sigma = 1.0)

```
% Set threshold level
T = 30;

% Convert grayscale image to bianry using a threshold value.
tIm = magIm > T;

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

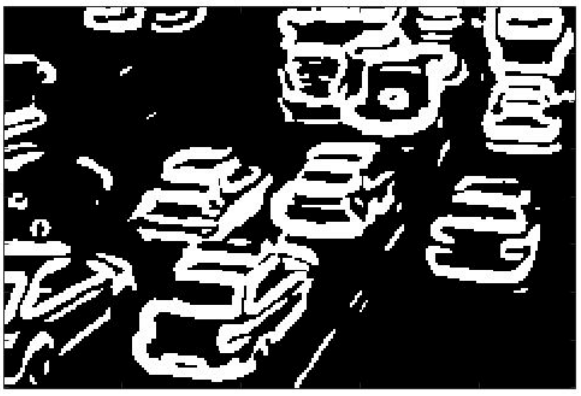
Threshold = 10

Threshold = 30

Threshold = 50



Sigma = 2.0 , Threshold = 10



Q5. Sobel mask applied with different threshold:

```
% Apply sobel mask. An ideal mask should be normalized to 1. Hence divided
% by 8
Fx = -fspecial('sobel')/8;
fxIm = imfilter(Im,Fx);
Fy=-fspecial('sobel');
fyIm = imfilter(Im,Fy)/8;

% Calculate magnitude
magIm = sqrt(fxIm.^2 + fyIm.^2);

% Different threshold for sobel mask
T = 50;
tIm = magIm > T;
imagesc(tIm);
axis('image');
colormap('gray');
```

Threshold = 30

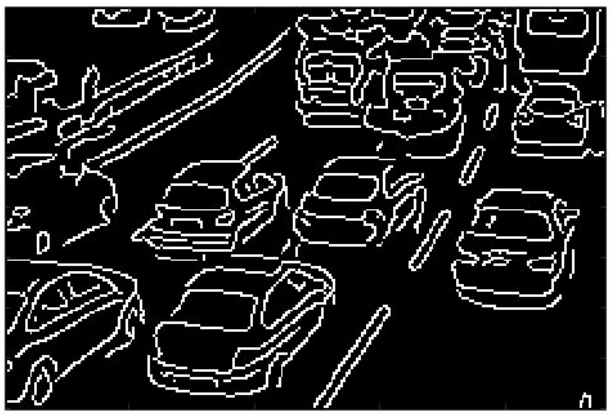
Threshold = 50

Threshold = 70

Q6.

Canny edge detector:

```
% Canny Edge. Using the in-built function provided by matlab
tIm = edge(Im, 'canny');
imagesc(tIm);
axis('image');
colormap('gray');
```



Comparison: Clearly Canny edge detector works best as it gives thinner edges without missing out the details and difficult edges. As can be seen from Gaussian derivative filter and sobel filter, either the edges are thick(low threshold) or the lighter edges are vanished (high threshold).
Also, Gaussian derivative filter may sometimes work better than sobel with appropriate values of sigma and threshold.