Worksheet completed with Octave.

Question 1

1. Convolution Theorem

   The convolution theorem states:

$$\mathcal{F}[h(x) * I(x)] = H[n]I[n]$$

$$\begin{aligned}
h(x) * I(x) &= \sum_{u=0}^{N-1} h\left(x-u\right) I(u) && (1)\\
&= \sum_{u=0}^{N-1} [\frac{1}{N}\sum_{n=0}^{N-1} H(n)e^{j2\pi(x-u)n/N}][\frac{1}{N}\sum_{l=0}^{N-1} I(l)e^{j2\pi ul/N}] && (2)\\
&= \frac{1}{N}\sum_{n=0}^{N-1} H(n)\sum_{l=0}^{N-1} I(l)\frac{1}{N}\sum_{u=0}^{N-1} e^{j2\pi(x-u)n/N}e^{j2\pi ul/N} && (3)\\
&= \frac{1}{N}\sum_{n=0}^{N-1} H(n)\sum_{l=0}^{N-1} I(l)e^{j2\pi xn/N}\frac{1}{N}\sum_{u=0}^{N-1} e^{j2\pi u(l-n)/N} && (4)\\
&= \frac{1}{N}\sum_{n=0}^{N-1} H(n)\sum_{l=0}^{N-1} I(l)e^{j2\pi xn/N}\delta(l-n) && (5)\\
&= \frac{1}{N}\sum_{n=0}^{N-1} [H(n)I(n)]e^{j2\pi xn/N} && (6)
\end{aligned}$$

   This is the inverse transform of $H(n)I(n)$, and the corresponding forward transform is

$$\mathcal{F}[h(x) * I(x)] = \sum_{x=0}^{N-1}[h(x) * I(x)e^{-j2\pi xn/N} = H[n]I[n]$$

2. Laplacian Operator

   Show that the Laplacian is in fact rotation invariant. Essentially this boils down to showing

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{\partial^2 f}{\partial u^2} + \frac{\partial^2 f}{\partial v^2}$$

   where

$$u = xcos\theta + ysin\theta$$

$$v = -xsin\theta + ycos\theta$$

So,

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial x} = \frac{\partial f}{\partial u}cos\theta - \frac{\partial f}{\partial v}sin\theta$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial u}cos\theta - \frac{\partial f}{\partial v}sin\theta\right) = \frac{\partial}{\partial x}\frac{\partial f}{\partial u}cos\theta - \frac{\partial}{\partial x}\frac{\partial f}{\partial v}sin\theta$$

Now the problem is to compute $\frac{\partial}{\partial x}\frac{\partial f}{\partial u}$ and $\frac{\partial}{\partial x}\frac{\partial f}{\partial v}$.We can think of $\frac{\partial}{\partial x}\frac{\partial f}{\partial u}$ as $\frac{\partial}{\partial u}\frac{\partial f}{\partial x}$.

$$\frac{\partial}{\partial u}\frac{\partial f}{\partial x} = \frac{\partial}{\partial u}\left(\frac{\partial f}{\partial u}cos\theta - \frac{\partial f}{\partial v}sin\theta\right) = \frac{\partial^2 f}{\partial u^2}cos\theta - \frac{\partial f}{\partial u \partial v}sin\theta$$

$$\frac{\partial}{\partial v}\frac{\partial f}{\partial x} = \frac{\partial}{\partial v}\left(\frac{\partial f}{\partial u}cos\theta - \frac{\partial f}{\partial v}sin\theta\right) = \frac{\partial f}{\partial u \partial v}cos\theta - \frac{\partial^2 f}{\partial v^2}sin\theta$$

Now plug this back in to $\frac{\partial^2 f}{\partial x^2}$ giving,

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2} &= \left(\frac{\partial^2 f}{\partial u^2}cos\theta - \frac{\partial f}{\partial u \partial v}sin\theta\right)cos\theta - \left(\frac{\partial f}{\partial u \partial v}cos\theta - \frac{\partial^2 f}{\partial v^2}sin\theta\right)sin\theta && (7)\\
&= \frac{\partial^2 f}{\partial u^2}cos\theta^2 - 2\frac{\partial f}{\partial u \partial v}sin\theta cos\theta + \frac{\partial^2 f}{\partial v^2}sin\theta^2 && (8)
\end{aligned}$$

Repeat the process for $\frac{\partial^2 f}{\partial y^2}$ and add together.

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial^2 f}{\partial u^2}sin\theta^2 + 2\frac{\partial f}{\partial u \partial v}sin\theta cos\theta + \frac{\partial^2 f}{\partial v^2}cos\theta^2$$

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} &= \frac{\partial^2 f}{\partial u^2}cos\theta^2 - 2\frac{\partial f}{\partial u \partial v}sin\theta cos\theta + \frac{\partial^2 f}{\partial v^2}sin\theta^2 + \frac{\partial^2 f}{\partial u^2}sin\theta^2 + 2\frac{\partial f}{\partial u \partial v}sin\theta cos\theta + \frac{\partial^2 f}{\partial v^2}cos\theta^2 && (9)\\
&= \frac{\partial^2 f}{\partial u^2}\left(cos\theta^2 + sin\theta^2\right) + \frac{\partial^2 f}{\partial v^2}\left(cos\theta^2 + sin\theta^2\right) && (10)\\
&= \frac{\partial^2 f}{\partial u^2} + \frac{\partial^2 f}{\partial v^2} && (11)
\end{aligned}$$

3. Condition of Linear Variation

   Prove that their argument is mathematically sound, that is, under the condition they assume, the Laplacian picks up the direction perpendicular to the local direction of the edge.

Let $f(x, y)$ be a real-valued, twice continuously differentiable function on the plane. Let l be an open line segment along the axis $x = 0$. Then the two conditions

$$\text{(i)}\ \nabla^2 f = 0 \text{ on } l$$

$$\text{and (ii)}\ \partial^2 f/\partial x^2 = 0 \text{ on } l$$

are equivalent if and only if $f(0, y)$ is constant or linear on l.

Condition: f is constant or linear along the local direction of the edge L.

Prove:

From Question 1(b), we know that the Laplacian is in fact rotation invariant. So, we can change the coordinate (x, y) to a new coordinate system (x', y'), L lies on the open line segment along the axis x'=0.

f(0, y') is linear on L, $\dfrac{\partial^2 f}{\partial y'^2} = 0$. From the Laplacian's rotation invariant, we can get

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{\partial^2 f}{\partial x'^2} + \frac{\partial^2 f}{\partial y'^2} = 0$$

There implies that $\dfrac{\partial^2 f}{\partial x'^2} = 0$ on L. Therefore, when f is constant or linear on along the local direction of the edge L, $\dfrac{\partial^2 f}{\partial x^2} + \dfrac{\partial^2 f}{\partial y^2} = 0$ is equivalent to the second derivate on the direction perpendicular to the local direction of the edge.

Conversely, if $\dfrac{\partial^2 f}{\partial x'^2} = \dfrac{\partial^2 f}{\partial x'^2} + \dfrac{\partial^2 f}{\partial y'^2} = 0$ on L, then $\dfrac{\partial^2 f}{\partial y'^2} = 0$ on L. So, we can get f(0, y') varies at most linearly on L. f is constant or linear along the local direction of the edge L.

Question 2

1. 2D Gaussian

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\dfrac{x^2+y^2}{2\sigma^2}}$$

Listing 1: make2DGaussian

```
function g = make2DGaussian(N, sigma)
% N is assumed to be odd, and so the origin (0,0) is positioned at indices
% (M+1,M+1) where N = 2*M + 1.
% creates gaussian kernel with side length N and a sigma of sigma
ax = linspace(-(N - 1.) / 2., (N - 1.) / 2., N);
[xx, yy] = meshgrid(ax, ax);
kernel = exp(-0.5 * (xx.^2 + yy.^2) /(sigma.^2));
g = kernel./sum(kernel(:));
% the same as this function h = fspecial('gaussian',[3,3],0.5)
% surf(xx,yy,g)
end
```

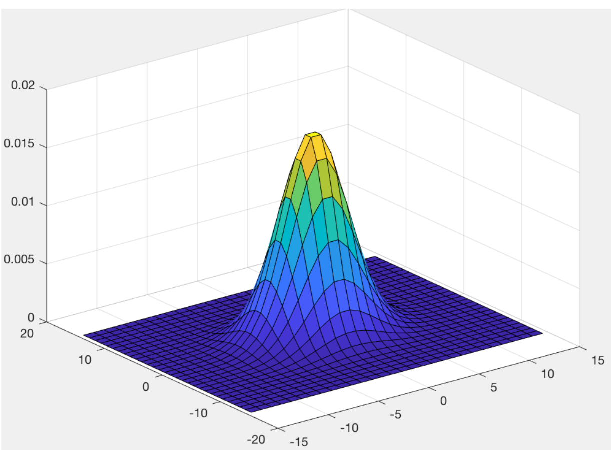

Figure 1: $Gaussian, N = 30, \sigma = 3$

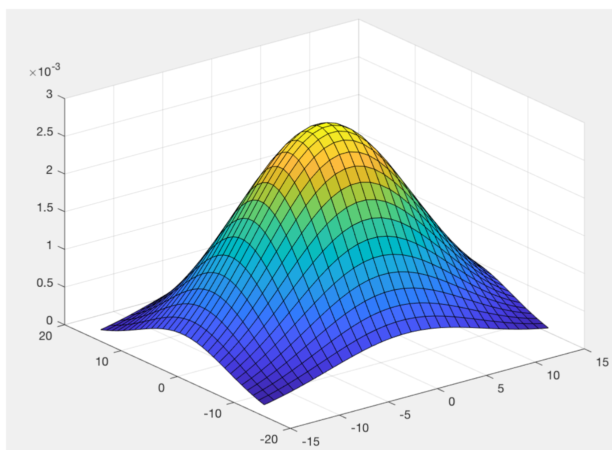

Figure 2: $Gaussian, N = 30, \sigma = 8$

2. 2D Laplacian of Gaussian

$$LoG_\sigma(x,y) = \frac{1}{\pi\sigma^4}\left(\frac{x^2+y^2}{2\sigma^2} - 1\right) e^{-\dfrac{x^2+y^2}{2\sigma^2}}$$

Listing 2: make2DLOG

```
function g = make2DLOG(N, sigma)
% N is assumed to be odd, and so the origin (0,0) is positioned at indices
% (M+1,M+1) where N = 2*M + 1.
% creates Laplacian of Gaussian with side length N and a sigma of sigma
ax = linspace(-(N - 1.) / 2., (N - 1.) / 2., N);
[xx, yy] = meshgrid(ax, ax);
kernel = exp(-0.5 * (xx.^2 + yy.^2) ./(sigma.^2));
LOG_t = kernel.*(xx.^2 + yy.^2 - 2* sigma^2)/(sigma^4 * sum(kernel(:)));
% make the filter sum to zero
LOG = LOG_t - sum(LOG_t(:))/N^2;
g=LOG;
% surf(xx,yy,g)
% the same as function g=fspecial('log',[3 3],0.5)
end
```

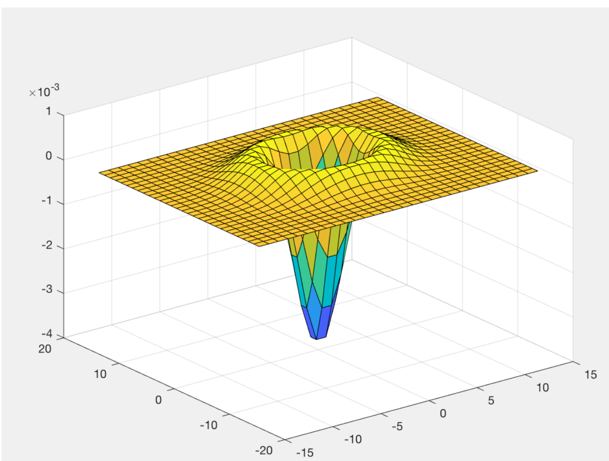

Figure 3: $LOG, N = 30, \sigma = 3$

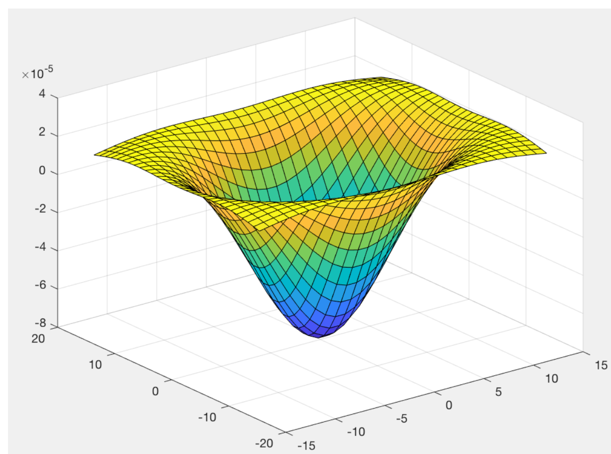

Figure 4: $LOG, N = 30, \sigma = 8$

3. 2D Gabor Filters Gabor filter is defined as the multiplication of a cosine/sine (even/odd) wave with a Gaussian windows.

$$Even : g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) cos\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

$$Odd : g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) sin\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

where

$$x' = xcos\theta + ysin\theta$$
$$y' = -xsin\theta + ycos\theta$$

Listing 3: make2DGabor

```
function [even, odd] = make2DGabor(N, lambda, angle)
% N is assumed to be odd, and so the origin (0,0) is positioned at indices
% (M+1,M+1) where N = 2*M + 1.
% lambda : wavelength of the Gabor filter
% angle : orientation of the Gabor filter
% Set sigma of Gaussian part of filter = wavelength lambda end
sigma = lambda;
ax = linspace(-(N - 1.) / 2., (N - 1.) / 2., N);
[x, y] = meshgrid(ax, ax);
% Rotation
x_angle=x*cos(angle)+y*sin(angle);
y_angle=-x*sin(angle)+y*cos(angle);
even= exp(-0.5*(x_angle.^2+y_angle.^2)./sigma^2).*cos(2*pi/lambda*x_angle);
odd= exp(-0.5*(x_angle.^2+y_angle.^2)./sigma^2).*sin(2*pi/lambda*x_angle);
% figure(1)
% surf(x,y,even)
% figure(2)
% surf(x,y,odd)
end
```

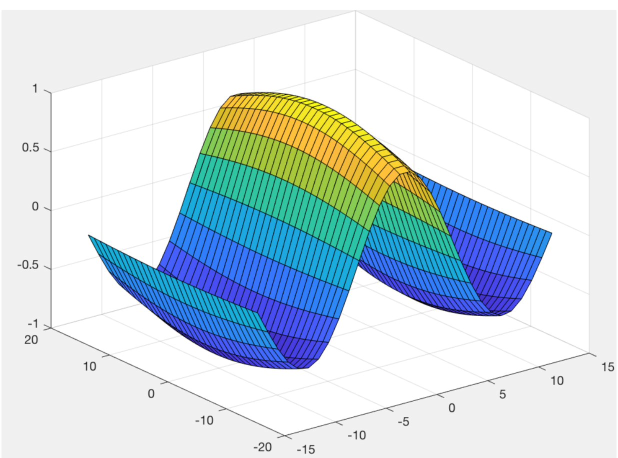

Figure 5: $Gabor, even, \lambda = 20, N = 30, angle = 0$

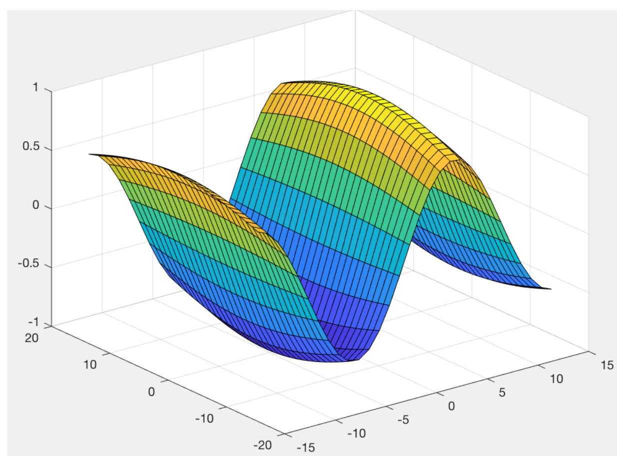

Figure 6: $Gabor, odd, \lambda = 20, N = 30, angle = 0$

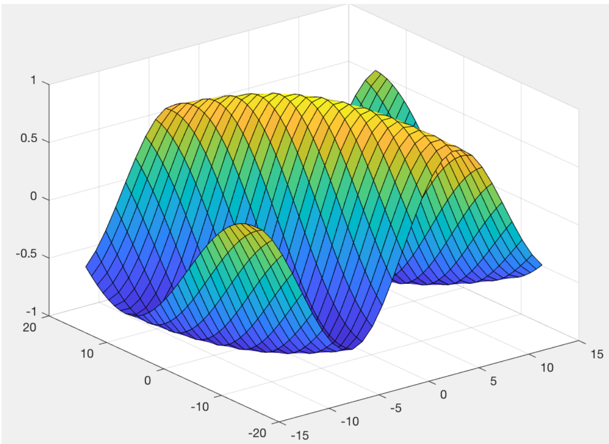

Figure 7: $Gabor, even, \lambda = 20, N = 30, angle = \pi/8$

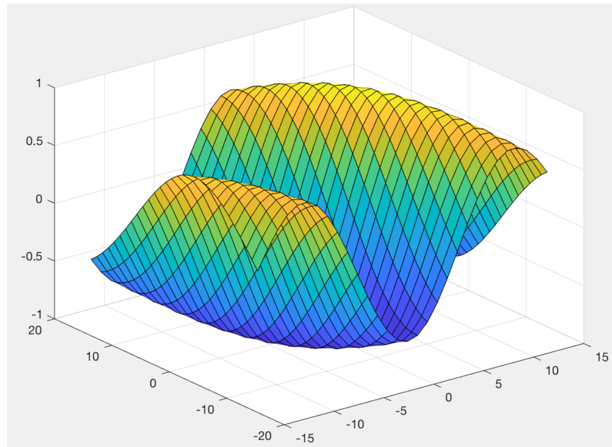

Figure 8: $Gabor, odd, \lambda = 20, N = 30, angle = \pi/8$

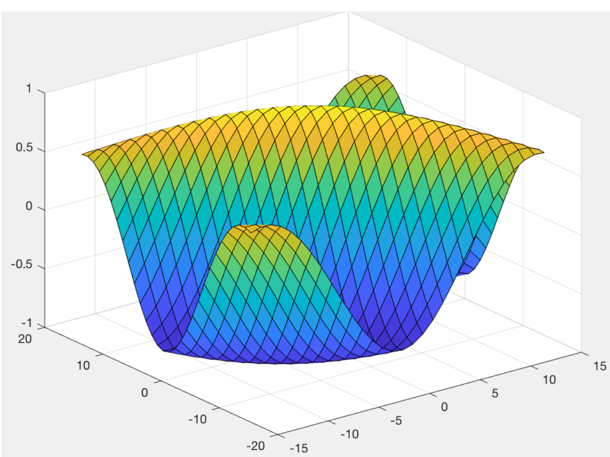

Figure 9: $Gabor, even, \lambda = 20, N = 30, angle = \pi/4$

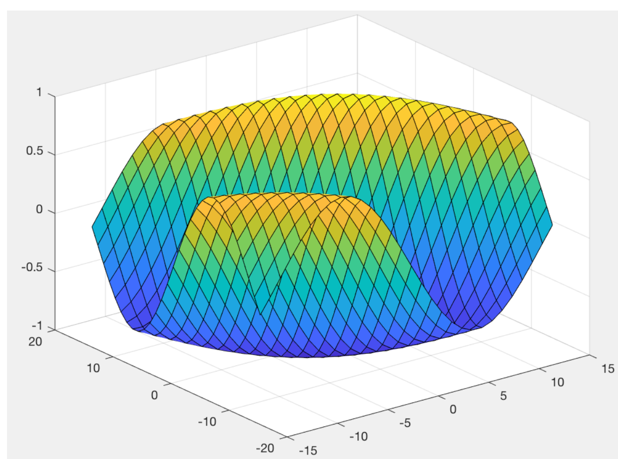

Figure 10: $Gabor, odd, \lambda = 20, N = 30, angle = \pi/4$

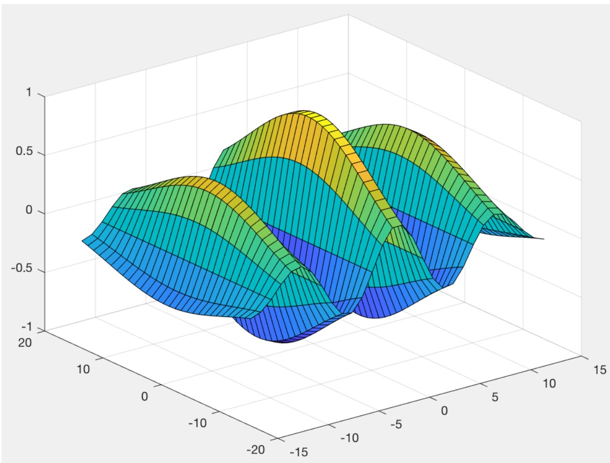

Figure 11: $Gabor, even, \lambda = 10, N = 30, angle = 0$

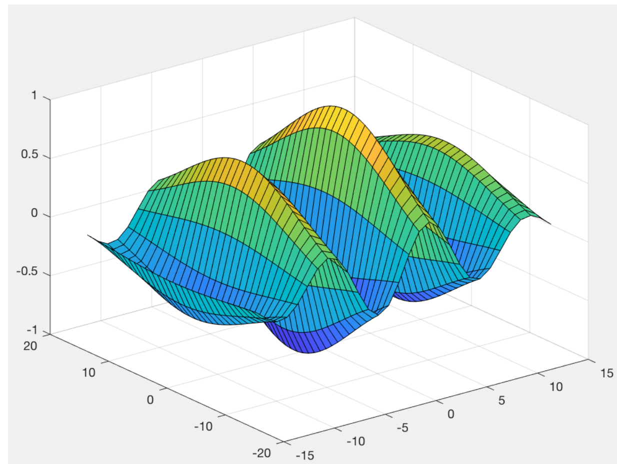

Figure 12: $Gabor, odd, \lambda = 10, N = 30, angle = 0$

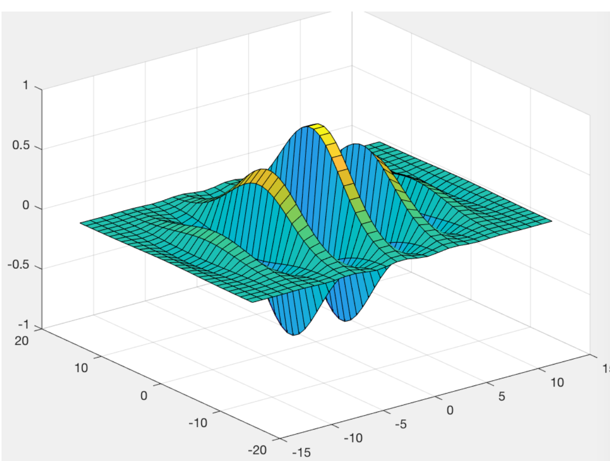

Figure 13: $Gabor, even, \lambda = 5, N = 30, angle = 0$

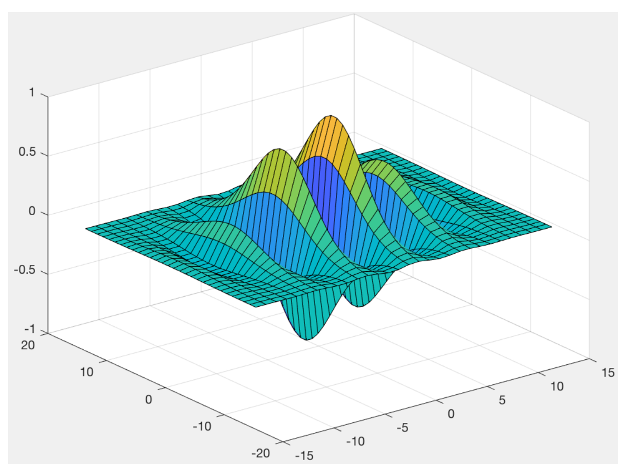

Figure 14: $Gabor, odd, \lambda = 5, N = 30, angle = 0$

Question 3

1. Convolve the 2D Laplacian of Gaussian filters from Question 2 b) with the input images

Listing 4: ConvolveLoG

```
clear
clc
sigma = 2; % ? parameter
N=15;
% Convolve the 2D Laplacian of Gaussian filters from Question 2 b) with the
    input images
I = imread('Paolina.jpg');
I = rgb2gray(I);
filtered_signal=conv2(I,make2DGaussian(N, sigma),'same');
figure(1)
imagesc(filtered_signal);
```

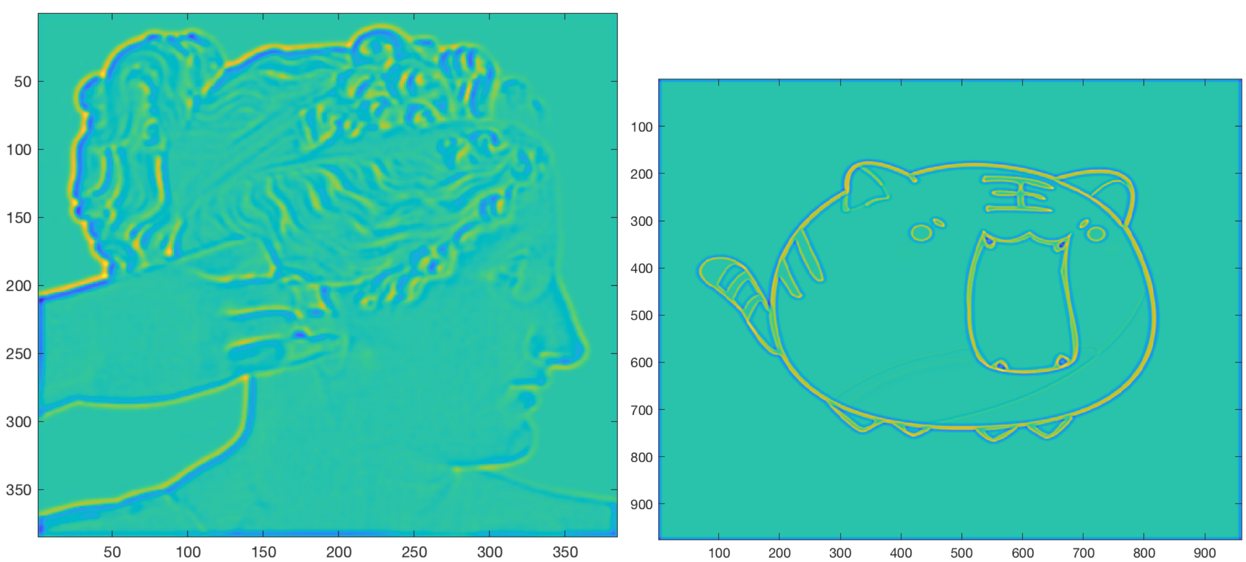

Figure 15: ConvolveLoG

2. Next find a way to detect zero-crossings of the results of these convolutions and display these results as images.

Listing 5: ZeroCrossFunc

```
function output = ZeroCrossFunc(LoG, threshold)
% LoG is the Convolve the 2D Laplacian of Gaussian filters with the input
    images
% threshold is the boundary we set
```

```
ABS = abs(LoG);
thres = mean(ABS(:)) * threshold;
output = zeros(size(LoG));
[h, w] = size(LoG);
for y = 2:1:(h-1)
    for x = 2:1:(w-1)
        patch = LoG(y-1:y+1, x-1:x+1);
        p=LoG(y,x);
        maxP = max(patch(:));
        minP = min(patch(:));
        if p>0
            if minP < 0
                zeroCross = true;
            else
                zeroCross = false;
            end
        else
            if maxP > 0
                zeroCross = true;
            else
                zeroCross = false;
            end
        end
        if((maxP - minP) > thres) && zeroCross
            output(y, x) = 1;
        end
    end
end
end
```

Figure 16: zero-crossings on Paolina.jpg

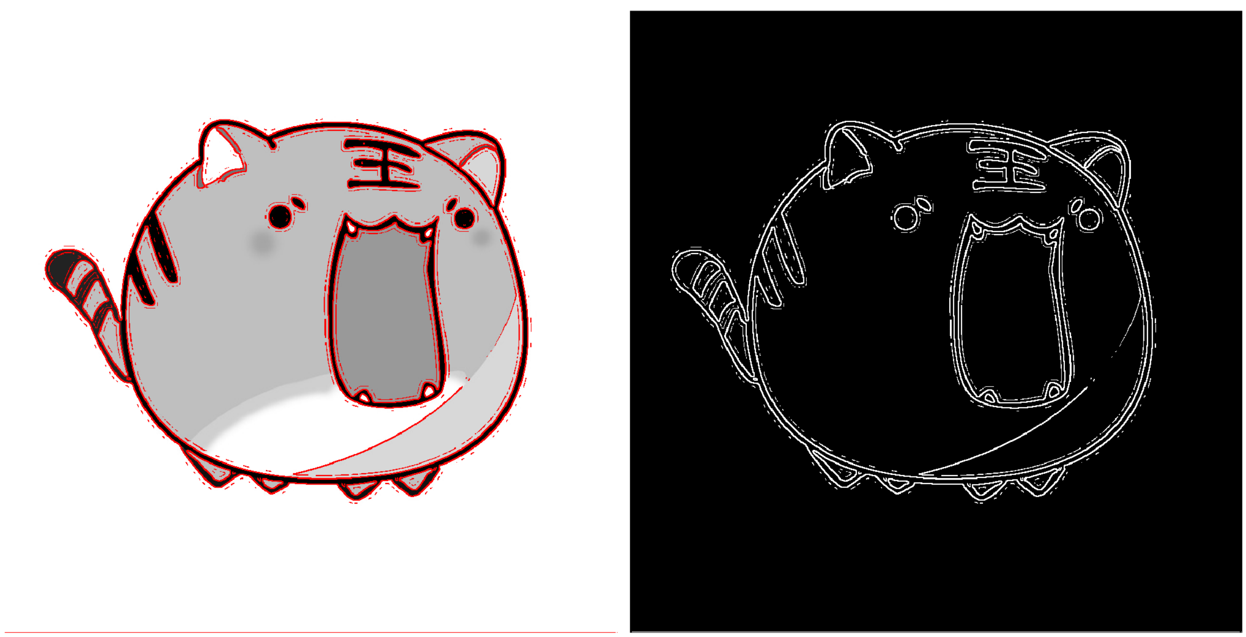

Figure 17: zero-crossings on tiger.jpg

3. Discuss your multi-scale edge detection results and in particular identify situations in which they appear to be successful versus those where they fail. In the Laplacian of Gaussian filters, the scale parameter σ is called the blur scale of the edge. This scale parameter should be adjusted based on the quality of image in order to avoid destroying true edges of the image. In this experiments, we choose $N = 15, threshold = mean(LoG) * 0.75, \sigma = 0.5, 1, 2, 3, 5, 8, 10$.

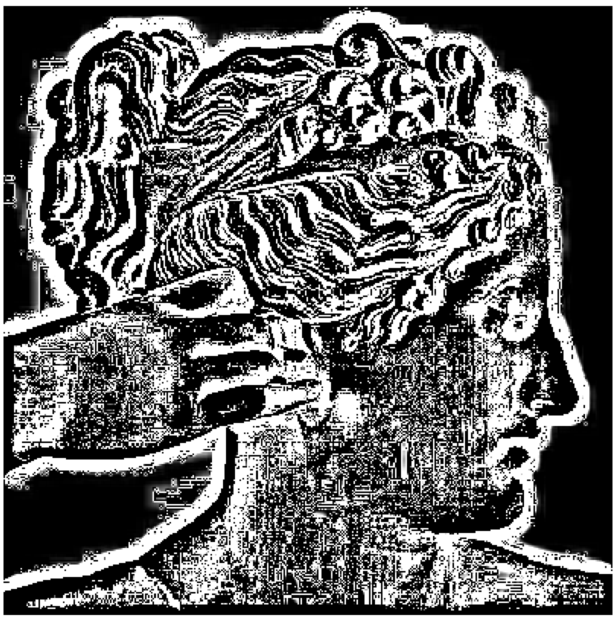

Figure 18: Convolve 2D LoG when $\sigma = 0.5$

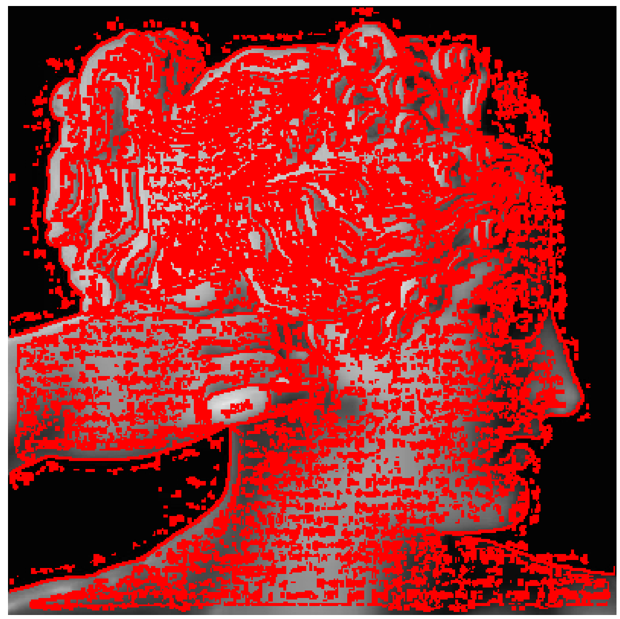

Figure 19: Edge detection results $\sigma = 0.5$

According to Question 1 c) , under the condition they assume, the Laplacian picks up the direction perpendicular to the local direction of the edge. We can crop some areas to see the detail of the edge detection results.

From several examples of different scale of signals above, we can know that when using large-scale signals, edge detection is reliable, but poor in localisation, and details on small areas are missing. On the other hand, small-scale signals capture detailed structures, but this detection suffers from false positives on clutters and textured regions.

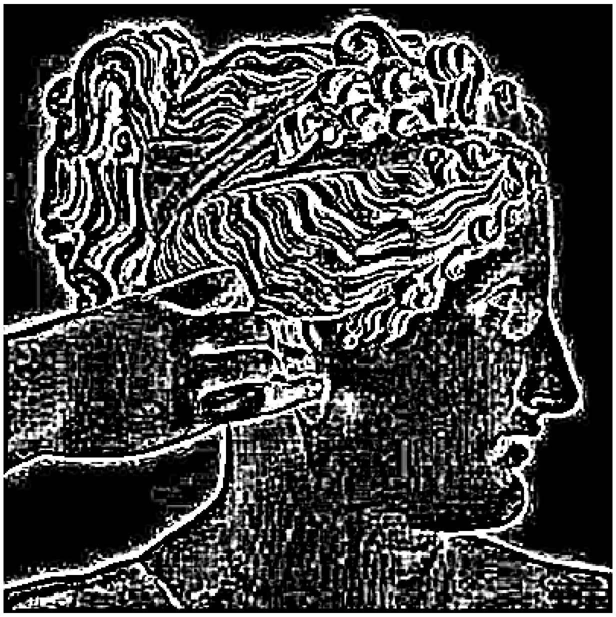

Figure 20: Convolve 2D LoG when $\sigma = 1$

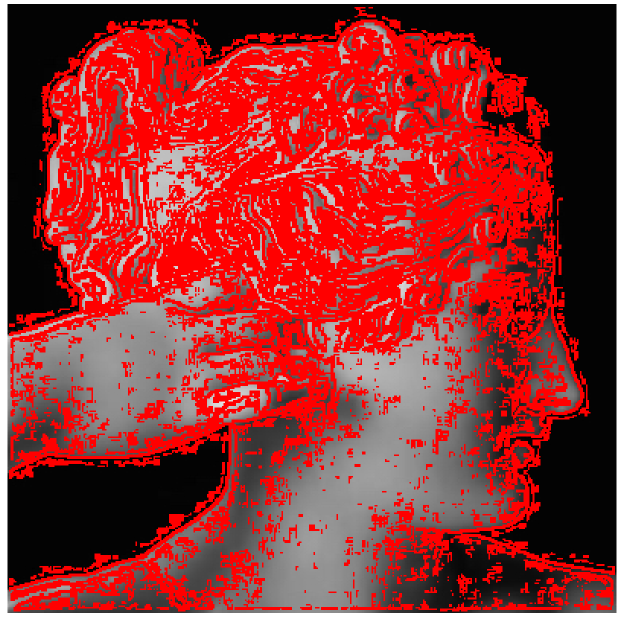

Figure 21: Edge detection results $\sigma = 1$

Figure 22: Convolve 2D LoG when $\sigma = 2$

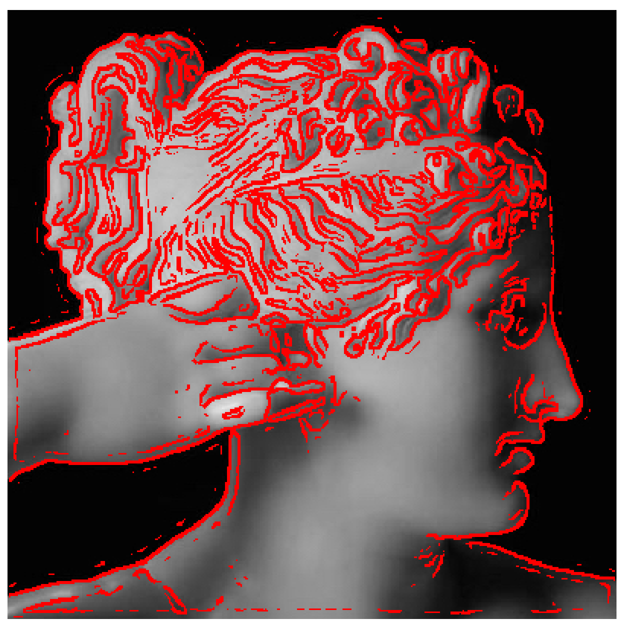

Figure 23: Edge detection results $\sigma = 2$

Figure 24: Convolve 2D LoG when $\sigma = 3$

Figure 25: Edge detection results $\sigma = 3$

Figure 26: Convolve 2D LoG when $\sigma = 5$

Figure 27: Edge detection results $\sigma = 5$

Figure 28: Convolve 2D LoG when $\sigma = 8$

Figure 29: Edge detection results $\sigma = 8$

Figure 30: Convolve 2D LoG when $\sigma = 10$

Figure 31: Edge detection results $\sigma = 10$

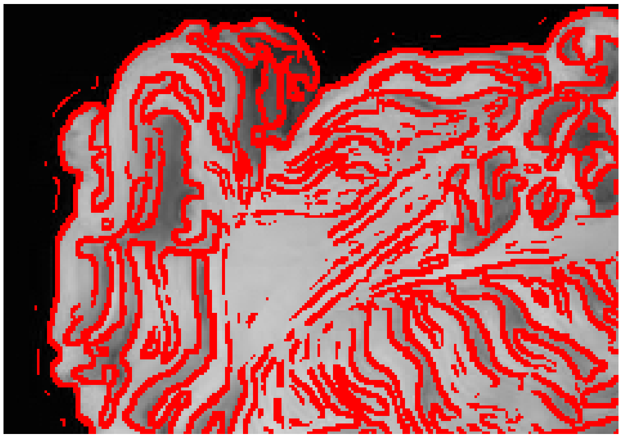

Figure 32: Edge detection results $\sigma = 2$

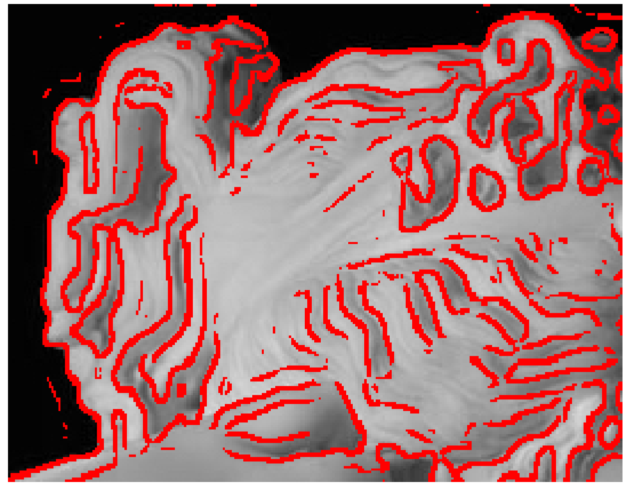

Figure 33: Edge detection results $\sigma = 8$

4. Repeat the analysis from parts 3 a) through 3 c) but now using odd Gabor filters of different wavelengths and orientation = 0, 45 and 90 degrees, instead of the Laplacian of Gaussian filters.

   Gabor filters are special classes of band pass filters, i.e., they allow a certain âĂŸbandâĂŹ of frequencies and reject the others. A Gabor filter can be viewed as a sinusoidal signal of particular frequency and orientation, modulated by a Gaussian wave. In this experiments, we choose $N = 7, threshold = mean(Gabor) * 0.75, \lambda = 5, 6, 7, 8, 10, 15, 20, orientation = \pi/4$

Listing 6: ConvolveGabor

```
clear
clc
sigma = 3; % ? parameter
N=7;
threshold_LOG=0.75;
threshold_Gabor=0.75;
lambda = 20;
angle = pi/4;

% Convolve the 2D Laplacian of Gaussian filters from Question 2 b) with the
    input images
I = imread('Paolina.jpg');
% I = imread('tiger.jpg');
image = rgb2gray(I);

[even, odd] = make2DGabor(N, lambda, angle);
```

```
Gabor_odd = conv2(image, odd, 'same');
Gabor_even = conv2(image, even, 'same');
figure(2)
% subfigure(1, 2, 1);
imshow(Gabor_odd);

output_odd = ZeroCrossFunc(Gabor_odd, threshold_Gabor);
output_even = ZeroCrossFunc(Gabor_even, threshold_Gabor);
figure(4)
imshow(I)
hold on
display = imoverlay(image, output_odd, [1,0,0]);
imshow(display)
figure(5)
imshow(output_odd)
% figure(6)
% imshow(output_odd + output_even)
```

Figure 34: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 5$

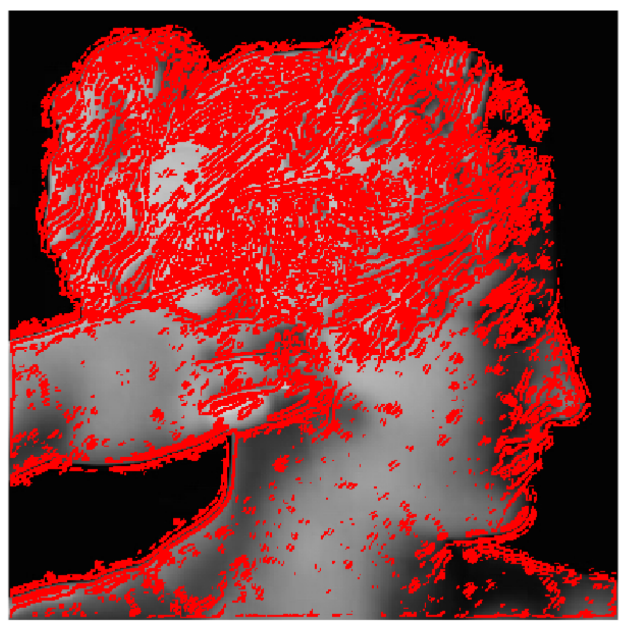

Figure 35: Edge detection results when $orientation = \pi/4, \lambda = 5$

Figure 36: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 6$

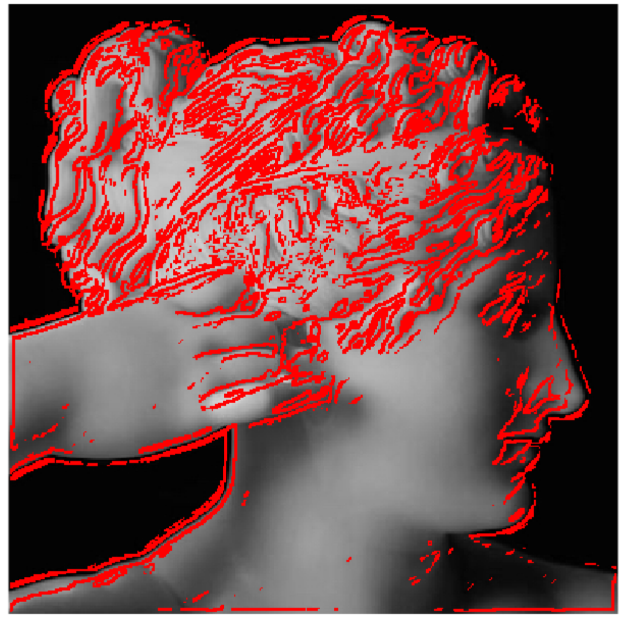

Figure 37: Edge detection results when $orientation = \pi/4, \lambda = 6$

Figure 38: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 7$

Figure 39: Edge detection results when $orientation = \pi/4, \lambda = 7$

Figure 40: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 8$

Figure 41: Edge detection results when $orientation = \pi/4, \lambda = 8$

Figure 42: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 10$

Figure 43: Edge detection results when $orientation = \pi/4, \lambda = 10$

Figure 44: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 15$

Figure 45: Edge detection results when $orientation = \pi/4, \lambda = 15$

Figure 46: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 20$

Figure 47: Edge detection results when $orientation = \pi/4, \lambda = 20$

Then we change the orientation = 0, 45 and 90 degrees, and fix the wavelength $\lambda = 10$.

Figure 48: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 10$

Figure 49: Edge detection results when $orientation = 0, \lambda = 10$

Figure 50: Convolve 2D Gabor when $orientation = \pi/8, \lambda = 10$

Figure 51: Edge detection results when $orientation = \pi/8, \lambda = 10$

Figure 52: Convolve 2D Gabor when $orientation = \pi/4, \lambda = 10$

Figure 53: Edge detection results when $orientation = \pi/4, \lambda = 10$

From the picture above, we can see that the edge detection results in Question 3 c) are better than the results in Question 3 d), especially in faint edge on the nose and hair on image 'Paolina.jpg'.

In the Gabor filter, the bandwidth or σ controls the overall size of the Gabor envelope. For larger bandwidth, the envelope increase allowing more stripes and with small bandwidth the envelope tightens. The orientation controls the orientation of the Gabor filter. When the orientation equals to 0 degree, it means the vertical position of the Gabor function. In the Gabor filter, the wavelength λ governs the width of the strips of Gabor function. Increasing the wavelength produces thicker stripes and decreasing the wavelength produces thinner stripes.

In the Laplacian of Gaussian filter, sigma is the scale of the filter. The large sigma leads to wider lines, the smoother edges and more noise ignored.

The following image reflects the effect between different wavelength parameters of the Gabor filter and the sigma in the Laplacian of Gaussian filter. The sigma in the Laplacian of Gaussian filter reflects the size of the filter an the wavelength parameters of the Gabor filter reflects the width of the strips of the Gabor filter.

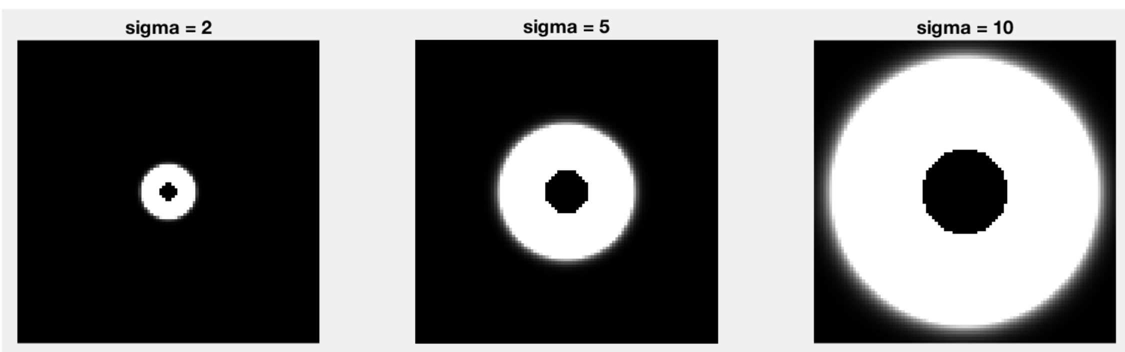

Figure 54: Laplacian of Gaussian, different sigma

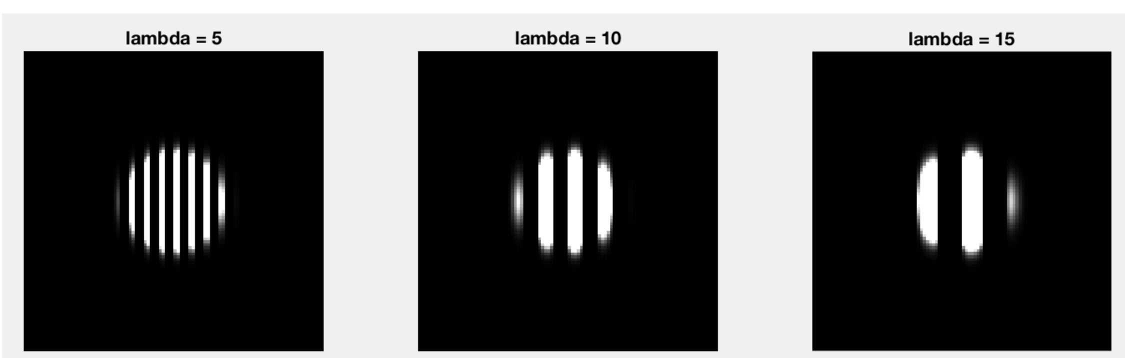

Figure 55: Gabor, fix orientation=0, different wavelength lambda