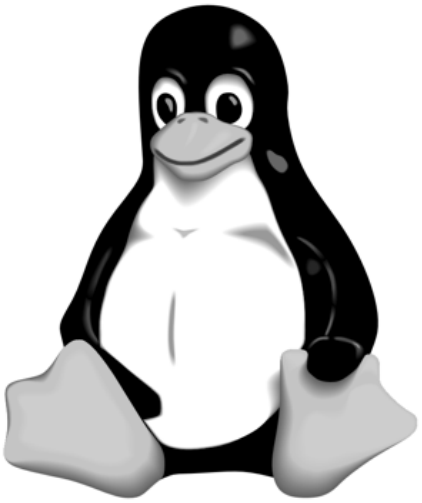
EEE 178 Homework 4

Curtis Muntz

Image preprocessing...

```
clear all;
clc;
close all;
%code for my custom functions can be found on
%https://github.com/curtismuntz/machine_vision/tree/master/commonFunctions
addpath ../commonFunctions
I = getIMG('Tux2.png');
I = imresize(I,[400,400]);
I = rgb2gray(I);
I = im2double(I);
imshow(I);
title('Original Image');
```

Original Image



I. Problem 1

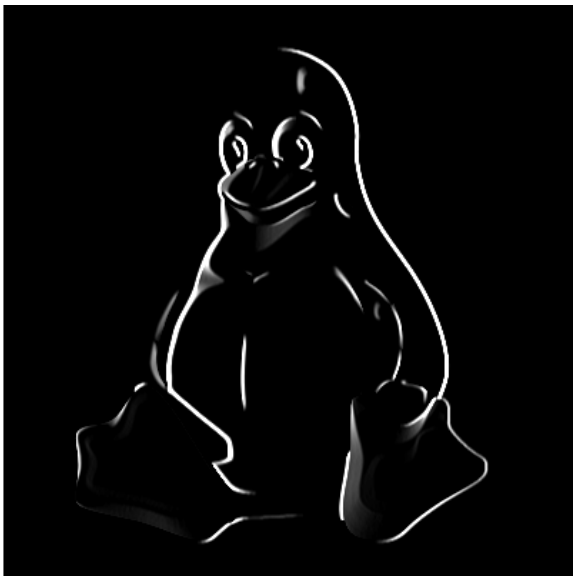
Edge detection in the space domain.
Pick an image with visible horizontal and vertical edges.

A. Part 1

```
%Apply the horizontal Sobel mask to the image. Show the result.
figure('name', 'horizontal sobel mask')
sobelX=double([-1 0 1; -2 0 2; -1 0 1])
horiz = convolution(I, sobelX);
imshow(im2uint8(horiz));
title('SobelX Masked Image')
```

```
sobelX =

    -1     0     1
    -2     0     2
    -1     0     1
```

SobelX Masked Image



B. Part 2

```
%Apply the vertical Sobel mask to the image. Show the result and compare
%with the previous question. Discuss your results.
figure('name', 'vertical sobel mask')
sobelY=double([1 2 1; 0 0 0; -1 -2 -1])
vert = convolution(I, sobelY);
imshow(im2uint8(vert));
title('SobelY Masked Image')
```

```
sobelY =

     1     2     1
     0     0     0
    -1    -2    -1
```

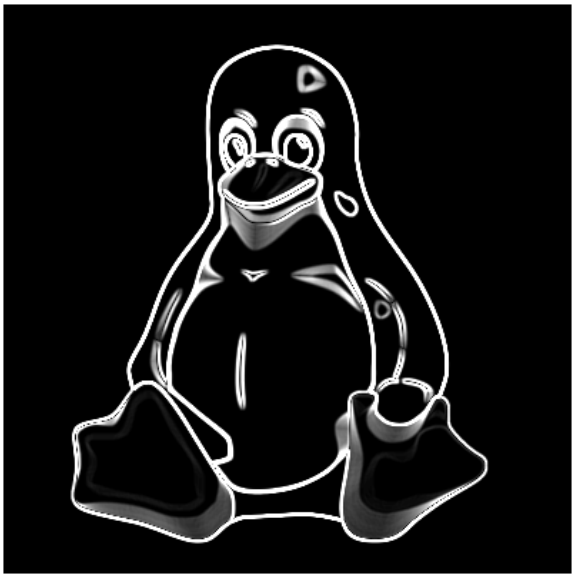
SobelY Masked Image

The horizontal sobel mask is detecting the vertical edges, and the vertical sobel mask is detecting the horizontal edges. There seems to be a little bit of crossover between the two (vertical is detecting a vertical edge, and vice versa).

C. Part 3

```
%Use the magnitude to combine horizontal and vertical edges. Show the result
    .
figure('name','magnitude'); %pop pop
magnitude = abs(horiz) + abs(vert);
imshow(magnitude);
title('Approximation of Magnitude');
```

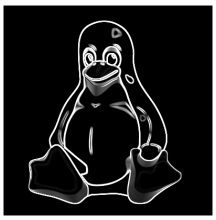
Approximation of Magnitude



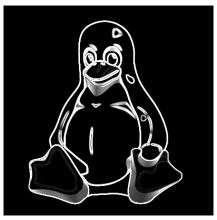
D. Part 4

```
%In the Sobel mask, the constant c = 2. We want to see the effect of c on
      the
%edge detection process, try different values and see what happens.
figure('name', 'Using Various Values for c in Sobel Mask')
subplot(221)
c=1;
sobelX=double([-1 0 1; -c 0 c; -1 0 1]);
sobelY=double([1 c 1; 0 0 0; -1 -c -1]);
h=convolution(I,sobelX);
v=convolution(I,sobelY);
C=(abs(h)+abs(v));
imshow(C);
title('c=1')

subplot(222);
c=3;
sobelX=double([-1 0 1; -c 0 c; -1 0 1]);
sobelY=double([1 c 1; 0 0 0; -1 -c -1]);
h=convolution(I,sobelX);
v=convolution(I,sobelY);
C=(abs(h)+abs(v));
imshow(C);
title('c=3')

subplot(223)
c=5;
sobelX=double([-1 0 1; -c 0 c; -1 0 1]);
sobelY=double([1 c 1; 0 0 0; -1 -c -1]);
h=convolution(I,sobelX);
v=convolution(I,sobelY);
C=(abs(h)+abs(v));
imshow(C);
title('c=5')


subplot(224)
c=10;
sobelX=double([-1 0 1; -c 0 c; -1 0 1]);
sobelY=double([1 c 1; 0 0 0; -1 -c -1]);
startS=tic; %asked for part 6
h=convolution(I,sobelX);
v=convolution(I,sobelY);
C=(abs(h)+abs(v));
sobeltime= toc(startS);
imshow(C);
title('c=10')
```

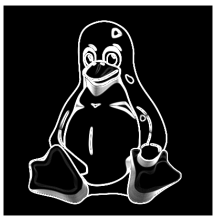
c=1



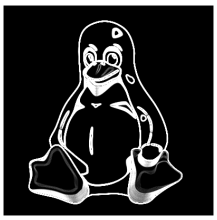
c=3



c=5



c=10

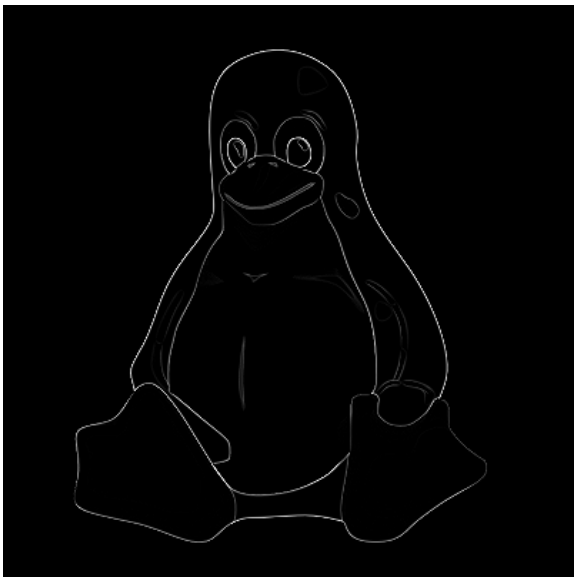


As the value for c increases, the resultant convolution of the masks detect more parts of the image as edges.

E. Part 5

```
%Use the Laplacian mask to perform edge detection on the same image.
%Compare with the results of question 3).
figure('name', 'laplacian mask')
L=[0 1 0; 1 -4 1; 0 1 0];
startL = tic;
laplacedI=convolution(I,L);
laplacetime = toc(startL);
imshow(im2uint8(laplacedI)), title('Laplacian Mask');
rmpath ../commonFunctions
```

Laplacian Mask



The laplacian mask seems to produce the same image as the magnitude of the Sobel masks did.

F. Part 6

```
%Compare between the Laplacian and Sobel masks in terms of the computation
%time, use Matlab command tic - toc. For the Sobel mask, you need to
%consider the complete implementation, that is edge detection in both
%directions.

%-- these were calculated in parts 4 and 5 respectively
sprintf('sobel approach: %s',sobeltime)
sprintf('laplacian approach: %s', laplacetime)
```

```
ans =

sobel approach: 1.966261e-01


ans =

laplacian approach: 7.457914e-02
```

The sobel approach takes longer, primarily because the operation is applying two masks, whereas the Laplacian approach only applies one.

II. Problem 2

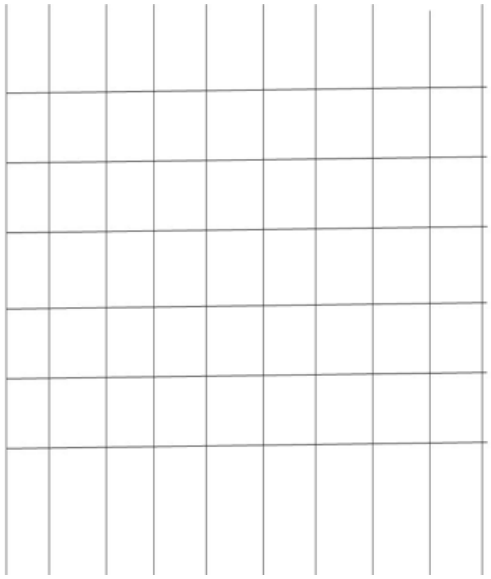
Edge detection in the frequency domain.

A. Part 1

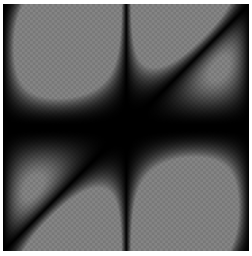
```
%Obtain the FFT of the horizontal and vertical Sobel masks.
%You need to perform zero padding.
addpath ../commonFunctions
%I2 = getIMG('Soyuz_TMA-19_spacecraft_departs_the_ISS.jpg');
I2 = getIMG('ed3.jpg');
rmpath ../commonFunctions
%I2 = im2double(rgb2gray(imresize(I2,[401,401])));
I2 = imresize(I2,[401,401]);
figure('name','Problem 2 Image');
imshow(I2);
title('Problem 2 Original Image');

figure('name','ffts of Sobel Masks');
c=2;
sobelX=double([-1 0 1; -c 0 c; -1 0 1]);
sobelY=double([1 c 1; 0 0 0; -1 -c -1]);
%work on making this more general.........
sobelX1 = padarray(sobelX,[199,199]);
sobelFFTX=fft2(sobelX1);
%work on making this more general.........
sobelY1 = padarray(sobelY,[199,199]);
sobelFFTY=fft2(sobelY1);

subplot(121), imshow(sobelFFTX), title('SobelX fft');
subplot(122), imshow(sobelFFTY), title('SobelY fft');
```

```
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
```

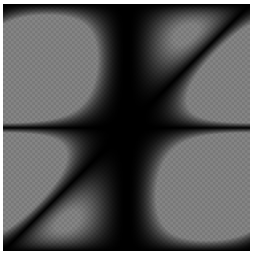
Problem 2 Original Image



SobelX fft



SobelY fft

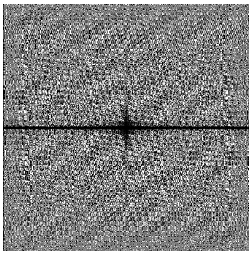


B. Part 2

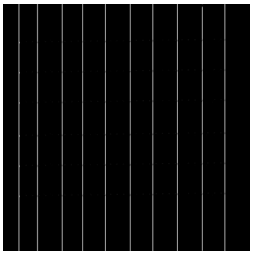
```
%Perform filtering in the frequency domain using the horizontal Sobel mask.

figure('name', 'frequency horizontal filtering');
horizF=fft2(I2).*sobelFFTX;
horizS=ifft2(horizF);
dmin = min(min(abs(horizS)));
dmax = max(max(abs(horizS)));
subplot(121), imshow(horizF), title('Horizontal Sobel fft Mask');
subplot(122), imshow(ifftshift(horizS),[dmin dmax]), title('Filtered Image')
      ;
```

```
Warning: Displaying real part of complex input.
```

Horizontal Sobel fft Mask



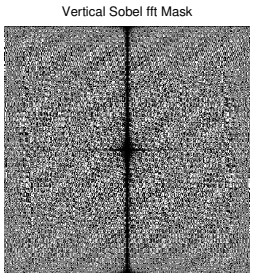
Filtered Image

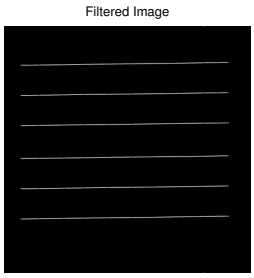


C. Part 3

```
%Perform filtering in the frequency domain using the vertical Sobel mask.
figure('name', 'frequency vertical filtering');
vertF=fft2(I2).*sobelFFTY;
vertS=ifft2(vertF);
dmin = min(min(abs(vertS)));
dmax = max(max(abs(vertS)));
subplot(121), imshow(vertF), title('Vertical Sobel fft Mask');
subplot(122), imshow(ifftshift(vertS),[dmin dmax]), title('Filtered Image');
```

```
Warning: Displaying real part of complex input.
```



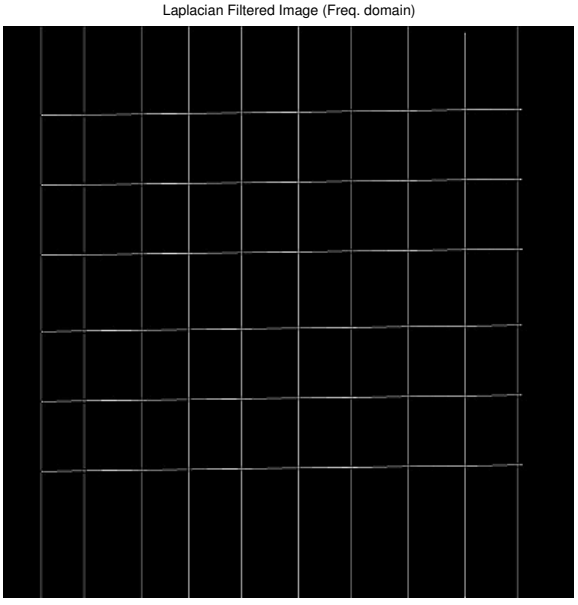


D. Part 4

```
%Obtain the FFT of the Laplacian mask. You need to perform zero padding.
L1 = fft2(padarray(L,[199,199]));
```

E. Part 5

```
%Perform filtering in the frequency domain using Laplacian to detect
%the edges.
figure('name','laplacian frequency filtering');
lapF=fft2(I2).*L1;
lapS=ifft2(lapF);
dmin = min(min(abs(lapS)));
dmax = max(max(abs(lapS)));
imshow(ifftshift(lapS),[dmin dmax]), title('Laplacian Filtered Image (Freq.
    domain)');
```



F. Part 6

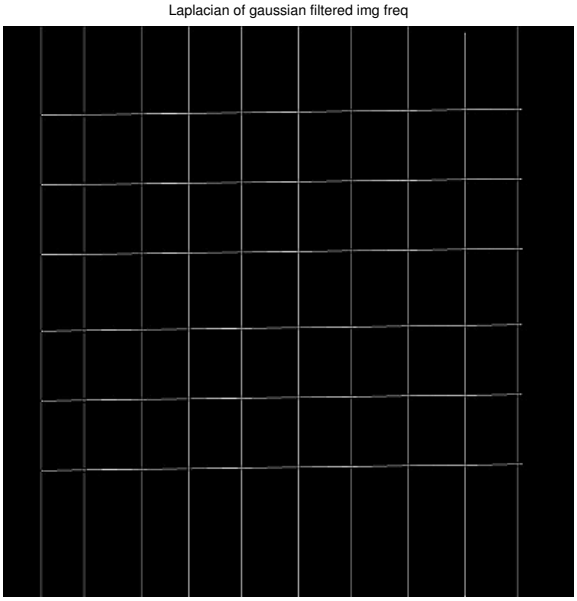
```
%Construct a 3 by 3 mask for the Laplacian of Gaussian (LoG) and then obtain
%its FFT. You can use fspecial for this question.
%addpath ../commonFunctions
%gauss=gaussfilter(3, 1/2);
%LoG=L*gauss;
%LoGF=fft2(LoG);
%rmpath ../commonFunctions
LoG=fspecial('log',[401,401]);
LoGF=fft2(LoG);
```

G. Part 7

```
%Perform filtering in the frequency domain using LoG to detect the edges.
I3=fft2(I2).*L1;
lapS=ifft2(I3);
dmin = min(min(abs(lapS)));
dmax = max(max(abs(lapS)));
```

H. Part 8

```
%Show and discuss your results.
figure('name','laplacian of gaussian freq filtering');
imshow(ifftshift(lapS),[dmin dmax]);
title('Laplacian of gaussian filtered img freq');
```

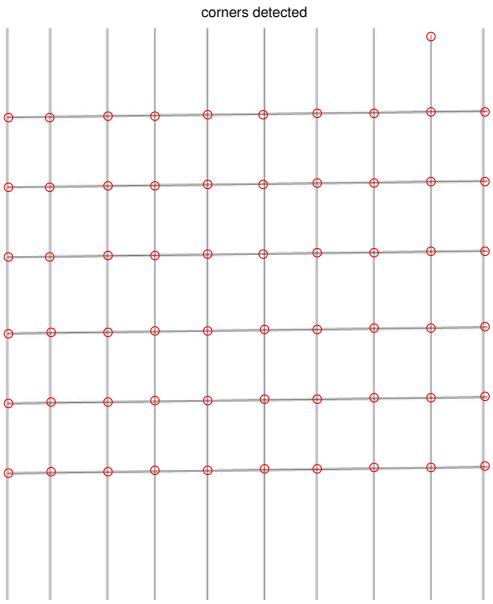


The FFT of the Laplacian of Gaussian is very effective at finding the edges in the image.

III. Problem 3

The Harris detector was introduced in 1988 for corner detection. An illustration is shown in figure 2. Apply the Harris detector to an image of your choice. Pick an image with visible corners. Mark all corners on the image in a similar way to figure 2 Do you think the Harris edge detector is a linear ltering operation? Explain

```
figure('name','corner detection');
C=corner(I2); %to find corner points
imshow(I2);hold on, plot(C(:,1),C(:,2),'ro')
title('corners detected')
```



The built in corner detection funciton was able to pick up all the corners in the image. The funciton is not linear, as it calculates its mask based off of gradients within a range of the pixel values, and not use a mask on the pixel values.