EEE 178 Take Home Exam 1

Curtis Muntz

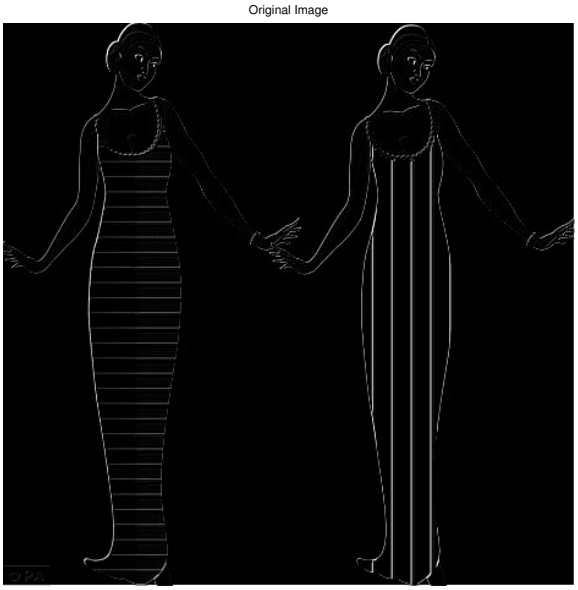
preprocessing

```
clear all, clc, close all;

%problem 1 data
edges = xlsread('TheEdges.xlsx');
row1 = xlsread('Row1.xlsx');
col1 = xlsread('Col1.xlsx');

%problem 2 data
I2 = xlsread('MatriceC.xlsx');

%problem 3 data
addpath ../commonFunctions
I3 = imread('Tux2.png');
rmpath ../commonFunctions

%problem 4 data
I4 = imread('mvProblem4.jpg');

%problem 5 data
I5 = imread('mvProblem5.jpg');
```

I. Problem 1: Recovering Data

Because we have the first row of the original image, and it is known what mask was used to produce our edged image, it is mathematically possible to calculate the original image. The horizontal Roberts mask $\nabla f = f(i,j) + f(i+1, j+1)$ was used to produce the edged image. This implies that solving for $f(i+1, j+1)$, and knowing the resultant ∇f, we can use a for loop to rebuild the image one row at a time, because the entire first row is known. First, we make a array of zeroes with the size of the original image, and load in the rows and coluns. Next we iteratively solve for the next row and next pixel by noting the relationship $\nabla f - f(i,j) = f(i+1, j+1)$. Without the first column of the image as well, this will produce all of the remaining image with the exception of the first row.

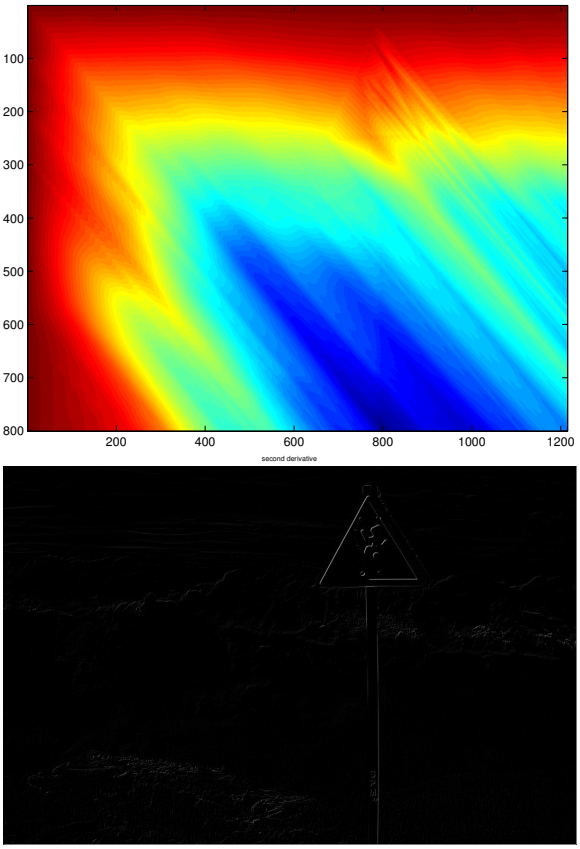
```
close all, clearvars -except I2 I4 I5 edges
    row1 col1 I3;

figure('name', 'ORIGINAL IMAGE OMG');
imshow(edges), title('Original Image');
[M,N]=size(edges);
recovered=zeros(M,N);
%load column 1
for i=1:M
    recovered(i,1)=col1(i,1);
end
%load row 1
for i=1:N
    recovered(1,i)=row1(1,i);
end

for i=1:(M-1)
    for j=1:(N-1)
        recovered(i+1,j+1)=-1*edges(i,j)+
    recovered(i,j);
    end
end

figure('name', 'recovered image');
imshow(recovered), title('recovered');
```



II. Problem 2:

I do not have an interpretation for the data shown by imagesc. If I had to guess, it has something to do with a DCT, based on how the data seems to be ordered. By performing the Roberts filter twice, we obtain an image that looks to be a gradient or an image of edges. Because we know that an edge is the derivative of an image, we look at only the first derivative instead of the second, and we discover the actual image. This must mean that the original image was the integral of our image. Looking back at the imagesc output from to part 1, we can see that the image looks to be a running sum of the image discovered in part 2.

```
close all, clearvars -except I2 I4 I5 edges
    row1 col1 I3;
%Part1
figure('name', 'imagesc');
imagesc(I2);

%Part 2
I=im2double(I2);
robertx=[1 0; 0 -1];
I1=imfilter(I,robertx);
I2=imfilter(I1,robertx);
figure('name','second derivative'), imshow(I2),
    title('second derivative');
figure('name','actual image'), imshow(I1),
    title('actual image');
```



III. Problem 3: Finite Differenece

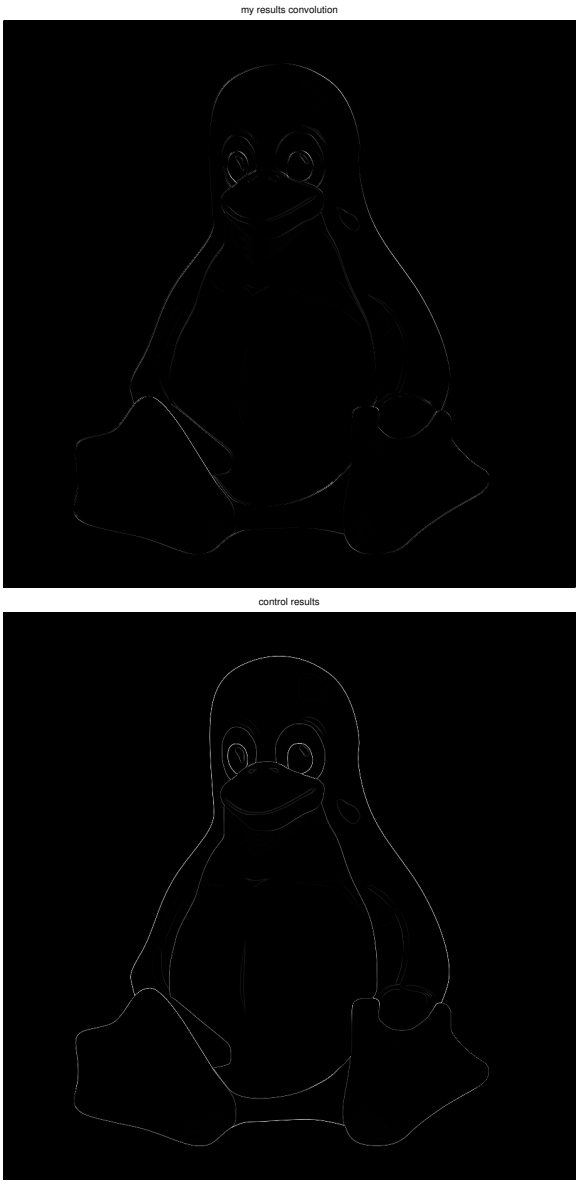
I have chosen the finite difference equation from Wikipedia: $f_{x,y}(x,y) \approx \frac{f(x+h,y+k)-f(x+h,y)-f(x,y+k)+2f(x,y)-f(x-h,y)-f(x,y-k)+f(x-h,y-k)}{2hk}$ This gets simplified to the following equation after choosing $h = 1$ and $k = 1$ and replacing the x and y parameters with i and j. $f_{x,y}(i,j) = f(i+1,j+1) - f(i+1,j) + 2f(i,j) - f(i-1,j) - f(i,j-1) + f(i-1,j-1)$. Solving for a mask, we obtain

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

(1)

Comparing the results we obtained from the fspecial definition of Laplacian, which is a two dimensional derivative of the image yields similar enough results. The results are similar whether the filtering was performed in the frequency or the time domain.

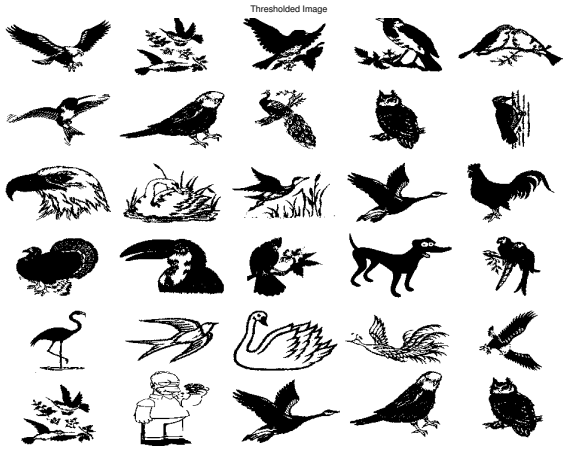
```
close all, clearvars -except I2 I4 I5 edges
    row1 col1 I3;
%gradFxy(i,j)=f(i+1,j+1)-f(i+1,j)+2*f(i,j)-f(i
    -1,j)-f(i,j-1)+f(i-1,j-1)
gradFxy=[1,-1,0;-1,2,-1;0,-1,1];
I=im2double(rgb2gray(I3));
[M,N]=size(I);
If=fft2(I);
gradFxyfft=fft2(gradFxy);
%my method frequency:
gradFxyf=padarray(gradFxyfft,[M-3 N-3],'post');
Ifilt=If.*gradFxyf;
addpath ../commonFunctions
If1=convolution(I,gradFxy);
rmpath ../commonFunctions
figure('name','my results'), imshow(If1); title
    ('my results convolution');


%control method:
grad=fspecial('laplacian');
Icontrol=imfilter(I,grad);
figure('name','control results'), imshow(
    Icontrol), title('control results');
```

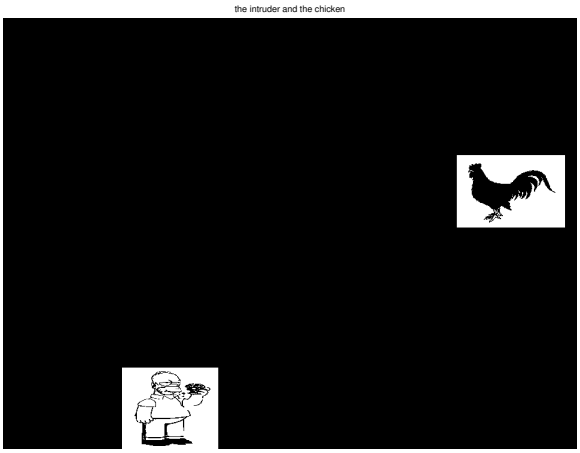


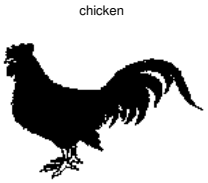

```
close all, clearvars -except I2 I4 I5 edges
    row1 col1 I3;

figure('name', 'original image'),
imshow(I4), title('Original Image');
I=rgb2gray(I4);
thresh=graythresh(I); %OTSU!!!! ('s method)
I=im2bw(I,thresh);
figure('name', 'thresh img');
imshow(I); title('Thresholded Image');

%define homer and his complement
intruder = I4([585:720],[200:360],:);
intruder=rgb2gray(intruder);
intruder=im2bw(intruder,thresh);
antiintruder=imcomplement(intruder);

%define chicken and his complement
chicken = I4([230:350],[760:940],:);
chicken=rgb2gray(chicken);
chicken=im2bw(chicken,thresh);
antichicken=imcomplement(chicken);

%intruder and the chicken
figure('name','searching');
subplot(121),imshow(intruder), title('intruder'
    );
subplot(122),imshow(chicken), title('chicken');

%isolate intruder
figure('name','isolated intruder');
B1=strel('arbitrary',intruder);
B2=strel('arbitrary',antiintruder);
Img=bwhitmiss(I,B1,B2);
Img=imdilate(Img,B1);
imshow(Img), title('intruder isolated');

%isolate chicken
figure('name','isolated chicken');
B3=strel('arbitrary',chicken);
B4=strel('arbitrary',antichicken);
Img1=bwhitmiss(I,B3,B4);
Img1=imdilate(Img1,B3);
imshow(Img1), title('chicken isolated');

%the intruder and the chicken
figure('name','the intruder and the chicken');
Img2=Img1+Img; imshow(Img2), title('the
    intruder and the chicken');
```

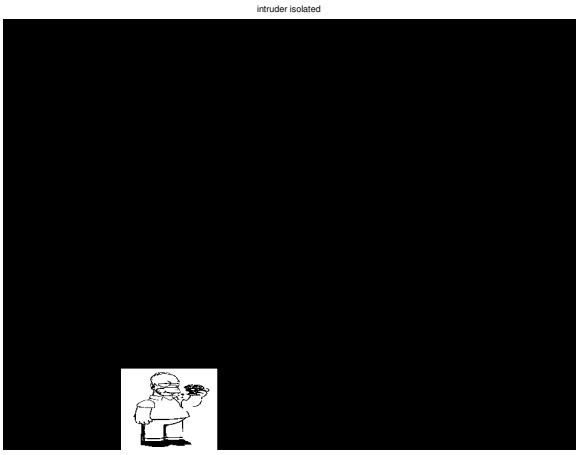
IV. Problem 4: Intruder and Chicken

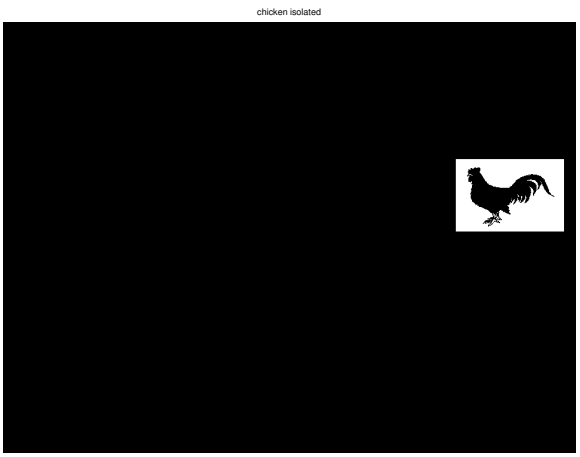
The easiest way to detect an intruder is knowing what we're looking for initially. These two objects were defined by cropping portions of the original image, and thresholding them with the same threshold value that the original image was thresholded. Next, the hit and miss was performed on the image to detect the intruder and the chicken. This method, when performed with the mask of the object that we desired and the complement of the object, will return a black image with a single pixel that represents the center of the object that we're searching for. Next, a dilation is performed about this pixel with the original object and the intruder or the chicken is isolated. A total of three morphological operations are used to isolate these objects. The individual











V. Problem 5

Because this image is so faint, we have to use a threshold in the realm of 0.999 (chosen through experimentation). Because of the precision required for the threshold, I've elected to use the builtin graythresh() function in MATLAB. This function utilizes Otsu's method in order to find the most optimal thresholding value, which it chose to be 0.9922. The method that I've chosen performs the following logic: First, it dilates the image with a line angled at 40 degrees (chosen through experimentation), in order to reduce the amount of random noise in the image, and to clean up the black lines. Next it is dilated with a square structuring element in order to further remove noise. Finally, the image is eroded with a disk structuring element in order to make the black lines more bold. The total execution time is approximately 0.025 seconds on my computer.

```
close all, clearvars -except I2 I4 I5 edges
    row1 col1 I3;

%pre-processing...
I=rgb2gray(I5);
thresh=graythresh(I);
I=im2bw(I,thresh);
figure('name','original image');
imshow(I), title('original image');

%method
tstart=tic;
B=strel('line',4,40);
B1=strel('square',3);
B2=strel('disk',3);
img=imdilate(I,B);
img=imdilate(I,B1);
img=imerode(img,B2);
endofmethod=toc(tstart)

%display
figure('name', 'operations');
imshow(img);
```

original image