Homework 8

Meta information	
Name	Savan Kiran
Program	Masters in Computer Science
Questions skipped	PART B2
Questions substituted	N/A
Extra credit questions	N/A

PART A

1. Let's calculate the finite differences along X-direction using convolution with the X-gradient operator given by `dx=[1 -1]`. The magnitude of the gradient could have crossed the maximum threshold of 1 for a grayscale image. We will therefore, scale down the brightness by the maximum value. The resulting image is shown in Figure 1. Below is the code snippet for the same.

```
%1. Gradient
image=imread('climber.tiff');
I=rgb2gray(image);
figure;
imshow(I);
dx=[1 -1];
Ix=conv2(I,dx,'same');
m1=max(Ix(:));
Ix=Ix./m1;
figure;
imshow(Ix);
```

Figure 1. Magnitude of the gradient of 'climber.tiff' using finite differences along X-axis

2. We now find a threshold for the gradient magnitude to determine the real edges by plug and check. I found a value of 0.35 to be a good fit. Any pixel with magnitude above threshold will be colored white whereas any pixel with magnitude below threshold will be colored black. The resulting image is shown in Figure 2. Below is the code snippet for the same.

```
%2. Edge detection
I2=zeros(size(Ix,1),size(Ix,2));
for i=1:size(Ix,1)
    for j=1:size(Ix,2)
        if(Ix(i,j)>0.35)
            I2(i,j)=1;
        end
    end
end
figure;
imshow(I2);
```

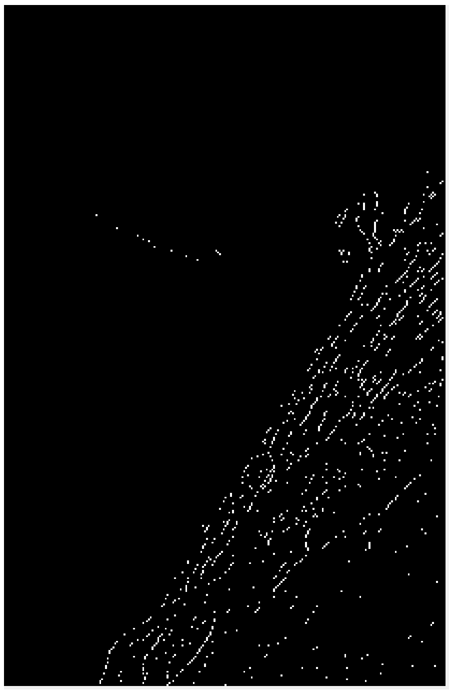

Figure 2. Real edges found with threshold for the gradient magnitude set at 0.35

3. We will now, smoothen the image with Gaussian blur. To create a 3x3 Gaussian mask with sigma=2, we will first create a mesh grid going from -1 to 1 (i.e., {-1,0,1}). We will use the coordinate vectors thus produced to create the Gaussian filter using the formula,

$$\exp(-((x^2 + y^2)/2\sigma^2))$$

We would have to normalize the Gaussian filter so that it sums to 1. We will also plot the surface of the Gaussian filter to see how values vary which is shown in Figure 3. We will now use a convolution routine with Gaussian filter to smoothen the image. The resulting image is shown in Figure 4. Below is the code snippet for the same.

```
%3. Gaussian mask - smoothing
sigma=2;
val=-1 : 1;
[X Y]=meshgrid(val,val);
guass_filter=exp(-(X.^2+Y.^2)/(2*sigma*sigma));
guass_filter=guass_filter/sum(guass_filter(:));
figure;
surf(X,Y,guass_filter);
Ismooth=conv2(I,guass_filter,'same');
m2=max(Ismooth(:));
Ismooth=Ismooth./m2;
figure;
imshow(Ismooth);
```

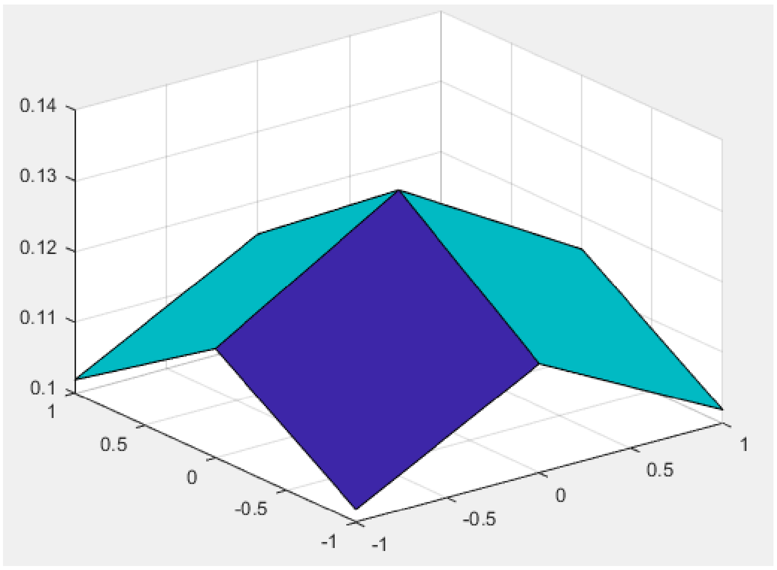

Figure 3. Surface plot of the calculated Gaussian mask

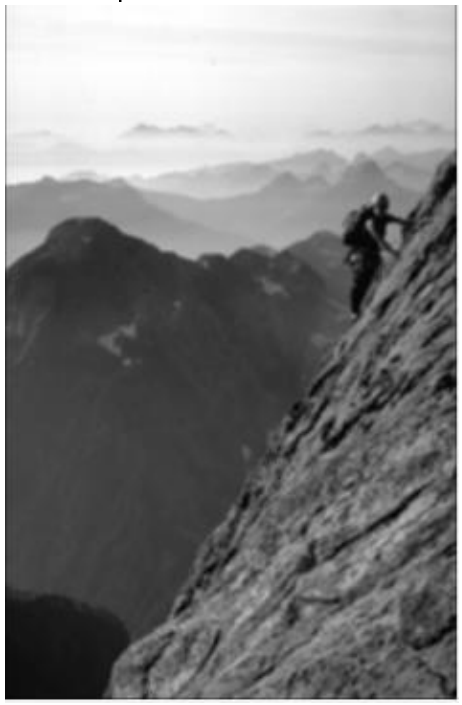
Figure 4. Image smoothened with Gaussian mask convolution

4. We will now use the smoothened image (obtained from last step by Gaussian filter) and follow procedure from 1 and 2 to find edges. The threshold is now set to 0.05. We can see that the edges are now wider and are more connected than earlier. The resulting image is shown in Figure 5. Below is the code snippet for the same.

```
%4. Edge detection in smoothened image
dx=[1 -1];
Ix=conv2(Ismooth,dx,'same');
Ie=zeros(size(Ix,1),size(Ix,2));
for i=1:size(Ix,1)
    for j=1:size(Ix,2)
        if(Ix(i,j)>0.05)
            Ie(i,j)=1;
        end
    end
end
figure;
imshow(Ie);
```

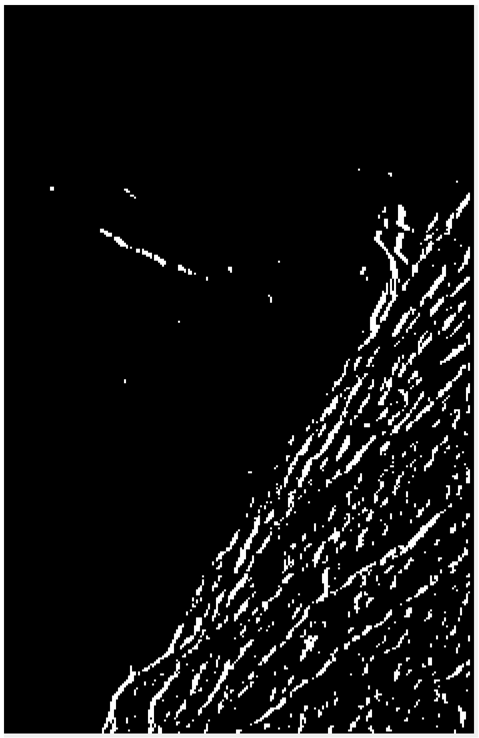

Figure 5. Edge detection with smoothened image (smoothened w/ Gaussian filter) with threshold set at 0.05

5. We will now combine Gaussian blur and edge detection into one filter. Now, the Gaussian mask is calculated with sigma=4. We will convolve both smoothing by Gaussian mask and finite difference along X direction. We will do the same for Y direction. We will combine both the magnitudes as sum of squares. The resulting image is shown in Figure 6 which depicts best edge detection so far. Most of the edges are identified however, most edges are disconnected. Below is the code snippet for the same.

```
%5. Edge detection using combined filter
sigma=4;
val=-1 : 1;
[X Y]=meshgrid(val,val);
guass_filter=exp(-(X.^2+Y.^2)/(2*sigma*sigma));
guass_filter=guass_filter/sum(guass_filter(:));
[dx dy]=gradient(guass_filter);
Ix=conv2(conv2(I,dx,'same'),guass_filter,'same');
%Ix=conv2(I,conv2(dx,guass_filter,'same'),'same');
Iy=conv2(conv2(I,dy,'same'),guass_filter,'same');
mx=max(Ix(:));
Ix=Ix./mx;
my=max(Iy(:));
Iy=Ix./my;
Im=sqrt(Ix.*Ix+Iy.*Iy);
figure;
imshow(Im);
```

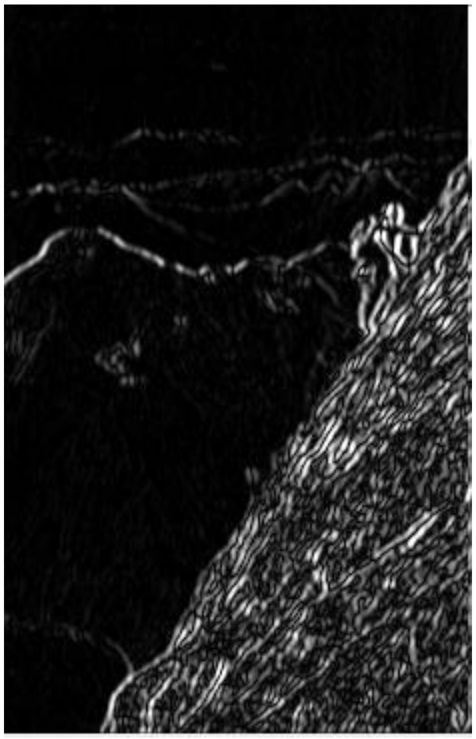

Figure 6. Edge detection using combined filter for Gaussian mask and Finite difference along X and Y directions

6. Algebraically, f(x,y)=g(x)h(y) is a function that is separable (in x and y). If f(x,y) is a convolution along x and y, representing Gaussian filter along both x and y, then it can equivalently be implemented as a two separate 1D convolutions g(x) and h(y), one followed by another.

$$G(x,y) * f = \left(g_x * g_y\right) * f = \ g_x * \left(g_y * f\right)$$

$$G(x,y) = e^{-\left(x^2+\frac{y^2}{2\sigma^2}\right)}$$

$$g(x) = e^{-\left(\frac{x^2}{2\sigma^2}\right)}$$

$$g(y) = e^{-(y^2/2\sigma^2)}$$

When the Gaussian filters along both directions are combined, we can speed up the calculation as compared to them being applied separately. We can observe the results below.

```
Time (Separately applied): 0.67098
Time (Combined): 0.34985
```

The Gaussian filter when combined is almost twice as fast as applied separately. The iteration ran for 1000 iterations and the code snippet is provided below.

```
%6. Speed check
tic;
val=-1 : 1;
[X Y]=meshgrid(val,val);
gfx=exp(-(X.^2)/(2*sigma*sigma));
gfx=gfx/sum(gfx(:));
gfy=exp(-(Y.^2)/(2*sigma*sigma));
gfy=gfy/sum(gfy(:));
for i=1:1000
    I1=conv2(I,gfx,'same');
    I2=conv2(I1,gfy,'same');
end
t1=toc;
disp(['time 1: ' num2str(t1)]);

tic;
for i=1:1000
    I3=conv2(I,guass_filter,'same');
end
t2=toc;
disp(['time 2: ' num2str(t2)]);
```

PART B1 (Edge detection)

To implement the gradient based edge detection, we will have to implement the non-maximal suppression and edge linking. We can do this by looking for a maximum along a slice normal to the curve (along the gradient). We predict the next edge point from the given point by constructing the tangents to the edge curve and use this to predict the next edge point. For edge linking, we pick the next edge point with sufficiently large gradient by following a chain of tangential points.

We start with calculating the orientation of the edge,

$$df = [\frac{df}{dx}, \frac{df}{dy}]$$

The image gradient orientation is given by,

$$\Theta = tan^{-1}\left(\frac{\frac{df}{dy}}{\frac{df}{dx}}\right)$$

And edge direction is given by,

$$\Theta = tan^{-1}\left(-\frac{\frac{df}{dy}}{\frac{df}{dx}}\right)$$

Once, we have the edge direction, we pick next point that is in the edge direction and has the gradient magnitude over the threshold. We display the resulting image with jet colormap and the colorbar shown in Figure 7. Below is the code snippet for the same.

```
%Part B
Ith=atan2(Iy,Ix);
figure;
imagesc(Ith.*(Im>0.1));
axis image;
colormap(jet);
caxis([0.1 0.9]);
colorbar;
```

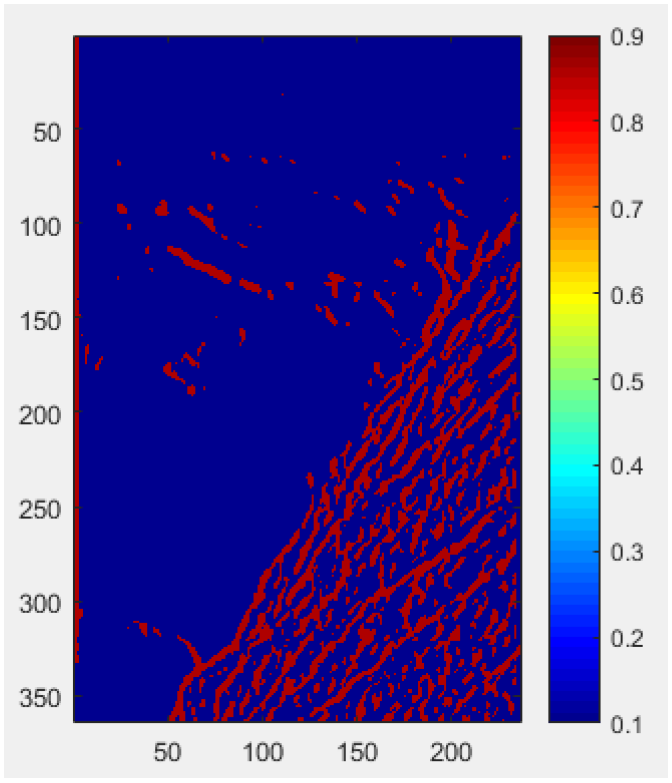

Figure 7. Edge detection using non-maximal suppression and edge linking