Assignment #1

1) Evaluate the convolution between *x* and both *h1* and *h2*
x = [1 1 1 1 2 3 4 5 6 6 6 6 0 -6 -12 12 12 12]
h1 = [1 -1]
h2 = [1 -2 1]
solve the problem analytically, and write use octave/matlab to check your solution

Sol:
x = [1 1 1 1 2 3 4 5 6 6 6 6 0 -6 -12 12 12 12]
h1 = [1 -1]
h2 = [1 -2 1]
y1 = conv(x,h1)
y2 = conv(x,h2)

y1= [1 0 0 0 1 1 1 1 1 0 0 0 -6 -6 -6 24 0 0 -12]
y2=[1 -1 0 0 1 0 0 0 0 -1 0 0 -6 0 0 30 -24 0 -12 12]

The first filter (h1) represents first difference . Hence, its response is zero when the input is constant, and its response = the slope of the line if it has constant slope. The second filter represents the second difference. Hence, it has zero response when the signal is constant or has a constant slope

2) Evaluate the convolution between the image *I* and the masks *w1* and *w2*

I = [

255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	0
0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	0	0	0
0	0	0	0	0	0	0	0	0

]
w1 = [1 -2 1]
w2 =[1;-2;1]

Calculate the convolution both analytically and using octave/matlab. What are the roles of *w1* and *w2?*

I=[

0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0

```
0 0 0 255 255 255 255 0 0 0
0 0 0 255 255 255 255 0 0 0
0 0 0 255 255 255 255 0 0 0
0 0 0 255 255 255 255 0 0 0
0 0 0 255 255 255 255 0 0 0
0 0 0 0 0 0 0 0 0 0];
w1 = [1 -2 1]
w2 =[1;-2;1]
Iout1 = conv2(I,w1);
Iout2 = conv2(I,w2);
```

Iout1=

0	0	0	0	0	0	0	0	0	0	0	0
0	255	-255	0	0	0	0	0	0	-255	255	0
0	255	-255	0	0	0	0	0	0	-255	255	0
0	255	-255	0	0	0	0	0	0	-255	255	0
0	0	0	255	-255	0	0	-255	255	0	0	0
0	0	0	255	-255	0	0	-255	255	0	0	0
0	0	0	255	-255	0	0	-255	255	0	0	0
0	0	0	255	-255	0	0	-255	255	0	0	0
0	0	0	255	-255	0	0	-255	255	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Iout2 =

0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	255	255	255	255	0
0	-255	-255	-255	-255	-255	-255	-255	-255	0
0	0	0	0	0	0	0	0	0	0
0	-255	-255	0	0	0	0	-255	-255	0
0	255	255	0	0	0	0	255	255	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	-255	-255	-255	-255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	0	0	0	0	0	0

first filter is used to detect vertical edges, and second one is used to detect horizontal edges

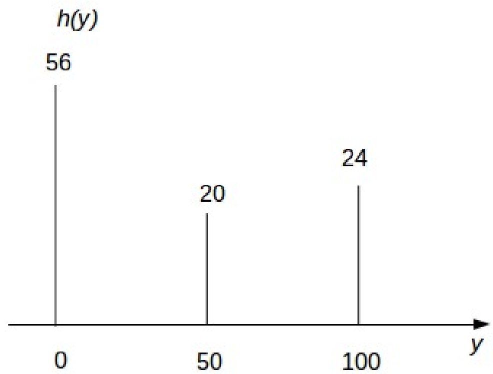
3) Apply the Sobel in Figure 3.41 (d),(e) to the *I* matrix above

4) Draw the histogram of the following matrix

I2 = [

0	0	0	0	0	0	0	0
100	100	100	100	100	100	100	0
100	100	100	100	100	100	100	0
100	100	100	100	100	100	100	0
0	50	50	50	50	0	0	0
0	50	50	50	50	0	0	0
0	50	50	50	50	0	0	0
0	50	50	50	50	0	0	0
0	50	50	50	50	0	0	0
0	0	0	0	0	0	0	0

]

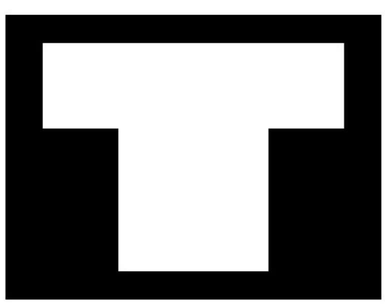


5) Suggest an intensity transformation function that can increase the contrast of the previous image

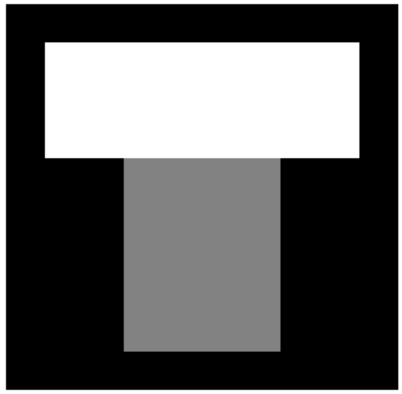
→ Original image:

y2 = 0 if y<10, y2=255 if y>=10

This way, the foreground will be all white, and the background will be all black



Another function
y2= 0 if y<10, y2 = 128 if 10<=y<70 , y2= 255 if y>=70



6) In Gimp, in the Filters → blur. There is an option to (Pixelize). Write an octave/matlab program to do the same task

```
function I2 = pixelize2(I1,width,height)
% I1 is the imput image, width and height are the widths and heghts of different blocks
[tot_rows,tot_cols] = size(I1);
num_block_rows = ceil(tot_rows/height);
num_block_cols = ceil(tot_cols/width);
I2 = I1; % output image
for u=1:num_block_rows
  for v = 1:num_block_cols
    Iblock = I1(1+(u-1)*height:min(height+(u-1)*height,tot_rows),1+(v-1)*width:min(width+(v-1)*width,tot_cols));
    % I have chosen the min of the argument and the tot_rows or tot_cols to take care of the edges
    I2(1+(u-1)*height:min(height+(u-1)*height,tot_rows),1+(v-1)*width:min(width+(v-1)*width,tot_cols)) = round(mean(Iblock(:)));
  end
end
```

7) In Gimp, in the Filters → blur. There is an option to (Motion blur). Write an octave/matlab program to do the task of (Linear) blur.

Linear bluring is done by a convolution between a mask and the original image. Depending on the mask shape, bluring will be in a cerain direction. For example, for a diagonal 45 degrees blurring and length of 4 pixels, the mask will be

¼ 0 0 0
0 ¼ 0 0
0 0 ¼ 0
0 0 0 ¼

```
function I2 = motion_blur(I,h_blur)
I2 = conv2(I,h_blur);
```

A more accurate and complicated implementation can be understood from here

http://www.mathworks.com/matlabcentral/answers/15895-what-is-the-kernel-of-linear-motion-blur-in-fspecial-function

and in octave, after loading the image processing package, writed (edit fspecial) and find (motion) to understand how it is written

This is a snapshot from a paper that describes motion bluring

II. MATHEMATICAL MODEL OF LINEAR MOTION BLURRING AND ITS ATTRIBUTES

Commonly used linear model for image blur is given by:

$$g(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h(x,\alpha,y,\beta) f(\alpha,\beta)\, d\alpha\, d\beta \tag{1}$$

where $h(x,\alpha,y,\beta)$ is a linear *PSF*, $f(x,y)$ is the ideal image, $g(x,y)$ is the observed image .If we consider the Spatially Invariant case of uniform linear motion along horizontal direction, the *PSF* $h(x,y)$ is given by:

$$h(x,y) = \begin{cases} \frac{1}{L} & if \sqrt{x^2+y^2} \le \frac{L}{2} \; and \; \frac{x}{y} = -\tan\varphi \\ 0 & otherwise \end{cases} \tag{2}$$

As seen in equation(2), motion blur depends on two parameters: Motion length(L) and Motion direction (φ).The