Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

4C8 – DIGITAL IMAGE AND VIDEO PROCESSING

Lab 2 – 2D Signal Processing

Department of Electrical and Electronics

(e-Report submission)

Table of Content

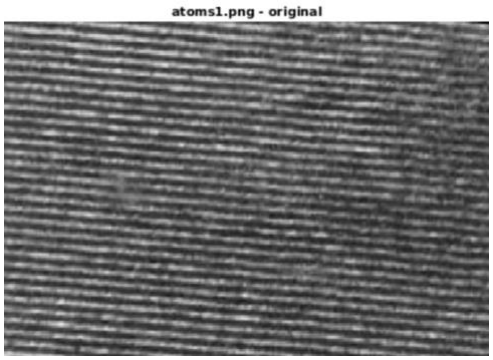
Submitted by:

Rohan Taneja
19323238

Submission: 08/03/2021

1. Low Pass Filter/Separable Filter

The image shown is for comparison of convoluted images with the original image.



1.1. Convolution Mask 2D – Transfer Function

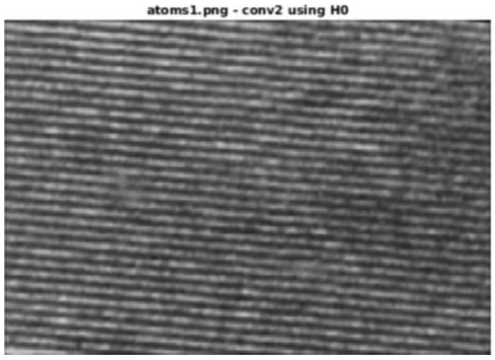
```
%1.1
H0 = [1 3 1;3 9 3;1 3 1]/25;
```

The 2D mask provided in the lab is written as showing in code snippet 1.1.

1.2. I_0 – 2D Convolution Filter Applied

Using function conv2 on image I using mask H0 in the code snippet 1.2. The resulting image is shown your right.

```
%1.2
I0 = conv2(I,H0, 'same');
figure; imshow(I0); axis image;
```



1.3. H0 – Separable Mask

```
%1.3
C = [1; 3; 1]/5;
R = [1 3 1]/5;
M = C*R;
```

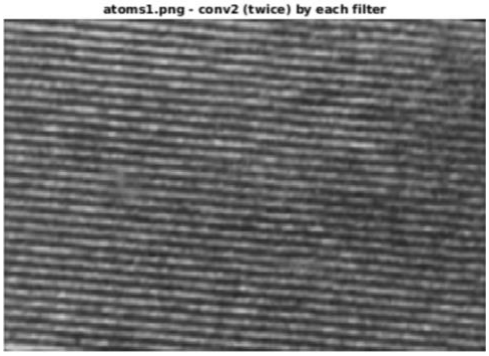
As specified in the code. The mask H0 is a separable mask written in identical column and a row matrix – C and R respectively.

On multiplying C and R, we obtain M. M is convolution 2D mask equal to original mask H0.

1.4. Convolution Applied using each 1D masks

The code snippet 1.4 – shows recursive call of conv2 on the image I. First, the 1D transfer function is applied using column mask onto the original image followed by recursive call of row mask and the resulting

```
% 1.4
I0_new =  conv2(conv2(I,C,'same'),R,'same');
figure; imshow(I0_new); axis image; title(file
```



1.5. Mean Absolute Error between Images

The code snippet 1.5 – shows calculation of mean absolute error between the new image masked using application of 1D mask along X and Y direction. The error is in range of exp(-17), which proves that both images are nearly/exactly the same after applying 1D convolution in each direction or directly applying the 2D convolution.

```
%1.5
N = numel(I0);
err = sum(abs(I0_new(:)-I0(:)))/N
```

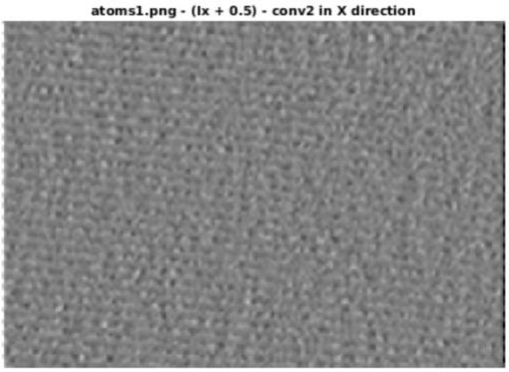
```
err =

   2.9069e-17
```

2. Computing the Image Gradient

2.1. Horizontal Derivative with mask H_x

```
%2.1
Hx = [1 0 -1];
Ix = conv2(I0, Hx, 'same');
figure; imshow(Ix+0.5); axis image;
```

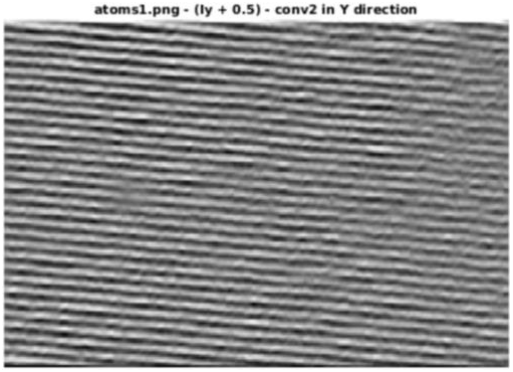
In the code snippet 2.1 we specify the 1D mask for H_x and apply convolution on I_0. The image shown is resulting in Ix horizontal derivative with addition 0.5 points to each value.



2.2. Vertical Derivative with mask H_y

```
%2.2
Hy = [1; 0; -1];
Iy = conv2(I0, Hy, 'same');
figure; imshow(Iy+0.5); axis image;
```

In the code snippet 2.2 we specify the 1D mask for Hx and apply convolution on I0. The image shown is resulting in Iy - vertical derivative with addition 0.5 points to each value.

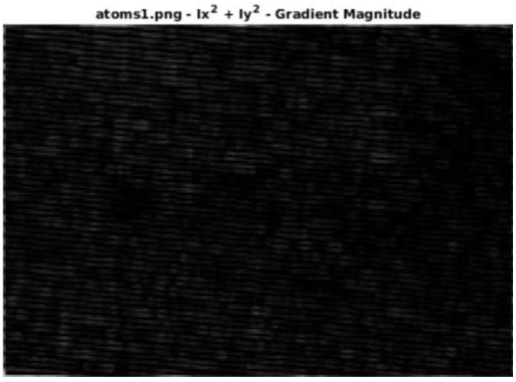


3. Gradient Magnitude and Angle Maps

3.1. Image Gradient Magnitude for I_x and I_y

```
%3.1
grad_mag = Ix.^2 + Iy.^2;
figure; imshow(grad_mag); axis image;
```

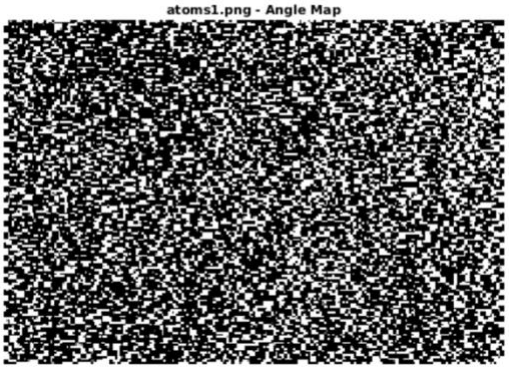
In the code snippet 3.1 we calculate gradient magnitude in grad_mag variable. The image shown is resulting in addition of squaring I_x and I_y derivates from the section 2.



3.2. Angle map using Theta

```
%3.2
theta = atan(Ix./Iy) * (180./pi);
figure; imshow(theta); axis image;
```

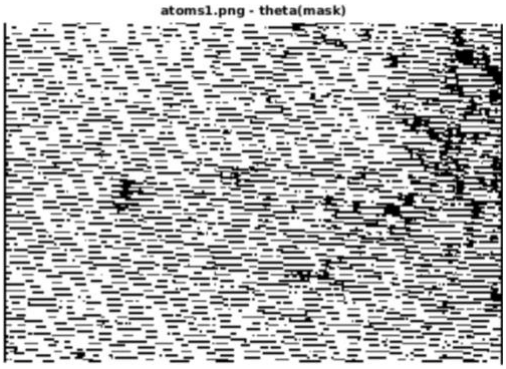
In the code snippet 3.2 we apply function provided in the manual to calculate the theta (in degrees). The result shown is the angle map for the image calculated earlier.



4. Orientation Measurement

4.1. Boolean Mask with specified range

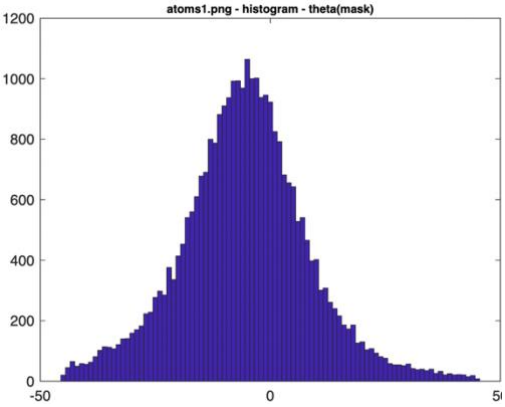
```
%4.1
mask = (grad_mag > 0.01) & (45 > theta & theta > -45);
figure; imshow(mask); axis image;
```



In the code snippet 4.1 we apply the mask provided in the manual. The result shown is the image of matrix with either 0 or 1 value at respective coordinates satisfying the conditions of the mask.

4.2. Histogram for theta(mask) vector

```
%4.2
theta_mask = theta(mask);
figure; hist(theta_mask(:), -45:1:45);
```



In the code snippet 4.2 we compute the histogram of vector generated using mask to the corresponding theta values. The plot is projected for theta between -45 and +45 degree and the peak corresponding to its orientation.

4.3. Estimate orientation of the image

Using the histogram the peak points towards an estimate of the image orientation as commented earlier. To find the exact value corresponding to the peak we have written code snippet 4.3.

```
%4.3
theta_cap = sum(theta_mask)/numel(theta_mask);
```

For the image – ‘atoms1.png’ – which corresponds to -5 degree in orientation to normal. The result obtained using the code is as shown below.

```
theta_cap =

   -5.4728
```

The theta_cap calculates to -5.4728 degrees which in turn is relevant to given data despite a little error which possibly can be a human error.

4.4. Evaluate the orientation of provided images

We run the code snippet 4.3 on atoms2.png which corresponds to orientation shift towards +3 degrees. The result obtained is +3.3923 degrees – nearly equal to given data.

```
theta_cap =

    3.3923
```

We again perform the run on atoms3.png – which corresponds to normal orientation that is 0 degrees. The result obtained is 0.0841 degrees.

```
theta_cap =

    0.0841
```

Hence, our experiment performed on the provided images is successful with the obtained results.