*Department of Computer Science and Software Engineering*
*Comp 6771 Image Processing*
*Assignment 4*

---

*Mandana Samiei, ID: 40059116* *26/11/2017*

## *Theoretical Questions*

**1-** Otsu's thresholding algorithm

$\sigma_B{}^2$ is the between-class variance, we are supposed to prove the following equation:

$$\sigma_B{}^2 = P_1P_2\,(m_1 - m_2)^2 = \frac{(m_GP_1 - m)^2}{P_1(1-P_1)}$$

Let {0, 1, 2,…, L-1} denote the L intensity Levels in a digital image of size M * N pixels, and $n_i$ denote the number of pixels in each intensity level $i$. So the total number of pixels in the image will be $MN = n_0 + n_1 + \cdots + n_{L-1}$.
In the normalized histogram: probability density function of intensity levels: $p_i = {}^{n_i}/_{MN}$
Also sum of all the probability density function results 1:

$\sum_{i=0}^{L-1} p_i = 1 \quad p_i \geq 0$ (1)

Suppose that we select a threshold $T(k) = k\,; 0 < k < L-1$ and use it to threshold image into two classes $c1$: $[0-k]$ and $c2$: $[k+1-L-1]$.
The probability $P_1(k)$, that a pixel is assigned to class $c1$ is given by cumulative sum:

$P(c1) = P_1(k) = \sum_{i=0}^{k} p_i$ (2)

$P(c2) = P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$ (3)

The mean intensity value of pixels assigned to class $c1$: (based on Bayse's formula)

$m_1(k) = \sum_{i=0}^{k} iP(i|c1) = \sum_{i=0}^{k} i\frac{P(c1|i)P(i)}{P(c1)} = \frac{1}{P_1(k)}\sum_{i=0}^{k} ip_i$ (4)

The mean intensity value of pixels assigned to class $c2$:

$m_2(k) = \sum_{i=k+1}^{L-1} iP(i|c2) = \sum_{i=K+1}^{L-1} i\frac{P(c2|i)P(i)}{P(c2)} = \frac{1}{P2(k)}\sum_{i=k+1}^{L-1} ip_i$ (5)

The cumulative mean up to level k (average intensity):

$m(k) = \sum_{i=0}^{k} ip_i$ (6)

The average intensity of the entire image: $m_G = \sum_{i=0}^{L-1} ip_i$ (7)

We can drive these two formula from the above relations:

$P_1(k) + P_2(k) = \sum_{i=0}^{k} p_i + \sum_{i=k+1}^{L-1} p_i = \sum_{i=0}^{L-1} p_i = 1 \rightarrow P_1(k) + P_2(k) = 1$ (8)

$P_1(k)m_1(k) + P_2(k)m_2(k) = P_1(k)\left(\frac{1}{P_1(k)}\sum_{i=0}^{k} ip_i\right) + P_2(k)\left(\frac{1}{P2(k)}\sum_{i=K+1}^{L-1} ip_i\right) = \sum_{i=0}^{L-1} ip_i =$

$m_G \rightarrow P_1(k)m_1(k) + P_2(k)m_2(k) = m_G$ (9)

The between class variance is:

$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 = P_1P_2(m_1 - m_2)^2$

In order to prove the above formula, we are going to use (8) and (9) relations: (Green texts are substitute by a new value in each line)

$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 = P_1[m_1 - (P_1m_1 + P_2m_2)]^2 + P_2[m_2 - (P_1m_1 + P_2m_2)]^2 = P_1\left[m_1 - \left((1-P_2)m_1 + P_2m_2\right)\right]^2 + P_2[m_2 - (P_1m_1 + (1-P_1)m_2)]^2$

$= P_1(P_2m_1 - P_2m_2)^2 + P_2(P_1m_2 - P_1m_1)^2 = P_1P_2^2(m_1 - m_2)^2 + P_2P_1^2(m_1 - m_2)^2$

$= (m_1 - m_2)^2P_1P_2(P_1 + P_2) = P_1P_2(m_1 - m_2)^2$ $(according\ to\ this\ fact\ P_1 + P_2 = 1)$

$$\sigma_B{}^2 = P_1P_2\,(m_1 - m_2)^2 = \frac{(m_GP_1 - m)^2}{P_1(1-P_1)}$$

Now, we want to prove the second part of the above relation:

First, we need to achieve the value of $m_1$ and $m_2$ and after that we are going to substitute these parameters in the first part of this formula:

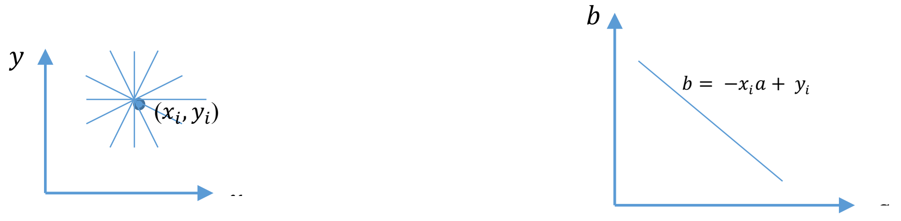
$$m_1(k) = \frac{1}{P_1(k)}\sum_{i=0}^{k} ip_i = \frac{1}{P_1(k)}\big(m(k)\big) = \frac{m(k)}{P_1(k)}$$

$$m_2(k) = \frac{1}{P_2(k)}\sum_{i=K+1}^{L-1} ip_i = \frac{1}{P_2(k)}\left(\sum_{i=0}^{L-1} ip_i - \sum_{i=0}^{k} ip_i\right) = \frac{1}{1 - P_1(k)}(m_G - m(k))$$

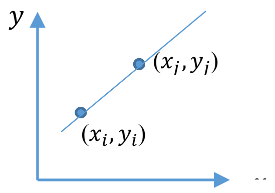
$$\sigma_B{}^2 = P_1P_2\,(m_1 - m_2)^2 = P_1P_2\,\left(\frac{m(k)}{P_1(k)} - \frac{m_G - m(k)}{1-P_1(k)}\right)^2 = P_1(1 - P_1)\left[\frac{m(1-P_1)-(m_G-m)(P_1)}{P_1(1-P_1)}\right]^2 = \frac{(m - mP_1 + mP_1 - P_1m_G)^2}{P_1(1-P_1)} = \frac{(P_1m_G - m)^2}{P_1(1-P_1)}$$

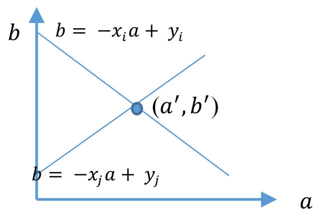
**2- Hough Transform for lines cannot be carried out in the Cartesian (x, y) coordinate system explain why and give detail of how it works.**

The purpose of Hough transform is to find sets of pixels lies on curves of a specified shape. Consider that we have a point $(x_i, y_i)$ and a general line equation $y_i = ax_i + bi$. Infinitely many lines pass through $(x_i, y_i)$ and all of them satisfying equation $y_i = ax_i + b$ for different values of a and b.



If we convert the general equation in xy-plane to $b = -x_ia + y_i$ in ab-parameter space, each point $(x_i, y_i)$ in xy-plane is equivalent to a single line in parameter space.
Similarly another edge pixel also has a line in parameter space: $b = -x_ja + y_j$





The intersection point $(a', b')$ of these lines indicate the slope and intercept of the line containing both $(x_j, y_j)$ and $(x_i, y_i)$ in the xy-plane. All the points on this line in xy-plane has a line in parameter space that intersect at point $(a', b')$.
Similarly, for n points in xy-plane, we will have n lines in parameter space. Each intersection point (s,p) in parameter space indicate a line in Cartesian coordination that is containing those points who has a line in parameter space that has intersect with (s, p).
In practice, ab parameter space is quantized as a 2-dimensional matrix of accumulator cells.
The parameter lines are drawn in accumulator matrix cells and increment the cell value by one.

$$\begin{array}{c|c|c|c|c|c|}
 & b_{min} & & 0 & & b_{max} \\ \hline
a_{min} & & & \cdots & & \\ \hline
 & & & & & \\ \hline
0 & & & \vdots & & \\ \hline
 & & & & & \\ \hline
a_{max} & & & \vdots & & \\ \hline
\end{array}$$

The maximum values in the accumulator matrix indicate the most probable line parameters.
As a result, we can find significant edge lines containing the most edge pixels in xy-plane.
A problem with this approach is for those lines that have infinity slope $a = \infty$ and $b = \infty$.
a' approaches to infinity as the line approaches to the vertical direction.
This problem can be solved if use the normal representation of a line in polar coordination.

$$x \cos\theta + y \sin\theta = \rho$$

Then each image pixel corresponds to a curve in the parameter space. The parameter lines are again found with accumulator matrix.

**3- Show that the number of operations required to implement the accumulator cell approach discussed in section 10.2.7 is linear if n is the number of non-background points in the image plane.**

$$\begin{array}{c|c|c|c|c|c|}
 & \theta_{min} & & 0^\circ & & \theta_{max} \\ \hline
\rho_{min} & & & \cdots & & \\ \hline
 & & & & & \\ \hline
0 & & & \vdots & & \\ \hline
 & & & & & \\ \hline
\rho_{max} & & & \vdots & & \\ \hline
\end{array}$$

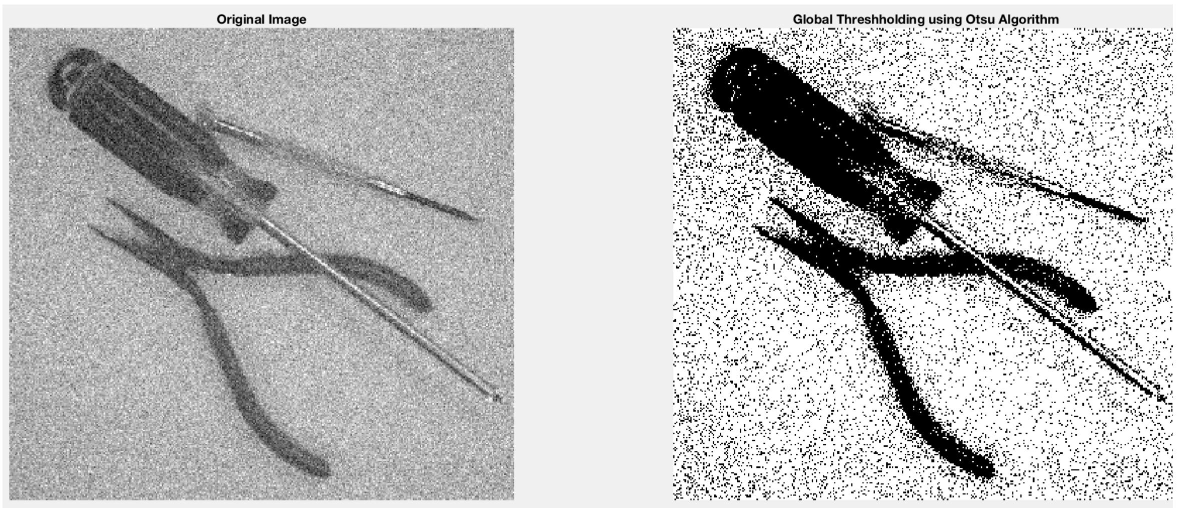
For every non-background point $(x_k, y_k)$ in the xy-plane we let $\theta$ equal to each of the allowed subdivisions values between $-90°$ to $90^{\circ}$. Then we will solve the correspond polar equation $(x \cos\theta + y \sin\theta = \rho)$ to reach $\rho$. After that we round off the value of $\rho$ to the nearest allowed cell value along $\rho$ axis. All of the above-mentioned operations are linear, the first one is an iteration through a finite list of valid $\theta$, the second one is a calculation of $\cos\theta$and $\sin\theta$, multiplying them by $x_k$ and $y_k$ that are constant values and then adding them to reach the $\rho$. As the value of $\theta$ is bounded and also we have the value of $x_k$ and $y_k$, the time complexity of these operations is constant O(1).
If we have 'n' non-background points, we need to do these operations for n times so the time complexity of this algorithm for n point is:

$$n * O(1) \equiv O(n)$$

## *Programming Questions*

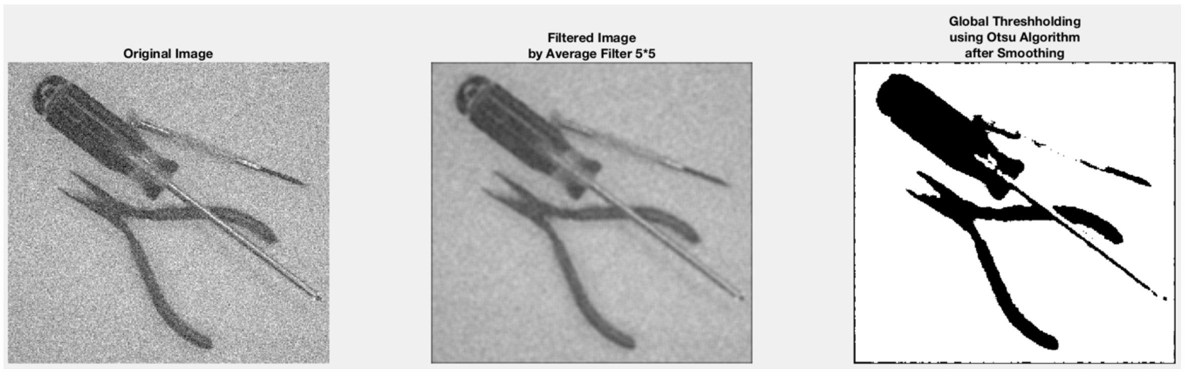
### **4-** Otsu Thresholding

a)
```
%%----------Question 4-a -------------
%%--------- Mandana Samiei-------------
%%--------- Student Id: 40059116 ------
%% Applying Otsu method to a noisy image

I = im2double(imread('tools_noisy.png'));
[level, EM] = graythresh(I);
BW = imbinarize(I, level);
figure;
subplot(1,2,1);
imshow(I); title('Original Image');
subplot(1,2,2);
imshow(BW); title('Global Threshholding using Otsu Algorithm');
```



b)
```
%%----------Question 4-b --------------
%%--------- Mandana Samiei-------------
%%--------- Student Id: 40059116 ------
%% Applying Otsu method to a noisy image after smoothing the image

I = im2double(imread('tools_noisy.png'));

H = ones(5,5) / 25;
Filtered_Image = imfilter(I, H);
```

```
level = graythresh(Filtered_Image);
BW = imbinarize(Filtered_Image,level);

figure;
subplot(1,3,1);
imshow(I); title('Original Image');
subplot(1,3,2);
imshow(Filtered_Image); title({'Filtered Image', 'by Average Filter 5*5'});
subplot(1,3,3);
imshow(BW); title({'Global Threshholding', 'using Otsu Algorithm', 'after
Smoothing'});
```



Compare a) and b):

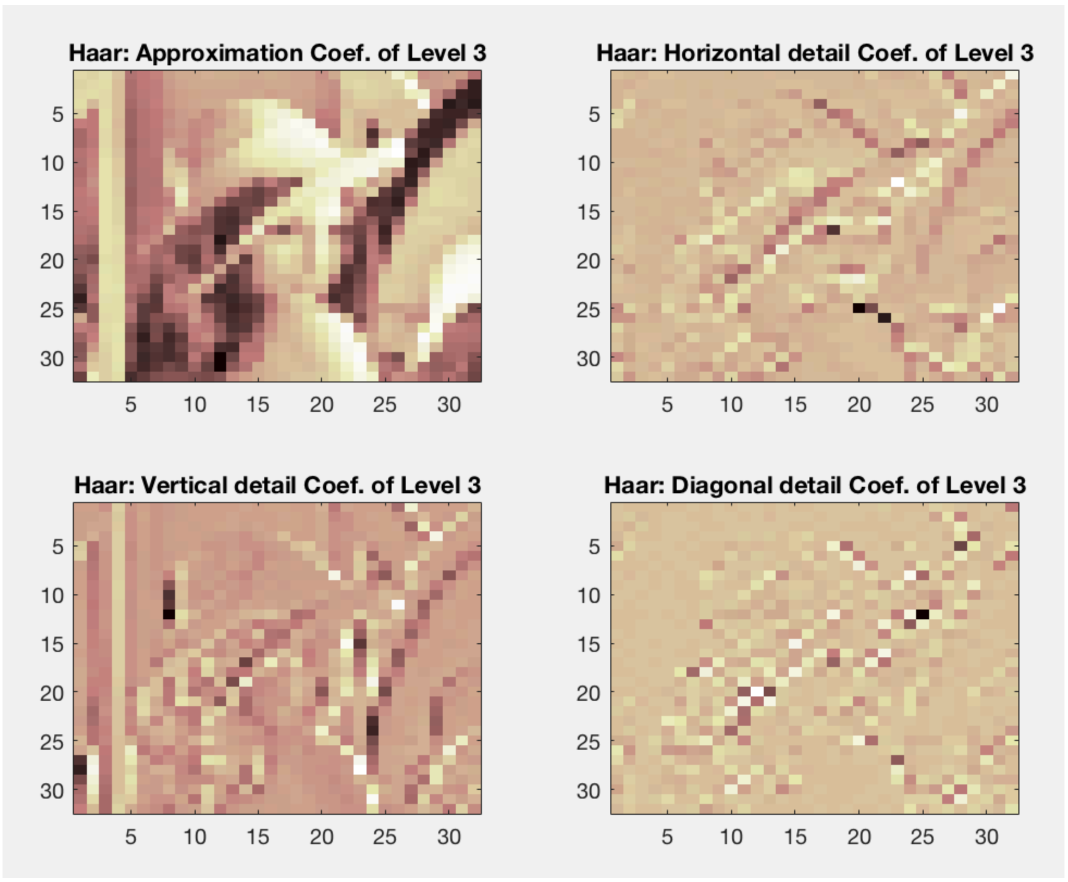
In part ‘a’ after applying Otsu thresholding algorithm, objects have been segmented and seemed clearer but also noise is boosted but in part ‘b’ that we applied a smoothing filter before Otsu thresholding, in this case noise is eliminated and segmentation is nearly perfect. However, we can see a slight distortion of the boundary between object and background in the segmented smoothed image that was caused by the blurring of boundaries because of using a smoothing filter.

## 5- Wavelet Transform

a)

```
%%----------Question 5-a --------------
%% Applying Wavelet transform using Haar Wavelet

I = imread('lena.bmp');

if ndims(I) == 3
    J = rgb2gray(I);
end
[Lo_D,Hi_D,Lo_R,Hi_R] = wfilters('haar');
[c1A,c1H,c1V,c1D] = dwt2(J, 'haar');
%%cA: approximation coefficients matrix
%%cH: Horizontal edges details coefficients matrices
%%cV: Vertical edges details coefficients matrices
%%cD: Diagonal edges details coefficients matrices
[c2A,c2H,c2V,c2D] = dwt2(c1A,Lo_D,Hi_D); % Level 2
[c3A,c3H,c3V,c3D] = dwt2(c2A,Lo_D,Hi_D); % Level 3

TH = 50;
figure;
subplot(2,2,1);
imagesc(c3A);
colormap pink(255);
title('Haar: Approximation Coef. of Level 3');
```

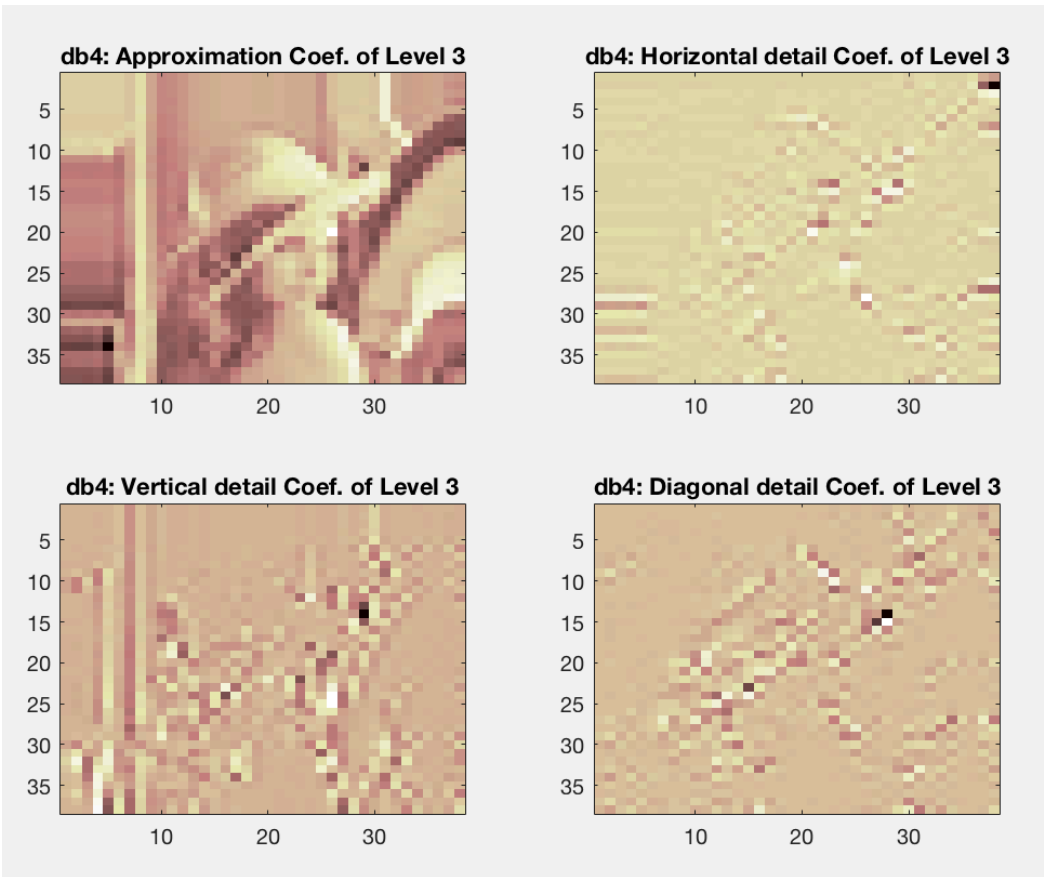
```
subplot(2,2,2)
imagesc(c3H);
title('Haar: Horizontal detail Coef. of Level 3');

subplot(2,2,3)
imagesc(c3V);
title('Haar: Vertical detail Coef. of Level 3');

subplot(2,2,4)
imagesc(c3D);
title('Haar: Diagonal detail Coef. of Level 3');
```



b)

```
%%----------Question 5-b --------------
%% Applying Wavelet transform using Daubechies-4 Wavelet
I = imread('lena.bmp');

if ndims(I) == 3
    J = rgb2gray(I);
end
[Lo_D,Hi_D,Lo_R,Hi_R] = wfilters('db4');
[c1A,c1H,c1V,c1D] = dwt2(J, 'db4');
%%cA: approximation coefficients matrix
%%cH: Horizontal edges details coefficients matrices
%%cV: Vertical edges details coefficients matrices
%%cD: Diagonal edges details coefficients matrices
[c2A,c2H,c2V,c2D] = dwt2(c1A,Lo_D,Hi_D); % Level 2
[c3A,c3H,c3V,c3D] = dwt2(c2A,Lo_D,Hi_D); % Level 3
```

```
figure;
subplot(2,2,1);
imagesc(c3A);
colormap pink(255);
title('db4: Approximation Coef. of Level 3');

subplot(2,2,2)
imagesc(c3H);
title('db4: Horizontal detail Coef. of Level 3');

subplot(2,2,3)
imagesc(c3V);
title('db4: Vertical detail Coef. of Level 3');

subplot(2,2,4)
imagesc(c3D);
title('db4: Diagonal detail Coef. of Level 3');
```



c)
According to my visual comparison, approximation images in Daubechies-4 transform are more precise and accurate in terms of detecting edges and high frequency changes. Also, I can see that transform window in the Daubechies-4 is smaller that Haar transform that allows us to detect changes in a higher precision and smoother.