CENG 466
ASSIGNMENT 2
REPORT

Prepared by : Asli Gulen 112887-5
Kayhan Ince 116246-0
Submitted to : Turan Yuksel

Quantization Part

As described in the specifications, we divide the image into 8x8 pixel segments and work on these pixels.

When we examine the characteristic of the Fourier Transformed form of the image, we find out the following properties:

For the real part;

- There exists a unique value, which is quite large, at the top first position of every 8-by-8 block of the entire image
- Every top left 3 values have their corresponding symmetries on the right hand side of the same row according to the 5th element of the row
- Above situation also holds for the first column.
- Every element of the remaining 7x7 part of the block is symmetric with respect to the mid-element of the 8-by-8 block.

For the imaginary part the similar symmetry conditions are present except the sign difference coming from the conjugate symmetry of the complex numbers.

Details of the symmetry properties are shown in the appendix.

Encoding: Due to this results and having the knowledge that the information of an image after Fourier Transformation is mostly focused on the four corners, we rather keep just the values mentioned above, which correspond to the positions of the entire image.

0,0	0,1	0,2	0,3				
1,0	1,1	1,2					
2,0	1,2	2,2					
3,0							
	6,1	6,2					
	7,1	7,2					

In order to be able to store the floating-point values of the transform table, we have to map each value into the range [0,255] of unsigned char using a normalization process by finding the maximum and minimum of these elements. Normalizing process is this:

$$coefficient = \frac{Max - Min}{255}$$

$$normalizedValue = \frac{(value - Min)}{coefficient}$$

This operation leads to some loss of data but we try to minimize this effect that mainly depends on the value of coefficient by keeping the coefficient small.

Again when we examine the transform table, [0][0] entry is a very large positive number (of the order of 10.000) but the remaining elements are respectively small

([-1000,1000]) . If we consider [0][0] element in normalization, the coefficient will be a number like 40 and this will cause a very large error during normalization. We overcome this problem by storing the [0][0] element apart from the remaining elements. This optimization decreased the error in the normalization in a perceivable amount.

We find the minimum and maximum of the selected elements (except [0][0] of real part) of both real and imaginary parts and calculate a common coefficient for all the numbers to be normalized.

Normalized values and other necessary data are stored in the resulting 8x16 bytes segment is:

0,1- re	0,2- re	0,3- re	(COEFFICIENT)												
1,0- re	1,1- re	1,2- re	6,1- re	6,2- re											
2,0- re	2,1- re	2,2- re	7,1- re	7,2- re											
3,0- re	(MIN)		(0,0) - RE												
1,0- im	0,1--im	0,2-im	0,3-im	(MARK)											
2,0- im	1,1-im	1,2-im	6,1-im	6,2-im											
3,0- im	2,1-im	2,2-im	7,1-im	7,2-im											

We write only five columns and seven rows of these 8x16 segments to the output image because gathering the empty spaces together results more compression during png conversion.

To eliminate the similar segments to be stored repeatedly, we calculate the rms values for every consecutive segment pairs while tracing through the image, and compare the results with a specified rms value. If the result is less than the acceptable error amount we only take the information on one of the segments and put a mark on the other segment to indicate that the corresponding values are stored on another segment, which is directed by the two pointers of x and y dimensions. These pointers are situated at positions [4][0]-[4][1] and [4][2]-[4][3] (two bytes each), since we don't have any other information there in this case. Elsewhere, if rms value is greater than the specified error value, we take both segments into account.

Instead of using short int we prefer implementing conversion of int to 2 bytes short int and its reverse by hand, not to be machine-dependent. Also instead of using a 4-bytes long float coefficient, we convert the coefficient into fixed-point number of 2-bytes long. Maximum value of coefficient did not exceed 30, a precision of 1/1000 is obtained with 2 bytes.

Decoding: The reverse process of encoding is followed, calculating the original values form the quantization table entries by the formula:

$$originalValue = (normalizedValue * coefficient) + \min,$$

and then restoring these blocks according to the symmetry properties. After restoring the real and imaginary parts of transform values, inverse Fourier transform is applied and original pixel values of the 8x8 image segment are obtained.

Image	**Initial Size**	**Compressed Size**	**RMS**	**Compression Rate**
Lena	245.805	141.973	4.58	42%
Oguz and selim	155.198	110.959	2.88	26%
Generated 1	25.823	31.421	11.78	- 22%

Histogram Equalization

In trichromatic images the image is processed for each color independently. In the process, we first count the number of pixels corresponding to each of the 256 intensity levels for each color, R, G and B. Then, for each color, we calculated the transform function values, s_k, for each intensity level, where the transformation function is:

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n}$$ n: number of total pixels

Afterwards, we mapped s_k values from [0,1] range to the range of [0,255] using the formula :

$$\text{int}\left[\frac{(s - s_{\min})}{1 - s_{\min}} * 255 + 0.5\right]$$

As a final step, we replaced each value in the image with its corresponding s_k value.

Histograms of the original and equalized images are in the appendix.

If we were to process graylevel images, we needn't to repeat the procedure three times; we would apply the algorithm just once.

Noise Reduction

We used Median Filtering method for noise reduction. An alternative filter could be averaging filter but it would distribute the noise to the non-noisy pixels rather than reducing it.

In the median filter, as noisy pixel is differs in value with respect to other pixels in the mask, after sorting, it will be placed at the end or beginning of the list and the median element will be the pivot element itself or another pixel near to the pivot element in intensity.

We first tried 5-by-5 filter, which resulted in good noise reduction but a very blurred image such that it destroys the edges especially in artificial images. Therefore, we agreed that a 3-by-3 filter leads to a better result. We take its 5^{th} element as the median, since the noise value is supposed to be either in the first or last position among the sorted values within the mask, so the 5^{th} element's value is closer to its neighbors.

We then assign the median's value to the pivot element each time shifting the filter.

Edge Detection

We used Sobel Mask pairs. After finding horizontal and vertical edges we take the absolute value of their sum. In order to do this, we coded a function that takes two filters at the same time, convolves the image with both mask individually and returns the specified sum

We normalize the resulting image in order to get a better result.

Here are the convolution mask pair that we use:

-1	-2	-1
0	0	0
1	2	1

Horizontal edge detection filter

-1	0	1
-2	0	2
-1	0	1

Vertical edge detection filter

APPENDIX A

Symmetry properties of the Fourier transformation of the 8x8 image segment:

Real Part :

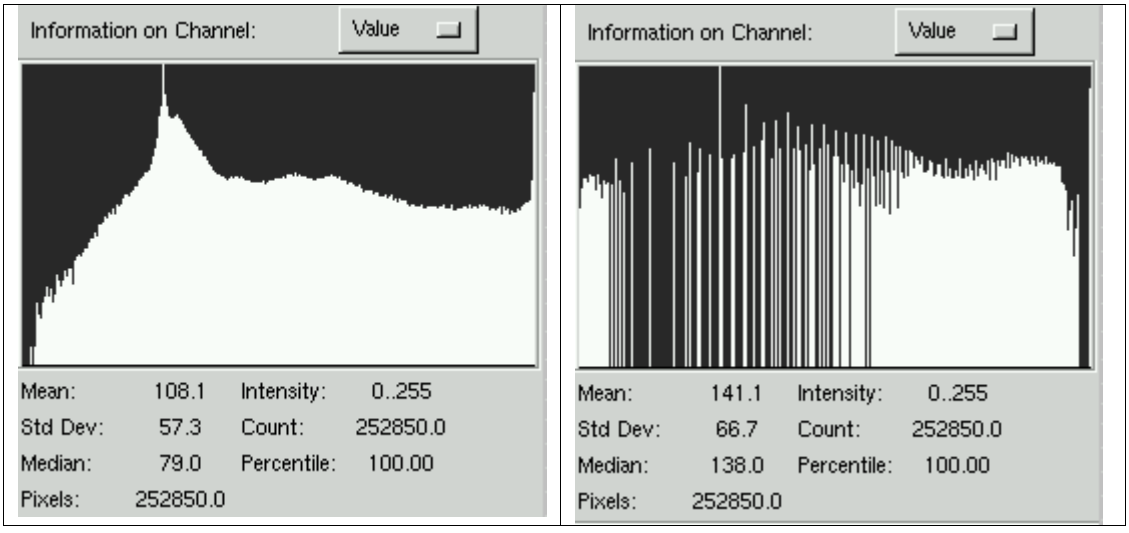
0,0	0,1 = 0,7	0,2 = 0,6	0,3 = 0,5	0,4	0,5 = 0,3	0,6=0,2	0,7=0,1
1,0 = 7,0	1,1 = 7,7	1,2 = 7,6	1,3 = 7,5	1,4 = 7,4	1,5 = 7,3	1,6 = 7,2	1,7 = 7,1
2,0 = 6,0	2,1 = 6,7	2,2 = 6,6	2,3 = 6,5	2,4 = 6,4	2,5 = 6,3	2,6 = 6,2	2,7 = 6,1
3,0 = 5,0	3,1 = 5,7	3,2 = 5,6	3,3 = 5,5	3,4 = 5,4	3,5 = 5,3	3,6 = 5,2	3,7 = 5,1
4,0	4,1 = 4,7	4,2 = 4,6	4,3 = 4,5	4,4	4,5 = 4,3	4,6 = 4,2	4,7 = 4,1
5,0 = 3,0	5,1 = 3,7	5,2 = 3,6	5,3 = 3,5	5,4 = 3,4	5,5 = 3,3	5,6 = 3,2	5,7 = 3,1
6,0 = 2,0	6,1 = 2,7	6,2 = 2,6	6,3 = 2,5	6,4 = 2,4	6,5 = 2,3	6,6 = 2,2	6,7 = 2,1
7,0 = 1,0	7,1 = 1,7	7,2 = 1,6	7,3 = 1,5	7,4 = 1,4	7,5 = 1,3	7,6 = 1,2	7,7 = 1,1

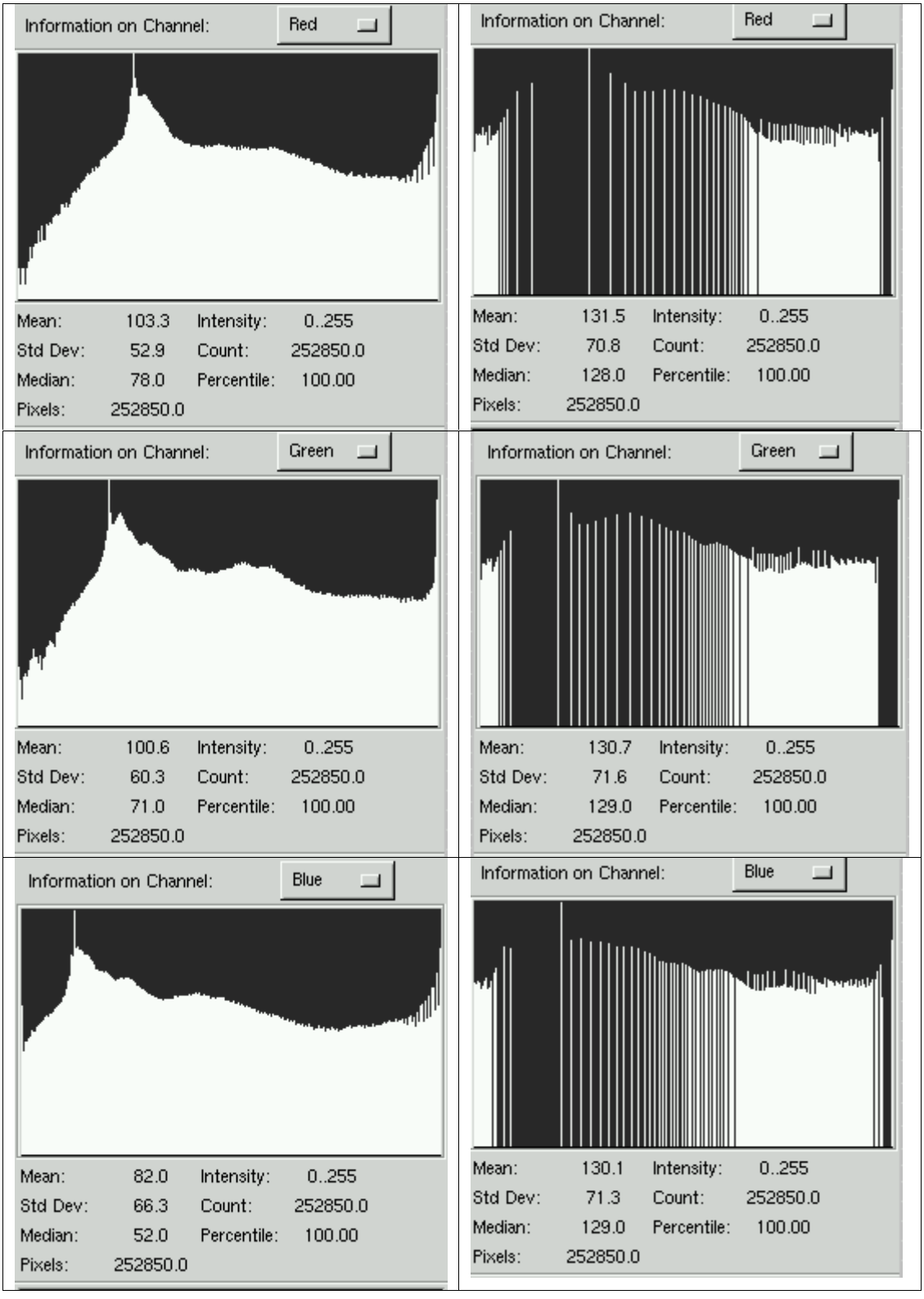
Imaginary Part:

0,0	0,1 = -0,7	0,2 = -0,6	0,3 = -0,5	0,4	0,5 = -0,3	0,6= -0,2	0,7 = -0,1
1,0 = -7,0	1,1 = -7,7	1,2 = -7,6	1,3 = -7,5	1,4 = -7,4	1,5 = -7,3	1,6 = -7,2	1,7 = -7,1
2,0 = -6,0	2,1 = -6,7	2,2 = -6,6	2,3 = -6,5	2,4 = -6,4	2,5 = -6,3	2,6 = -6,2	2,7 = -6,1
3,0 = -5,0	3,1 = -5,7	3,2 = -5,6	3,3 = -5,5	3,4 = -5,4	3,5 = -5,3	3,6 = -5,2	3,7 = -5,1
4,0	4,1 = -4,7	4,2 = -4,6	4,3 = -4,5	4,4	4,5 = -4,3	4,6 = -4,2	4,7 = -4,1
5,0 = -3,0	5,1 = -3,7	5,2 = -3,6	5,3 = -3,5	5,4 = -3,4	5,5 = -3,3	5,6 = -3,2	5,7 = -3,1
6,0 = -2,0	6,1 = -2,7	6,2 = -2,6	6,3 = -2,5	6,4 = -2,4	6,5 = -2,3	6,6 = -2,2	6,7 = -2,1
7,0 = -1,0	7,1 = -1,7	7,2 = -1,6	7,3 = -1,5	7,4 = -1,4	7,5 = -1,3	7,6 = -1,2	7,7 = -1,1

APPENDIX B

Histograms on the left column are histograms of original image and the ones in the right column are histograms of the image after histogram equalization.




Information on Channel:
Red
Mean: 103.3 Intensity: 0..255
Std Dev: 52.9 Count: 252850.0
Median: 78.0 Percentile: 100.00
Pixels: 252850.0
Information on Channel:
Red
Mean: 131.5 Intensity: 0..255
Std Dev: 70.8 Count: 252850.0
Median: 128.0 Percentile: 100.00
Pixels: 252850.0
Information on Channel:
Green
Mean: 100.6 Intensity: 0..255
Std Dev: 60.3 Count: 252850.0
Median: 71.0 Percentile: 100.00
Pixels: 252850.0
Information on Channel:
Green
Mean: 130.7 Intensity: 0..255
Std Dev: 71.6 Count: 252850.0
Median: 129.0 Percentile: 100.00
Pixels: 252850.0
Information on Channel:
Blue
Mean: 82.0 Intensity: 0..255
Std Dev: 66.3 Count: 252850.0
Median: 52.0 Percentile: 100.00
Pixels: 252850.0
Information on Channel:
Blue
Mean: 130.1 Intensity: 0..255
Std Dev: 71.3 Count: 252850.0
Median: 129.0 Percentile: 100.00
Pixels: 252850.0