Korea Advanced Institute of Science and Technology

EE535 Digital Image Processing Spring 2018

Student’s Name: Dinh Vu

Student’s ID: 20184187

**Homework 6**

1. **Parallel beam image reconstruction algorithm**
   1. ***Theoretical background & Programming strategy***

The given sinogram image contains pixels with real values from 0.0 to 2.0, and the number of samples , and the number of views is 180 (1 to 180 degrees). The vertical axis is degree (index i) and the horizontal axis (index j) is parameter . The pixel value of the corresponding point is the value of that can be defined as Radon Transform of which is the intensity function of the original image.

Consequently, back projection is , calculated by:

The back projection image is obtained by discrete summing corresponding to all and combinations (180 in total) for each pixel of the image. However, it requires filtering it because blurring (multiplied by the offsets in the frequency domain) inevitably occurs during back projection.

Blurred image by back projection:

De-blurring process:

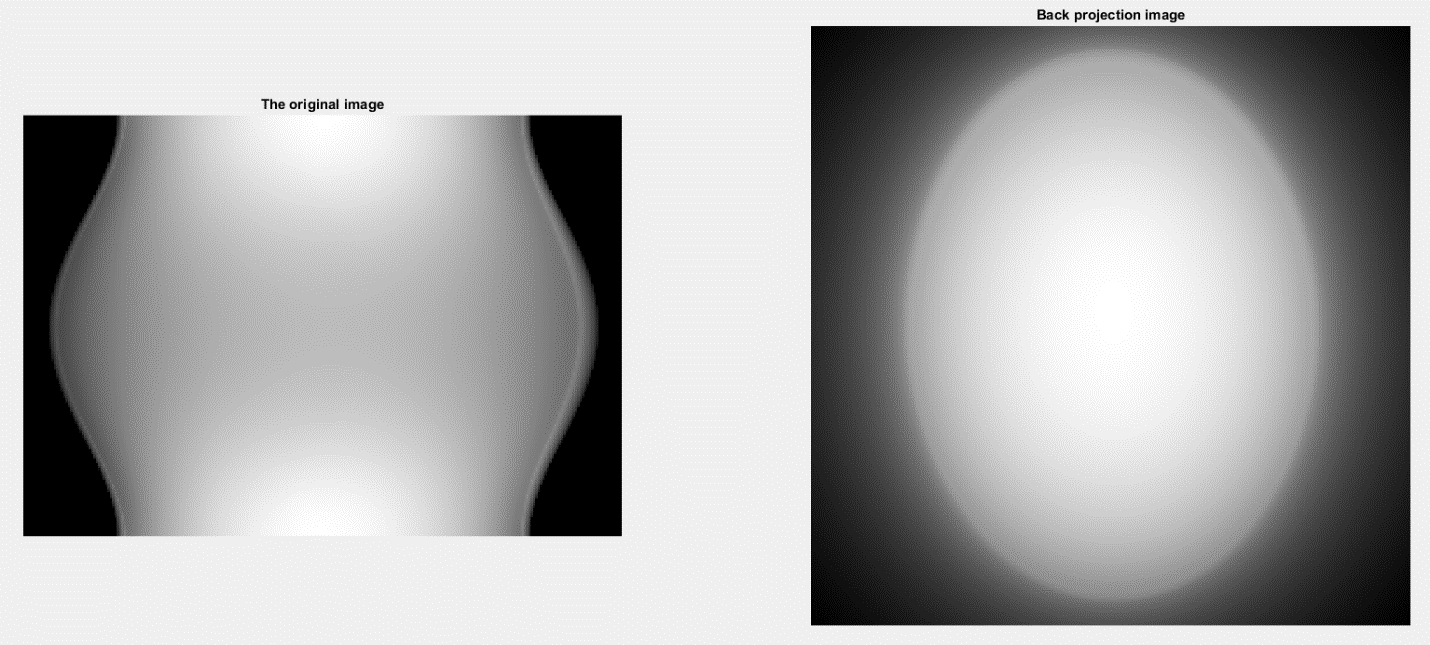
To do this, use the Parallel-Beam mode expression:

is the filter.

Ram-Lak and Shepp-Logan filters are given. These filters suppress the appropriate B to prevent noise amplification. Select appropriate B to consider the trade-off between good resolution and noise amplification. Therefore, before back projection, a new sinogram is created by using filter and 1-D convolution in the direction of the sinogram. Cut view is the intensity of the object center.

* 1. ***Results & Analysis results***

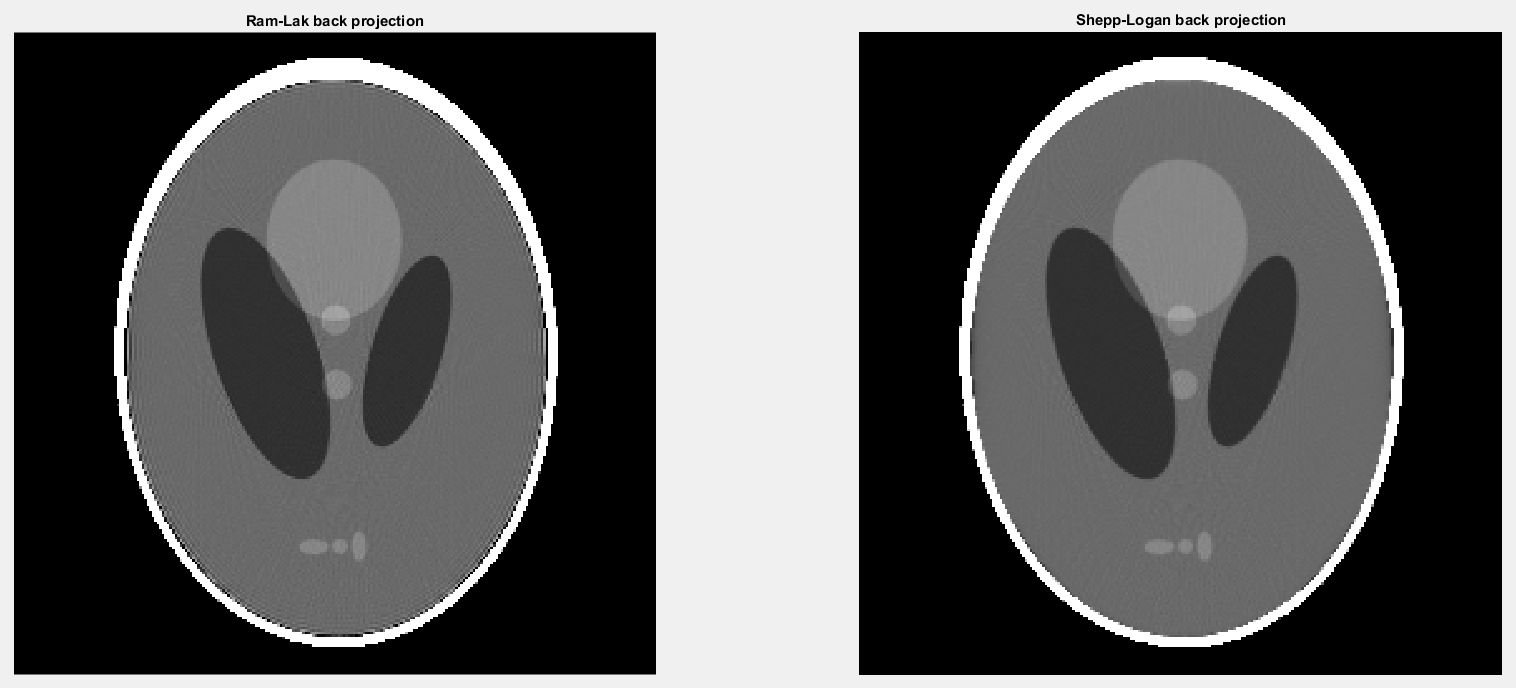
Figure 1.1 shows back projection without filtering. The high frequency region is amplified very much because of term. Therefore, the edge components and noise corresponding to that parts are amplified. Consequently, the boundary is blurred and the details of the center are not visible.



**Figure 1.1.** The original image and the back projection image without filtering

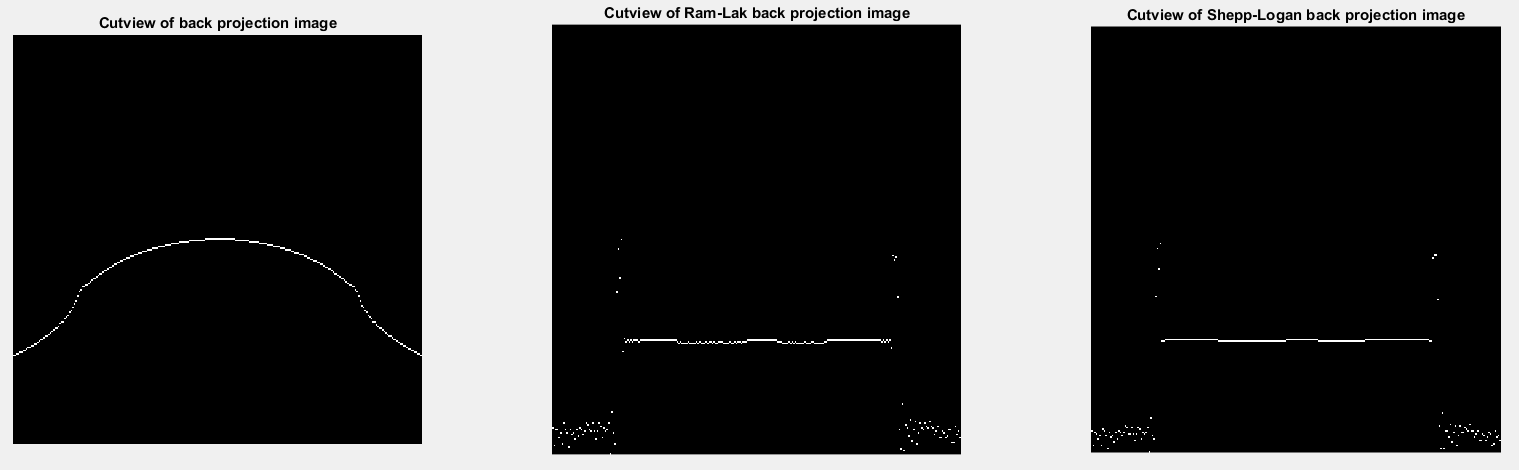
The Ram-Lak filter assuming that is band-limited for spatial frequency B. Therefore, it prevents the edge and noise from being amplified at the proper boundary B. Hence, it can be seen as in Figure 1.2. However, ringing artifacts occurred in the outcome image because it suddenly collapsed at part B. The window size is [0.44, 0.48].

The Shepp-Logan filter suppress the high frequency components by bending it down a little more smoothly in the neighbor of B to reduce the ringing artifact. The boundaries of the geometry became clearer with smaller ringing artifacts.



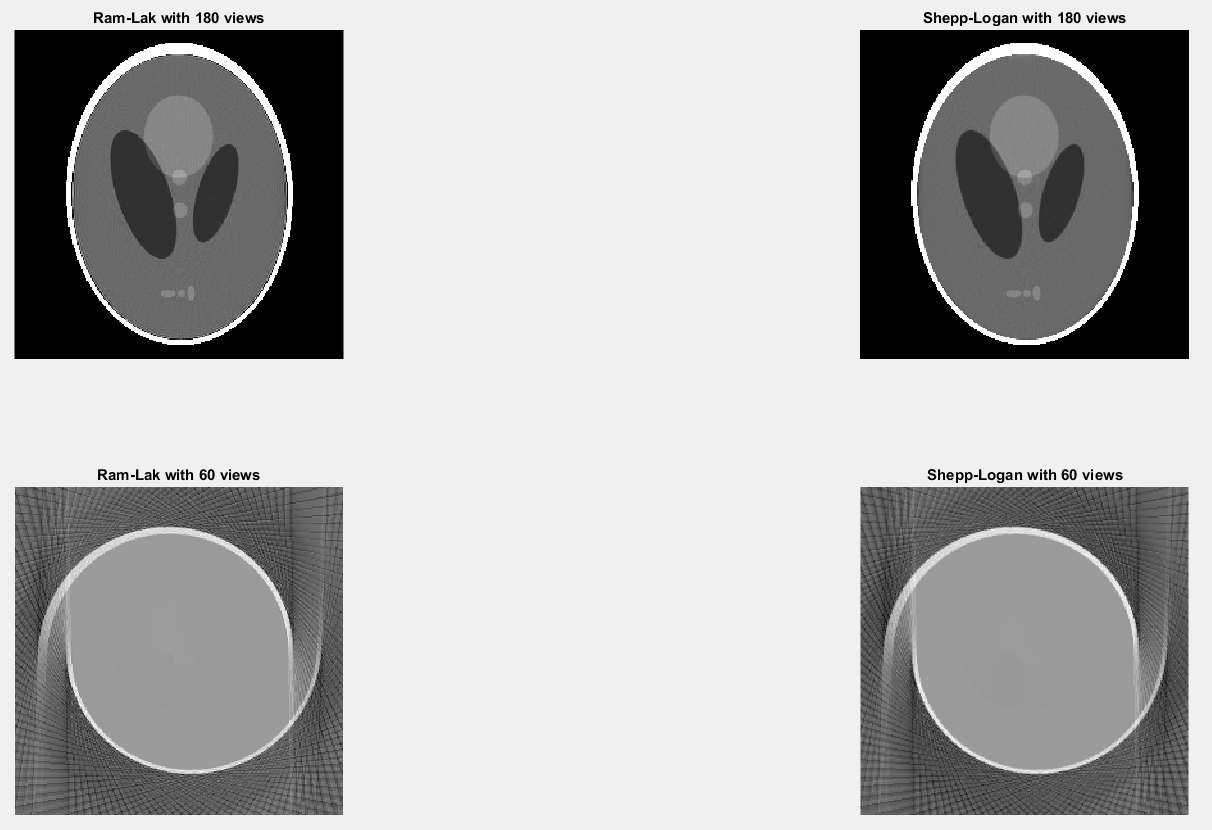
**Figure 1.2.** The back projection image after filtering by Ram-Lak and Shepp-Logan

Figure 1.3 compares the cut view of back projection without filtering and with filtering. The cut view without filtering implies an over-amplification of the high-frequency portion increasing intensity value outside the ellipse boundary which causes blurred effect. On the other hand, using filters, it can be seen that the region is separated by the original object value and a few splashing intensities near the object are considered ringing artifacts. The cut view of Shepp-Logan shows that its object boundary is more rapid and the ringing artifact is less than the cut view of Ram-Lak



**Figure 1.3.** Cut view of three back projection images

With smaller number of view (60 views), some frequency components are lost. Hence, the boundary of object is discontinuous and the detail in the center is invisible.



**Figure 1.4.** The filtered back projection with 180 views and 60 views

1. **Huffman coding**
   1. ***Theoretical background & Programming strategy***

Huffman coding is a lossless variable-length coding algorithm, with average codeword length satisfying Shannon’s noiseless coding theorem.

(2.1)

Where:

* is the source entropy of the probability vector
* is the average code length, also called the coding entropy
* is the probability of the corresponding symbol
* presents the length of Huffman codeword corresponding the symbol
* is the number of symbols

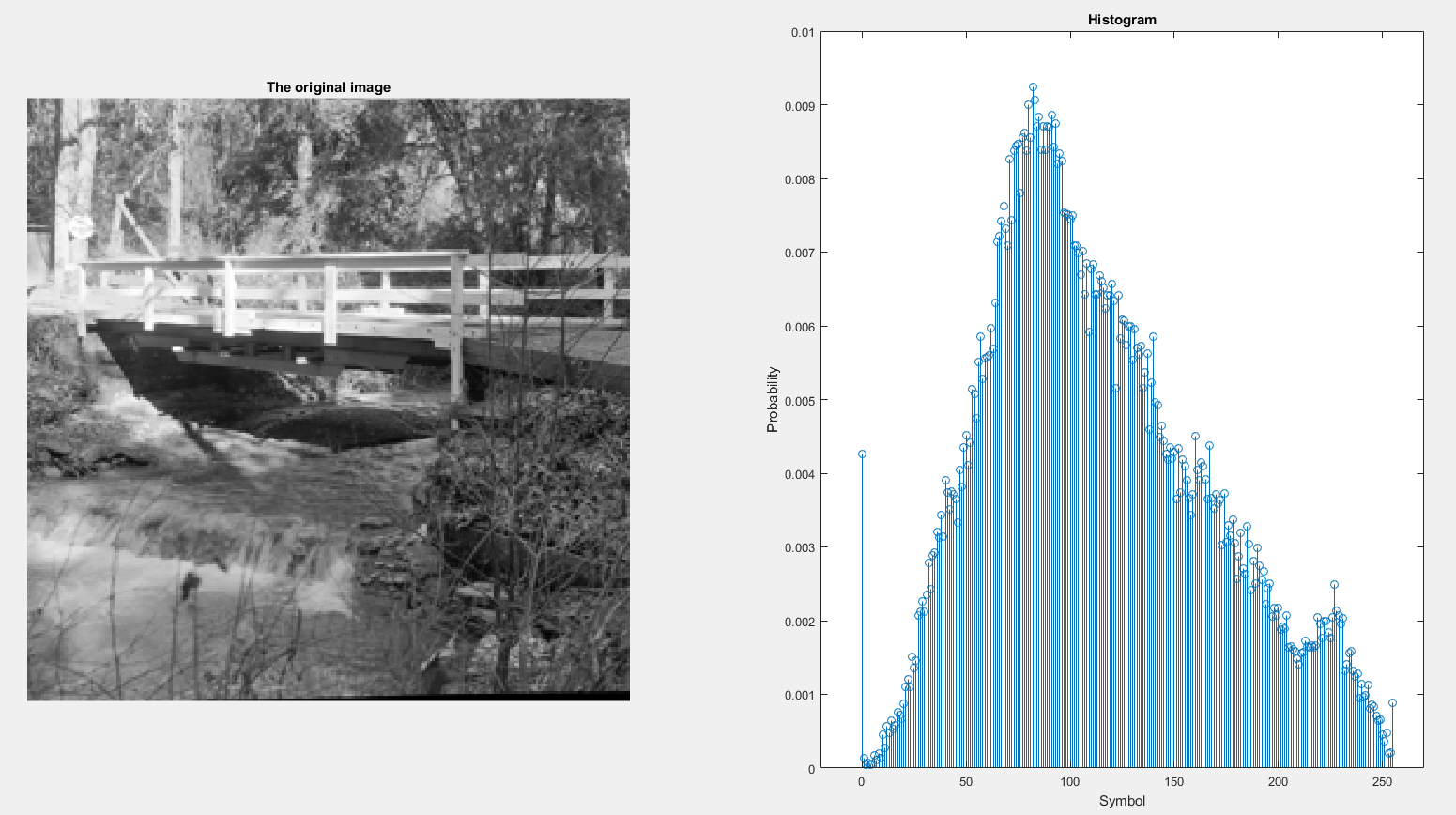
In this problem, symbols is understand as histogram of the original grayscale image, implying that there are 256 symbols from 0 to 255. Therefore, first, histogram of the original image is computed to get the probability array. Next, create cell array corresponding with 256 grayscale values. Then, applying Huffman code as below until the length of cell array not greater than 2.

* Arrange the probability array in ascending order
* Rearrange the cell array following probability
* Merge two first cells into the second cell with two smallest probabilities
* Remove the first cell and sum two smallest probabilities

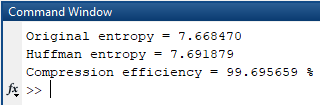
Finally, trace back to the location of each symbol and assign 0 and 1 to each pair of merging cells to take codeword for each grayscale.

The efficiency of the encoding scheme is defined as

* 1. ***Results & Analysis results***



**Figure 2.1.** The original image and its histogram



**Figure 2.2.** The entropies and compression efficiency

The codewords is written to huffman\_code.txt file. The experimental entropies describe similar (2.1) condition (7.67 < 7.69 < 7.67 + 1). Because the histogram of the original image is fluctuated, including almost grayscales from 0 to 255, the performance of Huffman encoder in this circumstance is very high, 99.7 percent.

1. **Transform coding and quantization**
   1. ***Theoretical background & Programming strategy***



**Figure 3.1.** A transform coding system: (a) encoder and (b) decoder

Transformation decorrelates the pixels of each sub-image or packs as much information as possible into the fewest number of transform coefficients. 8×8-block DCT is chosen because DCT has near-optimal performance for first-order Markov sequences with , therefore, DCT provides a good compromise between information packing ability and computational complexity. Moreover, 8×8-block DCT minimizes the Mean Square Error (MSE) and the required memory size because 8 is an integer power of 2 and computation load with 8×8 block is acceptable. The other advantages of DCT is reducing the blocking artifact by periodic mirroring.

1-D DCT from DFT by using symmetric property:

Because image is 2 dimension and DCT is symmetric. So, we can use formula:

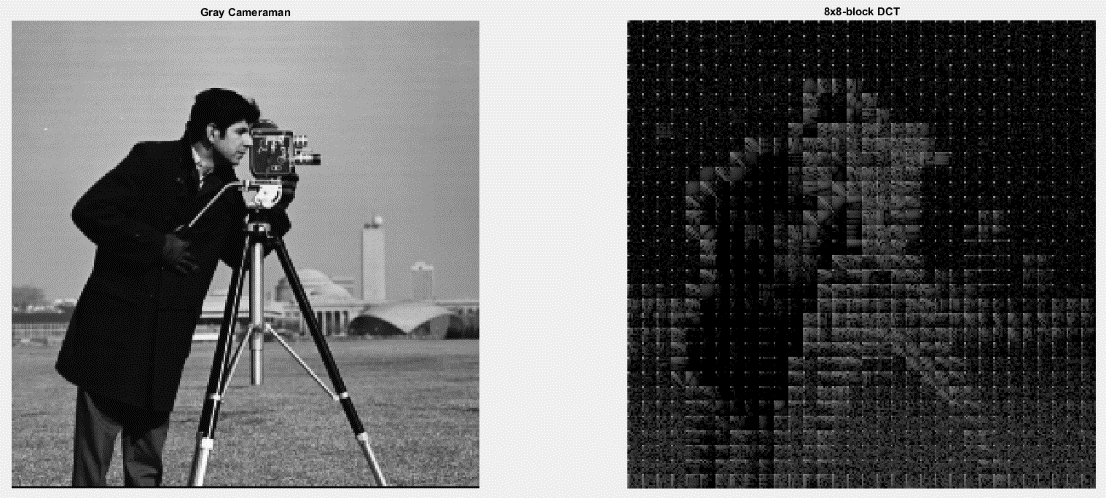
Where

Quantization selectively eliminates or more coarsely quantizes the coefficients that have the smallest impact on reconstructed sub-image quality.

Where is an element of transform normalization array (quantization parameter).

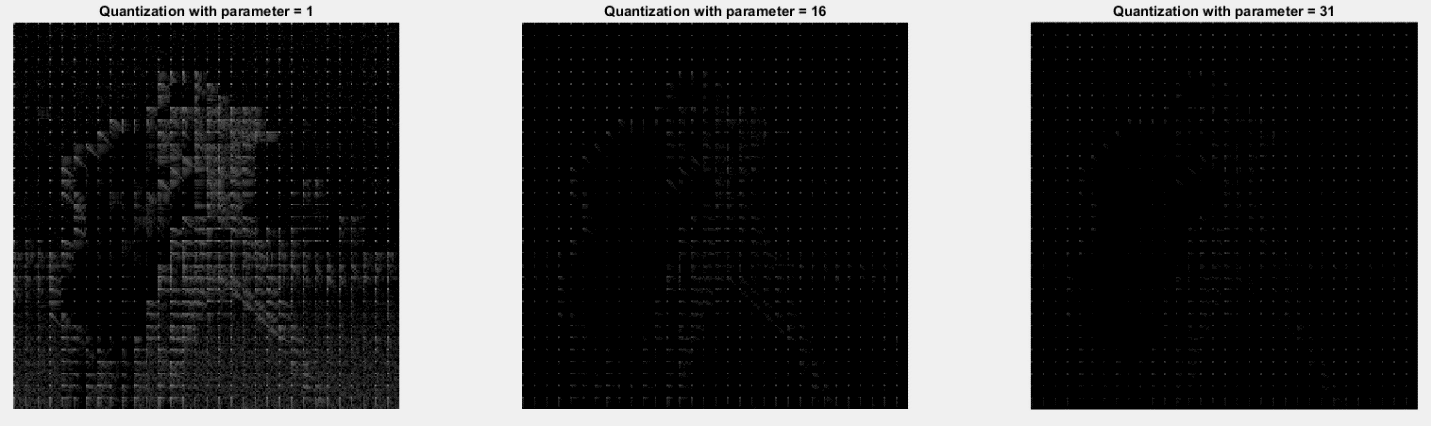
* 1. ***Results & Analysis results***

After using 8×8-block DCT for the original image, most of the low frequency components locate at the top left corner of each 8×8 block, meaning the unwanted frequency components are eliminated (see Figure 3.2).



**Figure 3.2.** The original image and its 8×8-block DCT

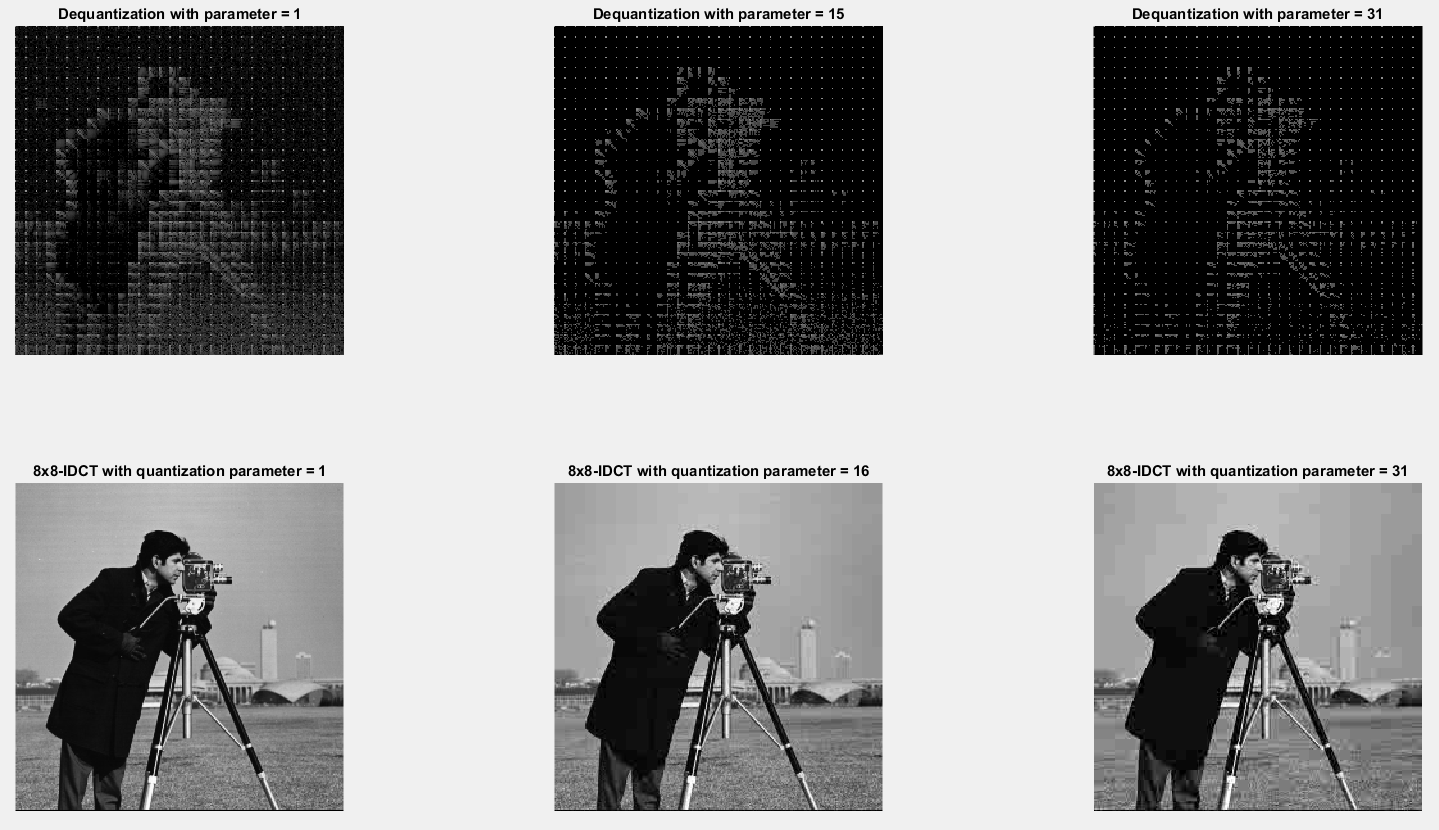
When quantize DCT image with higher quantization parameter, more and more the number of higher frequency components are removed. With the smallest parameter equals to 1, most of low frequency parts remain but the magnitudes are changed. For parameter equals to 16 and 32, only very low frequency terms are keep and quantization images look almost dark (see Figure 3.3).



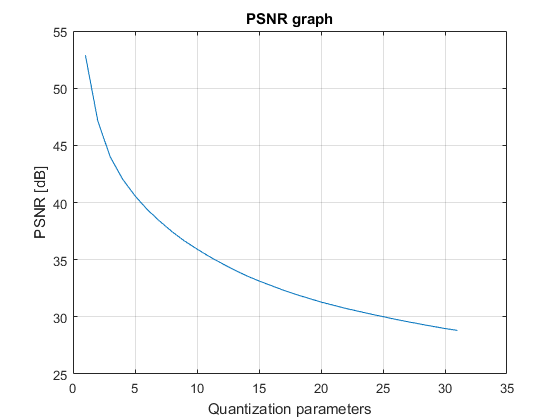
**Figure 3.3.** The quantization results for three parameter 1, 16 and 31

Obviously, after dequantization, for greater parameters, fewer frequency components are recovered implying that more amount of information is lost also called quantization error. Hence, with greater quantization parameters, by inverse DCT, the comparable reconstruction images are more different and more blurred. For parameter equals to 1, the recovered image look similar the original image. However, for parameters equal to 16 and 31, in Figure 3.4, the background of output images is not smooth and shows us the discontinuation due to increase quantization error.

As mention above, the raising of the quantization parameter comes along with the increasing of quantization error. Therefore, PSNR reduces while quantization parameter increases (see Figure 3.5).



**Figure 3.4.** The dequantization with difference parmeters and the corresponding inverse 8×8-block DCT images



**Figure 3.5.** PSNR graph