Image and Video Processing
Lab 1: Getting Started with Matlab, Quantization, Sampling, Filtering, 2DFT, Weber Law

Yuliang Zheng, Valentine Santarelli

October 2016

1 Image and Color Tables

1.1 Read and Show the two images

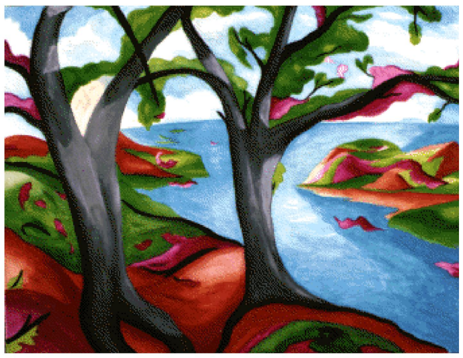

Figure 1. trees.tif in indexed format

Figure 2. lena.tif in truecolor format

Figure 1 shows the image of *trees.tif* with indexed format, while Figure 2 is the image of *lena.tif* with true color format.

Gray level and negative images

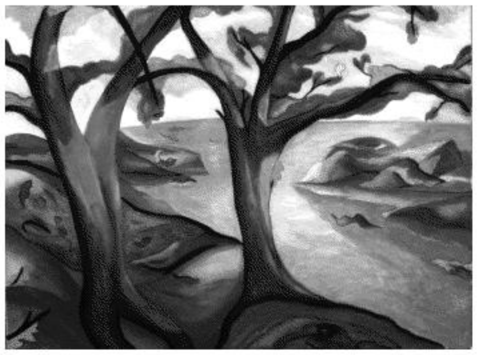

Figure 3. trees.tif in gray level

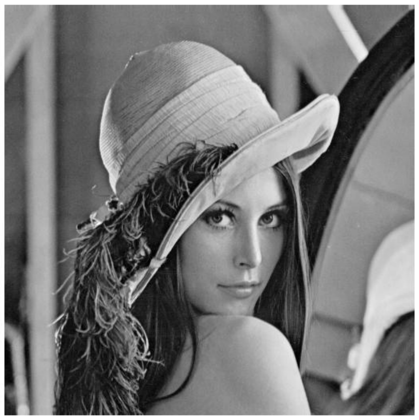

Figure 4. lena.tif in gray level

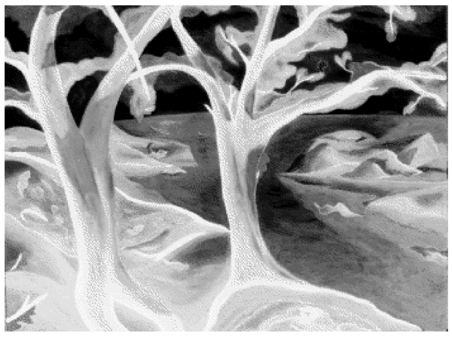

Figure 5. trees.tif in negative

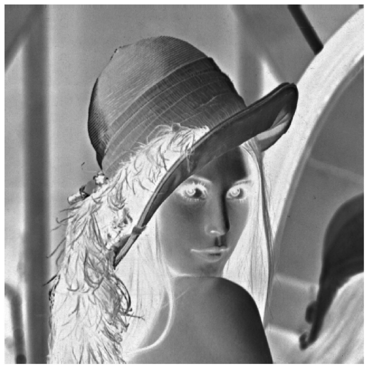

Figure 6. lena.tif in negative

Figures 3 and 4 give the *tree.tif* and *lena.tif* in gray level. To show the negative of the two images, the results are in Figure 5 and Figure 6.

To get the gray level image of *trees.tif*, we used the Matlab function *ind2gray*; and to get its negative image, the gray level matrix was subtracted to a matrix of similar size with all elements put to 255. For *lena.tif*, as it is a in RGB format, Matlab function *rgb2gray* was used to get the gray level image, *i.e.* a 2D matrix. The same implementation was done to obtain the negative image of *lena.tif*.

1.2 Images with different gamma corrections

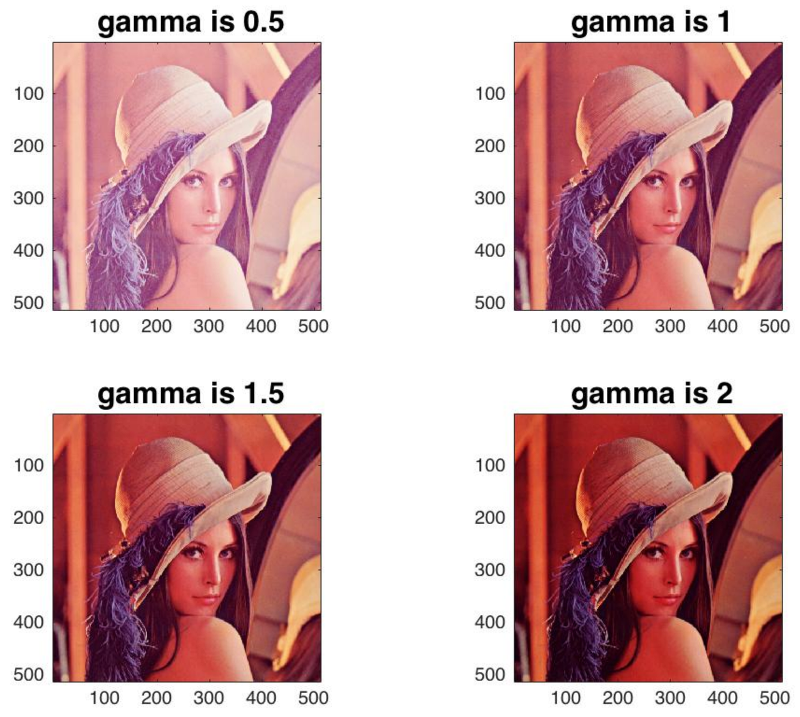

Figure 7: lena.tif with different gamma corrections

With $\gamma > 1$, the colors of the corrected image seem darker, respectively, with $\gamma < 1$ the color seems brighter. The gamma correction seems to control the brightness of the image as we can observe on the Figures 7 and 8.

To implement the γ correction for the indexed images, the elements of the color map matrix were modified as follow: $a_{ij} = a_{ij}^{\gamma}$ using $A = A.^{\gamma}$ in Matlab.

To implement the γ correction for the *truecolor* format images, each RGB value is normalized, *i.e.* divided by 255, before applying the γ correction : $r = r^{\gamma}$, $g = g^{\gamma}$, $b = b^{\gamma}$.

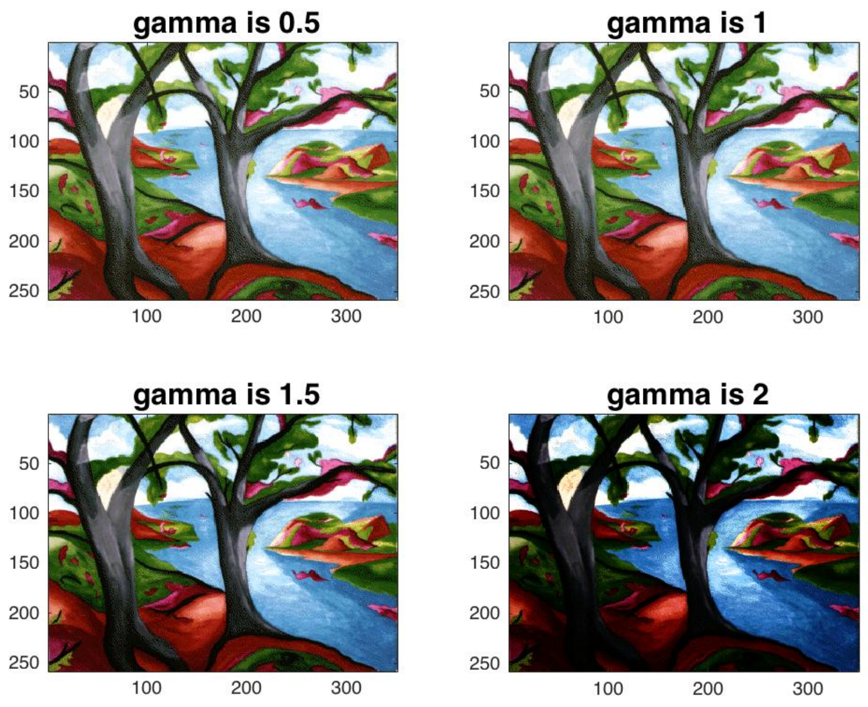

Figure 8: trees.tif with different gamma corrections

1.3 Chessboard

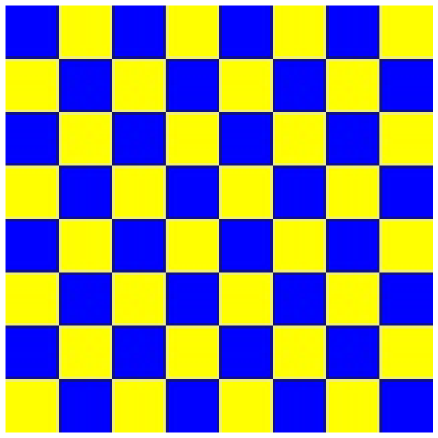

Figure 9: Chess board

For the *truecolor* format, the chess board was created as follow:

- Creation of a blue and a yellow square as a 3D matrix of size $20 \times 20 \times 3$ with the RGB values equal to blue=(0,0,255) and respectively yellow=(255,255,0);
- Creation of a block of four squares with 2 blue (and respectively 2 yellow) squares in diagonal using block matrix;
- Repetition of the blocks using the *repmat* function to create the chess board.

For the *indexed* format, the chess board was created as follow:

- Creation of a color map matrix of size 2×3, *i.e.*, two indexed colors with their normalized RGB values blue=(0,0,1) and yellow=(1,1,0);
- Creation of a blue and a yellow square as a 2D matrix of size 20×20 with the index of the corresponding color in the color map *i.e.* blue's index=1 and yellow's index=2.
- Creation of a block of four squares with 2 blue (and respectively 2 yellow) squares in diagonal using block matrix;
- Repetition of the bloc using the *repmat* function to create the chess board.

2 Image Quantization

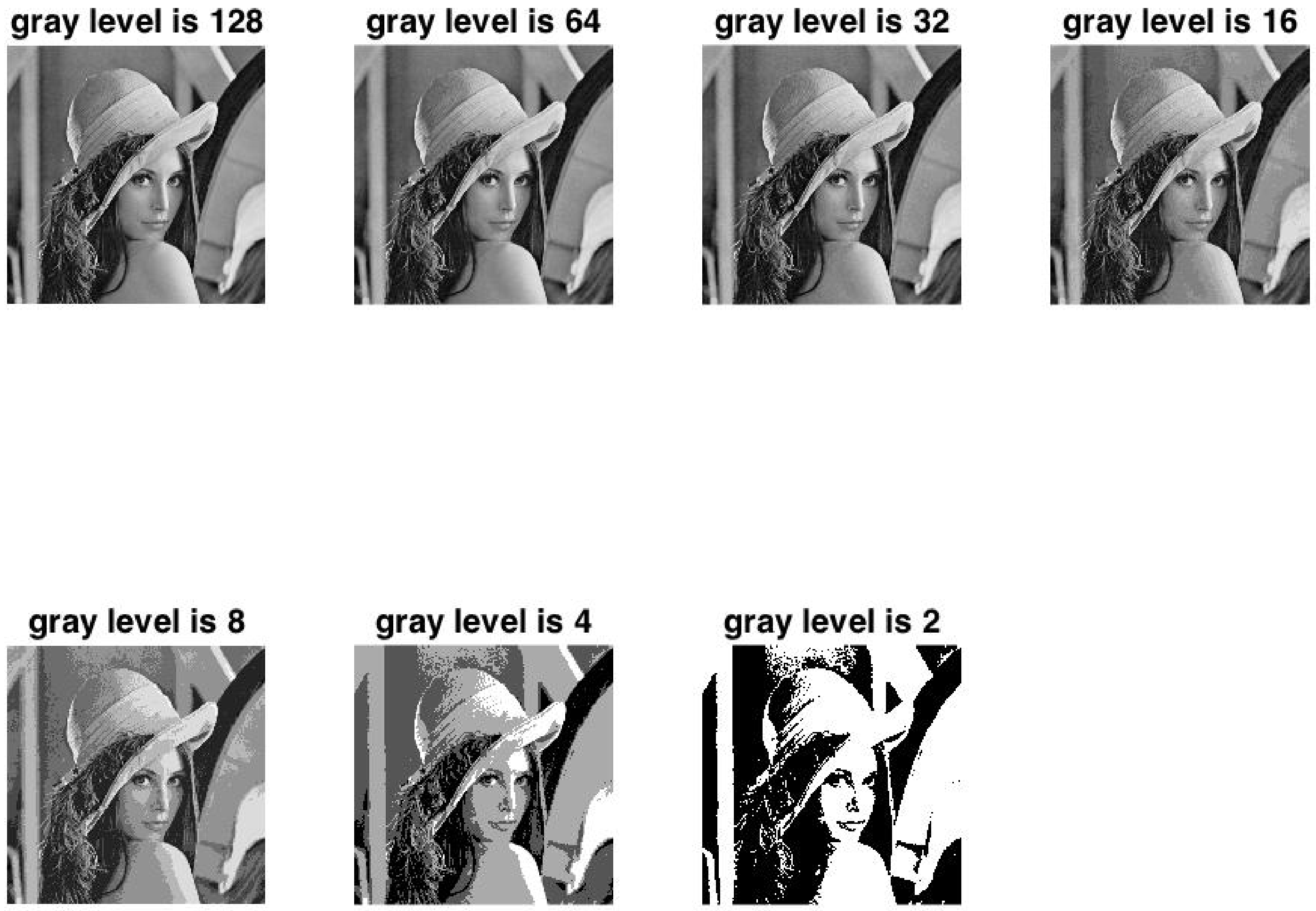

Figure 10: Quantization in different steps

Figure 10 offers the uniform quantization result while taking different steps, namely 128, 64, 32, 16, 8, 4 and 2 gray levels. And when the gray level is 16, we can see the "false contour".

The quantization is implemented by taking each element of the gray level image and applying the following operation:

$a_{ij} = stepsize \times \lfloor \frac{a_{ij}}{stepsize} \rfloor$

3 Filtering

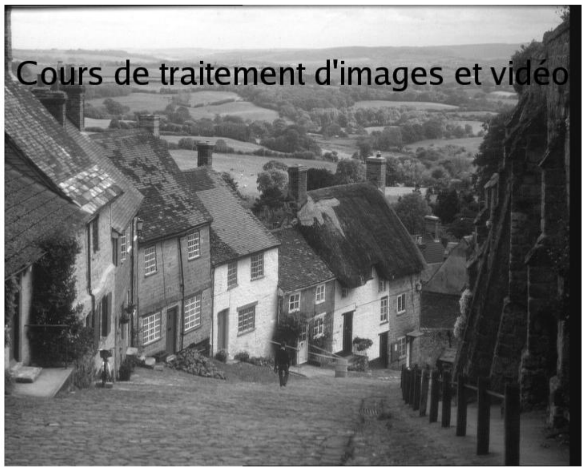

Figure 13: gold-text.png orginal

Figure 13 gives how the original *gold-text.png* looks like, while Figure 14 shows the blurred image after applying the 5×5 filter and Figure 15 shows the resulting image of the 3×3 filter applied to the blurred image.

The 3×3 filter seems to add a blur noise to the original image whereas the 5×5 filter seems to retrieve the original image but with all little bit of noise left.

$$\text{Frequency response of the } 5 \times 5 \text{ filter} = \begin{pmatrix} 0.0061 & 0.0421 & 0.0784 & 0.0421 & 0.0061 \\ 0.0421 & 0.2891 & 0.5376 & 0.2891 & 0.0421 \\ 0.0784 & 0.5376 & 1.0000 & 0.5376 & 0.0784 \\ 0.0421 & 0.2891 & 0.5376 & 0.2891 & 0.0421 \\ 0.0061 & 0.0421 & 0.0784 & 0.0421 & 0.0061 \end{pmatrix}$$

$$\text{Frequency response of the } 3 \times 3 \text{ filter} = \begin{pmatrix} 6.0544 & 5.1667 & 4.6180 & 5.1667 & 6.0544 \\ 5.1667 & 3.4456 & 2.3820 & 3.4456 & 5.1667 \\ 4.6180 & 2.3820 & 1.0000 & 2.3820 & 4.6180 \\ 5.1667 & 3.4456 & 2.3820 & 3.4456 & 5.1667 \\ 6.0544 & 5.1667 & 4.6180 & 5.1667 & 6.0544 \end{pmatrix}$$

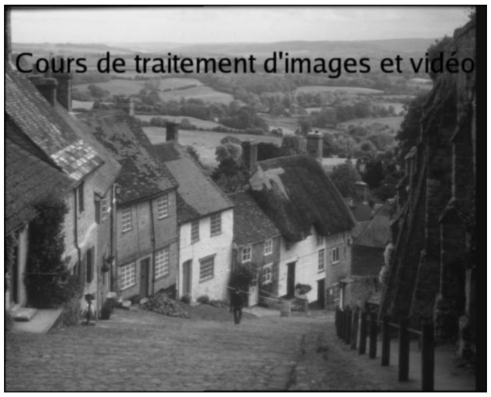

Figure 14. gold-text.png after the 5 × 5 filter

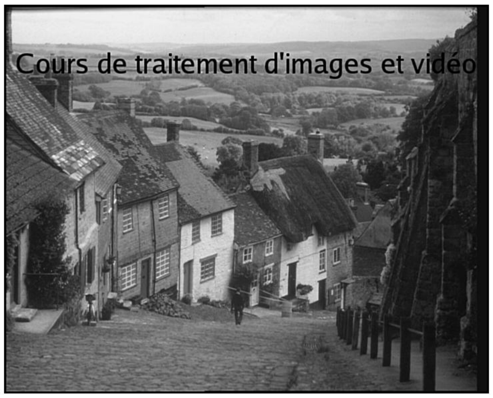

Figure 15. Blurred image after the 3 × 3 filter

And Figure 16 and Figure 17 show the frequency response (only the magnitude) of the two filters. The frequency response of filter 5 × 5 is more like a hump, while that of the filter 3 × 3 is also a hump but in an inverted (upside-down) version.

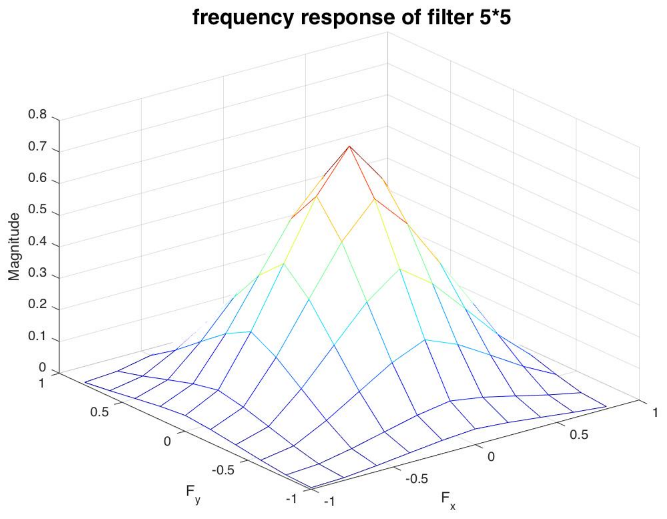

Figure 16. Frequency Response of the 5 × 5 filter

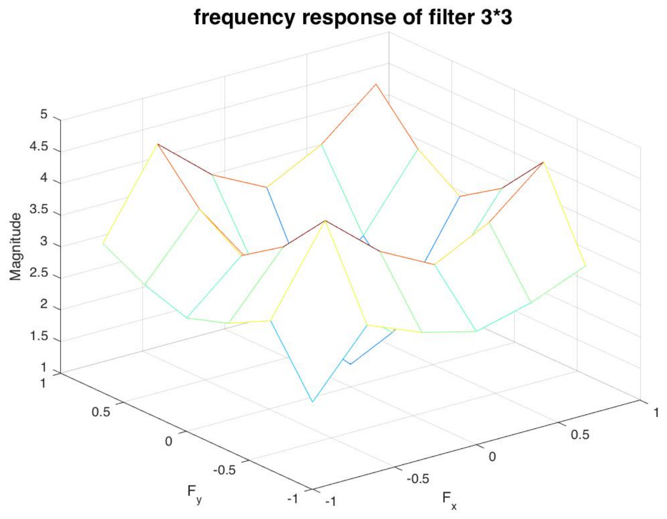

Figure 17. Frequency Response of the 3 × 3 filter

4 Correlation

The position of the letter g corresponds to the coordinates (in the initial image) which correspond to the maximal element's coordinates in the resulting matrix of the correlation of the flipped image of the letter g and the image with text.

To find the position of the g letter in the image with the correlation in the spatial domain, we proceed as follow:

- Horizontal and vertical flip of the image of the g letter, *i.e.* a rotation of 180°.

- Centralization of the color values of the images by subtracting 128 to all the elements of the corresponding 2D matrices.
- Implementation of the correlation function as the convolution of the image with text with the rotated image of the g letter.
- Finding the maximal element's indexes in the correlation's resulting matrix .
- Finding the central point of letter g in the text's image by taking into account the translation introduced by the calculation of the correlation as follow:

  (x, y)= coordinates in the text's image
  (x', y')= coordinates in the correlation's resulting image

  $x = x' - \frac{width-of-the-g-letter-image}{2}$
  $y = y' - \frac{height-of-the-g-letter-image}{2}$

To find the position of the g letter in the image with the correlation in the frequency domain, we proceed as follow:

- Horizontal and vertical flip of the image of the g letter, *i.e.* a rotation of 180°.
- Centralization of the color values of the images by subtracting 128 to all the elements of the corresponding 2D matrices.
- Applying the 2D Fourrier transform on the images to switch to the frequency domain
- Implement the correlation function as the convolution of the image with text with the rotated image of the g letter.
- Applying the 2D inverse Fourrier transform on the resulting correlation matrix to come back to the spatial domain
- Finding the maximal element's indexes in the correlation's resulting matrix .
- Finding the central point of letter g in the text's image by taking into account the translation introduced by the calculation of the correlation as follow:

  (x, y)= coordinates in the text's image
  (x', y')= coordinates in the correlation's resulting image

  $x = x' - \frac{width-of-the-g-letter-image}{2}$
  $y = y' - \frac{height-of-the-g-letter-image}{2}$

  g letter coordinates in the text image = (493,92)

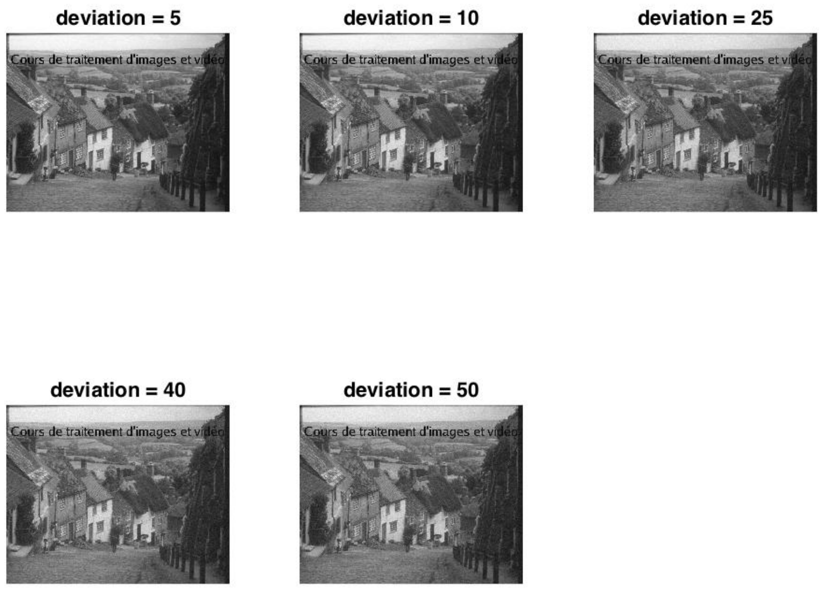

Figure 18: *gold-text.png* with different noise

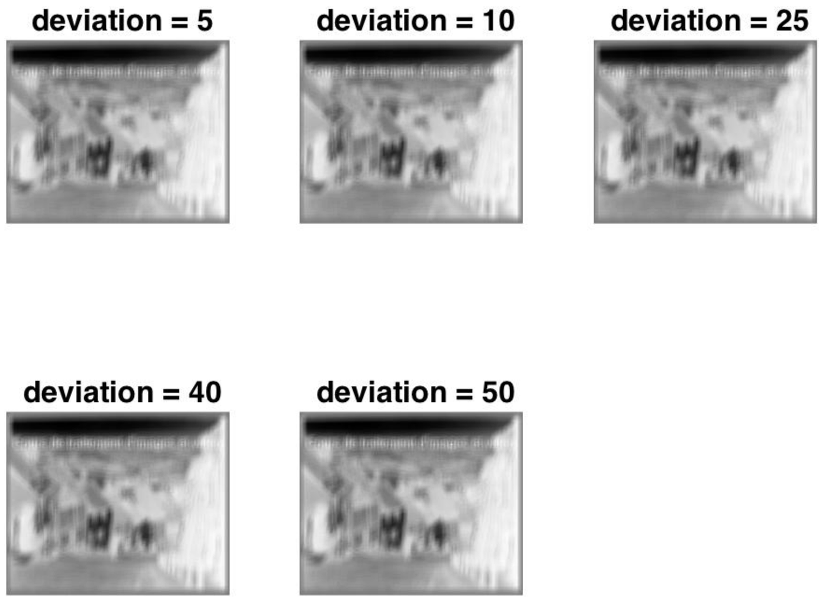

Figure 19: Correlation with noise in spatial domain

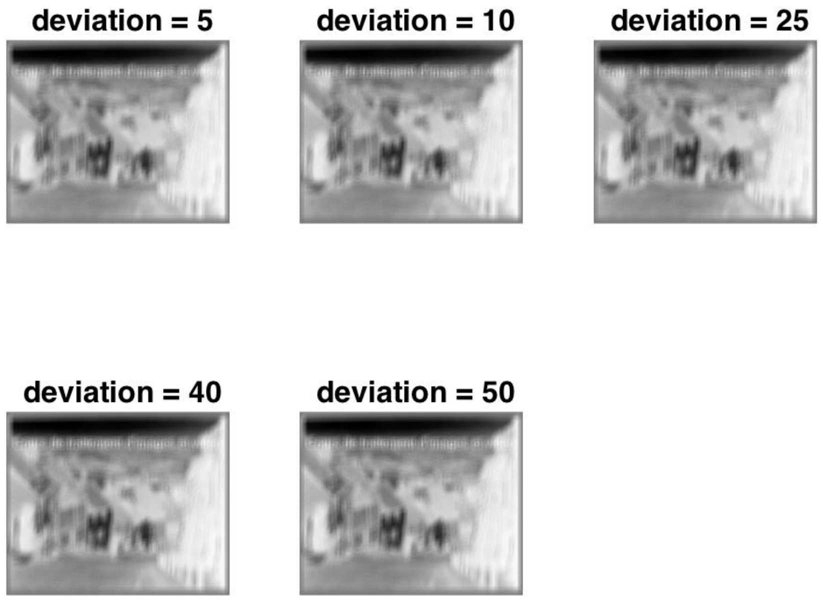

Figure 20: Correlation with noise in frequency domain

The noise was implemented by adding to the each element of the text image 2D matrix the value : *sqrt(V(i))*randn(size(X5))*.

The resulting images of the correlation matrices are very similar. We can conclude that the correlation is less sensitive to this type of noise *i.e.* a normal distribution with different standard deviations.

5 Resampling

Figure 21 shows the original *sub4.tif* without doing any down sampling, while Figure 22 and Figure 23 give the two down-sampled images with factor 2 and 4. As the principle of the down sample is to take one pixel out of n in every direction, while the sampling factor is n, the image after down sample must be blurred and its size must decrease.

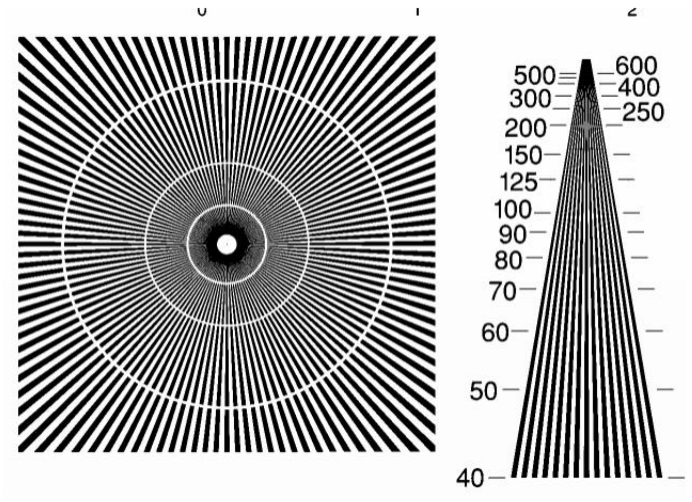

Figure 21: sub4.tif original

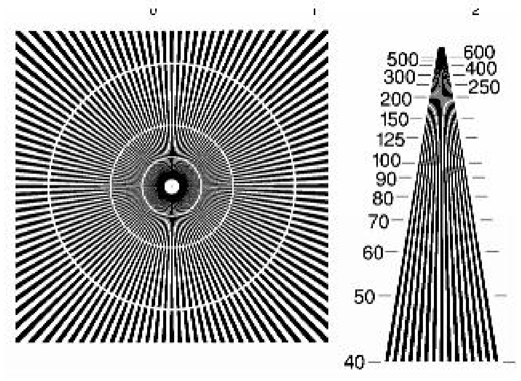

Figure 22. sub4.tif Down-sampling with factor 2

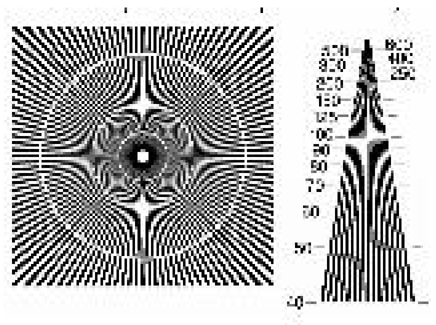

Figure 23. sub4.tif Down-sampling with factor 4

6 Phase and magnitude of the 2DFT

Figure 24 shows the original *lena-y.png*. By doing 2DFT to the original image, and deleting either its imaginary part or real part, after inverse 2DFT, we got Figure 25 and Figure 26. When deleting the real part after 2DFT, the inverse 2DFT figure seems to be mirror symmetric along the diagonal line. When deleting the imaginary part, still a mirror symmetry along the diagonal line is exhibited, but one mirrored ”lena” seems to be a negative of another one.

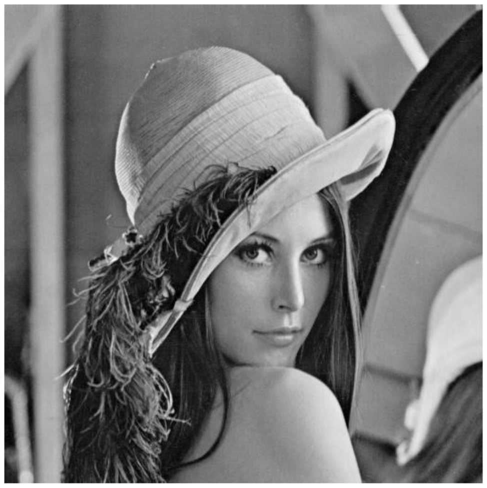

Figure 24: lena-y.png original

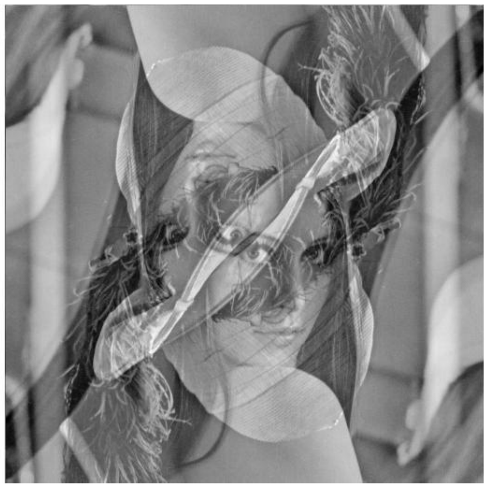

Figure 25. lena-y.png deleting the imaginary part of the FFT

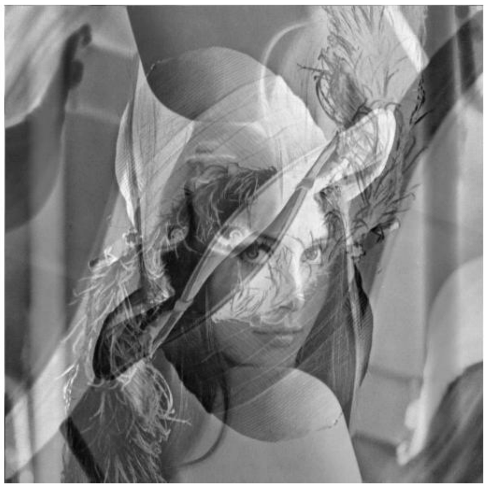

Figure 26. lena-y.png deleting the real part of the FFT

Those observations can be explained as follow.

By deleting the imaginary part, only the real part of the FFT remains, the inverse FFT on the latter gives:

$$\mathfrak{F}^{-1}(\Re(X(m,n))) = \mathfrak{F}^{-1}(\frac{1}{2}X(m,n) + \frac{1}{2}X^*(m,n))$$

$$\mathfrak{F}^{-1}(\Re(X(m,n))) = \frac{1}{2}\mathfrak{F}^{-1}(X(m,n)) + \frac{1}{2}\mathfrak{F}^{-1}(X^*(m,n))$$

$$\mathfrak{F}^{-1}(\Re(X(m,n))) = \frac{1}{2}x(k,l) + \frac{1}{2}x(-k,-l)$$

$\frac{1}{2}x(-k,-l)$ represents the "mirrored" original *lena-y.png* flipped horizontally and vertically.

By deleting the real part, only the imaginary part of the FFT remains, the inverse FFT on the latter gives:

$$\mathfrak{F}^{-1}(j\Im(X(m,n))) = \mathfrak{F}^{-1}(j\frac{1}{2j}X(m,n) - j\frac{1}{2j}X^*(m,n))$$

$$\mathfrak{F}^{-1}(j\Im(X(m,n))) = \frac{1}{2}\mathfrak{F}^{-1}(X(m,n)) - \frac{1}{2}\mathfrak{F}^{-1}(X^*(m,n))$$

$$\mathfrak{F}^{-1}(j\Im(X(m,n))) = \frac{1}{2}x(k,l) - \frac{1}{2}x(-k,-l)$$

$-\frac{1}{2}x(-k,-l)$ represents the "mirrored" negative *lena-y.png* flipped horizontally and vertically.

After calculating the 2DFT of the image *lena-y.png*, setting the phase of the 2DFT to be zero, we got Figure 27 after inverse 2DFT. Setting the phase to be zero means to only maintain the magnitude information, and Figure 27 is in fact the logarithm of the intensity to highlight the variations.

Repeat the operation by setting the magnitude to 1 while keeping the phase as it is this time, Figure 28 shows the result, which gives the phase information of the image. From the figure, it is easy to understand that the phase of 2DFT contains the information about edges and contours of an image. It's also intuitive since the phase part contains the frequency variable which is connected to the intensity of color's variation. Since the contour can be defined by a strong variation of the color, it explains why the contours are highlighted in the Figure 28 illustrated the phase part.

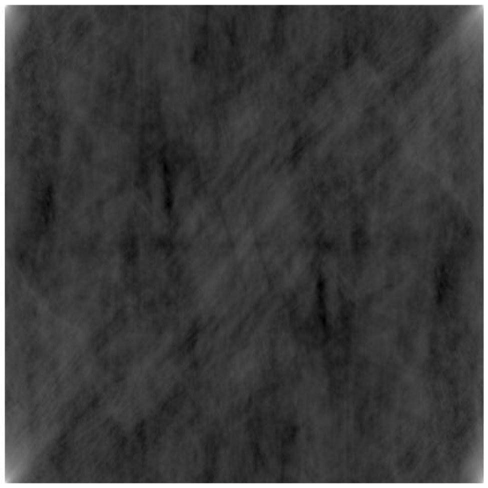
Figure 27. lena-y.png setting the phase of the 2DFT to zero

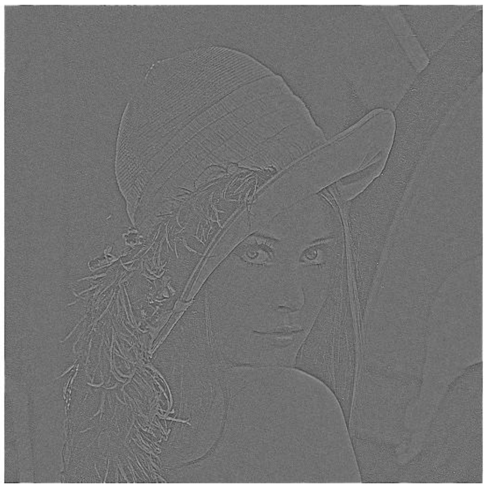
Figure 28. lena-y.png setting the magnitude to 1

7 Weber Law

Create function *weber*

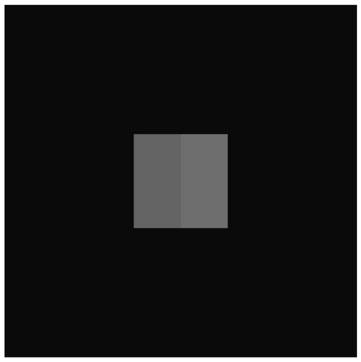

Figure 29: One experiment: $Lb = 10$, $L1 = 100$, $L2 = 110$

The 2D matrix for the Weber experimentation is implemented by creating 4 matrices of 3 sizes and 3 colors and combining them using a block matrix:

L1 and L2 matrices are of size 160x80 with the corresponding color L1 and L2; L2 matrix = 160x80 of color L2;

The up and down part of the background are of same size 220x600 with the color Lb.

The left and right part of the background are of the same size 160x220 with the color Lb.

Weber matrix $= \begin{pmatrix} & L_{b,up} & & \\ L_{b,left} & L_1 & L_2 & L_{b,right} \\ & L_{b,down} & & \end{pmatrix}$

Determination of the Weber constant for $L_b = 10$ and $L_b = 200$

As Figure 29 shows one test image for Weber constant determination we created, by comparing the result of different L2 while fixing L1 in a specific number, we selected the image where we can distinguish the contour of the square (L1 and L2) by human eyes. By doing so with a few values of L1, and also changing Lb with two values ($Lb = 10$, and $Lb = 200$), we plotted the curve lines as Figure 30.

In Figure 30, the red line gives the Weber constant for $Lb = 10$, while the green one is for $Lb = 200$. While for $Lb = 10$ which corresponds to a more black background, $Lb = 200$ refers to a more white background, their Weber constants roughly are the same at a specific L1. So, the two Weber constant curve lines have similar shape. When L1 is small, Weber constant is large, as L1 increases, Weber constant declines rapidly. However, after a certain value of L1, the Weber constant becomes stable and gradually approaches zero.

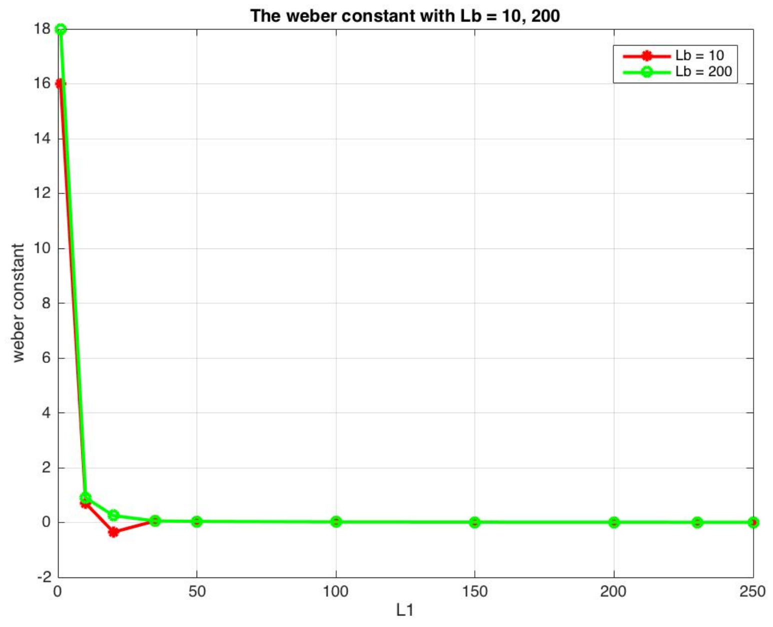

Figure 30: Weber constant for $Lb = 10, 200$