Image and Video Processing
Lab 2: Dithering

Yuliang Zheng, Valentine Santarelli

October 2016

Fixed threshold method

Figure 1 and Figure 2 show the two original images. To implement the *Fixed Threshold Method*, we compared each single pixel's value, and set the threshold to be 128(if it is smaller than this threshold, we set it as 0, otherwise we set it at 255). As a result, in Figure 3 and Figure 4, we only got the images with only black or white colors, which are of a poor quality.

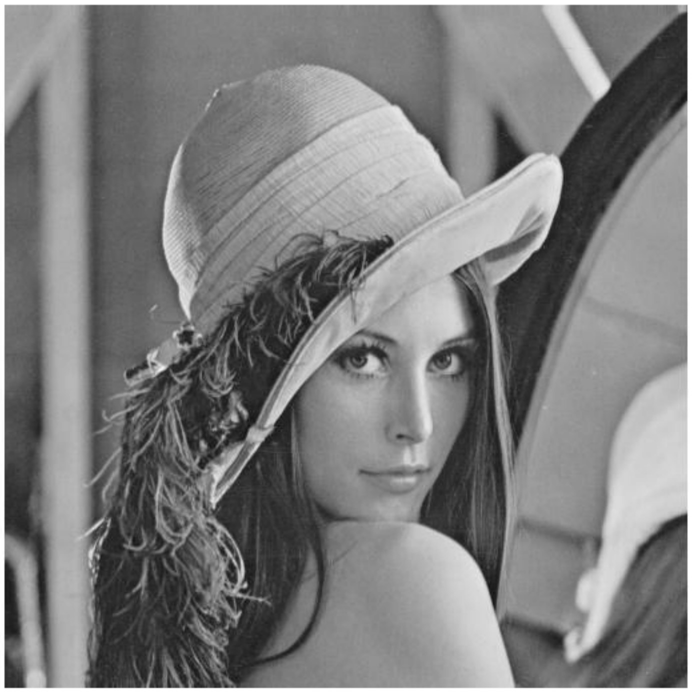
Figure 1: lena-y.png Original

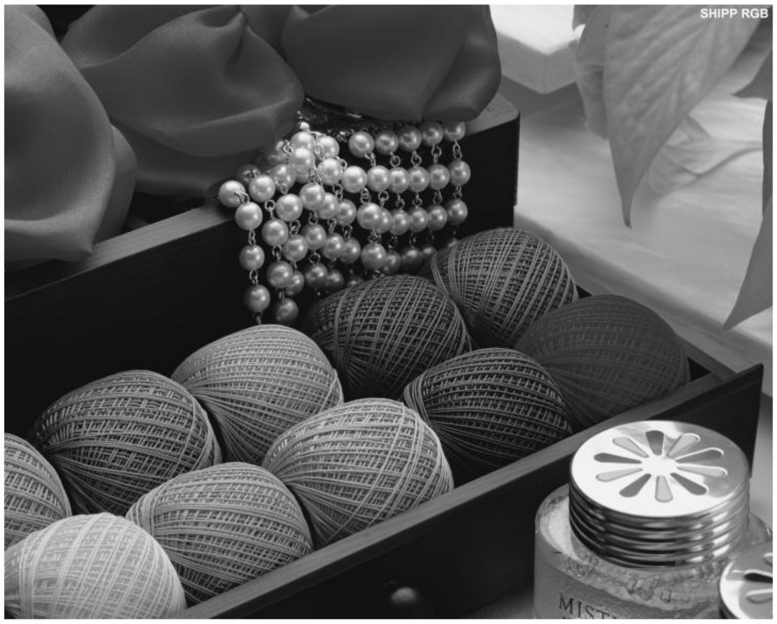

Figure 2: wool.png Original

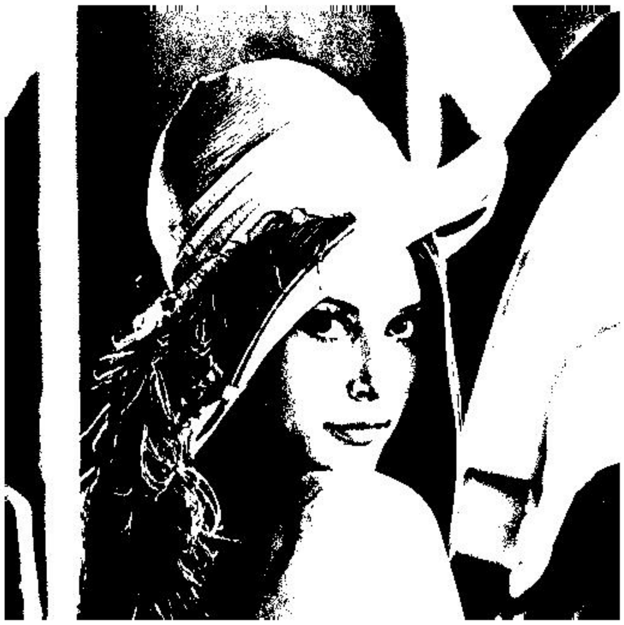

Figure 3: lena-y.png by Fixed Threshold Method

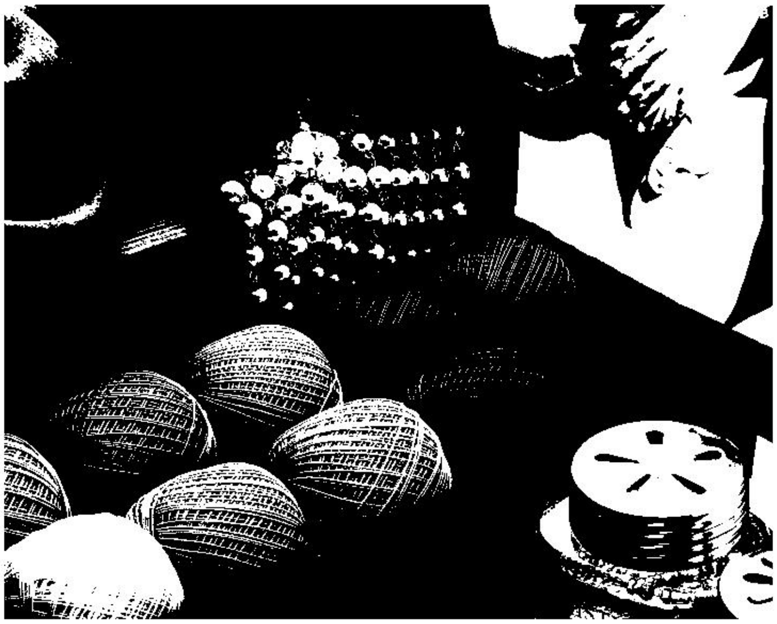

Figure 4: wool.png by Fixed Threshold Method

Visual quality

Entire regions of the image are either black or white, resulting in the disappearance of many details because the different gray levels couldn't be reproduce with this method. In conclusion,this method doesn't give a good rendering of the image because we lose a lot of information from the original image.

Mean square error between the dithered image and the original

MSE for lena: 0.1303
MSE for wool: 0.0949

Random threshold method

First of all, adding noises to one image will induce more gray levels rendering, on the other hand it will definitely blur the contours. Here, we added noises with amplitude A=10, 30, 60, 90, 120 respectively to the images, and then centered the noise by subtracting half of its amplitude. If the noise isn't centered the image tends to be whiter because we only had positive value to the original pixel value (so it tends towards 255 *i.e.* the white color).

$$Im_{noisy} = Im_{original} + unidrnd(noise_{amplitude}, size(Im_{original})) - \frac{noise_{amplitude}}{2};$$

After that we implemented the *Fixed Threshold Method* on the image.

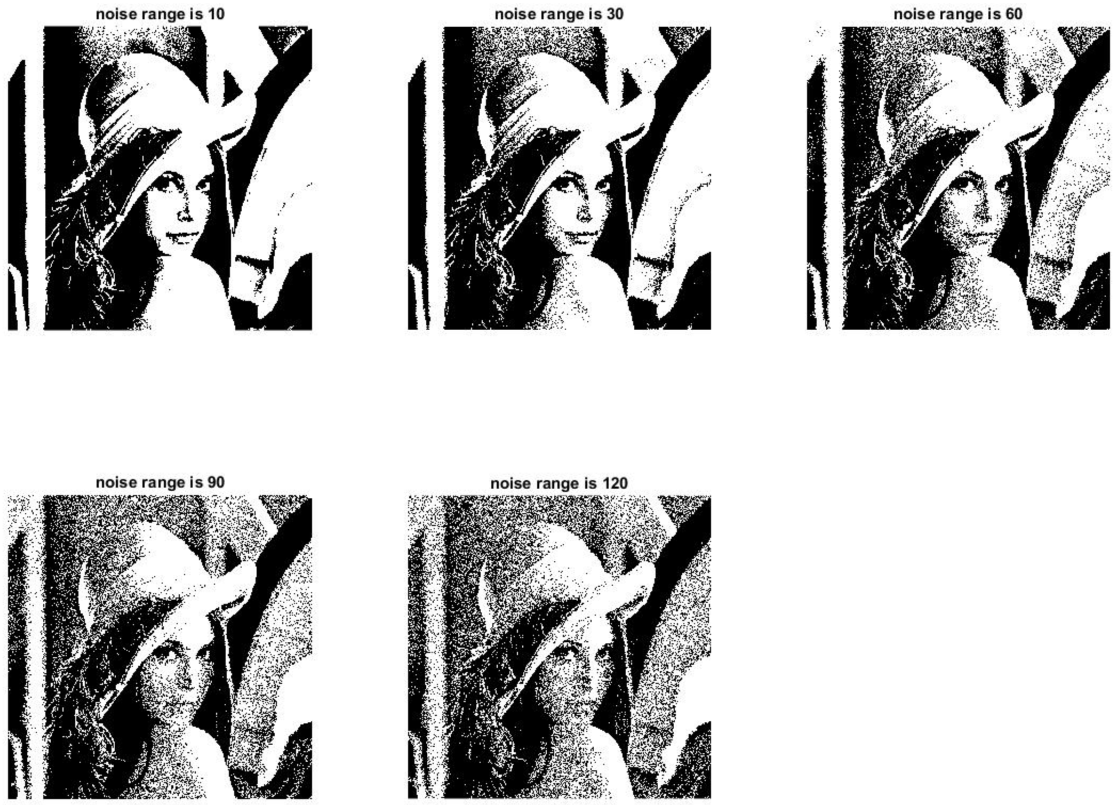

Figure 5: lena-y.png by Random Threshold Method

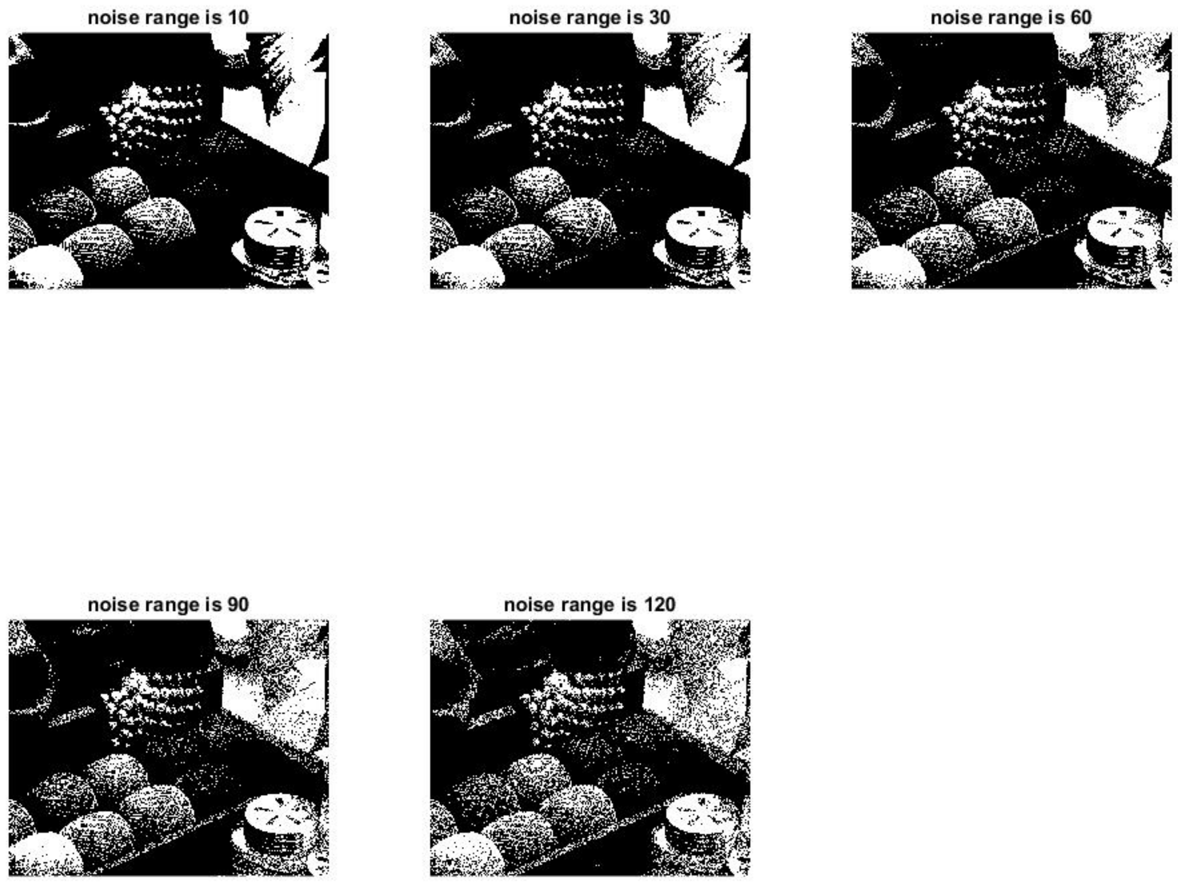

Figure 6: wool.png by Random Threshold Method

Visual quality

The noise level for which there is a good interpretation of gray levels and not too much deterioration of the image is : 30 for *lena.png* and 60 for *wool.png* (we can see the detail of the wool ball). But with a higher level of noise range (e.g. 120), the result is unsatisfactory (see in Figure 5 and Figure 6). Less information are missing compared to the result of the fixed threshold method, but it's still not a good rendering.

Mean square error between the dithered image and the original

Here, we used a vector to save the Mean Square Error between the dithered images in different noise ranges and the original image.

MSE for lena: [0.1305, 0.1324, 0.1388, 0.1488, 0.1596];

MSE for wool: [0.0951, 0.0964, 0.1005, 0.1072, 0.1149].

Ordered threshold method

Before implementing the *Ordered Threshold Method*, the original images must be quantized. As we are asked to write the quantization function Q(x) without operations such as *floor*, *ceil* or *round*. Here, we utilized the *find* function to fulfill the quantization, and the following is our logic:

We defined the corresponding Matlab Function as $X = quantization(Y, n)$, where Y refers to the input image, n as the gray levels we want for the resulting quantized image.

- $stepsize = 256/n$ (the quantization interval)
- for i^{th} gray level, find the pixels of range $[stepsize \times (i-1), stepsize \times (i))$, and set all these pixels to range $i - 1$; do the iteration for each gray level.

After the quantization of images, we need to implement the *Ordered Threshold Method* by the Matlab Function $X = ordered\_threshold(Y, s)$. By applying *repmat* to S, a new paving matrix of the same size as the input image is created. Then, comparing each component at the same position for two matrices (if the pixel in the input image is of higher value than the paving matrix, we set it as 255; otherwise we change it to 0), we will get the image after ordered threshold.

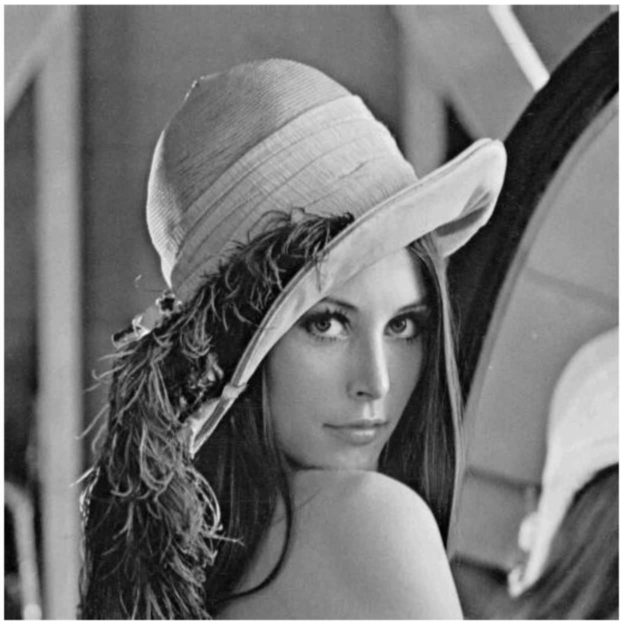

Figure 7: Quantized lena-y.png

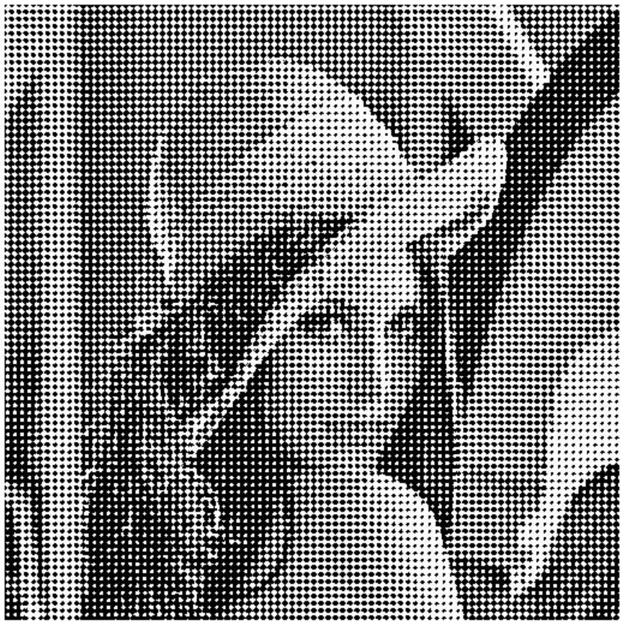
Figure 8: lena-y.png by Ordered Threshold Method

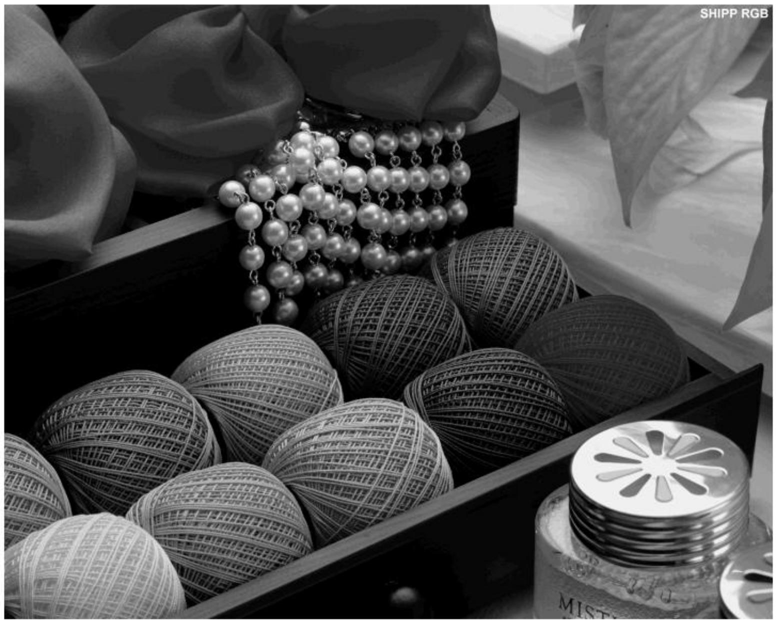

Figure 9: Quantized wool.png

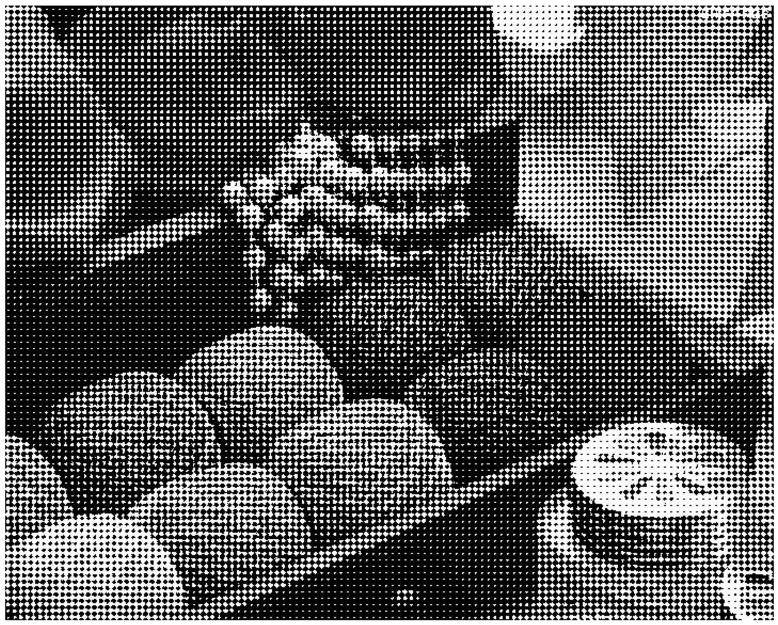

Figure 10: wool.png by Ordered Threshold Method

Visual quality

Each portion (of size of S) of the image is compared to the threshold matrix S. The S matrix is composed of low threshold levels at the center getting higher as it gets closer to the border of S. This results in a dot effect (like Roy Lichtenstein's art) on the image, reproducing the gray levels with the size of the dot (varying with the original pixel value), its color (black or white) and its background (respectively white or black). This proportion of black and white in the portion of image (of size of S) will render a gray level. The *dots* are too large to render all the details of the image. The dots are also aligned, due to the paving, resulting in a grid pattern effect.

Mean square error between the dithered image and the original

MSE for lena: 0.2129
MSE for wool: 0.1758

Ordered matrix with centered points

With the Matlab Function $X = ordered_threshold(Y, s)$, we only need to change the matrix S to get the results here. Figure 11 and Figure 13 display the results of implementing matrix C_6. This method creates dots of white points, and a bigger sized dot for lighter gray levels. But crosses of black points are formed, as the left space of the white dots. However, with matrix E_6, instead of having black crosses, we have a more balanced view with both black and white dots. Compared to the result of matrix C_6, the crosses disappear dithered images.

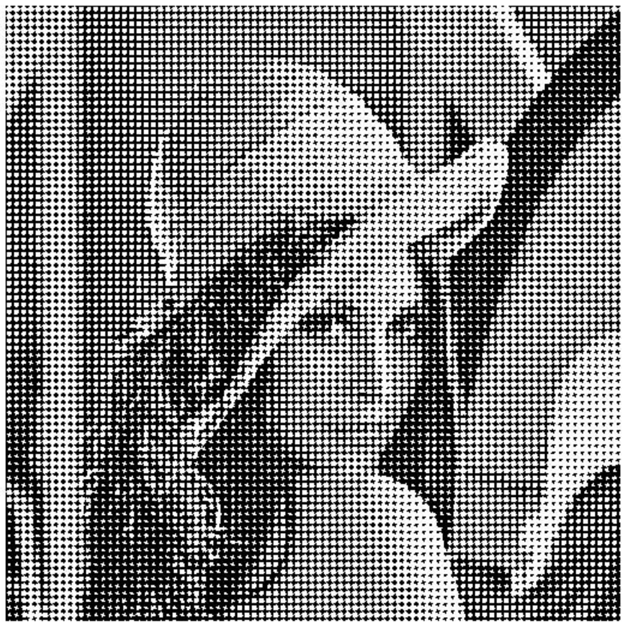

Figure 11: lena-y.png With Ordered Matrix C

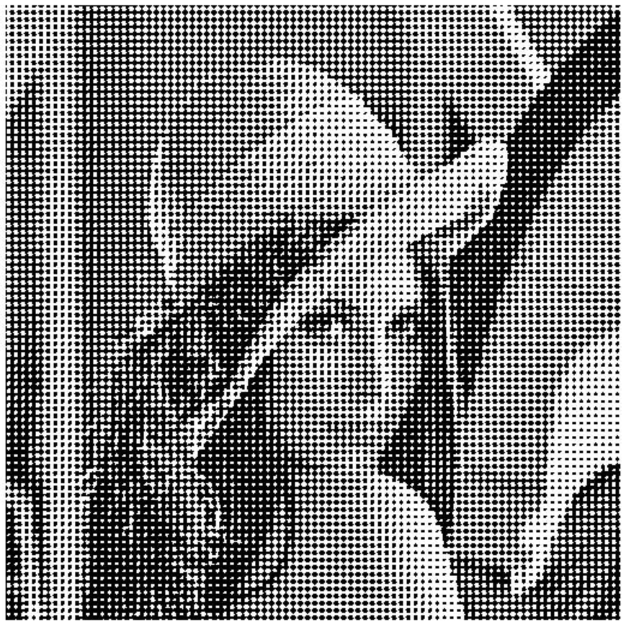

Figure 12: lena-y.png With Ordered Matrix E

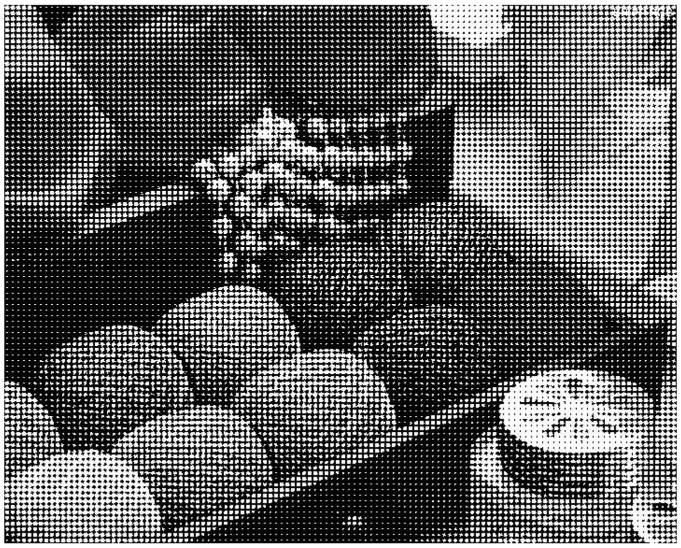

Figure 13: wool.png With Ordered Matrix C

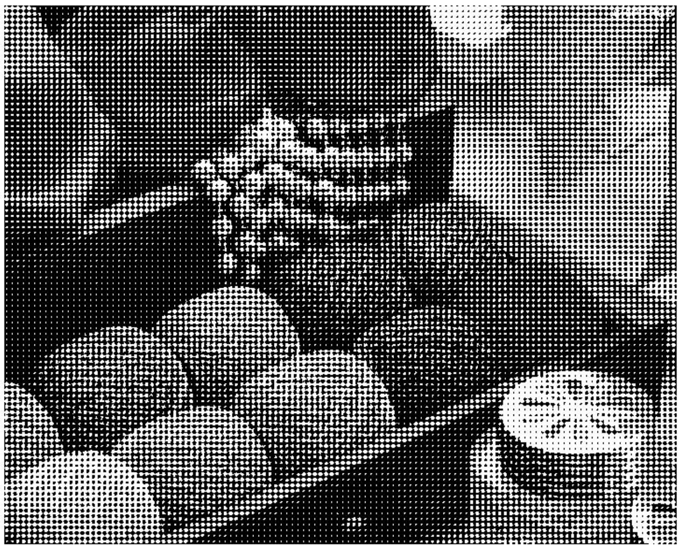

Figure 14: wool.png With Ordered Matrix E

Visual quality

The threshold matrix results in a different pattern reproducing the image like the dot effect seen earlier. The difference is observable when zooming on the same region of both image with different threshold matrix. Otherwise the difference isn't obvious. As seen previously, because of the paving, the repeated pattern of the threshold matrix results in a grid effect.

Mean square error between the dithered image and the original

MSE for lena:
(1) With matrix C: 0.2127
(2) With matrix E: 0.2127

MSE for wool:
(1) With matrix C: 0.1759
(2) With matrix E: 0.1758

Diagonal ordered matrix with balanced centered points

Still using the Matlab Function $X = ordered <_t hreshold(Y, s)$ with a new paving matrix, we can get the result as Figure 15 and Figure 16. Similar as the previous matrix E_6, matrix O_8 is also a balanced matrix. But now, instead of a vertical or horizontal orientation, we have a diagonal one.

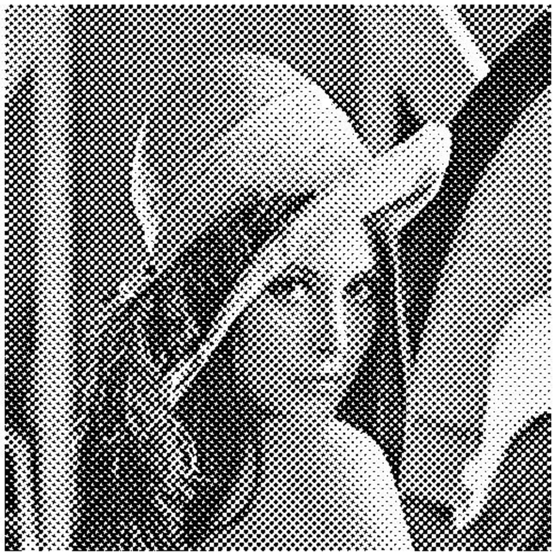

Figure 15: lena-y.png With Diagonal Ordered Matrix O8

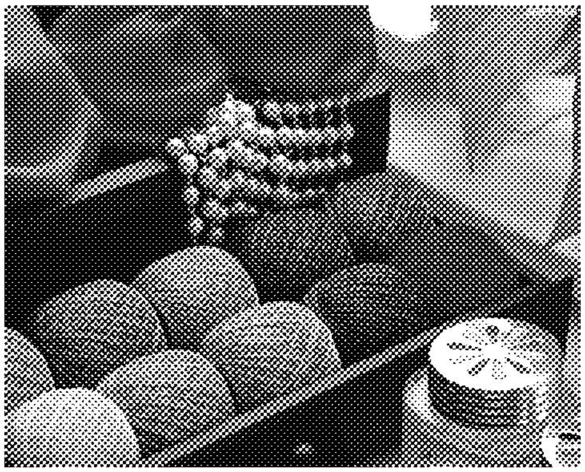

Figure 16: wool.png With Diagonal Ordered Matrix O8

Visual quality

The rendered image is better than the results with the previous ordered threshold methods. More details are observable using the threshold matrix with the diamond pattern. There is still a grid effect but it less affects the rendering of the image compared to the previous grid effect due to the aligned dot patterns; therefore we can conclude that the eye is more sensible to horizontal and vertical orientations than the diagonal ones.

Mean square error between the dithered image and the original

MSE for lena: 0.2054
MSE for wool: 0.1772

Ordered matrix with dispersed dots

The Bayer method creates matrices to minimize the occurrence of low frequencies, which are more sensitive to human eyes, so the dispersed dots resulting by the ordered thresholding are less annoying than previous methods with centered matrices. However, as seen in Figure 17 and Figure 18, new motif of grid pattern appears.

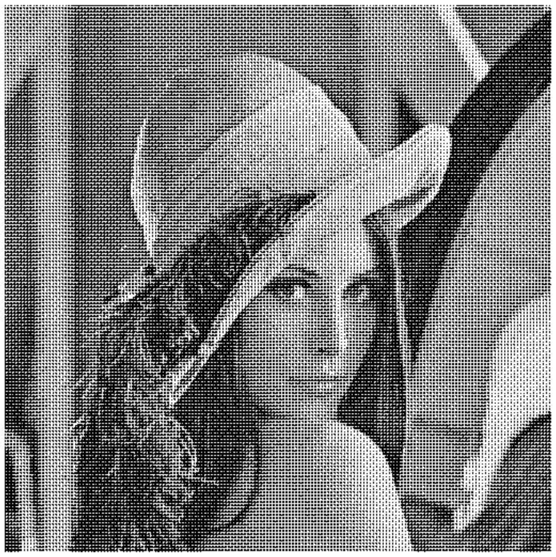

Figure 17: lena-y.png with Ordered matrix with dispersed dots

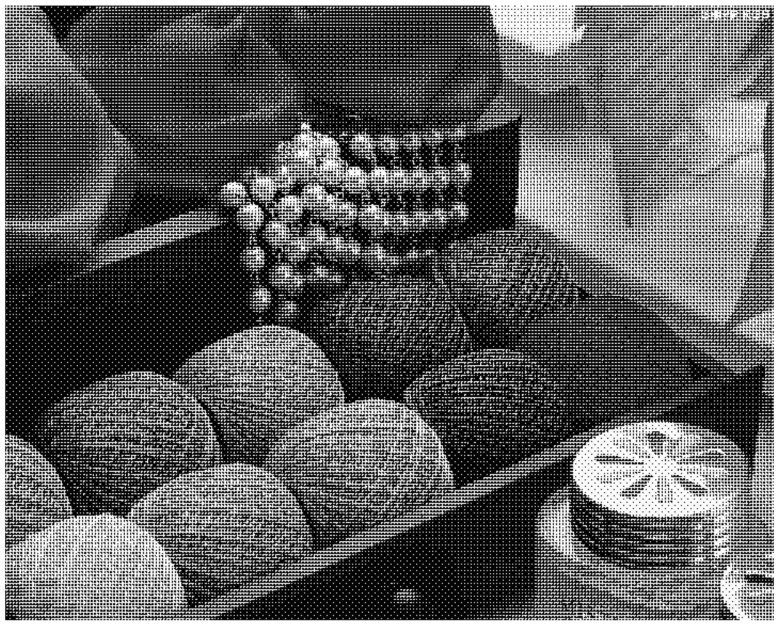

Figure 18: wool.png with Ordered matrix with dispersed dots

Visual quality

The patterns are smaller, and induce a better rendering on the details of the image, leading to a better representation of the image compared to the results of the previous methods. The grid effect still appears again because of the paving of the threshold matrix on the image.

Mean square error between the dithered image and the original

MSE for lena: 0.2128
MSE for wool: 0.1760

Error diffusion method

To implement the *Error Diffusion method*, a Matlab Function $Y = err\_diff(X, D)$ was created, where X refers to the input image and D as the matrix we select. After doing a normalization to the diffusion matrix D, and a zero padding to add necessary borders for the image, by thresholding the image, we can get the diffused result.

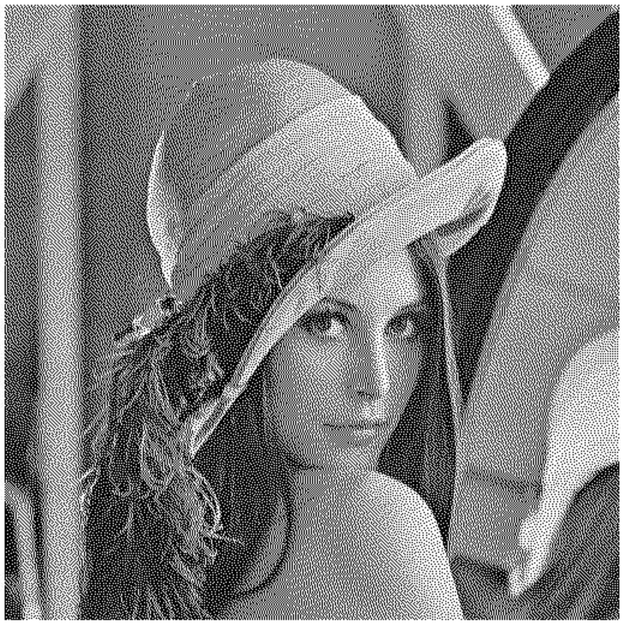

Figure 19: lena-y.png with Error Diffusion Method (Floyd)

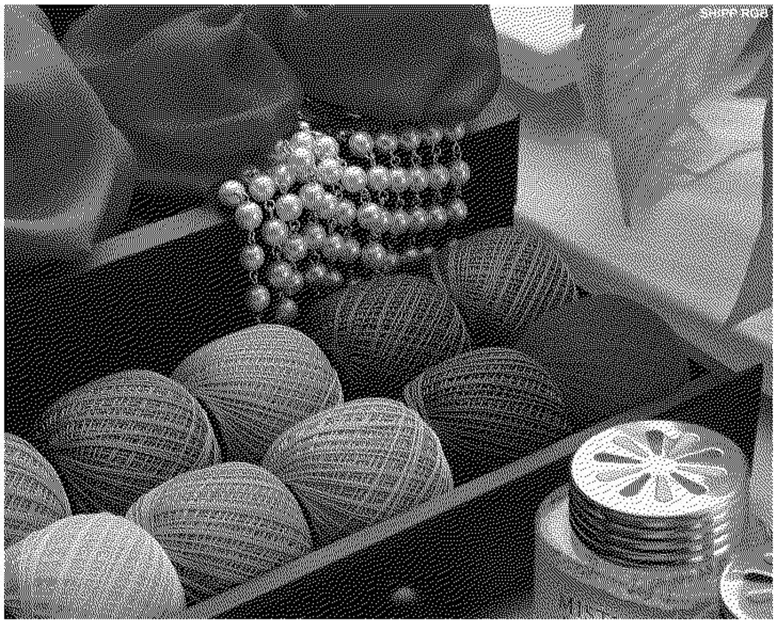

Figure 20: wool.png with Error Diffusion Method (Floyd)

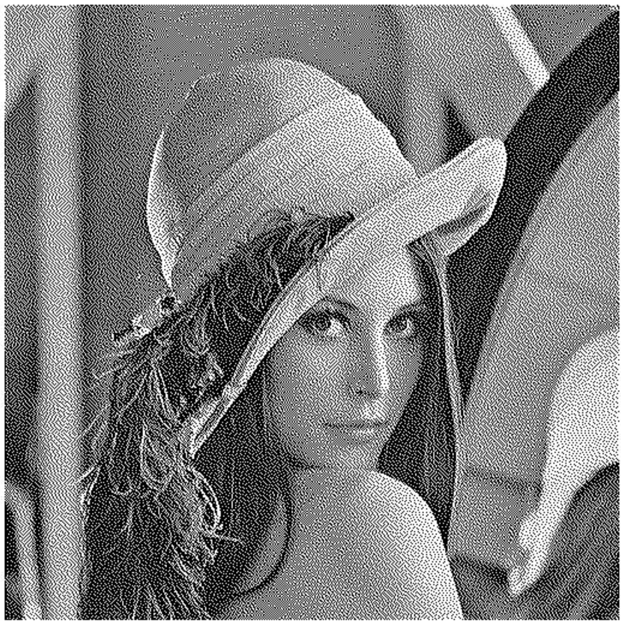

Figure 21: lena-y.png with Error Diffusion Method (Stucki)

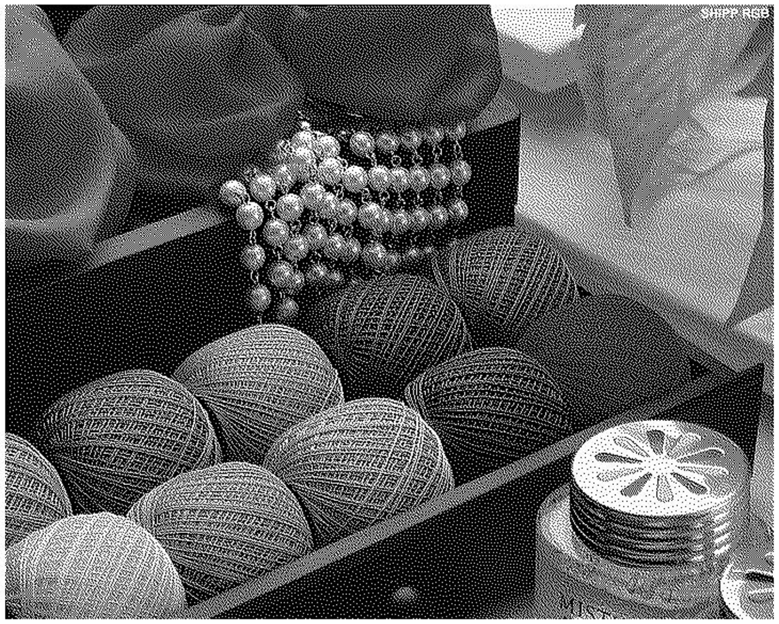

Figure 22: wool.png with Error Diffusion Method (Stucki)

Visual quality

Finally the error diffusion method gives the best rendering of the original image *i.e.* the more details. There is still an observable disturbance specific to the error diffusion matrix used. Using the Stucki matrix, the resulting image is more detailed (notice the sharp contour of the feathers, or the details of the hair) compare to the resulting image using the Floyd matrix (the contour of the feathers is less distinguishable, more blurred).

Mean square error between the dithered image and the original

MSE for lena:
(1) Floyd: 0.2133
(2) Stucki: 0.2110
MSE for wool:
(1) Floyd: 0.1769
(2) Stucki: 0.1718

Discussion

Visual quality

Judging from our human eyes, we can rank the methods above as (from the best quality to the worst one):

- Error Diffusion Method
- Ordered matrix with dispersed dots
- Diagonal ordered matrix with balanced centered points
- Ordered matrix with centered points (while E_6 with a more balanced display than C_6)
- Ordered threshold method
- Random threshold method
- Fixed threshold method

Those matrices with centered points offer a more balanced view. And among these centered ones, diagonal ordered matrix gives the best performance. All the methods cannot eliminate the grid effects (i.e. the motif of the thresholding matrix). As human eyes are more sensitive to low frequencies, when implementing *Ordered matrix with dispersed dots*, the visual quality is improved compared to the previous. However, among these methods, *Error diffusion method* outperforms.

Mean square error between the dithered image and the original

Here, for simplicity, we only select wool.png to analyze the Mean Square Error of different methods. While lena-y.png definitely will get the same conclusion as well.
By doing the normalization to compute MSE, we just divided each pixel by 255.

The MSE is not representative of the quality of the rendered image. Maybe this is due to some constant error on the rendered image not affecting the information like details, but influencing the MSE.

Mean Square Error Comparison	
Method Name	MSE
Fixed threshold method	0.0949
Random threshold method ($noiserange = 10$)	0.0951
Ordered threshold method	0.1758
Ordered matrix with centered points (C_6)	0.1759
Ordered matrix with centered points (E_6)	0.1758
Diagonal ordered matrix with balanced centered points	0.1760
Ordered matrix with dispersed dots	0.1772
Error Diffusion Method (Floyd)	0.1769
Error Diffusion Method (Stucki)	0.1718

Complexity

Ranking of the methods by complexity starting with the simplest one :

- The *Fixed threshold method* is the simplest method, its complexity in terms of number of operations is proportional to the size of the image.
- The *Random threshold method* is more complex because we need to create a noise matrix of the size of the image and add it to the original image before implementing the *Fixed threshold method.*
- The *Ordered threshold method* is even more complex because we need to compute a quantization of the imag ebefore the threshold part and we need to compare each pixel of the matrix with the corresponding element of paving matrix (not a constant threshold).
- The *Ordered matrix with centered points method* is of the same complexity as the previous method if we consider matrices of the same size which is the case here.
- The *Diagonal ordered method* is more complex because additional operations are necessary to make the concatenation of the threshold matrix O_8 which is also bigger in size, therefore induces more computation.
- The *Ordered matrix with dispersed dots method* is even more complex because in addition to the concatenation like the previous method, we need to compute scalar-matrix multiplications and matrices additions.
- The *Error diffusion method* is the more complex method because of the feedback loop, and because the diffusion matrix is add around each pixel , not like the paving done previously. It also depends on the size of the diffusion matrix used (higher complexity for a bigger matrix).