Ioannis Pitas

Digital Image Processing Fundamentals

Chapter 3

Digital image filtering and enhancement Lab exercises in ÅÉÊÏÍÁ

Thessaloniki 1998

Chapter 3: Digital image filtering and enhancement

EIKONA contains several tools to filter out noise from a digital image. In the following exercises, we use some of the filters that EIKONA implements in order to filter an image corrupted with noise. The latter is assumed to be a version of BABOON corrupted with impulsive ("salt and pepper") noise. We will assume that the image BABOON is loaded in image buffer 0. We can create a corrupted version of this image by following the sequence "***Black and white Processing Impulsive***" as seen in Exercise 4, Chapter 1. In the following, we assume that the corrupted input image has undergone impulsive noise addition with the noise probability set equal to 0.1 and the minimum and maximum impulse values are set equal to 0 and 255 respectively. This image can be seen in Figure 1.

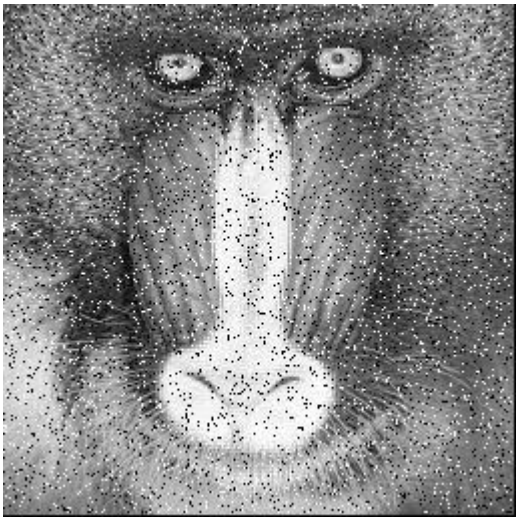


Figure 1. Input image to be used for filtering.

Exercise 3.1: Image filtering using the median filter

We can easily filter out the noise from the corrupted image by utilizing a median filter. This can be done by selecting "***Black and white Filtering Median***" from the program menu. If the image to be filtered (the corrupted version of BABOON in our example) was stored in BW buffer 1, then you should enter 1 and **<NewBuffer>** as the source and destination buffers, respectively. The filter window size is also required. A 3×3 window is usually employed, thus we fill in the number 3 in the relative fields and press the "OK" button. The output pixels are equal to the median of their 3×3 neighbourhood in the input image. The result of this operation can be seen in Figure 2. As you can see, the filtered image closely resembles the original one and the impulsive noise is removed.

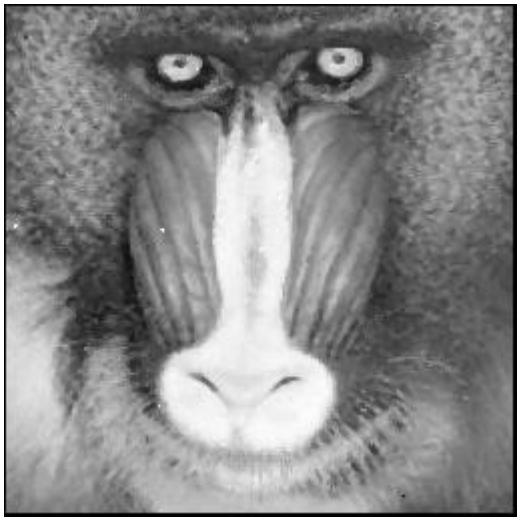
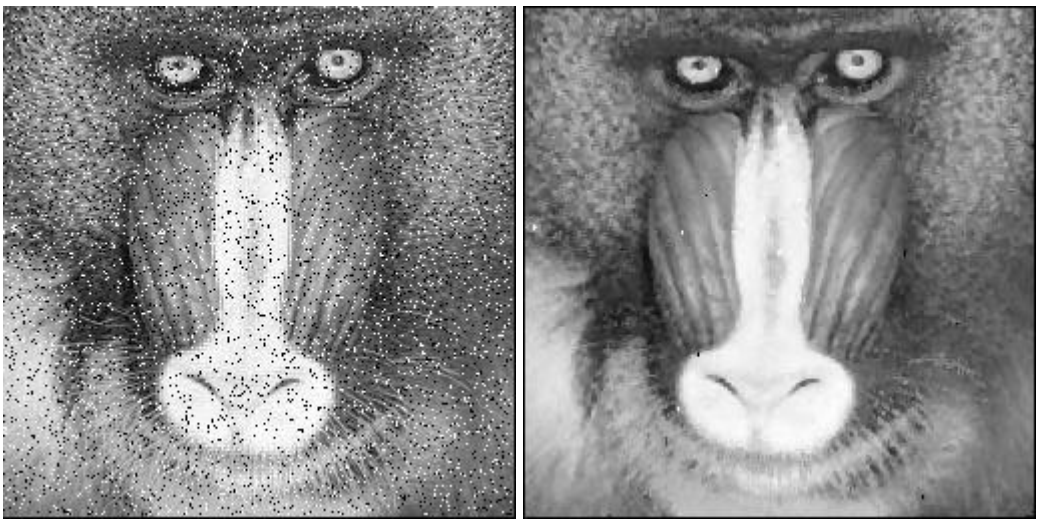


Figure 2. The corrupted image after median filtering with a window of size 3×3.

Exercise 3.2: The separable median filter

In this exercise, we demonstrate the use of the separable median filter which is a non-linear filter. We use the image BABOON corrupted by impulsive noise (Noise probability=0.1, Min. impulse value=0, Max. impulse value=255) as input. We can apply a separable median filter to this image by selecting "***Black and white Non-linear filtering Median Filters Separable Median***". We specify the corrupted image as the source image and **<NewBuffer>** as the destination buffer. In the window size fields we enter 3, 3. These values define the size of the filter window. The resulting image is depicted in Figure 3b. The impulsive noise is eliminated.



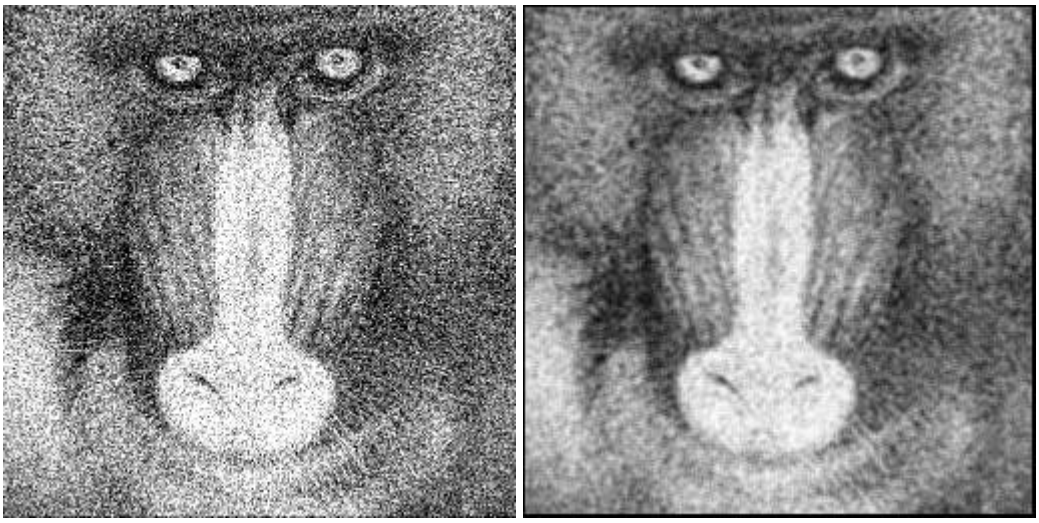
(a) (b)

Figure 3. (a) BABOON corrupted by impulsive noise, (b) The same image filtered by a seperable median filter.

Exercise 3.3: Image filtering using the moving average filter

In this exercise, we use EIKONA in order to apply the moving average filter on an image. This filter can be accessed by selecting ***"Black and white Filtering Movav"*** on the program menu. In the dialog box that pops up, we specify the source and destination buffers and the filter window size.

In the following, we use the moving average filter to filter a version of BABOON that has been corrupted with Gaussian additive noise as explained in Chapter 1 (Figure 4a). By taking a look at the filtered image (Figure 4b), we conclude that the moving average filter efficiently removes Gaussian additive noise. However, it has the disadvantage of blurring the edges in the image, thus reducing its quality.

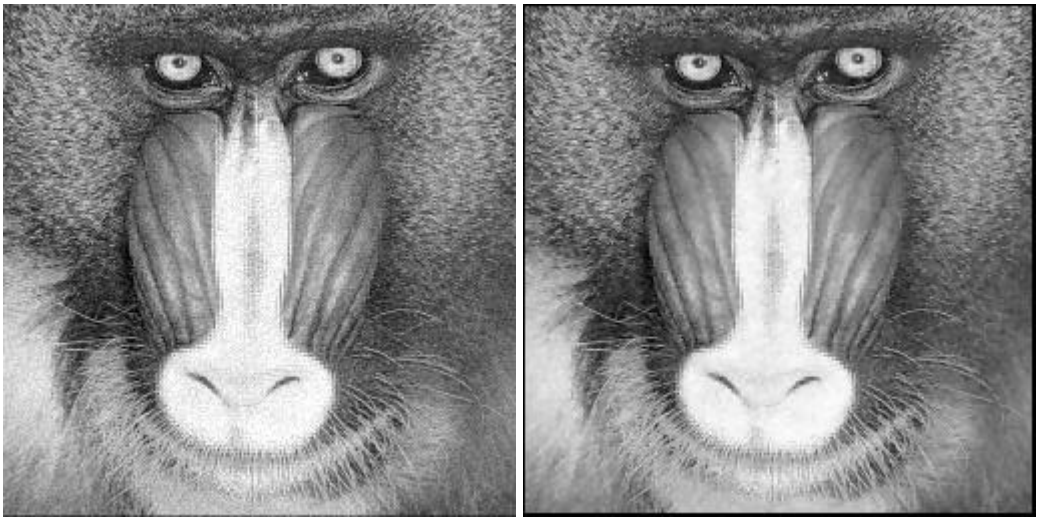


(a) (b)

Figure 4. The image BABOON corrupted with Gaussian additive noise (Noise standard deviation=50) and the result of its filtering with the moving average filter.

Exercise 3.4: The local adaptive filter

Another non-linear filter is the local adaptive filter, which is based on local statistics. Let us suppose that we have an image of BABOON corrupted by additive uniform noise of range 30 as shown in Figure 5a. Now, the local adaptive filter can be utilized to filter out the noise. We select the corrupted image and **<NewBuffer>** as the source and destination buffer respectively. In the window size fields we enter 3,3 and in the *Noise Variance* field we enter 75. The result of the filtering can be seen in Figure 5b.



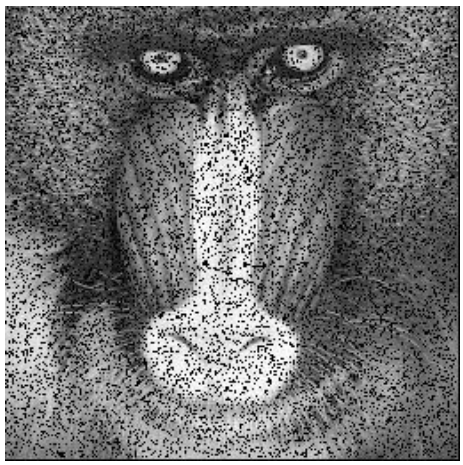
(a) (b)

Figure 5. (a) BABOON corrupted by additive uniform noise, (b) The same image filtered by a local adaptive filter.

Exercise 3.5: Image filtering using the minimum and maximum filters

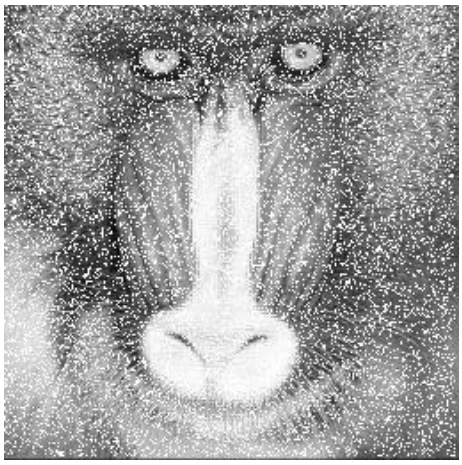
The minimum and maximum filters are available through the menu options ***"Black and white Filtering Mini"*** and ***"Black and white Filtering Maxi"*** respectively. Assuming that the corrupted version of BABOON is loaded in the BW image buffer 1, we choose 1 and **<New Buffer>** as source and destination buffers respectively, and set the filter window size to 3×3. This selection creates a 3×3 filter window (local neighbourhood). The output pixel value is the maximum or minimum brightness value within the respective local input image neighbourhood.

In Figure 6, we present some examples of the maximum and minimum filters. We can easily point out that the minimum filter efficiently removes the positive impulses (white dots), but not the negative ones (black dots), while the opposite stands for the maximum filter.



(a)

(b) (c)



(d)

(e) (f)

Figure 6. (a) BABOON corrupted with negative impulsive noise (Noise probability=0.2, maximum and minimum impulse values=0, 255), (b) maximum filter output on image 6a, (c) minimum filter output on image 6a, (d) BABOON corrupted with positive impulsive noise (Noise probability=0.2, maximum and minimum impulse values=0, 255), (e) minimum filter output on image 6d, (f) maximum filter output on image 6d.

Exercise 3.6: Grayscale opening and closing

Assuming that BABOON is loaded in BW buffer 1, we can perform a grayscale opening operation by performing a grayscale erosion followed by a grayscale dilation. In the same way, we can perform a grayscale closing operation by a dilation followed by an erosion.

The first step can be performed by selecting the menu option *"**Black and white Non-linear filtering Morphology Grayscale Erode**"* and specifying BABOON as the source and **<NewBuffer>** as the destination. The opened image can be obtained by performing a grayscale dilation operation on the eroded image. We select the menu option *"**Black and white Non-linear filtering Morphology Grayscale Dilate**"* and enter the eroded image as the source and **<NewBuffer>** as the destination. The final opened image is shown in Figure 7c.

The closed image can be obtained by performing a grayscale erosion on the dilated image. We select the menu option *"**Black and white Non-linear filtering Morphology Grayscale Dilate**"* and enter BABOON as the source and **<NewBuffer>** as the destination. Then we select the menu option *"**Black and white Non-linear filtering Morphology Grayscale Erode**"* and enter the dilated image as the source and **<NewBuffer>** as the destination. The final closed image is shown in Figure 7d.

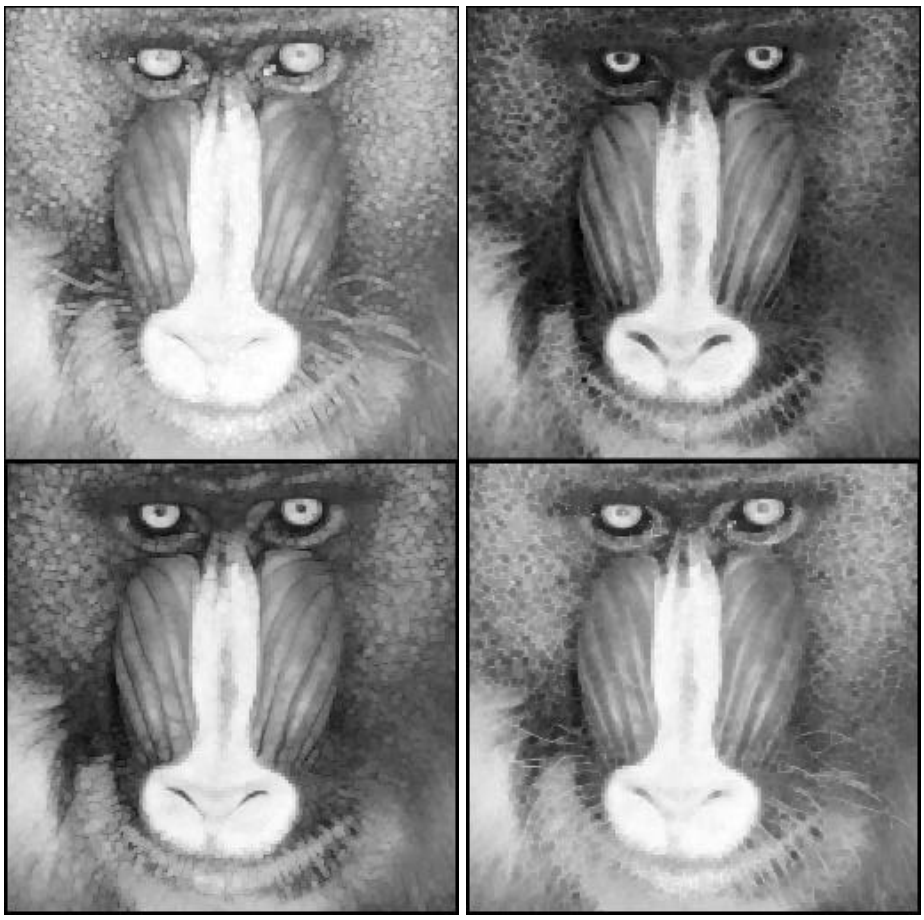


Figure 7. (a) Result of dilation on BABOON, (b) result of erosion on BABOON, (c) final opened image, (d) final closed image.

Exercise 3.7: Histogram Equalization

In many cases we are interested in improving the visual quality of an image. For example, the contrast of an image can be improved, by performing histogram equalization. In this exercise, we perform an example of this operation, by equalizing the histogram of the image BABOON.

The image histogram provides us with information regarding its pixel intensity values. It can be computed by selecting ***"Black and white Analysis Histogram"***. If the image of BABOON is loaded in image buffer 0, then we choose BW buffer 0 in the window that appears. After pressing the "OK" button, the histogram is displayed in a new window. The histogram can be seen in Figure 8.

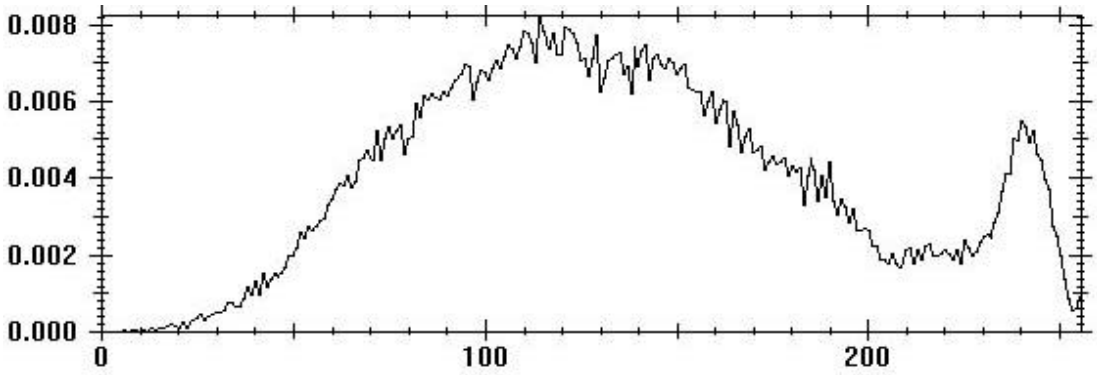


Figure 8. The histogram of image BABOON.

Now, we perform histogram equalization on BABOON and store the resulting image in a new buffer. Histogram equalization redistributes the brightness values of the pixels in an image so that they represent more evenly the entire range of brightness levels. It can be performed by selecting the menu option "***Black and white Filtering Histeq***". In the dialog box that appears, we specify BW buffer 0 and **<NewBuffer>** as the source and destination buffers, respectively. The resulting image is shown in Figure 9, while its histogram is given in Figure 10. We can see that the resulting histogram is uniformly spread in the entire range [0 255].

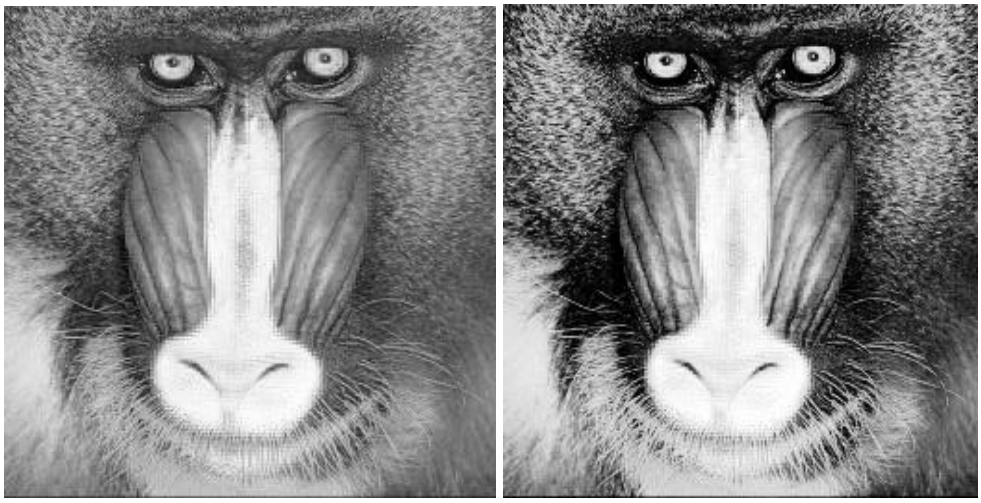


Figure 9. The image BABOON before and after histogram equalization.

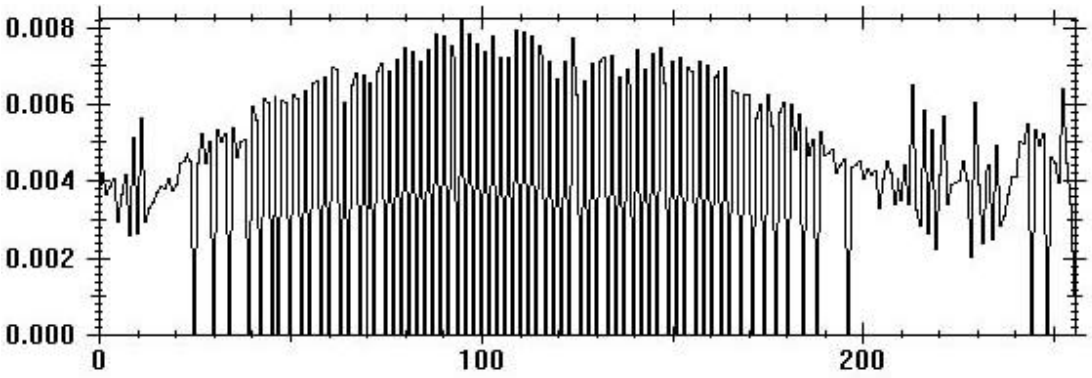


Figure 10. The histogram of the image of Figure 9b.

Exercise 3.8: Image sharpening

In this exercise, we use EIKONA to perform image sharpening. Sharpening "restores" blurry images by increasing the contrast of adjacent pixels. It is essentially a high-pass operator.

Assuming that the image BABOON is loaded in BW buffer 0, we can sharpen the left half of it as follows: First we make a copy of the image in a buffer by selecting the menu option "***Black and White Basic Copy***". Using the mouse, we define a ROI that covers the left half of the original image. Then, we select "***Black and White Processing Sharp***". In the dialog box that appears, we specify the original image and its copy as the source and destination buffers respectively. The result of image sharpening on the half part of the original image can be seen in Figure 11. Sharpening has greatly improved image contrast.

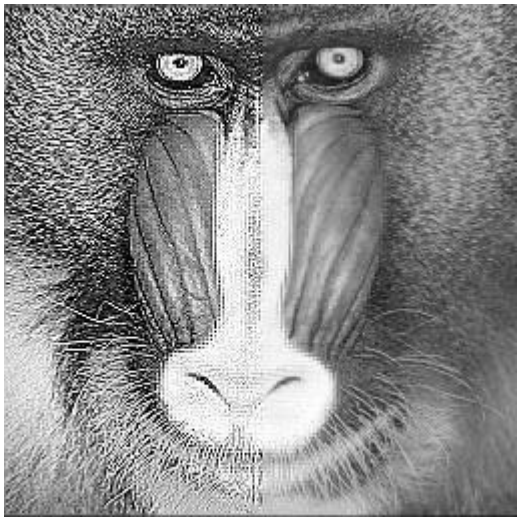


Figure 11. The image BABOON after sharpening of its left part.

Exercise 3.9: Image filtering using convolution

In this exercise, we filter BABOON by computing its convolution with a floating point coefficient mask. However, special preparation for this procedure is required. We need to alter the contents of the file WIN_F.PAR which exists in the EIKONA directory. The user must edit this file prior to applying the filtering operation and fill in the filter coefficients. The contents of the WIN_F.PAR file could be the following:

```
 3
 3
 0
 1
 1
-1
 0
 1
-1
-1
 0
```

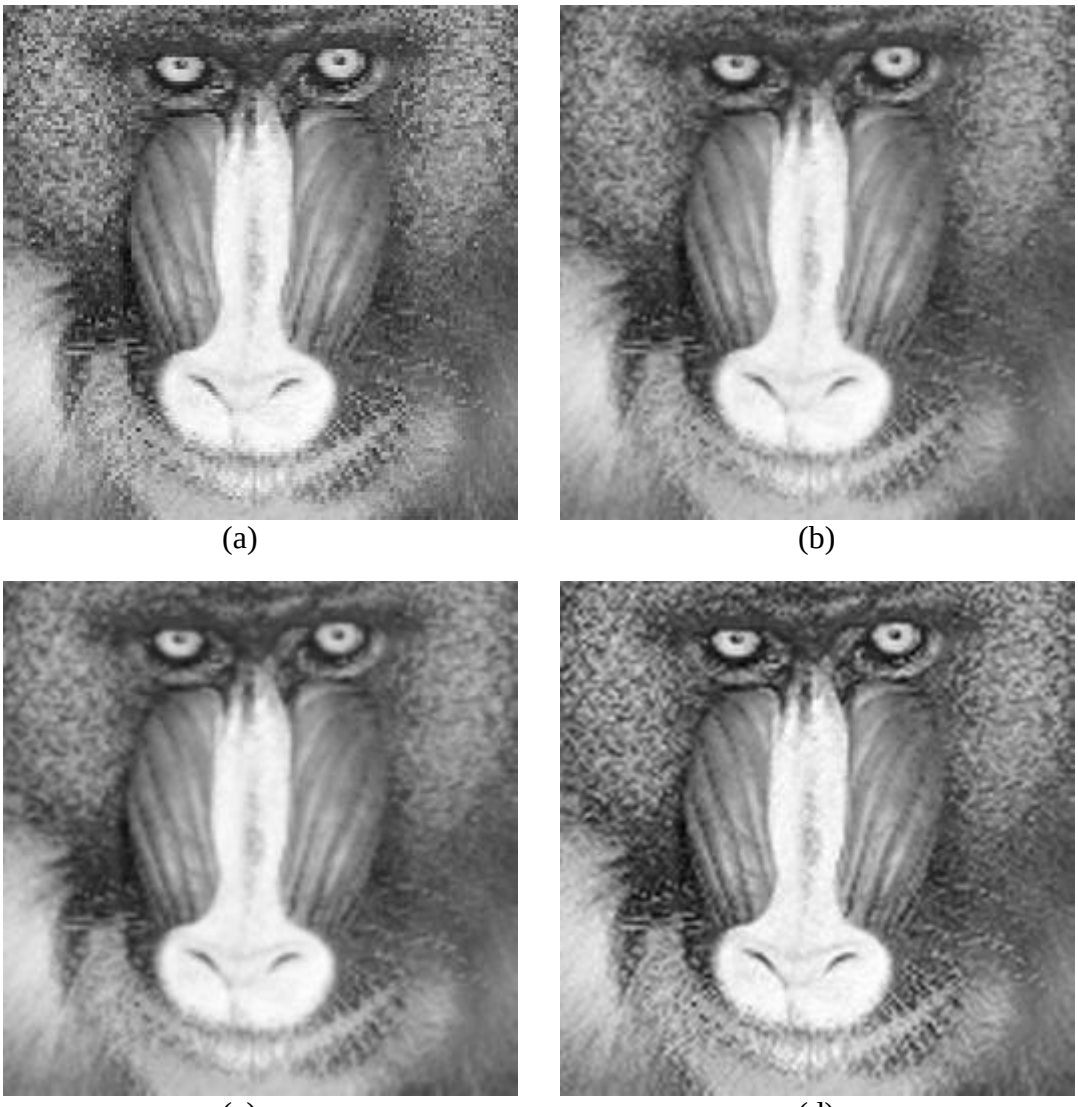
The first two numbers represent the columns and rows respectively of the floating point mask (3×3 in this case). Nine floating point numbers follow, that correspond to the mask elements written in a row-wise manner. This mask, when used with convolution, corresponds to the vertical line detection filter. The convolution operation is available through the menu option "***Black and White Filtering Convolution***". In the dialog box that pops up, we specify the source and destination buffers and then we press the "OK" button. The result of the operation can be seen in Figure 12. We note that in the convolution procedure negative values in the output image are displayed as white. This happens because EIKONA uses unsigned integers to represent an image buffer, which causes a negative value A to be represented by the value 255-A. This explains the presence of many white pixels in Figure 12. Pixels of this kind are not present in the output of the vertical line detector (Chapter 5, Exercise 3). In the latter, the output image consists of the absolute values of the results the line vertical detector produces.

Figure 12. Convolution of BABOON with the above mask.

Exercise 3.10: Image interpolation (zooming)

In this exercise, we zoom an image of size 128×128 to a size of 512×512 pixels by interpolation. The interpolation process is accessible through the menu option "***Black and White Processing Zoom***". In the dialog that pops up, we specify the source and destination buffers. For the source buffer, we select the 128×128 image BW buffer that contains BABOON and for the destination buffer **<New Buffer>**. Then we enter 4 in the *Zoom factor* field and select one of the four supported interpolation routines. Finally, we press the "OK" button.

After completing the above procedure, the destination buffer (of size 512×512) contains the interpolated version of BABOON. In Figure 13, we present the results of the interpolation procedure four times, one for each interpolation routine. Notice how the choice we make for the interpolation order effects the output image. The cubic B-spline interpolation produces the best results.



(a) (b)

(c) (d)

Figure 13. (a) Zero-order interpolation of BABOON, (b) Linear interpolation of BABOON, (c) Bell-shaped interpolation of BABOON, (d) Cubic B-spline interpolation of BABOON.

Exercise 3.11: Image halftoning

In this exercise, we perform halftoning on a grayscale image. Halftoning changes a grayscale image to a binary one. The latter is still perceived as greyscale due to an interesting characteristic of the human vision: the human eye interprets neighbouring black and white pixels as gray levels in the image. Halftoning is commonly used when we want to print an image using a printer. Most current printer technologies can print only binary images. Thus, halftoning is useful in converting a grayscale image to a binary format in order to print it.

The halftoning operation is available through the menu option "***Black and White Basic Halftone***". In the dialog box that appears, we specify the source and destination BW image buffers and enter a value for the *Number of halftones* field (this must be of the form $n \times n$, n=2,3,4).

In Figure 14, the output of the halftoning process on BABOON is depicted. The number of halftones used in this operation was 4. By using a larger number of halftones, we increase the perceived luminance values but we decrease the resolution of the haftoned image.

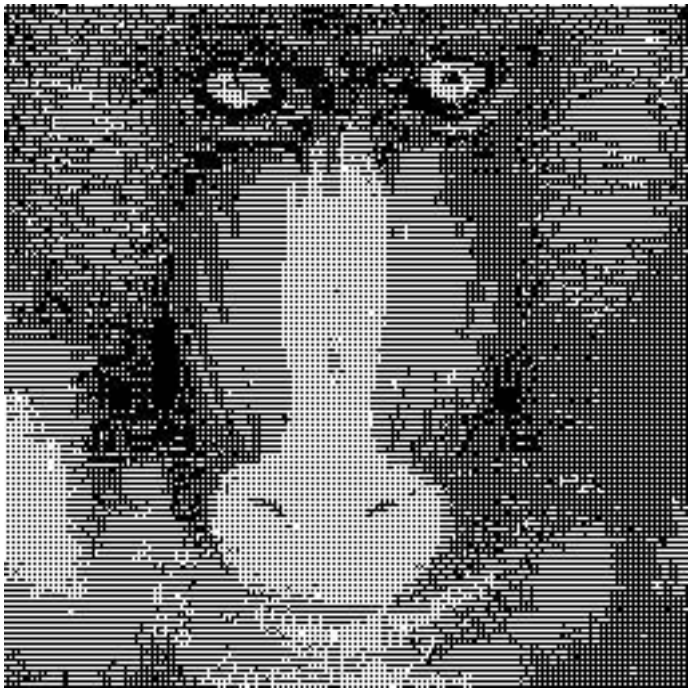


Figure 14. The image BABOON after halftoning with four halftones.