VNUHCM - UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

MORPHOLOGICAL OPERATOR

ADVANCED DIGITAL IMAGE AND VIDEO PROCESSING

Lâm Thanh Ngọc - 21127118

Class: 21TGMT

Lecturers:

Lý Quốc Ngọc

Phạm Minh Hoàng

Nguyễn Mạnh Hùng

22nd March 2024

Contents

1 Assessment 2
2 Supporting functions 2
3 Morphological operators functions 3
3.1 Dilation 3
3.2 Erosion 5
3.3 Opening 7
3.4 Closing 9
3.5 Boundary Extraction 11
3.6 Morphological Gradient 12
3.7 Top-Hat 13
3.8 Smoothing 14
4 References 15

1 Assessment

Function	Level of completion	Assessment
Dilation	100%	Completed for both binary and grayscale image
Erosion	100%	Completed for both binary and grayscale image
Opening	100%	Completed for both binary and grayscale image
Closing	100%	Completed for both binary and grayscale image
Boundary Extraction	100%	Completed for binary image
Morphological Gradient	100%	Completed for grayscale image
Top-Hat	100%	Completed for grayscale image
Smoothing	100%	Completed for grayscale image

2 Supporting functions

def pre_process(image_path) and def pre_process_gray(image_path)

- **Input:** image_path is the path to the image.
- **Output:** The function returns the image in RGB and grayscale format.

Both functions are all used for reading the image and converting it to appropriate format (binary or grayscale). The function pre_process is used for binary images with the function cv.imread(image_path, 0) of opencv where 0 is the flag for reading the image in grayscale and the function cv.threshold(img, 128, 255, cv.THRESH_BINARY cv.THRESH_OTSU)| of opencv to convert the image into binary format.

Similarly, the function pre_process_gray is used for reading a grayscale image with the function cv.imread(image_path) of opencv to read the image and converting it to grayscale format using the function cv.cvtColor(img, cv.COLOR_BGR2GRAY) of opencv.

def print_img(imgs, row, col)

The function is used to display the array of images in a grid format. It takes the array of images, the number of rows and the number of columns as input, then uses the function plt.subplot(row, col, index_of_images) of matplotlib to display the images in a grid format.

3 Morphological operators functions

3.1 Dilation

The dilation operator is used to expand the boundaries of the object in the image, so that the small holes in the object are filled with thicker boundaries to make the object more obvious. It comes up with the rule that the pixel value of the output image is the maximum pixel value of the input image under the kernel which means if any pixel value is overlapped by the kernel, the pixel value of the output image at the center of the kernel will be the maximum pixel value of the input image. This idea is implemented in the 2 following functions:

def dilation(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the dilated image.

This function is used to manually perform the dilation operation on the binary image. From the above idea, the function first creates a zero matrix with the same size as the input image and add zero padding to the input image. Then, it iterates through the input image and at each pixel, it checks if there is any pixel is overlapped by the kernel with np.any() of numpy. If it is, the pixel value of the output image at the center of the kernel will be the maximum pixel value of the input image under the kernel, in this case, it is 1. Finally, the function returns the dilated binary image.

def dilation2(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the dilated image.

Similarly to the dilation function used for the binary image, this function is implemented with the same idea but the maximum pixel value of the input image under the kernel in this case is taken by np.max() function of numpy. The function returns the dilated grayscale image.

Result images:

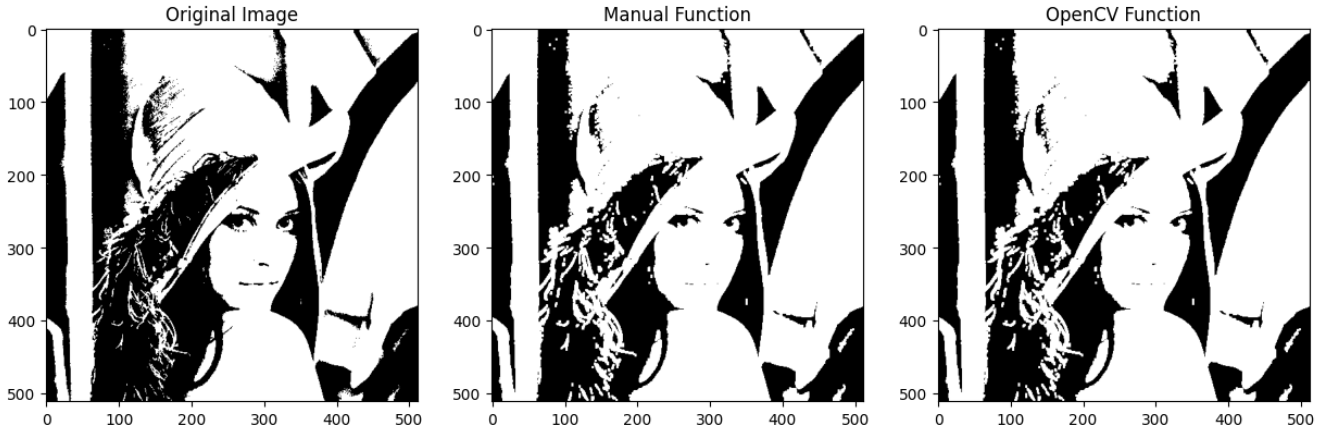

Figure 1: Dilation on binary image with kernel size 3x3

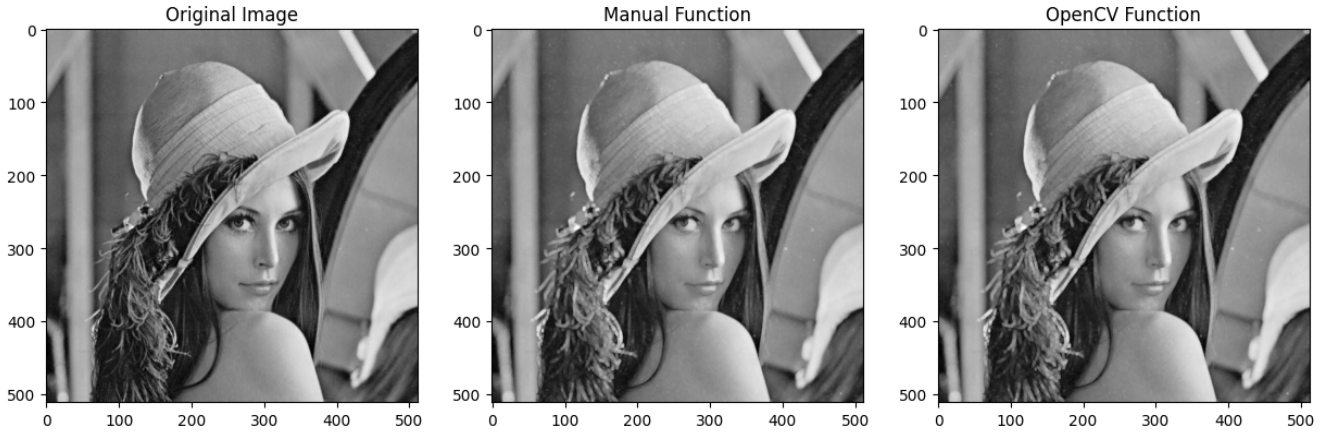

Figure 2: Dilation on grayscale image with kernel size 3x3

As it can be seen from the result images, the small holes in the object are filled with the maximum pixel value of the input image under the kernel and the filled shapes become larger. For comparision, the manual dilation operator function does not show much differences from the built-in function of opencv `cv.dilate(image, kernel)`.

3.2 Erosion

The erosion operator is used to shrink the boundaries of the object in the image, so that the small holes in the object are removed to make the object more obvious. It comes up with the rule that the pixel value of the output image is the minimum pixel value of the input image under the kernel which means if any pixel value is overlapped by the kernel, the pixel value of the output image at the center of the kernel will be the minimum pixel value of the input image. This idea is implemented in the 2 following functions:

def erosion(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the eroded image.

This function is used to manually perform the erosion operation on the binary image. From the above idea, the function first creates a zero matrix with the same size as the input image and add zero padding to the input image. Then, it iterates through the input image and at each pixel, it checks if there is any pixel is overlapped by the kernel with np.all() of numpy. If it is, the pixel value of the output image at the center of the kernel will be the minimum pixel value of the input image under the kernel, in this case, it is 0. Finally, the function returns the eroded binary image.

def erosion2(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the eroded image.

Similarly to the erosion function used for the binary image, this function is implemented with the same idea but the minimum pixel value of the input image under the kernel in this case is taken by np.min() function of numpy. The function returns the eroded grayscale image.

Result images:

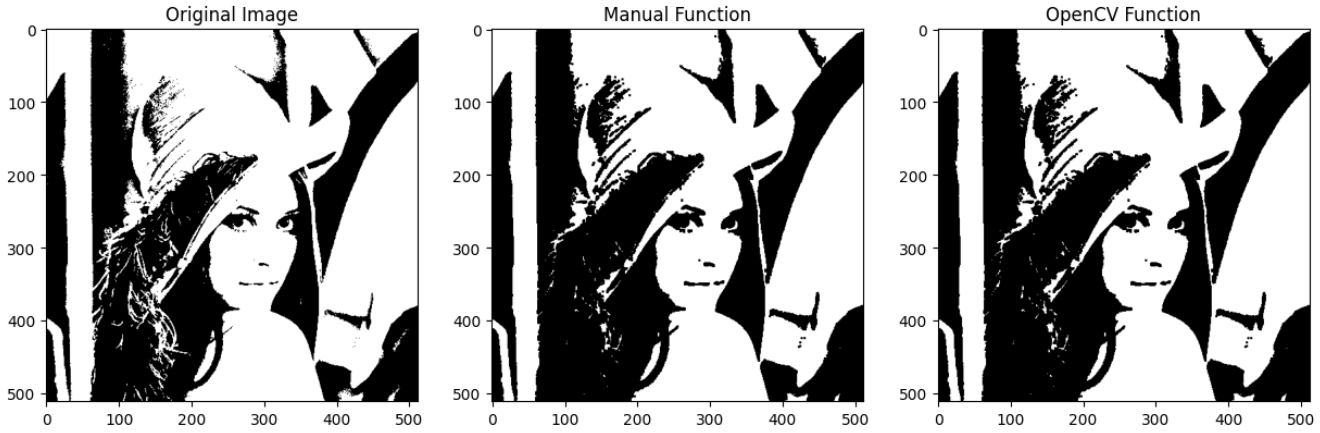

Figure 3: Erosion on binary image with kernel size 3x3

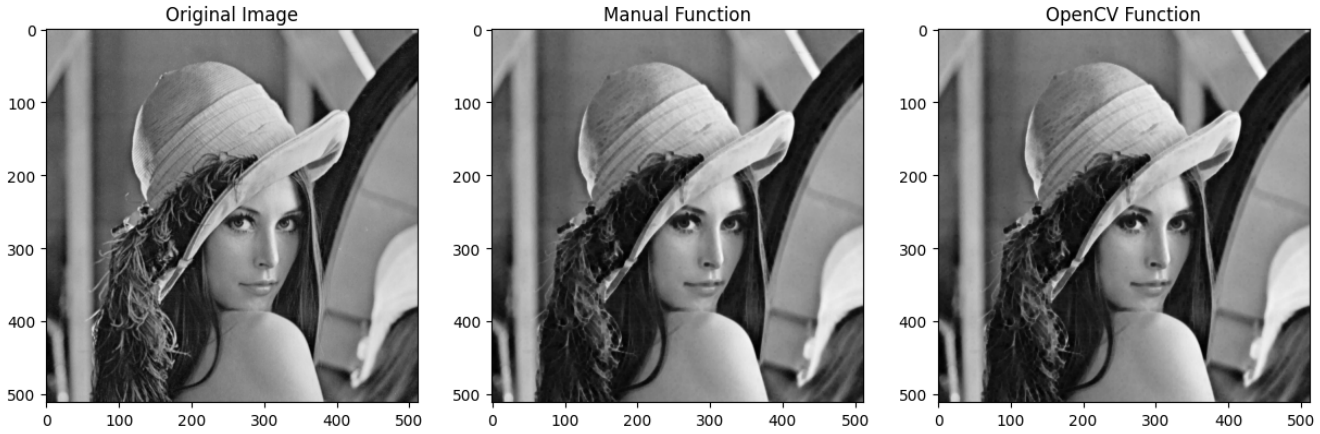

Figure 4: Erosion on grayscale image with kernel size 3x3

As it can be seen from the result images, the small holes in the object are removed and the object becomes smaller. For comparision, the manual erosion operator function does not show much differences from the built-in function of opencv cv.erode(image, kernel).

3.3 Opening

The opening operator is used to remove the small objects in the image and smooth the object boundaries. It is the combination of the erosion operator followed by the dilation operator. This idea is implemented in the 2 following functions:

def opening(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the opened image.

This function is used to manually perform the opening operation on the binary image. From the above idea, the function first performs the erosion operation on the input image with the function erosion(img, kernel) and then calls the dilation operation on the eroded image with the function dilation(img, kernel). Finally, the function returns the opened binary image.

def opening2(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the opened image.

Similarly to the opening function used for the binary image, this function is implemented with the same idea but the erosion and dilation operations are replaced by the functions erosion2(img, kernel) and dilation2(img, kernel). The function returns the opened grayscale image.

Result images:

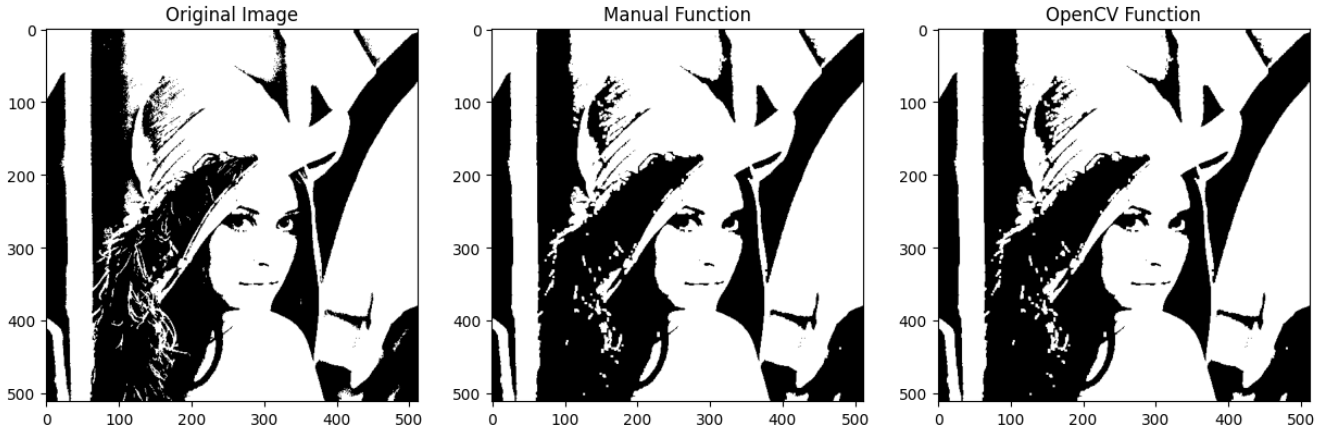

Figure 5: Opening on binary image with kernel size 3x3

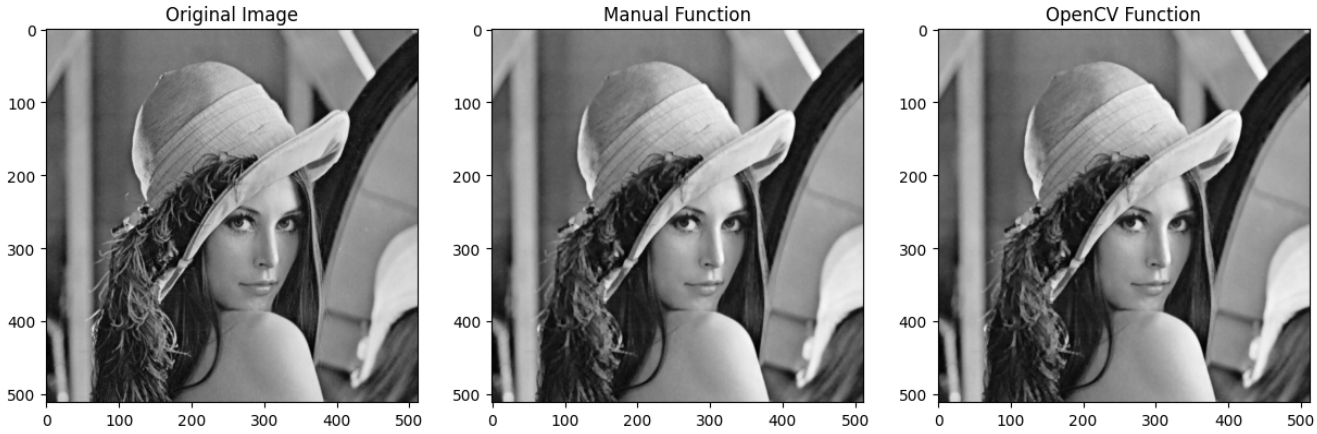

Figure 6: Opening on grayscale image with kernel size 3x3

As it can be seen from the result images, the small objects in the image are removed and the object boundaries are smoothed. For comparision, the manual opening operator function does not show much differences from the built-in function of opencv cv.morphologyEx(image, cv.MORPH_OPEN, kernel).

3.4 Closing

The closing operator is used to remove the small holes in the object in the image and smooth the object boundaries. It is the combination of the dilation operator followed by the erosion operator. This idea is implemented in the 2 following functions:

def closing(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the closed image.

This function is used to manually perform the closing operation on the binary image. From the above idea, the function first performs the dilation operation on the input image with the function dilation(img, kernel) and then calls the erosion operation on the dilated image with the function erosion(img, kernel). Finally, the function returns the closed binary image.

def closing2(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the closed image.

Similarly to the closing function used for the binary image, this function is implemented with the same idea but the dilation and erosion operations are replaced by the functions dilation2(img, kernel) and erosion2(img, kernel). The function returns the closed grayscale image.

Result images:

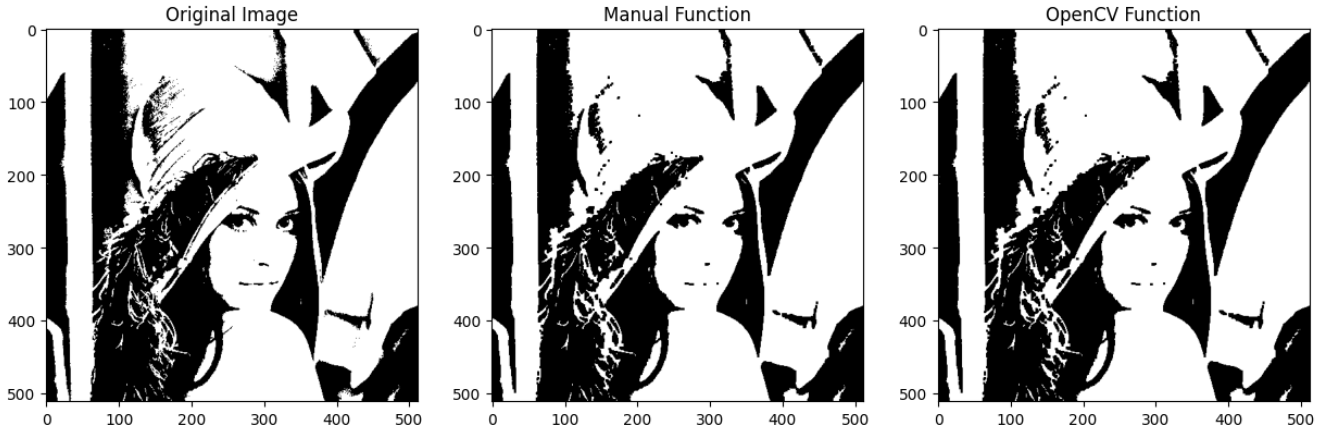

Figure 7: Closing on binary image with kernel size 3x3

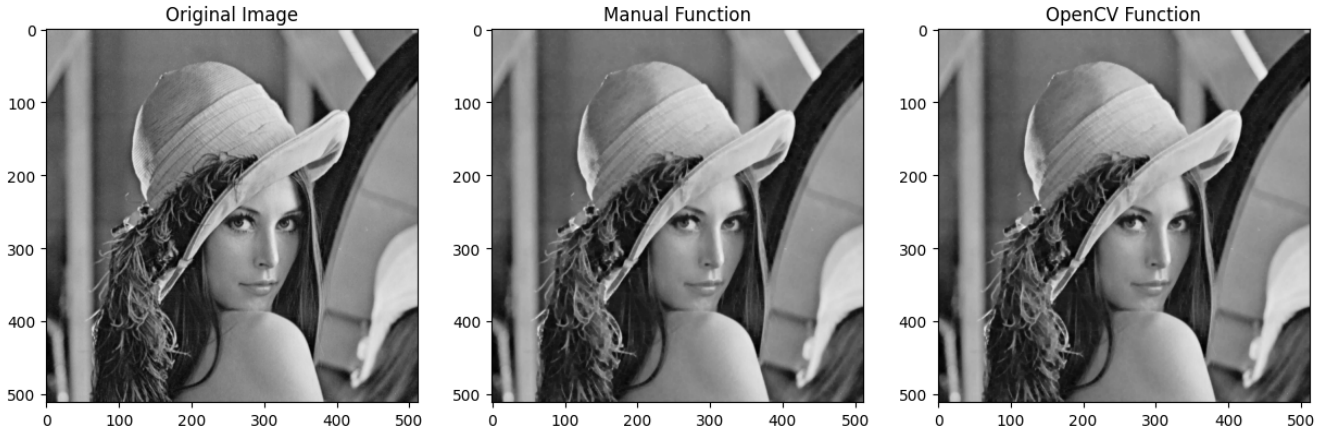

Figure 8: Closing on grayscale image with kernel size 3x3

As it can be seen from the result images, the small holes in the object in the image are removed and the object boundaries are smoothed. For comparision, the manual closing operator function does not show much differences from the built-in function of opencv cv.morphologyEx(image, cv.MORPH_CLOSE, kernel).

3.5 Boundary Extraction

The boundary extraction operator is used to extract the boundaries of the object in the binary image. It is the difference between the dilation and the erosion of the input image. This idea is implemented in the following function:

def boundary_extraction(img, kernel)

- **Input:** img is the image in binary format and kernel is the structuring element.
- **Output:** The function returns the boundary extracted image.

This function is used to manually perform the boundary extraction operation on the binary image. From the above idea, the function first performs the dilation operation on the input image with the function dilation(img, kernel) and the erosion operation on the input image with the function erosion(img, kernel). Then, it finds the difference between the 2 functions and returns the boundary extracted binary image.

Result images:

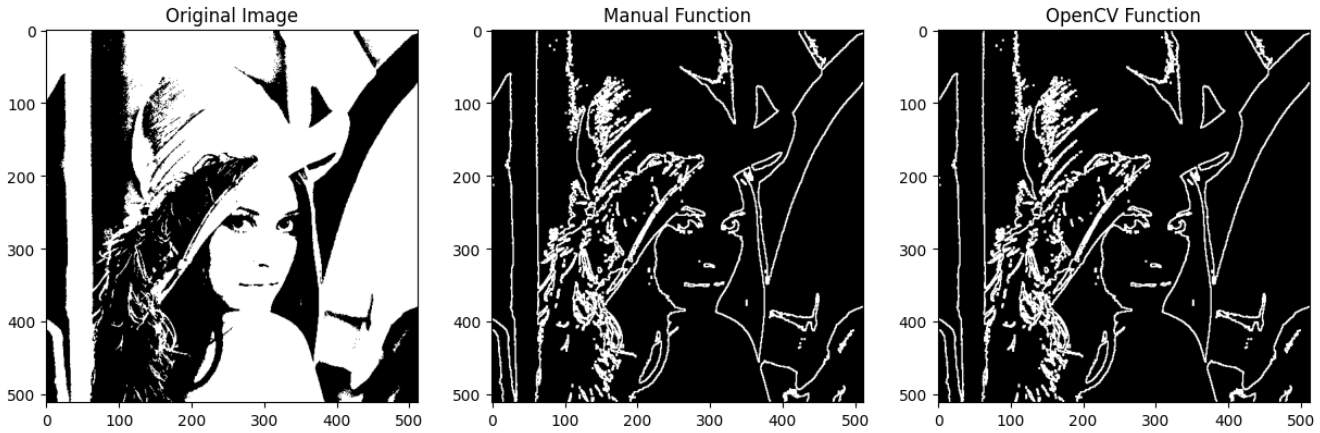

Figure 9: Boundary extraction on binary image with kernel size 3x3

As it can be seen from the result images, the boundaries of the object in the image are extracted. For comparision, the manual boundary extraction operator function does not show much differences from the built-in function of opencv cv.morphologyEx(image, cv.MORPH_GRADIENT, kernel).

3.6 Morphological Gradient

The morphological gradient operator is used to extract the boundaries of the object in the grayscale image. It is the difference between the dilation and the erosion of the input image. This idea is implemented in the following function:

def morphological_gradient(img, kernel)

- **Input:** img is the image in grayscale format, kernel is the structuring element.
- **Output:** The function returns the morphological gradient image.

This function is used to manually perform the morphological gradient operation on the grayscale image. From the above idea, the function first performs the dilation operation on the input image with the function dilation2(img, kernel) and the erosion operation on the input image with the function erosion2(img, kernel). Then, it finds the difference between the 2 functions and returns the morphological gradient grayscale image.

Result images:

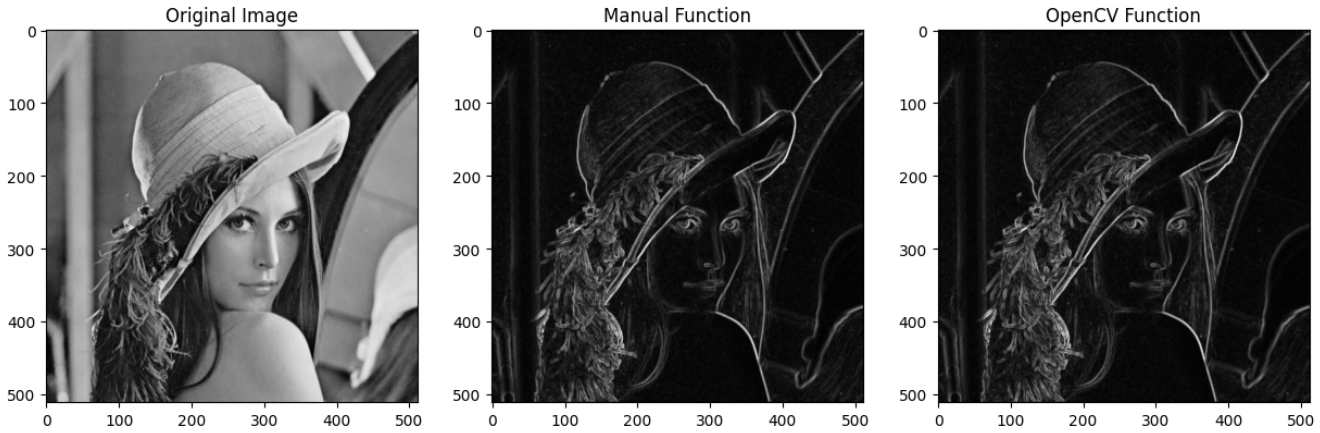

Figure 10: Morphological gradient on grayscale image with kernel size 3x3

As it can be seen from the result images, the boundaries of the object in the grayscale image are extracted. For comparision, the manual morphological gradient operator function seems showing the blurrer with less detail result than the built-in function of opencv cv.morphologyEx(image, cv.MORPH_GRADIENT, kernel).

3.7 Top-Hat

The top-hat operator is used to extract the small objects in the grayscale image. It is the difference between the input image and the opening of the input image. This idea is implemented in the following function:

def top_hat(img, kernel)

- **Input:** img is the image in grayscale format, kernel is the structuring element.
- **Output:** The function returns the top-hat image.

This function is used to manually perform the top-hat operation on the grayscale image. From the above idea, the function first performs the opening operation on the input image with the function opening2(img, kernel). Then, it finds the difference between the input image and the opened image and returns the top-hat grayscale image.

Result images:

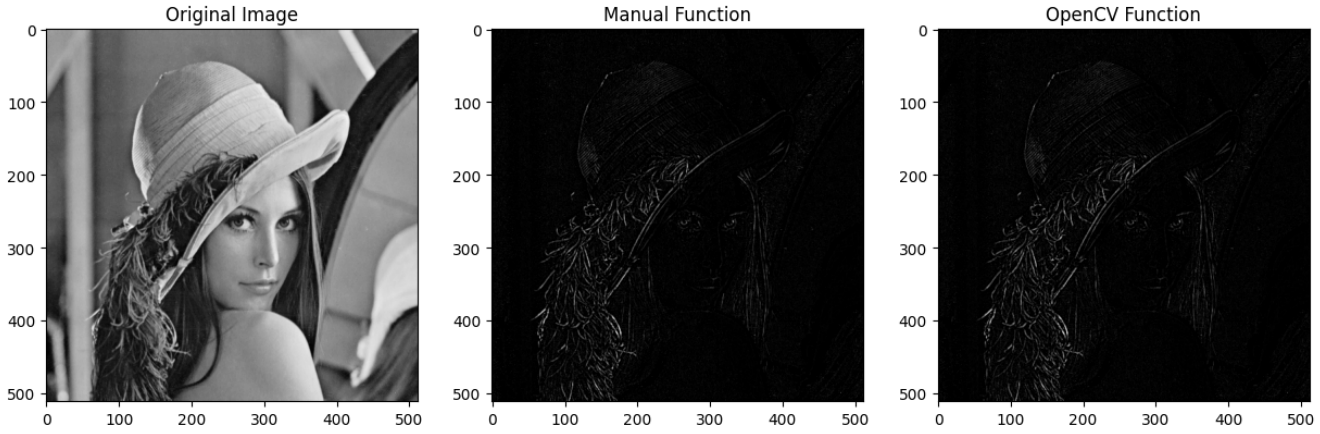

Figure 11: Top-hat on grayscale image with kernel size 3x3

As it can be seen from the result images, the small objects in the grayscale image are extracted. For comparision, the manual top-hat operator function does not show much differences from the built-in function of opencv cv.morphologyEx(image, cv.MORPH_TOPHAT, kernel).

3.8 Smoothing

The smoothing operator is used to remove the noise in the grayscale image. It is the combination of the opening and the closing of the input image. This idea is implemented in the following function:

def smooth(img, kernel)

- **Input:** img is the image in grayscale format, kernel is the structuring element.
- **Output:** The function returns the smoothed image.

This function is used to manually perform the smoothing operation on the grayscale image. From the above idea, the function first performs the opening operation on the input image with the function opening2(img, kernel) and then calls the closing operation on the opened image with the function closing2(img, kernel). Finally, the function returns the smoothed grayscale image.

Result images:

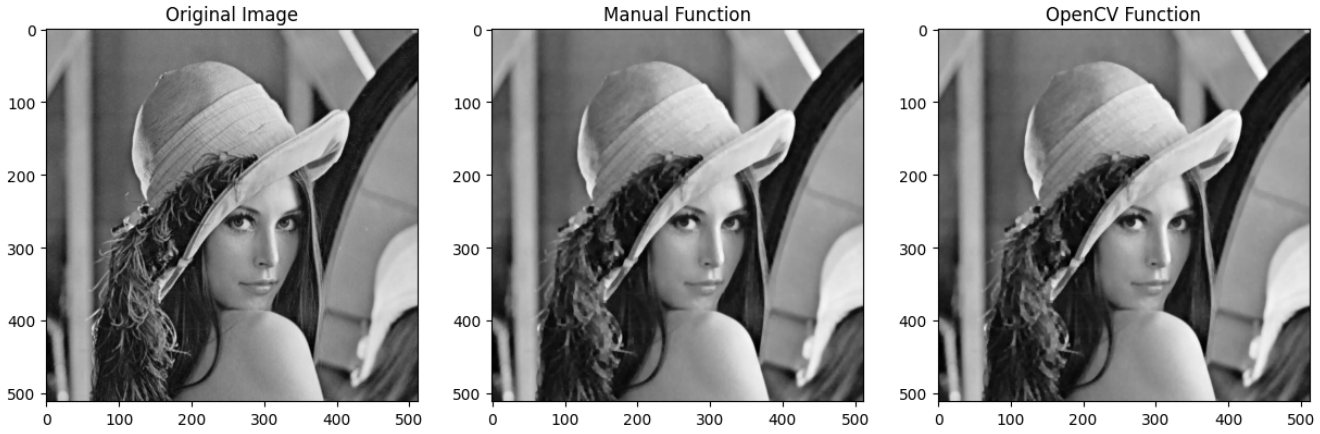

Figure 12: Smoothing on grayscale image with kernel size 3x3

As it can be seen from the result images, the noise in the grayscale image is removed. For comparision, the manual smoothing operator function does not show much differences from the built-in function of opencv cv.morphologyEx(image, cv.MORPH_CLOSE, kernel).

4 References

Practice #01 sample code published on moodle.

Slides of theory lecture provided by Prof. Lý Quốc Ngọc.

Ideas of dilation and erosion operators

Ideas of opening and closing operators

Implementation of morphological operators with opencv in both binary and grayscale images

Implementation of morphological operators with opencv