

Report : Morphological Operators

Algorithm:

This program is a collection of different morphological operations such as Erosion, Dilation, Opening, Closing, and Boundary that can be applied on an input image.

main(): This is the entry point to the program. Given an **input file**, **action**, **kernel_size**, this function reads an input image from a file and performs a specified image processing operation(specified by action variable) on it. The program also displays the statistics, in terms of pixel values and count of each pixel value, on the input and output images, displays the output image, and saves the output image to a file.

dilation(img, kernel_size): This function is used to dilate the image and fill the gaps in the image. This function follows the following steps to achieve this

- It initializes a kernel defined by the kernel_size, and pads the image appropriately
- It compares each kernel sized matrix from the image with the kernel. The traversal is done from left to right and top to bottom of the image
- Each time, it checks if any(at least one pair) corresponding pixels in these matrices(size : kernel_size X kernel_size) are foreground(255 pixel value), then this particular pixel in the original image is marked as foreground(255).
- Finally once the whole image is traversed, the dilated image is returned.

erosion(img, kernel_size): This function is used to erode the image, ie make it thinner. This function follows the following steps to achieve this

- It initializes a kernel defined by the kernel_size, and pads the image appropriately
- It compares each kernel sized matrix from the image with the kernel. The traversal is done from left to right and top to bottom of the image
- Each time, it checks if **all** corresponding pixels in the kernel matrices(size : kernel_size X kernel_size) are matching, then this particular pixel in the original image is marked as foreground(255).
- Finally once the whole image is traversed, the eroded image is returned.

opening(img, kernel_size): This function is used to open the image, which it in turn removes bridges or narrow regions in the image. This function follows the following steps to achieve this

- It calls the erosion function to erode the image
- It calls the dilation function to dilate the eroded image.

closing(img, kernel_size): This function is used to close the image, which in turn fills in the gaps or cracks in the image. This function follows the following steps to achieve this

- It calls the dilation function to dilate the image
- It calls the erosion function to erode the dilated image.

boundary(img, kernel_size): This function gets the boundary of the image. This function follows the following steps to achieve this

- It calls the erosion function to erode the image
- It takes the difference between the original image and the eroded image to get the boundary.

Results and Analysis:

The morphological operations implemented were able to successfully dilate, erode, open, close and get the boundary of the images. Experimenting with different filter_sizes was helpful in gaining insights on how these operations respond to different structures of the filter, and how it changes the output image.









Different experiments were done using filter sizes of 3x3, 5x5, 7x7. But the best output was obtained on a kernel size of 3x3, which is given below. (Output for the other filters are attached in the submission inside k5 and k7 folders).



As the kernel sizes increased to 5x5 or 7x7 the images were getting too dilated and losing the actual shape, this is expected as the it now considers a bigger set of neighbors (5x5 , or 7X7 surrounding pixels), and as a consequence more pixels are converted to white(255).

As the kernel sizes increased to 5x5 or 7x7 the images were getting too eroded and becoming closer to a dotted shape as well, this is also expected behavior as the it now considers a bigger set of neighbors (5x5 , or 7X7 surrounding pixels) and expects the kernel to be a subset of these, which restricts the amount of white pixels(255) in the output image.

Because of how dilation and erosion function responded to higher kernel_size, all the other functions(opening, closing, boundary) also reflected that, and the boundary was more filled in with white pixels and not as clear as the boundary obtained with 3x3.

output obtained on a kernel size of 3x3 :

kernel size	operation	gun.bmp	palm.bmp
3x3	dilation		
	erosion		
	opening		
	closing		

	boundary		
--	----------	--	---