Computer Vision HW#4

B01902040 資訊四 鍾毓安

**Task Description**

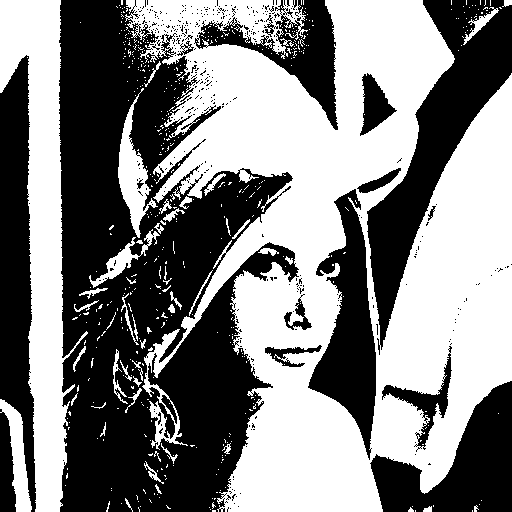
Implement binary morphological dilation, erosion, opening, closing, and hit-and-miss transform on a binary image. For the first four tasks, use the octagonal 3-5-5-5-3 kernel; for hit-and-miss transform, use the two “L” shaped kernels (same as ppt).

**Language & Tool**

* Python + Numpy
* OpenCV (for reading and writing image only)

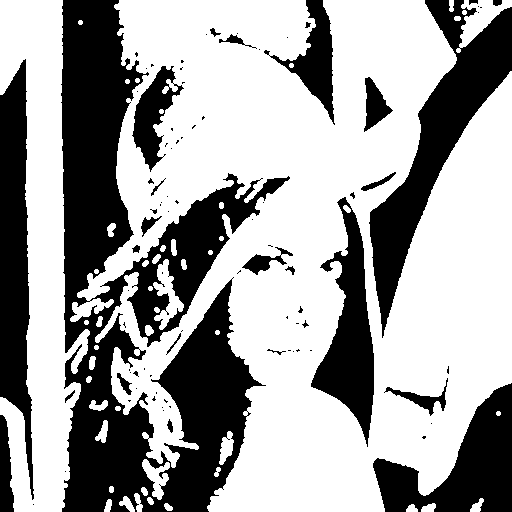
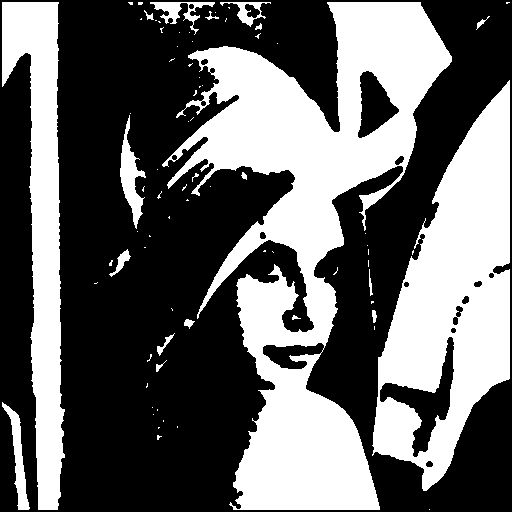
**Work Flow & Results**

We first read in lena.bmp using function cv2.imread() and store the pixel values in a 2D array `img`. Before performing any morphological task, img is first binarized (threshold at 128) and stored in a 2D array `img\_bin`. The original image and the binarized one are as follows:

lena.bmp lena.bin.bmp

Then, we follow the definition of dilation and erosion and complete those two functions. Details of implementation are stated in the source code. Once the two APIs are available, closing and opening should be easy: for closing, just simply call erosion(dilation(img\_bin, octagon), octagon); and dilation(erosion(img\_bin, octagon), octagon) for opening. The four output images are as follows:

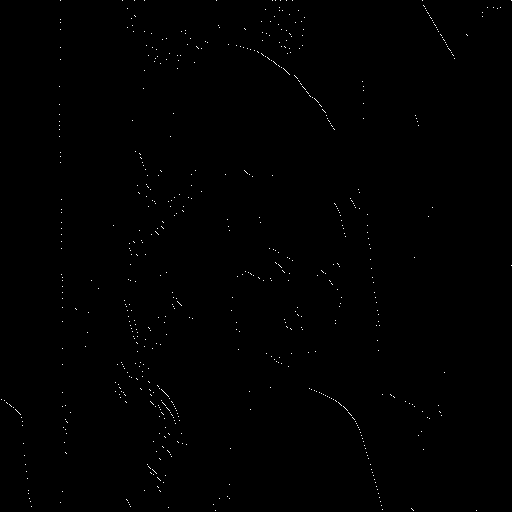
 

lena.bin.dil.bmp lena.bin.ero.bmp

lena.bin.close.bmp lena.bin.open.bmp

Hit-and-miss transform should also be easy since my dilation() and erosion() support any kinds of kernels, just call (erosion(img\_bin, J\_kernel) + erosion(img\_comp, K\_kernel)) / 2, where img\_comp is the complement of img\_bin, and the “add and divide by 2” implements the tricks of intersection . The image after hit-and-miss transform is on the next page.



lena.bin.ham.bmp

To reproduce the result, execute the following command:

>> python ./main.py