Computer Vision HW4, Binary Morphology Report

tags: **NTU CS** **Computer Vision** **Writeup** **Report**

NTU CSIE, R08922024, Alfons Hwu
Prequisites and env as the following

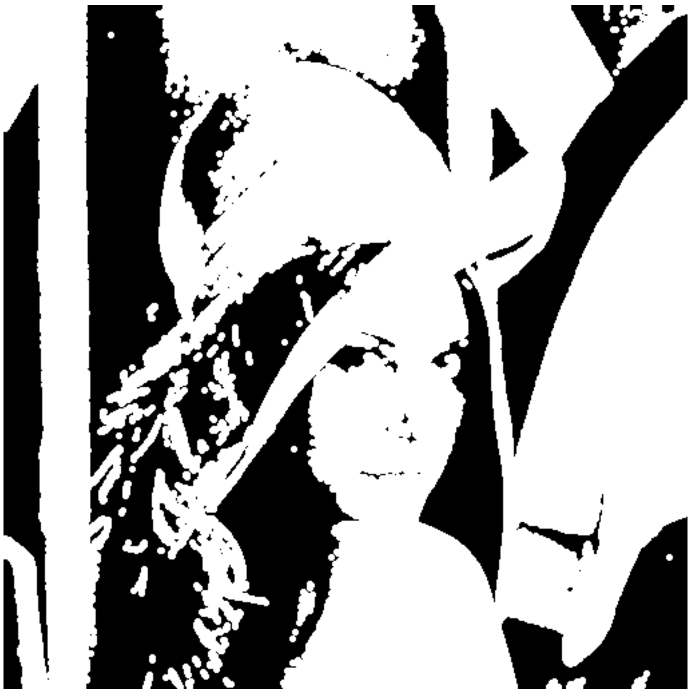
```
Ubuntu WSL for windows with jupyter notebook
Python3.6.7
OpenCV for image IO
Matplotlib for displayig image
```

a, dilation

Search the neighbor pixels according to the shifting value of filter kernel, and extend them.

$$A\ dilate\ B = \{c \in E^N |\ c = a + b\ \exists a \in A \wedge \exists b \in B\}$$

```
def dilation(a, b):
    ra, ca = a.shape
    res = np.zeros(a.shape, dtype = 'int32')

    for ai in range(ra):
        for aj in range(ca):
            if a[ai, aj] == 0xff:

                # assign original image position
                res[ai, aj] = 0xff
                for b_each in b:
                    bi, bj = b_each
                    if  ai + bi >= 0 and ai + bi < ra \
                    and aj + bj >= 0 and aj + bj < ca:
                        # extend the value
                        res[ai + bi, aj + bj] = 0xff

    return res
```



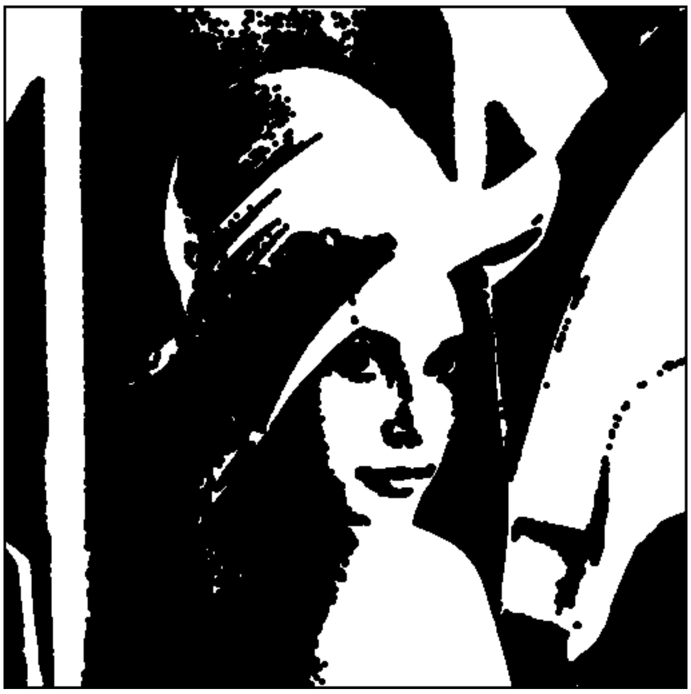
Time complexity, kernel size K: $O(MNK)$

b, erosion

Search the neighbor pixels of target pixel according to the shifting value of filter kernel, and delete such target pixel if an element does not lie in the original structure.

$$A\ \Theta\ B = \{c \in E^N |\ c = a + b\ \forall a \in A \wedge \forall a + b \in A\}$$

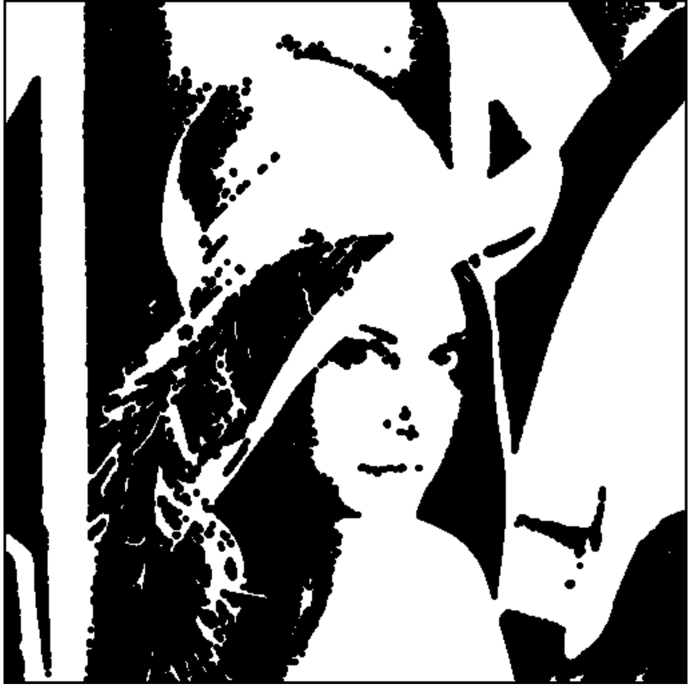
```
def erosion(a, b):
    ra, ca = a.shape # original image
    res = np.zeros(a.shape, dtype = 'int32')

    for ai in range(ra):
        for aj in range(ca):
            if a[ai, aj] > 0:

                ok = 1
                for b_each in b:
                    bi, bj = b_each
                    if ai + bi >= ra or aj + bj >= ca \
                    or ai + bi <  0  or aj + bj <  0  \
                    or a[ai + bi, aj + bj] == 0:
                        ok = 0
                        break

                if ok == 1:
                    res[ai, aj] = 255

    return res
```



Time complexity, kernel size K: $O(MNK)$

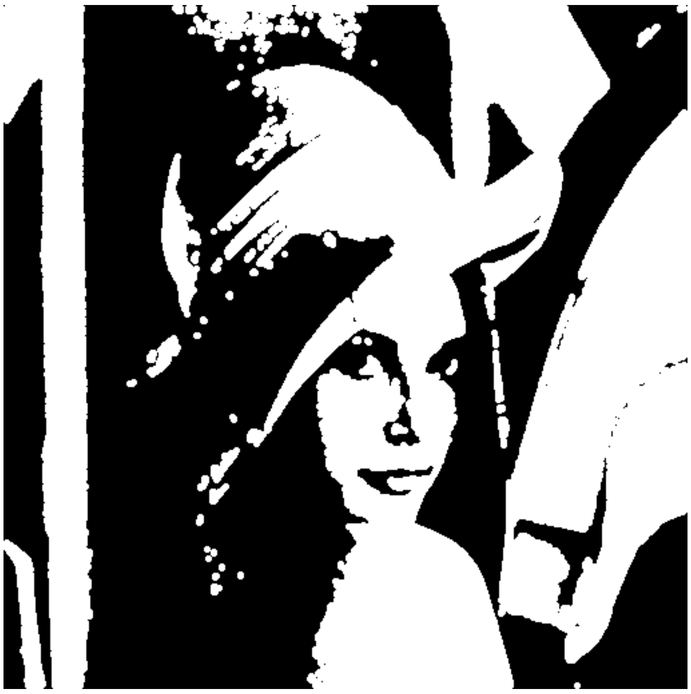
c, closing

```
def closing(a, b):
    return erosion(dilation(a, b), b)
```


Time complexity, kernel size K: $O(MNK)$

d, opening

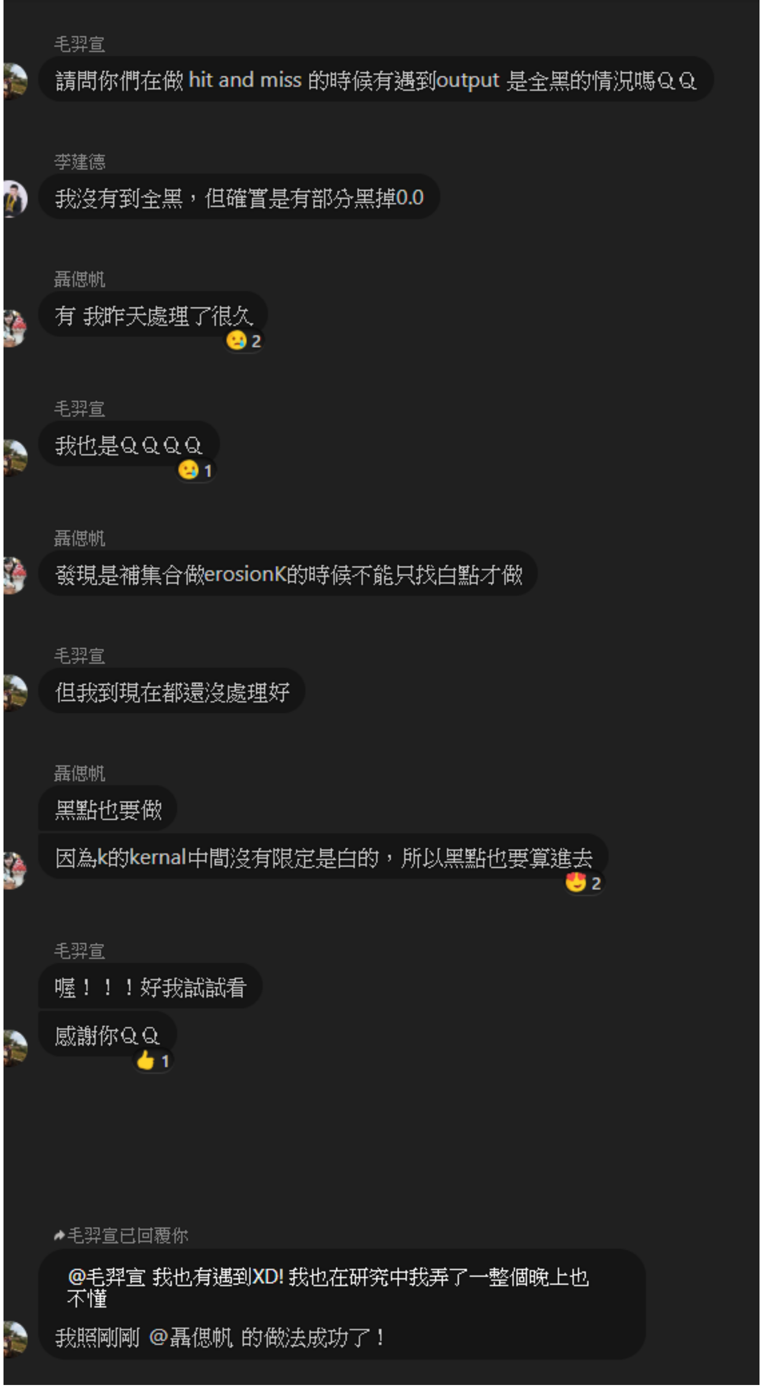
```
def opening(a, b):
    return dilation(erosion(a, b), b)
```


Time complexity, kernel size K: $O(MNK)$

e, hit and miss transformation

Use -a + 255 to complement an image since numpy can process multiple pixels at the same time.

NOTE: We should do the black and white pixel for the complement images at the same time since we have the following discussion and it may be a pitfall for this assignment.


毛羿宣

請問你們在做 hit and miss 的時候有遇到output 是全黑的情況嗎Q Q

李建德

我沒有到全黑，但確實是有部分黑掉0.0

聶偲帆

有 我昨天處理了很久

😢 2

毛羿宣

我也是Q Q Q Q

😢 1

聶偲帆

發現是補集合做erosionK的時候不能只找白點才做

毛羿宣

但我到現在都還沒處理好

聶偲帆

黑點也要做

因為k的kernal中間沒有限定是白的，所以黑點也要算進去

😍 2

毛羿宣

喔！！！好我試試看

感謝你Q Q

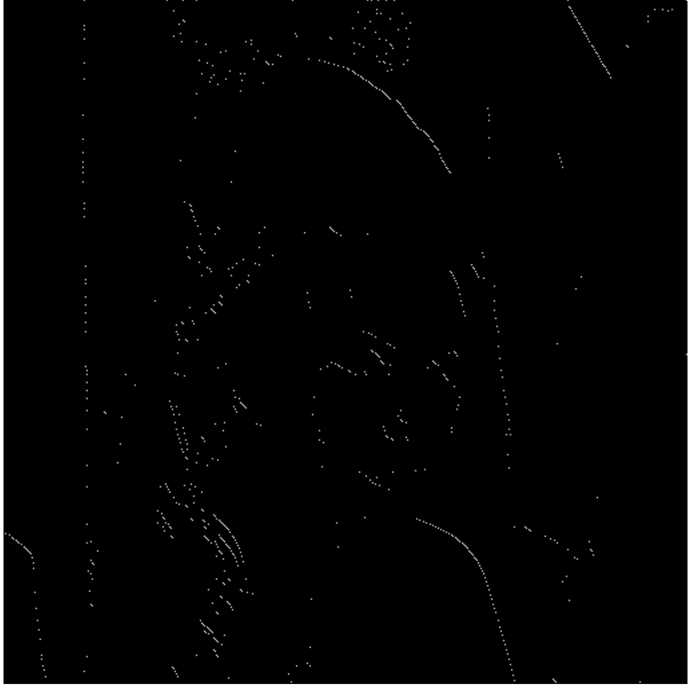
👍 1

毛羿宣已回覆你

@毛羿宣 我也有遇到XD! 我也在研究中我弄了一整個晚上也不懂

我照剛剛 @聶偲帆 的做法成功了！

```
def erosion_2(a, b):
    ra, ca = a.shape # original image
    res = np.zeros(a.shape, dtype = 'int32')

    for ai in range(ra):
        for aj in range(ca):
            # assign original image position erode the pixel or not
            res[ai, aj] = 0xff
            ok = 1
            for b_each in b:
                bi, bj = b_each
                if ai + bi >= ra or aj + bj >= ca \
                or ai + bi <  0  or aj + bj <  0 \
                or a[ai + bi, aj + bj] != 0xff:
                    ok = 0
                    break

            if ok == 0:
                # erode the pixel
                res[ai, aj] = 0

    return res

def hit_and_miss(a, j, k):
    def hit_and_miss(a, j, k):
    return (((erosion(a, j) + erosion_2((-a + 0xff), k)) // 2) == 0xff) * 0xff
```


Time complexity, kernel size K: $O(MNK)$