Computer Vision I (922 U0610) - Homework 4

Author: alanhc

ID: r10944007

Date: 10/21

README

1. create env: `conda env create -f environment.yml`
2. enter env: `conda activate ntu-cv`
3. run jupyter `jupyter notebook`

Write programs which do binary morphology on a binary image:

- (a) Dilation
- (b) Erosion
- (c) Opening
- (d) Closing
- (e) Hit-and-miss transform

In [1]:
```
from PIL import Image
import numpy as np

# Todo: 讀檔，確定影像大小
img = Image.open("input/lena.bmp")
img = np.array(img)

h, w = img.shape
for y in range(h):
    for x in range(w):
        img[y][x] = (255 if img[y][x]>=128 else 0)


print("image shape:", img.shape)
show = Image.fromarray(img).resize((256,256))
show
```

```
image shape: (512, 512)
```

Out[1]:

In [2]:
```
import matplotlib.pyplot as plt
kernel_35553 = np.array([
    [0,1,1,1,0],
    [1,1,1,1,1],
    [1,1,1,1,1],
    [1,1,1,1,1],
    [0,1,1,1,0],
])
```

Dilation

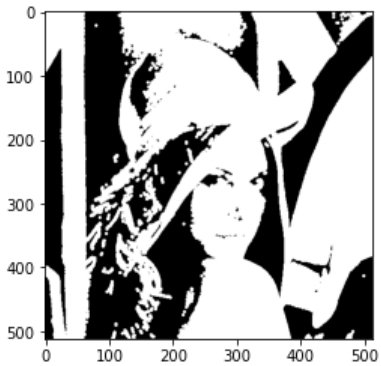
In [3]:
```
# Todo: Dilation
# Algorithm:
## 1.檢查中心有值
## 2. 填值

def dilation(src, ans, kernel):
    def fill_color(A, y,x,kernel,h,w):

        half_k_y = kernel.shape[0]//2
        half_k_x = kernel.shape[1]//2
```

```
        for ky in range(kernel.shape[0]):
            for kx in range(kernel.shape[1]):
                # kernel[ky][kx]==1
                if (kernel[ky][kx]):
                    now_y, now_x = y+ky-half_k_y, x+kx-half_k_x
                    if (now_y<0 or now_x<0 or now_y>h-1 or now_x>w-1):
                        # 邊界條件要跳過
                        continue
                    # 2. 填值
                    A[now_y][now_x] = 255
    for y in range(h):
        for x in range(w):
            # 1.檢查中心有值
            if (src[y][x]):
                fill_color(ans, y,x,kernel,h,w)
    return ans
img_dilation = dilation(img, np.zeros((h,w)),kernel_35553)
plt.imshow(img_dilation, cmap="gray")
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x7f9ae2f85760>
```



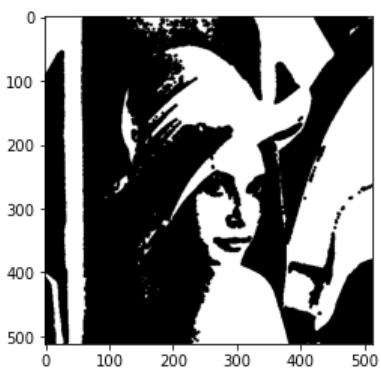
Erosion

In [4]:
```
# Todo: Erosion
# Algorithm:
## 1. 檢查是否kernel與圖形有碰撞
## 2. isCollision(y,x) 差集 img(y,x) (有碰撞就丟掉)
def erosion(img, ans, kernel):
    def isCollision(A, y,x,kernel,h,w):
        half_k_y = kernel.shape[0]//2
        half_k_x = kernel.shape[1]//2

        for ky in range(kernel.shape[0]):
            for kx in range(kernel.shape[1]):

                if (kernel[ky][kx]==1):
                    now_y, now_x = y+ky-half_k_y, x+kx-half_k_x
                    if (now_y<0 or now_x<0 or now_y>h-1 or now_x>w-1):
                        continue
                    else:
                        # 2. isCollision(y,x) 差集 img(y,x) (有碰撞就丟掉)
                        if (A[now_y][now_x]==0):
                            return 0
        # 沒有碰撞
        return 255
    for y in range(h):
        for x in range(w):
            # 1. 檢查是否kernel與圖形有碰撞
                ans[y][x]=isCollision(img, y,x,kernel,h,w)
    return ans
img_erosion = erosion(img, np.zeros((h,w)), kernel_35553)
plt.imshow(img_erosion, cmap="gray")
```

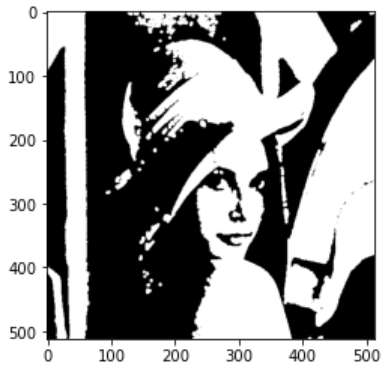
Out[4]:
```
<matplotlib.image.AxesImage at 0x7f9ac00dcd00>
```



Opening

In [5]:
```
# Todo: Opening
# Algorithm:
## 1. 先erosion
## 2. 後dilation
img_opening = erosion(img, np.zeros((h,w)),kernel_35553)
img_opening = dilation(img_opening, np.zeros((h,w)),kernel_35553)
plt.imshow(img_opening, cmap="gray")
```

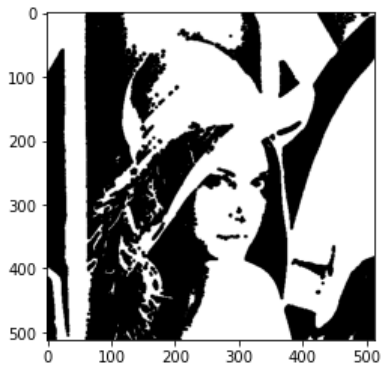
Out[5]:
```
<matplotlib.image.AxesImage at 0x7f9af03e7190>
```



Closing

In [6]:
```
# Todo: Closing
# Algorithm:
## 1. 先dilation
## 2. 後erosion
img_closing = dilation(img, np.zeros((h,w)),kernel_35553)
img_closing = erosion(img_closing, np.zeros((h,w)),kernel_35553)
plt.imshow(img_closing, cmap="gray")
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x7f9ae34816d0>
```



Hit-and-miss transform

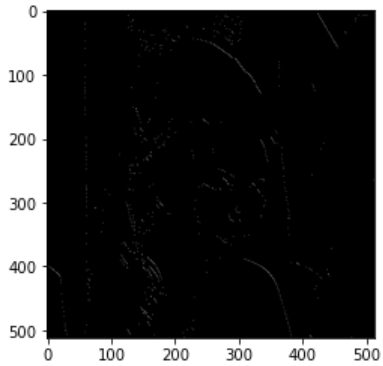
In [7]:
```
# Todo: Opening
# Algorithm:
## 1. 定義J, K kernel
## 2. 計算原圖的complement (A_C)
## 3. left = erosion(A)使用J kernel
## 4. right = erosion(A_C)使用K kernel
## 5. 聯集(left,right)

# 1. 定義J, K kernel
J = np.array([
    [0,0,0],
    [1,1,0],
    [0,1,0],
])
K = np.array([
    [0,1,1],
    [0,0,1],
    [0,0,0],
])
# 2. 計算原圖的complement (A_C)
A_C = np.full((h,w),255) - img.copy()
A = img.copy()
# 3. left = erosion(A)使用J kernel
left = erosion(A, np.zeros((h,w)), J)
# 4. right = erosion(A_C)使用K kernel
```

```
right = erosion(A_C, np.zeros((h,w)),K)

img_hit = np.zeros((h,w))
# 5. 交集(left,right)
for y in range(h):
    for x in range(w):
        img_hit[y][x] = 255 if(left[y][x] and right[y][x]) else 0
        #if (left[y][x] and right[y][x]):
        #    print(y,x)
plt.imshow(img_hit, cmap="gray")
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x7f9ad0113d60>
```



Ref

- https://youtu.be/7-FZBgrW4RE
- https://en.wikipedia.org/wiki/Opening_(morphology)
- https://en.wikipedia.org/wiki/Closing_(morphology)
- textbook

In [8]:
```
imgs = [img_dilation, img_erosion, img_opening, img_closing, img_hit]
names = ["a_img_dilation", "b_img_erosion", "c_img_opening", "d_img_closing", "e_img_hit"]
for i in range(len(names)):
    im = Image.fromarray(imgs[i]).convert('RGB').save("output/"+names[i]+".png")
```