Labsheet 12
Morphological Image Pocessing

Ragja Palakkadavath
(SC18M003)

Sub: Image and Video Processing Lab
Department: Mathematics(MLC)
Date of Lab sheet: November 1, 2018
Date of Submission: December 1, 2018

---

1. Dilation and Erosion

Perform Dilation and Erosion operations on the given image A using the two kernels B1 and B2 given.

$$\text{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{B1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\text{B2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Aim

To perform erosion and dilation morphological operation on the given image using the two given kernels.

Theory/Discussion

Dilation

$$D(A, B) = A \oplus B = \{x \mid (\hat{B}_x \cap A) \neq \Phi\}$$

where Φ represents empty set. Alternatively,

$$D(A, B) = A \oplus B = \bigcup_{b \in B} A_b = \bigcup_{a \in A} B_a$$

Erosion

$$E(A, B) = A \ominus B = \{x \mid B_x \subseteq A\}$$

Alternatively,

$$E(A, B) = A \ominus B = \bigcap_{b \in B} A_b$$

Algorithm

- Find reflection $\hat{B}$ of set B by flipping it about its origin.If B is central symmetric, this step makes no difference;
- Translate (shift, slide) $\hat{B}$ by displacement x over A;
- $A \oplus B$ is the set of all displacements x's that keep the intersect (overlap) of $\hat{B}_x$ and A non-empty (keep them "in touch").
- Dilation of a binary image shape A by B expands the shape by half of the size of B.
- Translate (or slide) B by x over A
- $A \ominus B$ is the set of all translations x's that will keep the translated version of B to be entirely contained in A.
- Erosion of a binary image shape A by B shrinks the shape by half of the size of B.

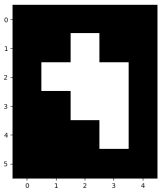
Code

```
# -*- coding: utf-8 -*-
"""
Created on Sat Dec  1 15:28:43 2018

@author: IIST
"""
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
A=[[0, 0, 0, 0, 0],
   [0, 0, 1, 0, 0],
   [0, 1, 1, 1, 0],
   [0, 0, 1, 1, 0],
   [0, 0, 0, 1, 0],
   [0, 0, 0, 0, 0]]

A_org=np.asarray(A)
A=np.flipud(A_org)

B1=[[0, 0, 0],
    [0, 0, 0],
    [1, 1, 0]]
#B1=[0, 0, 0,0, 0, 0,1, 1, 0]

B2=[[0, 1, 0],
    [1, 1, 1],
    [0, 1, 0]]
B1=np.asarray(B1)
B2=np.asarray(B2)

B=np.flipud(B1)
index=np.where(B)
l1=np.ndarray.tolist(index[0])
l2=np.ndarray.tolist(index[1])
print(l1,l2)
A_new=np.zeros((A.shape))
for k in range(len(l1)):
```

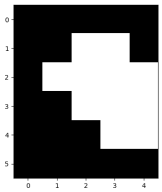
```
    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            x=l1[k]
            y=l2[k]
            if(i+x<A.shape[0] and j+y<A.shape[1]):
                if(A[i,j]==1):
                    A_new[i+x,j+y]=1
A_new=np.flipud(A_new)
plt.imshow(A_org,cmap='gray')
plt.figure()
plt.imshow(A_new,cmap='gray')

# -*- coding: utf-8 -*-
"""
Created on Sat Dec  1 15:49:13 2018

@author: IIST
"""
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
A=[[0, 0, 0, 0, 0],
   [0, 0, 1, 0, 0],
   [0, 1, 1, 1, 0],
   [0, 0, 1, 1, 0],
   [0, 0, 0, 1, 0],
   [0, 0, 0, 0, 0]]

A_org=np.asarray(A)
A=np.flipud(A_org)

B1=[[0, 0, 0],
    [0, 0, 0],
    [1, 1, 0]]
#B1=[0, 0, 0,0, 0, 0,1, 1, 0]

B2=[[0, 1, 0],
    [1, 1, 1],
    [0, 1, 0]]
B1=np.asarray(B1)
B2=np.asarray(B2)

B=np.flipud(B2)
index=np.where(B)
l1=np.ndarray.tolist(index[0])
l2=np.ndarray.tolist(index[1])
print(l1,l2)
A_new=np.zeros((A.shape))
for i in range(A.shape[0]):
    for j in range(A.shape[1]):
        count=0
        for k in range(len(l1)):
            x=l1[k]
            y=l2[k]
            if(i+x<A.shape[0] and j+y<A.shape[1]):
                if(A[i+x,j+y]==1):
                    count=count+1
        print(count)
```

```
            if(count==len(l1)):
                A_new[i,j]=1
A_new=np.flipud(A_new)
plt.imshow(A_org,cmap='gray')
plt.figure()
plt.imshow(A_new,cmap='gray')
```

Result

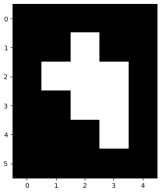


(a) Given Image

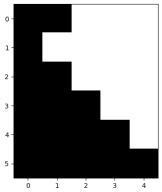


(b) Dilated Image

Figure 1: Dilation with B1

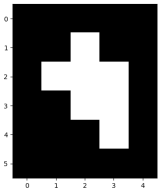


(a) Given Image

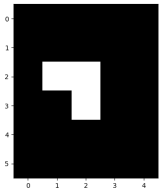


(b) Dilated Image

Figure 2: Dilation with B2

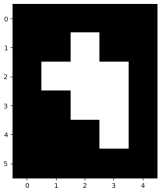


(a) Given Image

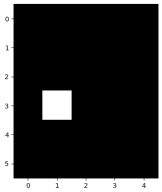


(b) Eroded Image

Figure 3: Erosion with B1



(a) Given Image



(b) Eroded Image

Figure 4: Erosion with B2

Inference

Erosion and Dilation operations were performed using both the kernels, B1 and B2.

2. Opening and Closing

Perform Opening and Closing operations on the given image A using the given kernel B.

$$\mathrm{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Aim

To perform opening and closing on a binary image with the given kernel.

Theory/Discussion

Opening

$$O(A,B) = D(E(A,B),B), \qquad A \circ B = (A \ominus B) \oplus B$$

The opening of A and B is the dilation of the erosion of A by B.

Closing

$$C(A,B) = E(D(A,B),B), \qquad A \bullet B = (A \oplus B) \ominus B$$

The closing of A and B is the erosion of the dilation of A by B.

Algorithm

- Dilation and erosion are not a pair of opposite operations in the sense that their effects do not cancel each other.
- The erosion carried out first eliminates small shapes (assumed to be noise) as well as shrinking the object shape, while the following dilation grows the object back (but not the noise).
- The effect of dilation and erosion do not cancel each other.
- The dilation carried out first eliminates small holes inside the object shape (assumed to be noise) as well as expanding the object shape, while the following erosion shrinks the object back (but not the noise).

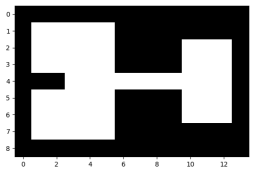
Code

```
# -*- coding: utf-8 -*-
"""
Created on Sat Dec  1 17:45:10 2018

@author: IIST
"""

import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

A = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
     [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
```

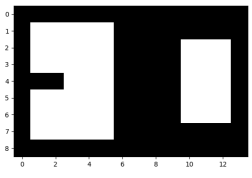
```
      [0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],
      [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
      [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
      [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

B1= [[1, 1, 1,], [ 1, 1, 1,], [ 1, 1, 1,]]

A_org=np.asarray(A)
A=np.flipud(A_org)
B=np.flipud(B1)
index=np.where(B)
l1=np.ndarray.tolist(index[0])
l2=np.ndarray.tolist(index[1])
print(l1,l2)
A_new=np.zeros((A.shape))
for i in range(A.shape[0]):
    for j in range(A.shape[1]):
        count=0
        for k in range(len(l1)):
            x=l1[k]
            y=l2[k]
            if(i+x<A.shape[0] and j+y<A.shape[1]):
                if(A[i+x,j+y]==1):
                    count=count+1
        if(count==len(l1)):
            A_new[i,j]=1
#A_new=np.flipud(A_new)
A_new=np.flipud(A_new)
A=np.flipud(A_new)
#plt.figure()
#plt.imshow(A_new,cmap='gray')
A_new=np.zeros((A.shape))
for k in range(len(l1)):
    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            x=l1[k]
            y=l2[k]
            if(i+x<A.shape[0] and j+y<A.shape[1]):
                if(A[i,j]==1):
                    A_new[i+x,j+y]=1
A_new=np.flipud(A_new)
plt.imshow(A_org,cmap='gray')
plt.figure()
plt.imshow(A_new,cmap='gray')

# -*- coding: utf-8 -*-
"""
Created on Sat Dec  1 17:43:19 2018

@author: IIST
"""
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

A = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
      [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
      [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
      [0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],
      [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
      [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
      [0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

B1= [[1, 1, 1,],[ 1, 1, 1,],[ 1, 1, 1,]]

A_org=np.asarray(A)
A=np.flipud(A_org)
B=np.flipud(B1)
index=np.where(B)
l1=np.ndarray.tolist(index[0])
l2=np.ndarray.tolist(index[1])
print(l1,l2)
A_new=np.zeros((A.shape))
for k in range(len(l1)):
    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            x=l1[k]
            y=l2[k]
            if(i+x<A.shape[0] and j+y<A.shape[1]):
                if(A[i,j]==1):
                    A_new[i+x,j+y]=1
A_new=np.flipud(A_new)
A_new2=np.asarray(A_new)
A=np.flipud(A_new2)

index=np.where(B)
l1=np.ndarray.tolist(index[0])
l2=np.ndarray.tolist(index[1])
print(l1,l2)
A_new=np.zeros((A.shape))
for i in range(A.shape[0]):
    for j in range(A.shape[1]):
        count=0
        for k in range(len(l1)):
            x=l1[k]
            y=l2[k]
            if(i+x<A.shape[0] and j+y<A.shape[1]):
                if(A[i+x,j+y]==1):
                    count=count+1
        print(count)
        if(count==len(l1)):
            A_new[i,j]=1
A_new=np.flipud(A_new)
plt.imshow(A_org,cmap='gray')
plt.figure()
plt.imshow(A_new,cmap='gray')
```

Result

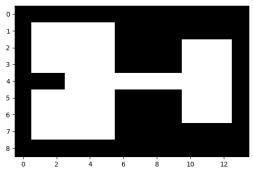


(a) Original Image

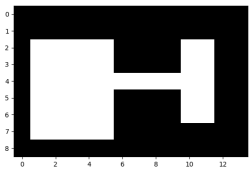


(b) After Opening

Figure 5: Opening



(a) Original Image



(b) After Closing

Figure 6: Closing

Inference

Opening and Closing morphological operations were performed on the given image and using the given kernel.

3. Hit or Miss Operation

Perform Hit or Miss operation on the given image A to find the given the object X.

$$B=\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Aim

To convert a gray image to a binary image by finding the threshold value using Otsu's global thresholding algorithm

Theory/Discussion

The purpose of the hit-and-miss algorithm is to detect certain patterns in a binary image, using a structuring element containing 1's, 0's and blanks for don't cares. The structuring element is used as a template that slides over the binary image (similar to a convolution kernel) and the pixel corresponding to the center of the template is set to 1 if the template matches the image, or 0 otherwise. For example, the following four structuring elements can detect corners of four different orientations. The final result is an OR of four images each obtained by one of the four structuring elements.

Algorithm

1. Hit-and-miss algorithm can be used to thin and skeletonize a shape in a binary image.
2. This is an iterative process containing repeated steps to thin the shape by hit-and-miss method.
3. In each iteration, some different structuring elements are used to identify the edge pixels to be removed, followed by the actual removal of them:

$$I'_k = O_i(I_k), \quad I_{k+1} = I_k - I'_k, \quad k = k + 1$$

where I_k is the image after k iterations, $O_i(I_k)$ is the application of a structuring element O_i to I_k, and $I_k - I'_k$ is a set subtraction $A - B = A \cap \bar{B}$. After each such iteration, the elements are rotated to identify and remove other edge pixels in different orientations.

4. The process continues until no further edge pixels can be identified, and the shape has been thinned to a skeleton.

Code

```
# -*- coding: utf-8 -*-
"""
Created on Sat Dec  1 21:39:02 2018

@author: IIST
"""

A = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
     [0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
     [0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

B = [0,0,0,0,0,0,1,1,1,0,0,1,1,1,0,0,1,1,1,0,0,0,0,0,0]

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt



def W_calculation(x,y):
    k=list()
    x = x+2
    y = y+2
    for i in range(x-2,x+3):
        for j in range(y-2,y+3):
            k.append(a1[i][j])
    return k
A=np.array(A)
A_new = np.copy(A)

a1 = np.pad(A_new, ((1,1),(1,1)), 'constant')
a1 = np.pad(a1, ((1,1),(1,1)), 'constant')

for i in range(A_new.shape[0]):
    for j in range(A_new.shape[1]):
```

```
            neighbors = W_calculation(i,j)
            if (neighbors == B):
                rx = i
                ry = j


l1 = np.arange(rx-2,rx+3)
l2 = np.arange(ry-2,ry+3)

for i in range(A_new.shape[0]):
    for j in range(A_new.shape[1]):
        if not((i in l1) and (j in l2)):
            A_new[i][j] = 0

plt.imshow(Image.fromarray(A*255))
plt.figure()
plt.imshow(Image.fromarray(A_new*255))
```

Result

(a) Original Image

(b) After Hit and Miss

Figure 7: Hit and Miss

Inference

Hit and Miss was performed and the given X pattern was identified inside the given image A.