Computer Vision
Homework 5 Report

林義聖
B03902048

November 10, 2016

1 Introduction

In this homework assignment, we're going to do *Mathematical Morphology — Gray Scaled Morphology* and try to implement four algorithm: erosion, dilation, opening, and closing.

I use *Python* as my programming language and *Pillow* as my image library. It is a fork of *PIL*, which is the original image library of *Python*. And I use *Pillow* for reading input image, transfering image data into *List* of *Python*, and saving the processed data as a gray-scaled image.

Figure 1: original lena.bmp

2 Program Structure

There's only one program in my submission, and I use program parameters to decide which algorithm we want to apply on the input image. All the things done in my program is:

1. Import required library
2. Define kernel
3. Open input image
4. Decide which algorithm to apply to and process the image data
5. Create a gray-scale image and put the result into image
6. Save the output image

```
import sys
from PIL import Image

# get system arguments
...

# define the 3-5-5-5-3 kernel
kernel = [\
    (-1,-2),(0,-2),(1,-2),\
    (-2,-1),(-1,-1),(0,-1),(1,-1),(2,-1),\
    (-2,0),(-1,0),(0,0),(1,0),(2,0),\
    (-2,1),(-1,1),(0,1),(1,1),(2,1),\
    (-1,2),(0,2),(1,2)\
]

# the required 4 algorithms are implemented as functions
...

# main function
def main():
    # get input image
    try:
        in_img = Image.open(infilename)
    except Exception as e:
        print('Error:', str(e))
        exit(1)
    ...

    # transfer image data into Python list
    data_seq = list(in_img.getdata())

    # handle which algorithm we want to apply on image
    if option == 'erosioin':
      ...

    # save output image
```

```
37      out_img = Image.new('L', in_img.size)
38      out_img.putdata(out_data)
39      out_img.save(outfilename, 'bmp')
40
41  if __name__ == '__main__':
42      main()
```

3 Dilation $f \oplus k$

The definition of *Gray Scale Dilation* is,

$$f \oplus k = T\{U[f] \oplus U[k]\}$$

When we are implementing this algorithm, we use this definition,

$$(f \oplus k)(x) = \texttt{max}\{f(x-z) + k(z) \mid z \in K, x - z \in F\}$$

Therefore, to do dilation, I use following steps:

1. Loop through every pixels
2. For each pixel, get value of translated pixels and put into a *list*
3. Find the maximum value in the *list*
4. Assign that value to the current pixel

```
1  def dilation(data, hei, wid):
2      global kernel
3      result = [0] * len(data)
4      for y in range(hei):
5          for x in range(wid):
6              translated = []
7              for x_k, y_k in kernel:
8                  x_t = x + x_k
9                  y_t = y + y_k
10                 if 0 <= x_t < wid and 0 <= y_t < hei:
11                     translated.append(data[y_t * wid + x_t])
12                 else:
13                     pass
14             result[y * wid + x] = max(translated)
15     return result
```

Figure 2: dilated lena.bmp

4 Erosion $f \ominus k$

The definition of *Gray Scale Erosion* is,

$$f \ominus k = T\{U[f] \ominus U[k]\}$$

When we are implementing this algorithm, we use this definition,

$$(f \ominus k)(x) = \texttt{min}\{f(x+z) - k(z)\}$$

Therefore, to do erosion, I use following steps:

1. Loop through every pixels
2. For each pixel, get value of translated pixels and put into a *list*
3. Find the minimum value in the *list*
4. Assign that value to the current pixel

```
1 def erosion(data, hei, wid):
2     global kernel
3     result = [0] * len(data)
4     for y in range(hei):
5         for x in range(wid):
6             translated = []
7             for x_k, y_k in kernel:
8                 x_t = x + x_k
9                 y_t = y + y_k
```

```
                if 0 <= x_t < wid and 0 <= y_t < hei:
                    translated.append(data[y_t * wid + x_t])
                else:
                    pass
            result[y * wid + x] = min(translated)
    return result
```

Figure 3: eroded lena.bmp

5 Opening $f \circ k$

The *opening* on f by k is denoted by $f \circ k$. And it is defined by *dilation* and *erosion*, that is,

$$f \circ k = (f \ominus k) \oplus k$$

Hence, to do *opening*, I just apply *erosion* on image first, and then apply *dilation* on the result of previous step.

```
def opening(data, hei, wid):
    return dilation( erosion(data, hei, wid), hei, wid )
```

Figure 4: lena.bmp after opening

6 Closing $f \bullet k$

The *closing* on f by k is denoted by $f \bullet k$. And it is defined by *dilation* and *erosion*, that is,

$$f \bullet k = (f \oplus k) \ominus k$$

Hence, to do *closing*, I just apply *dilation* on image first, and then apply *erosion* on the result of previous step.

```
1 def closing(data, hei, wid):
2     return erosion( dilation(data, hei, wid), hei, wid )
```

Figure 5: lena.bmp after closing

7 How to Use

There's only one executable program in my submission, and its name is *gs-morphology.py*. To run this program, just type this command "`./gs-morphology.py [option] [input image] [output image]`". And the `[option]` could be: dilation, erosion, opening, closing.