Computer Vision
Homework 7 Report

林義聖
B03902048

December 5, 2016

Introduction

In this homework assignment, we're going to do *thinning* on image. 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, and transfering image data into *List* of *Python*.

Figure 1: original lena.bmp

Program Structure

There's only one program *thinning.py* in my submission. And its structure is as following:

1. Import required library (i.g PIL)
2. Open input image
3. Binarize the image
4. Downsample the image
5. Thin the image
   (a) Mark the border and interior pixels
   (b) Mark deletable border pixels
   (c) Calculate Yokoi connectivity number(YCN) of deletable pixels
   (d) Remove deletable pixels which have `YCN` = 1
   (e) Go back to `(a)` until no border pixels can be shrinked anymore
6. Output thinned result to a text file

```
#!/usr/local/bin/python3.5
import sys
from PIL import Image

def binarize(data):
  ...

def downsampling(data, hei, wid):
  ...

def expand(data, hei, wid):
  ...

def markIB(data, hei, wid):
  ...

def mark_deletable(data, hei, wid):
  ...

def h(b, c, d, e):
  ...

def thinning(data, hei, wid):
  ...

def main():
  # initial setup, handle system parameters
```

```
    ...

  # get input image
    ...

  # get 1D image data
  pixellist = list(img.getdata())

  # binarize the image
  data = binarize(pixellist)

  # downsampling the image
  data, hei, wid = downsampling(data, hei, wid)

  # thinning the image
  data, hei, wid = thinning(data, hei, wid)

  # output to text file
    ...

if __name__ == '__main__':
  main()
```

Thinning

To do thinning on an image, firstly, I need to mark the interior and border pixels on image.

```
def markIB(data, hei, wid):
  ret = [0] * len(data)
  for y in range(1, hei-1):
    for x in range(1, wid-1):
      curr = y * wid + x
      count = 0
      if data[curr]:
        count += data[curr-1] + data[curr+1]
        count += data[curr-wid-1] + data[curr-wid] + data[curr-wid+1]
        count += data[curr+wid-1] + data[curr+wid] + data[curr+wid+1]
        if count < 8:
          ret[curr] = 1
        elif count == 8:
          ret[curr] = 2
  return ret
```

And then from marked image, I find the border pixel that next to some interior pixel and give them a specific label.

```
def mark_deletable(data, hei, wid):
  for y in range(1, hei-1):
    for x in range(1, wid-1):
      curr = y * wid + x
      if data[curr] == 1:
        if 2 in [data[curr-1], data[curr+1], data[curr-wid-1], data[curr-
            wid], data[curr-wid+1], data[curr+wid-1], data[curr+wid], data[
            curr+wid+1]]:
```

```
            data[curr] = 3
```

In order to calculate Yokoi connectivity number, I define a function to tell the pattern of the neighbor of a pixel.

$$a_1 = h(x_0, x_1, x_6, x_2)$$
$$a_2 = h(x_0, x_2, x_7, x_3)$$
$$a_3 = h(x_0, x_3, x_8, x_4)$$
$$a_4 = h(x_0, x_4, x_5, x_1)$$

And the deletable pixel x that can really be shrinked is

$$f(a_1, a_2, a_3, a_4, x) = g \text{ if excatly one of } a_1, a_2, a_3, a_4 = 1$$

```python
def h(b, c, d, e):
  return 1 if b == c and (d != b or e != b) else 0
```

And finally, this is the complete steps to do thinning on a given image.

```python
def thinning(data, hei, wid):
  # expand the border of image
  exp_data, exp_hei, exp_wid = expand(data, hei, wid)

  prev_data = exp_data[:]

  while True:
    # mark border(1) and interior(2)
    marked = markIB(exp_data, exp_hei, exp_wid)

    # find deletable border(3)
    mark_deletable(marked, exp_hei, exp_wid)

    for y in range(1, exp_hei-1):
      for x in range(1, exp_wid-1):
        curr = y * exp_wid + x
        if marked[curr] == 3:
          # calculate yokoi connectivity number
          f = 0
          f += h(exp_data[curr], exp_data[curr+1], exp_data[curr-exp_wid
              +1], exp_data[curr-exp_wid])
          f += h(exp_data[curr], exp_data[curr-exp_wid], exp_data[curr-
              exp_wid-1], exp_data[curr-1])
          f += h(exp_data[curr], exp_data[curr-1], exp_data[curr+exp_wid
              -1], exp_data[curr+exp_wid])
          f += h(exp_data[curr], exp_data[curr+exp_wid], exp_data[curr+
              exp_wid+1], exp_data[curr+1])
          if f == 1:
            marked[curr] = 0
            exp_data[curr] = 0

    # compare shrinked data to previous data
    for i in range(len(exp_data)):
      if exp_data[i] != prev_data[i]:
```

```
            break
        else:
            break

    # backup previous image
    prev_data = exp_data[:]

    return exp_data, exp_hei, exp_wid
```

Result

I use * to represent the white pixels that remain after thinning.

Figure 2: thinned lena.bmp

How to Use

There's only one executable program in my submission, and its name is *thinning.py*. To run this program, just type this command "`./thinning.py [input image] [output file]`".