Thierry Cantenot

Report

Digital Image Processing

« Assignments »

2014-2015

Summary

A. Image representation and description

A.1 Problem statement

(a) Develop a program to implement the boundary following algorithm, the resampling grid and calculate the chain code and the first difference chain code. Use the image 'noisy_stroke.tif' for test. (For technique details, please refer to pp.818-822 (3rd edition, Gonzalez DIP) or boundaryfollowing.pdf at the same address of the slides.)

(b) Develop a program to implement the image description by the principal components (PC). Calculate and display the PC images and the reconstructed images from 2 PCs. Use the six images in 'washingtonDC.rar' as the test images.

A.2 Python implementation

Four programs :

- Boundary following : **boundary.py**
  Usage : **boundary.py [-h] [–smooth] image_path**
  Use **python boundary.py -h** to see the help.

- Resampling grid : **resampling.py**
  Usage : **resampling.py [-h] [-s SAMPLING [SAMPLING ...]] boundary_image**
  Use **python resampling.py -h** to see the help.

- Chain code : **chaincode.py**
  Usage : **chaincode.py [-h] [-s SAMPLING [SAMPLING ...]] boundary_image**
  Use **python chaincode.py -h** to see the help.

- Image description by the principal components (PC) : **pc.py**
  Usage : **pc.py [-h] [-n N] [–debug] [–diff] [–nshow]**
  Use **python pc.py -h** to see the help.

A.3 Boundary following

python boundary.py noisy_stroke.tif.

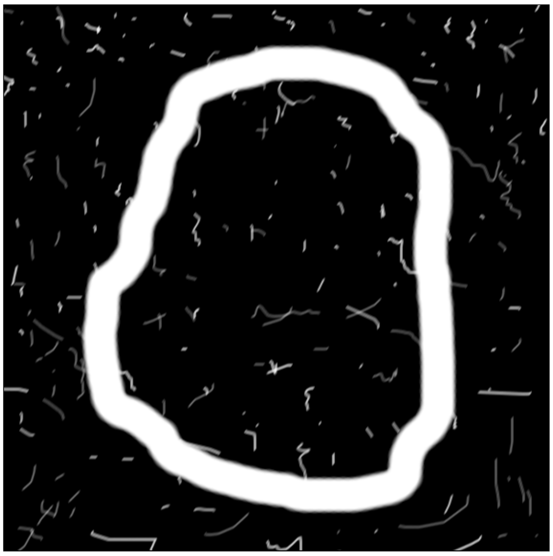

Figure A.1 – Original image

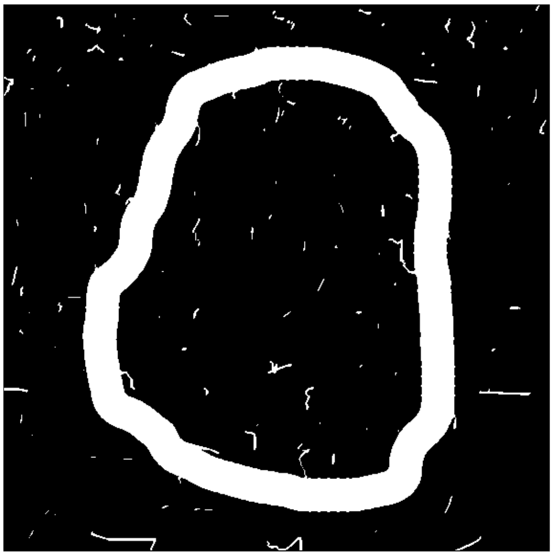

Figure A.2 – Black & white

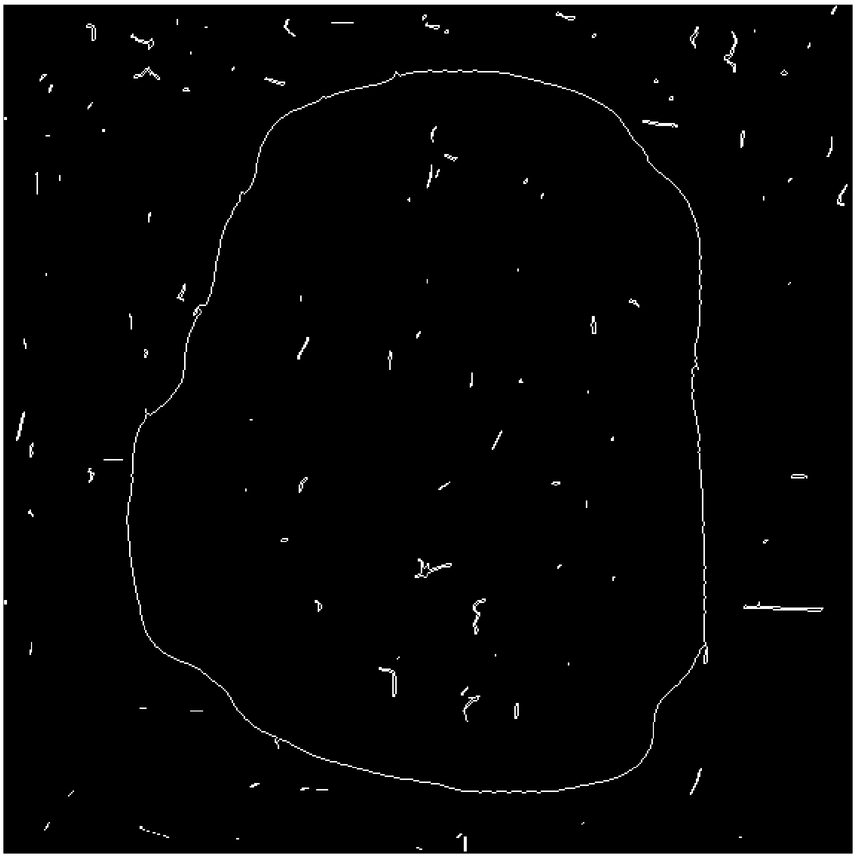

Figure A.3 – Boundaries

Even the boundaries of the noise are found... We need to remove the noise beforehand. For that, let use a Gaussian blur of mean 0 and variance 10.

python boundary.py noisy_stroke.tif –smooth.

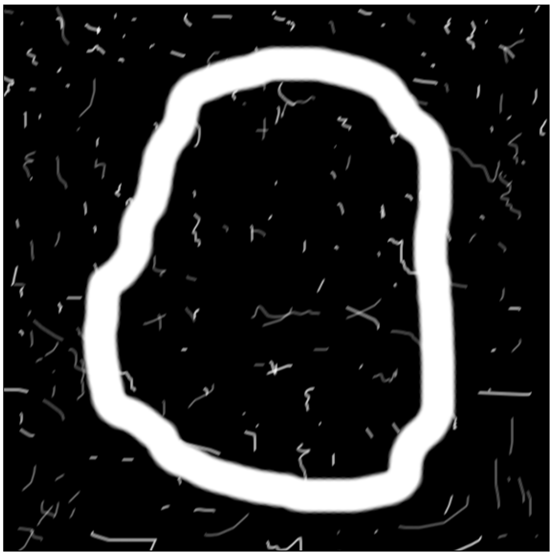

Figure A.4 – Original image

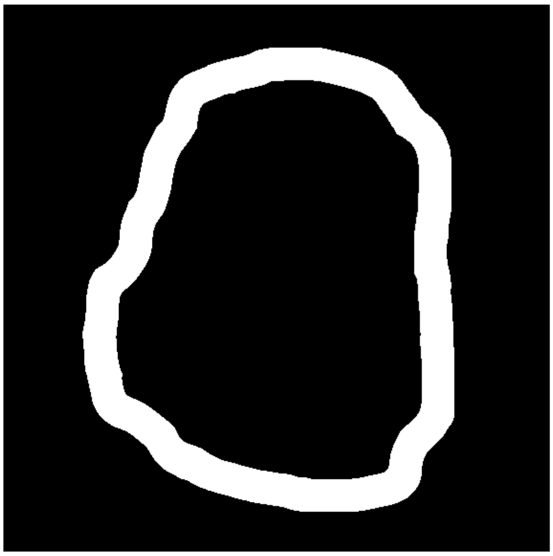

Figure A.5 – Smoothing + binarisation

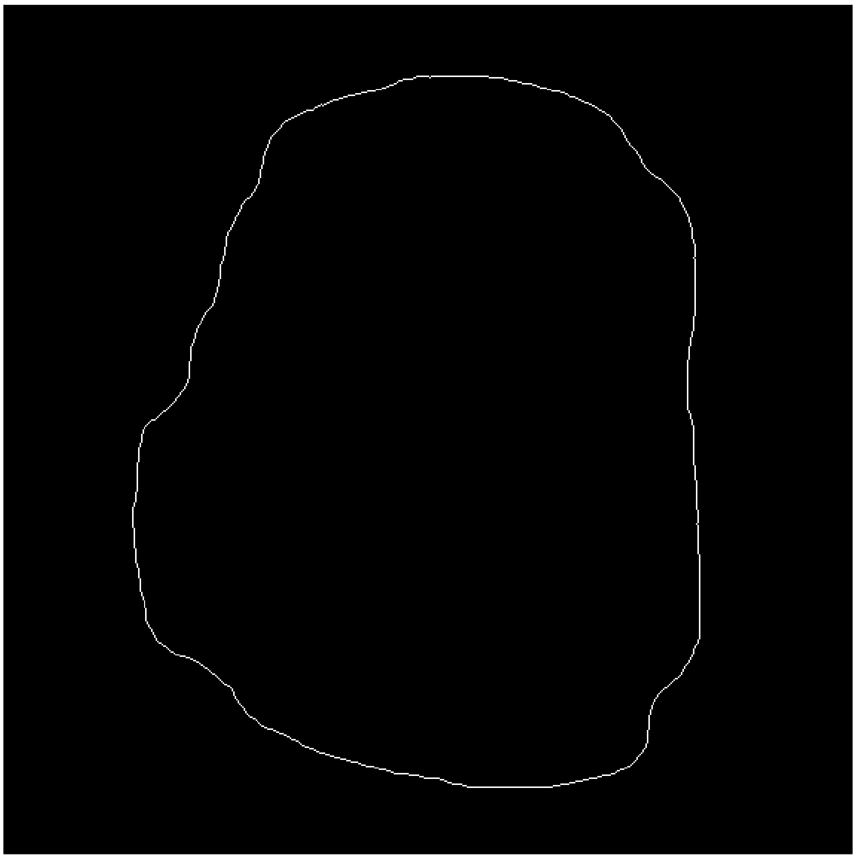

Figure A.6 – Boundaries

A.4 Resampling grid

python resampling.py noisy_stroke_boundary.png -s Sx Sy, where *Sx* and *Sy* are the sampling intervalles along the X and Y axis.

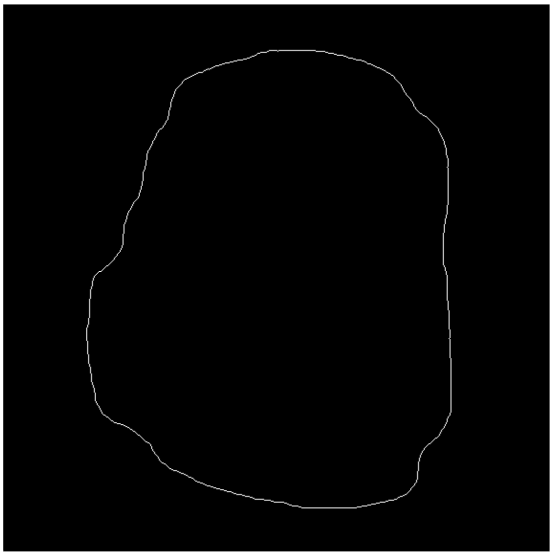

Figure A.7 – Original image

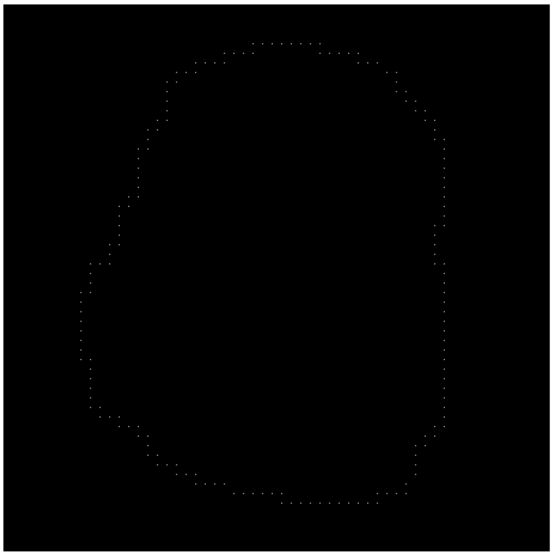

Figure A.8 – R-grid (S = (10, 10))

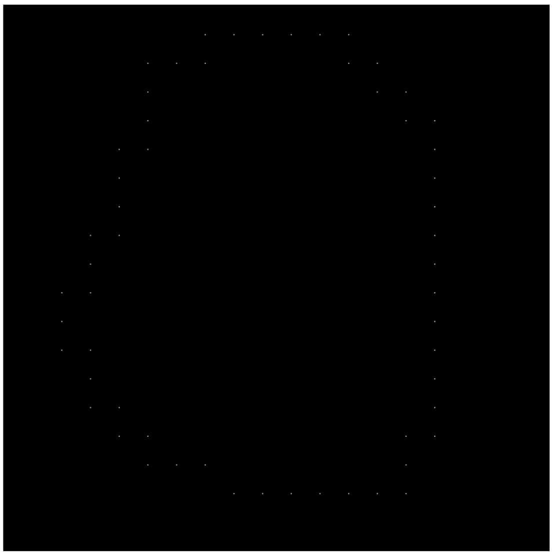

Figure A.9 – R-grid (S = (30, 30))

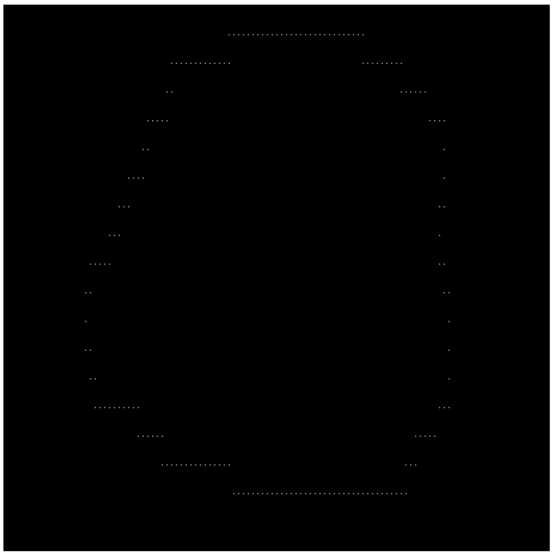

Figure A.10 – R-grid (S = (5, 30))

A.5 Chain code and first difference chain code

A.5.1 Chain code - resampling grid (10, 10)

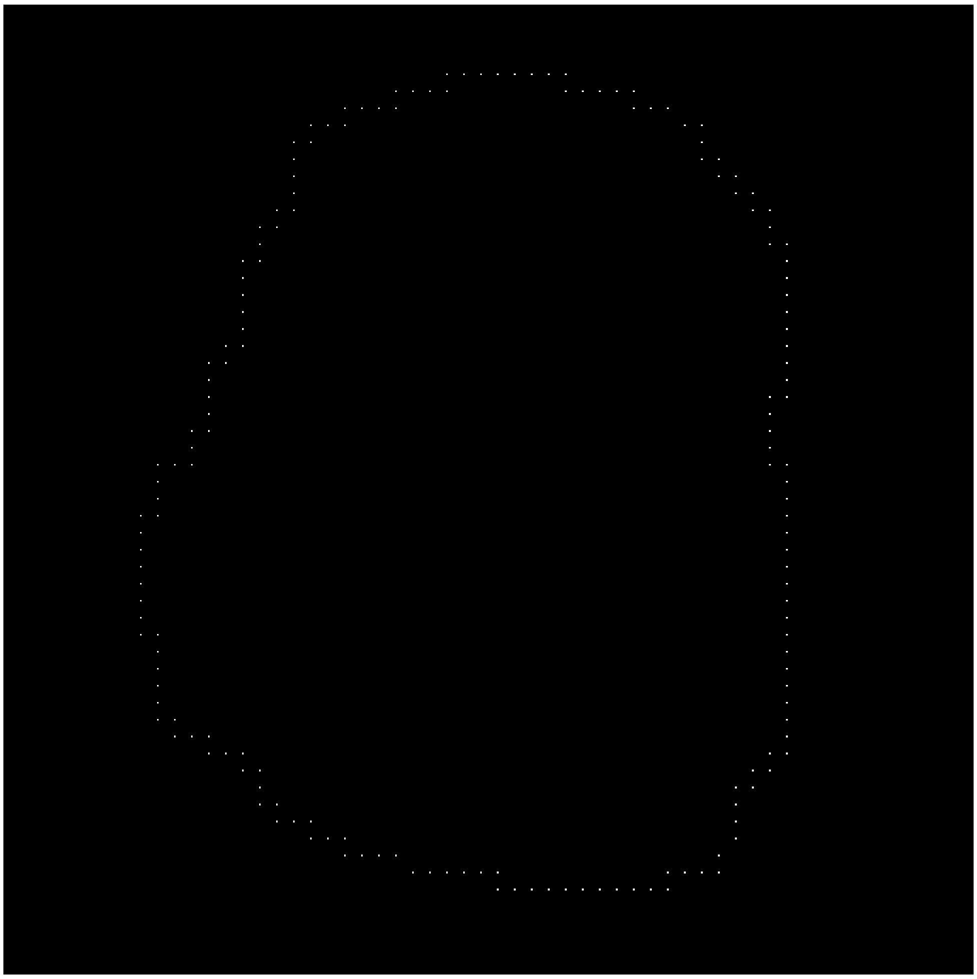

Figure A.11 – Resampling grid (S = (10, 10))

Chaincode (length = 170) :

```
00000060000600706606060606606666666
66466660666666666666666664646466656
44464444444444244444344424424424224
24424424222224222222202220022022220
     202222202202022220200200020002
```

First difference (length = 169) :

```
00000620006207160262626260260000000
06200026000000000000000062626200716
00260000000006200007100620620626026
20620626000026000000620060206200062
     62000062062620006260260026002
```

A.5.2 Chain code - resampling grid (30, 30)

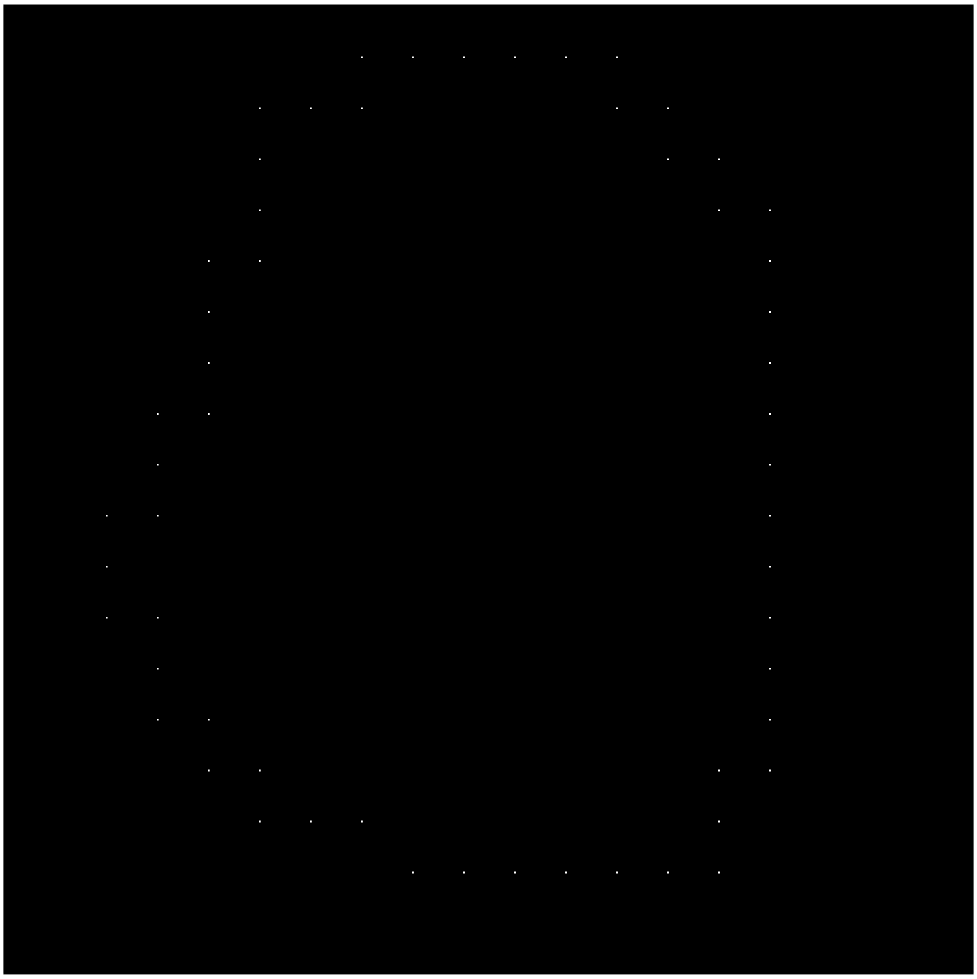

FIGURE A.12 – Resampling grid (S = (30, 30))

Chaincode (length = 56) :

00060606066666666666466444444344242422422022022202220020

First difference (length = 55) :

0062626260000000000620600000710626260260620620062006026

A.6 Principal components

```
python pc.py –diff -n 2
```