

A. Introduction

This program is designed for feature extraction of a binary image and built on the environment of Mac OS X Mountain Lion with the tool of Xcode. Since Mac OS X is an Unix-like OS, the program is compatible with the OS like Unix and Linux if using the same compilers. The next section will provide you with the description of how to compile and execute this program. Moreover, in the last section it will give you the idea how the program deals with the process of feature extraction from a binary image.

B. How to compile and run the program?

This program is compiled by [GNU compiler collection \(GCC\)](#) of the latest version 4.7.2, which is suitable for operating systems, such as Unix, Linux, and Mac OS X. Once you install the compiler, you will be able to compile and run this program on your computer with those operating systems listed above. Here is the instruction how you can compile the program. (Assume you installed the compiler on your computer already)

Step 1: unzip the file.

Step 2: change your current directory into the directory containing the files from Step 1.

Step 3: type the command as below.

`g++ main.cpp -o [the name of executable file]`

Step 4: type the name of executable file with prefix " ./ " as below.

`./[the name of executable file]`

After step 4, the program will start to execute until all the process is completed, and it will produce one text file named "[result.txt](#)" to show the result of feature extraction from input images.

C. How is the structure of this program?

In the folder of the project, there are four files in total shown as follows. The header file is named Config.h, which is included by the other three files because it defines several libraries and definitions of structs that the program needs and it can be commonly used in the other three files. The structure of the folder is shown as below.

Project folder

- |-- Config.h
- |-- Main.cpp
- |-- FileController.cpp
- |-- FeatureExtractor.cpp

The program is composed of three parts: main function and two classes. What the task of main function here is to be an entry of the program, which will show you a high-level flow of how the program executes. In addition, two classes would dominate how this program works specifically. One is the class [FileController](#), which is in charge of the process of loading input data and outputting a text file named "[result.txt](#)" as the result from the computation of feature extraction from input images. The other is the class [FeatureExtractor](#), which is responsible for all kinds of computations about feature extraction such as area, centroid, perimeter, and axis of least inertia.

D. Feature extraction

Since the input binary images contain two values inside, the foreground has values of '0' and the background has values of '255', the program will transfer those two values into '0' and '1' for further use. Therefore, the value of foreground would be changed into '1', and the other '0', which means the value '1' represents foreground or the objects we are interested in and the value '0' stands for background when feature extraction is running.

255	255	255	255
255	0	255	255
255	0	0	255
255	0	255	255

transfer →

0	0	0	0
0	1	0	0
0	1	1	0
0	1	0	0









As mentioned in the section C, the program will extract four kinds of features, area, centroid, perimeter, and axis of least inertia, from input binary images. The following content will simply describe how the program computes those features.

Feature: Area and centroid

In the program, these two features will be computed at the same time. For the feature Area, the program will count the number of the value '1' in a binary image. For centroid, it will be computed with the coordinates of pixels in the binary based on x-y coordinate system, which means the program will sum up the x-value of every pixel with the value '1' and then have the sum divided by the value of feature area. The same process will proceed with the y-value also. Finally, the program will get the feature centroid.

Feature: perimeter

To compute this feature, the program takes advantage of the algorithm named [boundary following algorithm](#), which scans 8-connected neighbors one by one in a clockwise order for the current pixel. In addition, we still need to pay attention to how we define the clockwise order in different moving direction, since the first neighbor in the scanning order would be shifted as moving direction changes, or we cannot get correct boundary pixels if the order keep fixed when the algorithm is proceeding. Furthermore, here is how we define the first neighbor in the clockwise order for each of 8 directions as follow.

Types of Moving Direction	Figure of Direction	Types of clockwise order									
$(c, r) \rightarrow (c+1, r)$		<table><tr><td>7</td><td>B</td><td>1</td></tr><tr><td>6</td><td>C</td><td>2</td></tr><tr><td>5</td><td>4</td><td>3</td></tr></table>	7	B	1	6	C	2	5	4	3
7	B		1								
6	C	2									
5	4	3									
$(c, r) \rightarrow (c+1, r+1)$											
$(c, r) \rightarrow (c, r+1)$		<table><tr><td>5</td><td>6</td><td>7</td></tr><tr><td>4</td><td>C</td><td>B</td></tr><tr><td>3</td><td>2</td><td>1</td></tr></table>	5	6	7	4	C	B	3	2	1
5	6		7								
4	C	B									
3	2	1									
$(c, r) \rightarrow (c-1, r+1)$											
$(c, r) \rightarrow (c-1, r)$		<table><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>2</td><td>C</td><td>6</td></tr><tr><td>1</td><td>B</td><td>7</td></tr></table>	3	4	5	2	C	6	1	B	7
3	4		5								
2	C	6									
1	B	7									
$(c, r) \rightarrow (c-1, r-1)$											
$(c, r) \rightarrow (c, r-1)$		<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>B</td><td>C</td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table>	1	2	3	B	C	4	7	6	5
1	2		3								
B	C	4									
7	6	5									
$(c, r) \rightarrow (c+1, r-1)$											

As the table shown above, the character 'B' means background pixels and the next pixel in the clockwise order would be the first pixels in each of moving directions.

Feature: axis of least inertia

For this feature, the program will be running based on the results of the features: Area and Centroid. Here is the equation used in the feature extraction as follow.

$$\begin{aligned}
 \tan 2\hat{\alpha} &= \frac{2 \sum (r - \bar{r})(c - \bar{c})}{\sum (r - \bar{r})(r - \bar{r}) - \sum (c - \bar{c})(c - \bar{c})} \\
 &= \frac{\frac{1}{A} 2 \sum (r - \bar{r})(c - \bar{c})}{\frac{1}{A} \sum (r - \bar{r})(r - \bar{r}) - \frac{1}{A} \sum (c - \bar{c})(c - \bar{c})} \\
 &= \frac{2 \mu_{rc}}{\mu_{rr} - \mu_{cc}}
 \end{aligned}$$

To get the final value of alpha, we need to transfer the result from the above equation with arc tangent, so the equation is shown as below.

$$\alpha = \frac{\text{art tangent}\left(\frac{2 \mu_{rc}}{\mu_{rr} - \mu_{cc}}\right)}{2}$$