



# 图像信息处理实验报告

**Digital Image Processing (Experiments)**

|  |  |
| --- | --- |
| 姓 名 |  |
| 指导老师 |  |
| 学 号 |  |
| 专业班级 |  |

**二〇一九年**

**秋冬学期**

## 实验一

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称：** | 作业2 | **指导老师：** |  | **成绩：** |  |

### 一、实验目的和要求（必填）

1.对一张BMP图像选取合适的阈值，进行二值化（Image binarization）

2.对1中产生的二值化图片进行腐蚀（Binary image erosion）

3.对1中产生的二值化图片进行膨胀（Binary image dilation）

4.对1中产生的二值化图片进行开操作，即先腐蚀后膨胀（Binary image opening）

5.对1中产生的二值化图片进行闭操作，即先膨胀后腐蚀（Binary image closing）

### 二、实验内容和原理（必填）

1、YUV与RGB之间的转换

Y = 0.299\*R + 0.587\*G + 0.114\*B;

U = -0.147\*R - 0.289\*G + 0.436\*B;

V = 0.615\*R - 0.515\*G - 0.100\*B;

R = Y + 1.14\*V;

G = Y - 0.39\*U - 0.58\*V;

B = Y + 2.03\*U;

2、大津otsu算法求最佳分割阈值

（1）原理：利用阈值将原图像分成前景、背景两个部分，当取到最佳阈值时前景与背景差别最大，在otsu算法中用最大类间方差来衡量这种差别。

（2）性能：otsu算法对噪点和大小十分敏感，仅对类间方差为单峰的图像产生较好的分割效果

（3）公式推导：

·记T为前景和背景的分割阈值，前景点数占图像比例w\_below，平均灰度u\_below，后景点数占图像比例w\_above，平均灰度u\_above

·则图像平均总灰度为u=w\_below\*u\_below+w\_above\*u\_above

·前景和背景的方差为g=w\_below\*(u\_below-u)^2+w\_above\*(u-u\_above)^2

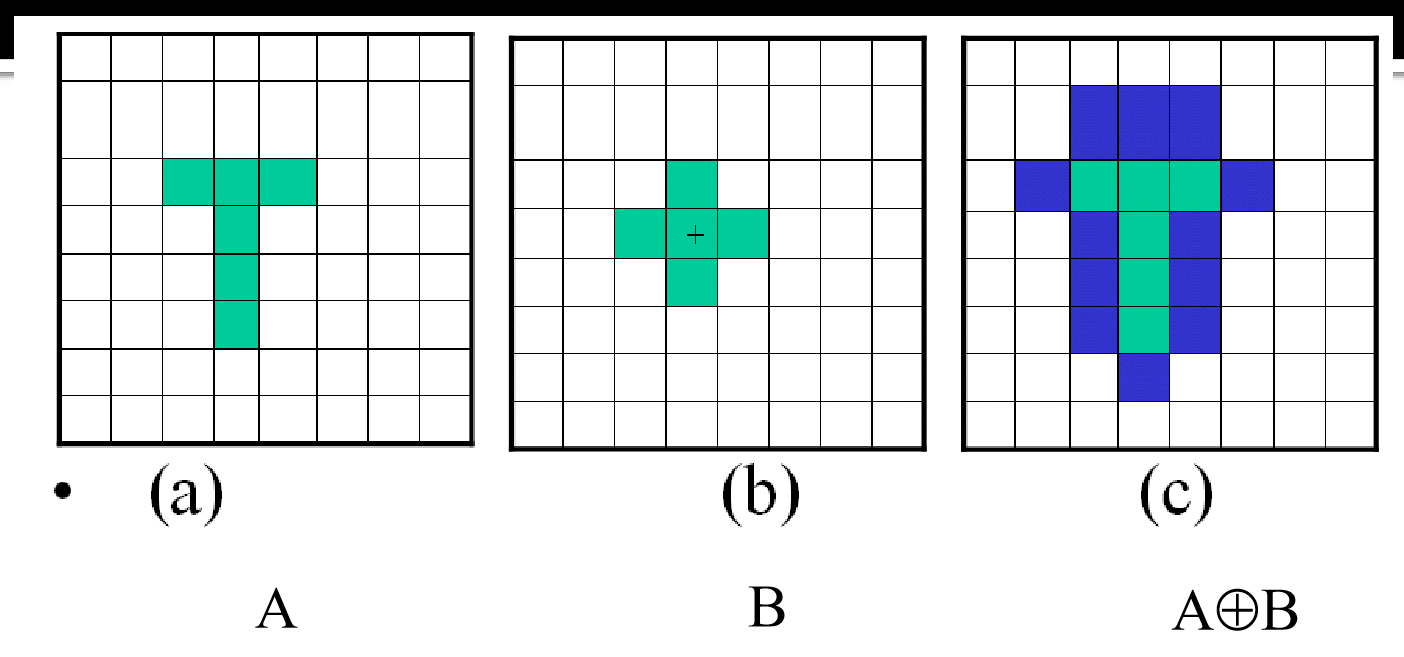
·当g最大时所取的T值，即为该图的最佳阈值，记为perfect\_T

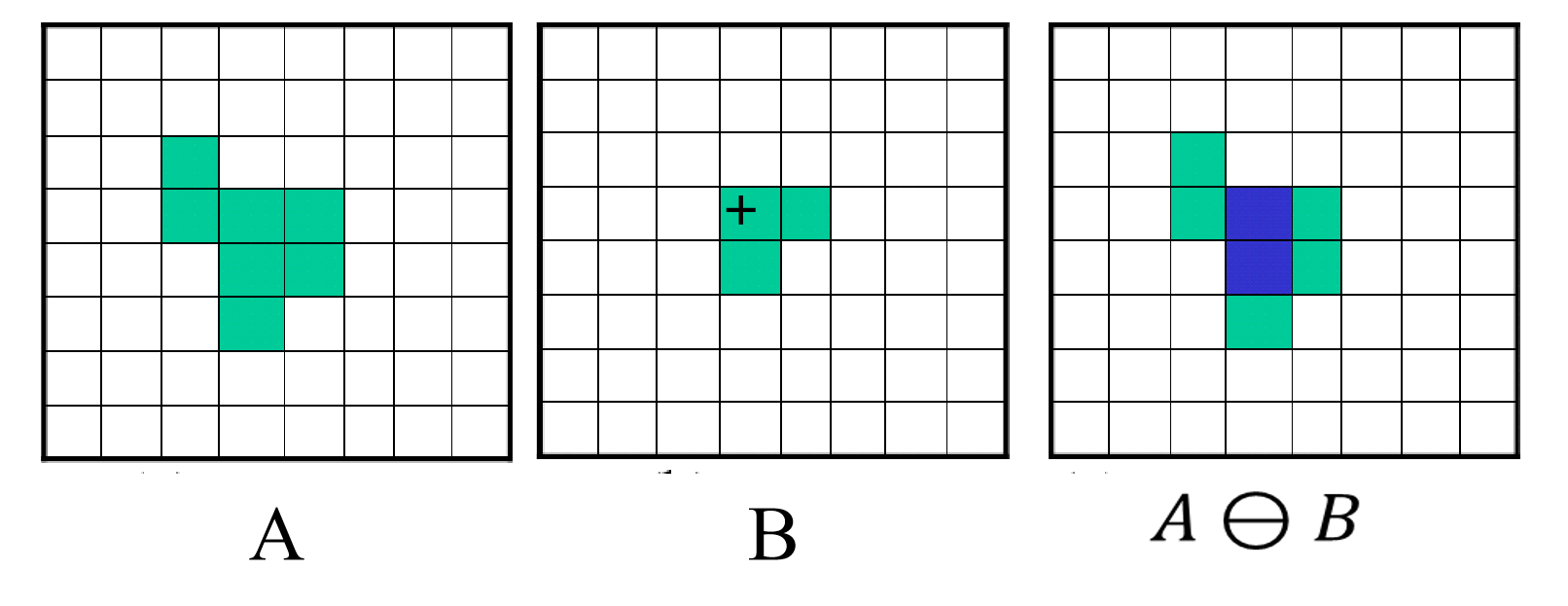
（4）算法实现：

1. **void** Binarization()//data from Y[] to output\_Y[]
2. {
3. //get Threshold
4. unsigned **char** T = 0, perfect\_T = 0;
5. **double** w\_below, w\_above, u\_below, u\_above, u;
6. **long** number\_below\_T, number\_above\_T;
8. **for** (**int** num = 0; num <= 255; num++) {
9. **long** all\_below\_T = 0L, all\_above\_T = 0L;
10. number\_below\_T = 0L, number\_above\_T = 0L;
12. **for** (**int** j = 0; j < height; j++) {
13. **for** (**int** i = 0; i < width; i++) {
14. **if** (T >= Y[j][i]) {
15. number\_below\_T++;
16. all\_below\_T += Y[j][i];
17. }
18. **else** {
19. number\_above\_T++;
20. all\_above\_T += Y[j][i];
21. }
22. }
23. }
24. //if it is zero ,this loop is useless
25. **if** (number\_below\_T == 0 || number\_above\_T == 0) **continue**;
27. //密度
28. u\_below = (**double**)all\_below\_T / number\_below\_T;
29. u\_above = (**double**)all\_above\_T / number\_above\_T;
31. w\_below = (**double**)number\_below\_T / (height\*width);
32. w\_above = (**double**)number\_above\_T / (height\*width);
34. //图像平均总灰度
35. u = w\_below \* u\_below + w\_above \* u\_above;
37. **double** fc[255] = { 0.0 }, max\_fc = 0.0;
38. fc[num] = w\_below \* pow(abs(u\_below - u), 2.0) + w\_above \* pow(abs(u\_above - u), 2.0);
39. //循环比较获得最大方差，对应的记为最佳阈值
40. **if** (fc[num] > max\_fc) {
41. perfect\_T = T;
42. max\_fc = fc[num];
43. }
44. T++;
45. }
47. //对像素进行二值化处理
48. **for** (**int** j = 0; j < height; j++)
49. {
50. **for** (**int** i = 0; i < width; i++)
51. {
52. **if** (Y[j][i] < perfect\_T)
53. output\_Y[j][i] = 0;
54. **else**
55. output\_Y[j][i] = 255;
56. output\_U[j][i] = 0;
57. output\_V[j][i] = 0;
58. }
59. }
60. printf("Binarization finished!\n");
61. }

3、腐蚀操作与膨胀操作

（1）原理：

Dilation（膨胀）：即往外扩展一圈

Erosion（腐蚀）：即向内收缩一圈

具体结果与所选的dilation/erosion模型（即B的形状）有关

（2）算法实现：

膨胀函数：

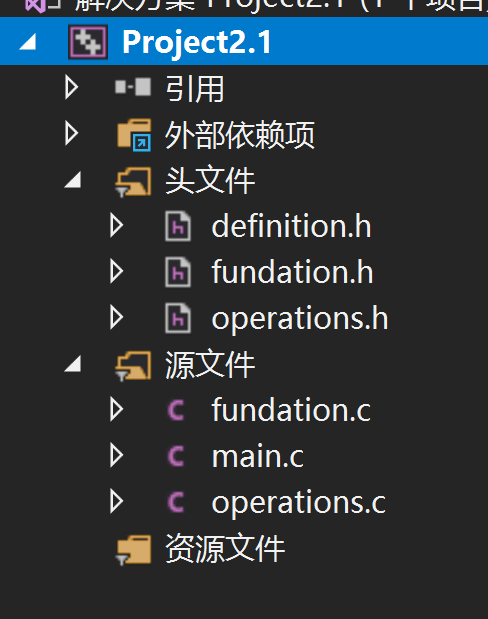
1. **void** Dilation() {
2. //Binarization();
3. **int** Struture\_Element[elementline][elementline] = { {0} };//假设是膨胀模型十字型
4. **int** centre = (elementline - 1) / 2;
5. /\*生成十字形element\*/
6. **for** (**int** j = 0; j < elementline; j++)
7. {
8. **for** (**int** i = 0; i < elementline; i++)
9. {
10. **if** (j == centre || i == centre)
11. Struture\_Element[j][i] = 1;
12. **else**
13. Struture\_Element[j][i] = 0;
14. }
15. }
17. //传入经过binary后的二值图数据
18. **for** (**int** j = 0; j < height; j++)
19. {
20. **for** (**int** i = 0; i < width; i++)
21. {
22. output\_Y[j][i] = output\_Y[j][i];//\*\*\*\*\*//
23. output\_U[j][i] = 0;
24. output\_V[j][i] = 0;
25. }
26. }
28. /\*膨胀一小圈\*/
29. **for** (**int** j = 0; j < height - elementline; j++)
30. {
31. **for** (**int** i = 0; i < width - elementline; i++)
32. {
33. **int** Istrue = 0;
34. **for** (**int** i1 = 0; i1 < elementline; i1++)
35. **for** (**int** j1 = 0; j1 < elementline; j1++)
36. {
37. **if** (Struture\_Element[i1][j1] == 1 && Y[j + j1][i + i1] == 0)
38. Istrue = 1;
39. }
40. /\*黑色是Y=0 ，白色时Y=255\*/
41. **if** (Istrue == 1)
42. output\_Y[j + centre][i + centre] = 0;
44. }
45. }
46. printf("Dilation successed!\n");
47. }

侵蚀函数：

1. **void** Erosion()
2. {
3. //Binarization();
4. **int** Struture\_Element[elementline][elementline] = { {0} };
5. **int** centre = (elementline - 1) / 2;
6. //生成十字形element，也可更改参数生成不同形状的element
7. **for** (**int** j = 0; j < elementline; j++)
8. {
9. **for** (**int** i = 0; i < elementline; i++)
10. {
11. **if** (j == centre || i == centre)
12. Struture\_Element[j][i] = 1;
13. **else**
14. Struture\_Element[j][i] = 0;
15. }
16. }
18. //导入图片数据
19. **for** (**int** j = 0; j < height; j++)
20. {
21. **for** (**int** i = 0; i < width; i++)
22. {
23. output\_Y[j][i] = output\_Y[j][i];
24. output\_U[j][i] = 0;
25. output\_V[j][i] = 0;
26. }
27. }
29. //腐蚀一小圈
30. **for** (**int** j = 0; j < height - elementline; j++)
31. {
32. **for** (**int** i = 0; i < width - elementline; i++)
33. {
34. **int** Istrue = 0;
35. **for** (**int** i1 = 0; i1 < elementline; i1++)
36. **for** (**int** j1 = 0; j1 < elementline; j1++)
37. {
38. **if** (Struture\_Element[i1][j1] == 0)
39. **continue**;
40. **else** **if** (Struture\_Element[i1][j1] == 1 && output\_Y[j + i1][i + j1] != 0)//不同时黑色
41. Istrue = 1;//不全等
42. }
44. **if** (Istrue == 0)
45. output\_Y[j + centre][i + centre] = 0;
46. }
47. }
48. printf("Erosion successed!\n");
49. }

### 三、源代码与分析

1、项目框架

 main.c：主函数

fundation.c：进行有关bmp图像读入读出、bmp与yuv格式间转换等与图形相关的基本函数

operations.c：包含binarization、dilation、erosion、opening、closing等实验要求重要函数

对应存在各个.h头文件，进行对函数的声明

definition.h：定义全局变量与常量

2、主函数代码分析

1. #include<string.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <math.h>
5. #include <time.h>
7. #include"definition.h"
8. #include"fundation.h"
9. #include"operations.h"
11. **int** main(**int** argc, **char**\* argv[])
12. {
13. unsigned **char** \*fp\_temp;
15. **FILE** \*fpbmp;
16. **FILE** \*fpout;
17. Initialization();
19. fpbmp = openfile("input.bmp");  //read the bmp
20. bmpDataPart(fpbmp);
22. fpout = writefile("binar.bmp"); //the name decided by the operations
23. addHeadertofile(fpbmp, fpout);  //printf the fileinfo and header in the outfile
24. RGBtoYUV(); //turn the RGBinfo to YUVinfo
26. //------------here to choose which operation you wanna to use---
27. Binarization();
28. //Dilation();
29. //Erosion();
30. //Opening();
31. //Closing();
33. YUVtoRGB(); //turn back to RGBinfo
34. bmpoutput(fpout);   //print the answer
36. fclose(fpbmp);  //close the input file
37. fclose(fpout);  //close the output file
39. }

22行&27-31行：由需要进行的操作决定

例如：若要获得dilation图，则取消28行注释符号，修改输出文件名为dilation.bmp

3、definition.h

1. #ifndef COMPILE\_DEFINITION
2. #define COMPILE\_DEFINITION
3. #define BITMAPFILEHEADERLENGTH 14   // The bmp FileHeader length is 14
4. #define BM 19778                    // The ASCII code for BM
5. #define elementline 3               //designed by the operation erosion and dilation
6. #define MAXSIZE 10000
7. unsigned **char** Y[MAXSIZE][MAXSIZE], output\_Y[MAXSIZE][MAXSIZE];
8. unsigned **char** U[MAXSIZE][MAXSIZE], output\_U[MAXSIZE][MAXSIZE];
9. unsigned **char** V[MAXSIZE][MAXSIZE], output\_V[MAXSIZE][MAXSIZE];
11. **long** width;          // The Width of the Data Part
12. **long** height;         // The Height of the Data Part
13. unsigned **char** r[MAXSIZE][MAXSIZE], output\_r[MAXSIZE][MAXSIZE];
14. unsigned **char** g[MAXSIZE][MAXSIZE], output\_g[MAXSIZE][MAXSIZE];
15. unsigned **char** b[MAXSIZE][MAXSIZE], output\_b[MAXSIZE][MAXSIZE];
16. unsigned **char** records[MAXSIZE][MAXSIZE];
18. #endif // !COMPILE\_DEFINITION

对于全局变量与常量进行定义

4、关于fundation.c

对应头文件fundation.h及函数声明

1. #ifndef COMPILE\_FUNDATION
2. #define COMPILE\_FUNDATION
4. **FILE** \*openfile(**char** filename[]);
5. **FILE** \*writefile(**char** filename[]);
7. **void** RGBtoYUV();    //turn the RGBinfo to YUVinfo
8. **void** YUVtoRGB();    //turn back and be ready to print
9. **void** bmpDataPart(**FILE**\* fpbmp);
10. **void** Initialization();
11. **void** addHeadertofile(**FILE** \*input, **FILE** \*output);    //add the standard data to the head of the file
12. **void** bmpFileTest(**FILE**\* fpbmp);      //test that if it is a BM
13. **void** bmpHeaderPartLength(**FILE**\* fpbmp);  //get the length of the header
14. **void** BmpWidthHeight(**FILE**\* fpbmp);   //get the width and height of the header
15. **void** bmpoutput(**FILE** \*fpout);    //print out the file
16. #endif // !COMPILE\_FUNDATION

1）RGBtoYUV&YUVtoRGB:对于图片rgb数据与yuv数据之间进行互相转换

1. **void** RGBtoYUV()
2. {
3. //convert RGB to YUV
4. **for** (**int** j = 0; j < height; j++)
5. {
6. **for** (**int** i = 0; i < width; i++)
7. {
8. Y[j][i] = 0.299\*r[j][i] + 0.587\*g[j][i] + 0.114\*b[j][i];
9. U[j][i] = -0.147\*r[j][i] - 0.289\*g[j][i] + 0.436\*b[j][i];
10. V[j][i] = 0.615\*r[j][i] - 0.515\*g[j][i] - 0.100\*b[j][i];
11. }
12. }
13. }
14. **void** YUVtoRGB()
15. {
16. //convert YUV to GRB
17. **for** (**int** j = 0; j < height; j++)
18. {
19. **for** (**int** i = 0; i < width; i++)
20. {
21. output\_r[j][i] = output\_Y[j][i] + 1.140\*output\_V[j][i];
22. output\_g[j][i] = output\_Y[j][i] - 0.394\*output\_U[j][i] - 0.581\*output\_V[j][i];
23. output\_b[j][i] = output\_Y[j][i] + 2.032\*output\_U[j][i];
24. }
25. }
26. }

2）bmpDataPart：从input.bmp中读取rgb数据，分配数组空间

1. **void** bmpDataPart(**FILE**\* fpbmp)
2. {   //clarify the data in the bmp
3. **int** i, j = 0;
4. **int** stride;
5. unsigned **char**\* pix = NULL;
7. fseek(fpbmp, OffSet, SEEK\_SET);     //get the data
8. stride = (24 \* width + 31) / 8;
9. stride = stride / 4 \* 4;
10. pix = malloc(stride);
12. **for** (j = 0; j < height; j++)
13. {
14. fread(pix, 1, stride, fpbmp);
16. **for** (i = 0; i < width; i++)
17. {
18. r[height - 1 - j][i] = pix[i \* 3 + 2];
19. g[height - 1 - j][i] = pix[i \* 3 + 1];
20. b[height - 1 - j][i] = pix[i \* 3];
21. //init the output\_info
22. output\_r[height - 1 - j][i] = 255;
23. output\_g[height - 1 - j][i] = 255;
24. output\_b[height - 1 - j][i] = 255;
25. }
26. }
27. }

3）Initialization：对输出数组进行初始化定义，以防在height\*width空间内数组元素已有定义，影响最终输出结果

1. **void** Initialization()
2. {
3. **for** (**int** j = 0; j < height; j++)
4. {
5. **for** (**int** i = 0; i < width; i++)
6. {
7. output\_Y[j][i] = 255;
8. output\_U[j][i] = 0;
9. output\_V[j][i] = 0;
10. records[j][i] = 0;
11. }
12. }
13. }

4）addHeadertofile：通过输出文件指针，将头文件（图形经变换后头文件信息不变）写入文件

1. **void** addHeadertofile(**FILE** \*input, **FILE** \*output)
2. {
3. unsigned **char** \*fp\_temp;
5. fseek(input, 0L, SEEK\_SET);
6. fseek(output, 0L, SEEK\_SET);
8. fp\_temp = malloc(OffSet);
9. fread(fp\_temp, 1, OffSet, input);//读取头文件
10. fp\_temp[18] = (**int**)width;   //write the width
11. fp\_temp[22] = (**int**)height;  //write the height
12. fp\_temp[2] = (**int**)(OffSet + height \* width \* 3);
13. fp\_temp[34] = (**int**)(height\*((24 \* width / 8 + 3) / 4 \* 4));
14. fwrite(fp\_temp, 1, OffSet, output);//输出头文件
15. }

5）bmpFileTest：判断输入文件是否时标准的bmp文件（bmp文件文件头常量已定义为BM），本实验仅对bmp格式进行操作

1. **void** bmpFileTest(**FILE**\* fpbmp)
2. {
3. unsigned **short** bfType = 0;
5. fseek(fpbmp, 0L, SEEK\_SET);//seek\_set 起始位置
6. fread(&bfType, **sizeof**(**char**), 2, fpbmp);
7. **if** (BM != bfType)
8. {
9. printf("This file is not bmp file!!!\n");
10. exit(1);
11. }
12. }

6）bmpHeaderPartLength：读取bmp图像头文件

1. **void** bmpHeaderPartLength(**FILE**\* fpbmp)
2. {
3. fseek(fpbmp, 10L, SEEK\_SET);
4. fread(&OffSet, **sizeof**(**char**), 4, fpbmp);
5. printf("The Header Part is of length %d.\n", OffSet);
6. }

7）BmpWidthHeight：获取输入bmp文件图像的长宽像素值

1. **void** BmpWidthHeight(**FILE**\* fpbmp)
2. {
3. **int** size;
4. fseek(fpbmp, 18L, SEEK\_SET);
5. fread(&width, **sizeof**(**char**), 4, fpbmp);
6. fseek(fpbmp, 2L, SEEK\_SET);
7. fread(&size, **sizeof**(**char**), 4, fpbmp);
8. printf("The Size of the bmp file is %ld.\n", size);
9. fseek(fpbmp, 22L, SEEK\_SET);
10. fread(&height, **sizeof**(**char**), 4, fpbmp);
11. printf("The Width of the bmp file is %ld.\n", width);
12. printf("The Height of the bmp file is %ld.\n", height);
13. }

8）bmpoutput：输出操作后图像

1. **void** bmpoutput(**FILE**\* fpout)
2. {
3. **long** i, j = 0;
4. **long** stride;
5. unsigned **char**\* pixout = NULL;
7. stride = (24 \* width + 31) / 8;
8. stride = stride / 4 \* 4;
9. pixout = malloc(stride);
11. fseek(fpout, OffSet, SEEK\_SET);
13. **for** (j = 0; j < height; j++)
14. {
15. **for** (i = 0; i < width; i++)
16. {
17. pixout[i \* 3 + 2] = output\_r[height - 1 - j][i];
18. pixout[i \* 3 + 1] = output\_g[height - 1 - j][i];
19. pixout[i \* 3] = output\_b[height - 1 - j][i];
20. }
21. fwrite(pixout, 1, stride, fpout);
22. }
23. }

5、关于operation.c

对应头文件operation.h及函数声明

1. #ifndef COMPILE\_OPERATIONS
2. #define COMPILE\_OPERATIONS
3. **void** Binarization();
4. **void** Dilation();
5. **void** Erosion();
6. **void** Opening();
7. **void** Closing();
8. #endif // !COMPILE\_OPERATIONS
9. #pragma once
10. Binarization:图像二值化操作，具体函数见实验原理
11. Dilation：图像二值化膨胀操作，具体函数见实验原理
12. Erosion：图像二值化腐蚀操作，具体函数见实验原理

4）Opening：图像开操作，先进行腐蚀操作再进行膨胀操作

1. **void** Opening() {
2. Erosion();
3. **for** (**int** j = 0; j < height; j++)
4. {
5. **for** (**int** i = 0; i < width; i++)
6. {
7. Y[j][i] = output\_Y[j][i];
8. U[j][i] = output\_U[j][i];
9. V[j][i] = output\_V[j][i];
10. }
11. }
12. Dilation();
13. printf("Opening finished, Now outer noises are removed!\n");
14. }

5）Closing：图像闭操作，先进行膨胀操作再进行腐蚀操作

1. **void** Closing() {
2. Dilation();
3. **for** (**int** j = 0; j < height; j++)
4. {
5. **for** (**int** i = 0; i < width; i++)
6. {
7. Y[j][i] = output\_Y[j][i];
8. U[j][i] = output\_U[j][i];
9. V[j][i] = output\_V[j][i];
10. }
11. }
12. Erosion();
13. printf("Closing finished, Now inner noises are removed!\n");
14. }

### 四、心得体悟

1、对于图像的二值化处理：

实验过程中进行资料收集时，发现图像的二值化有很多不同的算法，但目前的主流算法时ostu大津算法，能够较有效率地区分图像前景和背景，了解到matlab中对于图像二值化的处理操作也是使用了ostu算法。

2、采用不同的操作处理会得到不同的图片（虽然区别很小，但是在细节处的区别还是比较明显的）

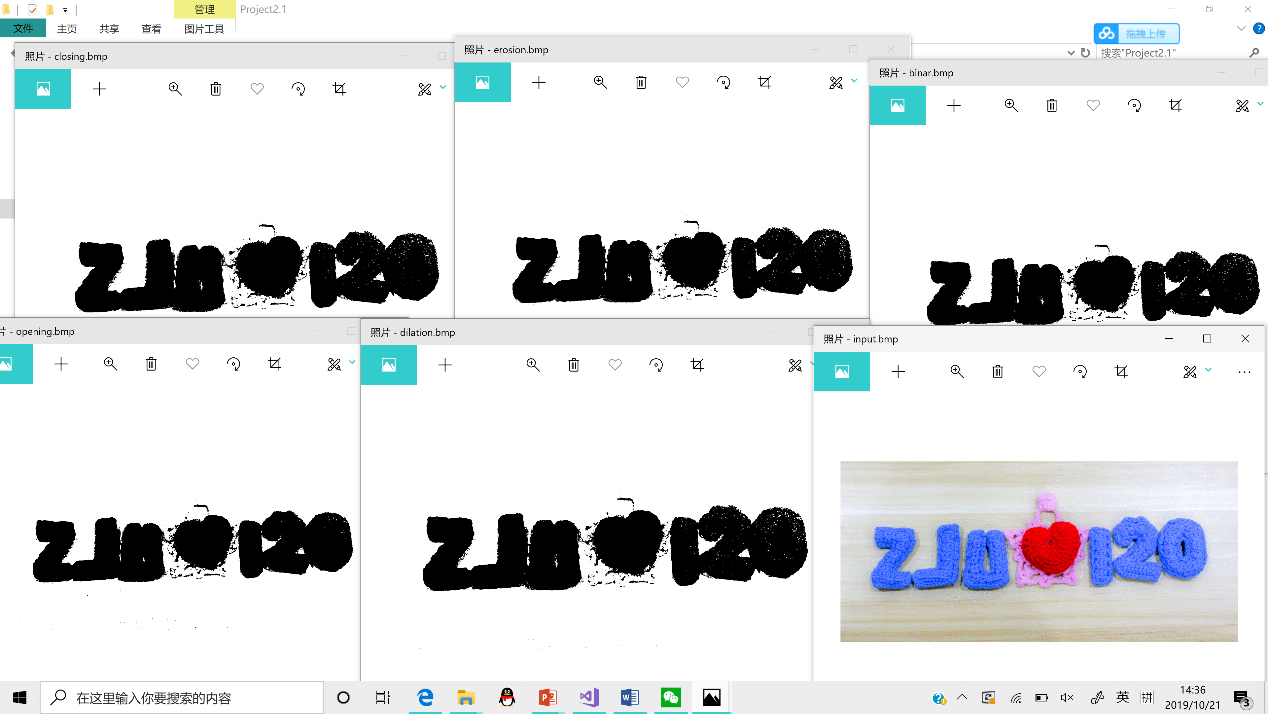
### 五、参考材料

1、<https://blog.csdn.net/liu_xiao_cheng/article/details/50420084>

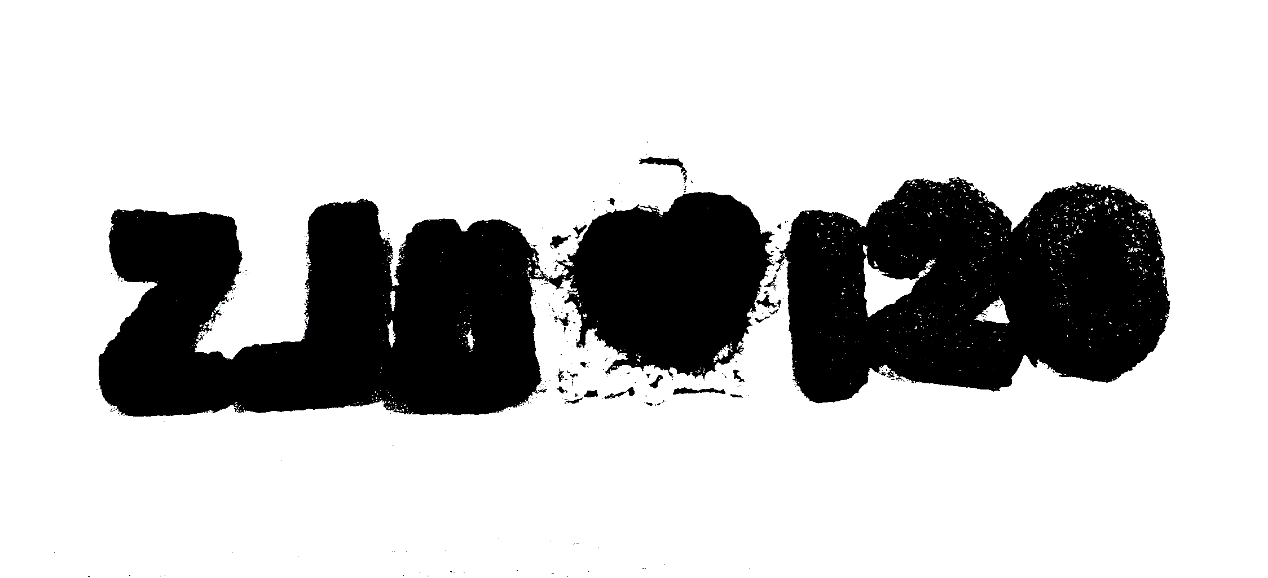
图像处理算法基础（十）---大津法求最佳分割阈值

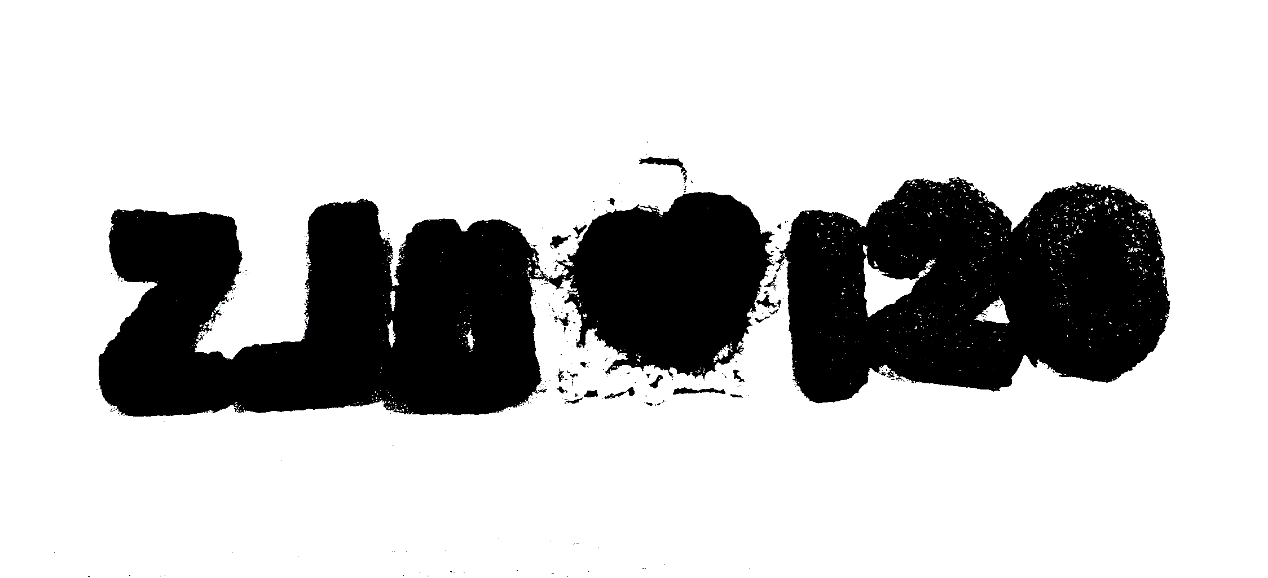
### 六、成果展示

（原图input.bmp）

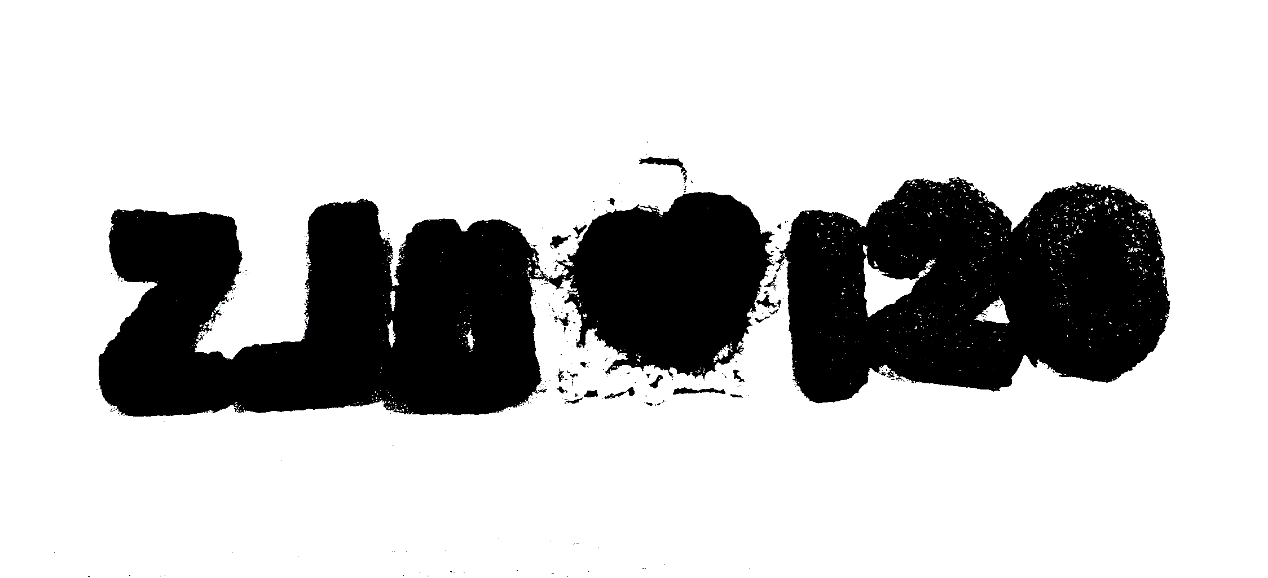
（结果）

(二值化binar.bmp)

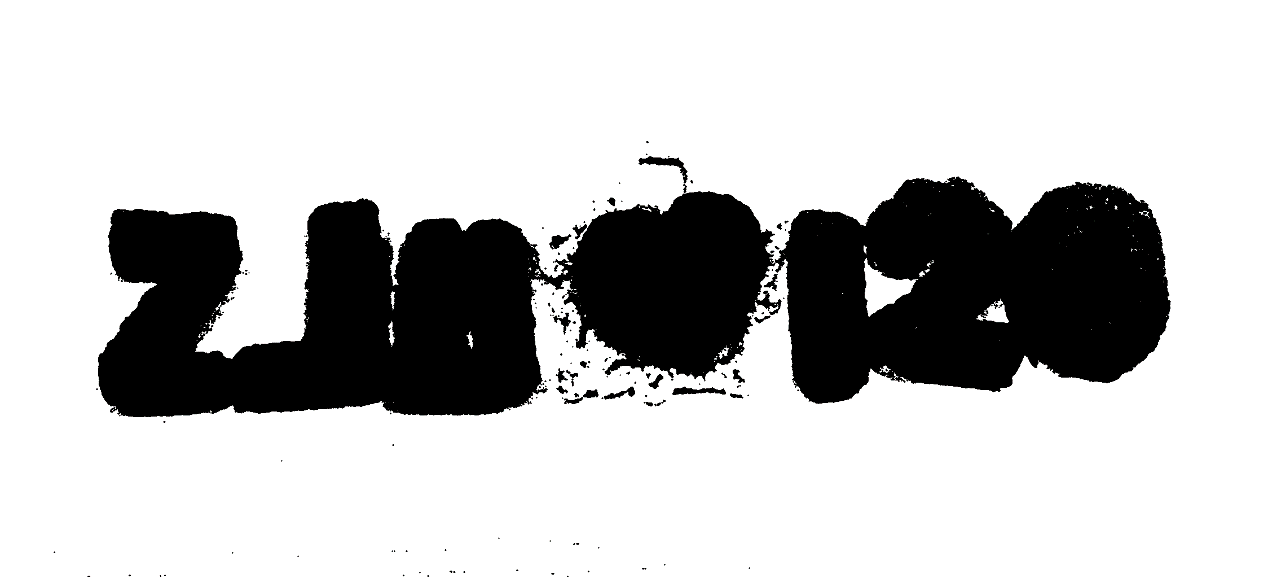
（腐蚀erosion.bmp）



(膨胀dilation.bmp)



（开操作open.bmp）



（闭操作closing.bmp）

