**实验报告**

专业：计算机科学与技术

姓名：颜晗

学号：3200105515

日期：2021/10/31

课程名称： 图像信息处理 指导老师： 宋明黎 成绩：

实验名称： 灰度图转二值图像与形态学操作

**一、实验目的和要求**

·了解灰度图转二值图的方法，学会使用大津法找到相应的阈值；

·了解各项形态学操作的原理和方法。

**二、实验内容和原理**

实验内容：使用C语言实现灰度图转二值图片，使用大津法找到对应图片的对应阈值；得到二值图片后再实现各项形态学操作并输出结果图片到不同文件中比较它们的不同。

实验原理：图片数据的本质是一个个数，因此我们可以通过改变它们的值使之呈现不同形态，而形态学操作就是通过一个像素点周围的数据情况来决定它自身的情况，借以消除噪声点或使图片更符合人类预期。

**三、实验步骤与分析**

1.彩色图转灰度图。

由于很难直接找到灰度图，所以所有使用的灰度图片需要自己转换，另外因为第一次作业的转换灰度图采取了偷懒的方法，直接对原像素数据或调色板数据进行变化，这样会对后续的操作带来麻烦，这次重新调整了转灰度图的函数，无论8位还是24位都调整为0-255的灰度调色盘，像素数据也重新变化为对应索引，这样可以方便后续进行形态学操作。

1. void cor2gray(myBmpFile\* bmpImg)
2. {
3. int i,j,k;
4. unsigned char r,g,b;
6. if(bmpImg->Plalette){*//8位图像修改调色板信息*
7. for(i=0;i<256;i++)
8. {
9. r=bmpImg->Plalette[i].rgbRed;
10. g=bmpImg->Plalette[i].rgbGreen;
11. b=bmpImg->Plalette[i].rgbBlue;
12. *//存储分量信息，便于计算，精简表达式*
13. if((bmpImg->Plalette[i].rgbRed=(unsigned char)(0.299\*r+0.587\*g+0.114\*b))>255)
14. bmpImg->Plalette[i].rgbRed=(unsigned char)255;
15. bmpImg->Plalette[i].rgbGreen=bmpImg->Plalette[i].rgbRed;
16. bmpImg->Plalette[i].rgbBlue=bmpImg->Plalette[i].rgbRed;
17. }
18. myRgbQuad \*quad=bmpImg->Plalette;
19. *//原彩色调色板转换为灰度调色板，虽然调色板需要换，*
20. *//但是还需要原调色板来转换像素点数据，所以需要一个指针*
21. bmpImg->Plalette=(myRgbQuad\*)malloc(sizeof(myRgbQuad)\*256);
22. for(i=0;i<256;i++){
23. bmpImg->Plalette[i].rgbRed=i;
24. bmpImg->Plalette[i].rgbGreen=i;
25. bmpImg->Plalette[i].rgbBlue=i;*/////*
26. bmpImg->Plalette[i].rgbReserved=0;
27. }*//新调色板*
29. long height=bmpImg->BIH.biHeight;
30. long width=bmpImg->BIH.biWidth;
31. int offset=0;
32. if(width%4!=0){
33. offset=4-width%4;
34. }
35. long width\_2=width+offset;*//图像高宽，4的倍数，*
36. for(i=0;i<height;i++){
37. for(j=0;j<width\_2-1;j++){
38. bmpImg->imageData[i\*width\_2+j]=(int)quad[(int)bmpImg->imageData[i\*width\_2+j]].rgbRed;
39. *//由于新调色板的顺序排列，新索引就是原索引对应的调色板值。*
40. }
41. }
42. }
43. else{*//24位图像不仅需要需要增加调色板，还需要更改图像信息。*
44. long height=bmpImg->BIH.biHeight;
45. long width=bmpImg->BIH.biWidth;
47. int offset=0;
48. if(width\*3%4!=0){
49. offset=4-width\*3%4;
50. }
51. long width\_=width\*3+offset;*//计算一行实际字节数*
52. if(width%4!=0){
53. offset=4-width%4;
54. }
55. long width\_2=width+offset;*//转为8位一行字节数仍是4的倍数。*
57. bmpImg->BIH.biBitCount=8;*//颜色位数*
58. bmpImg->BFH.bfOffBits=(unsigned long)1078;*//偏移量*
59. bmpImg->BFH.bfSize=width\_2\*height+(unsigned long)1078;*//文件大小*
60. bmpImg->BIH.biSizeImage=width\_2\*height;*//图片实际像素大小*
61. bmpImg->Plalette=(myRgbQuad\*)malloc(sizeof(myRgbQuad)\*256);
62. for(i=0;i<256;i++){
63. bmpImg->Plalette[i].rgbRed=i;
64. bmpImg->Plalette[i].rgbGreen=i;
65. bmpImg->Plalette[i].rgbBlue=i;*/////*
66. bmpImg->Plalette[i].rgbReserved=0;
67. } *//增加调色板*
68. unsigned char\* bmpdata=bmpImg->imageData;*//需要一个指针指向原像素数据*
69. bmpImg->imageData=(unsigned char\*)malloc(sizeof(unsigned char)\*bmpImg->BIH.biSizeImage);
70. *//新的像素数据*
71. for(i=0;i<height;i++){
72. for(j=0;j<width;j++){
73. r=bmpdata[i\*width\_+j\*3];
74. g=bmpdata[i\*width\_+1+j\*3];
75. b=bmpdata[i\*width\_+2+j\*3];
76. if((bmpdata[i\*width\_+j\*3]=(unsigned char)(0.299\*r+0.587\*g+0.114\*b))>255)
77. bmpdata[i\*width\_+j\*3]=(unsigned char)255;
78. *//计算灰度*
79. for(k=0;k<256;k++){
80. if(bmpImg->Plalette[k].rgbRed==bmpdata[i\*width\_+j\*3])
81. bmpImg->imageData[i\*width\_2+j]=k;
82. *//改变灰度索引值*
83. }
84. }
85. for(;j<width\_2;j++){
86. bmpImg->imageData[i\*width\_2+j]=(unsigned char)255;
87. }
88. }
90. }
91. }

2.大津法求取阈值与灰度图像的二值化

使灰度图像二值化，需要确定一个阈值来区分前景和背景，突出我们的目标对象，淡化无关的背景。对于特定的图像，一些固定的阈值可能会有极好的效果，但是固定的阈值不具有普适性，对不同图片效果相差较大；而大津法 特定的图片可能效果不如一些固定阈值，却具有普适性，对于各种图片计算的阈值都可以尽可能的区分前后景。

得到阈值后，遍历像素点数据，大于阈值置255，小于或等于阈值置0，如此完成了灰度图的二值化。

大津法的计算方法：1.计算所有像素点平均灰度值u0。2.计算所有前景像素点占总像素点数的比例w1。3.计算所有前景像素点的平均灰度。4.同2、3，计算背景的比例和平均灰度。

5.根据公式计算方差（由于前景、背景与全景的数值有一定关系，可调整公式减少第四步使程序更简洁）。6.遍历所有可能阈值，找到最大方差对应的阈值。

1. unsigned char get\_Thresh(myBmpFile \*bmpImg)
2. {
3. unsigned char Tnum=0;*//阈值*
4. int i;
5. float var;*//类间方差*
6. float maxvar;*//最大方差*
7. float k=0;*//计算前景平均灰度的中间变量*
8. float w1=0,u1=0;*//w1为前景像素数占总数比例*
9. *//u1为前景像素平均灰度*
10. float u0=0;*//全图像灰度平均值*
12. float gram[256];
13. for(i=0;i<256;i++){
14. gram[i]=0;
15. }*//记录所有灰度值的像素数*
17. unsigned long size=bmpImg->BIH.biSizeImage;*//像素数*
18. unsigned char \*bmpdata=bmpImg->imageData;
20. for(i=0;i<size;i++){
21. gram[bmpdata[i]]++;
22. }*//记录所有灰度值的像素数*
24. for(i=0;i<256;i++){
25. u0+=gram[i]\*i/size;
26. }*//计算总平均灰度*
28. for(i=0;i<256;i++){
29. w1+=gram[i]/size;*//前景比例*
30. k+=i\*gram[i]/size;
31. u1=k/w1;  *//前景平均灰度*
32. var=w1/(1-w1)\*(u1-u0)\*(u1-u0);*//方差计算*
33. if(var>maxvar){
34. maxvar=var;*//最大方差更新*
35. Tnum=i;*//阈值更新*
36. }
37. }
38. return Tnum;
39. }
40. void gray2bin(unsigned char threshold,myBmpFile \*bmpImg)
41. {
42. unsigned long size=bmpImg->BIH.biSizeImage;
43. unsigned char\* bmpdata=bmpImg->imageData;
44. int i;
46. for(i=0;i<size;i++){
47. if(bmpdata[i]<=threshold) bmpdata[i]=0;
48. else bmpdata[i]=255;
49. *//根据阈值向两个极端靠拢*
50. }
51. return ;
52. }

3.膨胀操作与腐蚀操作

两种操作都需要一种结构元来进行对应操作，根据不同图像情况选取不同的形状和大小，此次作业仅了解原理，因此结构元选择九宫格形状方便操作。

膨胀操作和腐蚀操作为对偶操作，在具体代码实现时，除了判断依据和更改数据有所差异，在形式上两者是相同的，也即代码相似性极高。操作过程需要移动结构元逐个覆盖所有像素点，膨胀即为在结构元的每次覆盖范围内，只要存在黑色，中心元就是黑色（或者只要不全为白色，中心就是黑色）；腐蚀即在结构元的每次覆盖范围内，只要存在白色，中心元就是白色（或者只要不全为黑色，中心就是白色）；只需对每一个中心位置，遍历周围九宫格，查看其颜色即可决定中心颜色，注意，操作过程需要复制一份原二值数据，在复制的数据中操作，否则会导致出错。

1. void delation(myBmpFile \*bmp\_d,myBmpFile \*bmpImg)*//膨胀操作*
2. {
3. int i,j,m,n;
4. long width=bmpImg->BIH.biWidth;
5. long height=bmpImg->BIH.biHeight;
6. unsigned long size=bmpImg->BIH.biSizeImage;
7. unsigned char \*datap=bmpImg->imageData;
8. *//所需数据的准备*
10. long offset=0;
11. if(width%4!=0){
12. offset=4-width%4;
13. }
14. long width\_2=width+offset;*//转为8位一行字节数仍是4的倍数。*
15. bmp\_d->BFH=bmpImg->BFH;
16. bmp\_d->BIH=bmpImg->BIH;
17. bmp\_d->Plalette=(myRgbQuad\*)malloc(sizeof(myRgbQuad)\*256);
18. for(i=0;i<256;i++){
19. bmp\_d->Plalette[i].rgbRed=i;
20. bmp\_d->Plalette[i].rgbGreen=i;
21. bmp\_d->Plalette[i].rgbBlue=i;
22. bmp\_d->Plalette[i].rgbReserved=0;
23. }
24. bmp\_d->imageData=(unsigned char\*)malloc(sizeof(unsigned char)\*size);
25. memcpy(bmp\_d->imageData,bmpImg->imageData,size);
26. *//将源bmp文件的信息克隆到目标bmp文件中*
27. for(i=1;i<height-1;i++){
28. for(j=1;j<width-1;j++){
29. int flag=1;*//退出循环以及膨胀标志*
31. for(m=i-1;m<i+2;m++){
32. for(n=j-1;n<j+2;n++){*//遍历结构元内的像素值*
33. if(datap[i\*width\_2+j]==0||datap[m\*width\_2+n]==0){*//结构元内一个为黑，中间元为黑*
34. flag=0;
35. break;
36. }
37. }
38. if(!flag) break;
39. }
40. if(!flag) bmp\_d->imageData[i\*width\_2+j]=0;
41. }
42. }
43. return ;
44. }
45. void erosion(myBmpFile \*bmp\_e,myBmpFile \*bmpImg)*//腐蚀操作*
46. {
47. int i,j,m,n;
48. long width=bmpImg->BIH.biWidth;
49. long height=bmpImg->BIH.biHeight;
50. unsigned long size=bmpImg->BIH.biSizeImage;
51. unsigned char \*datap=bmpImg->imageData;
52. *//所需数据的准备*
54. long offset=0;
55. if(width%4!=0){
56. offset=4-width%4;
57. }
58. long width\_2=width+offset;*//转为8位一行字节数仍是4的倍数。*
59. bmp\_e->BFH=bmpImg->BFH;
60. bmp\_e->BIH=bmpImg->BIH;
61. bmp\_e->Plalette=(myRgbQuad\*)malloc(sizeof(myRgbQuad)\*256);
62. for(i=0;i<256;i++){
63. bmp\_e->Plalette[i].rgbRed=i;
64. bmp\_e->Plalette[i].rgbGreen=i;
65. bmp\_e->Plalette[i].rgbBlue=i;
66. bmp\_e->Plalette[i].rgbReserved=0;
67. }
68. bmp\_e->imageData=(unsigned char\*)malloc(sizeof(unsigned char)\*size);
69. memcpy(bmp\_e->imageData,bmpImg->imageData,size);
70. *//将源bmp文件的信息克隆到目标bmp文件中*
72. for(i=1;i<height-1;i++){
73. for(j=1;j<width-1;j++){
74. int flag=1;*//退出循环以及腐蚀标志*
76. for(m=i-1;m<i+2;m++){
77. for(n=j-1;n<j+2;n++){
78. if(datap[i\*width\_2+j]==255||datap[m\*width\_2+n]==255){*//结构元内一个为白，中间元为白*
79. flag=0;
80. break;
81. }
82. }
83. if(!flag) break;
84. }
85. if(!flag) bmp\_e->imageData[i\*width\_2+j]=255;
86. }
87. }
89. return ;
90. }

4.开操作和闭操作

开操作和闭操作也是对偶操作，实际上也只是膨胀操作和腐蚀操作不同顺序的产物。只要按顺序执行膨胀和腐蚀即可。

1. void opening(myBmpFile \*bmp\_o,myBmpFile \*bmpImg)*//开操作，先腐蚀后膨胀*
2. {
3. myBmpFile bmp\_t;
5. erosion(&bmp\_t,bmpImg);
6. delation(bmp\_o,&bmp\_t);
7. return ;
8. }
9. void closing(myBmpFile \*bmp\_c,myBmpFile \*bmpImg)*//闭操作，先膨胀后腐蚀*
10. {
11. myBmpFile bmp\_t;
12. delation(&bmp\_t,bmpImg);
13. erosion(bmp\_c,&bmp\_t);
14. return ;
15. }

**四、实验环境及运行方法**

本人使用dev-c++ 5.11版C语言工程运行代码，如果电脑上安装有dev-c++,单击文件“bmpwork.dev”即可打开工程文件，点击编译运行的图标或按f11即可。

使用vscode运行的方法：打开程序所处文件夹（包含 bmp.h 、bmp.c以及main.c文件），在终端输入“gcc bmp.c main.c -o main”,敲击enter键后即可编译生成main.exe，再直接在终端输入“.\main”或在文件夹中打开main.exe即可运行程序得到结果。

**注意：**在运行程序前，保证待操作的bmp文件在对应的路径（sample文件夹）中。

**五、实验结果展示**

测试时使用的图片较多，难以一一放在报告中，如果有需要，可以在main.c中改变文件名，输出不同图片结果，此处只放两组图片。

|  |  |  |  |
| --- | --- | --- | --- |
| 原图 | 灰度图 | 二值图 |  |
|  |  |  |  |
| 膨胀 | 腐蚀 | 开 | 闭 |
|  |  |  |  |
| 原图 | 灰度图 | 二值图 |  |
|  |  |  |  |
| 膨胀 | 腐蚀 | 开 | 闭 |
|  |  |  |  |

**六、心得体会**

本次作业我们将灰度图转换为二值图像，还进行了形态学操作，得到了同一图片的不同效果。由于大部分操作为数据计算和比较，相比上一次作业的各种指针内存问题频出来说，基本还算顺利。二值图像只有黑白二色，能表现的信息有限，但是当我们只在意特定的信息时，二值图像及各种形态学操作有助于突出这些信息，更好识别。