

BMP灰度图二值化及形态学操作

实 验 报 告

姓名：欧翌昕

专业：软件工程

学号：3190104783

课程名称：图像信息处理

指导老师：宋明黎

2020~2021秋冬学期 2020 年 11 月 7 日

目录

[1 实验目的和要求 2](#_Toc56025564)

[2 实验内容和原理 2](#_Toc56025565)

[2.1 大律法实现图像二值化 2](#_Toc56025566)

[2.2 腐蚀 3](#_Toc56025567)

[2.3 膨胀 3](#_Toc56025568)

[2.4 开运算 3](#_Toc56025569)

[2.5 闭运算 3](#_Toc56025570)

[3 实验步骤与分析 4](#_Toc56025571)

[4 实验结果 7](#_Toc56025572)

[4.1 原始图 7](#_Toc56025573)

[4.2 二值化结果 8](#_Toc56025574)

[4.3 腐蚀结果 9](#_Toc56025575)

[4.4 膨胀结果 10](#_Toc56025576)

[4.5 开运算结果 11](#_Toc56025577)

[4.6 闭运算结果 12](#_Toc56025578)

[5 心得体会 12](#_Toc56025579)

# 1 实验目的和要求

1.将BMP图像转化为灰度图并进行二值化

2.对二值化图像进行腐蚀

3.对二值化图像进行膨胀

4.对二值化图像进行开运算

5.对二值化图像进行闭运算

# 2 实验内容和原理

## 2.1 大律法实现图像二值化

大律法的核心思想是，选择使得划分出来的前景与背景有最大方差的划分为最优划分。主要公式如下：

 1.背景像素占比

\omega1 = \frac{N1}{Sum}

 2.前景像素占比

\omega2 = 1- \omega1 = \frac{N2}{Sum} =1- \frac{N1}{Sum}

 3.背景的平均灰度值

C:\Users\OUYIXI~1\AppData\Local\Temp\gif.gif

 4.前景的平均灰度值

C:\Users\OUYIXI~1\AppData\Local\Temp\gif-1.gif

 5.0~M灰度区间的灰度累计值

\mu = \mu1*\omega 1 + \mu2*\omega 2

 6.类间方差

g = \omega 1 * (\mu - \mu1)^{2} + \omega 2 * (\mu - \mu2)^{2}

 7.最终简化公式

g = \omega 1 * \omega2 * (\mu1 - \mu2)^{2}

## 2.2 腐蚀

对于输出图像的所有像素点P，调查原图像中对应窗口中的像素集合S，若S中至少有一个0，则P为0。

## 2.3 膨胀

对于输出图像的所有像素点P，调查原图像中对应窗口中的像素集合S，若S中至少有一个1，则P为1。

## 2.4 开运算

先腐蚀后膨胀称为开运算。开运算能够除去孤立的小点，毛刺和小桥，而总的位置和形状不变。

## 2.5 闭运算

先膨胀后腐蚀称为开运算。闭运算能够填平小湖（即小孔），弥合小裂缝，而总的位置和形状不变。闭运算是通过填充图像的凹角来滤波图像的。

# 3 实验步骤与分析

统计各灰度值的个数和比例：

int num[256];  //各灰度值的个数

    double pro[256];  //各灰度值的比例

    memset(num, 0, sizeof(int)\*256);

    int i, j;

    for (i = 0; i < h; i++)

    {

        for (j = 0; j < w\*3-skip; j += 3)

        {

            int t = i\*bytesPerLine + j;

            int B = data[t];

            int G = data[t+1];

            int R = data[t+2];

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

            if (Y < 0) Y = 0;

            if (Y > 255) Y = 255;

            num[Y]++;

        }

    }

    for (i = 0; i < 256; i++)

    {

        pro[i] = (double)num[i]/h/(w\*3-skip)\*3;

    }

大律法确定阈值：

double w1;    //前景像素占比

    double w2;    //背景像素占比

    double exp1;  //前景像素灰度值的数学期望

    double exp2;  //背景像素灰度值的数学期望

    double u1;    //前景的平均灰度值

    double u2;    //背景的平均灰度值

    double g;     //类间方差

    int thresh = 0;

    double MaxVar = 0;

    for (int th = 0; th < 256; th++)

    {

        w1 = w2 = 0;

        exp1 = exp2 = 0;

        u1 = u2 = 0;

        g = 0;

        for (i = 0; i < 256; i++)

        {

            if (i < th)

            {

                w1 += pro[i];

                exp1 += i\*pro[i];

            }

            else

            {

                w2 += pro[i];

                exp2 += i\*pro[i];

            }

        }

        u1 = exp1/w1;

        u2 = exp2/w2;

        g = w1\*w2\*(u1-u2)\*(u1-u2);

        if (MaxVar < g)

        {

            MaxVar = g;

            thresh = th;

        }

    }

根据阈值写入图像数据完成二值化：

    for (i = 0; i < h; i++)

    {

        for (j = 0; j < w\*3-skip; j += 3)

        {

            int t = i\*bytesPerLine + j;

            int B = data[t];

            int G = data[t+1];

            int R = data[t+2];

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

            if (Y <= thresh) Y = 0;

            else Y = 255;

            data[t] = data[t+1] = data[t+2] = (BYTE)Y;

        }

}

以十字形的五个像素为结构元素进行腐蚀操作：

    for (i = 0; i < h-2; i++)

    {

        for (j = 0; j < w\*3-skip-6; j += 3)

        {

            int t = i\*bytesPerLine + j;

            int Y;

            if (data[t] == 0 && data[t+bytesPerLine] == 0 && data[t+2\*bytesPerLine] == 0 && data[t+3] == 0 && data[t+6] == 0) Y = 0;

            else Y = 255;

            data[t] = data[t+1] = data[t+2] = (BYTE)Y;

        }

    }

以十字形的五个像素为结构元素进行膨胀操作：

    for (i = 0; i < h-2; i++)

    {

        for (j = 0; j < w\*3-skip-6; j += 3)

        {

            int t = i\*bytesPerLine + j;

            int Y;

            if (data[t] == 0 || data[t+bytesPerLine] == 0 || data[t+2\*bytesPerLine] == 0 || data[t+3] == 0 || data[t+6] == 0) Y = 0;

            else Y = 255;

            data[t] = data[t+1] = data[t+2] = (BYTE)Y;

        }

    }

# 4 实验结果

## 4.1 原始图



## 4.2 二值化结果



## 4.3 腐蚀结果



## 4.4 膨胀结果



## 4.5 开运算结果



## 4.6 闭运算结果



# 5 心得体会

通过这次实验，我更好的掌握了对BMP文件格式图像的操作方法，同时对二值化图像形态学操作有了更直观的认识。同时也思考了一些老师上课时提到过的问题，当然也有很多地方还存在不足，比如对于一些图片使用局部自适应算法获取阈值会有更好的效果，还有BMP文件的读写其实在第一次作业中已经实现，但当时并没有进项比较好的封装，导致代码复用率比较高，之后会考虑将代码进行重构并进行封装。对这次实验整体非常满意，很有成就感。