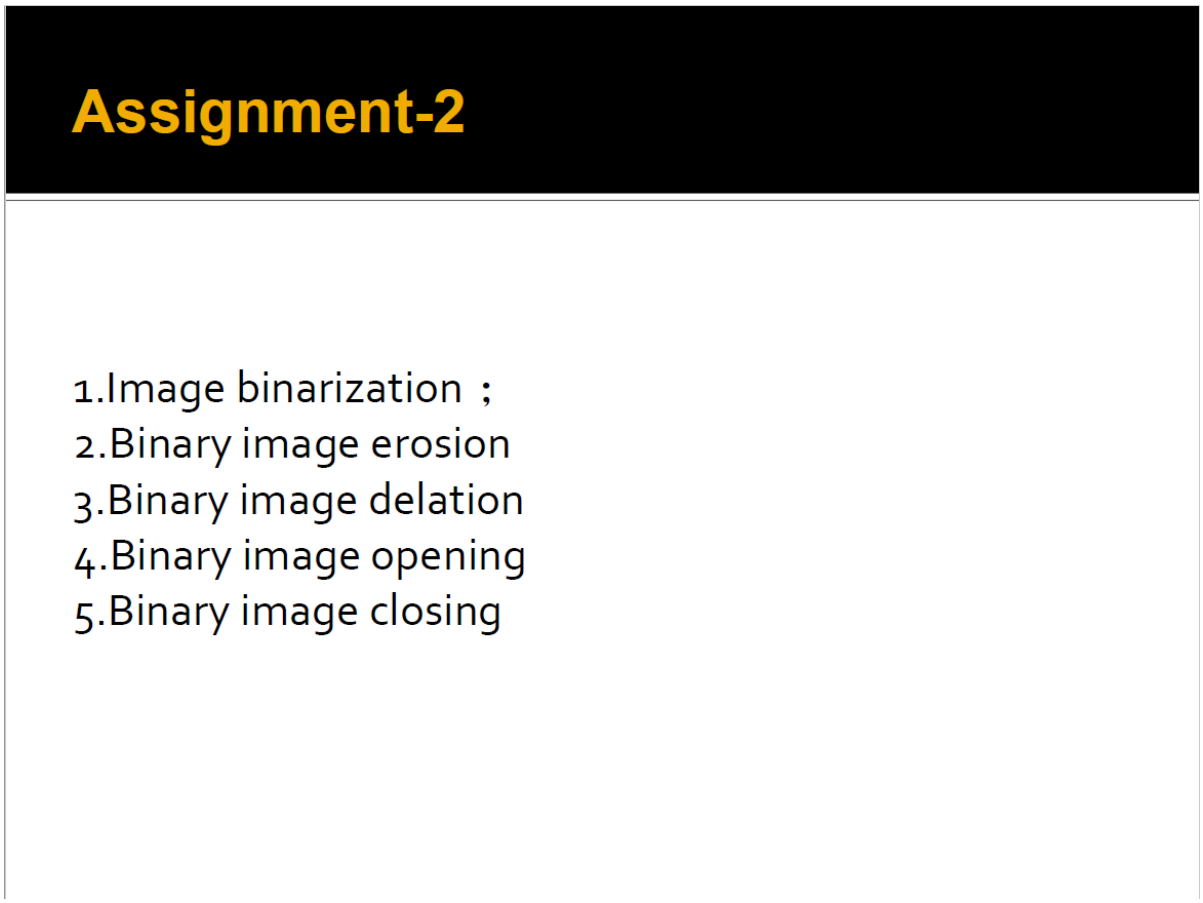
浙江大学实验报告

课程名称：图像信息处理 指导老师：宋明黎 成绩：

实验名称：二值化、腐蚀膨胀以及开闭运算

一、实验目的和要求

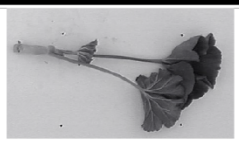


二、实验内容和原理

1、二值化

Binary image

Binary image



Grayscale image

Binary Image: Pixel value is limited to **0** or **1**. Thus, only **1** bit is required for each pixel.
For convenience in programming, we usually use **0** OR **255** instead of **0** OR **1** to represent the binary image.

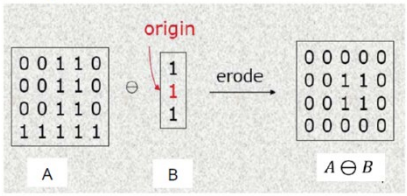
2、腐蚀与膨胀

Erosion

A: Binary image
B: binary template, structure element

$$A \ominus B = \{(x,y)|(B)_{xy} \subseteq A\}$$

Physical meaning: remove boundary, remove unwanted small objects.



Dilation

A: Binary image
B: binary template, called structure element

Dilation: enlarging the foreground

A is dilated by *B*

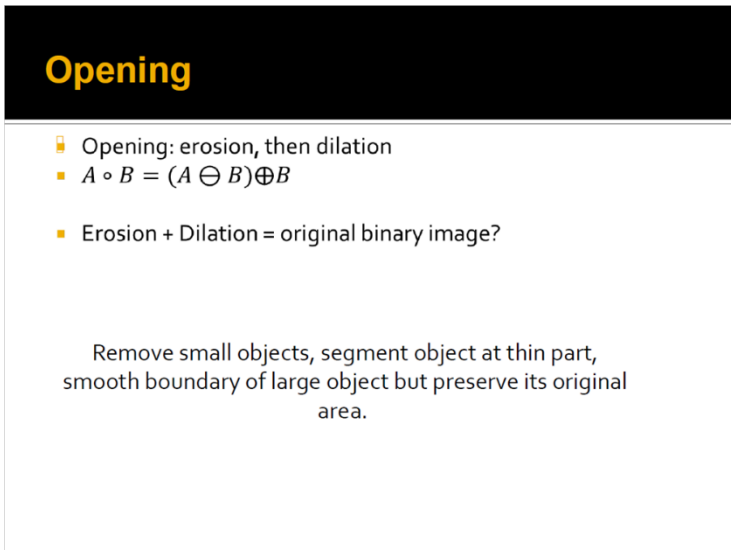
$$A \oplus B = \{z|(B)_z \cap A \neq \emptyset\}$$

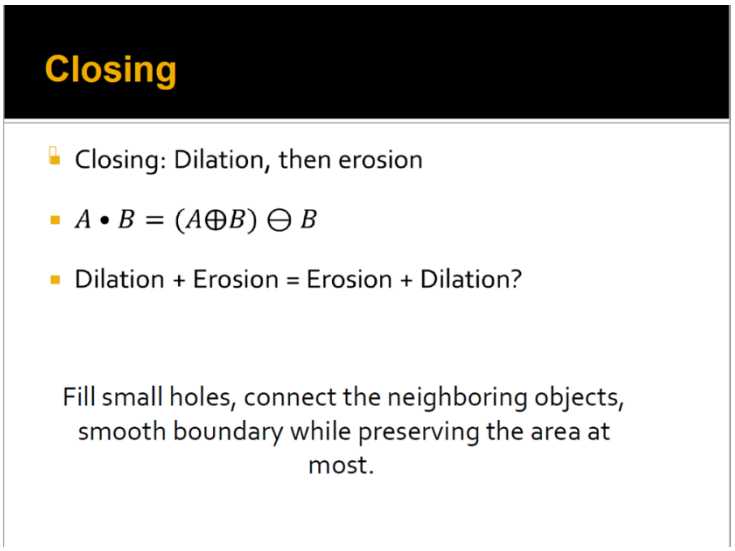
The intersection set between *A* and the translated *B* is not empty

Physical meaning

Dilation adopts the connected background pixels into the foreground, which extends its boundary and fill the holes in the foreground.
And whether "connected" is decided by the structure element.

3、开运算与闭运算





三、实验步骤与分析

1、结构体创建

经上次实验后，我发现在 C 语言 Windosw.h 库中自带描述 bmp 文件结构的结构体，不不再需要额外创建结构体

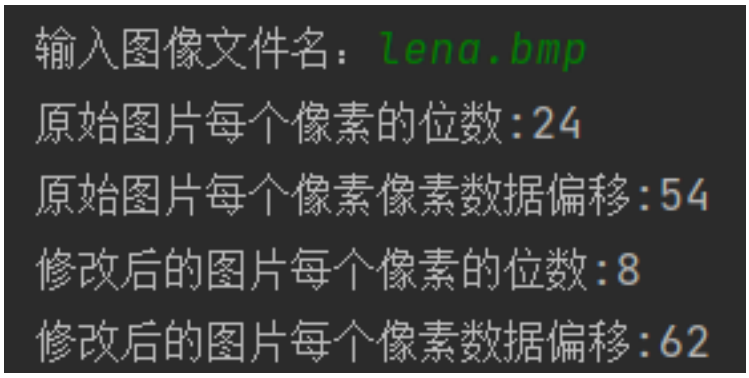
2、bmp 的二值化

上次实验灰度化才用 3 个相同字节一存的 24 位图，本次实验打算实现正规的灰度化，将灰度 bmp 存为 8 位图。

```
/*创建位图文件头，信息头，调色板*/
fileHeader=(BITMAPFILEHEADER *)malloc(sizeof(BITMAPFILEHEADER));
infoHeader=(BITMAPINFOHEADER *)malloc(sizeof(BITMAPINFOHEADER));
ipRGB=(RGBQUAD *)malloc( _Size: 2*sizeof(RGBQUAD));

/*读入源位图文件头和信息头*/
fread(fileHeader,sizeof(BITMAPFILEHEADER), _Count: 1,fpBMP);
fread(infoHeader,sizeof(BITMAPINFOHEADER), _Count: 1,fpBMP);
//经过这两条程序把BMP图像的信息头、文件头赋给fileHeader和infoHeader变量，可以根据fileHeader和infoHeader得到图像的各种属性。
  printf("原始图片每个像素的位数:%d\n" ,infoHeader->biBitCount);
  printf("原始图片每个像素像素数据偏移:%d\n" ,fileHeader->bfOffBits);
//修改信息头
//信息头共有11部分，灰度化时需要修改4部分
infoHeader->biBitCount=8;//转换成二值图后，颜色深度由24位变为8位
infoHeader->biSizeImage=((infoHeader->biWidth + 3) / 4) * 4 * infoHeader->biHeight;//每个像素由三字节变为单字节，同时
infoHeader->biClrUsed=2;//颜色索引表数量，二值图为2
infoHeader->biClrImportant=0;//重要颜色索引为0，表示都重要
//修改文件头
//文件头共有5部分，灰度化时需要修改两部分
fileHeader->bfOffBits=sizeof(BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER)+2*sizeof(RGBQUAD);//数据区偏移量，等于文件头
fileHeader->bfSize=fileHeader->bfOffBits+infoHeader->biSizeImage;//文件大小，等于偏移量加上数据区大小
ipRGB[1].rgbBlue=ipRGB[1].rgbGreen=ipRGB[1].rgbRed=ipRGB[1].rgbReserved=0;//调色板颜色为黑色对应的索引为0
ipRGB[0].rgbBlue=ipRGB[0].rgbGreen=ipRGB[0].rgbRed=200;//白色对应的索引为150-255
```

修改 24 位图中与 8 位图不同的变量：深度、索引表数量、调色板等



测试代码打印结果，变化正确

```
//读取BMP图像的信息头、文件头、BMP调色板到新建的图片
fwrite(fileHeader, sizeof(BITMAPFILEHEADER), _Count: 1, binarization);
fwrite(infoHeader, sizeof(BITMAPINFOHEADER), _Count: 1, binarization);
fwrite(ipRGB, _Size: 2*sizeof(RGBQUAD), _Count: 1, binarization);

/*将彩色图转为二值图*/
a=(unsigned char *)malloc( _Size: (infoHeader->biWidth*3+3)/4*4);//给变量a申请源图每行像素所占大小的空间,考虑四字节对齐问题
c=(unsigned char *)malloc( _Size: (infoHeader->biWidth+3)/4*4);//给变量c申请目标图每行像素所占大小的空间,同样四字节对齐

for(i=0;i<infoHeader->biHeight;i++)//遍历图像每行的循环
    {
    for(j=0;j<((infoHeader->biWidth*3+3)/4*4);j++)//遍历每行中每个字节的循环
        {
        fread( _DstBuf: a+j, _ElementSize: 1, _Count: 1,fpBMP);//将源图每行的每一个字节读入变量a所指向的内存空间
        }
    for(j=0;j<infoHeader->biWidth;j++)//循环像素宽度次,就不会计算读入四字节填充位
        {
        b=(int)(0.114*(float)a[k]+0.587*(float)a[k+1]+0.299*(float)a[k+2]);//a中每三个字节分别代表BGR分量，乘上不同权值转化为灰度值
        if(120>(int)b) //将灰度值转化为二值
            b=1;
        else b=0;
        c[j]=b;   //存储每行的二值
        k+=3;
        }
    fwrite(c, _Size: (infoHeader->biWidth+3)/4*4, _Count: 1, binarization);//将二值像素四字节填充写入文件,填充位没有初始化,为随机值
    k=0;
    }
fclose(binarization);
```

进行二值化操作，将阈值设为 120。由于之后为方便操作需读取文件 binarization，先关闭以只读形式打开的 binarization

2、腐蚀与膨胀

```
if ((binarization = fopen(filename2, _Mode: "rb")) == NULL)
{
    printf( _Format: "打开图片失败");
    exit( _Code: 0);
}

fwrite(fileHeader, sizeof(BITMAPFILEHEADER), _Count: 1, erosion);
fwrite(infoHeader, sizeof(BITMAPINFOHEADER), _Count: 1, erosion);
fwrite(ipRGB, _Size: 2*sizeof(RGBQUAD), _Count: 1, erosion);
fseek(binarization, fileHeader->bfOffBits, _Origin: 0);
unsigned char map[infoHeader->biHeight][(infoHeader->biWidth+3)/4*4];
for(i=0;i<infoHeader->biHeight;i++)//遍历图像每行的循环
    {
    for(j=0;j<((infoHeader->biWidth+3)/4*4);j++)//遍历每行中每个字节的循环
        {
        fread( _DstBuf: map[i]+j, _ElementSize: 1, _Count: 1,binarization);//将源图每行的每一个字节读入变量a所指向的内存空间
        }
    }
```

首先要读入 fileheader、infoheader 和 ipRGB 信息，由于采用最新的 C11 标准，可直接建立二维变长数组 map，将颜色值存入 map

```
    }
for(i=0;i<infoHeader->biHeight;i++)//遍历图像每行的循环
    {
    for(j=0;j<infoHeader->biWidth;j++)//遍历每行中每个字节的循环
        {
        if (j == 0 && i == 0)
        {
            if (map[i][j+1] == 0 || map[i+1][j] == 0)
                map[i][j] = 2;
        }
        else if (j == 0 && i == infoHeader->biHeight - 1)
        {
            if (map[i-1][j] == 0 || map[i][j+1] == 0)
                map[i][j] = 2;
        }
        else if (j == 0)
        {
            if (map[i][j+1] == 0 || map[i+1][j] == 0 || map[i-1][j] == 0)
                map[i][j] = 2;
        }
        else if (j == infoHeader->biWidth - 1 && i == 0)
        {
            if (map[i][j-1] == 0 || map[i+1][j] == 0)
                map[i][j] = 2;
        }
        else if (j == infoHeader->biWidth - 1 && i == infoHeader->biHeight - 1)
        {
            if (map[i-1][j] == 0 || map[i][j-1] == 0)
                map[i][j] = 2;
        }
        else if (j == infoHeader->biWidth - 1)
        {
            if (map[i][j-1] == 0 ||  map[i+1][j] == 0 || map[i-1][j] == 0)
                map[i][j] = 2;
        }
        else if (i == 0)
        {
            if (map[i][j-1] == 0 || map[i][j+1] == 0 || map[i+1][j] == 0)
                map[i][j] = 2;
        }
        else if (i == infoHeader->biHeight - 1)
        {
            if (map[i-1][j] == 0 || map[i][j-1] == 0 || map[i][j+1] == 0)
                map[i][j] = 2;
        }
        else
        {
            if (map[i-1][j] == 0 || map[i][j-1] == 0 || map[i][j+1] == 0 || map[i+1][j] ==
                map[i][j] = 2;
        }
        }
    }
for(i=0;i<infoHeader->biHeight;i++)//遍历图像每行的循环
    {
    for(j=0;j<infoHeader->biWidth;j++)//遍历每行中每个字节的循环
        if (map[i][j] == 2)
            map[i][j] = 0;
    }
for(i=0;i<infoHeader->biHeight;i++)//遍历图像每行的循环
    fwrite(map[i], Size: (infoHeader->biWidth+3)/4*4, Count: 1, erosion);
fclose(erosion);

/*********************************************************************/
```

erosion 的主要算法如图，设计用十字型的结构元素去腐蚀全图，被腐蚀的将值先设为 2 以防无限腐蚀，最后将 2 设为 0，达成腐蚀

dilation 算法同理，只不过将判断值中的 0 设为 1 罢了

3、开运算与闭运算

由于之前已经得到腐蚀图与膨胀图了，我们可以直接读取得到的图进行膨胀/腐蚀来得到开运算/闭运算的结果。即：对开运算，读入 erosion.bmp，使用 dilation 算法，即可得到开运算结果；对闭运算，读入 dilation.bmp，使用 erosion 算法，即可得到闭运算结果

4、结束

```
/*释放内存空间，关闭文件*/
free(fileHeader);
free(infoHeader);
free(ipRGB);
free(a);
free(c);
fclose(fpBMP);
fclose(binarization);
printf( _Format: "Done\n");
return 0;
```

注意释放内存以及关闭文件

四、实验环境及运行方法

编译环境：C 语言，使用最新 C11 标准。ide 使用 clion，利用 cmakelist 编译。用 dev 可直接打开 main.c 源文件进行编译。由于 clion 本身默认会对中文输出乱码，故修改为 UTF-8 解码及 GBK 编码，dev 打开可能出现乱码，visual studio 打开无影响

测试方法：断点单步测试、使用 UltraEdit 查看画出的 bmp 的二进制数据来找输出算法的问题

五、实验结果展示

原图

二值化后

erosion　　dilation

opening　　closing

生成的图片与编译文件在同一路径下，使用 clion 时在 cmake-build-debug 目录下

六、心得体会

本次 bmp 处理加深了我对 bmp 文件的认识，了解了二值化相关知识，做出的结果让我颇有成就感。