```
#include <opencv.hpp>
#include "function.h"

using namespace cv;
using namespace std;


/*************************************************************************************************************************************
          第四周练习1：
图像形态学处理，分别对图像进行腐蚀、膨胀、开运算、闭运算
*************************************************************************************************************************************/
void morphology()
{
//读取图片并转化为灰度图
Mat srcMat = imread("d:\\coin.png",0);

//判断读取图片是否失败
if (srcMat.empty()) {
cout << "fail to read pic!" << endl;
return;
}
//定义图像容器
Mat thresh_Mat;
Mat dilate_Mat;
Mat erode_Mat;
Mat open_Mat;
Mat close_Mat;

//二值化
threshold(srcMat, thresh_Mat, 100, 255, THRESH_OTSU);

/***********************************************
getStructuringElement（） 函数模型：
getStructuringElement(int shape, Size ksize, Point anchor = Point(-1,-1));

参数介绍：
. int shape: 这个函数的第一个参数表示内核的形状，有三种形状可以选择。矩形：MORPH_RECT;交叉形：MORPH_CROSS;椭圆形：MORPH_ELLIPSE;
. Size ksize: 内核的尺寸
. Point anchor: 锚点的位置
***********************************************/
//定义结构元素
Mat element = getStructuringElement(MORPH_RECT, Size(5, 5), Point(-1, -1));

//腐蚀
/***********************************************
erode（） 函数模型：
 erode( InputArray src, OutputArray dst, InputArray kernel,
                          Point anchor = Point(-1,-1), int iterations = 1,
                          int borderType = BORDER_CONSTANT,
                          const Scalar& borderValue = morphologyDefaultBorderValue() );

参数介绍：
. InputArray src: Mat类，通道数量不限，但深度应为CV_8U,CV_16U...
. OutputArray dst: 输出图像，需要有和原图片一样的尺寸和类型
. InputArray kernel: 腐蚀操作的内核，一般用3*3的核
. Point anchor:锚的位置，一般用（-1，-1）
.int iterations:使用函数的次数
.int borderType:用于推断图像外部像素的某种边界模式
. const Scalar& borderValue:边界为常数时的边界值

***********************************************/

erode(thresh_Mat, erode_Mat, element, Point(-1,-1),1);

//膨胀
/***********************************************
dialte（） 函数模型：
dilate( InputArray src, OutputArray dst, InputArray kernel,
                          Point anchor = Point(-1,-1), int iterations = 1,
                          int borderType = BORDER_CONSTANT,
                          const Scalar& borderValue = morphologyDefaultBorderValue() );

参数介绍：
. InputArray src: Mat类，通道数量不限，但深度应为CV_8U,CV_16U...
. OutputArray dst: 输出图像，需要有和原图片一样的尺寸和类型
. InputArray kernel: 腐蚀操作的内核，一般用3*3的核
. Point anchor:锚的位置，一般用（-1，-1）
.int iterations:使用函数的次数
.int borderType:用于推断图像外部像素的某种边界模式
. const Scalar& borderValue:边界为常数时的边界值

***********************************************/
dilate(thresh_Mat, dilate_Mat, element, Point(-1, -1), 1);

//开运算
/***********************************************
 morphologyEx（） 函数模型：
 morphologyEx( InputArray src, OutputArray dst,
                               int op, InputArray kernel,
                               Point anchor = Point(-1,-1), int iterations = 1,
                               int borderType = BORDER_CONSTANT,
                               const Scalar& borderValue = morphologyDefaultBorderValue() );
```

参数介绍：
. InputArray src: Mat类，通道数量不限，但深度应为CV_8U,CV_16U...
. OutputArray dst: 输出图像，需要有和原图片一样的尺寸和类型
. int op:表示形态学运算的类型，如MORPH_OPEN、MORPH_CLOSE分别代表开运算和闭运算
. InputArray kernel: 腐蚀操作的内核，一般用3*3的核
. Point anchor:锚的位置，一般用（-1，-1）
. int iterations:使用函数的次数
. int borderType:用于推断图像外部像素的某种边界模式
. const Scalar& borderValue:边界为常数时的边界值

**/
```
morphologyEx(thresh_Mat, open_Mat, MORPH_OPEN, element, Point(-1, -1), 1);

// 闭运算
morphologyEx(thresh_Mat, close_Mat, MORPH_CLOSE, element, Point(-1, -1), 1);


//显示结果
imshow("thresh_Mat", thresh_Mat);
imshow("erode_Mat", erode_Mat);
imshow("dilate_Mat", dilate_Mat);
imshow("open_Mat", open_Mat);
imshow("close_Mat", close_Mat);
waitKey(0);
}
```

/**
第四周练习2：
连通域标记
**/
```
void connectedwithstats()
{
//读取图片并转化为灰度图
Mat srcMat = imread("d:\\coin.png");

//判断读取图片是否失败
if (srcMat.empty()) {
cout << "fail to read pic!" << endl;
return;
}

//转化为灰度图
Mat gryMat;
cvtColor(srcMat, gryMat, COLOR_BGRA2GRAY);

//定义图像容器
Mat stats;
Mat centroids;
Mat labels;
Mat thresh_Mat;

//大津法处理图像
threshold(gryMat, thresh_Mat, 100, 255, THRESH_OTSU);


//进行连通域标记
/**********************************************
connectedComponentsWithStats（） 函数模型：
connectedComponentsWithStats(InputArray image, OutputArray labels,
                                               OutputArray stats, OutputArray centroids,
                                               int connectivity = 8, int ltype = CV_32S);
```


参数介绍：
. InputArray image: 输入8位单通道二值图像；
. OutputArray labels: 输出和原图image一样大的标记图，label对应于表示是当前像素是第几个轮廓，背景置0
. OutputArray stats:输出nccomps（标签数）×5的矩阵 ，表示每个连通区域的外接矩形和面积（pixel）
. OutputArray centroids: 对应的是轮廓的中心点。nccomps×2的矩阵 表示每个连通区域的质心
. int connectivity:使用8邻域或者4邻域
. int ltype:输出标签的数据类型

**/
```
int nComp = connectedComponentsWithStats(thresh_Mat, labels, stats, centroids, 8, CV_32S);

//减去背景0，并输出
cout << "硬币个数为：" << nComp - 1 << endl;

//对识别出的连通域加最小外接边框
for (int i = 1;i < nComp;i++)
{
//定义Rect类
Rect bandbox;
bandbox.x = stats.at<int>(i, 0);
bandbox.y = stats.at<int>(i, 1);

bandbox.width = stats.at<int>(i, 2);
bandbox.height = stats.at<int>(i, 3);
//画出矩形
/**********************************************
rectangle（） 函数模型：
rectangle(CV_IN_OUT Mat& img, Rect rec,
                          const Scalar& color, int thickness = 1,
                          int lineType = LINE_8, int shift = 0);
```

```
参数介绍:
. ICV_IN_OUT Mat& img: CV_IN_OUT Mat& img
. Rect rec: Rect类成员（包含矩形的左上角坐标以及长宽）
. const Scalar& color:输出颜色信息
. int thickness: 表示线的粗细
. int lineType :邻接关系，一般设置默认值
. int shift: 偏移，一般设0
***********************************************/
rectangle(thresh_Mat, bandbox, 255, 1, 8, 0);
}

}



/*****************************************************************************************************************************
第四周练习3:
原点计数
*****************************************************************************************************************************/

void origincount()
{
//读取图片并转化为灰度图
Mat srcMat = imread("d:\\1.jpg");

//判断读取图片是否失败
if (srcMat.empty()) {
cout << "fail to read pic!" << endl;
return;
}

//转化为灰度图
Mat gryMat;
cvtColor(srcMat, gryMat, COLOR_BGRA2GRAY);

//反色
gryMat = 255 - gryMat;

Mat stats;
Mat centroids;
Mat labels;
Mat thresh_Mat;
Mat erode_Mat;

//大津法处理图像
threshold(gryMat, thresh_Mat, 100, 255, THRESH_OTSU);


//定义结构元素
Mat element = getStructuringElement(MORPH_RECT, Size(5, 5), Point(-1, -1));

//对图像进行腐蚀处理，只保留要求的点
erode(thresh_Mat, erode_Mat, element, Point(-1, -1), 2);



//进行连通域标记
int nComp = connectedComponentsWithStats(erode_Mat, labels, stats, centroids, 8, CV_32S);

//减去背景，并输出个数
cout << "原点个数为: " << nComp - 1 << endl;

}

/*****************************************************************************************************************************
第四周练习4:
回型针计数
*****************************************************************************************************************************/
void clipcount()
{
//读取图片并转化为灰度图
Mat srcMat = imread("d:\\clip.png");

//判断读取图片是否失败
if (srcMat.empty()) {
cout << "fail to read pic!" << endl;
return;
}

//转化为灰度图
Mat gryMat;
cvtColor(srcMat, gryMat, COLOR_BGRA2GRAY);

//反色
gryMat = 255 - gryMat;

Mat stats;
Mat centroids;
Mat labels;
Mat thresh_Mat;
Mat open_Mat;

//大津法处理图像
```

```
threshold(gryMat, thresh_Mat, 100, 255, THRESH_OTSU);


//定义结构元素
Mat element = getStructuringElement(MORPH_RECT, Size(3, 3), Point(-1, -1));

//对图像进行开运算处理，消除一些杂点
morphologyEx(thresh_Mat, open_Mat, MORPH_OPEN, element, Point(-1, -1), 1);



//进行连通域标记
int nComp = connectedComponentsWithStats(open_Mat, labels, stats, centroids, 8, CV_32S);


//比较长宽比，筛选掉干扰连通域
for (int i = 1;i < nComp;i++)
{
int width = stats.at<int>(i, 2);
int height = stats.at<int>(i, 3);
int ratio =  height/ width;
if (ratio > 10)
{
nComp--;
}
}

//减去背景连通域，输出回形针格个数
cout << "回型针个数为：" << nComp - 1 << endl;

}
```