#include<opencv2/opencv.hpp>

#include <iostream>

//包含相关库

using namespace std;

using namespace cv;

VideoCapture capture; //知道这是个可以用于抓捕实时图像的东西就好

Mat srcImage; //我习惯于把源图像定义为全局变量，便于多文件和同文件里的函数直接使用

int main()

{

Mat srcImage = imread("SAST.jpg",0);//读入一个灰度图像到srcImage中

//图像的模糊处理

boxFilter(srcImage,dstImage,-1,Size(3,3));//方框滤波过程，先进行图像归一化，模糊处理，为边缘提取作铺垫

blur(srcImage,dstImage,Size(5,5));//均值滤波

GaussianBlur(srcImage,dstImage,Size(3,3),0,0);//高斯滤波

medianBlur(srcImage,dstImage,7);//中值滤波，防止出现中心频率远高于边缘的点

//在这里不再使用双边滤波，因为双边滤波非常慢

//图像的锐化

Mat grayImage, imgCanny;

cvtColor(srcImage, grayImage, COLOR\_BGR2GRAY);//将原图像转换为灰度图像

Canny(grayImage, imgCanny, 50, 150, 3);//进行canny边缘检测

//sobel算子

Sobel(srcIamge, grad\_x, CV\_16S, 1, 0, 3, 1, 1, BORDER\_DEFAULT);//求X方向梯度

convertScaleAbs(grad\_x, abs\_grad\_x);//使用线性变换转换输入数组元素成8位无符号整型

Sobel(grayImage, grad\_y, CV\_16S, 0, 1, 3, 1, 1, BORDER\_DEFAULT);//求Y方向梯度

convertScaleAbs(grad\_y, abs\_grad\_y);//使用线性变换转换输入数组元素成8位无符号整型

addWeighted(abs\_grad\_x, 0.5, abs\_grad\_y, 0.5, 0,dstIamge);//合并梯度(近似)

//图像的合成

//轮廓检测(寻找)

vector<vector<Point>> contours;

findContours(imgCanny, contours, RETR\_EXTERNAL, CHAIN\_APPROX\_NONE);

drawContours(srcImage, contours, index, color, 3, 8, hierarchy);

}

Mat imread(

  const string& filename,//需要载入的图片的路径名

  int flags=1); //指定加载图像的颜色类型

imshow("const string&winname",//显示图像的窗口标识名称

   srcImage); //所显示的Mat型变量的图像[p]

//方框滤波

boxFilter(

  InputArray src, //输入图像

  OutputArray dst, //输出图像

  int ddepth, //输出图像的深度，-1代表使用原图的深度

  Size ksize(n,n), //内核尺寸，n越大效果越明显

  Point anchor=Point(-1,-1), //锚点，若点坐标为负值，则表示锚点在核的核心

  bool normalize=true, //ture表示被归一化，false表示没有被归一化

  int borderType=BORDER\_DEFAULT);//用于推断图像外部像素的某种边界模式，一般不用管它

//均值滤波

blur(

  InputArray src, //输入图像

  OutputArray dst, //输出图像

  Size ksize(n,n), //内核尺寸，n越大效果越明显

  Point anchor=Point(-1,-1), //锚点，若点坐标为负值，则表示锚点在核的核心

int borderType=BORDER\_DEFAULT); //用于推断图像外部像素的某种边界模式，一般不用管它

//高斯滤波

GaussianBlur(

boxFilter(

  InputArray src, //输入图像

  OutputArray dst, //输出图像

  Size ksize(n,n), //内核尺寸，n越大效果越明显

  double sigmaX, //高斯核函数在X方向的标准偏差

  double sigmaY=0; //高斯核函数在Y方向的标准偏差

  int borderType=BORDER\_DEFAULT);//用于推断图像外部像素的某种边界模式，一般不用管它

/\*其中，若sigmaY为零，就将它设为sigmaX；

若sigmaX和sigmaY都是0，那么就由ksize.width和ksize.height计算出来\*/

//中值滤波

void medianBlur(

  InputArray src, //输入图像

  OutputArray dst, //输出图像

  int ksize); //注意啊！！！这里是int类型的，不是Size类的啊！！！

//还有！！！这里必须要填大于1的奇数啊！！！不然会报错啊！！！

//边缘检测（canny算子）

void Canny(

  InputArray image, //输入图像，需为单通道8位图像

  OutputArray edges, //输出的边缘图

  double threshold1, //低阈值

double threshold2, //高阈值

  int apertureSize=3, //使用的Sobel算子孔径大小

  bool L2gradient=false);//计算图像梯度幅值的标识，有默认值false

Canny(src,dst,n,n\*3,3);

//边缘检测（sobel）

void Sobel (

  InputArray src, //输入图像

  OutputArray dst,//目标图像

  int ddepth, //输出图像的深度

  int dx, //x方向上的差分阶数

  int dy, //y方向上的差分阶数

  int ksize=3, //表示Sobel核的大小,必须取1，3，5或7

 //一般都是用ksize×ksize内核来计算导数的

 //但当ksize=1，往往用3×1或1×3的内核，且这种情况下并没有进行高斯平滑

  double scale=1, //计算导数值时可选的缩放因子，默认情况下是没有应用缩放的

  double delta=0, //表示在结果存入目标图之前可选的delta值

  int borderType=BORDER\_DEFAULT );//边界模式

void cv::convertScaleAbs(

cv::InputArray src, // 输入数组

cv::OutputArray dst,// 输出数组

double alpha = 1.0, // 乘数因子

double beta = 0.0); // 偏移量

void addWeighted(

  InputArray src1,//要叠加的第一个图像

  double alpha, //第一个叠加图像的权重

  InputArray src2,//要叠加的第二个图像，需和第一个拥有同样的尺寸和通道

  double beta, //第二个叠加图像的权重

  double gamma, //累加到权重总和上的偏差

  OutputArray dst,//输出图像

  int dtype = -1);//输出阵列的深度

//寻找轮廓

void findContours(

  InputArray image, //输入图像，需为8位单通道的二进制图像

  OutputArray contours, //存储检测到的轮廓，每个轮廓存储为一个点向量

  OutputArray hierarchy, //包含图像的拓扑信息，作为轮廓数量的表示。

 //每个轮廓对应4个hierarchy元素，分别表示后一个轮廓、

//前一个轮廓、父轮廓、内嵌轮廓的索引编号。

//若没有对应项，对应的hierarchy[i]值设置为负数。

  int mode, //轮廓检索模式

  int method, //轮廓的近似办法

  Point offset=Point()); //每个轮廓点的可选偏移量

//绘制轮廓

void drawContours(

  InputArray image, //需要绘制的图像

  OutputArray contours,//所有输入轮廓，每个轮廓存储为一个点向量

  int contourIdx, //轮廓绘制的指示变量，若其为负值，则绘制所有轮廓

  const Scalar&color, //轮廓的颜色

  int thickness=1, //轮廓线条的粗细度，若其为负值，则绘制在轮廓内部

int lineType=8, //线条的类型

  InputArray hierarchy=noArray(),//可选的层次结构信息

int maxLevel= INT\_MAX,//绘制轮廓的最大等级

  Point offset=Point());//可选的轮廓偏移参数

minAreaRect() 寻找最小包围矩形

boundingRect() 返回外部矩形边界

minEnclosingCircle() 寻找最小包围圆形

fitEillipse() 用椭圆拟合二维点集

approxPolyDP() 用指定精度逼近多边形曲线

contourArea() 计算整个轮廓或部分轮廓的面积

arcLength() 计算封闭轮廓的周长或曲线的长度

难点：很多很多，东西太难学习，任务量太大，基本上相当于学了一个升级版c++;