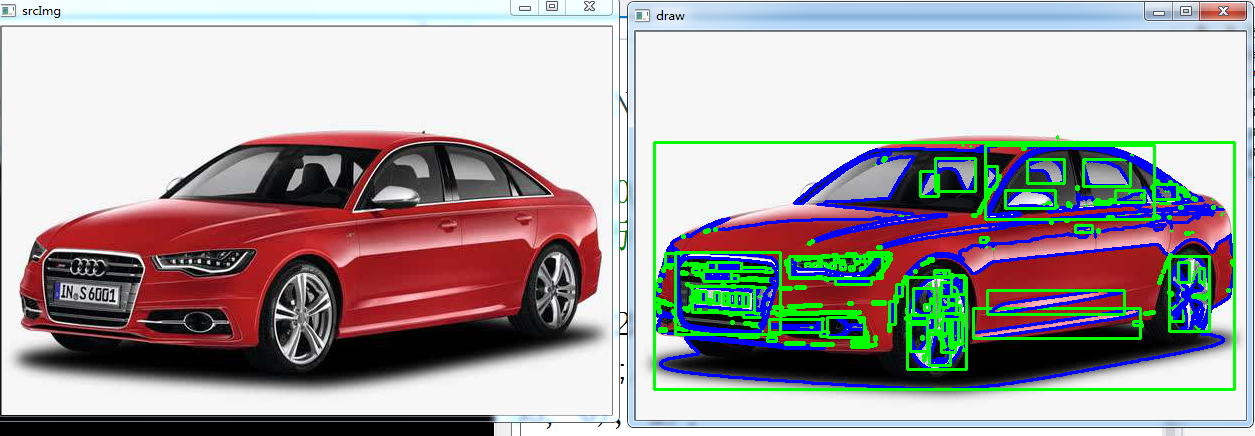
# 绘制图像的最小旋转矩形

## 1、 复习： 轮廓



#include<opencv2\opencv.hpp>

#include<iostream>

using namespace std;

using namespace cv;

int main()

{

//1.查找轮廓时，一般使用二值化图像，进行阈值化处理或者Canny边缘检测

Mat srcImg = imread("e:\\aodi.jpg", 1);

imshow("srcImg", srcImg);

Mat copyImg = srcImg.clone();//将原图像备份（查找轮廓会改变原图像）

cvtColor(srcImg, srcImg, COLOR\_BGR2GRAY);//转换成灰度图（阈值化需要图像会灰度图）

threshold(srcImg, srcImg, 100, 255, THRESH\_BINARY\_INV); //阈值化并保证黑色背景白色轮廓

imshow("threshold", srcImg);

//查找轮廓并绘制轮廓

vector<vector<Point>> contours;//存放检测到的轮廓,每个轮廓均由点组成

vector<Vec4i> hierarchy;

findContours(srcImg, contours, RETR\_TREE, CHAIN\_APPROX\_SIMPLE);

//访问每个轮廓中的每个点，绘制在原图上

// drawContours(copyImg, contours, -1, Scalar(0, 255, 0), 1);

for (int i = 0; i < contours.size(); i++)//遍历每个轮廓

{

drawContours(copyImg, contours,i, Scalar(0, 0, 255), 2);//用蓝色绘制轮廓

Rect rect = boundingRect(Mat(contours[i]));//计算点集的最外面矩形边界

rectangle(copyImg, rect, Scalar(0, 255, 0), 2);//用绿色绘制 矩形

}

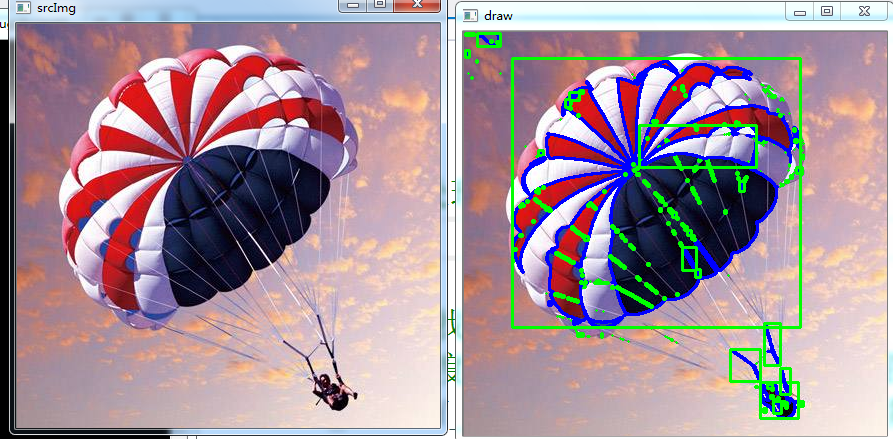
imshow("draw", copyImg);

waitKey(0);

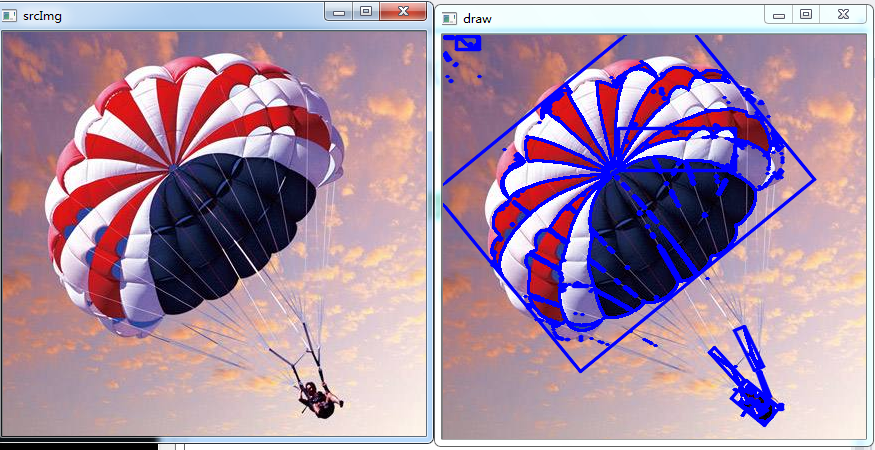
return 0;

}

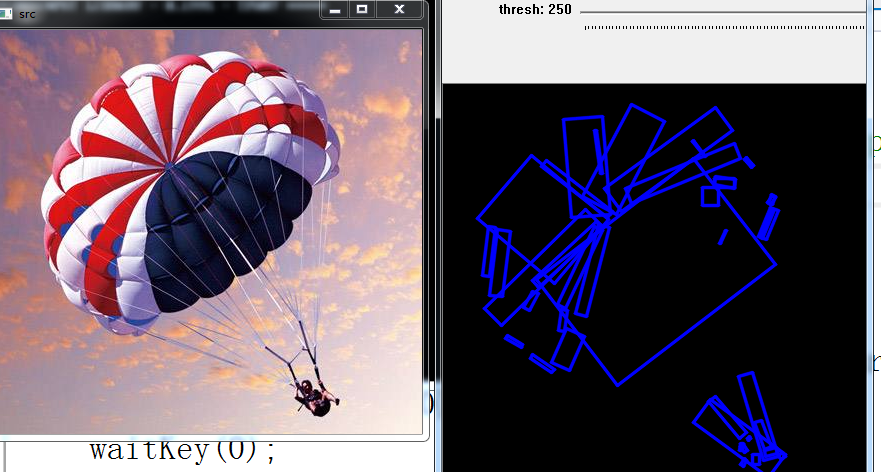
### 尝试：换成图像e:\\san.png试试



希望：



更希望通过滑动滑块，实现旋转矩形的标记



## 2学习绘制旋转矩形

### 2.1、类RotatedRect

class RotatedRect

{

public:

RotatedRect(); //构造函数

RotatedRect(const Point2f& center, const Size2f& size, float angle);

RotatedRect(const CvBox2D& box);

void points(Point2f pts[]) const; //返回矩形的4个顶点 ，存在参数pts中

Rect boundingRect() const; //返回包含旋转矩形的最小矩形

operator CvBox2D() const; //转换到旧式的cvbox2d结构

float angle; //旋转角度，当角度为0、90、180、270等时，矩形就成了一个直立的矩形

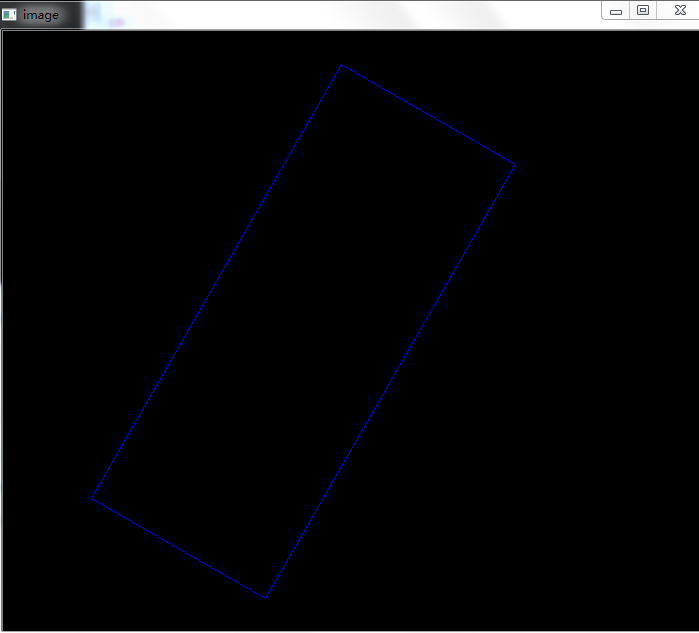
Point2f center; //矩形的质心

Size2f size; //矩形的边长

}

OpenCV中，坐标的原点在左上角，与x轴平行的方向为角度为0，逆时针旋转角度为负，顺时针旋转角度为正。

### 2.2代码：绘制一个旋转矩形



#include<opencv2\opencv.hpp>

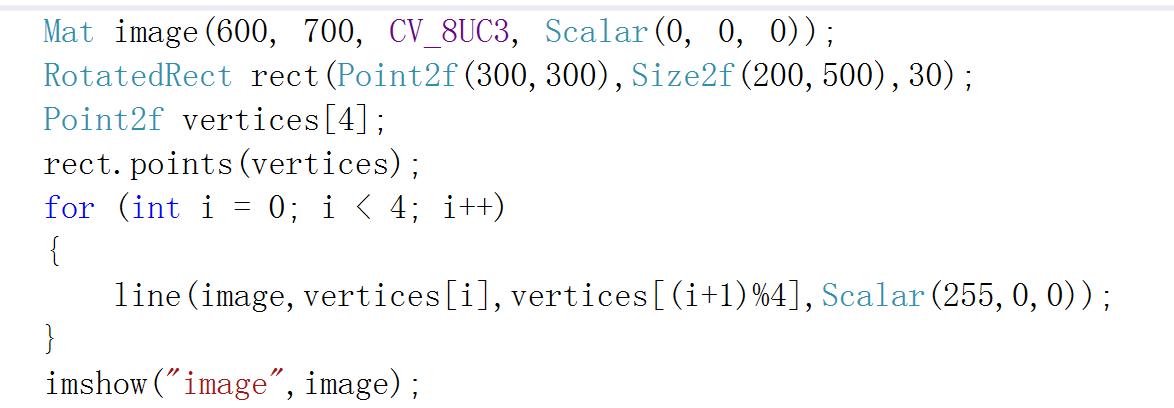
#include<iostream>

using namespace std;

using namespace cv;

int main()

{

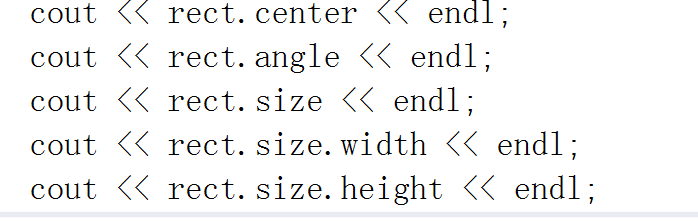


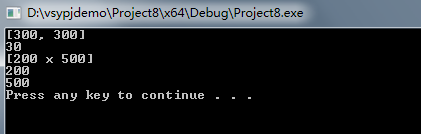
waitKey(0);

return 0;

}

在imshow下面输入以下代码：

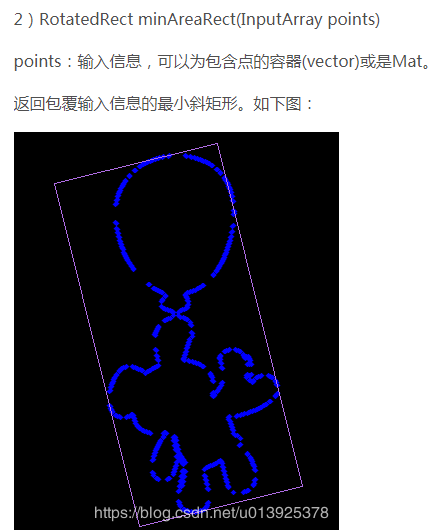
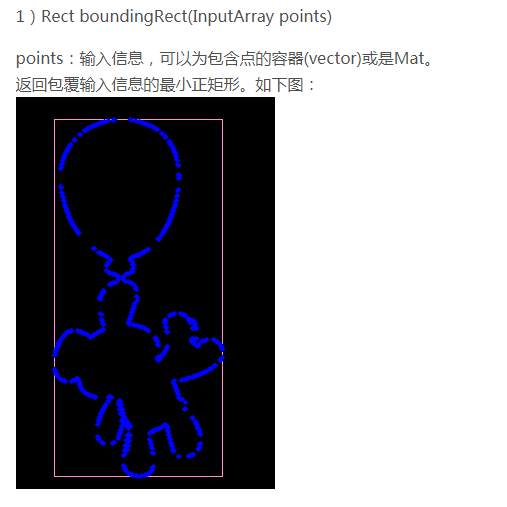
 system("pause");



## 3、函数：minAreaRect

当得到对象轮廓后，可用boundingRect()得到包覆此轮廓的最小正矩形

minAreaRect()得到包覆轮廓的最小面积的旋转形。



## 4绘制点集的最小面积旋转矩形

#include<opencv2\opencv.hpp>

#include<iostream>

using namespace std;

using namespace cv;

int main()

{

Mat srcImage(Size(600, 600), CV\_8UC3, Scalar(0));

RNG &rng = theRNG();

char key;

while (1)

{

key = waitKey(0);

if (key == 27) break;

else srcImage = Scalar::all(0);

//随机生成一些点,首先就是随机生成点的总数量

int g\_nPointCount = rng.uniform(3, 30);

//接下来就是随机生成一些点的坐标

vector<Point> points;

for (int i = 0; i < g\_nPointCount; i++)

{

Point midPoint;

midPoint.x = rng.uniform(srcImage.cols / 4, srcImage.cols \* 3 / 4);

midPoint.y = rng.uniform(srcImage.rows / 4, srcImage.rows \* 3 / 4);

points.push\_back(midPoint);

}

//显示刚刚随机生成的那些点

for (int i = 0; i < g\_nPointCount; i++)

{

circle(srcImage, points[i], 0, Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.uniform(0, 255)), 3);

}

//补充 一个轮廓 最下旋转矩形 代码

imshow("【绘制结束后的图像】", srcImage);

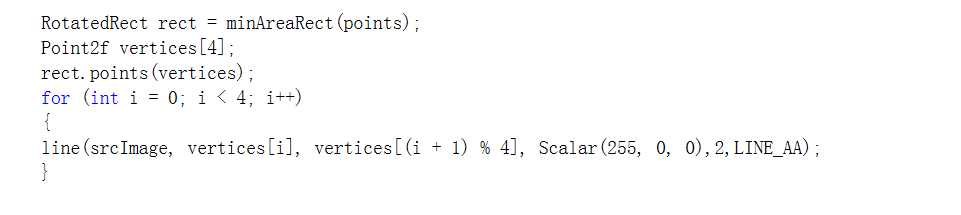
}

waitKey(0);

return 0;

}

### 代码参考



## 5、绘制图像的最小面积旋转矩形

#include<opencv2\opencv.hpp>

#include<iostream>

using namespace std;

using namespace cv;

int main()

{

//1.查找轮廓时，一般使用二值化图像，进行阈值化处理或者Canny边缘检测

Mat srcImg = imread("e:\\aodi.jpg", 1);

imshow("srcImg", srcImg);

Mat copyImg = srcImg.clone();//将原图像备份（查找轮廓会改变原图像）

cvtColor(srcImg, srcImg, COLOR\_BGR2GRAY);//转换成灰度图（阈值化需要图像会灰度图）

threshold(srcImg, srcImg, 100, 255, THRESH\_BINARY\_INV); //阈值化并保证黑色背景白色轮廓

imshow("threshold", srcImg);

//查找轮廓并绘制轮廓

vector<vector<Point>> contours;//存放检测到的轮廓,每个轮廓均由点组成

vector<Vec4i> hierarchy;

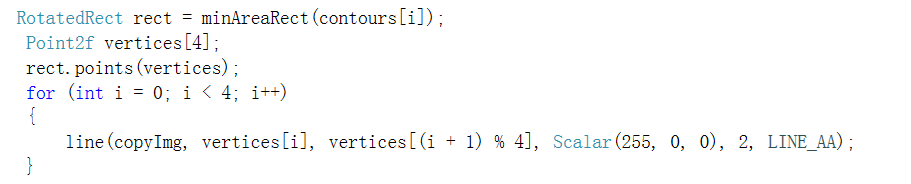
findContours(srcImg, contours, RETR\_TREE, CHAIN\_APPROX\_SIMPLE);

//访问每个轮廓中的每个点，绘制在原图上

// drawContours(copyImg, contours, -1, Scalar(0, 255, 0), 1);

for (int i = 0; i < contours.size(); i++)//遍历每个轮廓

{



}

imshow("draw", copyImg);

waitKey(0);

return 0;

}

## 6、完成实验内容