opencv自动光学检测、目标分割和检测的详细分析

发布时间：2020-07-21 09:18:22

作者：小猪

来源：亿速云　　阅读：64

这篇文章主要讲解了opencv自动光学检测、目标分割和检测的详细分析，内容清晰明了，对此有兴趣的小伙伴可以学习一下，相信大家阅读完之后会有帮助。

步骤如下：

1.图片灰化；

2.中值滤波 去噪

3.求图片的光影（自动光学检测）

4.除法去光影

5.阈值操作

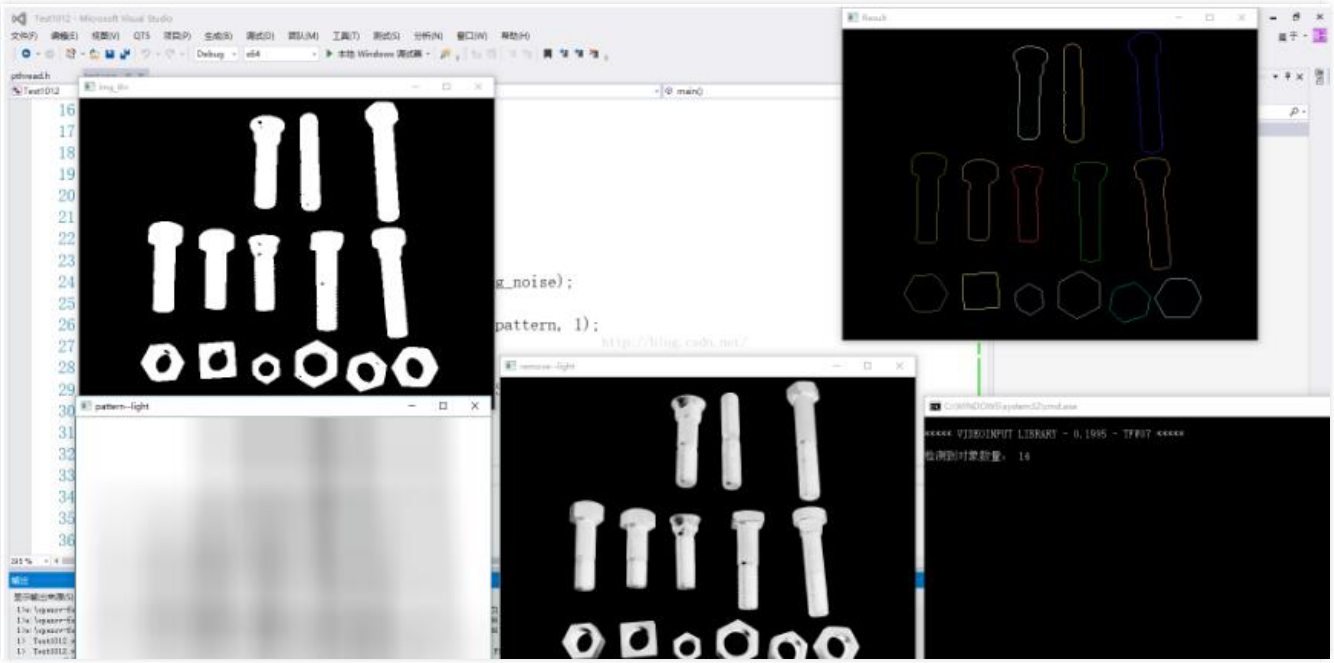
6.实现了三种目标检测方法

主要分两种连通区域和findCont

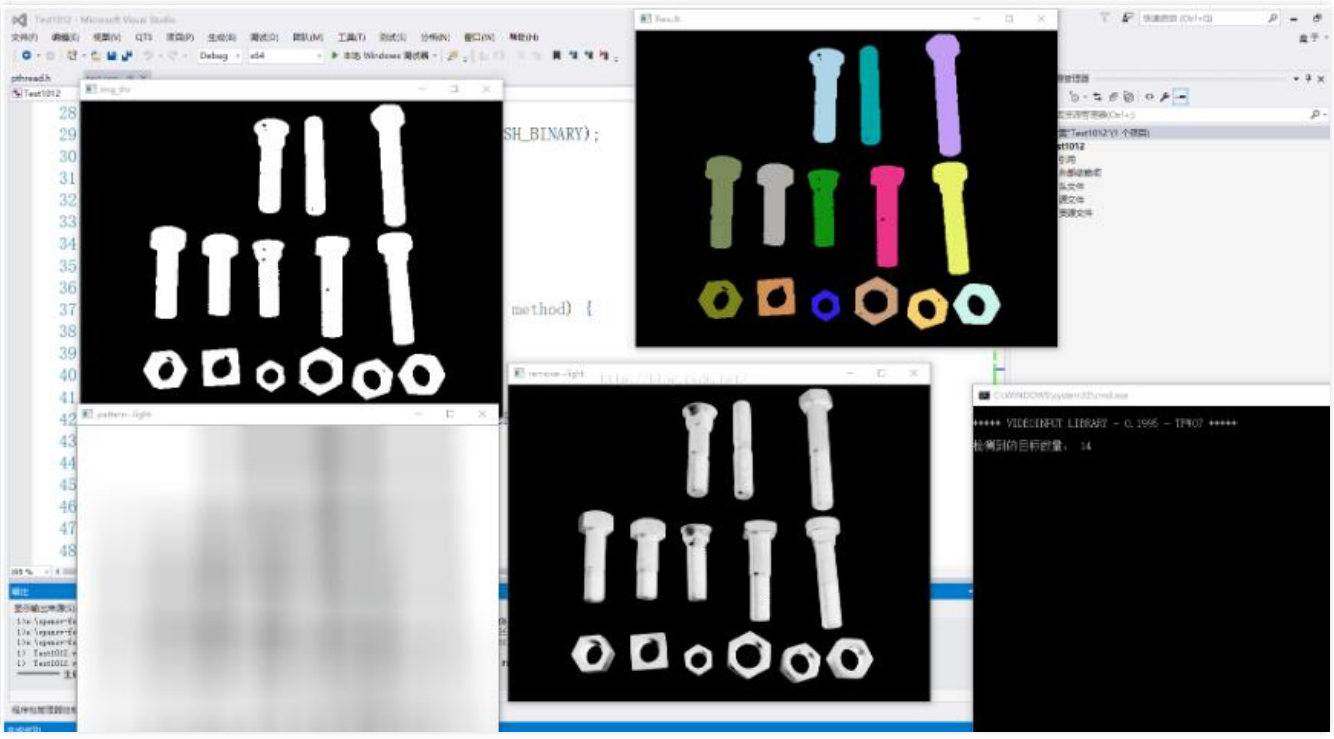
ours

过程遇到了错误主要是图片忘了灰化处理，随机颜色的问题。下面代码都已经进行了解决

这是findContours的效果



下面是连通区域的结果



```
#include <opencv2\core\utility.hpp>

#include <opencv2\imgproc.hpp>
#include <opencv2\highgui.hpp>
#include<opencv2\opencv.hpp>
#include <opencv2\core\core.hpp>
#include <opencv2\core\matx.hpp>
#include<string>
#include <iostream>
#include <limits>
using namespace std;
using namespace cv;
Mat img = imread("C:\\Users\\hasee\\Desktop\\luosi.jpg",0);
Mat removeLight(Mat imge, Mat pattern, int method);
Mat calculateLightPattern(Mat img);
static Scalar randomColor(RNG& rng);

void ConnectedComponents(Mat img);
void ConnectedComponetsStats(Mat img);
void FindContoursBasic(Mat img);
void main()
{
Mat img_noise;
medianBlur(img,img_noise,3);
Mat pattern = calculateLightPattern(img_noise);

Mat re_light = removeLight(img_noise, pattern, 1);

Mat img_thr;
threshold(re_light,img_thr,30,255,THRESH_BINARY);

//ConnectedComponents(img_thr);
ConnectedComponetsStats(img_thr);
//FindContoursBasic(img_thr);
waitKey(0);

}
Mat removeLight(Mat imge, Mat pattern, int method) {
Mat aux;
if (method == 1) {
Mat img32, pattern32;
imge.convertTo(img32, CV_32F);
pattern.convertTo(pattern32, CV_32F);
aux = 1 - (img32 / pattern32);
aux = aux * 255;
aux.convertTo(aux, CV_8U);
}
else {
aux = pattern - imge;
}
return aux;
}
```

```
Mat calculateLightPattern(Mat img) {
Mat pattern;
blur(img, pattern, Size(img.cols / 3, img.cols / 3));
return pattern;
}
static Scalar randomColor(RNG& rng)
{
int icolor = (unsigned)rng;
return Scalar(icolor & 255, (icolor >> 8) & 255, (icolor >> 16) & 255);
}
void ConnectedComponents(Mat img) {
Mat lables;
int num_objects = connectedComponents(img, lables);

if (num_objects < 2) {
cout << "未检测到目标" << endl;
return;
}
else {
cout << "检测到的目标数量： " << num_objects - 1 << endl;
}
Mat output = Mat::zeros(img.rows,img.cols,CV_8UC3);
RNG rng(0xFFFFFFFF);

for (int i = 1; i < num_objects;i++) {
Mat mask = lables == i;
output.setTo(randomColor(rng),mask);
}
imshow("Result",output);
}

void ConnectedComponetsStats(Mat img) {
Mat labels, stats, centroids;
int num_objects = connectedComponentsWithStats(img,labels,stats,centroids);
if (num_objects<2) {
cout << "未检测到目标" << endl;
return;
}
else {
cout << "检测到的目标数量： " << num_objects - 1 << endl;
}
Mat output = Mat::zeros(img.rows, img.cols, CV_8UC3);
RNG rng(0xFFFFFFFF);
for (int i = 1; i < num_objects; i++) {
Mat mask = labels == i;
output.setTo(randomColor(rng), mask);
stringstream ss;
ss << "area: " << stats.at<int>(i,CC_STAT_AREA);
putText(output,ss.str(), centroids.at<Point2d>(i),FONT_HERSHEY_SIMPLEX,0.4,Scalar(25
5,255,255));
}
imshow("Result", output);
}

void FindContoursBasic(Mat img) {
```

```
vector<vector<Point>> contours;
findContours(img, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);
Mat output = Mat::zeros(img.rows, img.cols, CV_8UC3);
if (contours.size()==0) {
cout << "未检测到对象" << endl;
return;
}else{
cout << "检测到对象数量： " << contours.size() << endl;
}
RNG rng(0xFFFFFFFF);
for (int i = 0; i < contours.size(); i++)
drawContours(output,contours,i,randomColor(rng));
imshow("Result", output);
}
```

补充知识：**SURF**特征点检测与匹配之误匹配点删除

SURF特征点检测与匹配之误匹配点删除

SURF（SpeededUp Robust Feature）是加速版的具有鲁棒性的算法，是SIFT算法的加速

版。

但是SURF特征匹配之后有大量的误匹配点，需要对这些误匹配点进行删除。

这里不从理论上讲解SURF原理等，直接说用法。

特征匹配的步骤分为三步：

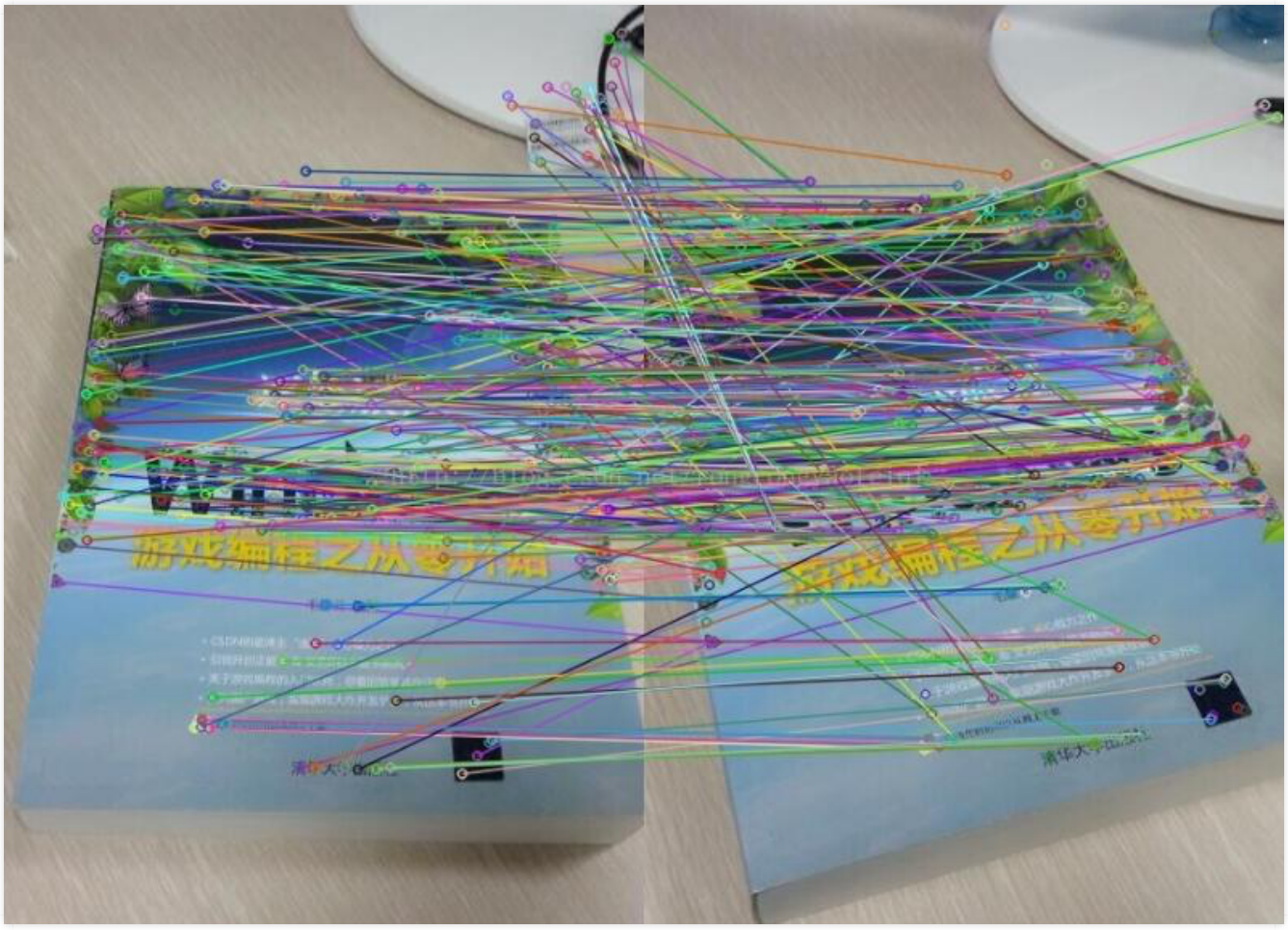
1、找出特征点

2、描述特征点

3、特征点匹配

具体基本代码见最后。具体的可以看毛星云的书籍，但是个人认为其编程风格不严谨，自己有做改动。

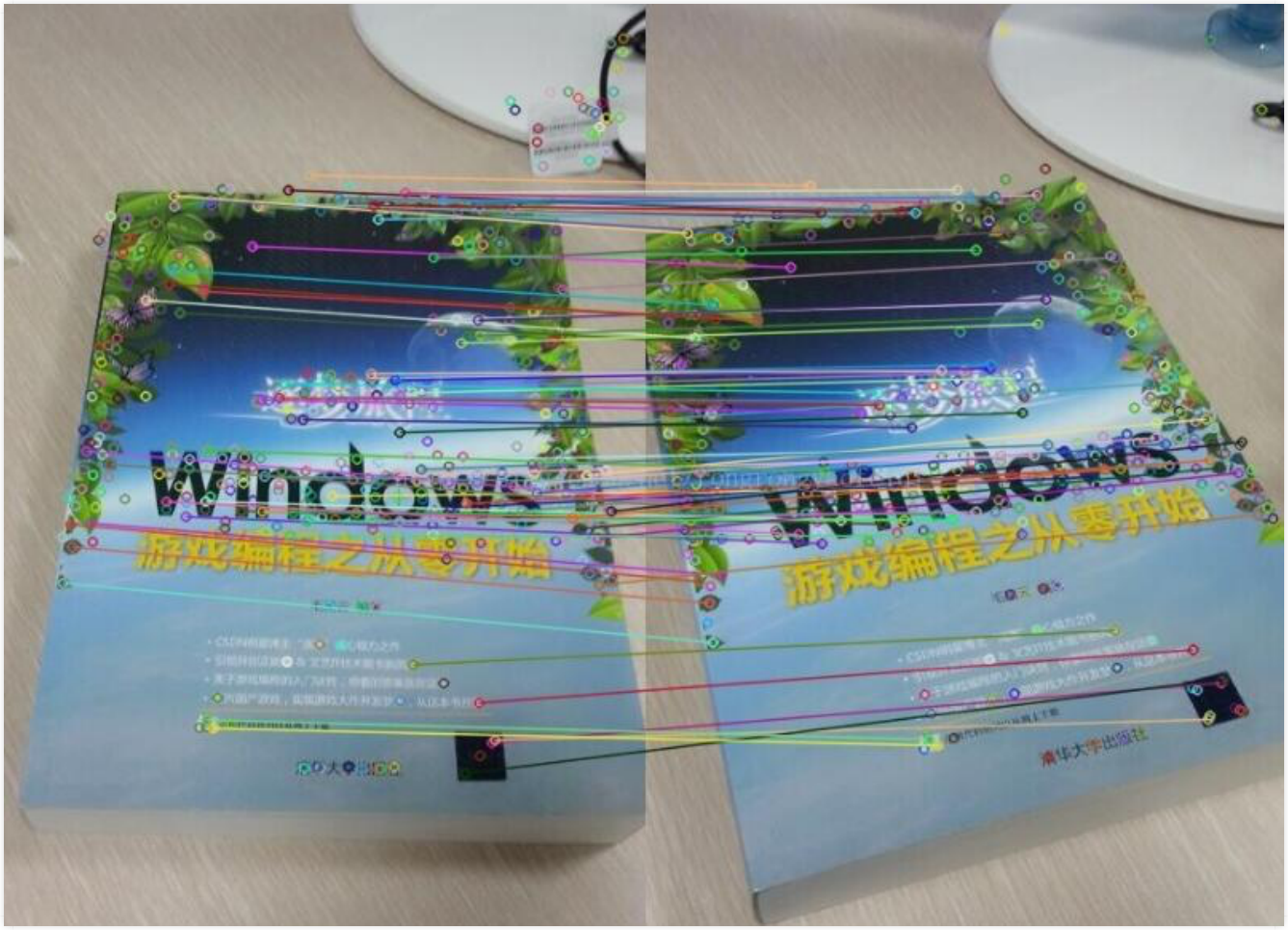
但是匹配出来的结果如下：



有很多的误匹配点，如何对误匹配点进行删除呢。

双向匹配加距离约束。

实验结果如下：效果还是非常好的。



```
#include "stdafx.h"
#include <opencv2\opencv.hpp>
#include <opencv2\nonfree\nonfree.hpp>
#include <opencv2\legacy\legacy.hpp>

#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{
 //读取图片
 cv::Mat srcImg1 = cv::imread("1.jpg", 1);
 cv::Mat srcImg2 = cv::imread("2.jpg", 1);
 if (srcImg1.empty() || srcImg2.empty())
 {
  std::cout << "Read Image ERROR!" << std::endl;
  return 0;
 }
 //SURF算子特征点检测
 int minHessian = 700;
 cv::SurfFeatureDetector detector(minHessian);//定义特征点类对象
 std::vector<cv::KeyPoint> keyPoint1, keyPoint2;//存放动态数组，也就是特征点

 detector.detect(srcImg1, keyPoint1);
 detector.detect(srcImg2, keyPoint2);

 //特征向量
 cv::SurfDescriptorExtractor extrator;//定义描述类对象
 cv::Mat descriptor1, descriptor2;//描述对象

 extrator.compute(srcImg1, keyPoint1, descriptor1);
 extrator.compute(srcImg2, keyPoint2, descriptor2);

 //BruteForce暴力匹配
 cv::BruteForceMatcher <cv::L2<float>>matcher;//匹配器
 std::vector <cv::DMatch> matches;
 matcher12.match(descriptor1, descriptor2, matches);

 //绘制关键点
 cv::Mat imgMatch;
 cv::drawMatches(srcImg1, keyPoint1, srcImg2, keyPoint2, matches, imgMatch);

 cv::namedWindow("匹配图", CV_WINDOW_AUTOSIZE);
 cv::imshow("匹配图", imgMatch);
 cv::imwrite("匹配图.jpg", imgMatch);
 cv::waitKey(10);
 return 0;
}
```

看完上述内容，是不是对openc

v自动光学检测、目标分割和检测的详细分析有进一步的了解，如果还想学习更多内容，欢迎关注亿速云行业资讯频道。

opencv

上一篇：雪花算法（04）机器信息

下一篇：如何在Phpcms中卸载模块

相关阅读

1. 【OpenCV学堂】图像处...

2. OpenCV中几何形状识别...

3. 汉信码在iOS客户端中的...

4. 基于android studio开发的 ...

5. win10下cmake编译Androi...

6. OpenCV的基本绘图函数

7. opencv归一化函数normali...

8. opencv 实现图像像素点反转

9. ViCANdo工具和OpenCV...

10. 从零开始的Opencv图像...

产品服务

地区划分

帮助支持

关于我们

广州亿速云计算有限公司

7*24小时在线电话：400-100-