上海市计算技术研究所

课程作业报告

报告名称：交通标志识别

课程名称：《图像处理》

专　　业：计算机软件与理论

班　　级：87901200

姓　　名：陶书衡

学　　号：879012005

2021-5-11

一、 识别目标

禁止左右转标志

二、 开发环境

- 操作系统： Mac OS Big Sur 版本 11.3.1
- 集成开发环境： CLion 2021.1.1 内部版本号 #CL-211.7142.21，April 22, 2021
- 编译环境：
  - gcc：stable 11.1.0 (bottled), HEAD
  - clang：Apple clang version 12.0.5 (clang-1205.0.22.9)
  - target: x86_64-apple-darwin20.4.0
  - thread model: posix
- 编译工具：CMake version 3.19
- 依赖库：
  - opencv 3.4.14

三、 算法与主要步骤

可靠的交通标志检测是对其进行准确识别的前提，本次作业首先利用交通标志的红色特征域分割出潜在的交通标志区域，然后通过中值滤波、腐蚀、膨胀与填充操作，初步抑制目标图像的噪声，分割出交通标志独立的图像元素，接着针对交通标志具有规则的形状这一关键特征，通过轮廓提取，Harris 角点检测方法最终确定出图像中的交通标志区域，实验结果表明了该算法的有效性。将圆形标志切割出来后，使用 ORB 特征检测与匹配，将原图片与目标图片（标志样本）进行特征点匹配，若匹配到的特征点个数大于等于 7 个，即可认为该图片可靠存在该标志。

3.1 转为 HSV 模型识别红色区域

一般对颜色空间的图像进行有效处理都是在 HSV 空间进行的，然后对于基本色中对应的 HSV 分量需要给定一个严格的范围，下面是通过实验计算的模糊范围。

- H: 0 — 180
- S: 0 — 255
- V: 0 — 255

此处把部分红色归为紫色范围：根据 hsv 分量模型，各种颜色范围分布如下：

	黑	灰	白	红		橙	黄	绿	青	蓝	紫
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34 ;	77	99	124	155
smin	0	0	0	43		43	43	43	43	43	43
smax	255	43	30	255		255	255	255	255	255	255
vmin	0	46	221	46		46	46	46	46	46	46
vmax	46	220	255	255		255	255	255	255	255	255

红色的范围是：[0, 43, 46]~[10,255,255]∪[156, 43, 46]~[180,255,255]。识别出的图片样例为：

3.2 中值滤波

中值滤波是一种典型的非线性滤波，是基于排序统计理论的一种能够有效抑制噪声的非线性信号处理技术，基本思想是用像素点邻域灰度值的中值来代替该像素点的灰度值，让周围的像素值接近真实的值从而消除孤立的噪声点。该方法在取出脉冲噪声、椒盐噪声的同时能保留图像的边缘细节。这些优良特性是线性滤波所不具备的。

中值滤波的原理很简单，就是用像素点邻域灰度值的中值来代替该像素点的灰度。由于中值滤波是基于排序统计理论，所以其对脉冲噪声、椒盐噪声会有很好的滤波效果，并能保留图像的边缘细节。但是代价就是效率低，通常花费的时间是均值滤波的数倍以上。opencv 中提供了 medianBlur()函数实现了中值滤波操作，其原型如下：

```
C++: void medianBlur(InputArray src, OutputArray dst, int ksize)
```

中值滤波完成后的图像如下图所示。

3.3 腐蚀与膨胀

OpenCV 提供的两种最基本的形态学操作,腐蚀与膨胀(Erosion 与 Dilation)。

腐蚀会把物体的边界腐蚀掉，卷积核沿着图象滑动，如果卷积核对应的原图的所有像素值为 1，那么中心元素就保持原来的值，否则变为零。主要应用在去除白噪声，也可以断开连在一起的物体。

膨胀是卷积核所对应的原图像的像素值只要有一个是 1，中心像素值就是 1。一般在除噪是，先腐蚀再膨胀，因为腐蚀在去除白噪声的时候也会使图像缩小，所以我们之后要进行膨胀。当然也可以用来将两者物体分开。

腐蚀的函数原型：

```
void erode(InputArray src, OutputArray dst, InputArray kernel,
Point anchor=Point(-1, -1), int iterations=1, int
borderType=BORDER_CONSTANT, const Scalar&
borderValue=morphologyDefaultBorderValue());
```

膨胀的函数原型：

```
dilate( InputArray src, OutputArray dst, InputArray kernel,
Point anchor = Point(-1,-1), int iterations = 1,
int borderType = BORDER_CONSTANT,
const Scalar& borderValue = morphologyDefaultBorderValue() );
```

经过腐蚀和膨胀后的图片分别如下两张图所示。

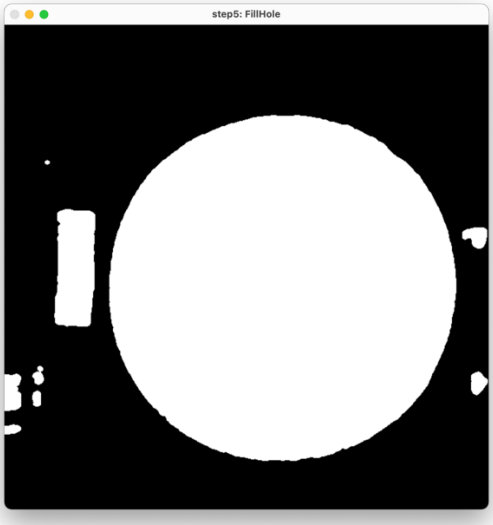
3.4 空洞填充

空洞图像的定义：由前景像素相连接的边界所包围的一个背景区域。对于二值图像，可以理解为被白色区域所包围的黑色区域就是空洞区域。

大致思路如下：

1. 设原图像为 A。
2. 首先 A 向外延展一到两个像素，并将值填充为背景色（0）,标记为 B。
3. 使用 floodFill 函数将 B 的大背景填充，填充值为前景色（255），种子点为（0，0）即可（步骤一可以确保（0，0）点位于大背景），标记为 C。
4. 将填充好的图像裁剪为原图像大小（去掉延展区域），标记为 D。
5. 将 D 取反与 A 相加即得填充的图像，E=A|（~D）

经过空洞填充的图像如下图所示。



3.5 检测轮廓

对于图像中的物体寻找轮廓（外围或全部），这里使用 OpenCV 里的 findContours 进行提取，并结合 drawContours 将轮廓绘制出来。

findContours 函数原型：

```
findContours( InputOutputArray image, OutputArrayOfArrays contours,
      OutputArray hierarchy, int mode,int method, Point
offset=Point());
```

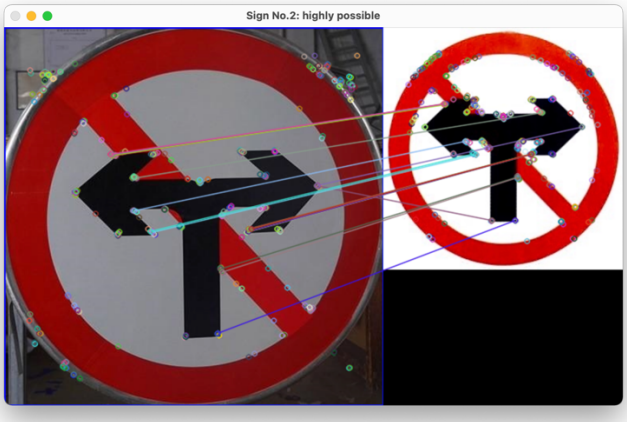
检测出来的边界如下图所示。之后对图像检测的区域进行剪切。

3.6 ORB 特征检测与匹配

将剪切好的图片与模板图片进行 ORB 特征检测与匹配。图像的特征点可以简单的理解为图像中比较显著显著的点，如轮廓点，较暗区域中的亮点，较亮区域中的暗点等。ORB 的全称是 ORiented Brief，采用 FAST（features from accelerated segment test）算法来检测特征点。

与 Brisk，AKAZE 类似，ORB 也分两部分，即特征点提取和特征点描述。特征提取是由 FAST 算法发展来的，它基于特征点周围的图像灰度值，检测候选特征点周围一圈的像素值，如果候选点周围领域内有足够多的像素点与该候选点的灰度值差别够大，则认为该候选点为一个特征点。而特征点描述是根据 BRIEF（Binary Robust Independent Elementary Features）特征描述算法改进的。将 FAST 特征点的检测方法与 BRIEF 特征描述子结合起来，并在它们原来的基础上做了改进与优化。据说 ORB 算法的速度是 sift 的 100 倍，是 surf 的 10 倍。ORB 算法是为解决 BRIEF 的缺陷而改进的，主要解决两个缺点：噪声敏感、旋转不变性。

匹配后返回特征点个数，若为 0 个，则判断为完全不符合模板图片特征；若特征点个数为 0～3，那么判断为小概率存在该标志；若特征点个数为 3～7，则可判断为特征上符合该标志；若特征点个数大于 7 个，则判断为存在该标志。特征点匹配的图像如下图所示。



并将结果在控制台予以显示，如下图所示。

```
Detected Sign No.1
Matched feature points number of No.1 sign = 0
Detected sign No.1 is completely impossible to be the target sign.

Detected Sign No.2
Matched feature points number of No.2 sign = 19
Detected sign No.2 is highly possible to be the target sign.

RESULT: At least one target sign is in this image.
```

四、 使用说明

4.1 运行程序

运行项目下的 cmake-build-debug/opencv（MacOS）或 cmake-build-debug/opencv.exe（Windows）。

4.2 编译程序

请先安装配置对应版本的 g++、gcc、opencv、cmake，并相应地配置环境变量。如果直接在 Visual Studio 中运行，可能导致出错。

1. 先 cd 到项目目录，例如：
   cd /Users/Bureaux/Documents/workspace/CLionProjects/opencv
2. 使用**已经配置好环境变量**的 cmake 进行编译，例如：
   cmake --build cmake-build-debug --target opencv -- -j 6
3. 步骤 2 将在 cmake-build-debug 中生成对应的可执行文件，直接运行即可。

4.3 使用说明

输入 1 为使用本报告中的带目标标志的样本图片。输入 2 为使用项目自带的不带目标标识的样本图片。输入 3 为输入第三方图片的**绝对路径**。