基于 OpenCV 和 Tesseract-OCR 的表带字符识别算法研究

徐　阳

（安徽理工大学，安徽 淮南 232000）

摘　要　随着我国智能制造的迅速发展，信息技术和制造业的覆盖面越来越广泛，本文针对智能穿戴中表带字符识别，提出了利用 OpenCV 和 Tesseract-OCR（Google 开源光学文字识别引擎）的方法。经过一系列图像预处理、图像降噪、定位等操作，再通过 Pyteseract 模块的处理，达到了快速、准确的文字识别目的。

关键词　OpenCV　字符识别　Tesseract-OCR 技术

中图分类号：TP3　　**文献标识码**：A　　**文章编号**：1007-0745(2022)05-0016-03

1 研究背景及理论基础

当下智能穿戴在人群中非常流行，专属化以及个性化的服务让人们有着满满的时尚感、科技感。人们在追求高品质生活的同时，对穿戴样式及美观需求更高，作为可更换的配件手表表带，除了对材质要求外，对图案样貌以及一些专属字符等也有需求。在生产车间里，用传统方式面对字符检测，耗时、耗力、效率低。所以为了更加简洁快速地识别表带字符，剔除劣质品，快速找到合格品[1]，本文着重研究基于 OpenCV 和 Tesseract-OCR 的表带字符识别算法的实现。

OpenCV 创建之初，是为了提供来源优化过的基础代码，可以实现许多图像处理的基础算法，具有更强的可读性、移植性且免费，支持 C++，Python 等语言。二十世纪八十年代初开发的一个开源 OCR 引擎在测试中精准度很高，经过不断地改进后，结合 Python 语言逐渐发展为 Pytesseract 的模块。它支持 PIL 库中各式各样的图片文件，也可以灵活地处理 OpenCV 图像，和 NumPy 数组相互转化，同时为了提高图像转对文本的处理能力，可以训练自己的库[2-3]。

2 表带字符识别流程

本文将采集到的表带图像进行预处理，包括图像灰度化、二值化，再进行定位，包括寻找字符轮廓、绘制轮廓等操作，将所需检测部分进行截取，最后 OCR 字符识别输出结果。

1. 获取图像。
2. 预处理。
3. 字符定位。
4. 字符图像提取。
5. OCR 字符识别。

3 表带图像预处理

3.1 图像颜色空间转换

在颜色空间转换中，灰度图像经过转换后，彩色图像中的所有通道都相同，其中 CV_8U 类型图像范围在 0 到 255 之间，CV_16U 类型图像范围在 0 到 6535 之间，CV_32F 类型图像范围在 0 到 1 之间。与之相反，彩色图像转换为灰色图像，就要使用加权公式，如下：

$$Y=(0.299)R+(0.587)G+(0.114)B$$

其中，R 代表彩色图像中红色部分（Red）的像素值，G 代表彩色图像中绿色部分（Green）的像素值，B 则代表彩色图像中蓝色部分（Blue）的像素值，加权的最终结果 Y 代表灰度的像素值。

```
gray=cv.cvtColor(dst,cv.COLOR_BGR2GRAY)
```

其中 gray 代表处理之后图像，COLOR_BGR2GRAY 代表 BGR 格式图像转化为 GRAY 格式图像。以下是得到的灰度图像（见图 1）：

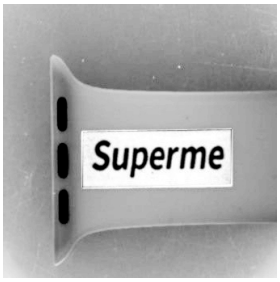

图 1 字符灰度图像

3.2 高斯滤波降噪

我们经常使用的图像滤波算法包括：中值滤波、均值滤波、高斯滤波。中值滤波特点是计算模板内所有像素的中值，然后用计算出来的值分别代替该像素

点的灰度值，这种方法能很好地保护边缘信息，但是花费时间较长。均值滤波的特点是计算模板内所有像素的平均值，然后用计算出来的值分别代替该像素点的灰度值，对于目标图像只能相对减弱噪声，无法克服边缘像素信息丢失部分。结合前两种方法不同的优缺点，本文采用高斯滤波的方法去除噪声，因为它对图像邻域内像素进行平滑时，不同位置的邻域像素有着不同的权值，能够更多地保留图像总体灰度分布特征。一维零均值高斯函数：

$$G(x,y)=\exp(-\frac{x^2+y^2}{2\sigma^2})$$

其中 x^2 和 y^2 分别表示的是邻域内其他像素与邻域内中心像素的距离，σ 是标准差。σ 值越大，函数图形越宽，图形越平坦，类似于平均模板。

σ 值越小，分布越集中，图形宽度越窄。

```
dst=cv.GaussianBlur(image,(3,3),0)
```

表带图像经过高斯滤波降噪处理后，如图 2 所示：

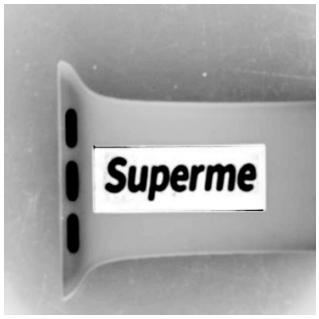

图 2 高斯滤波处理后

4 表带字符定位

为了使图像定位更加准确，首先需要将图像的轮廓清晰地表现出来，可以选用阈值分割法或者边缘检测的方法。

4.1 阈值分割法

简单的图像分割方法可以利用阈值来处理，图像阈值化适用于背景和目标占据不同灰度级范围的图像，因为它计算量小，性能稳定，成为最广泛的图像分割技术。我们可以给出一个数组以及一个阈值，然后根据每个数组中的值与阈值进行比对，不管是高了还是低了分别进行相应的处理。给定原始图像 f(x,y)，T 为阈值，则 g(x,y) 满足下式：

$$g(x,y)=\begin{cases}1, f(x,y)\geq T\\ 0, f(x,y)<T\end{cases}$$

使用全局阈值方法进行阈值图像处理：

```
ret,binary=cv.threshold(gray,0,255,cv.THRESH_BINARY|cv.THRESH_OTSH)
```

其中阈值类型 THRESH_BINARY 中，可以将超过阈值部分取最大值，反之取 0。THRESH_OSTH 遍历所有可能的阈值，然后对每个阈值结果的两类像素计算方差 σ_i^2。OTSU 算法计算方差使下列表达式最小：

$$\sigma_w^2 \equiv W_1(t)\cdot\sigma_1^2+W_2(t)\cdot\sigma_2^2$$

其中的 $W_1(t)$ 和 $W_2(t)$ 是根据两种类型像素的数量计算的权重，σ_1^2 和 σ_2^2 表示两类像素的方差。图 3 是经过全局阈值处理后的表带图像：

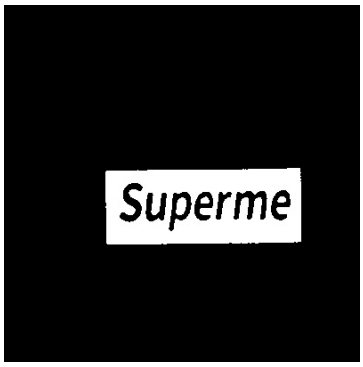

图 3 阈值化后的图像

4.2 边缘检测

这些轮廓是通过将滞后阈值应用于像素而形成的，并且采用了两个阈值。两个阈值中分为较大的数值 a 和较小的数值 b，像素的梯度被计算出后大于 a 就接受，反之小于 b 则舍弃，但如果介于 a 与 b 之间，那么接受它的方式只有它连接到一个高于阈值的像素时 [4–5]。Canny 内部调用 Sobel 算子，不但用了高斯平滑还有微分导数，来计算该图像灰度函数的近似梯度 [6]。

```
xgrad=cv.Sobel(gray,cv_16SC1,1,0)
ygrad=cv.Sobel(gray,cv_16SC1,0,1)
```

得到 x 和 y 方向的梯度后，对图像进行 Canny(xgrad,ygrad,50,150) 操作后得到图像（见图 4）：

图 4 边缘检测后的图像

4.3 两种定位方法的比较

阈值分割是用于强调图像中感兴趣的部分的方法，经过二值化处理后，显示出该目标的灰度值，强调主体本身具有灰度特性，使用阈值分割来表现。与之相

似的，边缘检测重点在于利用算法表现边缘的灰度特性，常用于发现物体的边缘，特征更加明显，方便后续操作[7-8]。

5 表带字符图像提取

将表带中需要提取的部分定位好后，准备进行字符图像提取，需要寻找轮廓并绘制轮廓：

cv.findContours(binary,cv.RETR_EXTERNAL,cv.CHAIN_APPROX_SIMPLE)

cv.drawContours(image,contours,i,(0,0,255),1)

其中 cv.RETR_EXTERNAL 表示只检测最外层的轮廓，cv.CHAIN_APPROX_SIMPLE 表示水平、垂直、对角线方向的元素被压缩，只保留该方向的最终坐标位置。表带字符的矩形外框只需 4 个点来保存轮廓信息。执行结束后得到的图像为（见图 5）：

图 5 字符图像提取

6 表带字符识别

6.1 预处理

由于使用 Tesseract-OCR 有字符识别方面的要求，所以在字符背景和像素方面要进行一些处理。首先进行图像预处理，去除干扰线与点，防止识别过程中出现错误[9]。

开操作是先腐蚀后膨胀的过程，它相当于一个几何运算的滤波器[10]。

open_out=cv.morphologyEx(binary,cv.MORPH_OPEN,kernel)

作为形态学中核心的 API 函数 morphologyEx()，cv.MORPH_OPEN 可以代表开运算，kernel 是其中的内核，可以通过 cv.getStructuringElement(cv.MORPH_RECT,(3,3)) 返回具有特定形状和大小的结构元素。得到开操作后的图像为（见图 6）：

图 6 开操作后的图像

6.2 Tesseract-OCR 字符识别

Tesseract 是一种开源 OCR（光学字符识别），可以识别不同格式的图像文件并将其转换为文本。首先我们需要下载 windows 下相应的安装文件，Pytesseract 是由 Python 封装，支持 PIL 图像或者 NumPy 图像，使用 Image.fromarray(open_out) 函数实现数组 array 到 image 的转换，使用 OCR 模块的 API 的方法：

text=tess.image_to_string(textImage)

最后以字符串的形式返回结果（见图 7）：

------Python OpenCV OCR-------
识别结果：Superme

图 7 识别结果

7 结语

本文利用一些传统的 Opencv 图像处理方式和目前较为流行的 OCR 方法，主要是使用 Python 语言，对表带识别进行了初步的研究。对于一些简单的图像，处理迅速、识别准确。但是关于如何应对复杂场景下的字符识别、能否进行算法的扩充以达到提高识别准确率的效果，还需要更进一步的分析研究。

参考文献：

[1] 郝辉，哈里木拉提·买买提，乔萨础拉，等．字符识别研究现状和发展趋势计量分析 [J]. 现代电子技术，2018,41(22):154.
[2] 阮秋琦，阮宇智．冈萨雷斯数字图像处理（第二版）[M]. 北京：电子工业出版，2010.
[3] Hongkun Tian,Tianhai Wang,Yadong Liu,Xi Qiao,Yanzhou Li.Computer vision technology in agricultural automation[J].Information Processing in Agriculyure,2020,07(01):1-19.
[4] 孙汝萍．一种改进的图像边缘提取方法 [J]. 科技通报，2018,34(10):158-161.
[5] RGV Gioi,G Randall.A Sub-Pixel Edge Detector:an Implementation of the Canny/Devernay Algorithm[J].Image Processing on Line,2017,07(01):347-372.
[6] 赵芳，周旺辉，陈岳涛，等．改进的 Canny 算子在裂缝检测中的应用 [J]. 电子测量技术，2018,41(20):107-111.
[7] 曾建华，黄时杰．典型图像边缘检测算子的比较与分析 [J]. 河北师范大学学报（自然科学版），2020,44(04):295-301.
[8] 张少伟．基于机器视觉的边缘检测算法研究与应用 [D]. 上海：上海交通大学，2013.
[9] 沈同平，王元茂．基于灰度形态学的图像边缘特征提取算法研究 [J]. 兰州纹理学院学报，2018,32(02):58-61.
[10] 刘晓刚，闫红方，张荣．基于形态学多尺度多结构的熔池图像边缘检测 [J]. 热加工工艺，2019,48(05):216-219.