上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

视频多媒体检索实践课程大作业

基于 OpenCV 和 CNN 的车牌识别系统

学院　电子信息与电气工程学院

组员　杜卓轩　郭佳毅　吴瀚

组员　徐恺阳　严楠

指导老师　何大治

2024 年 12 月 30 日

目录

基于 OpenCV 和 CNN 的车牌识别系统

1 前言

1.1 任务简述

本次视频多媒体检索实践课程大作业中，我们需要搭建一个车牌识别系统。基于课程所学内容方法，如 Canny 边缘检测、SIFT 算法等等，完成对车牌的检测、定位，字符的分割、识别以及最终的输出等。

具体功能需求如下：

1. 正确检测到图像中是否存在车牌；
2. 正确定位车牌在图像中的位置，输出裁切后的车牌图像；
3. 车牌字符分割后，与字符库匹配、检索并输出结果；
4. 正确识别车牌的颜色、字符数量、字符识别结果等；
5. 设计车牌识别系统的 GUI 界面；
6. 生成 exe 文件，供用户运行，并且支持从本地文件中上传图像。

1.2 总体流程

- **Step1:** 利用传统的 OpenCV 方法检测车牌并裁剪出来；
- **Step2:** 将裁减好的车牌二值化，根据横向波峰分布分割字符；
- **Step3:** 搭建卷积神经网络，
- **Step4:** 根据用户需求以及内部函数的接口设计 GUI 界面；
- **Step5:** 设计 LOGO，将所有代码封装，统一打包成 exe 文件并发行。

2 车牌检测与裁剪

车牌检测与裁剪、车牌字符分割这两个核心步骤的实现代码封装在 PlatesLocator 类中，主要提供以下方法：

accurate_place(self, card_img_hsv, limit1, limit2, color): 根据车牌颜色进一步精确裁剪车牌边界。

locate_plates(self, car_pic, resize_rate=1): 从原始图片中裁剪出车牌区域的完整操作。

separate_characters(plate_img, color): 从裁剪好的车牌中分离字符。

2.1 基本思路

- 找出图片中所有的矩形轮廓，即为可能的车牌区域；
- 获取轮廓的最小外接矩形，并根据车牌的理论长宽比进行筛选；
- 对倾斜的矩形区域进行仿射变换修正；
- 根据矩形中像素点的主体颜色判断车牌颜色，并筛去非蓝色、绿色的区域；
- 根据车牌颜色裁剪去非蓝色、绿色区域，进一步精确裁剪车牌边界。

2.2 具体实现

Step1: 图像预处理

首先，我们通过 cv2.resize() 和 cv2.GaussianBlur() 这两个函数统一所有图像大小，并进行一遍高斯模糊，目的是降低图像噪声，减少干扰。同时，通过 cv2.cvtColor() 将图像转化灰度图像，为后续边缘检测和二值化做准备。原始图像和经高斯模糊后的灰度图像如图 1和图 2所示。

图 1: 原始图像

图 2: 经高斯模糊后的灰度图像

然后，我们通过 cv2.morphologyEx() 函数对灰度图进行一遍开运算，以去除图片中不相关的小物体。cv2.addWeighted() 将经过开运算后的灰度图与原始灰度图进行加权叠加，将两幅图像合成为一幅图像，增强车牌区域的对比度。效果如图 3所示。

图 3: 开运算并加权叠加

图 4: Canny 边缘检测

Step2: 边缘检测

首先，我们通过 cv2.threshold() 二值化，对二值化化的图像通过 cv2.Canny() 进行 Canny 边缘检测，效果如图 4所示。然后，通过 cv2.morphologyEx() 进行闭运算和开运算让图像边缘成为一个整体，连接边缘中的小缺口，同时消除边缘区域中的小噪声。效果如图 5所示，可以看出车牌所在处已连接成一整块区域。

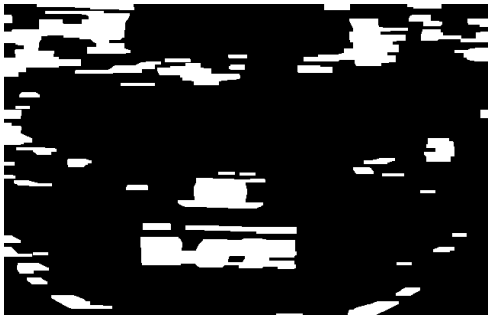

图 5: 闭开运算

图 6: 所有矩形区域

Step3: 轮廓检测并筛选矩形

我们通过 cv2.findContours() 查找图像边缘整体形成的轮廓，并且过滤掉面积过小或者是过大的轮廓。经筛选后，画面中仅剩如图 6所示的零星的几个矩形轮廓。

Step4: 矫正车牌

由于某些矩形区域可能是倾斜的，所以我们通过如下代码（以正角度倾斜为例）对该区域进行仿射变换，从而将其调整至水平。

```
new_right_point = [right_point[0], heigth_point[1]]
pts2 = np.float32([left_point, heigth_point, new_right_point])
pts1 = np.float32([left_point, heigth_point, right_point])
M = cv2.getAffineTransform(pts1, pts2)
dst = cv2.warpAffine(old_img, M, (pic_width, pic_height))
card_img = dst[int(left_point[1]):int(heigth_point[1]),
int(left_point[0]):int(new_right_point[0])]
ard_imgs.append(card_img)
```

Step5: 确定车牌颜色

首先，我们通过 cv2.cvtColor() 将车牌图像转换至 HSV 色彩空间，并统计其中在蓝色或者绿色范围内的像素点的个数，根据最多的像素颜色决定该区域是否为车牌，以及是什么颜色的车牌。

- 绿色 HSV 范围：$35 < H <= 99$ and $S > 43$ and $V > 46$
- 蓝色 HSV 范围：$99 < H <= 124$ and $S > 43$ and $V > 46$

Step6: 进一步裁剪边界

确定好车牌的颜色后，我们据此进一步精细地裁剪车牌边界。统计车牌图像每一行（列）的像素点在车牌颜色范围的个数，如果没有达到阈值，则将该行（列）裁剪出去。该部分由函数 accurate_place(self, card_img_hsv, limit1, limit2, color) 实现。

裁剪后的车牌如图 7所示。

图 7: 裁剪后的车牌

3 车牌字符分割

该部分目的是将切割好的车牌图像转换成黑白图像，并将每一个字符切割出来，用于后续神经网络的识别，由函数 find_waves(threshold, histogram) 和 separate_characters(plate_img, color) 实现。

3.1 基本思路

- 以灰度图的方式读取车牌图像，并二值化为黑白图像；
- 根据黑白图像每一列白色像素个数判断该列是否属于字符；
- 根据字符位置分割车牌，得到字符。

3.2 具体实现

Step1: 图像预处理

首先，我们通过 cv2.cvtColor() 读入车牌的灰度图像。然后，对于蓝色车牌，直接通过 cv2.threshold() 二值化得到黑底白字的黑白图像；对于绿色车牌，先通过 cv2.bitwise_not() 取反，再二值化得到黑白图像。效果如图 8所示。

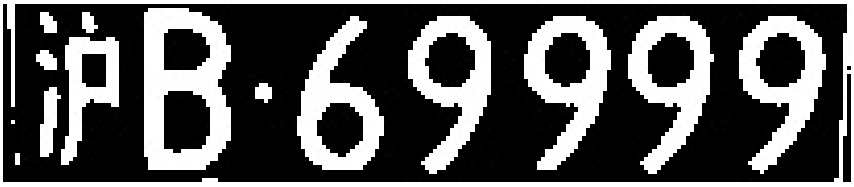

图 8: 二值化后的车牌

Step2: 寻找字符的位置

由于经预处理，字符呈白色，所以我们在每个 x 位置遍历该列的所有像素点，计算白色像素的比例，一旦这个比例超过设定的阈值，即认为这一列上有白色字符的信息。

从左到右遍历，如果 x_1 列开始出现较多的白色像素，一直持续到 x_2 列，则认为水平坐标位于 (x_1, x_2) 的这部分图像是一个字符（或汉字的一部分）。

首先，我们将车牌上下 2% 的位置裁去，即去除上下白边。然后，考察图像的水平投影直方图，倘若每一列的白色像素高于设定的阈值就是存在字符的部分。接着，我们定义 find_waves(threshold, histogram) 函数，根据设定的阈值和图像的水平投影直方图，找出所有的波峰，用于分隔字符。

Step3: 裁去多余的竖直白线

分割字符过程中，我们发现裁剪的图片很多时候最左边会有一条白边，如图 8所示。因此，这里我们判断第一个波峰的宽度是否小于最后一个波峰宽度的 1%，如果比这个小，则认为是白线，将其去除。代码如下：

```
if wave_peaks[0][0] / wave_peaks[-1][1] < 0.01:
    wave_peaks.remove(wave_peaks[0])
```

Step4: 根据字符位置分割车牌

首先，我们判断 find_waves() 函数得到的波峰个数，如果过少，即认为车牌裁剪不正确。然后，由于一个汉字的左右部分可能会被识别成两个峰，会导致一个汉字被分成两部分，所以需要将汉字的左右部分合在一起分割。

由于我们已经去掉了车牌的左侧边缘白线，所以第一个峰就是汉字的一部分，对后面的峰进行遍历，如果后面峰的结束位置减去第一个峰的宽度大于最后一个峰（即字母或者数字）的宽度的 70%，则认为这几个峰合起来是一个完整的汉字，然后将这个合成的新的峰取代遍历过的几个峰。

Step5: 去除车牌第 2、3 字符之间的白点

这里，我们对所有的峰进行遍历，找到峰最大的宽度，将所有峰中小于这个最大宽度的 30% 的峰都删除，这样剩下的峰的宽度就和最宽的相近，即认为都是字母、数字和汉字字符，除去了车牌第 2、3 字符之间的白点。

Step6: 分割字符

根据几个峰的位置信息切割图片，代码如下：

```
part_cards = [gray_img[:, wave[0]:wave[1]] for wave in wave_peaks]
```

最终效果如图 9所示。

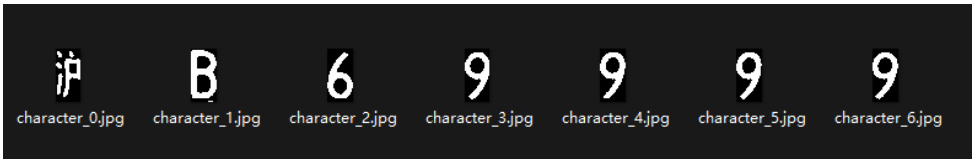

图 9: 车牌字符分割结果

4 字符匹配

4.1 基本思路

4.2 具体实现

5 GUI 界面

6 打包发行

由于本车牌识别系统基于 Python 实现，并且用到 OpenCV 和 PyTorch 等相关库，故设计如图 10所示的 LOGO，作为 exe 文件的图标。

图 10: LOGO

7 结语

本团队使用 GitHub 进行多人协作，仓库详见https://github.com/PasserbyZzz/PlateRecognition。欢迎提出 Issues 以及 Pull requests。Wish for your Star!!

任务分工

- 杜卓轩：设计 GUI 界面、生成 exe 文件
- 郭佳毅：报告撰写、讲解视频录制
- 吴瀚：神经网络搭建

- 徐恺阳：车牌检测与裁剪、LOGO 设计、报告撰写
- 严楠：车牌字符分割

参考文献

[1] 小布啊啊啊. Yolov5、cnn、svm 实现车牌检测, 2022. https://blog.csdn.net/swust512017/article/details/125637044.

[2] 自大人. python+opencv 只抠出车牌部分, 2020. https://blog.csdn.net/u014431739/article/details/107525601.

[3] wzh191920. 车牌号识别 python + opencv, 2018. https://blog.csdn.net/wzh191920/article/details/79589506.

[4] 荣仔！最靓的仔！. 基于 opencv 和 python 的车牌提取和字符分割, 2020. https://blog.csdn.net/IT_charge/article/details/107427133.

[5] detectRecog. Ccpd (chinese city parking dataset, eccv), 2019. https://github.com/detectRecog/CCPD.

[6] wzh191920. License-plate-recognition, 2018. https://github.com/wzh191920/License-Plate-Recognition.

[7] Dara to win. License-plate-recognition, 2018. https://github.com/Dara-to-win/Plate-Recognition.

[8] S. M. Silva and C. R. Jung. License plate detection and recognition in unconstrained scenarios. In *2018 European Conference on Computer Vision (ECCV)*, pages 580–596, Sep 2018.