基于序列化的车辆牌照识别系统

计 1702 王子杰
1401170224

2019 年 12 月 13 日

1 任务描述

自动车辆牌照识别（ANPR，Automatic Number Plate Recognition）系统，是智能交通系统的基础和核心技术之一，在交通管理自动化和智能化中占据重要地位。目前，ANPR 已广泛应用于公路卡口、收费站、车载移动车辆违章检查、门禁管理和停车场管理等场合。ANPR 通常采用光学字符识别技术（OCR，Optical Character Recognition）和其他分割、检测方法来识别汽车牌照。

每个国家车牌的大小和规格不同。其中，中国大陆地区大型汽车前牌照、小型汽车号牌、领馆汽车号牌、港澳入出境车号牌和教练号牌主要有 4 种车辆牌照，分别是民用蓝底白字牌照、民用黄底黑字牌照、军警用白底黑字或红字牌照以及国外驻华机构用黑底白字牌照。

本系统主要针对民用蓝底白字牌照的特征进行相关开发，其主要特征包括：（1）主要由蓝白两色构成；（2）车牌轮廓尺寸为 $440mm \times 140mm$，宽度和高度比大约是 3.14；（3）车牌中字符区长度为 $409mm$，高度为 $90mm$，字符的大小是 $45mm \times 90mm$；（4）相邻符号的间隔是 $12mm$，第 2、3 字符之间间隔为 $34mm$，两字符间有一圆点，圆点直径为 $10mm$；（5）字符的面积大约占整个车牌面积的 20%。

牌照区共有 7 个字符。左侧第一个汉字表示省、自治区、直辖市的简称，后面用一个英文字母代表发牌机关代号。右侧是 5 位号码，可以是阿拉伯数字或英文大写字母（为了避免混淆，不含英文字母 O 和 I，但是发牌机关代号中可以包含），表示机动车的注册登记编号。除了第一个汉字以外，其余字符各自笔画间都是互相连接的。

2 算法设计与实现

本系统主要包含图像预处理、车牌定位、车牌畸变矫正与车牌信息识别四部分。开发及实验过程中使用的图片通过 Python 爬虫程序收集得到，共包含约 500 张图片，大多为包含一张车牌的 RGB 图像。部分图像列举如下：

2.1 图像预处理

由于爬虫抓取到的图片尺寸大小不一，为避免过大的图片所包含过多像素点所带来的运算负担，在读入图片后首先将其变形至统一尺寸，本系统采用 640×480。接着通过高斯模糊去除细微噪声，然后将处理后的图像利用 OpenCV 进行通道分离和色彩空间转换，分别得到相关的 R、G、B 三个单通道图像，以及对应的灰度图像和 HSV 图像，为后续的操作做好准备。

```
def pre_process(src):
    img = cv2.resize(src, (640, 480))
    img_gaus = cv2.GaussianBlur(img, (5,5), 0)
    img_B = cv2.split(img_gaus)[0]
    img_G = cv2.split(img_gaus)[1]
    img_R = cv2.split(img_gaus)[2]
    img_gray = cv2.cvtColor(img_gaus, cv2.COLOR_BGR2GRAY)
    img_hsv = cv2.cvtColor(img_gaus, cv2.COLOR_BGR2HSV)
    return img, img_gaus, img_B, img_G, img_R, img_gray, img_hsv
```

2.2 车牌定位

目前主流的车牌定位方式主要可以分为两类：一类是基于车牌颜色特征的定位方式；另一类是基于车牌轮廓形状特征的定位方式。本系统主要采用基于颜色特征的方式进行车牌定位。而基于颜色特征的方式又可分为基于 RGB 空间识别和基于 HSV 空间识别两种。经过一系列实验后发现，单独使用任何一种方法的最终效果都不太理想。因此，本系统将二者结合，以 HSV 空间的 H 分量为主，以 RGB 空间的 R 分量和 B 分量为辅，后续再用车牌的长宽比例排除干扰。

具体而言，经过一系列实验尝试并且查阅了相关资料后发现，车牌的颜色特征主要具有以下特征：（1）在 HSV 空间下，H 分量的值通常都在 115 附近，S 分量和 V 分量因光照不同而差异较大；（2）在 RGB 空间下，R 分量通常较小，一般在 30 以下，B 分量通常较大，一般在 80 以上，G 分量波动较大。

通过 HSV 空间与 RGB 空间颜色特征的联合判断下，可以初步筛选得到车牌的可疑区域。将可疑位置的像素值置为 255，其他位置的像素值置为 0，实现图像二值化，二值图像中可疑区域用白色表示，其他区域均为黑色。随后通过膨胀、开闭等形态学操作和高斯模糊进一步降噪，并且再一次二值化后完成对可疑区域的最终筛选。

```
    def raw_ID(img_gray, img_hsv, img_B, img_R):
        for i in range(img_gray.shape[0]):
            for j in range(img_gray.shape[1]):
                if ((abs(img_hsv[:,:,0][i,j]-115) < 15)
                        and (img_B[i,j] > 70)
                        and (img_R[i,j] < 40)):
                    img_gray[i, j] = 255
                else:
                    img_gray[i, j] = 0
        kernel1 = np.ones((3, 3))
        kernel2 = np.ones((7, 7))
        img_gray = cv2.GaussianBlur(img_gray, (5, 5), 0)
        img_dilate = cv2.dilate(img_gray, kernel1, iterations=4)
        img_close = cv2.morphologyEx(img_dilate, cv2.MORPH_CLOSE, kernel2)
```

```
    img_close = cv2.GaussianBlur(img_close, (5, 5), 0)
    _, img_bin = cv2.threshold(img_close, 100, 255, cv2.THRESH_BINARY)
    return img_bin
```

此时，理想情况下，图像中应当只有车牌位置的像素颜色为白，其余像素均为黑色，但有些图像由于噪声干扰，会将非车牌部分保留下来。经过实验观察和比对，车牌区域与噪声之间存在较大的差异，且车牌区域特征比较明显：(1) 根据我国常规车牌的形状可知，车牌的形状为扁平矩形，长宽比约为 3：1 到 4：1；(2) 车牌区域面积通常远大于噪声区域，一般为图像中最大的白色区域。

在对白色区域做边缘提取后，根据上述特征筛选出车牌区域的轮廓。

```
def position_detection(img, img_bin):
    _, contours,_ = cv2.findContours(img_bin, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    det_x_max = 0 # 记录当前长最大值
    det_y_max = 0 # 记录当前宽最大值
    pts = 0    # 记录选出的轮廓序号
    for i in range(len(contours)):
        x_min = np.min(contours[i][ :, :, 0])
        y_min = np.min(contours[i][ :, :, 1])
        x_max = np.max(contours[i][ :, :, 0])
        y_max = np.max(contours[i][ :, :, 1])
        det_x = x_max - x_min
        det_y = y_max - y_min
        # 判断长宽比时留一些余量
        if ((det_x / det_y > 1.8)
            and (det_x > det_x_max)
            and (det_y > det_y_max)):
            det_x_max = det_x
            det_y_max = det_y
            pts = i
    points = np.array(contours[pts][:, 0])
    return points
```

最终得到的车牌轮廓上的点集 points 数组的 shape 为 (n, 2)，即存放了 n 个点的坐标，这 n 个点均分布在车牌的边缘上。接着可以利用 OpenCV 获取到点集的最小外接矩形以及该矩形的四个顶点坐标。此时并不清楚这四个坐标点各对应着矩形的哪一个顶点，因此无法充分地利用这些坐标信息，需要从坐标值的特征入手，将四个坐标与矩形的四个顶点匹配起来。而在 OpenCV 的坐标体系下，纵坐标最小的是 top_point，纵坐标最大的是 bottom_point，横坐标最小的是 left_point，横坐标最大的是 right_point。

```
def find_vertices(points):
    # 获取点集的最小外接矩形，返回值为该矩形的中心点坐标、宽高及倾斜角度
    rect = cv2.minAreaRect(points)
    # 获取该矩形的四个顶点坐标，返回值为浮点型，需将其转化为整型
    box = cv2.boxPoints(rect)
    box = np.int0(box)
    left_point_x = np.min(box[:, 0])
    right_point_x = np.max(box[:, 0])
    top_point_y = np.min(box[:, 1])
    bottom_point_y = np.max(box[:, 1])
    left_point_y = box[:, 1][np.where(box[:, 0] == left_point_x)][0]
```

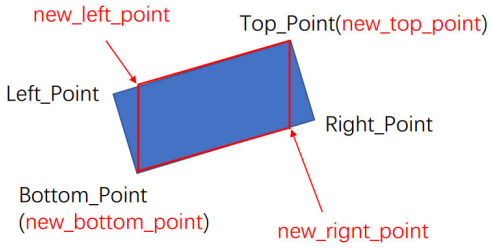
```
    right_point_y = box[:, 1][np.where(box[:, 0] == right_point_x)][0]
    top_point_x = box[:, 0][np.where(box[:, 1] == top_point_y)][0]
    bottom_point_x = box[:, 0][np.where(box[:, 1] == bottom_point_y)][0]
    vertices = np.array([
                    [top_point_x, top_point_y],
                    [bottom_point_x, bottom_point_y],
                    [left_point_x, left_point_y],
                    [right_point_x, right_point_y]
                ])
    return vertices, rect
```

至此，车牌定位已基本完成。

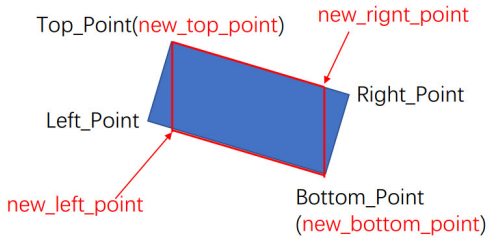
2.3 车牌畸变矫正

在车牌的图像采集过程中，相机镜头通常都不是垂直于车牌的，因此图像中车牌或多或少都会有一定程度的畸变，这会给后续的车牌信息识别带来一定困难，因此需要对车牌进行畸变校正，消除畸变带来的不利影响。由此前步骤定位探测到的畸变车牌图像示例如图所示：

要实现车牌的畸变矫正，必须找到畸变前后对应点的位置关系，由于拍摄的角度不同，可能出现如下图所示的两种不同的畸变情况。可以根据矩形倾斜角度的不同来判断具体是哪种畸变情况。



畸变情况1



畸变情况2

可以发现，平行四边形四个顶点与矩形四个顶点之间有如下关系：矩形顶点 Top_Point、Bottom_Point 与平行四边形顶点 new_top_point、new_bottom_point 重合，矩形顶点 Top_Point 的横坐标与平行四边形顶点 new_right_point 的横坐标相同，矩形顶点 Bottom_Point 的横坐标与平行四边形顶点 new_left_point 的横坐标相同。判断出具体的畸变情况后，选用对应的几何相似关系，即可轻易地求出平行四边形四个顶点坐标，即得到了畸变后车牌四个顶点的坐标。

此外，要实现车牌的校正，还需得到畸变前车牌四个顶点的坐标。由于我国车牌的标准尺寸为 440×140，可规定畸变前车牌的四个顶点坐标分别为：(0,0)，(440,0)，(0,140)，(440,140)。

顺序上需与畸变后的四个顶点坐标相对应。

```
def distortion_correction(vertices, rect):
    if rect[2] > -45:
        new_right_point_x = vertices[0, 0]
        new_right_point_y = int(vertices[1, 1] - (vertices[0, 0]- vertices[1, 0]) /
            (vertices[3, 0] - vertices[1, 0]) * (vertices[1, 1] - vertices[3, 1]))
        new_left_point_x = vertices[1, 0]
        new_left_point_y = int(vertices[0, 1] + (vertices[0, 0] - vertices[1, 0]) /
            (vertices[0, 0] - vertices[2, 0]) * (vertices[2, 1] - vertices[0, 1]))
        pts1 = np.float32([[440, 0],[0, 0],[0, 140],[440, 140]])
    elif rect[2] < -45:
        new_right_point_x = vertices[1, 0]
        new_right_point_y = int(vertices[0, 1] + (vertices[1, 0] - vertices[0, 0]) /
            (vertices[3, 0] - vertices[0, 0]) * (vertices[3, 1] - vertices[0, 1]))
        new_left_point_x = vertices[0, 0]
        new_left_point_y = int(vertices[1, 1] - (vertices[1, 0] - vertices[0, 0]) /
            (vertices[1, 0] - vertices[2, 0]) * (vertices[1, 1] - vertices[2, 1]))
        pts1 = np.float32([[0, 0],[0, 140],[440, 140],[440, 0]])

    new_box = np.array([(vertices[0, 0], vertices[0, 1]), (new_left_point_x,
        new_left_point_y), (vertices[1, 0], vertices[1, 1]), (new_right_point_x,
        new_right_point_y)])
    pts0 = np.float32(new_box)
    return pts0, pts1, new_box
```

该畸变是由于摄像头与车牌不垂直而引起的投影造成的，因此可用 cv2.warpPerspective() 来进行校正。

```
def transform_license(img, pts0, pts1):
    mat = cv.getPerspectiveTransform(pts0, pts1)
    license = cv.warpPerspective(img, mat, (440, 140))
    return license
```

至此，车牌畸变矫正已基本完成。

预处理、定位以及畸变矫正相关完整代码参见文件 CarPlate.py。

2.4 车牌信息识别

目前主流的车牌信息识别通常是通过对字符进行分割，进而用 KNN、SVM 或 CNN 等相关算法进行识别。本系统采用基于序列化的识别方式，利用递归神经网络（RNN）中的 GRU（Gate Recurrent Unit），设计了一个车牌识别网络（CPRNet，Car Plate Recognition Network）。

此前定位提取并且矫正完毕的车牌的灰度图像维度为 440×140，将其沿着 440 一维序列化输入到双层 GRU 中，GRU 每层隐藏节点数为 7，依次得到 7 个 140 维的输出向量 $V_i, i \in \{1, \cdots, 7\}$，其中 V_1 对应省份缩写，其余对应字母以及阿拉伯数字。接着，各个向量分别通过七组相同但独立的全连接模块和 softmax 函数得到最终的特征向量：

$$P_i = Softmax(W_2 \times ReLU(W_1 \times BN(V_i) + b_1) + b_2), \tag{1}$$

其中 BN 表示 batch normalization 层，$W_1 \in \mathbb{R}^{140 \times n}$，$b_1 \in \mathbb{R}^n$，$W_2 \in \mathbb{R}^{140 \times n}$，$b_2 \in \mathbb{R}^n$，W 和 b 表示全连接层，n 相应为省份缩写数 30 或大写字母与阿拉伯数字总数 36。P_1 即为省份

缩写对应的特征向量 $province\_feature \in \mathbb{R}^{30}$。将 P_2 到 P_7 拼接后，得到后六位字符的特征矩阵 $license\_feature \in \mathbb{R}^{36\times 6}$。

训练用到的损失函数为交叉熵（Cross Entropy），对七位字符分别计算损失后求和。优化器采用了 Adam 优化器，以 0.002 的初始学习率开始训练 250 个 epoch，batchsize 设置为 32。训练中采用了学习率衰减策略来加速收敛，具体为每 100 个 epoch 衰减为原先的十分之一。

```
def adjust_learning_rate(optimizer, epoch):
    lr = args['lr'] * (0.1 ** (epoch // 100))
    for param_group in optimizer.param_groups:
    param_group['lr'] = lr
```

模型训练用到的数据集为通过先前步骤定位提取并且矫正畸变后得到的车牌图像，从中选取了 388 张，人工标注车牌号。本节相关的完整代码参见文件 CPRNet_Train.ipynb。

3 结果展示

模型训练完成后，在用于制作数据集之外的图像中进行了测试。

正确识别情况较多：

Result: 湘A99999

Result: 闽D00169

Result: 辽A09030

Result: 晋O00888

Result: 冀A99999

Result: 蒙A00002

Result: 吉AG6666

Result: 川A88888

Result: 苏N00000

Result: 苏JAY888

Result: 青C79888

Result: 云F66666

Result: 鄂AB8088

Result: 苏A88888

个别位错误情况：

Result: 冀A22220

Result: 川C78888

Result: 云A5269G

Result: 皖A00566

有些图像由于噪声干扰未能精准提取车牌区域，但仍然能正确识别车牌信息：

Result: 闽FA8888

Result: 浙J66666

4 分析总结

至此，本基于序列化识别的车辆牌照识别系统开发完毕。在车牌定位方面，本系统结合了 RGB 空间和 HSV 空间的颜色特征并且利用长宽比和面积信息，从而提高了定位精度。在车牌信息识别方面，采用序列化的方式，省去了字符分割所需要的相关工作，且比基于卷积神经网络的模型引入了更少的模型参数和计算量，使得模型更加轻量，训练和运算都更加快捷，并且得到了较好的效果。但是一些车牌的定位和畸变矫正由于噪音干扰，仍做不到精准定位，

需要进一步优化算法策略。此外，CPRNet 模型的识别精度也有待进一步提高。

本次开发环境为 Manjaro Linux 18.1.4 + Python 3.7 + OpenCV 3.4.4 + Pytorch 1.3.1。