# 停车场车牌识别技术说明书

润和洑阳成虎

2019/11/25

## 1 项目开发目的和意义

随着科技经济的不断发展，汽车开始普及普通的家庭，大量的车辆为停车场的管理带来了新的问题。传统的停车场管理主要是通过给进入车场的车辆分发IC卡，记录车辆的出入时间，不管是固定车辆还是临时车辆，进出停车场都必须在出入口停车刷卡后，才能进出停车场，在车辆出入的繁忙的时段这种管理方式往往造成塞车的现象，耽误车主宝贵的时间。

针对以上的现象，我们提供了一种自动识别车牌的方法，该方法是自动识别进出的车辆，并使用模型推导出车牌号，并展现在图形展示的界面presentserver上。

本项目在华为Atlas200DK上实现了借助摄像头对车辆进行实时检测和识别，简化了停车场的车辆进入流程，具有实用性，便捷性等特征。减少了高峰期车辆进入的堵塞时间。

## 2 总体设计

系统可以划分为模型处理、车牌识别两个主要子系统，各子系统相对独立，但存在数据关联。其中模型处理包括数据集处理、模型训练等；车牌识别包括车辆和车牌检测和车牌识别。

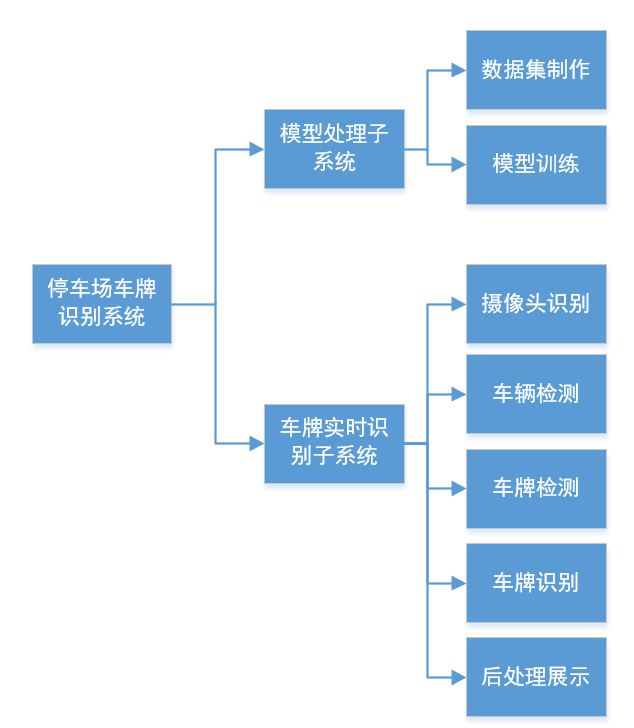


图2-1 系统整体功能结构图

## 3 算法设计

整个算法包括训练和推断两个阶段。训练阶段是使用车牌图片数据集使用caffe生成模型。推断阶段包括摄像头采集，车辆识别、车牌识别、车牌检测等模块分析摄像头读取的内容识别到车辆坐标和车牌信息，反馈给presentserver做输出。

### 3.1 数据集

网上搜到车牌数据集，共3922张车牌图片。采集到的图片远远不够，发现用户 szad670401在 Github 上开源的一个[车牌生成器](http://www.voidcn.com/article/javascript:void()" \t "http://www.voidcn.com/article/_blank)，当然也提供了相应的端到端识别模型。但是还是感觉生成的图片和真实的图片有所差距，故借鉴其代码中的添加高斯噪声、旋转、仿射变换、调整HSV、添加背景图像等操作对3922张图片做样本增强，先将图片resize至272\*72，这个分辨率大小是为了之后符合模型的输入，再做样本增强

### 3.2 模型训练

3922张图片每张图片均做20次样本增强，共80000多张图片，将71547张图片保存于train，10815张保存于val，每张图片的名字已经被修改为诸如0\_41\_31\_31\_31\_31\_31\_6\_4.jpg的名字，前7位数字即为该车牌号的对应label

在caffe中经常使用的数据类型是lmdb，不是常见的jpg,jpeg,png,tif等格式。比起单张图片，它具有I/O效率高、支持多线程并发读写、节省内存、语义完全符合ACID性等特点。本例中使用caffe自带的imagenet样例中的create\_imagenet.sh文件做部分修改生成Imdb文件。

制作好数据集后，按照如下链接对模型进行训练

<http://www.voidcn.com/article/p-gozhzosf-bpe.html>

### 3.3 部署流程设计

根据车牌检测需求，共设计了六个Engine，分别为摄像头模块、车辆检测模块、车辆检测处理模块、车牌检测模块、车牌识别模块、后处理模块，部署流程图如图3-14所示。

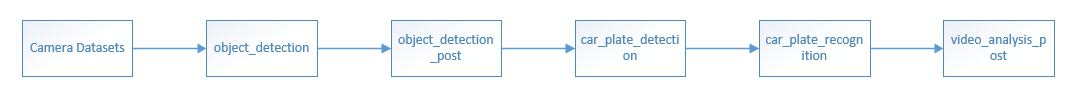
摄像头模块与Camera驱动进行交互，设置摄像头的帧率、图像分辨率、图像格式等相关参数，从摄像头中获取YUV420SP格式的视频数据，每一帧传给推理引擎进行计算。以此工程为例，其中帧率fps为5，图像分辨率取1280x720，摄像头图像格式为默认的YUV420SP。

车辆检测模块和车辆检测后处理模块接收摄像头数据，对YUV420SP格式的每帧图像进行处理，将摄像头图片经过VGG\_SSD模型推理的结果进行保存，将原图转为jpg格式的图片保存在结构体中，同时将推理出来的小图抠出来，但是还要是YUV420SP格式的图片，保存在结构体中，再将模型推理结果，也就是车辆的坐标点bbox记录下来，存放在结构体中，再将结构体保存下来发给下一个引擎

车牌检测模块接受上一个引擎的数据，对抠出来的车辆小图进行模型推理，找到车牌后将车牌抠成小图以供下一个模块使用，同时将原图和车辆坐标点bbox信息传递到发给下一个引擎的结构体中

车牌识别模块接受上一个引擎的数据，并且对抠出来的车牌小图进行模型推理，推理出车牌号码后，将车牌号码存放在结构体中。

后处理模块接受jpg格式的原图、车辆在此图中的bbox坐标点、车牌内容信息。与presentserver进行交互，将获取到的这些信息存放在交互的特殊结构体中，发给presentserver。



3-14 部署流程图

## 4 结果

使用树莓派摄像头对车辆图片进行实时识别。识别效果如下



## 5 关键代码说明

1. 车辆识别后将原图片和裁剪图片发往下一个Engine代码

**void** ObjectDetectionPostProcess::FilterBoundingBox(

**float**\* bbox\_buffer, **int32\_t** bbox\_buffer\_size,

shared\_ptr<VideoDetectionImageParaT>& detection\_image,

vector<ObjectImageParaT>& car\_plate\_imgs) {

**float**\* ptr = bbox\_buffer;

**int32\_t** num\_car = 0;

**int32\_t** num\_bus = 0;

**uint32\_t** base\_width = detection\_image->image.img.width;

**uint32\_t** base\_height = detection\_image->image.img.height;

**for** (**int32\_t** k = 0; k < bbox\_buffer\_size; k += kSizePerResultset) {

ptr = bbox\_buffer + k;

**int32\_t** attr = **static\_cast**<**int32\_t**>(ptr[BBoxDataIndex::kAttribute]);

**float** score = ptr[BBoxDataIndex::kScore];

**if** (score < confidence\_ || (attr != kLabelCar && attr != kLabelBus)) {

**continue**;

}

// bbox coordinate should between 0.0f and 1.0f

**uint32\_t** lt\_x = CorrectCoordinate(ptr[BBoxDataIndex::kTopLeftX])

\* base\_width, lt\_y = CorrectCoordinate(ptr[BBoxDataIndex::kTopLeftY])

\* base\_height, rb\_x = CorrectCoordinate(

ptr[BBoxDataIndex::kLowerRightX]) \* base\_width, rb\_y =

CorrectCoordinate(ptr[BBoxDataIndex::kLowerRightY]) \* base\_height;

**if** (rb\_x - lt\_x < kMinCropPixel || rb\_y - lt\_x < kMinCropPixel) {

**continue**;

}

// crop image

ObjectImageParaT object\_image;

BoundingBox bbox = { lt\_x, lt\_y, rb\_x, rb\_y };

HIAI\_StatusT crop\_ret = CropObjectFromImage(detection\_image->image.img,

object\_image.img, bbox);

**if** (crop\_ret != HIAI\_OK) {

**continue**;

}

object\_image.object\_info.score = score;

object\_image.box = bbox;

car\_plate\_imgs.push\_back(object\_image);

detection\_image->obj\_imgs.push\_back(object\_image);

//need change

//detection\_image->box = bbox;

printf("%d,%d,%d,%d\n",bbox.lt\_x,bbox.lt\_y,bbox.rb\_x,bbox.rb\_y);

}

}

1. 车牌识别相关关键代码

**if** (output\_data\_vec.empty()) { // check the output data vector is empty

HIAI\_ENGINE\_LOG("[CarPlateRecognition] output\_data\_vec is empty!");

**return** false;

}

**int** image\_number = image\_handle->obj\_imgs.size();

**for** (**int** ind = 0; ind < 1; ind++) { // loop for each batch

**if** (batch\_index + ind >= image\_number) { // check is valid image number

**break**;

}

string car\_palte\_str = "";

**float** car\_plate\_score = 0;

// loop for each output data vector, the vector size is 7

**for** (**int** i\_vec = 0; i\_vec < output\_data\_vec.size(); ++i\_vec) {

shared\_ptr<hiai::AINeuralNetworkBuffer> result\_tensor =

static\_pointer\_cast<hiai::AINeuralNetworkBuffer>(

output\_data\_vec[i\_vec]);

//get confidence result

**int** size = result\_tensor->GetSize() / **sizeof**(**float**);

**float**\* result = (**float**\*) result\_tensor->GetBuffer();

**int** max\_index = 0;

**int** oneResultSize = size / batch\_size\_;

**for** (**int** i = 0; i < oneResultSize; i++) { // loop for each result value

**if** (\*(result + i) > \*(result + max\_index)) { // get max value index

max\_index = i;

}

}

car\_palte\_str += kCarPlateChars[max\_index];

printf("max\_index %d\n",max\_index);

printf("car\_plate\_str if for is %s\n",kCarPlateChars[max\_index].c\_str());

car\_plate\_score += \*(result + max\_index);

}

CarInfoT out; // create car plate recognition result data

out.box = image\_handle->obj\_imgs[batch\_index + ind].box;

//out.attribute\_name = kCarPlateStr;

out.inference\_result = car\_palte\_str;

printf("car\_plate\_str is %s\n",car\_palte\_str.c\_str());

// get the mean value of 7 car plate number inference score

out.confidence = car\_plate\_score / kCarPlateLength;

HIAI\_ENGINE\_LOG("[CarPlateRecognition] car plate number:%s",

car\_palte\_str.c\_str());

tran\_data->car\_infos.push\_back(out);

}

**if** (tran\_data->car\_infos.empty()) { // check car\_infos vector is empty

HIAI\_ENGINE\_LOG("[CarPlateRecognition] car\_infos vector is empty!");

**return** false;

}

**return** true;

1. 后处理和presentserver交互相关代码

HIAI\_StatusT VideoAnalysisPost::HandleResults(

**const** std::shared\_ptr<BatchCarInfoT> &inference\_res) {

HIAI\_StatusT status = HIAI\_OK;

**uint32\_t** width = inference\_res->image.img.width;

**uint32\_t** height = inference\_res->image.img.height;

**uint32\_t** img\_size = inference\_res->image.img.size;

// every inference result needs 8 float

// loop the result for every inference result

std::vector<DetectionResult> detection\_results;

//float \*ptr = result;

**int** image\_number = inference\_res->car\_infos.size();

**for** (**int32\_t** k = 0; k < image\_number; k ++) {

// confidence

**float** score = inference\_res->car\_infos[k].confidence;

//Detection result

DetectionResult one\_result;

// left top

Point point\_lt, point\_rb;

point\_lt.x = inference\_res->car\_infos[k].box.lt\_x;

point\_lt.y = inference\_res->car\_infos[k].box.lt\_y;

// right bottom

point\_rb.x = inference\_res->car\_infos[k].box.rb\_x;

point\_rb.y = inference\_res->car\_infos[k].box.rb\_y;

one\_result.lt = point\_lt;

one\_result.rb = point\_rb;

printf("point:%d %d %d %d\n",point\_lt.x,point\_lt.y,

point\_rb.x,point\_rb.y);

printf("score=%f\n",score);

printf("inference\_result=%s\n",inference\_res->

car\_infos[k].inference\_result.c\_str());

printf("width=%d height=%d\n",width,height);

// check results is valid

**if** (IsInvalidResults(score, point\_lt, point\_rb)) {

**continue**;

}

HIAI\_ENGINE\_LOG(HIAI\_DEBUG\_INFO,

"score=%f, lt.x=%d, lt.y=%d, rb.x=%d, rb.y=%d", score,

point\_lt.x, point\_lt.y, point\_rb.x, point\_rb.y);

**int32\_t** score\_percent = score \* 100;

one\_result.result\_text.append(inference\_res->

car\_infos[k].inference\_result);

one\_result.result\_text.append(":");

one\_result.result\_text.append(to\_string(score\_percent));

one\_result.result\_text.append("%");

// push back

detection\_results.emplace\_back(one\_result);

}

**int32\_t** ret;

ret = SendImage(height, width, img\_size, inference\_res->image.img.data.get(), detection\_results);

// check send result

**if** (ret == -1) {

status = HIAI\_ERROR;

}

//}

**return** status;

}

## 6 后续可扩展性

本项目只是做了简单的实时识别车辆车牌，后续可以加数据库和计费系统，形成一个完整的小区车牌识别计费系统。