光电信息处理综合实验

具有交通标志识别与行人检测功能的树莓派小车系统实现

作者: 韩 啸 (学号: 3160101136)

顾钰峰 (学号: 3160104444)

范元港 (学号: 3160105048)

指导老师: 楼东武

浙江大学 信息与电子工程学院

2019 年 6 月 18 日

摘要

本文基于椭圆检测算法、LeNet 卷积神经网络和 SSD 行人检测模型，提出了一种高速实时的微型自动驾驶系统。小车的硬件包括树莓派 PI 3B+ 实验板、51duino 电机驱动板以及 5MP RASPBERRY PI CAMERA 摄像头。实现的系统**把第三方依赖降至了最低**，除了驱动板的烧录程序外，其余程序和脚本全部由我们自己编写；采用了**可靠且简单的图像预处理方法**，达到了拔群的检测效果；在自己构建的数据集验证集上，达到了 **97.65%**的分类准确率，超越前人**三个百分点**；摆脱框架约束，创新性的使用 **UDP 协议传输视频，将视频传输延时压缩到 40ms 之内**；在原有功能的基础上，创新性的**加入了 SSD 行人检测模型及鸣笛报警机制**，且加入后不影响实时性；**充分考虑了小车驱动算法的合理性**，如小车只会执行检测到的最大标志所代表的命令以及小车在转弯后会继续执行上一条命令等。

关键词：椭圆检测，LeNet 卷积神经网络，SSD 目标检测算法，UDP/TCP 通讯协议

目录

1 引言 **1**

2 文献综述 **2**
2.1 目标检测算法 . . . 2
2.1.1 传统方法 . . . 2
2.1.2 非传统方法 . . . 2
2.2 图像分类算法 . . . 3
2.3 树莓派简单介绍 . . . 3

3 关键技术与算法实现概述 **4**
3.1 图像预处理 . . . 4
3.1.1 HSV 空间颜色提取 . . . 4
3.1.2 形态学处理 . . . 4
3.1.3 连通域计算 . . . 5
3.1.4 椭圆检测 . . . 6
3.2 LeNet 卷积神经网络分类器 . . . 6
3.2.1 C1-卷积层 . . . 6
3.2.2 S2-池化层（下采样层） . . . 7
3.2.3 C3-卷积层 . . . 7
3.2.4 S4-池化层（下采样层） . . . 8
3.2.5 C5-卷积层 . . . 8
3.2.6 F6-全连接层 . . . 8
3.2.7 OUTPUT-全连接层 . . . 8
3.3 SSD 目标检测算法 . . . 9
3.4 视频与指令传输 . . . 9
3.4.1 协议选择 . . . 9
3.4.2 数据格式 . . . 11
3.5 电机与音响控制 . . . 12
3.5.1 树莓派 GPIO . . . 12
3.5.2 转弯与变速原理 . . . 13
3.5.3 树莓派音响 . . . 13

4 系统测试与评估 **14**
4.1 性能指标量化 . . . 14
4.1.1 算法效率 . . . 14
4.1.2 预处理结果 . . . 15
4.1.3 数据集获取 . . . 15
4.1.4 模型准确率 . . . 16

4.1.5 传输性能 17
4.2 优势与不足 18
4.3 未来优化 18

5 总结与展望 **19**

6 参考文献 **20**

图像列表

1 需要识别的交通标志 4
2 HSV 模型 5
3 LeNet 网络结构示意图 6
4 SSD 网络结构示意图 9
5 TCP 与 UDP 的传输方式比较 10
6 TCP 协议信道利用率计算示意图 11
7 树莓派 GPIO 管脚分布图 12
8 整体系统结构 14
9 图像预处理过程输出 15
10 遥控器 UI 16
11 部分数据集图片示例 16
12 LeNet 训练曲线 17

表格列表

1 C1-卷积层参数表 7
2 S2-池化层参数表 7
3 C3-卷积层参数表 7
4 S4-池化层参数表 8
5 C5-卷积层参数表 8
6 电机指令映射表 12
7 音响指令映射表 12
8 驱动板电机控制 GPIO 管脚号 13
9 算法执行时间测试表 14

1 引言

无人驾驶作为汽车未来的研究方向，其对于汽车行业甚至是交通运输业有着深远的影响。无人驾驶汽车的来临将能够解放人类的双手，降低发生交通事故发生的频率，保证了人们的安全。同时随着人工智能、传感检测等核心技术的突破和不断推进，无人驾驶必将更加智能化，同时也能够实现无人驾驶汽车的产业化。

无人驾驶汽车是指给车辆装备智能软件和多种感应设备，包括车载传感器、雷达、GPS以及摄像头等，获取车辆周围环境信息，并根据所获取的信息像人的大脑一样进行智能处理、分析判断，从而控制车辆的行使方向与速度，实现车辆的自主驾驶，安全高效地到达目的地，并且最终实现完全消除交通拥堵和交通事故，为环境保护做出突出贡献。简单来讲，无人驾驶汽车是指一种达到全自动、行驶途中不需要人为干涉的智能汽车。无人驾驶系统平台十分复杂，主要涉及四个技术领域：环境感知、路径规划、计算机控制、决策控制。因此，无人驾驶系统主要由环境感知系统、计算机控制系统、决策控制系统和地理信息系统组成。

环境感知系统充当无人驾驶车的“眼睛”，主要是通过无人驾驶汽车所装载的外部传感器获取外部环境信息，对其进行建模，将汽车所处的地理信息、障碍物信息等准确快速地传输给计算机控制系统，所以这个系统主要由两部分组成：车载导航系统和障碍物检测系统。早期采用磁导航技术，通过埋设在道路上的磁钉或电线等导航设备获取信息；随后采用光学摄像头，通过对摄像头拍摄的图像进行分析处理，获取道路信息，但是这种方式对图像处理和模式识别技术要求十分高，否则难以保证系统的准确性与实时性。目前，谷歌所研发的无人驾驶汽车采用的是激光雷达技术，这种技术的主要原理是发射激光和接收激光信号、检测激光遇到障碍物的回波信号，通过回波信号检测作业现场的障碍物和当前激光雷达传感器之间的距离，还可以通过谷歌地图获取当前的地理位置。计算机控制系统，像人的大脑一样智能化地对各种传感器传输来的信息进行快速、准确的分析处理，根据系统软件的控制算法，再将控制策略指令传输给执行机构进行执行。这种智能化的决策需要强大的后台软件支持，才能够做到对大量的激光雷达、GPS、电池电压、工作电流、行使位姿等信息进行准确而快速的处理，到达严格的实时性要求。

本文所关注的交通标志识别属于环境感知系统中图像识别分支。通过椭圆检测对摄像头抓取到的图像进行分析，提取出交通标志。通过 LeNet 卷积神经网络对提取到的交通标志进行分类，以此来决定小车下一步的行动方式。为了验证算法的实际效果，我们将模型应用于树莓派与 51duino 试验板控制的小车，实现了交通信号识别系统的构建，模拟了未来无人驾驶汽车可能的应用场景。

2 文献综述

2.1 目标检测算法

目标检测指的是在图像中检测某个物体的位置，得到该物体在图像中的准确坐标与区域。

随着时代的发展，目标检测问题的内容与算法也经过了不断的演变：最初是直线、椭圆等简单几何图形的检测，一些诸如 Hough 变换、曲线拟合等传统算法即可较好解决该问题；紧接着为了实现基于图像配准原理的图像检测，众多学者致力于探索构造各种图像底层特征，经典的 SIFT、FAST、SURF 都是这一时期提出的典型特征；之后扩展到语义层面目标的检测，比如人脸、人眼等的检测，Haar 特征 +Cascade 等传统方法也可以较好地解决；深度学习领域的目标检测一般指检测并分类图像中的物体，得到 2D 或 3D bounding box 与类别标签，常用的算法有 R-CNN、Fast R-CNN、YOLO、SSD 等算法。

2.1.1 传统方法

Hough 变换是一种使用表决原理的参数估计技术，其原理是利用图像空间和 Hough 参数空间的点－线对偶性，把图像空间中的检测问题转换到参数空间；它的优点是有很好的容错性与鲁棒性，缺点是只能够检测能用简单的参数描述的图形、且参数较多时计算量过大。对于能够用少数参数来表示的几何图形，拟合也是一种可行的检测方法，比如通过用拉格朗日乘子法求解最小二乘约束问题来拟合椭圆。

对于基于特征的目标检测问题，Haar 特征 +Cascsde 方法是可行的：其原理是用一系列级联的弱分类器来分类 Haar 特征，通过投票表决得到检测结果；SIFT（Lowe et al. [1999]）和 SURF（Bay et al. [2006]）也都是可行的，它们的特征提取能够使图像在旋转平移，仿射变换的情况下保持不变性，具有一定的抗噪性。

2.1.2 非传统方法

深度学习领域的目标检测问题的对象一般是有较固定形状的 things，集位置检测与分类于一体。R-CNN（Girshick et al. [2014]）开创了基于候选区域的目标检测方法，其原理是先筛选出若干个候选区域、然后用 CNN 处理每个候选区域得到分类结果与 bounding box 偏差，是深度学习在目标检测领域的首次成功突破。Fast R-CNN（Girshick [2015]）算法是在 R-CNN 算法的基础上改进得到的，主要的改进在于将从原图中筛选候选区域改进为筛选得到特征图中的 RoI（Region of Interst），大大提升了训练速度。Faster R-CNN（Ren et al. [2015]）则是在 Fast R-CNN 基础上改进得到的，主要的改进在于提出了一个 RPN 网络用于得到 ROI，取代了之前计算量很大的选择性搜索方法，进一步提升了算法的运行速度。YOLO（Redmon et al. [2016]）算法不同于上述算法，是一种基于 end-to-end 学习的深度学习算法，其原理是先将图像划分为若干个格子、再对每个格子里物体的类别与 bounding box 信息进行预测，优点在于检测速度快、背景误检率低，缺点是物体定位误差较大。SSD（Liu et al. [2016]）是在 YOLO 的基础上融合了区域提名的思想改进得到的，在保持了快速检测的基础上改进了小物体的定位精度。

2.2 图像分类算法

图像分类问题指的是给出一张（主要）只含有一个种类的单个物体的图像，得到其包含的物体的类别。

该问题下的传统算法有朴素贝叶斯分类、支持向量机、K 最近邻等方法，但由于计算量过大等问题已逐步退出图像分类的舞台。目前更为常用的是基于卷积神经网络的方法：LeNet（LeCun et al. [1998]）是一个经典的 CNN 算法，最初在手写数字识别方面获得了很大的成功；AlexNet（Krizhevsky et al. [2012]）同样是经典的 CNN 算法，其特点是由并行的两个相同的网络组成，可以在两块 GPU 上同时分别运行；VGG（Simonyan and Zisserman [2014]）同样是经典的 CNN 算法，其网络的深度比前者要深得多，也取得了更好地效果；GoogLeNet（Szegedy et al. [2015]）则开创性地加入了 Inception moduel，将不同大小的卷积核处理后的特征图融合；ResNet（He et al. [2016]）则提出了 Residual learning，改变了网络的学习目标。

2.3 树莓派简单介绍

树莓派是由注册于英国的慈善组织“Raspberry Pi 基金会”开发的一款只有信用卡大小的微型电脑。它是一款基于 ARM 的微型电脑主板，以 SD/MicroSD 卡为内存硬盘，卡片主板周围有 4 个 USB 接口和一个 10/100 以太网接口，可连接键盘、鼠标和网线，同时拥有视频模拟信号的电视输出接口和 HDMI 高清视频输出接口，具备所有 PC 的基本功能。树莓派基金会提供了基于 ARM 的 Debian 和 Arch Linux 的发行版供大众下载。还计划提供支持 Python 作为主要编程语言，支持 Java、BBC BASIC、C 和 Perl 等编程语言。

作为一款为学习计算机编程教育而设计的微型电脑，树莓派广泛受到全世界青少年电脑爱好者的喜爱，在计算机程序设计教育中被广泛使用。

3　关键技术与算法实现概述

3.1　图像预处理

为了提高图像分类任务的准确率以及满足系统实时性的要求，我们有必要对摄像头捕获的帧进行底层操作，去除不必要的冗杂信息，强调重要特征信息。

3.1.1　HSV 空间颜色提取

我们需要识别的交通标志如下图1所示，可以看出以下交通标志的颜色以红蓝两色为主。我们可以在某一特定颜色空间上，将感兴趣的颜色范围加以提取。

图 1: 需要识别的交通标志

由于在传统的 RGB 颜色空间中，三种颜色分量的值呈线性叠加关系，不容易提取某一色调的颜色；而 HSV 模型则能更好的胜任这个任务。HSV 模型是 RGB 模型的一种非线性映射，它们的转换关系如下：

$$\mathrm{V} \leftarrow \max(\mathrm{R}, \mathrm{G}, \mathrm{B}) \qquad [3.1.1-1]$$

$$S \leftarrow \begin{cases} \frac{V-\min(R,G,B)}{0} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases} \qquad [3.1.1-2]$$

$$H \leftarrow \begin{cases} 60(G-B)/(V-\min(R,G,B)) & \text{if } V = R \\ 120+60(B-R)/(V-\min(R,G,B)) & \text{if } V = G \\ 240+60(R-G)/(V-\min(R,G,B)) & \text{if } V = B \end{cases} \qquad [3.1.1-3]$$

我们根据公式可以得到如下图2所示的 HSV 模型示意图，可以看到色调分量（H）在角度平面上把颜色很好的分离开来，我们只需要提取几段范围内的 H 分量，并相应的限制 S 和 V 分量，就可以得到我们想分离出的颜色。

3.1.2　形态学处理

正如我们所知道的，膨胀会扩大一幅图像的组成部分，而腐蚀则会缩小一幅图像中的组成部分。由腐蚀和膨胀能够组合出另外两个重要的形态学操作: 开操作和闭操作。开操作通

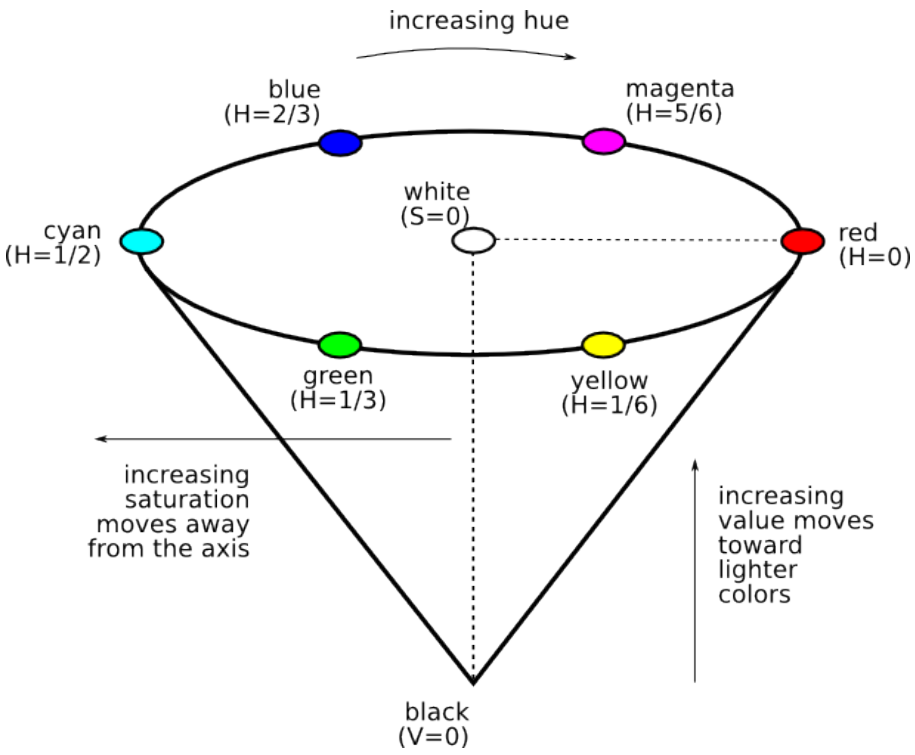


图 2: HSV 模型

常会平滑物体的轮廓，打破狭窄的狭颈，消除细小的突出物。闭操作也倾向于平滑轮廓的一部分，但与开操作相反，它通常融合窄的断口和细长的沟壑，消除小洞，并填补轮廓的间隙。

结构元 B 对集合 A 的开操作，表示为 $A \circ B$，定义如下：

$$A \circ B = (A \ominus B) \oplus B \qquad [3.1.2-1]$$

因此 B 对 A 的开操作就是 B 对 A 的腐蚀，紧接着用 B 对结果进行膨胀。

结构元 B 对集合 A 的闭操作，表示为 $A \cdot B$，定义如下：

$$A \cdot B = (A \oplus B) \ominus B \qquad [3.1.2-2]$$

与开操作呈对偶性，B 对 A 的闭操作就是 B 对 A 的膨胀，紧接着用 B 对结果进行腐蚀。

3.1.3　连通域计算

由于进行每一次形态学操作的结构元是一定的，单纯的形态学处理往往不能达到预期效果或者会使图像丧失部分信息。在这里，我们考虑使用通过求取连通域的方法，移除联通面积（即联通区域包含的总像素数）小于某特定阈值的区域。

在图像中，最小的单位是像素，每个像素周围有 8 个邻接像素，常见的邻接关系有 2 种：4 邻接与 8 邻接。4 邻接一共 4 个点，即上下左右。8 邻接的点一共有 8 个，包括了对角线位置的点。

3.1.4 椭圆检测

由于需要识别的交通标志全部为圆形（STOP 标志近似），所以我们可以将形状作为关键特征以提取交通标志并将其切割。由于具体的椭圆检测算法过于复杂，我们考虑使用提取物体轮廓的方法，大致流程如下：

- 检测二值图中的所有轮廓并标记，返回每一轮廓点的坐标与等级
- 考察同一等级的轮廓点，考察它们是否构成圆形，即是否距离某一中心点等距
- 在寻找到的所有椭圆里，选取**最大的**椭圆作为当前检测结果
- 将该椭圆轮廓点列表里横纵坐标的最大及最小值取出，组成图像的左上与右下端点后切割图像

3.2 LeNet 卷积神经网络分类器

LeNet 卷积神经网络（LeCun et al. [1998]）由 LeCun 在 1998 年首次提出，是一种用于手写体字符识别的非常高效的卷积神经网络。我们借用其思路将网络用于交通标志的分类。LeNet 卷积神经网络共有 7 层，其中不包含输入，每层都包含可训练参数；每个层有多个特征图，每个特征图通过一种卷积滤波器提取输入的一种特征，具有多个神经元。网络的结构图如图3所示。首先将输入图像尺寸统一归一化为 32×32。接下来对 7 层网络进行详细介绍。

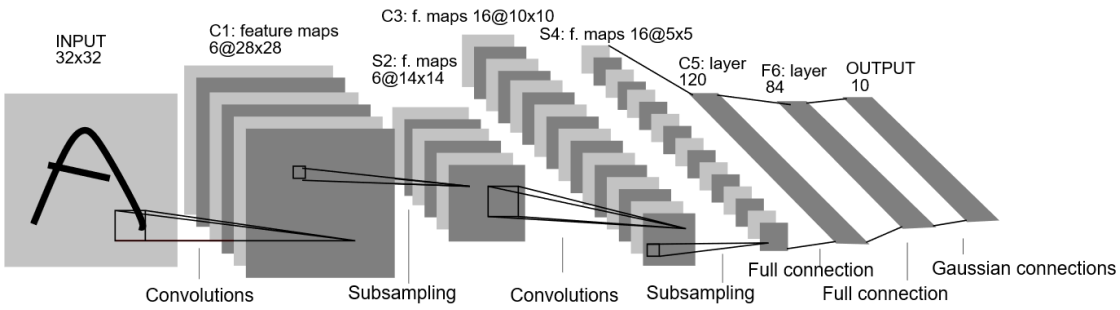


图 3: LeNet 网络结构示意图

3.2.1 C1-卷积层

使用 6 个大小为 5×5 的卷积核对输入特征图进行第一次卷积运算，得到 6 个大小为 28×28 的输出特征图。卷积核的大小为 5×5，因此总共就有 $6 \times (5 * 5 + 1) = 156$ 个参数，其中 $+1$ 是表示一个核有一个偏置。对于卷积层 C1，其中的每个像素都与输入图像中的 5×5 个像素和 1 个偏置有连接，所以总共有 $156 \times 28 \times 28 = 122304$ 个连接。

表 1: C1-卷积层参数表

名称	参数
输入特征图	32×32
卷积核尺寸	5×5
卷积核数目	6
输出特征图	28×28

表 2: S2-池化层参数表

名称	参数
输入特征图	28×28
采样区域	2×2
采样数目	6
输出特征图	14×14

3.2.2 S2-池化层（下采样层）

第一次卷积之后紧接着就是池化运算，使用 2*2 核进行池化，于是得到了 6 个 14×14 的输出特征图。S2 池化层是对 C1 中的 2×2 区域内的像素求和乘以一个权值系数再加上一个偏置，然后将这个结果再做一次映射。同时有 $5 \times 14 \times 14 \times 6 = 5880$ 个连接。

3.2.3 C3-卷积层

表 3: C3-卷积层参数表

名称	参数
输入特征图	14×14
卷积核尺寸	5×5
卷积核数目	16
输出特征图	10×10

C3 卷积层中的每个输入特征图是连接到 S2 中的所有 6 个或者几个特征图的，表示本层的特征图是上一层提取到的特征图的不同组合存在的一个方式是：C3 的前 6 个特征图以 S2 中 3 个相邻的特征图子集为输入。接下来 6 个特征图以 S2 中 4 个相邻特征图子集为输入。然后的 3 个以不相邻的 4 个特征图子集为输入。最后一个将 S2 中所有特征图为输入。这样操作有两个好处，其一是可以减少参数，减小训练量，其二是通过这种不对称的组合连接方式有利于提取多种组合特征。

表 4: S4-池化层参数表

名称	参数
输入特征图	10×10
采样区域	2×2
采样数目	16
输出特征图	5×5

3.2.4 S4-池化层（下采样层）

S4 是池化层，窗口大小仍然是 2×2，共计 16 个输入特征图，C3 层的 16 个 10×10 的图分别进行以 2×2 为单位的池化得到 16 个 5×5 的特征图。共有 $5 \times 5 \times 5 \times 16 = 2000$ 个连接。连接的方式与 S2 层类似。

3.2.5 C5-卷积层

表 5: C5-卷积层参数表

名称	参数
输入特征图	5×5
卷积核尺寸	5×5
卷积核数目	120
输出特征图	1×1

C5 层是一个卷积层。由于 S4 层的 16 个图的大小为 5×5，与卷积核的大小相同，所以卷积后形成的图的大小为 1×1。这里形成 120 个卷积结果。每个都与上一层的 16 个图相连。所以共有 $(5 \times 5 \times 16 + 1) \times 120 = 48120$ 个参数，同样有 48120 个连接。

3.2.6 F6-全连接层

全连接层的输入为卷积层 C5 输出的 120 维向量。再全连接层中，计算输入向量与权重向量之间的点积，再加上一个偏置，结果通过 sigmoid 函数输出。可训练参数为 $84 \times (120 + 1) = 10164$.

3.2.7 OUTPUT-全连接层

OUTPUT 层也是全连接层，共有 7 个节点，分别对应交通标志识别的 7 种结果——速度 10、速度 30、速度 80、左转、右转、停止以及其他。采用的是径向基函数（RBF）的网络连接方式。假设 x 是上一层的输入，y 是 RBF 的输出，则 RBF 输出的计算方式是

$$y_i = \sum_j (x_j - w_{ij})^2$$

3.3 SSD 目标检测算法

SSD 算法英文全名是 Single Shot MultiBox Detector. Single shot 指明了 SSD 算法属于 one-stage 方法，MultiBox 指明了 SSD 是多框预测。SSD 算法在准确度和速度上都比 Yolo 要好很多。对于 Faster R-CNN，其先通过 CNN 得到候选框，然后再进行分类与回归，而 Yolo 与 SSD 可以一步到位完成检测。相比 Yolo，SSD 采用 CNN 来直接进行检测，而不是像 Yolo 那样在全连接层之后做检测。其实采用卷积直接做检测只是 SSD 相比 Yolo 的其中一个不同点，另外还有两个重要的改变，一是 SSD 提取了不同尺度的特征图来做检测，大尺度特征图（较靠前的特征图）可以用来检测小物体，而小尺度特征图（较靠后的特征图）用来检测大物体；二是 SSD 采用了不同尺度和长宽比的先验框（Prior boxes, Default boxes，在 Faster R-CNN 中叫做锚，Anchors）（Liu et al. [2016]）。Yolo 算法缺点是难以检测小目标，而且定位不准，但是这几点重要改进使得 SSD 在一定程度上克服这些缺点。

SSD 采用 VGG16 作为基础模型，然后在 VGG16 的基础上新增了卷积层来获得更多的特征图以用于检测。SSD 的网络结构如图4所示。

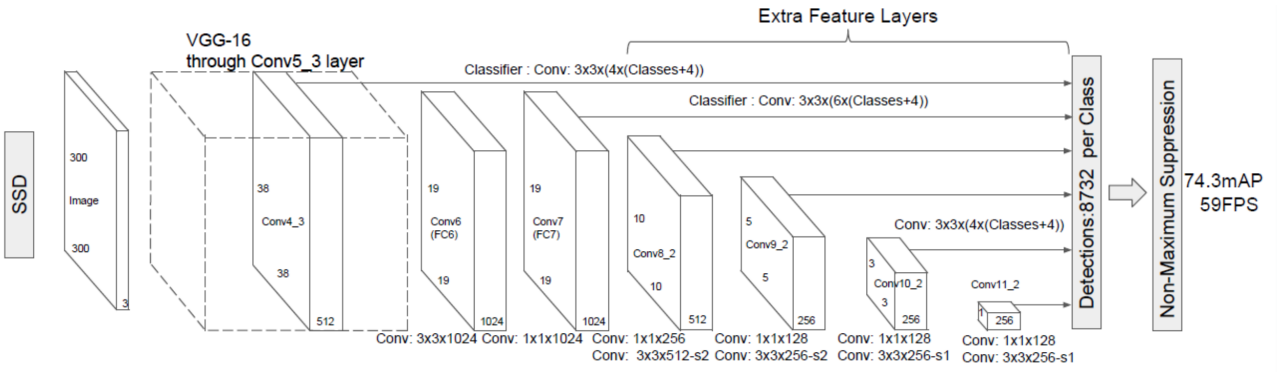


图 4: SSD 网络结构示意图

3.4 视频与指令传输

考虑到树莓派的计算能力不足，我们需要将数据传输到 PC 机之后完成复杂计算从而实现实时操作，即涉及视频与指令的传输。不同的协议标准有着各自的适应场景，由于 socket API 是面向运输层的，所以我们只需要考虑在运输层选择恰当的协议保证应用进程间能够进行端到端的、实时的逻辑通信。

3.4.1 协议选择

当前的互联网协议采用 TCP/IP 运输层，TCP/IP 运输层的两个主要协议都是互联网的正式标准，即：

(1) **用户数据报协议 UDP**（User Datagram Protocol）

(2) **传输控制协议 TCP**（Transmission Control Protocol）

TCP 协议5主要有下述特点：

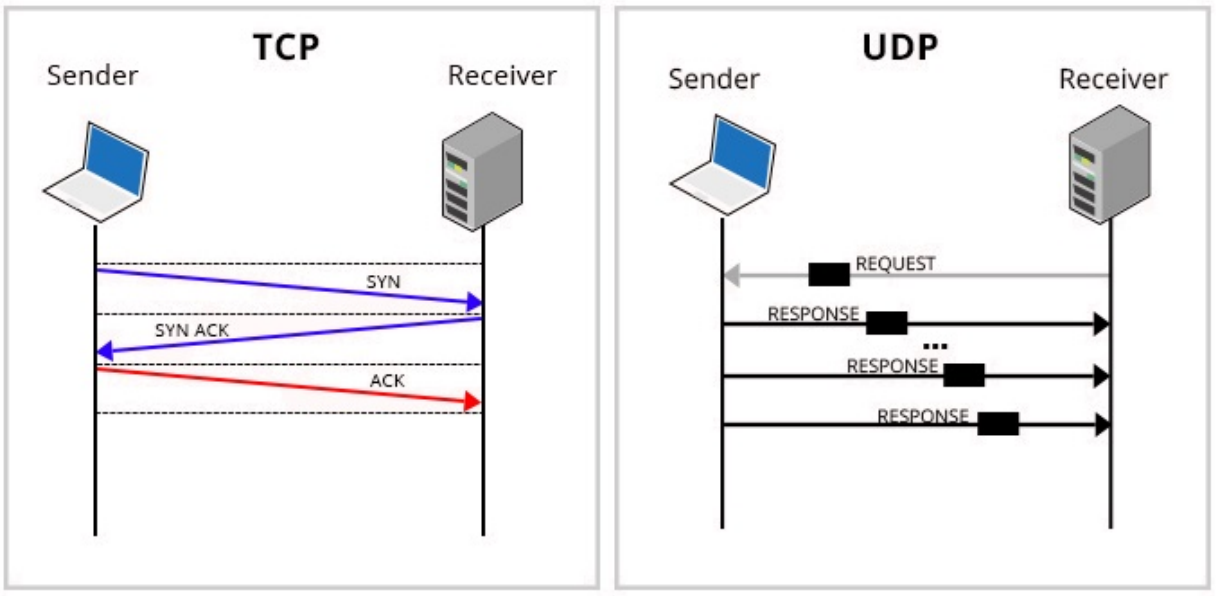


图 5: TCP 与 UDP 的传输方式比较

1. TCP 是**面向连接**的运输层协议。

2. 每一条 TCP 连接只能有两个端点 (endpoint)，每一条 TCP 连接只能是点对点的（一对一）。

3. TCP 提供**可靠交付**的服务。

4. TCP 提供**全双工**通信。

5. 面向字节流 TCP 中的“流”(stream) 指的是流入或流出进程的字节序列。
“面向字节流”的含义是：虽然应用程序和 TCP 的交互是一次一个数据块，但 TCP 把应用程序交下来的数据看成仅仅是一连串无结构的字节流。TCP 不保证接收方应用程序所收到的数据块和发送方应用程序所发出的数据块具有对应大小的关系。但接收方应用程序收到的字节流必须和发送方应用程序发出的字节流完全一样。

6. 信道利用率低。为了提供可靠通信，TCP 需要使用停止等待协议与自动重传请求，期间每接收到一条消息接收方都需要向发送方发送 ACK 确认帧，只有当发送方收到确认后才会继续发送下一消息，传输过程如下图6。那么 TCP 传输的信道利用率公式为：

$$U = \frac{T_D}{T_D + \mathrm{RTT} + T_A} \qquad [3.4.1-1]$$

其中，T_D 为分组发送时间，T_A 为 ACK 确认帧发送时间，RTT 为传输等待时间。

UDP 协议5主要有下述特点：

1. UDP 是**无连接**的，发送数据之前不需要建立连接，因此减少了开销和发送数据之前的时延。

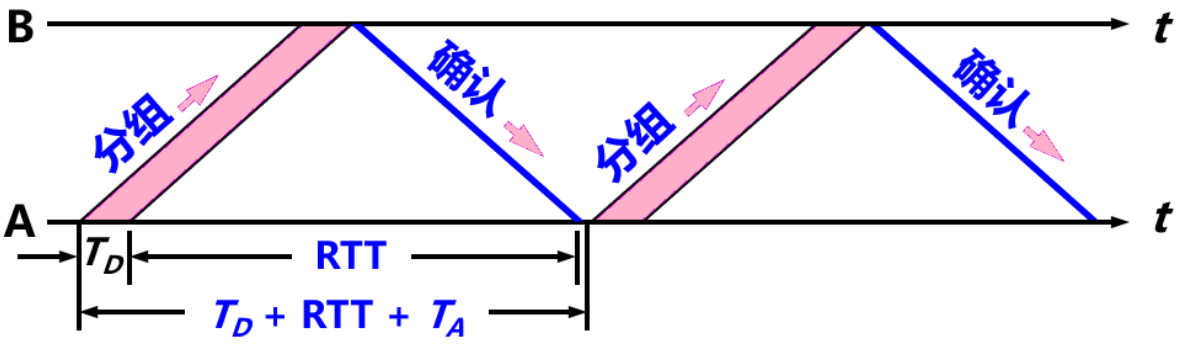


图 6: TCP 协议信道利用率计算示意图

2. UDP 使用尽最大努力交付，即**不保证可靠交付**，因此主机不需要维持复杂的连接状态表。

3. UDP 是**面向报文**的。UDP 对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界。UDP 一次交付一个完整的报文。

4. UDP **没有拥塞控制**，因此网络出现的拥塞不会使源主机的发送速率降低。这对某些实时应用是很重要的。很适合多媒体通信的要求。

5. UDP 支持一对一、一对多、多对一和多对多的交互通信。

6. UDP 的首部开销小，只有 8 个字节，比 TCP 的 20 个字节的首部要短。

根据上述讨论，考虑到视频传输中及时性比准确性更重要，我们在**视频传输时采用 UDP 协议，其中树莓派作为服务器端，PC 作为客户端**；相反地，为了保证指令的正确性，且因为指令数据长度较小而延时现象不明显，我们在**指令传输时采用 TCP 协议，其中树莓派作为客户端，PC 作为服务器端**。严格来讲，采用 UDP 协议通讯的两个进程是完全对等的，不存在客户端与服务端的区别，但由于在这里视频帧图像是单向通讯的，所以姑且命名为服务端与客户端以示两者间发送和接受的关系。

3.4.2　数据格式

- 由于摄像头采集到的帧率较高，如果对每一帧图像都进行标志及行人检测，那么一定会影响系统的实时性。于是我们人为的设定帧率为 **10fps**，并将帧图像按照 jpg 格式压缩，即相当于摄像头每隔 0.1 秒拍一次 jpg 格式的照片并发送给树莓派。之后发送端将帧图像进行 UTF-8 编码后发送，接收端接收到数据后进行 UTF-8 解码后复原图像。

- 指令共分为两种，其一是根据椭圆检测提取出的 ROI 得到的交通标志结果，其二是 SSD 行人检测模型判断当前帧图像中是否存在行人的结果。交通标志的结果共分为 7 类（6 类动作指令，1 类空指令），行人检测的结果分为两类（鸣笛指令与空指令各一个）。两类指令全部用字符代表，在发送时构成一个 1*2 的字符型列表进行发送。接收端收到后通过列表切片取出相应指令，并根据指令映射表67进行相应操作。

表 6: 电机指令映射表

指令编码	'0'	'1'	'2'	'3'	'4'	'5'	'6'
对应操作	speed_10	speed_30	speed_80	turn_left	turn_right	stop	do_nothing

表 7: 音响指令映射表

指令编码	'0'	'1'
对应操作	No	Yes

3.5　电机与音响控制

3.5.1　树莓派 GPIO

树莓派 3B+ 上有 40 个 GPIO 端口，我们可以为其分配高电平（3.3V）或低电平（GND），进而通过 51duino 控制电机转动。要使用这些端口，首先我们要导入能够驱动树莓派 GPIO 管脚的 RPi.GPIO 库。

RPi.GPIO 同时支持 BOARD 与 BCM 规则两种引脚编号。BOARD 规则根据树莓派上丝印层的序号驱动 GPIO，而 BCM 规则根据每个管脚固有的名字驱动 GPIO。具体的引脚分布如下图7所示。51duino 的电机驱动板[1]目前只支持 BCM 规则。

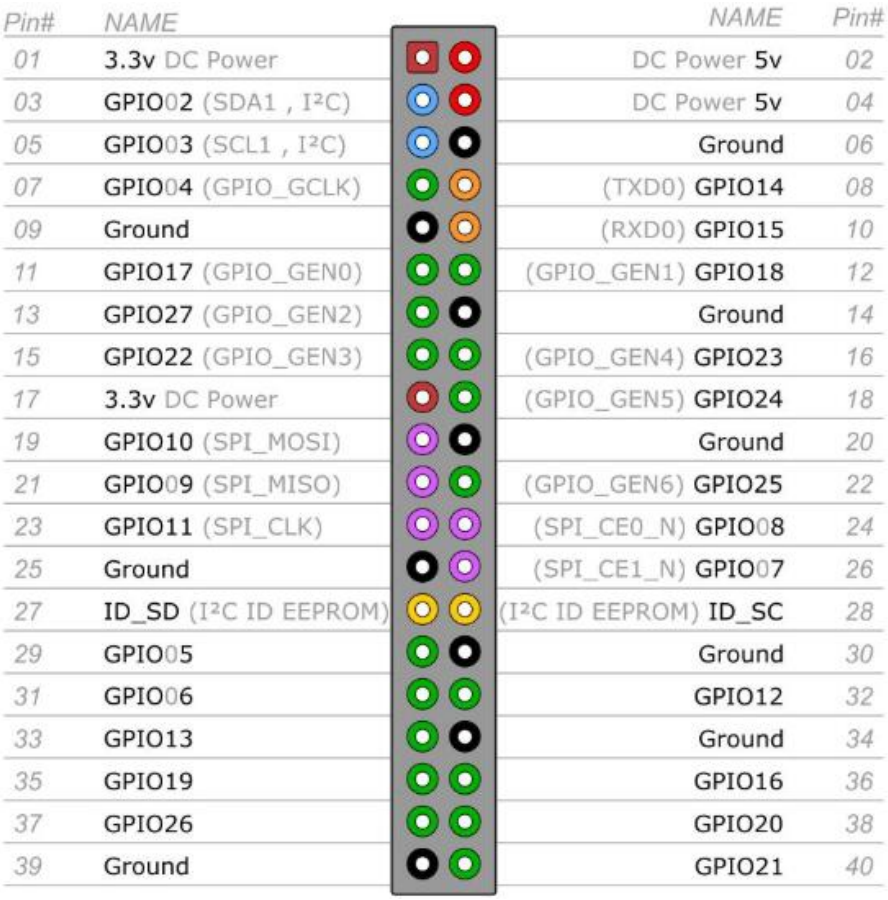


图 7: 树莓派 GPIO 管脚分布图

[1]驱动板手册 http://www.wifi-robots.com/thread-8996-1-1.html.

在使用一个引脚前，需要通过 GPIO 设置这些引脚作为输入还是输出。接下来就可以通过设置某个 GPIO 引脚的电平来驱动电机。综合利用上述步骤，我们就可以准确控制每一时刻每个 GPIO 引脚的电平高低，进而控制电机的转动。

51duino 驱动板相应的 GPIO 管脚号如下表8所示。

表 8: 驱动板电机控制 GPIO 管脚号

管脚名	GPIO 编号	功能
ENA	13	右轮使能
ENB	20	左轮使能
IN1	19	右轮正转
IN2	16	右轮反转
IN3	21	左轮正转
IN4	26	左轮反转

3.5.2 转弯与变速原理

- **转弯**：转弯的功能是通过设置两边的车轮之间的差速来完成的。由于该 51duino 电路板对电机的控制只有是否转动以及正转/反转三个 GPIO 电位控制，因此我们对转弯的实现是控制电机使一边的车轮以一定速度转动、另一边的车轮不转动，如此运行一段时间以实现转动一定角度。由于小车上并没有搭载可以获得小车角度的传感器，因此要达到转动 90 度的目标只能通过设置转动的时间，而这个时间只能通过不断尝试来得到。与此同时，我们利用有限状态机的原理，为不同的运动状态进行编码，当转弯结束后，小车会自动返回上一运动状态，这也与实际认知相符。

- **变速**：同样，由于驱动板的限制，我们无法直接控制电机的转动速度；我们采取的方法是控制电机高电平持续时间的占空比：即不是始终让电机运行，而是让电机在一个小循环中的部分时间内运行、部分时间内不运行，并不断重复这个小循环。又由于每个循环运行时间非常短，所以小车在表现上并不是走走停停、而是速度变慢了。综上所述，我们通过这种控制占空比的方法间接控制电机的转动速度，占空比越小，电机的转动速度就越慢。

3.5.3 树莓派音响

树莓派 3B+ 实验板上带有 3.5mm 耳机接口，将其与外接音箱连接就可以播放声音。我们在树莓派上安装了 omxplayer，它是一款能够在命令行运行的高保真音视频解码软件。之后我们在程序中输出播放指令到命令行，omxplayer 就可以解码文件目录下指定的音频文件并输出到外接音箱。

```
os.system('omxplayer -o local XXXXX.mp3')
```

os.system 语句具有阻塞当前主进程的功能，所以我们无需其它命令就可以让小车在鸣笛的同时停止运动。

4　系统测试与评估

根据上述算法原理，我们搭建整体系统如下图8所示。其中，树莓派作为整个系统的核心，主要完成信息收集与信息转发的功能。

测试时，需要在树莓派和 PC 上各开启两个进程用于各端的计算，且需要在每个进程间各开启两个线程分别进行视频与指令的传输。值得强调的是，我们没有使用小 R 科技公司提供的架构，因为其为了实现整个系统的功能而冗余缓慢。**自始至终的所有程序和脚本文件，除了驱动板的烧录程序外，全部由我们自己编写**。所有程序的实现代码已经在 Github[2]中开源。

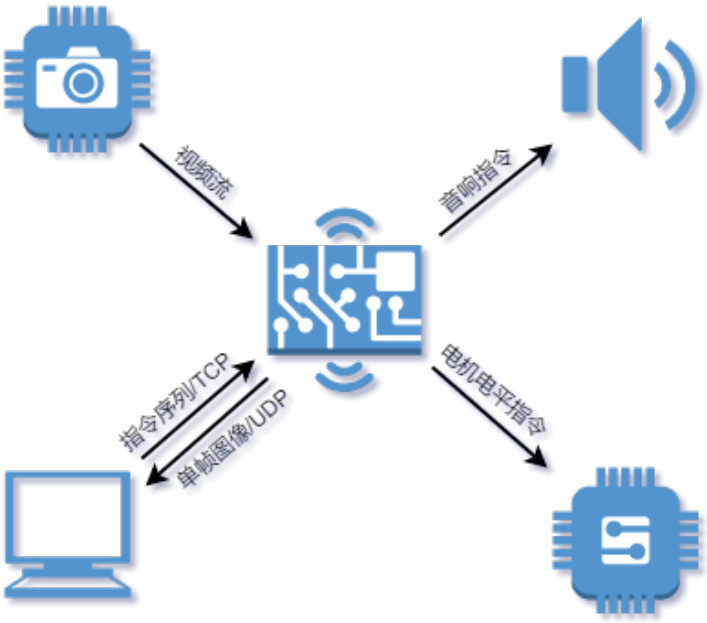


图 8: 整体系统结构

4.1　性能指标量化

以下所有结果，均为在具有 $Intel(R)\,Core(TM)\,i7-8650U\,CPU\,@\,1.90GHz\,2.11GHz$ 中央处理器，$8GB$ RAM 及 64 位 windows10 操作系统的 Surface Laptop2 笔记本电脑上测试所得。

4.1.1　算法效率

我们在 800*600 的 3 通道 RGB 输入图像上进行了算法的测试，程序计时如下9：

表 9: 算法执行时间测试表

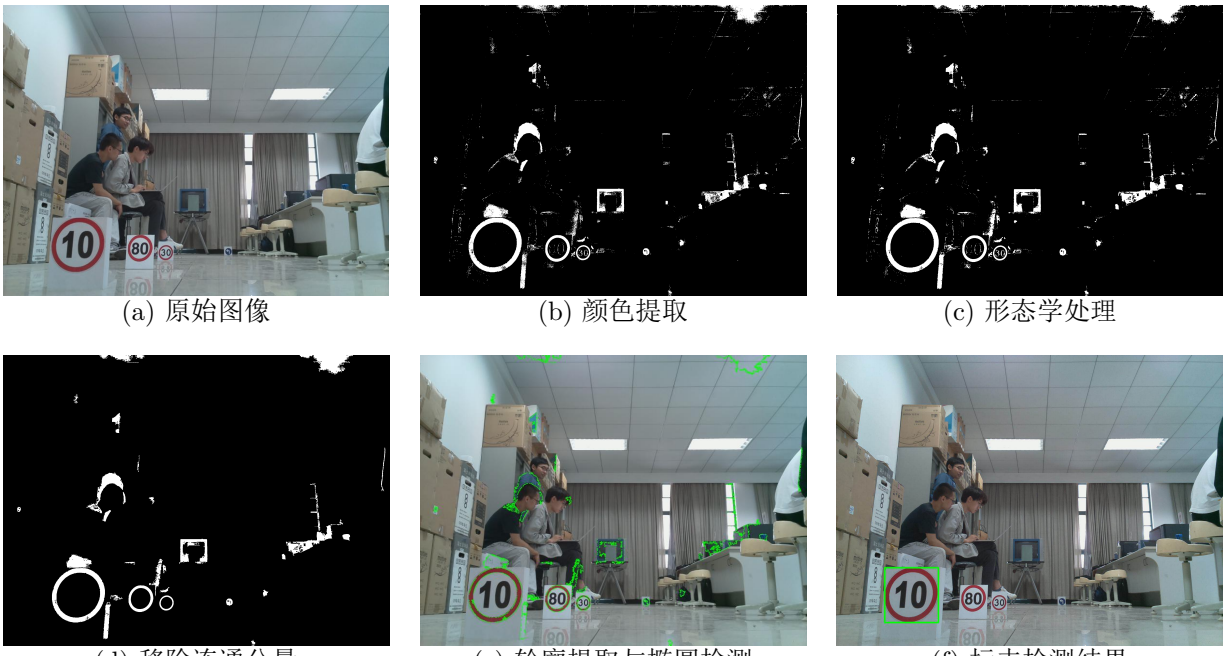
算法	颜色提取 + 闭操作	连通分量移除	边界提取	椭圆检测与选择	类别预测
时间 (ms)	4.027	4.623	0.432	2.972	13.185

[2]代码地址 `https://github.com/BrandonHanx/Traffic-sign-detection`.

如果累计上函数调用的时间，从单帧图像输入到交通标志类别提取仅仅需要 **43.135ms**。而我们传输的帧率为 10fps，也就相当于两张传输图片之间的间隔是 100ms，由于传输时间间隔大于图像计算时间，所以单纯的交通标志识别算法不会影响系统的实时性。如果我们开启行人检测功能，由于 SSD 网络较深，处理一张图片大约需要 40ms 左右，如果将两个网络级联，实时性会稍微收到一点影响。

4.1.2　预处理结果

下图9按照预处理的顺序，依次给出了每一步的结果输出。可以看到我们的算法效果显著、识别精确，与预期结果相符。



(a) 原始图像　(b) 颜色提取　(c) 形态学处理

(d) 移除连通分量　(e) 轮廓提取与椭圆检测　(f) 标志检测结果

图 9: 图像预处理过程输出

4.1.3　数据集获取

由于我们的交通标志分类任务是需要适应课程考核要求的，所以不能将网络上的公开数据集那来直接使用。我们通过编写程序手动控制小车移动，并在小车前依次放置不同的标志，之后命令程序自动将检测到的交通标志裁剪下来，从而获得了大量图像。其中，小车移动由如下图10所示的我们自己编写的遥控器控制完成。

为了提高模型的泛化能力，我们在网上公开的中国交通标志数据集[3]中取出一些图片作为补充。之后我们将所有图片随机的分为测试集和训练集用于模型训练，并在输入前进行数据增广运算（包括旋转、平移以及放缩）。上图11是一个类别中的部分图片，可以看到图像形式的覆盖范围较广。

[3]数据集地址 `http://www.nlpr.ia.ac.cn/pal/trafficdata/recognition.html`.

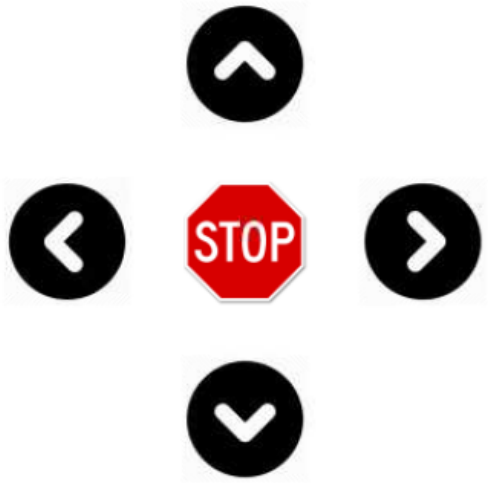


图 10: 遥控器 UI

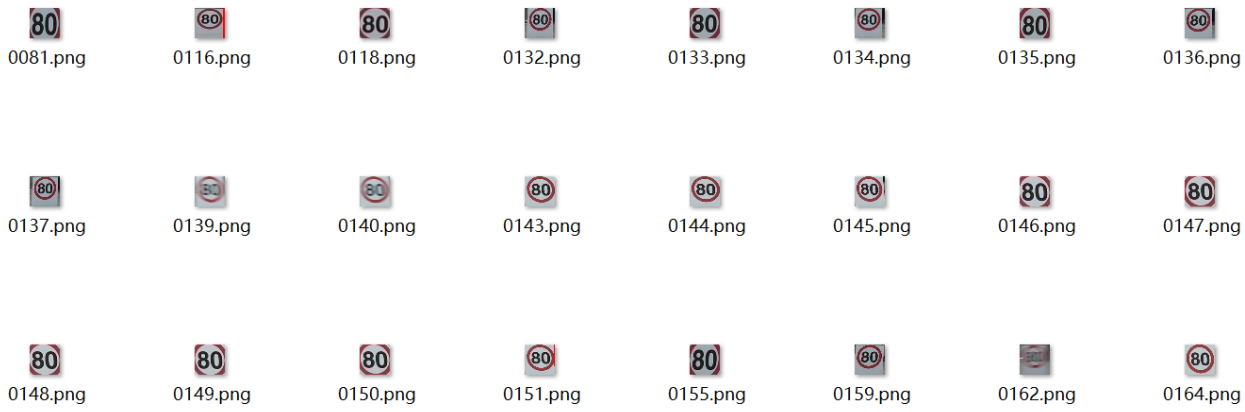


图 11: 部分数据集图片示例

4.1.4　模型准确率

对于交通标志识别任务，我们在自己构建的数据集上进行了模型的训练。由于分类任务较简单，为了提高计算速度，在读入图像之后我们统一将它们裁剪到 32*32 的大小。

训练参数设置：

- 学习率 $learning\_rate = 0.001$
- 批次大小 $batch\_size = 32$
- 迭代次数 $epochs = 50$
- 优化策略选择 $Adam$

在与上文相同的计算环境中，模型快速收敛，运算执行时间不足 3 分钟。训练过程在训练集和验证集上的损失函数和准确率变化曲线如下图12所示：

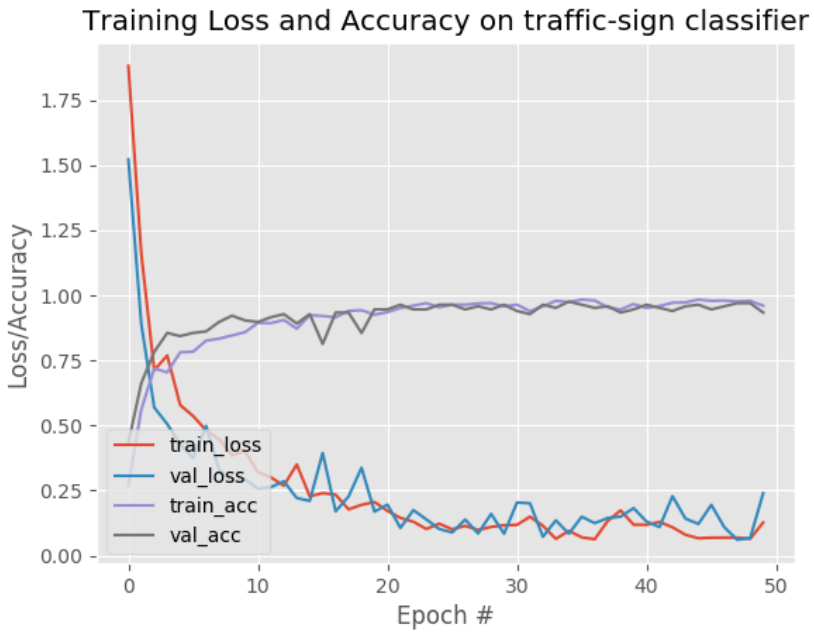


图 12: LeNet 训练曲线

最终得到的结果为：训练集损失值 0.0494，训练集准确率 0.9832，验证集损失值 0.1068，验证集准确率 **0.9765**。

对于行人检测任务，由于计算能力有限，我们直接使用了别人的模型[4]，而没有自己重新训练。此模型是一个在 caffe 平台下实现的 MobileNet-SSD 目标检测网络，采用 VOC0712 的预训练权重，且 mAP=0.727.

4.1.5　传输性能

我们将 PC 机作为上位机，在其上完成图像预处理、椭圆检测、交通标志分类和行人检测的功能并发出指令；将树莓派作为下位机，在其上完成视频采集和指令执行。

- 对于视频传输，由于采用 UDP 协议，存在视频帧丢包现象，但并不影响系统的实时性。
- 对于指令传输，采用 TCP 协议，由于树莓派执行指令需要一定时间，我们人为的将树莓派在执行指令时接收到的指令全部丢弃，**每次执行完一条命令后根据指令队列中的最后一条指令进行相应操作**，这样即保证了指令接受的及时性也实现了指令执行与图像识别的频率匹配。

[4]模型参考地址 `https://github.com/chuanqi305/MobileNet-SSD`.

4.2 优势与不足

相比于其他组以及往届学长的成果，我们的系统由如下显著的优势：

1. 我们实现的系统**把第三方依赖降至了最低**，除了驱动板的烧录程序外，其余程序和脚本全部由我们自己编写；
2. 采用了**可靠且简单的图像预处理方法**，达到了拔群的检测效果；
3. 在自己构建的数据集验证集上，达到了 **97.65%**的分类准确率，超越前人**三个百分点**；
4. 摆脱框架约束，创新性的使用 **UDP 协议传输视频，将视频传输延时压缩到 40ms 之内**；
5. 在原有功能的基础上，创新性的**加入了 SSD 行人检测模型及鸣笛报警机制**，且加入后不影响实时性；
6. **充分考虑了小车驱动算法的合理性**，如小车只会执行检测到的最大标志所代表的命令以及小车在转弯后会继续执行上一条命令等。

当然，我们的系统也不可避免地存在一些不足和亟待解决的问题：

1. 经过实践，我们发现小车车轮的转动速度与电池电量有关，而我们的方法中转弯角度与车轮转速有很大关系，所以随着电池电量的下降小车的转弯角度将会略小于 90 度；
2. 我们使用的椭圆检测方法只能够检测圆形或近似为圆形的路标，没有办法检测三角形路标，与实际需求不符；
3. 我们的检测准确率在一定程度上受光照条件的影响，若要应用于实际场景的话天气将是一个挑战因素；
4. 我们的方法目前没有办法应对路标被部分遮挡的情况。

4.3 未来优化

我们对于目前存在的不足提出了一些可行的解决方案：

1. 可以通过添设更多的传感器来获得小车角度的信息，直接控制转弯的角度；
2. 可以通过更改检测方法来使算法能够应对更多形状的路标，也可以解决部分遮挡问题，但这必然导致检测速度下降；
3. 可以通过更多的预处理手段来解决光照条件不同的影响；
4. 虽然现在的图像传输速度已经比较令人满意了，但通过网线直连等方法还可以进一步提高图像传输速度。

5 总结与展望

本文基于椭圆检测算法和 LeNet 卷积神经网络，提出了一种高速实时的交通标志识别与小车控制算法，并利用 SSD 目标检测算法实现行人检测及报警。算法在测试用数据集以及实际测试场地中都实现了极高的准确性与良好的扩展性，能够满足未来无人驾驶系统的实时性与可靠性要求。交通标志识别与计算平台的通讯是基于 UDP/IP 和 TCP/IP 协议的服务器 -客户端模式，在树莓派上完成视频流抓取，传至电脑完成运算处理，再将小车控制指令返回树莓派。树莓派通过 GPIO 库控制 51duino 电机驱动板，实现小车运动。

未来的汽车已经不仅局限于一种交通工具，更多的是向新一代互联网终端发展。无人驾驶汽车将感知、决策、控制与反馈整合到一个系统中，实现了汽车脱离驾驶员而能保证其驾驶操纵性与安全性。无人驾驶的出现将从根本上改变传统汽车的控制方式，对于交通系统的安全性与通行效率有了较大保障。随着大数据、物联网、云计算的不断深入发展，无人驾驶汽车的性能将会更加完善。

无人驾驶作为一个新兴的市场，在技术、经验上都需要不断地加深研究，无人驾驶的系统搭建完善到大范围的无人驾驶车辆开始运营，还有很漫长的路要走。但是，无人驾驶汽车已然成为汽车行业的发展趋势，可以肯定的是，在未来，无人驾驶汽车会进入我们的生活，在很大程度上改变和改善我们的出行方式，在环保、交通、经济等领域为人类谋得福祉。我们相信在未来的不远，无人驾驶汽车将会奔驰在中国的大地上。

6 参考文献

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

David G Lowe et al. Object recognition from local scale-invariant features. In *iccv*, volume 99, pages 1150–1157, 1999.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.