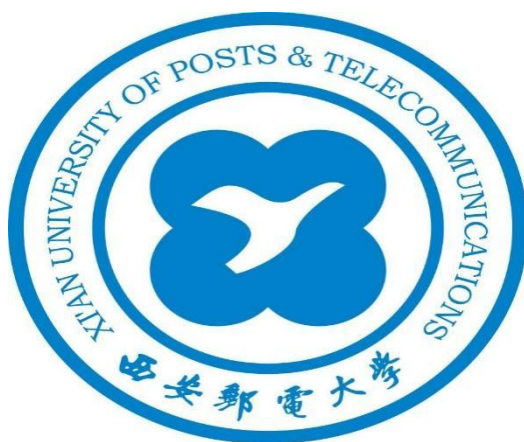


第十七届全国大学生智能汽车竞赛

百度完全模型竞速组

技 术 报 告



学 校：西安邮电大学

队伍名称：学长说的都队

参赛队员：梁德懿 史历 苟迦轩 马

远杰 段宝玉

带队教师：杨春杰 任锦瑞

关于技术报告和研究论文使用授权的说明

本人完全了解全国大学生智能汽车竞赛关于保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和恩智浦半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：梁德懿 马远杰 苟迦

轩 史历 段宝玉

带队教师签名：杨春杰 任锦瑞

日 期：2022.8.18

目录

第一章 引言	1
第二章 系统设计方案	2
2.1 系统结构框图	2
2.2 主要设备选型	2
第三章 系统硬件设计	4
3.1 总体硬件设计及电路图和说明	4
3.2 各模块介绍及硬件电路图	5
第四章 系统循迹软件设计	11
4.1 软件控制程序的整体思路	11
4.2 灰度图像处理思路	11
4.3 基本循迹扫线思路	12
4.4 赛道中线的拟合	13
4.5 十字识别与赛道中心线修补	14
4.6 赛道类型的分离	15
4.7 伺服器（舵机）转向打角控制策略	16
4.8 电机转向差速控制策略	17
第五章 识别部分代码软件设计	19
5.1 控制思想：	19
第六章 总结	20
参考文献	21

第一章 引言

全国大学生智能车竞赛是从2006年开始,由教育部高等教育司委托高等学校自动化类教学指导委员会举办的旨在加强学生实践、创新能力和培养团队精神的一项创意性科技竞赛,至今已经成功举办了十四届。在继承和总结前十四届比赛实践的基础上,竞赛组委会努力拓展新的竞赛内涵,设计新的竞赛内容,创造新的比赛模式,使得围绕该比赛所产生的竞赛生态环境得到进一步的发展。

为了实现竞赛的“立足培养、重在参与、鼓励探索、追求卓越”的指导思想,竞赛内容设置需要能够面向大学本科阶段的学生和教学内容,同时又能够兼顾当今时代科技发展的新趋势。比赛形式包括有竞速比赛与创意比赛两大类。竞速比赛中包含不同的组别,难度适合本科不同年级本科生学生参赛。在竞速赛基础上,适当增加挑战性,形成创意比赛的内容,适合部分有条件、能力强的本科生和研究生参加创意比赛。

本文采用第十七届全国大学生智能车竞赛的汽车模型作为研究平台,以32位单片机TC264和百度大脑Edgeboard作为控制单元,道路信息检测模块普遍采用简单、速度快的S320摄像头。本模块采用工业级芯片,以COMS传感器作为感光器件。其芯片最高分辨率为1280*720,以每秒30帧(fps)的形式输出。本届车模后置官方指定单电机RS-555,车模转向采用CS-3120官方指定伺服舵机。

本篇技术报告将从智能车对整体方案、机械结构、硬件电路、图像处理、图像识别、控制算法等方面详细介绍整个准备过程。

第二章 系统设计方案

2.1 系统结构框图

光电智能车系统的总体工作模式为：CMOS 图像传感器拍摄赛道，输出 8 位灰度信号及行场像素中断，主控制器检测各中断信号进行图像采集，同时使用伺服器和电机对于车模姿态和行进进行控制，并通过速度传感器等不同传感器获取车模姿态，进行闭环控制。

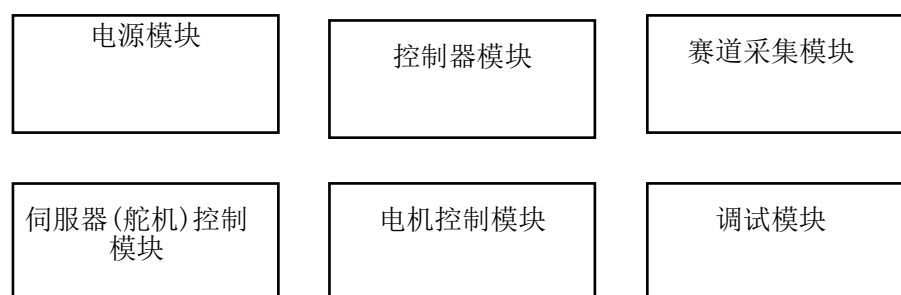


图2.1 系统框图

系统框图如上图1所示，该系统总共包括6个子系统：电源模块、主控制器模块、赛道采集模块、伺服器（舵机）控制模块、电机控制模块、调试模块。

2.2 主要设备选型

2.2.1 主控制器模块：

使用Infineon公司的SAK-TC264D-40F200N BC单片机(以下简称TC264)作为智能车系统的控制器。该单片机是功能强大的AURIX™微控制器，适用于汽车和工业应用，频率为133 MHz，闪存高达1MB，其创新的多核架构基于多达三个独立的32位TriCore CPU，旨在满足最高的安全标准，同时显着提高性能。TC26xD 的产品配备了高达 200 MHz 的双 TriCore、5V 或 3.3V 的单电压电源和功能强大的通用定时器模块旨在降低复杂性、实现一流的功耗和显著的成本节约，充分满足智能车设计需要。

2.2.2 赛道采集模块

使用赛曙科技提供的S320摄像头作为光电传感器进行赛道图像采集，该摄像头与控制器百度大脑Edgeboard进行通信，进而配置摄像头的相关信息，通过使用场中断触发主控制器开始图像采集、像素中断控制采集速度。

2.2.3 伺服器（舵机）控制模块

使用CS-3120舵机进行车模转向控制。

2.2.4 电机控制模块

使用RS-555电机进行车模的速度控制。

2.2.5 调试模块

调试模块由LCD屏幕、组合按键、无线透传串口组成。

LCD屏幕采用逐飞科技的1.8寸TFT彩屏，满足图像参数显示的需要；组合按键分为加、减、保存共3个按键，满足对于参数的微调需要；另外同时采用两个两路拨码开关，用于控制显示内容；无线透传串口使用逐飞科技的无线透传模块，可以在智能车运行中实时传输控制参数和赛道信息；

第三章 系统硬件设计

3.1 总体硬件设计及电路图和说明

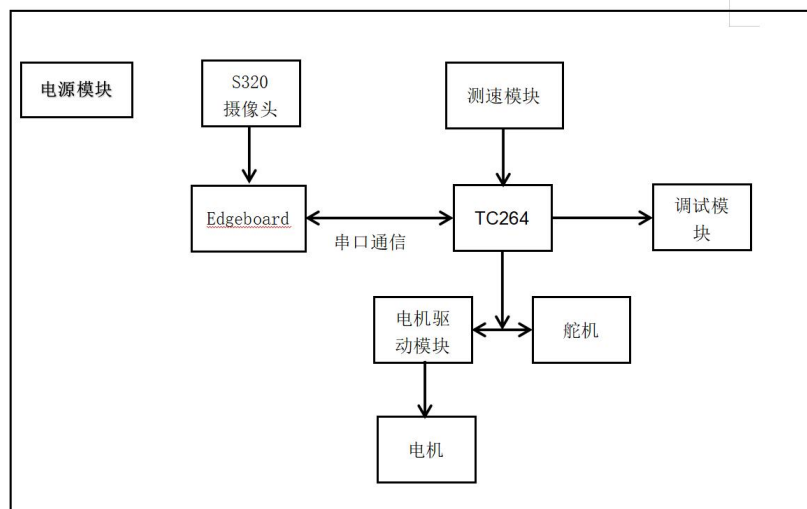
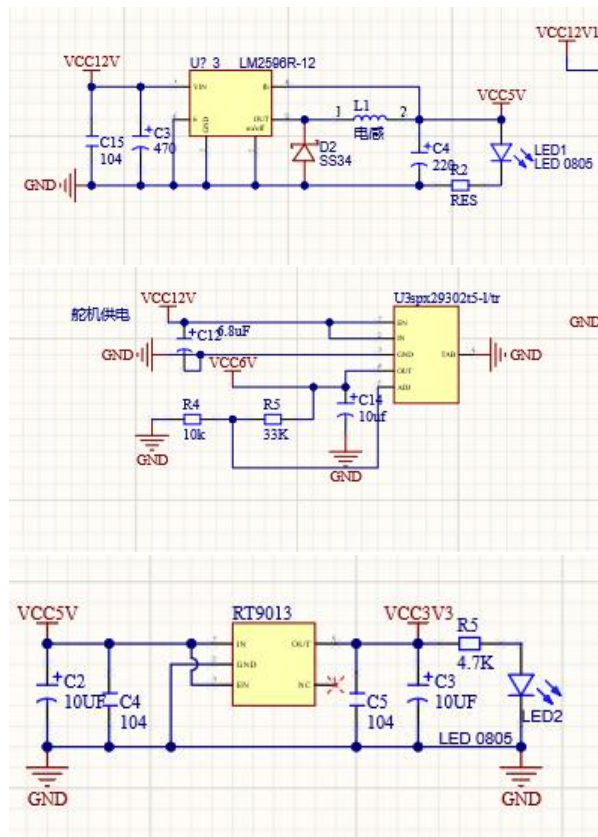


图3.1.1 总体硬件电路图

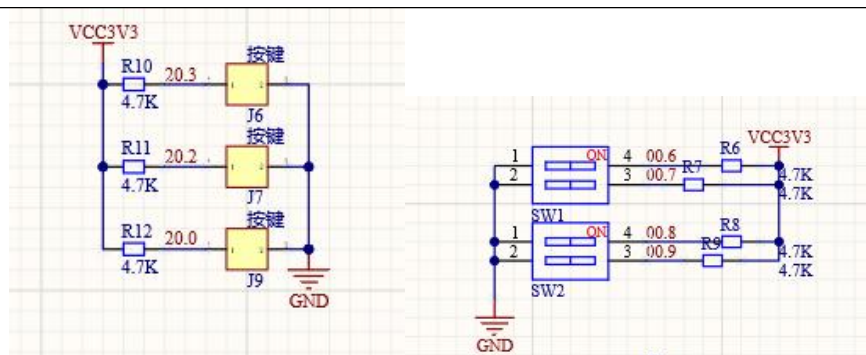


图3.1.2 电源管理模块总体框图



整辆小车都是用2200mAh的电池提供，但是各种元器件所需的电压不同，这就需要我们用不同的电源芯片来生成不同的电压。

3.2.3 拨码开关

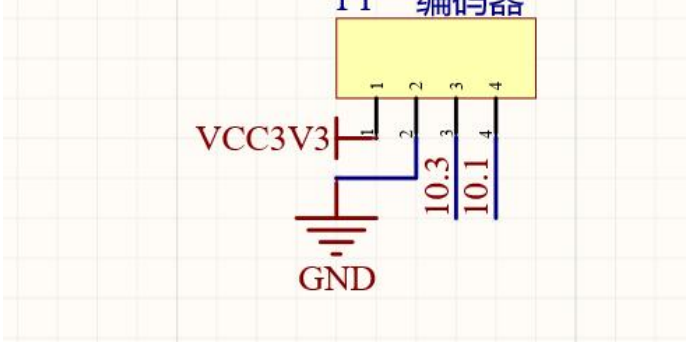


为了调参方便，我们用三路按键开关搭配2个2路拨码开关。

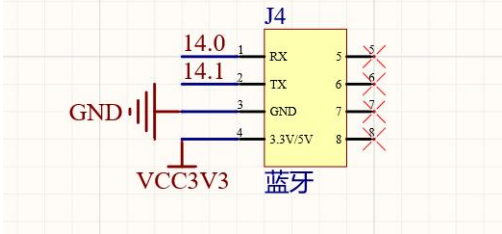
3.2.4 蜂鸣器电路

	<p>我们用三极管驱动蜂鸣器来实现判元素可视化。</p>
---	------------------------------

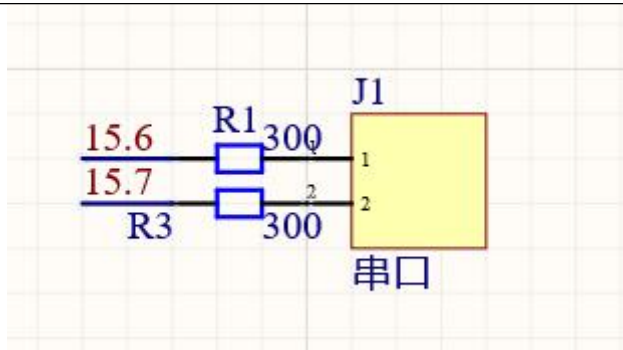
3.2.5 编码器电路

	<p>测速模块我们采用赛曙科技的CSPE5-500编码器。此编码器为AB相增量式光电编码器，精度为500CPR，满足需求。</p>
--	---

3.2.6 无线转串口电路

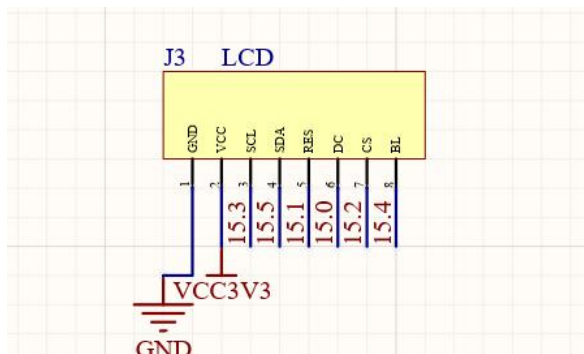
	<p>本串口采用高性能cortex-M3单片机（72MHz主频）来进行数据转发，转发速度更快。</p>
---	---

3.2.7 串口通信电路



百度大脑Edgeboard与TC264之间通过串口通信,满足之间的信息传输要求。

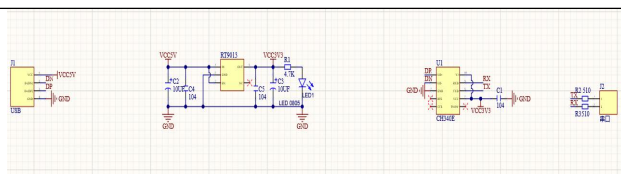
3.2.8 TFT屏电路



为使软件上参数调节方便,且兼顾硬件上的简洁与美观,我们特意将这部分设计在主板上,我们使用TFT显示模块。TFT体积小,分辨率高,功耗低。

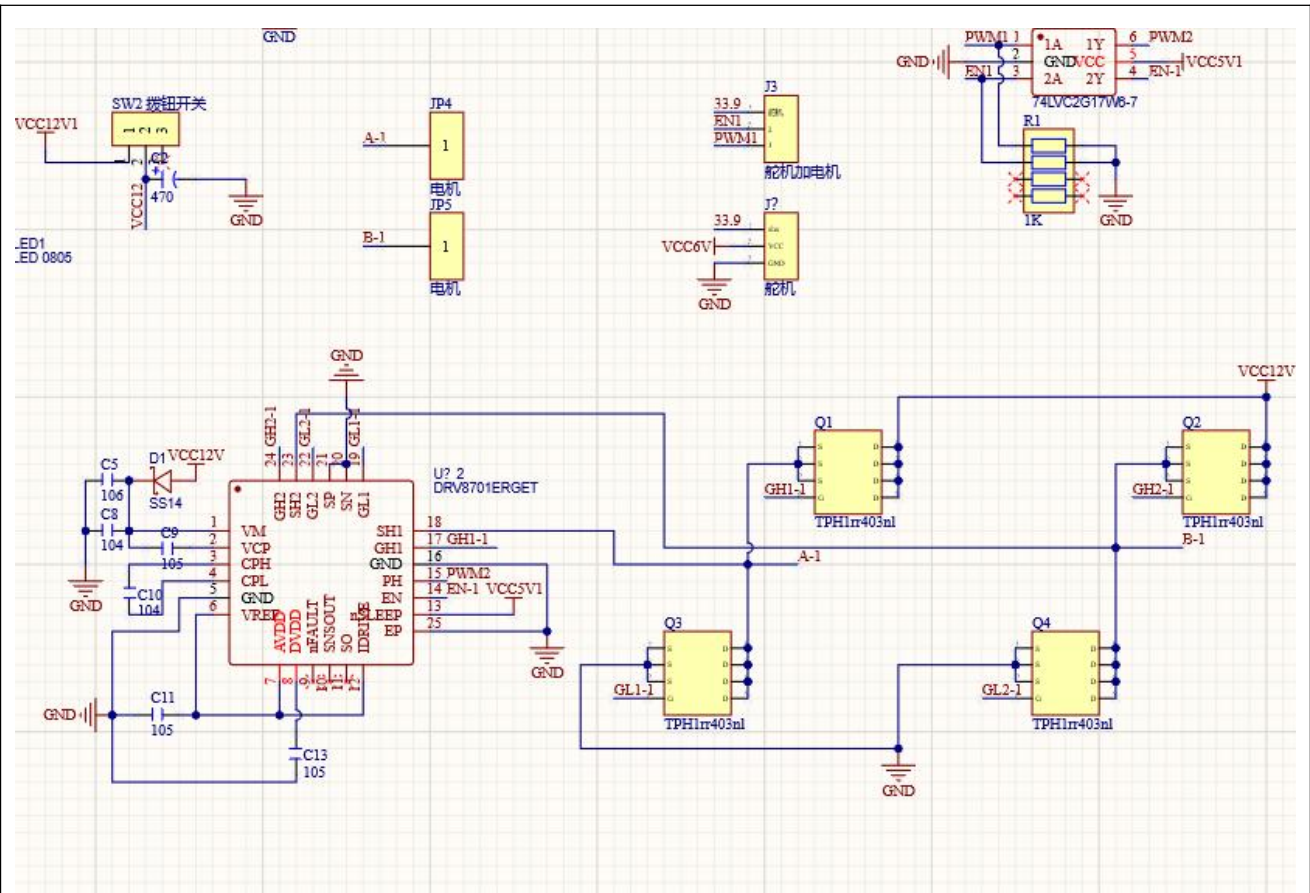
调试中,这个模块对参数的调试带来了极大的方便,可以减少烧录程序的次数而提高调试效率。

3.2.9 USB转TTL电路



此电路为Edgeboard与TC264之间传输电路的媒介。

3.2.10 驱动电路



电机驱动电路的作用指通过控制电机的旋转角度和运转速度，以此来实现对占空比的控制，来达到对电机变速控制的方式。

我们此次使用的是 DRV8701E驱动电路。DRV8701是一款采用4个外部N通道MOSFET的单路H桥栅极驱动器，主要用于驱动12V至24V双向有刷直流电机。并同时采用74LVC2G17W6-7隔离。

3.2.11 整车布局



舵机位置在前，立式安装，既保证响应速度又大多数符合阿克曼转角原理；电池固定在后方，平衡重心；CMOS摄像头通过碳杆、支架固定在主板后方，有效的采集赛道信息；外表车壳采用官方设计。

第四章 系统循迹软件设计

4.1 软件控制程序的整体思路

软件编写的目的，是控制车模在既定规则前提下，以最快的速度，跑完整个赛道。不论上哪种方案，软件的总体框架总是相似的，我们追求的就是稳定至上，兼顾速度。软件上大类分为图像采集、图像处理识别黑线，模型识别，速度控制以及速度反馈。

4.2 灰度图像处理思路

通过摄像头传输回来的图像是彩色图像，而处理图像的第一步就是先处理成为灰度图像，然后再对图像进行灰度处理，主要有以下两种思路：

4.2.1 二值化图像

通过给定一个阈值，将每个像素点与之进行比较，从而确定出来这个像素点是黑或白，之后的图像处理就使用这个二值化后的数组进行处理。

这种方法在图像处理的过程中仅仅有黑白两种元素，可以使图像处理更加简单。但由于摄像头高度较低，在实际的运行中当小车运行到不同的区域中的由于光线不均匀使得设定的单一阈值不同。

4.2.2 灰度图像处理

在处理灰度图像中，我们尝试了使用苏泊尔算子将图像进行卷积运算，运算之后的图像在黑白跳变处会呈现一个很明显的高值。

这种方法虽然在处理的过程中比较复杂，但是在对于虚线会车区、颠簸路段等元素的识别上更加容易。

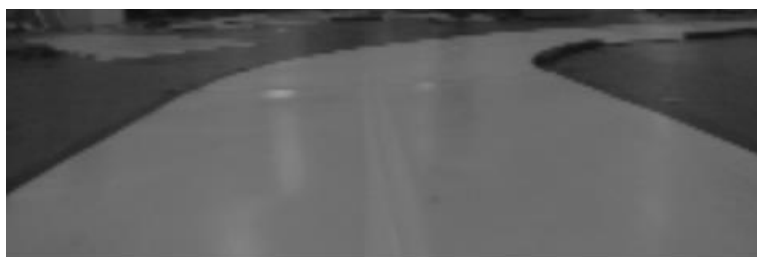


图4.2.2.1 原灰度图像



图4.2.2.2 苏泊尔算子卷积图像

4.2.3 动态阈值二值化处理

通过之前的尝试，我们在上位机仿真的过程中发现，使用最大协方差阈值法能够有效的计算出一幅图像的最佳阈值，实现对于图像前景与背景的分离。这样使用动态阈值二值化之后的图像能够像普通二值化图像一样处理，但在颠簸等黑白跳变不大的元素让需要借助灰度图像进行处理。

4.3 基本循迹扫线思路

4.3.1 中心扫线

中心扫线是指每行图像的起始扫线都是从上一行图像的中线位置开始的，从白点向左向右扫到黑点及寻找到所需要的边界线，进而拟合出赛道中心线。

优点：对于环境光线依赖比较小，如由于光线不均匀造成的反光仅仅会影响

反光存在的几行，而不会对整幅图像之后的边界线提取造成干扰；

缺点：比较消耗时序，需要遍历图像中大多数像素点才能够找到赛道的边界线，同时由于环岛等特殊元素的存在，使我们寻找的边界线与环岛的特征点有一定的差距，不利于元素的识别。

4.3.2 边缘扫线

边缘扫线是指每行图像的起始扫线根据左右边界线分别根据上一行左右边界线开始的，根据上一行的边界线在这行中同样位置的像素点黑白确定向左向右扫线，并找到黑白的切换点作为实际的赛道边界线。

优点：能够较少的利用时序，同时克服赛道中间白色区域中产生的噪点，同时可以较好的寻找环岛等元素的特征点；

缺点：若外界光线不均匀造成部分行的扫线错误，将一直影响之后整幅图像的扫线，同时不利于提取（斜入）十字等元素的特征。

4.3.3 根据不同的赛道元素类型，指定每行的扫线起始点

“根据不同的赛道元素类型，指定每行的扫线起始点”是指在每行扫线的过程中，同时对元素类型进行区分，根据先前确定下来的赛道元素类型明确扫线的方向及起始点。

如在正常扫线可以将下一行的扫线起点指定为本行的左右边线，在十字的判断中可以将扫线的起点通过十字前拐点下的直线斜率确定，方便寻找十字远

处的跳变点，而环岛扫线可以通过环岛阶段确定需要扫到的特征点进而确定扫线方向与扫线起始。

由此看来，这种方式的扫线可以最大限度的简化程序执行流程，同时我们可以在扫线的应用“搜索”中“剪枝”的思想。

如在扫线的过程中判断是否已经扫到了赛道外，如果已经扫出赛道即停止扫线，体现了“可行性剪枝思想”；在对十字和环岛的判断中，可以根据之前图像找到的特征点具体位置寻找这场图像的特征点，体现了“记忆化搜索思想”。

4.4 赛道中线的拟合

因为是双边的黑线赛道，所以我们在正确提取黑线信息的基础上，外加赛道中心拟合函数模块，其具体思想有以下几种。

4.4.1 已知左右线求边线

在已知左右线的基础上的处理最为简单，仅仅取每行左右线的中心位置即可得到近似的赛道中心线。

4.4.2 一边丢线的中线拟合

如果一侧的边线丢失，我们在最初的基础上使用最近一行能够找到的赛道宽度作为之后丢线区域的赛道宽度，之后的丢线区域就将已知边线平移该宽度的一半作为近似的赛道中心线。

4.4.3 丢线不严重区域的中线拟合

如果在丢线过程中全部按照 4.4.2 节中的思路进行拟合中心线，会导致在一些连续变向的弯道中有超调的问题，导致路径不太流畅。我们首先将正常直道的赛道宽度保存在一个数组中，之后再丢线不严重的区域寻找丢线边线的恢复行，计算恢复行的宽度相较正常直道赛道宽度的比值，再丢线区域按照这个比值，平移已知边线，拟合出需要的赛道中心线。

下图为上位机上的处理效果图。



图4. 4. 3. 1 原图像



图4. 4. 3. 2 左右边线和赛道中心线

4.5 十字识别与赛道中心线修补

十字元素是智能车光电组别近几年已知存在的赛道元素，我们创新了十字识别的思路，针对 20cm 高度摄像头图像进行一些调整。我们将十字分为斜入十字、直入十字、中入十字三种情况。

4.5.1 斜入十字



图 4. 5. 1 斜入十字图像

以上图举例具体说明斜入十字的识别思想：

- 1) 在右边界线过程中计算每两行边界线的差值，寻找边界线向外的跳变点并记录；
- 2) 从这个跳变点开始向上的扫线起始点就改为通过这个点下面一段线的斜率延伸上去；
- 3) 向上几行判断这个跳变点上下直线的斜率夹角，满足直角的范围；
- 4) 之后的扫线继续按照之前的斜率延伸上去，直到寻找到黑色像素点，即跳出了十字，便可确定下来斜入十字远端的行数；
- 5) 在补线的过程中确定斜入十字起始点和终点的位置，将赛道中心线改为这两个点相连接即可完成斜入十字的补线操作。

4.5.2 直入十字

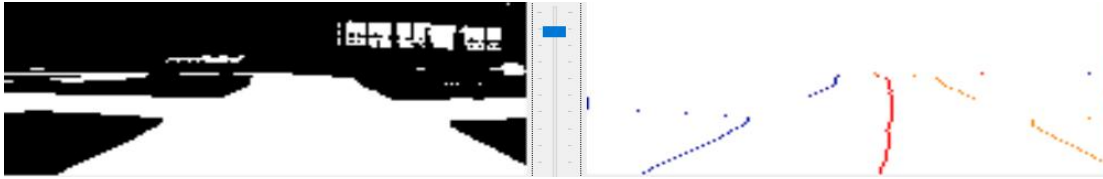


图 4.5.2 直入十字图像

以上图举例具体说明直入十字的识别思想：

- 1) 在左右边界线依次寻找边界线向外的跳变点，并且跳变点的上一行的边线在接近图像外边缘的位置，即边界线呈现近似垂直于镜头上前方向丢线，记录下来左右边线出现这种情况的行；
- 2) 在之后的扫线过程中，先确定出现边线在左右边界线附近同时丢线，再确定赛道宽度恢复正常直道情况的行；
- 3) 最后通过寻找边界线的恢复行产生直入十字的结束行，将中心线与起始行的中心线相连，即得出直入十字的赛道中心线。

4.5.3 中入十字



图 4.5.3 中入十字图像

以上图举例具体说明中入十字的识别思想：

- 1) 首先的识别前提条件在于前面存在直入十字或者斜入十字的情况，图像的显著特征在于图像下方有很大的左右边界线同时丢线的区域，当丢线达到一定的条件的时候可以确定这是中入十字；
- 2) 之后寻找边界线向内收缩并且赛道宽度恢复到正常直道范围的行数，确定出十字的结束行；
- 3) 由于 10cm 高度摄像头前瞻较短，不利与分析出十字之后赛道中心线的前进方向，我们直接将出中入十字行的中心线竖直地拉下来作为赛道的中心线。

4.6 赛道类型的分离

4.6.1 直道计算算法

赛道分离主要是将直道和弯道分离开，其关键在于将直道识别出来。

我们逐行处理赛道的中心线，连接每行赛道的中心线和第一行赛道的中心线，并计算这行以下图像拟合出来的赛道中心线与这行直线举例的差值，如果差别很大，就跳出这个循环，而这行就作为图像的直道有效行，可以认为第 1 行到这行是直道。

4.6.2 直道弯道分离思路

计算第 1 行中心线到之前计算出来的直道有效行的斜率，与直道有效行数一起参与直道弯道分离的协同计算。

其中，直道有效行非常高且直线较直的作为“长直道”，直道有效行较高且直线较直的作为“短直道”，直道有效行不高（仍存在一段直道）且斜率较小的作为“直入弯”，其他基本作为“弯道”。

按照上述完成对于赛道元素的分离，根据元素的不同采用不同的控制策略。

4.7 伺服器（舵机）转向打角控制策略

4.7.1 舵机整体控制思路

舵机通过一个由库函数输出的 PWM 波控制，通过各种参数的对比之后，我们确定使用 300Hz 的 PWM 波能够使舵机响应最快同时使舵机工作稳定。

在每个控制周期中，每当图像处理完成，我们根据不同的赛道类型和速度传感器返回的实际转速综合分析得出最佳的舵机打角行，计算这一行的中线偏离差值，带入到舵机位置式 PD 的计算中，在舵机中值的基础上经过舵机上下限的限幅计算得到实际的舵机值。

4.7.2 舵机PD计算式

舵机控制采用位置是 PID，经过尝试我们认为 I 项（积分项）不适用与舵机的控制中，便将 PID 简化为舵机的 PD 控制，控制公式如下：

$$PID_out = dir_P * dir_err + dir_D * (dir_err - dir_l_err);$$

4.7.3 动态打角行的选取

由于车模不是处于匀速的运动中，不能够通过固定的打角行来计算偏差。我们采用根据编码器的速度返回值计算出来当前打角行的高度，其策略是速度越快打角行越高，目的是使不同的速度下小车的路径相近，达到小车整体的闭环控制。

4.7.4 不同元素的舵机PD 选取

当车模行进到不同赛道元素的时候，如果同样的舵机 PD 会造成在直道上抖动和弯道上的转向不连续的问题，由此我们根据不同的赛道类型选取了不同的舵机控制 PD，使得能够在直道跑的直，在弯道流畅，同时在 S 型弯道能够基本沿着中线行进。

其中不同元素的 PD 选取有以下的规律：直道 PD 小，弯道 PD 大，S 型

弯道 P 小 D 略大。

4.7.5 电磁舵机PD控制表达式

电磁舵机控制我们依旧采用位置式 PID，经过尝试我们认为 I 项（积分项）不适用与舵机的控制中，便将 PID 简化为舵机的 PD 控制，此外我们加入了陀螺仪控制，以此解决了 C 车电磁短前瞻下车模控制抖动问题。控制公式如下：

$$\text{PID_out} = \text{dir_P} * \text{dir_err} + \text{dir_D} * (\text{dir_err} - \text{dir_l_err});$$

4.7.6 电磁舵机PD模糊控制

电磁舵机控制由于前瞻限制，为达到更好的转向效果，我们编写了模糊算法，采取PD模糊控制。综合考虑之后，我们选取了偏差和偏差的偏差进行二阶模糊，最后调出的P项模糊表参考下表：

```
#define P_NB 6.4
#define P_NM 5.9
#define P_NS 5.2
#define P_ZO 4.0
#define P_PS 5.2
#define P_PM 5.9
#define P_PB 6.4

float ruleKp[7][7] = {{P_PB,P_PB,P_PM,P_PM,P_PS,P_ZO,P_ZO},
                      {P_PB,P_PB,P_PM,P_PS,P_PS,P_ZO,P_NS},
                      {P_PM,P_PM,P_PM,P_PS,P_ZO,P_NS,P_NS},
                      {P_PM,P_PM,P_PS,P_ZO,P_NS,P_NM,P_NM},
                      {P_PS,P_PS,P_ZO,P_NS,P_NS,P_NM,P_NM},
                      {P_PS,P_ZO,P_NS,P_NM,P_NM,P_NM,P_NB},
                      {P_ZO,P_ZO,P_NM,P_NM,P_NM,P_NB,P_NB}};
```

4.8 电机转向差速控制策略

4.8.1 不同元素的速度控制策略

通过对智能车的调试我们发现，在直道使用较高的转速，在弯道使用稍慢一些的转速能够很流畅的跑完整个赛道，其中在直道（长直道、短直道）的加速能够非常关键的决定车模的整体速度，而弯道的速度能够提升车模的稳定性。

针对直道入弯的过程我们采取了特定的减速策略，即先根据车模当前速度给出减速场数，当减速场数较高的时候设定速度较低，当减速场数较少的时候设定速度较高，并随着减速场数的减少设定速度逐渐接近弯道速度使车模能够流畅的进入弯道，避免出现“猛加猛减”的情况。

4.8.2 I 型车模的差速策略

I 型车模有一个电机驱动，这电机可以通过自动差速来辅助车模的转向。

4.8.3 I 型车模的电机 PI 控制

I 型车模电机使用增量式 PID 控制,其中 I 项能够加快电机响应,P 项能够减少稳态误差,控制公式如下:

$$\text{MotorL_Out} += \text{MotorL_Ki} * \text{MotorL_Error}[2]; \text{MotorL_Out} += \text{MotorL_Kp} * (\text{MotorL_Error}[2] - \text{MotorL_Error}[1]);$$

其中 MotorL_Out 作为加在电机输出占空比上的增量,针对不同元素,我们给出不同的 PI,规律大概是“直道 PI 大,弯道 PI 小”,能够使车模长直道加速更快,弯道更顺畅。

第五章 识别部分代码软件设计

在本次智能车竞赛中，采用了百度的百度大脑搭载神经网络模型进行识别分类。

5.1控制思想：

在比赛中，识别与赛道寻迹任务分不开，因此，主要以 S320 摄像头寻迹判断元素，遇到元素后，EdgeBoard 寻找红色标识并用相应分类模型对图像内容分类，然后向 Mcu 以串口形式发送相应数据。Mcu 接受 EdgeBoard 识别判断所得结果并做出相应动作。

5.1.1 模型核心思想

在图像分类方面，其本质就是对获取的图像进行多分类，我们采用了基于 PaddlePaddle 搭建的卷积神经网络模型，采用了 `ssd_mobilenet_v1` 模型分类。在合适曝光率下，其准确率可达95.49%。

分类模型如下：

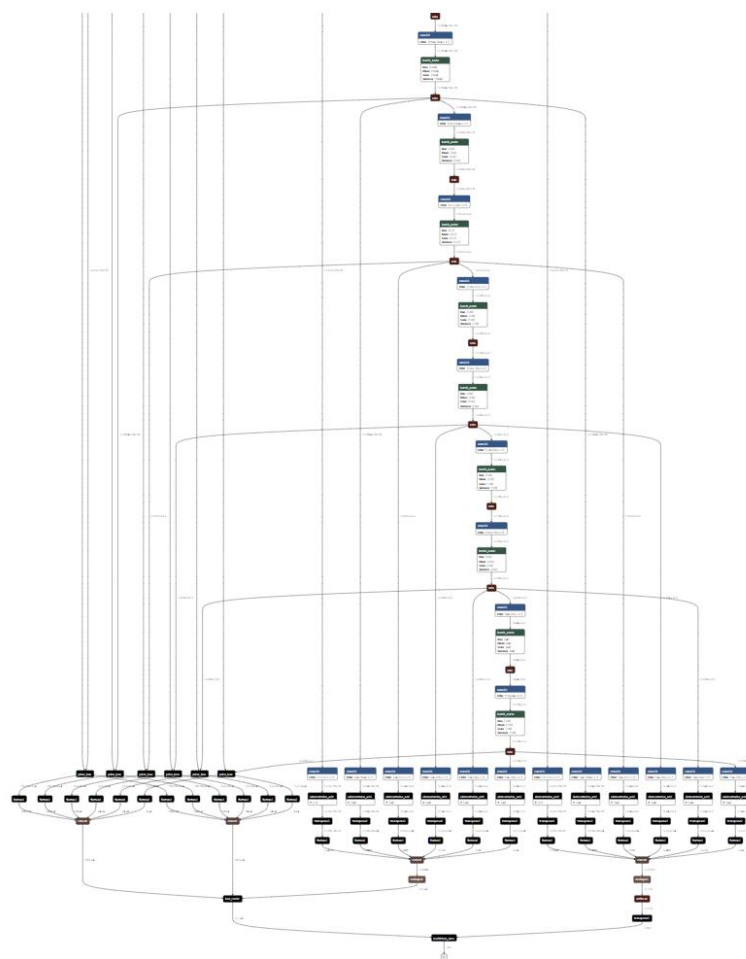


图5.2 分类模型

第六章 总结

在制作智能车期间,我们遇到过很多问题,从最初的传感器选型与方案确定,到后来的软硬件联合调试。在解决一个个问题之后,我们发现,我们技术上在不断成长,思想上在不断成熟。而在这过程中,离不开我们自己的努力与不放弃,更离不开学校,老师和同学的支持。在每次我们发现问题解决后,都能够及时总结自己的问题,这使得我们进步更明显,并且能够在赛道的跑的越来越快。在此次比赛我们组成员都进步了很多。

参考文献

- 【1】孙同景，陈桂友 Freescale 9S12 十六位单片机原理及嵌入式开发技术，北京-机械工业出版社，2008
- 【2】邵贝贝. 单片机嵌入式应用的在线开发方法. 北京-清华大学出版社，2004 年10月第1版
- 【3】卓晴，黄开胜，邵贝贝 学做智能车 北京-北京航空航天大学出版社，2007
- 【4】王威 HCS12 微控制器原理及应用 北京-北京航空航天大学出版社，2007
- 【5】李宁，刘启新 电机自动控制系统 北京-机械工业出版社，2003
- 【6】潘松，黄继业 现代数字电路基础教程 北京-科学出版社 ，2008
- 【7】魏彪，盛新志 激光原理及应用 重庆-重庆大学出版社，2007
- 【8】赵先奎 汽车前轮定位 黑龙江 黑龙江出入境检验检疫局，2008
- 【9】姚领田 精通 MFC 程序设计 人民邮电出版社，2006

附录：

重要部分源代码：

```
/*
*函数：初始化函数
*作者：LXJSWD
*日期：2021/6/12
*版本：第一版
*备注：C3,C27
*/
void init(void)
{
    systick_delay_ms(300);                                //
    延时
    /*=====初始化TFT屏幕=====*/
    lcd_init();

    lcd_clear(GREEN);
    lcd_showstr(0,0,"Initializing...");                    //如果屏幕没有任何显示，
    请检查屏幕接线
    /*=====初始化蜂鸣器引脚=====*/
    gpio_init(BEEP_PIN,GPO,0,GPIO_PIN_CONFIG);            //

    /*=====拨码开关和按键初始化=====*/
    gpio_init(SW1,GPI,0,GPIO_PIN_CONFIG);
    gpio_init(SW2,GPI,0,GPIO_PIN_CONFIG);
    gpio_init(SW3,GPI,0,GPIO_PIN_CONFIG);
    gpio_init(SW4,GPI,0,GPIO_PIN_CONFIG);
    gpio_init(SW5,GPI,0,GPIO_PIN_CONFIG);
    gpio_init(SW6,GPI,0,GPIO_PIN_CONFIG);
    gpio_init(SW7,GPI,0,GPIO_PIN_CONFIG);
    gpio_init(SW8,GPI,0,GPIO_PIN_CONFIG);

    gpio_init(K1,GPI,0,GPIO_PIN_CONFIG);                    //按键
```

```

    gpio_init(K2,GPI,0,GPIO_PIN_CONFIG);
    gpio_init(K3,GPI,0,GPIO_PIN_CONFIG);
    /*=====电磁=====*/
    adc_init(ADC_1,ADC1_CH5_B16,ADC_8BIT);//
    adc_init(ADC_1,ADC1_CH6_B17,ADC_8BIT);
    adc_init(ADC_1,ADC1_CH7_B18,ADC_8BIT);//
    adc_init(ADC_1,ADC1_CH8_B19,ADC_8BIT);
    adc_init(ADC_1,ADC1_CH10_B21,ADC_8BIT);
    adc_init(ADC_1,ADC1_CH12_B23,ADC_8BIT);
    /*=====舵机=====*/
    pwm_init(S_MOTOR_PIN,50,807); //初始化转向舵机
//    pwm_init(PWM4_MODULE3_CHA_C31,50,807); //初始化云台舵机
    /*=====编码器=====*/
    qtimer_quad_init(QTIMER_1,ENCODER1_A,ENCODER1_B); //编码器zuo

qtimer_quad_init(QTIMER_1,ENCODER2_A,ENCODER2_B); //you
    /*=====电机=====*/
    pwm_init(MOTOR1_A,15000,0);
    pwm_init(MOTOR1_B,15000,0);
    pwm_init(MOTOR2_A,15000,0);
    pwm_init(MOTOR2_B,15000,0);
    /*=====摄像头=====*/
    mt9v03x_csi_init(); //初始化摄像头 使用CSI接口
    //如果屏幕一直显示初始化信息，请检查摄像头接线
    //如果使用主板，一直卡在while(!uart_receive_flag)，请检查是否
    电池连接OK?
    //如果图像只采集一次，请检查场信号(VSY)是否连接OK?
    /*=====无线透传模块初始化=====*/
//    seekfree_wireless_init();
    /*=====串口通信初始化=====*/
    uart_init (USART_1,
WIRELESS_UART_BAUD,WIRELESS_UART_TX,WIRELESS_UART_RX); //初始换串口
    NVIC_SetPriority(LPUART1_IRQn,0); //设置串口中断优先级 范
    围0-15 越小优先级越高
    uart_rx_irq(USART_1,1);
    //配置串口接收的缓冲区及缓冲区长度
    extern uint8 example_rx_buffer;
    example_receivexfer.dataSize = 1;
    example_receivexfer.data = &example_rx_buffer;
    uart_set_handle(USART_1, &example_g_lpuartHandle, example_uart_callback,
    NULL, 0, example_receivexfer.data, 1);
    lcd_showstr(0,0," OK... ");
    /*=====中断=====*/
    pit_init(); //初始化pit外设

```

```

        pit_interrupt_ms(PIT_CH0,1);           //初始化pit通道0 周期
        NVIC_SetPriority(PIT_IRQn,15); //设置中断优先级 范围0-15 越小优先
级越高
        systick_delay_ms(500);
    }

/*
*函数： 主动差速增量式PID
*作者： LXJSWD
*时间： 2021/5/7
*参数： target, 目标速度, dir, 0为左轮, 1为右轮
*功能： 单电机控制
*
*/

void INC_PID(void)
{
    /*====左===*/
        l_err_l = err_l;
        err_l = target_L - car_speed_L;
        Dspeed_L += INC_KP * (err_l - l_err_l) + INC_KI *
if(Dspeed_L > 5000)
            Dspeed_L = 5000;
        if(Dspeed_L < -8000)
            Dspeed_L = -8000;
    /*====右===*/
        l_err_r = err_r;
        err_r = target_R - car_speed_R;
        Dspeed_R += INC_KP * (err_r - l_err_r) + INC_KI * err_r;
        if(Dspeed_R > 5000)
            Dspeed_R = 5000;
        if(Dspeed_R < -8000)
            Dspeed_R = -8000;
    }
    /*=====
    =====*/

/*
*函数： 速度采集
*作者： LXJSWD
*日期： 2021/6/16
*备注：

```

```

*/
/*****/

void GetSpeed (void)
{
    static int index = 0;
    L_car_speed = car_speed;
    L_RealSpeed_R = RealSpeed_R;
    L_RealSpeed_L = RealSpeed_L;
    RealSpeed_R = qtimer_quad_get(QTIMER_1,QTIMER1_TIMER0_C0);
    qtimer_quad_clear(QTIMER_1,QTIMER1_TIMER0_C0 );
    RealSpeed_L=qtimer_quad_get(QTIMER_1,QTIMER1_TIMER2_C2);
    qtimer_quad_clear(QTIMER_1,QTIMER1_TIMER2_C2);

    RealSpeed_R = (int) (0.7 * RealSpeed_R + 0.3 * L_RealSpeed_R);
    RealSpeed_L = (int) (0.7 * RealSpeed_L + 0.3 * L_RealSpeed_L);

    Encoder_Speed_Filter_L[index++] = RealSpeed_L;
    Encoder_Speed_Filter_R[index] = RealSpeed_R;
    index %= 5;

    car_encoder_speed_L = 0;
    car_encoder_speed_R = 0;

    for (int i = 0; i < 5; i++)
    {
        //左
        car_encoder_speed_L += Encoder_Speed_Filter_L[i];
        if (Encoder_Speed_L_Max < Encoder_Speed_Filter_L[i])
            Encoder_Speed_L_Max = Encoder_Speed_Filter_L[i];

        if (Encoder_Speed_L_Min > Encoder_Speed_Filter_L[i])
            Encoder_Speed_L_Min = Encoder_Speed_Filter_L[i];
        //右
        car_encoder_speed_R += Encoder_Speed_Filter_R[i];
        if (Encoder_Speed_R_Max < Encoder_Speed_Filter_R[i])
            Encoder_Speed_R_Max = Encoder_Speed_Filter_R[i];

        if (Encoder_Speed_R_Min > Encoder_Speed_Filter_R[i])
            Encoder_Speed_R_Min = Encoder_Speed_Filter_R[i];
    }

    car_encoder_speed_L = (car_encoder_speed_L - Encoder_Speed_L_Max -
Encoder_Speed_L_Min) / 3.0;

```

```
car_encoder_speed_R = (car_encoder_speed_R - Encoder_Speed_R_Max - Encoder_Speed_R_Min) / 3.0;
```

```
car_speed_L = car_encoder_speed_L * 0.035 * 100;  
car_speed_R = car_encoder_speed_R * 0.035 * 100;
```

```
car_speed_L = 0.8 * car_speed_L + 0.2 * L_car_speed_L;  
car_speed_R = 0.8 * car_speed_R + 0.2 * L_car_speed_R;  
car_speed = (car_speed_L + car_speed_R) / 2;
```

```
}
```

```
/*
```

```
*作者: LXJSWD
```

```
*时间: 2021/6/5
```

```
*功能: 位置式PID,
```

```
*待优化地方: */
```

```
void dirction(void)
```

```
{
```

```
    dir_l_err = dir_err; //方向偏差
```

```
    dir_err = img_W / 2 -
```

```
(mid_line[80] + mid_line[81] + 2 * mid_line[82] + 2 * mid_line[83] + 3 * mid_line[84] + 3 * mid_line[85] +
```

```
4 * mid_line[86] + 4 * mid_line[87] + 5 * mid_line[88] + 5 * mid_line[89]) / 30;
```

```
if(R_flag > 2)
```

```
    PID_out = 1 * dir_err + 0 * (dir_err - dir_l_err);
```

```
// else if( abs(dir_err) > 10) //弯道
```

```
//     PID_out = dir_P * dir_err + dir_D * (dir_err - dir_l_err);
```

```
// else
```

```
    PID_out = dir_P * dir_err + dir_D * (dir_err - dir_l_err);
```

```
if(abs(PID_out) < 5 || jun_flag == 1)
```

```
    PID_out = 0;
```

```
PID_OUT = steer_mid + PID_out;
```

```
if(PID_OUT > steer_left)
```

```
    PID_OUT = steer_left;
```

```
if(PID_OUT < steer_right)
```

```
    PID_OUT = steer_right;
```

```
pwm_duty(S_MOTOR_PIN, PID_OUT);
```

```
}
```

```

/*****
*****

* @file          main
* @author        LXJSWD
* @date          2021-06-10
* @备注： 按键0为发车键，按键1为-，按键2为+
          拨码开关：3为PID四个参数（增量式PI,位置式
          PD）总开关；2 1开关：00为IP 01为li 10为Dd 11为DP
          4为速度调节：
          5为显示页数调节：
          6为是否识别数字等图像

* @版本说明：
*****
*****/

int main(void)
{
    DisableGlobalIRQ();
    board_init();
    init();                                EnableGlobalIRQ(0);

    systick_start();                       while (1)
    {
        debug_key();
        if(mt9v03x_csi_finish_flag)
        {
            SysTick->VAL = 0x00;
            start_time = systick_getval_ms();
            mt9v03x_csi_finish_flag = 0;
//            AD_collect();
            get_image();                    img_filter();

            get_line();

//            chuku();
//            ruku();

            roundabout();

            road_junction();
            buxian();

            dirction();

            differential();

```

```

//          out_range();
            display();

        }

    }

}

/*=====
=====

```

Edgeboard模型预测 mobilenetV1-ssd

```
g_predictor = PaddleLitePredictor()
```

```

class PredictResult(object):
    """result wrapper"""
    def __init__(self, category, score, x, y, width, height):
        """init"""
        self.type = int(category)
        self.score = score
        self.x = int(x)
        self.y = int(y)
        self.width = int(width)
        self.height = int(height)

def predictorInit():
    """predictor config and initialization"""
    global g_model_config
    global g_system_config

    global g_predictor
    if g_model_config.is_combined_model:
        g_predictor.set_model_file(g_model_config.model_file)
        g_predictor.set_param_file(g_model_config.params_file)
    else:
        # TODO add not combined model load
        pass
    try:
        g_predictor.load()
        print("Predictor Init Success !!!")
        return 0
    except:
        print("Error: CreatePaddlePredictor Failed.")
        return -1

```

#预测初始化

IS_FIRST_RUN = True

```
def predict(frame, timer):                                #得到预测结果
    """predict with paddlelite and postprocess"""
    origin_frame = frame.copy()
    origin_h, origin_w, _ = origin_frame.shape
    if not g_system_config.use_fpga_preprocess:

        input_data = cpu_preprocess(frame, g_model_config)
        g_predictor.set_input(input_data, 0)
    else:
        input_tensor = g_predictor.get_input(0)
        fpga_preprocess(frame, input_tensor, g_model_config)

    if g_model_config.is_yolo:
        feed_shape = np.zeros((1, 2), dtype=np.int32)
        feed_shape[0, 0] = origin_h
        feed_shape[0, 1] = origin_w

        shape_tensor = g_predictor.set_input(feed_shape, 1)

    global IS_FIRST_RUN
    if IS_FIRST_RUN:
        IS_FIRST_RUN = False
        g_predictor.run()
    else:
        timer.Continue()
        g_predictor.run()
        timer.Pause()
    outputs = np.array(g_predictor.get_output(0))

    res = list()
    if outputs.shape[1] == 6:
        for data in outputs:
            score = data[1]
            type_ = data[0]
            if score < g_model_config.threshold:
                continue
            if g_model_config.is_yolo:
                data[4] = data[4] - data[2]
                data[5] = data[5] - data[3]
                res.append(PredictResult(*data))
            else:
```

```

        h, w, _ = origin_frame.shape
        x = data[2] * w
        y = data[3] * h
        width = data[4] * w - x
        height = data[5] * h - y
        res.append(PredictResult(type_, score, x, y, width, height))

    return res

def printResults(frame, predict_result):                                #打印结果
    """print result"""
    for box_item in predict_result:
        if len(g_model_config.labels) > 0:
            print("label: {}".format(g_model_config.labels[box_item.type]))
            str_ = "index: {}".format(box_item.type) + ", score:
{}".format(box_item.score) \
                + ", loc: {}".format(box_item.x) + ", {}".format(box_item.y) + ",
{}".format(box_item.width) \
                + ", {}".format(box_item.height)
            print(str_)

def boundaryCorrection(predict_result, width_range, height_range):
    """clip bbox"""
    MARGIN_PIXELS = 2
    predict_result.width = width_range - predict_result.x - MARGIN_PIXELS \
        if predict_result.width > (width_range -
predict_result.x - MARGIN_PIXELS) \
        else predict_result.width

    predict_result.height = height_range - predict_result.y - MARGIN_PIXELS \
        if predict_result.height > (height_range - predict_result.y -
MARGIN_PIXELS) \
        else predict_result.height

    predict_result.x = MARGIN_PIXELS if predict_result.x < MARGIN_PIXELS else
predict_result.x
    predict_result.y = MARGIN_PIXELS if predict_result.y < MARGIN_PIXELS else
predict_result.y
    return predict_result

def drawResults(frame, results):                                        #绘制结果图像
    """draw result"""
    frame_shape = frame.shape
    for r in results:
        r = boundaryCorrection(r, frame_shape[1], frame_shape[0])
        if r.type >= 0 and r.type < len(g_model_config.labels):

```

```
origin = (r.x, r.y)
label_name = g_model_config.labels[r.type]
cv2.putText(frame, label_name, origin, cv2.FONT_HERSHEY_PLAIN, 2,
(0, 0, 224), 2)
cv2.rectangle(frame, (r.x, r.y), (r.x + r.width, r.y + r.height), (0, 0,
224), 2)
```

