**基于树莓派的智能追人小车**

**随着半导体集成电路和人工智能的持续发展，智能化设备也朝着小型化，高能效化持续推进。小型消费电子产品也开始搭载人工智能来处理更加通用更加复杂的任务需求了。**

**第一章 绪论**

**1.1 课题背景及研究意义**

**1.1.1 课题背景**

**随着半导体集成电路和人工智能的发展，智能小车已经成为了我们生活中的一部分。树莓派，作为一款功能强大的微型计算机，被广泛应用于各种项目中，而智能小车就是树莓派应用中非常有前景的项目。**

**1.1.2 研究意义**

**然而，智能小车的硬件和软件设计仍然存在一些问题，需要进一步完善。因此，本文将以树莓派作为开发平台制作一款能够自动跟随人移动的智能小车。**

**1.2 国内外研究现状**

**目前，国内外对于智能跟人小车的研究主要集中在以下几个方面：**

**1.2.1 欧洲开发基金资助进行驾驶员监测、道路环境的感知、视觉增强、前车距离控制以及传感器融合方面的研究[1]。**

**1.2.2 基于树莓派的智能小车的设计与开发，通过车载摄像头采集图像，利用OpenCV技术对图像进行处理，实现循迹功能；通过建立深度学习模型，对红绿灯数据集进行训练，利用训练好的模型实现红绿灯识别功能；通过红外避障模块实现道路避障功能。[2]。**

**1.2.3 基于树莓派与YOLOv5-Lite模型的行人检测系统设计，把在PC端上训练好的YOLOv5s与YOLOv5-Lite目标检测模型分别部署在搭载Linux系统的树莓派4B平台上，并对二者进行对比[3]。**

**1.3 现阶段发展中的问题**

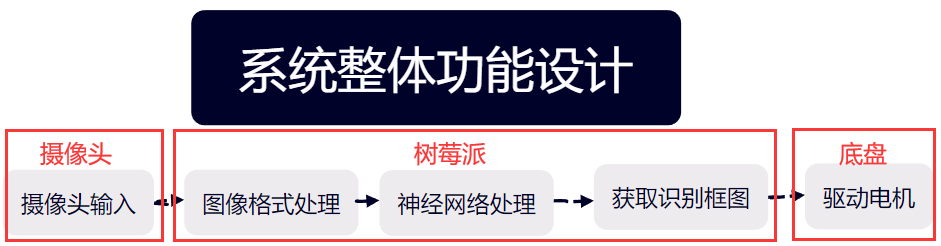
**尽管树莓派智能跟人小车已经可以初步开始完成跟人任务。但是任然存在非常严重的问题需要解决，作为低功耗计算平台，树莓派的功耗是非常令人满意的，但是在这低功耗的之下，代价确实性能在人工智能上的严重缺乏。在ONNX 模型下直接跑 YOLOv5-Lite 的网络模型，该状态下的FPS仅维持在5左右，效果比直接跑 YOLOv5 网络模型已经好很多了（YOLOv5的FPS在0.3FPS左右）[3]。或者使用 Intel神经棒进行加速，则需要安装 mvnc。最终是通过摄像头来跑深度学习从而识别人体，根据识别到的方框来判断人具体的位置，从而控制小车。[4]。为了解决性能问题，此前的研究主要集中在使用外置的神经网络加速设备，云计算运行或者更换性能更高的计算平台**

**1.4 主要研究内容**

**为了保证设备的数据隐私性，断网可靠性，和低成本性。因此该设备的人体识别必须部署在本地运行，并且尽量不使用外部神经网络加速设备。所以本文的主要研究内容就是在这种条件下保证性能的最大释放与效果的准确性保证。**

**第二章 方案设计**

**2.1 系统设计框架**



**图2.1 系统功能框架**

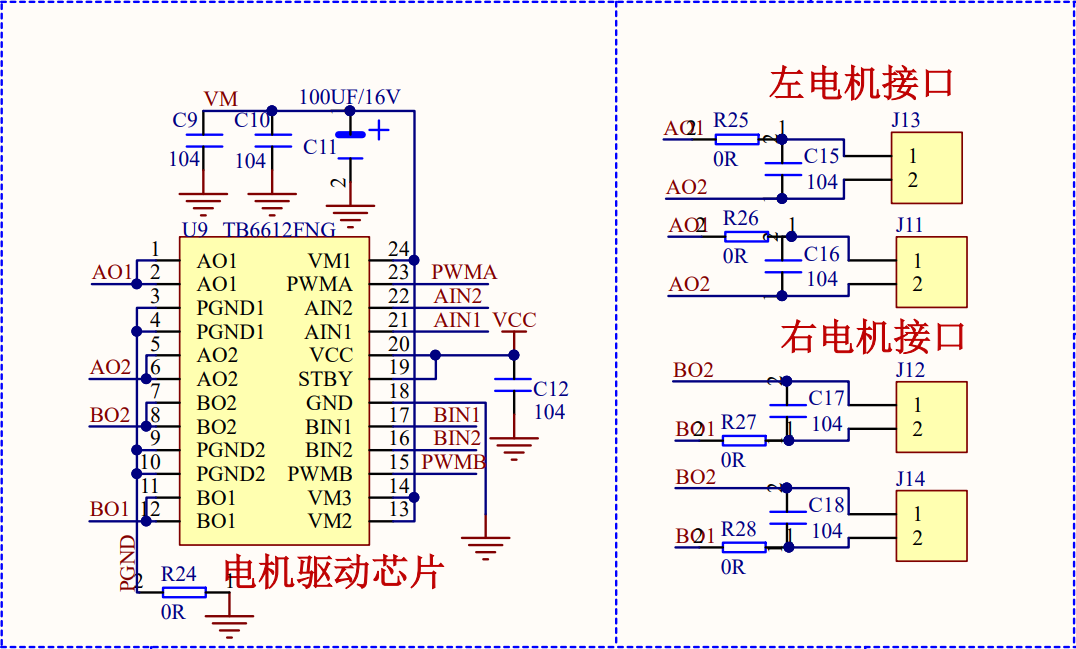
**摄像头:获取图像信息，发送图像信息至树莓派。**

**树莓派:读取摄像头图像信息，处理图像格式，运行神经网络，向底盘发送控制信号。**

**底盘:驱动电机与给树莓派供电。**

**第三章 硬件设计**

**在本次项目中，我采用了亚博智能的树莓派4WD小车作为基本的开发平台。采取这个设备的原因是亚博智能的开发文档全面且详细，并且拥有大量常用的硬件模块以及支持多种主控芯片。这将会使我们的开发过程更加便利和灵活。**



**图3.1电机驱动**

**3.1 电机驱动**

**如图3.1该车采用TB6612FNG直流电机驱动芯片。TB6612FNG 是东芝半导体公司生产的一款直流电机驱动器件，它具有大电流MOSFET-H 桥结构，双通道电路输出，可同时驱动2 个电机，它无需外加散热片，外围电路简单，只需外接电源滤波电容就可以直接驱动电机，利于减小系统尺寸。对于PWM 信号输入频率范围，高达100 kHz 的频率更是足以满足大部分的需求了。[5]。该车各向前轮与后轮使用同一个驱动接口，在节省了io的同时带来了无法单独调节前轮与后轮速度的缺陷。**

**3.2 树莓派4b介绍**

**嵌入式系统的核心部分是中央控制器，它负责决策和控制。在我们的系统中，我们选择了树莓派4B作为嵌入式开发板，它搭载了Linux操作系统。树莓派被誉为卡片式电脑，因为它的体积小巧，功能强大。它可以安装操作系统，连接显示器和输入设备，支持Python和C语言编程。树莓派有着丰富的社区资源，这对于学习和使用都非常有帮助。市场上还有许多树莓派配件，这使得它在开发中具有优势。相比于树莓派3代，树莓派4B在处理器的CPU方面进行了升级，从64位Crotex-A53提升至64位的Crotex-A72，主频也从1200MHz提升至1500MHz。最大内存从1GB提升至4GB，电源线路也被提升至3A@5V，以满足更高的电源需求。[6]**

**3.3 摄像头介绍**

**该车的摄像头组件是30w像素的USB高清摄像头，通过USB2.0与树莓派连接,经过代码测试，摄像头的帧数约为30HZ，较小的像素在树莓派这种低功耗的计算平台上面能做到更低的性能消耗和更加实时的图像数据获取。**

**第四章 软件设计**

**4.1 项目开发环境介绍**

**4.1.1 OpenCv介绍**

**OpenCV（开源计算机视觉库）是在 BSD 许可下发布的，因此它可以免费用于学术和商业用途。它具有 C++，Python 和 Java 接口，支持 Windows，Linux，Mac OS，iOS 和 Android。OpenCV 专为提高计算效率而设计，专注于实时应用。该库以优化的 C/C++ 编写，可以利用多核处理。通过 OpenCL 启用，它可以利用底层异构计算平台的硬件加速。用途范围从交互式艺术，到地雷检查，网上拼接地图或高级机器人。[7]**

**4.1.2NCNN介绍**

**ncnn 是一个为手机端极致优化的高性能神经网络前向计算框架。 ncnn 从设计之初深刻考虑手机端的部署和使用。 无第三方依赖，跨平台，手机端 cpu 的速度快于目前所有已知的开源框架。 基于 ncnn，开发者能够将深度学习算法轻松移植到手机端高效执行， 开发出人工智能 APP，将 AI 带到你的指尖。 ncnn 目前已在腾讯多款应用中使用，如：QQ，Qzone，微信，天天 P 图等。[8]**

**4.1.2.1 NCNN特征**

**支持卷积神经网络，支持多输入多分支结构，可计算部分分支。无第三方库依赖，不依赖BLAS/NNPACK或任何其他计算框架。纯C++实现，跨平台，支持Android、iOS等。ARM NEON组装水平精心优化，计算速度极高。先进的内存管理和数据结构设计，内存占用极低。支持多核并行计算加速，ARM big.LITTLE CPU 调度优化。通过下一代低开销 Vulkan API 支持 GPU 加速。可扩展模型设计，支持8bit量化和半精度浮点存储，可导入caffe/pytorch/mxnet/onnx/darknet/keras/tensorflow（mlir）模型。支持直接内存零拷贝参考加载网络模型。可以注册到自定义层实现和扩展。[8]**

**4.1.3 Jupyter 介绍**

**Jupyter 是一个涵盖许多不同软件产品和工具的项目，包括 流行的 Jupyter Notebook 和 JupyterLab 基于 Web 笔记本创作和编辑应用程序。Jupyter 项目及其 子项目都以提供工具（和标准）为中心 用于使用计算笔记本进行交互式计算。[9]**

**Jupyter Notebook是结合了计算机代码、通俗易懂语言的可共享文档 描述、数据、丰富的可视化效果，如 3D 模型、图表、图形和 图形和交互式控件。一个笔记本，以及一个编辑器（如 JupyterLab），为原型设计和 解释代码、探索和可视化数据。[9]**

**JupyterLab 是最新的基于 Web 的交互式开发环境，适用于笔记本、代码和数据。其灵活的界面允许用户配置和安排数据科学、科学计算、计算新闻和机器学习中的工作流程。模块化设计邀请扩展以扩展和丰富功能。[9]**

**4.2 项目开发流程**

**由于本项目是基于亚博智能的树莓派4wd小车进行开发，所以在项目的开始，是在原本亚博智能提供的官方例程即树莓派人脸检测上进行的二次开发，在完成了基本的人脸跟随功能和解决例程里面不合理的代码逻辑后，我们从python代码转向c++代码，从harr分类器转向yolo[10]检测模型，完成真正意义上的人体追踪。**

**4.2.2 树莓派智能追脸小车**

**为了验证树莓派追人小车项目的可行性，我们先是对亚博智能的人脸识别官方例程进行二次开发，以求达到小车跟随人脸转动的功能，并分析将代码功能移植为人体追踪小车的可能性。**

**在分析亚博智能的人脸识别代码控制时，发现代码对于舵机的控制是有问题的，例程给的代码，每一次对图像标记方框后，对舵机的操作是向最大角度或者最小角度转动，并没有进行舵机角度的限定，这意味着，每一帧图片的分析结果都会让舵机转向0°或者180°，若是设备性能足够，这将不会有任何问题，但在树莓派4b上，有时性能不够延迟加大时，摄像头会直接摆向最值从而失去对人脸的观测。同时，舵机代码是使用阻塞式的软件模拟pwm，并且亚博智能并没有对此有针对性的优化因此，每一帧图像识别后将会有将近40ms的时间在空等待，soc的使用率不高。**

**因此，本阶段的主要任务就是重构舵机代码，构造底盘转向的电机代码，构造多线程处理代码，优化代码构造等。**

**为了让舵机运动时能适应偶发的高延迟的情况，新的舵机代码将采用逐步积分式的舵机角度，使用一个变量来存储舵机当前角度并输出为对应的pwm波，每一帧结果经过pid调节后转变为当前帧的应转角度，并积分为下一帧的起始角度**

**为了适应并测试下一阶段的人体追踪，新代码在左右转向上抛弃了原先的舵机转车体不转的功能，改变为车体左右转，舵机上下转的形式，但是要实现追人，harr分类器只能识别人脸无法识别人体，无法适应小车追人的使用情景，因此下一阶段的人体目标检测将是主要任务。而当前阶段的转向电机代码与舵机代码类似，是使用软件模拟pwm来实现小车两侧电机控制。**

**在完成舵机代码的重构和底盘电机代码的编写后运行测试通过后，处理阻塞式软件模拟pwm带来的延迟问题成了重中之重。如此解决软件模拟pwm的阻塞式延迟有两个解决方案，一是将软件模拟pwm转换为硬件pwm生成器输出pwm信号。但是树莓派4b仅有一个硬件pwm生成器，无法满足项目的需要。或采用soc加mcu综合控制，将树莓派识别的框图坐标通过串口传送给下位机mcu例如stm32等，但是这种方案要求对硬件进行重新设计以同时兼容树莓派与stm32的同时使用，并且该方案还会增加项目的成本。二是针对现有代码进行优化，将软件模拟pwm时产生的阻塞时间利用起来，因此多线程优化成为了唯一选择。而如何针对多线程进行优化便成为了这一阶段的最重要的任务之一。**

**本项目的多线程工作逻辑为将舵机代码和电机代码封装在一个函数中，并且创建一个线程专门运行这个线程，同时创建一个变量用于标记当前舵机电机控制变量的来源帧，图像每一次识别都会改变该变量的值，而运行电机舵机代码时将会重置当前值，如果该标记为重置值，线程将会阻塞至图像识别完毕并将标记置位后再运行。该框架能够保证图像数据的读取与识别无阻塞的一直运行，确保了图像识别时的实时性。而线程的阻塞式状态检测保证了舵机与电机运行的代码不会多次针对同一帧图像，减少了延迟带来的滞后性问题。**

**4.2.1.1 harr分类器介绍**

**以Haar分类器为基础的检测方法是一种非常有效的对象检测技术，于2001年由Paul Viola和Michael Jones提出. Haar分类器实际上是Boosting算法的一个应用，Haar分类器用到了Boosting算法典型代表中的Adaboost算法，只是把Adaboost算法训练出的强分类器进行了级联，并且在底层的特征提取采用了高效率的矩形特征和积分图方法。boosting算法的核心思想就是串联若干个弱分类器。[11]**

**4.2.1.2人脸识别**

**人脸识别的基本逻辑为：初始化时加载harr分类器模型，在代码初始化完成之后，摄像头开始在主循环内获取一帧画面，将画面转向jrpg格式后输入到harr分类器中，计算出人脸的位置后，再用opencv在原图像帧中画出对应的方框，并将方框的两个x和y轴坐标输出到下一层**

**4.2.1.3 电机驱动**

**电机驱动的代码基本逻辑为：初始化时设置好舵机和电机的引脚模式和pwm频率。初始化完成后，在单独开辟出来的线程中接收人脸识别输出的两个x和y轴，并解算出当前人脸框图在图像帧中的相对位置与中心位置进行比较，将差值输入PID中算出最终的pwm波应有占空比，产生对应的pwm波输出为电机的转速。**

**4.2.1.4 线程实现**

**由于本项目所使用的电机与舵机驱动都是软件模拟的pwm波发生器，其中使用的是阻塞式pwm生成，在产生pwm波时，需要硬性地等待，这极大的浪费了cpu的使用率和增加了程序对每一帧图像处理的延时。因此，将pwm波的阻塞式延迟给消除，成为了优化程序的一个重点。所以，我将电机和舵机的驱动代码与运动代码重新封装成为一个函数，并开启一个线程对这个函数进行调用，同时为了同步代码，防止重复对同一帧图像进行运动，我用全局变量对改线程增加了一个线程锁，只有当harr分类器运算完一帧图像后，运动线程才能进行运动。**

**4.2.1.5 全部代码逻辑**

**这部分代码主要包括以下几个部分：**

**图像处理：使用OpenCV库进行图像处理，包括读取摄像头图像，将图像转换为灰度图像，然后使用Haar级联分类器进行人脸检测。**

**舵机控制：使用RPi.GPIO库控制舵机，根据人脸在图像中的位置调整舵机的角度，使摄像头能够跟踪人脸的移动。**

**PID控制：使用PID算法对舵机的运动进行精确控制，确保舵机能够平滑、准确地跟踪人脸的移动。**

**多线程：使用Python的threading库创建新的线程，使得人脸检测和舵机控制可以同时进行，提高了系统的实时性和响应速度。**

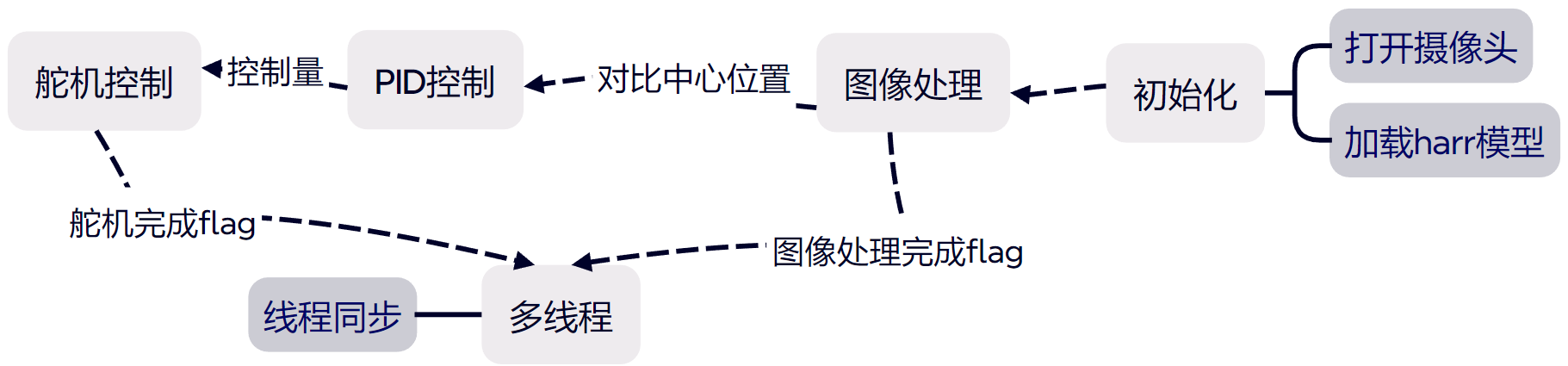
**具体来说，代码的执行流程如下：**

**导入所需的库，包括cv2、RPi.GPIO、threading等。并定义一些函数，包括将BGR格式的图像转换为JPEG格式的函数（bgr8\_to\_jpeg）、控制电机运动的函数（motor\_control）、产生PWM信号的函数（servo\_pulse）、控制舵机角度的函数（Servo\_control）以及人脸跟踪的函数（face\_follow）。**

**初始化GPIO口，设置为BCM编码方式，定义舵机引脚，设置为输出模式。分别创建图像显示窗口，设置窗口的大小。滑动条，用于调节PID控制器的参数。PID控制器实例，用于控制舵机的运动。**

**载入Haar级联分类器，用于人脸检测。开启摄像头，开始循环读取摄像头图像。在每一帧图像中，先将图像转换为灰度图像，然后使用Haar级联分类器进行人脸检测。如果检测到人脸，就计算人脸的中心位置，然后使用PID控制器计算出舵机应该转动的角度，最后控制舵机转动，使摄像头对准人脸。**

**使用多线程进行人脸跟踪，使得人脸检测和舵机控制可以同时进行，提高了系统的实时性和响应速度。**



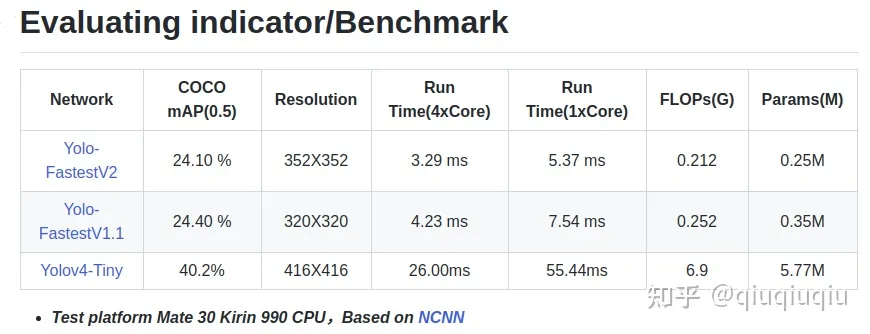
**4.2.2 树莓派智能追人小车**

**通过追脸功能的测试，智能追人的任务的首要的问题就是树莓派4b的性能问题。通过追脸任务可知，在无多线程优化的情况下，人脸识别的延迟最多可达到5秒，运行帧数约为7fps，在加上多线程优化后最大延迟降低至3秒左右，运行帧数达到15fps。但是由于使用的haar对人脸识别的图像要求较高，识别准确率不理想，导致实际运行的时候经常跑飞和不跟人。**

**4.2.2.1 yolofastestV2介绍**

**传统的目标检测算法只适应于有明显特征，背景简单的情形，而在实际应用中，背景复杂多变，而且待检测的目标复杂多变，很难通过一般的抽象特征完成对目标的检测，深度学习则可以提取同一目标丰富的特征，来完成目标的检测。但是由于网络结构复杂，参数多，运行效率低等原因，这些模型需要占用较大的内存，且在边缘设备上普遍无法满足实际应用中的实时性要求，因此减少网络的参数是解决这一矛盾的主要技术手段[12]**

**Yolo-FastestV2是由知乎用户马雪浩开发的一种类似于Yolo的超轻量化目标检测模型Yolo-Fastest的第二代。其目的是在低算力的边缘计算平台诸如树莓派，全志RK3399或RK3588等设备上进行高速运行目标检测功能相比传统的Yolo检测模型，Yolo-FastestV2以较低的精度损失换来了非常大的速度增益。**

****

**图4.2.2**

**由图4.2.2可知，在麒麟990上Yolo-FastestV2以16.1%的精度损失换来了22.71ms的速度提升，在单核运行的情况下，速度的提升更加可观。在大幅提升速度的情况下，模型的参数量也从5.77M降低至0.25M。使得运行平台在工作时能承担更多的任务和处理更加复杂的情况。Yolo-Fastest注重的就是单核的实时推理性能，在满足实时的条件下的低CPU占用，不单单只是能在手机移动端达到实时，还要在RK3399，树莓派4以及多种Cortex-A53低成本低功耗设备上满足一定实时性，毕竟这些嵌入式的设备相比与移动端手机要弱很多，但是使用更加广泛，成本更加低廉。[13]**

**4.2.2.2 模型训练**

**首先，准备好yolofastest v2的源代码和待训练的数据集。然后修改训练配置文件：打开训练配置文件，指定数据集路径、学习率等参数。(修改coco.name和coco.data)接下来开始训练模型：运行 train.py，传入训练配置文件路径。训练过程中会计算损失函数并更新模型权重(在终端输入python train.py --data data/coco.data)。接着评估模型效果：使用 evaluation.py 评估模型性能。运行 test.py 进行单张图像的测试。最后将Pytorch模型转换成 ONNX 模型：使用 pytorch2onnx.py 将训练好的模型转换为 ONNX 格式。**

**4.2.2.3 人体识别**

**1.导入库和定义类别：首先，代码导入了所需的库，包括 OpenCV 和 YOLO-FastestV2 模型。然后，定义了一个只包含 “person” 类别的数组 class\_names。**

**2.加载模型：创建了一个名为 yoloFastestv2 的 YOLO-FastestV2 模型实例。使用 loadModel 方法加载了预训练的模型参数和权重文件。**

**3.打开摄像头：使用 cv::VideoCapture 打开了摄像头。如果摄像头无法打开，会输出错误信息并退出。**

**4.循环处理图像：在一个无限循环中，不断从摄像头获取图像。对每个获取到的图像：运行模型进行目标检测，将检测结果存储在 boxes 向量中并在图像上绘制文本的背景框。**

**4.2.2.4电机驱动**

**与追脸任务相比，运动的代码要更为复杂，在原先转向的基础上还要加入前进后退和调速的代码，于是便采用串级pid对电机的转速进行pwm调制，以实现运动与差速的效果。为了简化代码和优化，此次电机的pwm发生器选用wriningPi的gpio库函数，避免了由阻塞式pwm发生带来的延迟问题和提高cpu使用率。**

**4.2.2.5全部代码逻辑**

**使用YOLOfastestv2模型进行实时人脸检测的程序，主要包括以下几个部分：**

**初始化：在main函数中，程序首先调用GPIO\_Init函数来初始化GPIO口和PWM脚，然后加载YOLO-Fastestv2模型，接着打开摄像头并获取图像。**

**目标检测：程序使用YOLO-Fastestv2模型对摄像头的图像进行目标检测，并将检测结果存储在boxes变量中。如果检测到目标，程序会调用chases函数，使机器人追踪目标。**

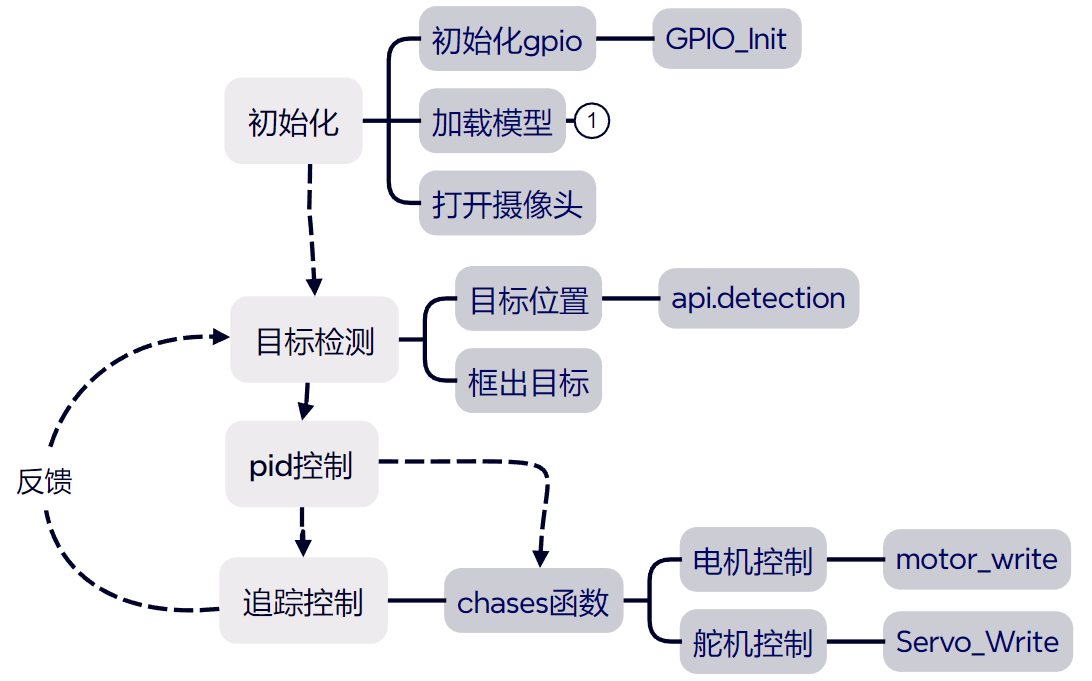
**电机控制：motor\_write函数用于控制电机的速度和方向。它首先根据输入的速度值来确定电机的方向，然后限制电机的速度在一个预设的范围内，最后通过softPwmWrite函数来控制电机的速度。**

**舵机控制：Servo\_Write和Servo\_Duty\_Get函数用于控制舵机的角度。Servo\_Write函数通过pwmWrite函数来发送PWM脉冲，从而控制舵机的角度。Servo\_Duty\_Get函数用于根据输入的y值调整舵机的角度。**

**追踪控制：chases函数用于根据目标的位置信息，计算并调整电机的速度和方向，使机器人能够追踪目标。它首先计算目标的位置与图像中心的偏移量，然后根据偏移量和预设的PID参数来计算电机的速度，最后调用motor\_write函数来控制电机。**

**具体来说，代码的执行流程如下：**

**在主函数main中，程序首先初始化GPIO口，然后加载YOLO-Fastestv2模型，接着打开摄像头并获取图像。然后，程序使用YOLO-Fastestv2模型对图像进行目标检测，并将检测结果显示在窗口中。如果检测到目标，程序会调用chases函数，使机器人追踪目标。**



**第五章 功能测试**

**5.1小车成品**

**5.2人脸追随**

**5.3人体追随**

**第六章 总结**

**6.1已完成任务**

**智能追脸代码：这是一个人脸跟踪系统，使用OpenCV和PID控制算法，通过摄像头捕获图像并识别出人脸，然后通过舵机控制摄像头的方向，使其始终对准人脸。**

**智能追人代码：这是一个使用YOLOfastest v2模型进行实时人脸检测的程序，通过摄像头捕获图像，然后将图像输入到YOLOfastest v2模型中进行人脸检测，最后在图像上标出检测到的人脸，并显示出人脸的类别和得分。**

**6.2 结论与展望**

**这两个任务都是基于计算机视觉的实时人脸检测和跟踪系统，它们都能够有效地识别和跟踪人脸。然而，由于树莓派4b的性能不足，这些系统还有很大的改进空间。例如，我们可以考虑使用更先进的人脸检测算法，或者选择使用云计算平台或者分分布式计算平台。此外，我们也可以考虑增加更多的功能，如表情识别、年龄估计、性别识别等，以使系统更加智能和实用。在硬件方面，我们可以考虑使用更高质量的摄像头使用多摄像头系统或者加上底盘控制器，以提高系统的性能和稳定性。总的来说，这些系统为我们提供了一个很好的起点，我们可以在此基础上进行进一步的研究和开发，以实现更多的应用。**

**参考文献**

**[1] 滕开雯，丁康，王硕. 基于树莓派的智能自动驾驶系统设计与实现. 国家科技图书文献中心, 2023.**

**[2] 韩改宁, 苏静池, 张瑞斌. 基于树莓派的智能小车的设计与开发[J]. 电子设计工程, 2024, 32 (01): 6-10.**

**[3] 郑尚坡, 陈德富, 邱宝象, 张龙. 基于树莓派与YOLOv5-Lite模型的行人检测系统设计[J]. 计算机时代, 2023, (09): 116-119.**

**[4] 梦远花落白衣衫.** **树莓派4B使用深度学习搭建自动跟随小车.CSDN博客,2020.**

**[5]** **何晓凤. 一种双轮自平衡小车的设计[J]. 电子器件, 2023, 46 (02): 391-396.**

**[6]** **尤泳茹, 陈梦凡, 陈俊霞, 杜蓝艺, 朱炫羽, 田丽鸿. 基于树莓派的目标识别检测研究[J]. 现代信息科技, 2023, 7 (21): 94-98.**

**[7] 张代伟. 基于OpenCV的视觉检测系统应用[J]. 机电信息, 2024, (01): 7-10./**

**[8] 曾泰恒. 基于YOLO的轻量化番茄实时检测算法及移动端部署[D].福建农林大学,2023.DOI:10.27018/d.cnki.gfjnu.2023.000274. [9] 蒋佳龙, 林木辉. Jupyter在人工智能课程实验平台的应用实践[J]. 实验室科学, 2022, 25 (03): 75-79.**

**[10]** **REDMON J,FARHADI A.YOLOv3:An Incremental Improvement[J/OL].arXiv:1804.02767[cs.CV].[2023-07-28].https://arxiv.org/ab s/1804.02767.**

**[11] 任亮朴, 徐美华, 陈高攀. Haar特征提取IP核设计及前方车辆检测系统实现[J]. 电子设计工程, 2018, 26 (14): 107-111+116.**

**[12] 李秉涛.** **基于轻量级CNN的实时目标检测研究.** **贵州大学,2023.**

**[13] 马雪浩.** **Yolo-FastestV2:更快，更轻，移动端可达300FPS，参数量仅250k.知乎,2021**