

中国工程机器人大赛暨国际公开赛（RoboWork）

工程创新项目工程创新设计赛

（视觉机器狗识别赛）

技术报告

参赛学校： 西交利物浦大学

队伍名称： Cyberpubby

参赛队员： 徐铭鸿、赖向康、江卓远

带队教师： 陈敏（手机：15995700090）

张权（手机：18262268572）

日期： 2021 年 7 月

2021 中国工程机器人大赛暨国际公开赛诚信比赛承诺书

本参赛队（队伍编号 202103210111463）是 2021 中国工程机器人大赛暨国际公开赛参赛者。本人已认真阅读工程机器人大赛官网发布的相关比赛信息，阅读并理解中国工程机器人大赛暨国际公开赛 2021 年关于比赛的相关规定，并郑重做出如下承诺：

1. 本参赛队承诺签名为本参赛队指导教师和队员本人真实姓名，承诺书一经签署就意味着承诺人接受并承担本承诺书的全部责任和义务。
2. 本参赛队承诺参赛所用机器人、场地以及其他比赛相关工具/设备均真实、有效，符合赛项规则要求，符合其他参赛相关要求。提交的所有参赛材料真实、有效。
3. 本参赛队自觉服从中国工程机器人大赛暨国际公开赛的统一安排，接受组委会的监督和检查。
4. 本参赛队自觉遵守相关法律和比赛纪律、比赛规则，诚信比赛，不违规、不作弊。
5. 本参赛队承诺整个比赛过程严格按照赛事规则进行，按照要求提交材料，所提交参赛材料均真实有效且符合参赛要求。
6. 本参赛队承诺本队参赛机器人属于本参赛队所有，同一机器人没有重复参加比赛。
7. 本参赛队保证所用比赛场地规范、计时计分准确、视频拍摄真实。
8. 本参赛队承诺正式比赛视频在录制意外中断时则需进行重新录制，不会对正式比赛视频中的比赛过程内容进行剪辑。
9. 大赛技术委员会和裁判有权对参赛队伍参赛所使用的机器人、场地以及其他相关比赛用物品的真实性、有效性提出质疑，参赛队伍有责任和义务回复质疑。对有重大异议的作品，大赛技术委员会和裁判有权要求参赛队伍到现场进行重新比测。
10. 非不可抗因素，需要参加现场比赛的队伍要按照要求按时参加现场比赛。

若本参赛队违背上述各项承诺，自愿承受因此产生的后果。指导教师对所有真实性负全责，各高校盖章前要检查真实性。

承诺人签名：

指导教师 1 姓名(手签) 陈永 身份证号 320683198108238024 联系方式 15995700090

指导教师 2 姓名(手签) 张权 身份证号 320203198312252532 联系方式 18262268572

队员 1 姓名(手签) 徐铭鸿 身份证号 310105200012131000

队员 2 姓名(手签) 赖向康 身份证号 44010420001202101X

队员 3 姓名(手签) 江卓远 身份证号 210303200010312000

所在单位公章

2021 年 7 月 8 日



关于技术报告使用授权的说明

本人完全了解 2021 中国工程机器人大赛暨国际公开赛 (RoboWork) 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和比赛组委会共同所有, 比赛组委会可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛机器人的视频、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: 徐铭鸿 赖向康 江钰

带队教师签名: 陈敏, 张权

日期: 2021.7.8

摘要

四足仿生移动机器人有着在较为复杂的地形环境中执行任务的能力，在救援，科学探索等工作上有着杰出的贡献。本文作为本参赛队伍的技术报告，首先在开头介绍了四足机器人的应用以及国内外对其研究的概况。本文的核心部分为对本队伍机器狗的机械、硬件及软件方面的分析。在机械部分，本文分析了五连杆机构在机器狗腿部的应用以及机体结构的设计思路；在硬件方面，本文主要根据比赛要求分析了如何选择主控板及使用总线舵机的优势。在软件设计部分，本队伍算法的创新之处在于通过状态机算法实现机器狗在比赛中不同状态之间的切换以便完成赛道要求。同时，在图像识别中，本队伍优化不同的算法以及尝试不同的色彩空间以提升巡线以及特定颜色物体识别的成功率。

关键词：四足机器人；机械及硬件设计分析；图像识别

Abstract

Quadruped mobile robots have the ability to execute tasks in complex terrain and make a great contribution to rescue, exploration, etc. This paper is a technical report of our team which firstly introduce domestic and foreign general research situation of the topic of quadruped robots. The core section of this paper is the analysis of machine design, hardware design and algorithm design. In terms of machine design, this paper analyses how the five-bar linkage applies to the leg of the robot dog and introduces the design thinking of structure design. In terms of hardware design, methods to choose main control board and the advantages of using bus servo motors are discussed. In terms of algorithm design, our program achieves status transition of robot dog which is required during the match, by building a state machine algorithm. Meanwhile, different algorithms and color spaces are applied to improve the success rate of tracking the line and recognizing specified objects.

Key words: Quadruped robot; Machine and hardware design; Image identification

目 录

0 引言/综述.....	6
1 系统整体设计.....	7
1.1 机体设计	7
1.2 程序设计.....	7
2 机械结构设计.....	7
2.1 腿部结构设计.....	7
2.2 机体结构设计.....	9
2.3 机体内部硬件安装.....	10
2.4 本体材料选择.....	11
2.5 整体效果图.....	12
3 硬件设计.....	12
3.1 主板—树莓派	12
3.2 多功能扩展板	13
3.3 锂电池.....	14
3.4 舵机.....	14
3.5 摄像头.....	17
3.6 机器狗机体参数.....	18
4 软件设计.....	18
4.1 运动控制.....	18
4.2 动作组控制.....	19
4.2 视觉识别.....	19
4.3 逻辑实现.....	22
5 系统开发与调试.....	23
5.1 巡线功能测试.....	23
5.2 识别标志功能测试.....	24
5.3 动作控制测试.....	24
5.4 完整功能测试.....	24
6 结论.....	24
参考文献.....	25
附录.....	26

0 引言/综述

随着机器人技术的发展，各类型、各领域的机器人应运而生。其中，移动机器人在救援、科学探索等工作上有着杰出的贡献。移动机器人按移动方式可分成轮式/履带式机器人及运动仿生机器人。轮式/履带式机器人由汽车行驶技术延伸而来，但其绝大部分只能在相对平整的地面上工作，无法胜任需在复杂地形上执行的任务。然而，基于仿生技术的运动仿生机器人可依靠其不同的移动方式（足式移动，蠕动，扑翼飞行等）适应更为广泛的地形及环境。近年来，国内外研究人员着重对足式机器人中的四足机器人进行研究，因其仿真对象牛，马，驴和羊拥有相应出色的负载及野外行动能力。这些特征也是让机器人能够在恶劣环境中工作的关键^[1]。

在美国，波士顿动力公司一直专注于研发人形机器人以及机器狗。其机器狗“Spot”能完成矿区深处安全检测、核事故区域探测，检查高压设备等任务。瑞士苏黎世联邦理工学院研制出的 Star1ETH 柔性四足机器人能通过关节处放置柔顺机构获得自然动力，这一优势可以提高机器人被动适应性。在意大利，意大利理工大学研制出 HyQ 电液混合驱动的四足机器人^[2]，其腿部髌侧摆关节采用电驱动而髌纵摆关节和膝关节采用液压驱动。

在国内，较有代表性的四足机器狗是宇树科技开发的莱卡狗，其内置的自主研发的电机系统能输出 18kW 的瞬时功率，并可依靠独立电源运行 2-3 小时，但其视觉导航及自主性仍有待加强。浙江大学的机器人绝影和赤兔是高校领域的代表。另外，上海交通大学，哈尔滨工业大学，国防科技大学等高校也研制出了一些具有代表性的产品。但相对于国外，国内对四足机器人研究起步较慢，基础较弱，在系统设计、关键器件等方面存在较大差距，机器人的研发仍处于模仿和追赶阶段，这导致了国内四足机器人的仿生结构、驱动系统性能指标、自主性能力等指标上处于落后^[2]。

随着人工智能、步态规划、计算机视觉等领域的发展，四足机器人已实现从最初的静态稳定步行到当今的动态稳定步行。未来的目标将是研发出负重能力更强，移动速度更快，续航时间更长以及环境适应力更强的四足机器人。在提高环境适应力方面，要实现自主定位、运动及导航，通过多传感器融合技术让机器人与环境交互的传感器已成研究热点^[2]。

该论文为 2021 年“中国工程机器人大赛暨国际公开赛（视觉机器狗）”技术报告，内容包括了本参赛组对参赛机器狗的机械、软/硬件设计思路分析以及针对比赛规则的机器狗调试过程。本队伍机器狗使用的是深圳幻尔科技有限公司的 PuppyPi Pro 机器狗。

1 系统整体设计

1.1 机体设计

本队伍机器狗 PuppyPi 机体设计分为腿部设计、机体外壳设计、机体内部硬件选择、安装及外壳材料选择。腿部设计理念分析来自波士顿动力机器狗，并在保持每条腿有 2 个自由度的基础上把腿部设计成连杆机构。外壳设计是基于机器狗腿部运动及对应舵机安装确定外壳形状。根据比赛要求，主控板选择树莓派 4B 并搭配幻尔公司的扩展板以便驱动舵机和方便舵机与主板相连。在选择外壳材料时，需考虑到材料使用广泛性及本次机器狗使用条件。

1.2 程序设计

在识别黑线程序设计中，本队伍使用天津二值化来提取黑线并通过设计相应算法消除天津二值化后依然存在的少量噪声，保证了黑线中心与图像中心偏差的准确性来使机器狗巡线稳定。

在识别彩色的过程中，本队伍通过采用不同的色彩空间作图像掩膜，最后发现采用 HSV 色彩空间能使识别率得以提高。

在整体逻辑选择上，本队伍使用有限状态机实现功能之间的切换。将比赛分为 6 个阶段，对应 PuppyPi 拥有的 6 种状态，每个状态下都有各自的动作和切换条件。

2 机械结构设计

2.1 腿部结构设计

2.1.1 腿部设计理念参考

参考了现阶段较为出色的四足机器人后（如波士顿动力的 BigDog、Spot 等）发现，大部分四足机器人单条腿部都有 3 个自由度，即 3 个电机。其中 1 号电机和 2 号电机使得腿在平面运动，而髋关节电机使得腿外旋或内旋，即使腿部在不同平面上运动。

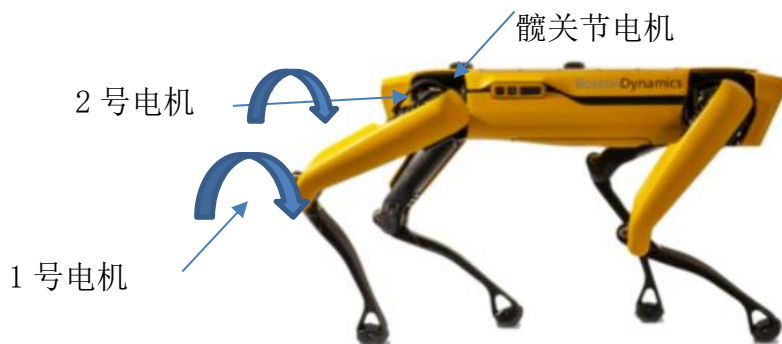


图 1 波士顿动力机器狗

基于参考发现及比赛要求，本队伍机器狗单条腿部也应满足 3 自由度的条件。4 条腿部的 1 号电机和 2 号电机可让机器狗实现直线前进运动、低姿态行走及上下桥。4 个髋关节电机和 4 条腿部的 1 号和 2 号电机相互配合使得机器狗实现转弯功能。

2.1.2 腿部设计改进

由于本组机器狗只是参赛机器，大重量负重、崎岖地形等因素皆无需考虑，因此从减轻重量、经济的角度出发，机器狗腿部无需设计成十分粗壮，取而代之的应是连杆机构。当腿部设计成连杆机构后，原先膝关节处电机无法放置在膝关节处，因此我们需要重新设计腿部结构。基于机构具有确定运动的条件：主动件数 $n =$ 机构自由度 F ，我们需要设计具有 2 自由度的连杆机构。基于公式 $F = 3n - (2P_l + P_h)$ ，其中 $P_h = 0$ 及草图推算，我们得知当 $n = 3, P_l = 5$ 时，表达式成立。基于以上分析和计算，腿部新设计如下：

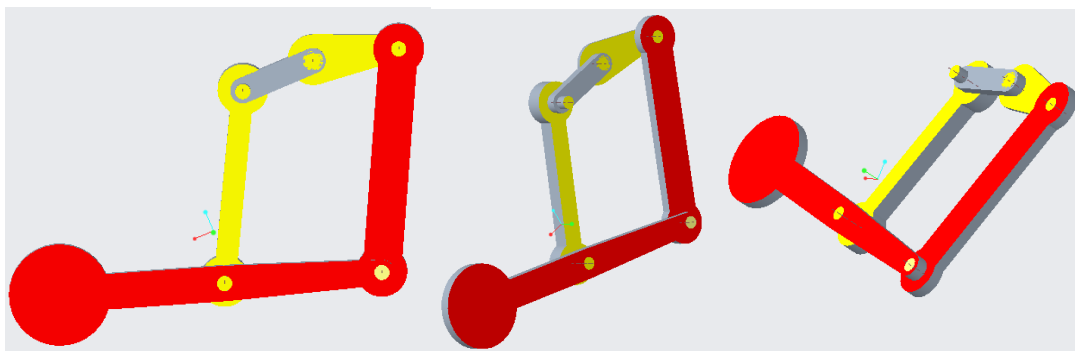


图 2 Cero 腿部设计图



图 3 腿部实物图

2.2 机体结构设计

2.2.1 机器狗机身设计

PuppyPi 机身整体造型应为长方体，但由于考虑到髋关节的内旋及外旋，在设计机体外壳时应为髋关节留出能自由转动的位置。由于机器狗 4 个髋关节占用了长方体的 8 个顶角的位置，于是外壳的上盖及底部被设计成了“十”字造型，上盖和底部的连接依靠机体侧边的两个连接件。下图为上盖和底部连接固定设计图。其中黄色件为连接件，红色件为机器狗上盖和底部。

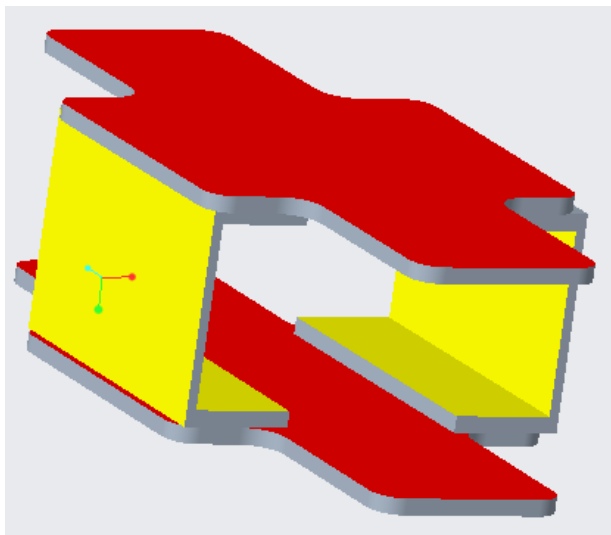


图 4 Cero 机体外壳设计图

2.2.2 机器狗头部设计

由于机器狗视觉赛要求机器狗具有巡线及识别特定颜色物体的要求，我们需要在机器狗头部安装可旋转的摄像头。摄像头及控制摄像头旋转的舵机被包装在头部外壳内，头部外壳的固定依靠与机身上盖和底部相连。

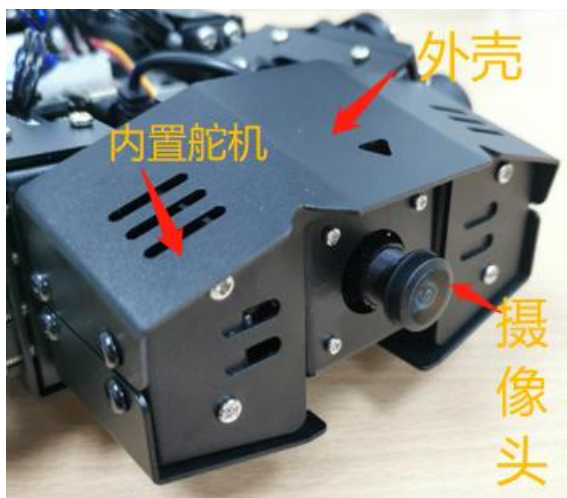


图 5 头部外壳及摄像头组件实物图

2.3 机体内部硬件安装

2.3.1 主板、扩展板及电源

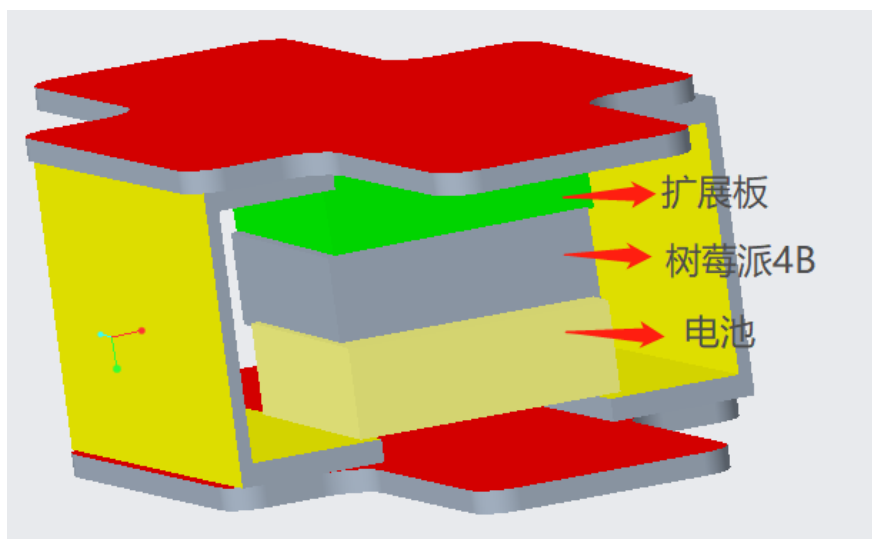


图 6 内部硬件放置图

2.3.2 腿部舵机

四个控制髋关节内旋外旋的电机分别两两固定在机体外壳底部，并与上盖底部连接件相紧贴。

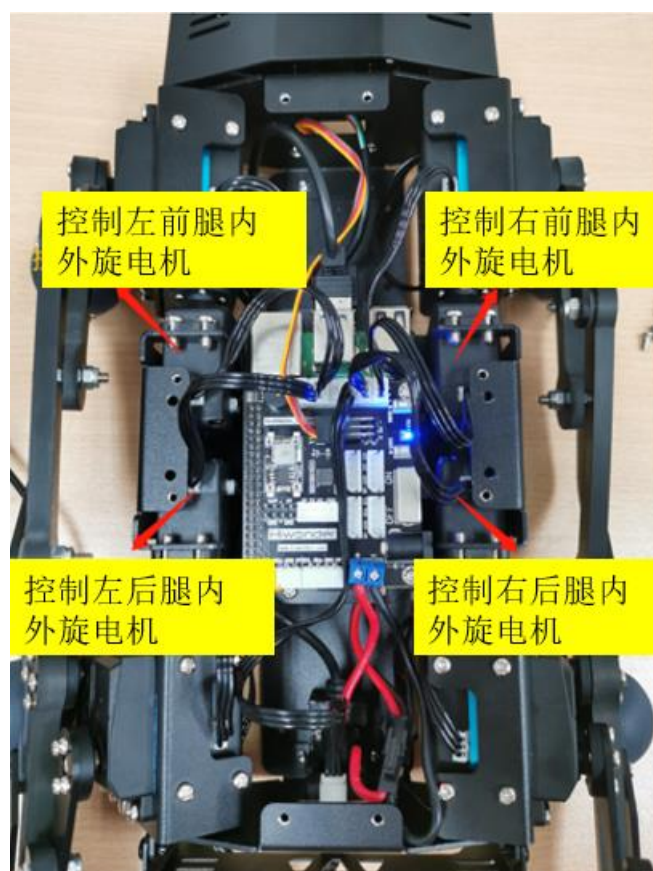


图 7 髋关节舵机

实现每条腿平面运动的两个舵机安装在预先设计机身时留出的位置处，并靠与髋关节舵机、头部外壳相连接来固定。

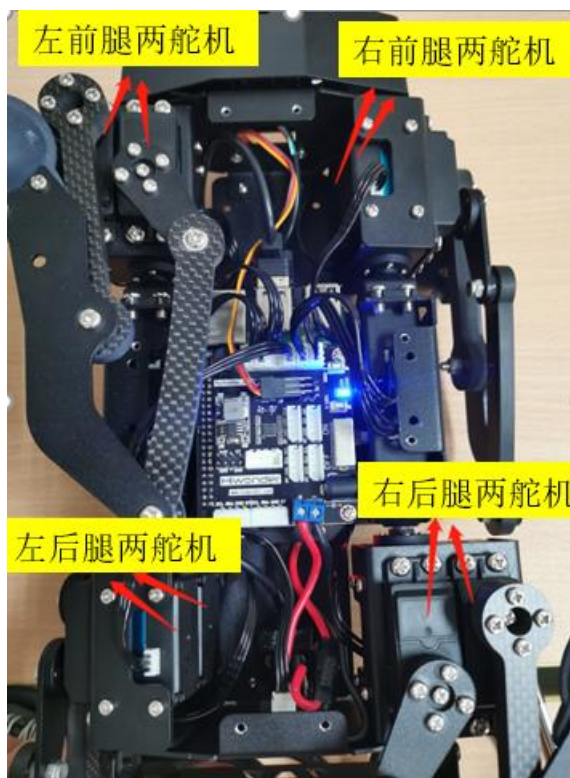


图 8 腿部舵机

2.4 本体材料选择

机身材料的选择通常要考虑到机器人使用广泛性、使用条件等因素。在权衡各因素之后，我们选择 6061 轻硬铝合金材料作为机体支架材料，腿部部分连杆机构采用了碳纤维板，足部则采用橡胶球。理由如下分述。

2.4.1 使用广泛性

由于 6061 铝合金具有良好的塑性、耐腐蚀性、可成型性并具有一定的强度，广泛用于工业机器人、电子产品外壳、航空、兵器等行业。其使用广泛性为我们选择材料提供了参考。

2.4.2 使用条件

本组机器人用于机器人比赛，并非工业级机器人，无需承受大重量负载及面对高温等恶劣工作环境，因此该机器人无需使用到硬度、强度、抗变形能力极强的钢材料。此外，由于本次机器人比赛需要机器人有较快的移动速度，在选择材料的时候就应选择轻质材料，减少机器人负载，故铝合金和碳纤维材料是较好的选择。

表 1 不同材料参数比较

材料	密度 ρ (kg/m^3)	弹性模量 E (GPa)
铝合金	2600-2800	70-79
碳纤维	1620	180-210
钢	7850	190-210

机器狗前进需要靠摩擦力。足底部与地面的摩擦力越大，可使机器狗前进得更加容易。本组机器狗选用橡胶球安装在机器狗足底部，一方面是考虑到橡胶相对地面能拥有较高的摩擦系数，另一方面是考虑到圆形的足部能较好地给腿部及机身各个角度的支撑，即便腿部对地面的作用力并非垂直于地面，橡胶球依然能为腿部及机体的平衡提供支持力。这一设计为机器狗实现多样的运动奠定基础。

2.5 整体效果图

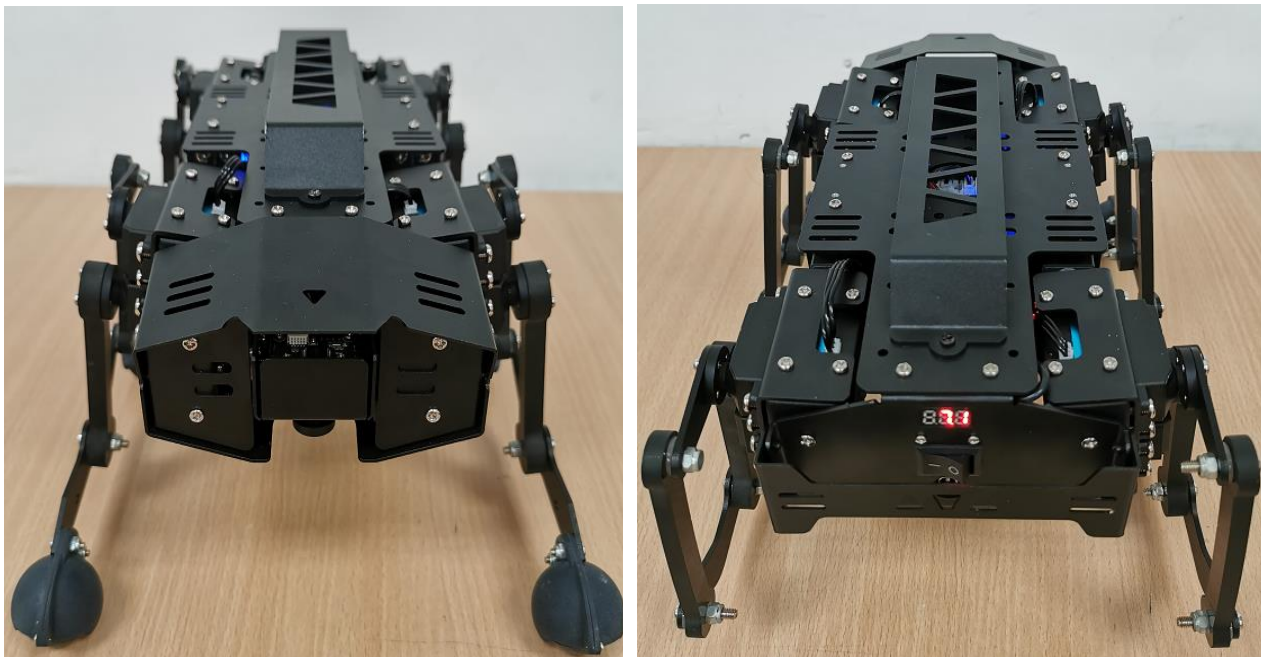


图 9 头部及尾部视角图

3 硬件设计

3.1 主板—树莓派

现状市面上存在的控制主板有开源硬件类（如树莓派、Arduino）及传统单片机（STM32、51）等。其中，树莓派所使用的 CPU 是属于 MPU（微处理器）类，而 Arduino 和 STM32、51 所使用的 CPU 是属于 MCU 类（微控制器）。微处理器可以被认为是小型电脑处理器，所以其相对于微控制器具有更强的运算能力以及更多的使用功能。本队伍使用树莓派 4B 作为开

发控制主板。



图 10 树莓派 4B

3.1.1 树莓派 4B 参数

表 2 树莓派 4B 控制器参数

树莓派 4B 控制器参数	
处理器	Arm Cortex-A72 64bit@1.5GHz
内存	4GB LPDDR2 SDRAM
无线网络	2.4G & 5GHz 802.11.b/g/n/ac
存储	16GB TF Card@class 10/UFS 1
部分 IO	1*I2C、8*GPIO、1*UART/6*Bus Servo Port

3.1.2 树莓派更强的运算能力和功能——机器视觉

本次机器人比赛需要进行巡线、特定颜色物体识别任务。实现机器视觉通常的办法是在操作系统上通过熟悉的编程语言（本队伍使用 Python）等语言和熟悉的库进行开发。由于树莓派的硬件性能（如 CPU、RAM 等）比 Arduino 等单片机的强大，我们可以在树莓派上运行完整的操作系统以便后期进行机器视觉编程开发，通过多线程控制运动、处理更多的数据等等。反观传统单片机，一次只能运行一个烧录进去的程序，功能相对单一。

3.2 多功能扩展板

树莓派 GPIO 输出的电压峰值为 3.3V，输出的电流峰值为 20mA，这并不足以驱动机器狗上的舵机运动，电机直连树莓派甚至可能导致烧坏树莓派上的电路。于是我们需要一个扩展板来增大树莓派输出的电流来驱动电机。本队伍所选的是幻尔科技公司设计的多功能扩展版。

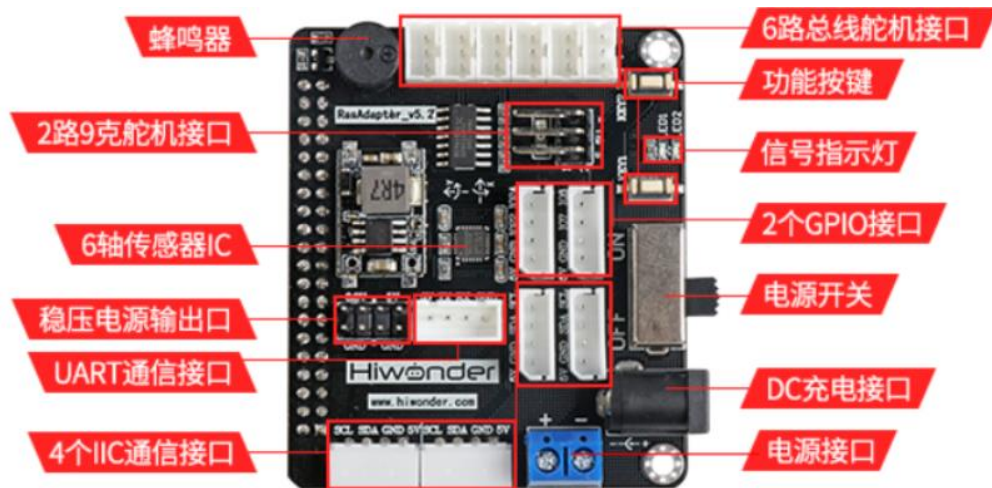


图 11 幻尔科技多功能扩展板

3.2.1 扩展板参数

表 3 扩展板参数

主要接口及器件	数量
9 克舵机接口	2 路
总线舵机 3pin 接口	6 路
UART 通信接口	1 个
IIC 接口	4 个
稳压电源出口	1 个
可编程 LED 灯和按钮	2 路

3.2.2 扩展板优势

扩展板除了可以增大电流外，也引出了多个树莓派主板上的接口，方便了电机与控制主板的连接。同时，本队伍部分电机选用了总线舵机和一个普通 PWM 舵机，该扩展板亦提供了相应特定的接口方便设计者使用。

3.3 锂电池

该机器狗供电电源为一个 7.4V，2500mah 的锂电池。

3.4 舵机

为了实现机器狗各种动作，每条腿部安装了 3 个舵机，其中 2 个控制腿部平面运动，另外 1 个控制髋关节的内外旋。其次，我们为实现摄像头的转动安置了一个舵机在机器狗头部。因此，本队伍机器狗全机体一共安装了 13 个舵机。在舵机类型的选择上，我们选择了 12 个串行总线舵机（8 个强磁总线舵机+4 个智能总线舵机）控制机器狗运动，选用了 1 个普通 PWM 微型防堵舵机控制机器狗头部的摄像头转动。

3.4.1 一般模拟舵机构成及工作原理

一个舵机内部包含了控制电路板、小型直流马达、变速齿轮组（减速器）、位置检测器（电位器/编码器）等部分。控制电路板接收来自信号线的控制信号，并将其转换为一个直流偏置电压，该电压与电位器的基准电压进行比较获得电压差，并输出到驱动芯片中从而控制电机转动。该电压差正负代表电机旋转方向而大小决定要旋转的角度。电机的转动提供了原始转动力矩，这一力矩带动变速齿轮组转动从而产生更高扭力的输出。位置检测器与转动轴相连并将位置信息通过电信号反馈到控制电路板上，控制电路板可根据反馈的位置信息决定电机的转动方向及速度，这样一个闭环控制可提高运动的精度。

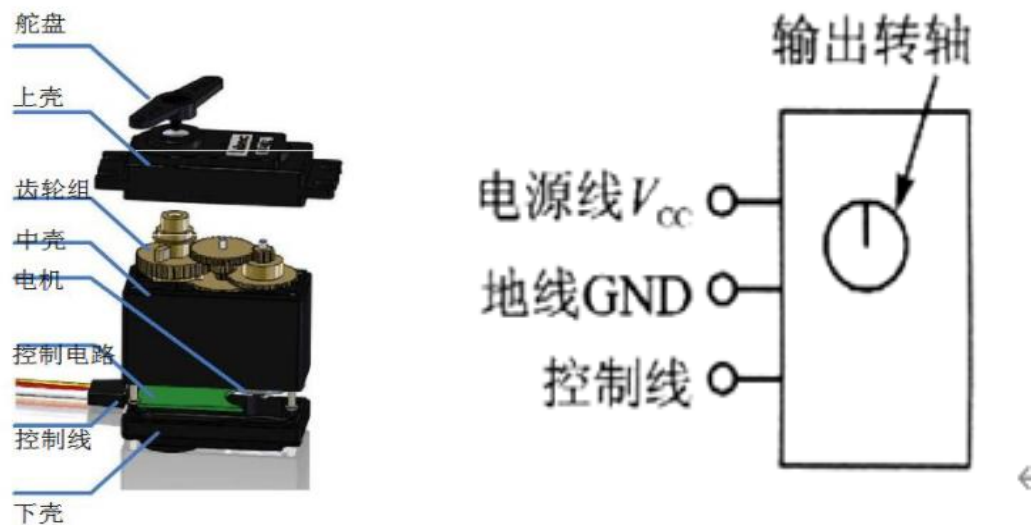


图 12 舵机结构及导线示意图

标准舵机拥有 3 条导线，分别是电源线、地线以及信号线。在模拟舵机中，信号线传输的是周期为 20ms（即频率为 50Hz）的宽度可调的方波信号。

3.4.2 总线舵机优势

数字舵机相比于模拟舵机，在其内部控制电路多了微处理器和晶振。总线舵机作为数字舵机的衍生品，具有以下优势特点

➤ 响应快

微处理器可将外部输入的 PWM 参数进行调整并向马达发送更高频率的动力脉冲，这意味着马达在同一时间内能获得更多的脉冲信号，响应时间更快，加速减速转动更加顺滑。

➤ 接线方便

若不使用总线舵机，本队伍中控制机器狗的 13 个舵机均需单独连到控制板上，这时将导致所需接口过多、布线混乱、排错困难等问题。总线电机可使机器狗关节处的电机相互串联，随后只需通过串口给总线发送一条控制所有具有不同 ID 的电机的指令，即可组成一个动作组。总线电机的选择降低了连线布线的困难，亦使电机控制更为方便。

➤ 反馈参数多

相对于传统模拟舵机，智能总线舵机能反馈电压、电流、位置等参数，方便了设计者的调试与排错。

3.4.3 单轴舵机尺寸图

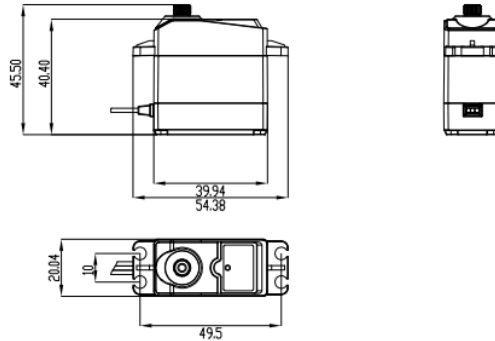


图 13 单轴舵机 CAD 设计图

3.4.4 舵机参数

表 4 强磁总线舵机（腿部舵机）

主要技术参数	数值
重量	62g
尺寸	54.38mm*20.04mm*45.50mm
工作电压	6-8.4V
堵转扭矩	20kg.cm (7.4V)
转动范围	0° -240°
控制角度范围	0-1000，对应 0° -240°
舵机精度	0.3°
转动速度	0.12sec/60° (7.4V)
控制方式	UART 串口指令
舵机 ID	0-235（用户可设置）
参数反馈	温度、电压、位置

表 5 LX-1501 智能总线舵机（控制髋关节内外旋舵机）

主要技术参数	数值
重量	62g
尺寸	54.38mm*20.04mm*45.50mm
工作电压	6-8.4V
堵转扭矩	17kg.cm (7.4V)
转动范围	0° -240°
控制角度范围	0-1000，对应 0° -240°

舵机精度	0.3°
转动速度	0.16sec/60° (7.4V)
控制方式	UART 串口指令
舵机 ID	0-235 (用户可设置)
参数反馈	温度、电压、位置

表 6 LFD-01M 防堵转舵机 (单目摄像头旋转舵机)

技术参数	数值
重量	9.9g
工作电压	4.8V-8.4V
扭矩	1.5KG.cm (4.8V) / 1.8KG.cm (6V)
转动角度	0° -180°
转动速度	0.12sec/60° (4.8V) /0.10sec/60° (6V)
控制方式	PWM 脉宽控制
PWM 脉宽范围	500-2500us, 对应 0° -180°

3.5 摄像头

本队伍所使用的摄像头为单目摄像头，其技术参数列于下表。

表 7 摄像头参数

技术参数	数值/内容
型号	hv3808
产品类别	USB 摄像头
传感器类型	CMOS
传感器像素	30 万
最高分辨率	640*480
输出格式	Mjpeg 输出
对焦	可手动对焦
连接方式	USB 免驱

3.6 机器狗机体参数

表 8 机器狗机体参数

技术参数	数值/内容
本机尺寸（长*宽*高）	335mm*176mm*218mm
本机重量	约 1.90kg
电池	7.4V 2500mah 锂电池
续航时间	在线调试 90min，持续运行 60min
机体支架材料	机身：6061 轻硬铝合金，足部采用橡胶球
控制方式	VNC 远程操作界面、PC 上位机控制
关节总数	13（头部*1，腿部 3*4）
舵机总数	13（强磁总线舵机*8，智能总线舵机*1， 微型防堵转舵机*1）

4 软件设计

4.1 运动控制

首先介绍两个概念：动作组和动作。一套动作可以看作是动作的组合，称为动作组。一个动作是动作组的最小单元。动作组至少含有一个动作。PuppyPi 的运动通过完成一个个动作组实现。

PuppyPi 由 13 个舵机驱动。只有通过控制各个舵机的转动，才能控制它的运动。

分析至此，可以将运动控制拆分成两个渐进的抽象层。首先是舵机控制，然后是动作组控制。运动控制的程序被划分成这两个模块。接下来将依次介绍这两个模块。

4.1.1 舵机控制

PuppyPi 每条腿上有三个由串口总线控制的舵机。一共十二个串口总线舵机。单目摄像头的俯仰由一个 PWM 舵机控制。所以 PuppyPi 的舵机一共由两种控制方式。一是串口总线控制，二是 PWM 控制。这两种控制方式将舵机控制模块拆分成两个部分。

4.1.2 控制串口总线舵机

PuppyPi 使用名叫 UART 的串行总线。其通信遵循 Hiwonder 在其舵机控制扩展板上定义的总线通信协议。为了使用方便，将 UART 数据包的构建，检验，收发封装成一个受保护的类 _UART。这个类负责最底层的通信用数据的处理。波特率，帧头，与串口舵机指令对应的数字，也根据 Hiwonder 的通信协议定义好一并封装在类中。

同样为了使用方便，舵机的功能被封装为一个个函数。PuppyPi 的结构限制了舵机的自由转动，所以有的函数内部进行了入参的边界限制。每个函数主要通过调用 _UART 的方法实现其函数名所述的功能。

控制舵机时并不关注某个特定的串口舵机，所以没有使用面向对象的思想进一步封装成类，而是保持为一个个独立的函数。

4.1.3 控制 PWM 舵机

PWM 舵机与串口总线舵机有两点不同之处。一， PWM 舵机直接影响程序设计时关注的主要对象——单目摄像头。二， PWM 舵机不遵循 Hiwonder 总线通信协议。为了之后设计摄像头类时能通过继承复用代码，这里将 PWM 舵机控制的代码封装为类 PWMServos。由于 PWM 舵机不遵循前述的通信协议，通过调用 pigpio 这个模块便能进行控制。因为实例化后实例的舵机 ID 不可更改，所以将成员变量 id 的访问限制设置为 private 并设置了其 getter。

4.2 动作组控制

将一个动作看作是一个舵机的运动，那么动作组就是以动作组起始时刻为基点，各个舵机运动时段的排列。各个舵机应该是异步运动并互不干涉的，所以考虑使用线程实现动作的并发。

按时序排列的舵机运动可以看作是一组定时启动的任务；因此考虑使用 threading.Timer。另一方面，为了利用 join() 阻塞主线程直到整个动作组执行完毕，需要重写 threading.Timer。重写后的线程类命名为 _SerialBusServoThreads。它不应在 locomotion_control 模块外被引用，所以访问限制设定为受保护。_SerialBusServoThreads 定时调用函数。它并不会在这次调用后立刻消灭，而是会动作执行时间期间保持存活状态，直到动作结束后才消灭。如此，可以保证主线程直到动作组结束都处于被阻塞状态。

一个动作，也就是一次舵机运动，被划分为四个字段：开始时刻 start_moment、串口总线舵机 ID serial_bus_servo_id、脉宽，或者说位置 pluse、与旋转时间 rotating_time；那么动作组是这四个字段组成的动作列表。动作组以这种数据结构，以 csv 为文件格式，永久化在存储设备上。实例化一个动作组时可以读入一个动作组数据文件，也可以选择留空。动作数据在读入后被分配到成员变量 __action_list 中。可以使用 add_action() 向其中添加新的动作。新添加的动作暂时存放在内存中。为了保存供之后使用，程序中实现了一个永久化方法 persist() 用来保存数据。

do() 方法用来使 PuppyPi 执行动作组。do() 的逻辑如下。首先使用 _SerialBusServoThreads 构建一个线程列表 __thread_list，再启动列表中的所有线程，最后启动所有线程的阻塞功能。

4.2 视觉识别

在视觉识别部分，程序采用 opencv 库中的函数对摄像头获取的图片进行处理，最终实现 PuppyPi 识别不同标志的功能。

4.2.1 白平衡

在不同的光照条件下，摄像头采集到的 BGR 图像的色温会有显著差异。这种差异不利于进行精确的色彩识别。为了减低光源对图像色彩的影响，就需要使用色彩平衡算法。程序使用 OpenCV 库中的灰度世界算法进行白平衡。白平衡后的 BGR 图像是后续色彩识别的材料。

4.2.2 转换色彩空间到 HSV

PuppyPi 摄像头拍摄图像后会返回一个矩阵。矩阵由 400 行 300 列像素组成。每一个像素存储一个像素值。数字化处理后的图片处于 BGR 色彩空间，也就是说每一个像素的 BGR 值表示该像素蓝色 (B)，绿色 (G)，红色 (R) 各自的程度^[3]。为了方便 PuppyPi 的色彩识别，我们使用函数 `cvtColor()` 把获得的图片转换到 HSV 色彩空间进行处理。

在 HSV 色彩空间中，H 指色彩、S 指饱和度，代表颜色深浅、V 指明亮的程度^[3]。由于在 HSV 色彩空间中，色彩和亮度信息被分开存储，可以减小环境光照对颜色识别的影响，因此选择在 HSV 色彩空间下进行图片处理。

4.2.3 巡线部分

本队伍实现巡线的代码由颜色识别模块与相对应的摄像头转动、腿部运动模块组合而成。这一部分介绍的是巡线的颜色识别模块。

4.2.3.1 程序逻辑概要

首先得到一个二值图^[4]。二值图中黑线部分为白色，其余部分为黑色^[4]。再计算白色部分的中心^[4]。最后返回中心与图像横坐标轴中心的像素差。

4.2.3.2 设计难点与对策

巡线程序的设计难点有三个。一是黑色容易与环境色混淆。二是赛场的黑线中部反光十分严重。三是单目摄像头无法判断距离，所以黑线的宽度无法判断^[5]。

针对第一个难点，经过观察，黑线总是处于摄像头捕捉到的图像的下半部分。因此，只取图像底部的一行像素进行处理，从而规避了环境色彩的干扰。针对第二个难点，我们放弃识别黑色，转而使用大津二值化算法自动处理。针对第三个难点，程序假定最宽的白色部分为需要巡行的黑线。

4.2.3.3 提取巡线路径

首先，我们需要将摄像机所获取的原图转为灰度图像。该步骤的操作是为了实现第二步大津二值化（或称 Otsu 方法）的操作，因为实现大津二值化的函数语句中输入的图像需要是灰度图。把原图 RGB 色彩空间转换为 GRAY 色彩空间和大津二值化的函数语句分别是：`cvtColor()` 和 `threshold()` 函数。采用大津二值化的优点是计算机会自动算出合适的阈值来实现二值化。

执行大津二值化函数后，黑线会变成白色，同时该函数会返回 2 个值，一个是大津二值化算法所计算出的使赛道黑线和周围环境分离的阈值，另外一个返回值是处理过后的二值图。

得到该二值图后，该图片上仍可能会有噪声的存在，于是我们采用开运算（先腐蚀后膨胀）来消除这些噪声。我们可以用 `morphologyEx()` 语句实现开运算。经过消噪处理后，理想的图像应是只存在白色矩形（即处理过的黑线）和黑色的背景。

4.2.3.4 找巡线路径的中点

在机器狗的运动过程中，白线不可能永远保持在图像中心，其中心与图像中心会产生偏差。为了让机器狗持续巡线，我们需要找出白线的中心，并计算出白线中心与图像中心的偏差，基于此偏差来确定机器狗是选择执行、左转或者右转。找出白色的中点需要找到其像素横坐标以及像素纵坐标。纵坐标可先人为设定。确定纵坐标后，横坐标的计算是基于该纵坐标所确定的横线上。白线中心横坐标的具体位置通过计算白色线两侧边缘的横坐标的平均值得到。

纵坐标的确定由下方语句。当参数 `ordinate` 确定后，也就找出某一行的像素。

```
a_row_of_pixels = opened_otsu_binary_img[ordinate]
```

先通过以下语句确定这一行像素的索引，即横坐标。

```
enumerate(a_row_of_pixels)
```

此时我们需要找出白色像素点在这一行像素中集中的位置。由于在灰度图中，数值 0 是黑色而 255 是白色，于是我们需要找出这一行像素点中数值是 255 的像素点。我们首先遍历由 `enumerate(a_row_of_pixels)` 中列举的像素值及其索引并通过条件语句 `intensity == 255` 在遍历的过程中找出这行中白色像素点值及其索引/坐标，分别赋值给 `abscissa`, `intensity`。

代码实现：

```
for abscissa, intensity in enumerate(a_row_of_pixels) if intensity == 255
```

找出白色像素横坐标后，把这些横坐标赋值给一个名叫 `white_pixel_abscissas` 元组。

代码实现：

```
white_pixel_abscissas=[abscissa for abscissa, intensity in
enumerate(a_row_of_pixels) if intensity == 255]
```

但在上述的元组中，所含的白色像素点有可能是二值图经过消噪处理后仍存在的噪声。白线两端所包含白色像素点的索引应是连续的，而噪声的则是离散的。在使用 `enumerate()` 函数返回上述元组的像素点坐标及其新索引时，函数会把每一个像素点及其对应的新索引放置在一个元组中（新索引，单个白像素点横坐标），即不断返回一个个具有两个元素的元组。我们让每个返回的元组的元素作差，若两个像素点横坐标是连续的，其各自元组两元素的差值应为一个常数，我们设其为 `key`；若两个点是离散的，其各自元组两元素的差值会不同。通过自定义作差函数加上 `groupby()` 函数使离散点和连续点相分离，并得到返回值 `key` 参数和属于连续白像素点的元组集合。通过函数 `itemgetter(1)` 获取每一个元组中第二个元素（即单个白像素点横坐标），并映射到新的列表中。这样就得到了包含连续白像素点的列表。

代码实现：

```
consecutive_abscissas = [list(map(itemgetter(1), group)) for key, group in
groupby(enumerate(white_pixel_abscissas), lambda _: _[1]-_[0])]
```

随后，通过找出这组像素点最远的两个点找出白线的边缘像素点横坐标，这两个像素点相加除以二得到中点坐标。

代码实现：

```
presumable_line_abscissas = builtins.max(consecutive_abscissas, key=len)
presumable_line_centre_abscissa=int((presumable_line_abscissas[0]+presumable_line_abscissas[-1])/2)
```

4.2.4 识别蓝色矮门、黄色横条

程序首先将摄像机捕捉到的图像转换到 HSV 空间，再根据预先设定的色彩范围二值化图像^[6]。从二值图中找到所有轮廓后，找出拥有最大包围面积的轮廓，并假定这个轮廓是期望被识别的物体的轮廓。将这个轮廓与标准图形进行形状匹配，得到一个偏差值。如果偏差值小于预先设定的阈值，则判断检测到了期望识别的物体。

4.2.5 识别黑色十字

程序首先将摄像头拍摄到的图像转换为灰度图像，再通过大津二值化方法将灰度图像转换成二值图。转换到二值图后使用的方法与识别矮门和黄色横条相同。

4.3 逻辑实现

4.3.1 状态机概述

本作品通过状态机实现各功能之间的切换和功能函数的运行。状态机的本质是一种描述有时序事件的数学模型，通常由状态，动作，条件，转换四个概念组成^[7]。

- (1) 状态表示对象在一段时间中具有的行为特征。
- (2) 动作指对象在特定状态中需要执行的操作。
- (3) 转换是从原状态切换到目标状态的过程。
- (4) 条件指切换状态需要满足的前提。

比赛由不同阶段组成，在每个阶段 PuppyPi 拥有不同的任务，这与状态机的设计非常符合，又由于状态机状态划分易于理解，逻辑清晰和执行高效的优点，因此本组使用状态机实现 PuppyPi 的多个功能。

4.3.2 状态机基本设计

状态转换关系需要明确源状态，目标状态和条件。比赛中，状态切换的条件主要是识别到指定标志。因此可以写出各个状态的转换关系：

- (1) 源状态：“IDLE”，目标状态：“FIND_DOOR”，条件：启动指令(start)。
- (2) 源状态：“FIND_DOOR”，目标状态：“THROUGH_DOOR”，条件：找到蓝色矮门(found_door)。
- (3) 源状态：“THROUGH_DOOR”，目标状态：“FIND_BRIDGE”，条件：一段时间后自动切换(through_door)。

(4) 源状态：“FIND_BRIDGE”，目标状态：“CROSS_BRIDGE”，条件：找到黄色横条(found_bridge)。

(5) 源状态：“CROSS_BRIDGE”，目标状态：“FIND_CROSS”，条件：找到黄色横条(crossed_bridge)。

(6) 源状态：“FIND_CROSS”，目标状态：“IDLE”，条件：找到黑色十字线(finished)。

4.3.3 比赛功能实现

比赛主要由 6 个阶段组成：寻找矮门并巡线、通过矮门、寻找台并的巡线、通过桥、寻找黑色十字并巡线和经过终点，可以对应状态机的不同状态。各阶段的转换以颜色识别作为条件，在各个阶段中 PuppyPi 有着不同的动作行为。

PuppyPi 的初始状态是“闲置 (IDLE)”，当程序开始运行，条件“开始(start)”发送，PuppyPi 的状态切换到“寻找矮门”。

在“寻找矮门(FIND_DOOR)”状态下，PuppyPi 启动蓝色矮门识别并进行巡线。PuppyPi 重复执行状态下的动作。矮门在直线道路上，因此以连续直行作为 PuppyPi 抬头识别蓝色矮门的条件。当 PuppyPi 识别出蓝色矮门，将发送“找到矮门(found_door)”条件，使状态切换到“穿越矮门”，并低下头。

在“穿越矮门(THROUGH_DOOR)”状态下，PuppyPi 会以较低的姿态运动，以通过矮门。当行进超过一定时间，会认定 PuppyPi 通过矮门，并发送“通过矮门(through_door)”的条件，使状态切换到“寻找台阶”并切换回初始姿态。

在“寻找台阶 (FIND_BRIDGE)”状态下，PuppyPi 会开启黄色横条识别并进行巡线。当识别到黄色横条，会发送“找到台阶(found_bridge)”条件，使状态转换为“通过台阶”。

在进入“通过台阶(CROSS_BRIDGE)”状态时，PuppyPi 会执行爬上台阶的一系列动作。在爬上台阶后，会继续进行巡线的动作直到再次识别到黄色横条。此时，它会执行爬下台阶的一系列动作，在执行完毕后，会发送“通过台阶(crossed_bridge)”的条件，使状态切换到“寻找十字标记”。

在“寻找十字标记(FIND_CROSS)”状态下，PuppyPi 会继续进行巡线并开启黑色十字识别。当识别到黑色十字，PuppyPi 会再运动一段时间以确保通过终点，之后会发送“完成(finished)”的信号，使状态切换到初始状态“闲置”。

5 系统开发与调试

根据比赛规则，PuppyPi 的功能包括巡线、识别标志和动作控制。我们在开发的过程中对 PuppyPi 的实现这些功能的方案进行测试，并记录表现，以便我们改进功能。我们还对不同方法进行对比来选择最佳的方案。

5.1 巡线功能测试

我们使用与比赛要求相同的场地作为测试场地，移去矮门和台阶，让 PuppyPi 从起点出发进行巡线，测试其在直线与弯道上的表现。

经过测试，我们发现分析单行像素中白色像素的中点与假定中点偏差的方法和分析整个轮廓的中点与假定中点偏差的方法都可以正常完成巡线任务，但前者的运算量相对较少。

5.2 识别标志功能测试

我们把 PuppyPi 放置在不同标志前不同距离处，并在电脑上展示它拍摄到并处理过的二值化图像，通过观察来判断它识别标志的表现。

经过测试，我们发现先将拍摄到的图像转换到 HSV 或 LAB 色彩空间，再根据不同颜色的范围二值化图像都可以达到识别颜色的目的，但转换到 LAB 空间的方法容易受环境光的影响。而转换到 HSV 空间的方法由于色彩信息储存在一个通道中，受环境光干扰小。

对黑色十字的识别，我们发现使用 HSV 和 LAB 色彩空间转换的效果并不好。经过文献搜索，发现转为灰度空间再进行二值化是常用的识别方法。最后，PuppyPi 采用了这种方法用于巡线和识别黑色十字。

5.3 动作控制测试

我们使用与比赛要求相同的台阶和矮门测试了 PuppyPi 上下台阶和爬行的动作。经过测试与调整相应动作组文件中的参数（即特定舵机的旋转位置），PuppyPi 可以完成上下台阶和爬行的动作。

5.4 完整功能测试

我们让 PuppyPi 在测试场地上完整地完成任务进行测试，将出现的问题记录并改进。经过调试，我们的 PuppyPi 可以以较高的成功率完成所有任务。

6 结论

本文作为本队伍 Cyberpubby 关于机器狗 PuppyPi 的技术报告，在正文首段即介绍了四足机器人的特点以及国内外对此类机器人研究的现状。其次，本文详细地分析了机械与硬件的设计思路。其中机械设计中腿部的连杆机构设计以及硬件中总线舵机的使用是两大创新之处。在软件部分，运动控制算法的主要创新之处在于创建有限状态机算法实现机器狗不同动作或状态之间的切换，以完成比赛中各项要求。同时，在彩色视觉识别算法上，本队伍发现将摄像头获取的图像转换为 HSV 色彩空间后配合 Opencv 库的相应函数和自主设计算法能提高识别率；在黑线和黑色十字识别中，本队伍发现将图片转为灰度空间再进行二值化是较好的识别方法。在巡线程序设计中，本队伍创新之处在于采用了天津二值化提取黑线并通过相应算法设计消除二值图中仍可能存在的影响机器狗判断黑线中心的噪声，并通过计算黑线中心与图像中心的偏差以让机器狗自主执行的直行、左转和右转动作。

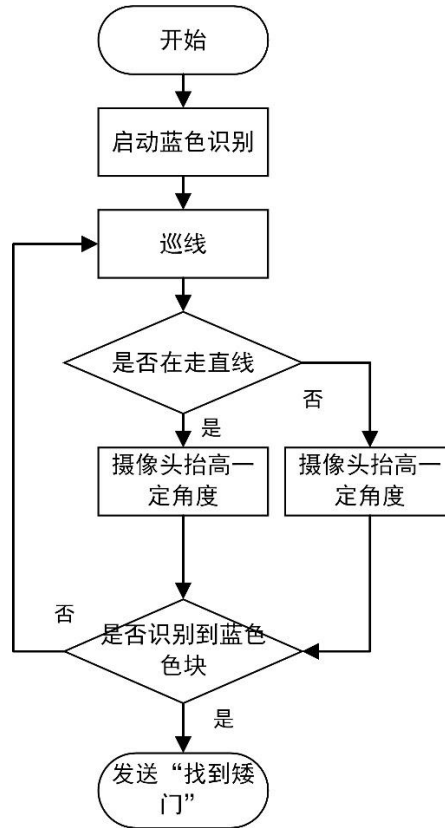
参考文献

- [1] 孟健, 刘进长, 荣学文等. 四足机器人发展现状与展望[J]. 科技导报, 2015, 33(21):59-63
- [2] 杨钧杰, 孙浩, 王常虹等. 四足机器人研究综述[J]. 导航定位与授时, 2019, 6(5):61-73
- [3] 赵宁, 赛奎春, 刘书娟. Python OpenCV 从入门到实践[M]. 吉林: 吉林大学出版社, 2021, 32-40
- [4] A.Ma' arif et al. Vision-Based Line Following Robot in Webots. 2020 FORTEI-
International Conference on Electrical Engineering
- [5] P. Croke, Robotics, Vision and Control. Spriner
- [6] 杜名欣. 可见光通信中颜色识别技术的研究与应用[D]. 吉林: 长春理工大学, 2012, 37-43
- [7] 石俊杰. 基于有限状态机的游戏角色控制系统设计与实现[D]. 武汉: 华中科技大学, 2016, 6-8

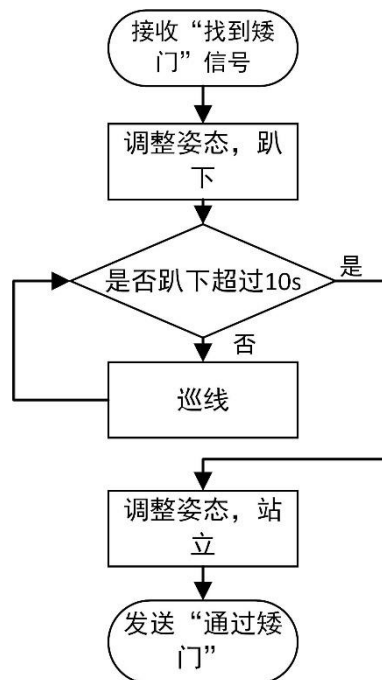
附录

1. 各部分程序流程图

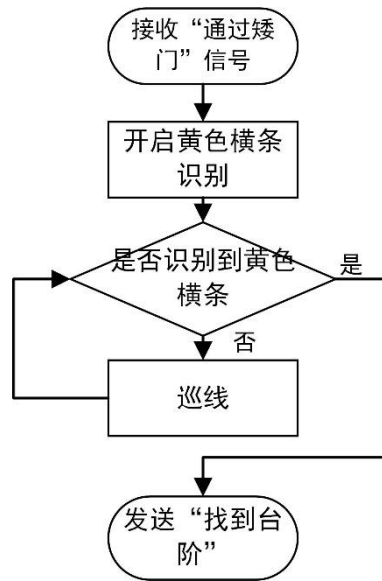
1.1 寻找蓝色矮门程序流程图



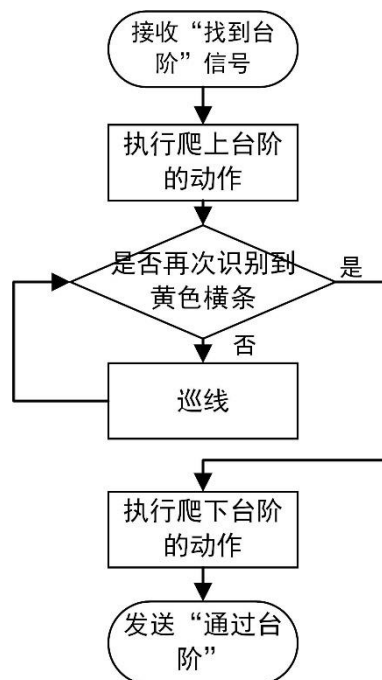
1.2 过蓝色矮门程序流程图



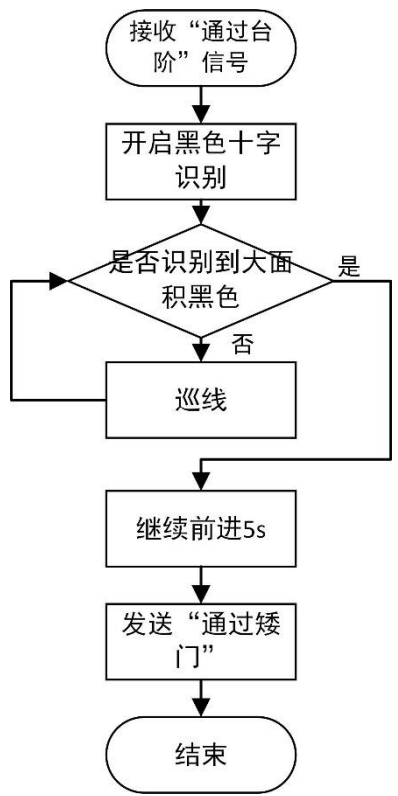
1.3 寻找黄条（台阶）程序流程图





1.4 上下台阶程序流程图



1.5 找到结束十字标识程序流程图



2. 程序

巡线程序附件	
主程序附件	

3. 代码

3.1 巡线部分代码

```
from itertools import groupby
from operator import itemgetter
import builtins

from cv2 import *

def calc_deviation_from_a_row_of_pixels(frame, ordinate: int) -> int:

    if not 0 <= ordinate <= frame.shape[0]:
        raise ValueError('Coordinate sys of the frame does not have this ordinate')

    otsu_thresh, otsu_binary_img = threshold(cvtColor(frame, COLOR_BGR2GRAY), 0,
    255, THRESH_BINARY_INV + THRESH_OTSU)

    opened_otsu_binary_img = morphologyEx(otsu_binary_img, MORPH_OPEN, (9,9))
    # remove tiny white regions

    # 取一行像素中最长的白色部分当作是线, 并计算它中点像素的横坐标
    a_row_of_pixels = opened_otsu_binary_img[ordinate]
    white_pixel_abscissas = [abscissa for abscissa, intensity in enumerate(a_row_of_pixels)
    if intensity == 255]
    if not white_pixel_abscissas:
        raise RuntimeError('Maybe the camera does not capture the line')
    consecutive_abscissas = [list(map(itemgetter(1), group)) for key, group in
    groupby(enumerate(white_pixel_abscissas), lambda _: _[1]-_[0])]
    presumable_line_abscissas = builtins.max(consecutive_abscissas, key=len)
    presumable_line_centre_abscissa =
    int((presumable_line_abscissas[0]+presumable_line_abscissas[-1])/2)

    return presumable_line_centre_abscissa - frame.shape[1]/2
```

3.2 主程序代码

```
from time import sleep
import sys
import numpy as np
from transitions import Machine
import cv2
import math
import threading

sys.path.append('/home/pi/puppy/')
from HiwonderPuppy import PUPPY, BusServoParams

sys.path.append('/home/pi/PuppyPi_PC_Software/')
from ServoCmd import *

sys.path.append('/home/pi/PuppyPi/')
from HiwonderSDK.Board import *
import HiwonderSDK.Board as Board
from HSVConfig import *
import Camera

sys.path.append('/home/pi/Desktop/AX-remoteProjs/developing/')
from line_following_methods import calc_deviation_from_a_row_of_pixels
```

```

SERVO_VALUE_LOOK_DOWN = 2450
SERVO_VALUE_LOOK_UP = 1500
SERVO_VALUE_DEVIATION = 40
def calc_max_contour_area(contours):

    if not contours: raise ValueError

    contours_area = np.array([cv2.contourArea(contour) for contour in contours])
    index = np.argmax(contours_area)
    """
    Contours is a Python list of all the contours in the image.
    Each individual contour is a Numpy array of (x,y) coordinates of boundary points of
    the object."""
    """
    return contours[index], contours_area[index]

def calc_longest_contour(contours):
    return max(contours, key=len)

puppy = PUPPY(setServoPulse = setBusServoPulse, servoParams = BusServoParams())

def stance(x = 0, y = 0, z = -15, x_shift = 2):# 单位 cm
    #x_shift 越小, 走路越前倾, 越大越后仰,通过调节 x_shift 可以调节小狗走路的平衡
    return np.array([
        [x + x_shift, x + x_shift, -x + x_shift, -x + x_shift],
        [y, y, y, y],
        [z, z, z, z],
    ])#此 array 的组合方式不要去改变

puppy.stance_config(stance(0,0,-15,2), pitch = 0, roll = 0)# 标准站姿
puppy.gait_config(overlap_time = 0.1, swing_time = 0.15, z_clearance = 3)
puppy.run() # 启动
Board.setPWMServoPulse(1, SERVO_VALUE_LOOK_DOWN +
SERVO_VALUE_DEVIATION, 100)
puppy.move_stop()

```

```
my_camera= Camera.Camera()
my_camera.camera_open()

camera_pos=2500

def line_tracking(deviation,enable_PWM):
    global camera_pos
    #print(deviation)
    if deviation is None:
        time.sleep(0.01)
        return
    sleep(0.003)
    if abs(deviation) < 65:
        puppy.move(x=12, y=0, yaw_rate=0) # go forward
        camera_pos -=27 if camera_pos > 1600 else 0
        if enable_PWM: Board.setPWMServoPulse(1, camera_pos, 100)
    if deviation >= 65:
        puppy.move(x=7, y=0, yaw_rate=-25 / 57.3) # turn right
        camera_pos += 22 if camera_pos < 2300 else 0
        if enable_PWM: Board.setPWMServoPulse(1, camera_pos, 100)
    if deviation <= -65:
        puppy.move(x=7, y=0, yaw_rate=25 / 57.3) # turn left
        camera_pos += 22 if camera_pos < 2300 else 0
        if enable_PWM: Board.setPWMServoPulse(1, camera_pos, 100)
```



```

class ImageProc:
    def __init__(self):
        self.block = False
        # 图像处理的各功能使能标记
        self.block_target_color = 'blue'
        self.enable_line = False # 使能巡线处理
        self.enable_block = False # 使能色块处理
        self.enable_PWM = True
        self.img_gate_standard=cv2.cvtColor(cv2.imread('gate_standard.jpg'),
cv2.COLOR_BGR2GRAY)
        self.img_cross_standard=cv2.cvtColor(cv2.imread('cross_standard.png'),
cv2.COLOR_BGR2GRAY)
        self.blue_area=0
        self.deviation=320
        # 启动图像处理线程
        threading.Thread(target=self.run, daemon=True).start()

    def tracking(self, frame):
        try:
            self.deviation = calc_deviation_from_a_row_of_pixels(frame, 420)
        except:
            pass
        line_tracking(self.deviation,self.enable_PWM)
#         return calc_deviation_from_a_row_of_pixels(frame, 350) # FIXME: ordinate

    def block_detect(self, img, color):
        img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        img_gb = cv2.GaussianBlur(img_hsv, (3, 3), 3)
        mask = cv2.inRange(img_gb, color_range[color][0], color_range[color][1])
        if color == 'blue':
            eroded = cv2.erode(mask, cv2.getStructuringElement(cv2.MORPH_RECT, (6, 6)))
# 腐蚀
            dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (3,
3))) # 膨胀
            contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_TC89_L1)[0] # 找出轮廓
            if not contours: # 没有找到轮廓
                return False
            cnt_large, area = calc_max_contour_area(contours) # 找出最大轮廓
            x,y,w,h = cv2.boundingRect(cnt_large)
            dist = cv2.matchShapes(self.img_gate_standard, dilated[y:y+h,x:x+w],
cv2.CONTOURS_MATCH_I1, 0)
            if (dist < 0.01 and w*h > 60000) or w*h > 190000 or area > 75000:
                self.blue_area=w*h if area < 75000 else 240000
            return True
        else:

```

return False

```

if color == 'yellow':
    eroded = cv2.erode(mask, cv2.getStructuringElement(cv2.MORPH_RECT, (6, 6)))
# 腐蚀
    dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (3,
3))) # 膨胀
    contours, _ = cv2.findContours(dilated, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_TC89_L1) # 找出轮廓
    if not contours: # 没有找到轮廓
        return False
    cnt_large, area = calc_max_contour_area(contours) # 找出最大轮廓
    if area > 40000:
        return True
    else:
        return False

```

```

if color == 'black':
    _, otsu_binary_img = cv2.threshold(cv2.cvtColor(img,
cv2.COLOR_BGR2GRAY), 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
    opened_otsu_binary_img =
cv2.morphologyEx(otsu_binary_img, cv2.MORPH_OPEN, (7, 7))
    contours, _ = cv2.findContours(opened_otsu_binary_img,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_TC89_L1) # 找出轮廓
    if not contours: # 没有找到轮廓
        return False
    cnt_large, area = calc_max_contour_area(contours) # 找出最大轮廓
    x, y, w, h = cv2.boundingRect(cnt_large)
    dist = cv2.matchShapes(self.img_cross_standard, opened_otsu_binary_img[y:y +
h, x:x + w], cv2.CONTOURS_MATCH_I1, 0)
    if dist < 0.009 and area > 110000:
        return True
    else:
        return False

```

```

def run(self):
    while True:
        fra = my_camera.frame
        if fra is None:
            sleep(0.01)
            continue
        img = fra.copy()
        sleep(0.00001)
        if self.enable_line: self.tracking(img)
        self.block = self.block_detect(img, self.block_target_color) if self.enable_block
else None

```

```

class LogicProc:
    def __init__(self, image_proc: ImageProc):
        self.image_proc = image_proc
        self.event = None
        self.states = [ # 状态列表
            {'name': 'IDLE', 'on_enter': 'on_enter_stop'}, # 空闲状态, 启动后默认
            {'name': 'FIND_CROSS', 'on_enter': 'on_enter_find_cross'},
            {'name': 'FIND_BRIDGE', 'on_enter': 'on_enter_find_bridge'},
            {'name': 'CROSS_BRIDGE', 'on_enter': 'on_enter_cross_bridge'},
            {'name': 'FIND_DOOR', 'on_enter': 'on_enter_find_door'},
            {'name': 'THROUGH_DOOR', 'on_enter': 'on_enter_through_door'},
        ]
        self.operations = { # 各个状态主操作
            "IDLE": self.idle,
            "FIND_CROSS": self.find_cross_action,
            "FIND_BRIDGE": self.find_bridge_action,
            "CROSS_BRIDGE": self.cross_bridge_action,
            "FIND_DOOR": self.find_door_action,
            "THROUGH_DOOR": self.through_door_action,
        }
        self.transitions = [ # 状态转换关系
            # 开始寻找门
            {'trigger': 'start', 'source': 'IDLE', 'dest': 'FIND_DOOR'},
            # 找到门, 过门
            {'trigger': 'found_door', 'source': 'FIND_DOOR', 'dest':
'THROUGH_DOOR'},
            # 完成过门, 找桥
            {'trigger': 'through_door', 'source': 'THROUGH_DOOR', 'dest':
'FIND_BRIDGE'},
            # 找到桥, 过桥
            {'trigger': 'found_bridge', 'source': 'FIND_BRIDGE', 'dest':
'CROSS_BRIDGE'},
            # 完成过桥, 找 cross
            {'trigger': 'crossed_bridge', 'source': 'CROSS_BRIDGE', 'dest':
'FIND_CROSS'},
            # 找到 cross, 结束
            {'trigger': 'finished', 'source': 'FIND_CROSS', 'dest': 'IDLE'}
        ]
        self.machine = Machine(model=self,
                                states=self.states,
                                transitions=self.transitions,
                                initial='IDLE',
                                # initial="CROSS_BRIDGE",
                                send_event=True,

```

```
before_state_change=self.state_change,  
queued=True)
```

```
def state_change(self, event):  
    self.event = event
```

```
def poll(self):  
    if self.state in self.operations:  
        self.operations[self.state]()
```

```
def switch_line_tracking(self):  
    if self.image_proc.enable_line == True:  
        self.image_proc.enable_line = False  
        puppy.move_stop()  
        sleep(0.01)  
    else:  
        puppy.move_stop()  
        sleep(0.01)  
        self.image_proc.enable_line = True
```

```
def down_stair(self):  
    puppy.move_stop()  
    sleep(0.3)  
    runActionGroup('coord_down_stair')  
    sleep(6)  
    puppy.stance_config(stance=stance(0, 0, -13, 4), pitch=20 / 57.3, roll=0)  
    puppy.gait_config(overlap_time=0.1, swing_time=0.2, z_clearance=2)  
    puppy.move_stop()  
    sleep(0.2)  
    puppy.move(x=3.7, y=0, yaw_rate=0)  
    sleep(4.5)  
    puppy.stance_config(stance=stance(0, 0, -15, 2))  
    puppy.gait_config(overlap_time=0.1, swing_time=0.15, z_clearance=3)  
    puppy.move_stop()  
    sleep(0.2)
```

```
def idle(self):  
    puppy.move(x=0, y=0, yaw_rate = 0)  
    puppy.move_stop()
```

```
def on_enter_find_door(self,event):  
    puppy.stance_config(stance(0,0,-15,2), pitch = 0, roll = 0)  
    puppy.move_stop()  
    self.image_proc.block_target_color = 'blue' # 设置色块识别目标颜色  
    self.image_proc.block = None
```

```
self.image_proc.enable_block = True # 启动色块识别
print("On enter find door")
self.switch_line_tracking()
sleep(5)

def find_door_action(self):
    sleep(0.01)
    if self.image_proc.block:
        sleep((-0.000204 * self.image_proc.blue_area + 70) / 10-0.5 )
        return self.found_door()
    return

def on_enter_through_door(self,event):
    self.image_proc.enable_PWM=False
    Board.setPWMServoPulse(1, 2100, 100)
    sleep(0.1)
    self.image_proc.enable_block = False
    self.switch_line_tracking()
    puppy.stance_config(stance(0,0,-12.5,2), pitch = 0, roll = 0)# 趴下
    puppy.gait_config(overlap_time = 0.1, swing_time = 0.15, z_clearance = 2)
    self.switch_line_tracking()
    sleep(1.8)
    print("On enter through door")

def through_door_action(self):
    Board.setPWMServoPulse(1, 2100, 100)
    sleep(8)
    return self.through_door()

def on_exit_through_door(self,event):
    self.switch_line_tracking()
    puppy.stance_config(stance(0,0,-15,2), pitch = 0, roll = 0)# 过完门站起来.
    self.switch_line_tracking()
    sleep(5)
    print("On exit through door")

def on_enter_find_bridge(self,event):
    self.image_proc.block_target_color = 'yellow'
    self.image_proc.block = None
    self.image_proc.enable_block = True
    print("On enter find bridge")
    sleep(1)
```

```
def find_bridge_action(self):
    sleep(0.005)
    if self.image_proc.block:
        sleep(0.9)
        return self.found_bridge()
    return

def on_enter_cross_bridge(self,event):
    self.switch_line_tracking()
    print("On enter cross bridge")
    puppy.move(x=5, y=0, yaw_rate = 0)
    puppy.move_stop()
    sleep(0.2)
    runActionGroup('coord_up_stair_1')
    sleep(8)
    puppy.stance_config(stance=stance(0,0,-13,-4), pitch = -20/57.3, roll = 0)
    puppy.gait_config(overlap_time = 0.2, swing_time = 0.4, z_clearance = 2)
    puppy.move_stop()
    sleep(0.1)
    puppy.move(x=2.5, y=0, yaw_rate = 0)
    sleep(4.5)
    puppy.move_stop()
    sleep(0.2)
    runActionGroup('coord_up_stair_2')
    sleep(7)
    puppy.stance_config(stance=stance(0,0,-13,0), pitch = 0, roll = 0)
    puppy.gait_config(overlap_time = 0.1, swing_time = 0.2, z_clearance = 1.5)
    self.switch_line_tracking()
    sleep(5.1)

def cross_bridge_action(self):
    sleep(0.005)
    if self.image_proc.block:
        sleep(2.2)
        self.switch_line_tracking()
        self.down_stair()
        self.switch_line_tracking()
        return self.crossed_bridge()
    return

def on_enter_find_cross(self,event):
    print("On enter find cross")
    puppy.stance_config(stance=stance(0,0,-15,2), pitch = 0, roll = 0)
    puppy.gait_config(overlap_time = 0.1, swing_time = 0.15, z_clearance = 3)
    sleep(5)
    self.image_proc.block_target_color = 'black'
```

```
self.image_proc.block = None
self.image_proc.enable_block = True
```

```
def find_cross_action(self):
    sleep(0.005)
    if self.image_proc.block:
        print("find cross")
        sleep(5)
        return self.finished()
    return
```

```
def on_enter_stop(self,event):
    my_camera.camera_close()
    cv2.destroyAllWindows()
    self.switch_line_tracking()
```

```
if __name__ == '__main__':
    img_proc = ImageProc()
    logic_proc = LogicProc(img_proc)
    logic_proc.start()
    #logic_proc.crossed_bridge()
    while True:
        logic_proc.poll()
```