

南京理工大学泰州科技学院

毕业设计说明书(论文)

作 者：张明惠 学 号：1901050210

学院(系)：电子电气工程学院

专 业：电气工程及其自动化

题 目：基于 YOLO v5 的服务机器人视觉采集

处理系统设计

指导者：周军 副教授

评阅者：张顺 高级工程师

2023 年 4 月

毕业设计说明书（论文）中文摘要

随着科技的不断发展，机器人在各行各业中得到了广泛的应用。其中，服务机器人是一项具有重要意义的应用，而视觉采集与处理系统是服务机器人的核心部分。但是当前服务机器人视觉采集处理系统存在多种问题，例如：算法性能不佳，无法满足高精度定位和物体识别的要求；系统硬件资源限制，难以支持大规模图像处理；视觉采集模块稳定性差，易出现图像噪点和失真等问题。

针对以上问题，本文论述了基于 ROS 的 Gazebo 仿真平台下的服务机器人视觉采集处理系统设计。首先介绍了机器人视觉技术的研究现状和发展趋势，重点阐述了基于 YOLO v5 的目标检测模型在机器人视觉中的应用。其次，针对机器人视觉采集处理系统的设计难点，提出了本研究的解决方案。最后，通过在 Gazebo 仿真平台上的实验结果，验证了本研究所提出的解决方案的有效性。

关键词 Gazebo ROS YOLO v5 移动机器人

毕业设计说明书（论文）外文摘要

Title Design of Visual Acquisition and Processing System
 for Service Robot Based on YOLO v5

Abstract

With the continuous development of technology, robots have been widely used in various industries. Among them, service robots are an important application, and the visual acquisition and processing system is the core part of service robots. However, there are various problems with the current service robot visual acquisition and processing system, such as poor algorithm performance, inability to meet the requirements of high-precision positioning and object recognition; system hardware resource constraints, difficulty in supporting large-scale image processing; and poor stability of visual acquisition modules, easily causing image noise and distortion.

To address these issues, this paper discusses the design of a service robot visual acquisition and processing system based on the ROS Gazebo simulation platform. Firstly, it introduces the research status and development trend of robot vision technology, and focuses on the application of YOLO v5-based object detection model in robot vision. Secondly, it proposes solutions to the design difficulties of the robot visual acquisition and processing system. Finally, through experiments on the Gazebo simulation platform, the effectiveness of the proposed solutions is verified.

Keywords Gazebo ROS YOLO v5 Mobile robot

目 录

1	引言	1
1.1	研究背景及意义	1
1.2	国内外研究现状	1
1.3	本章小结	3
2	硬件系统设计	4
2.1	引言	4
2.2	机器人底盘硬件设计	4
2.3	传感器设计	5
2.4	本章小结	7
3	软件系统设计	8
3.1	ROS 框架与工具介绍	8
3.2	视觉采集模块设计	9
3.3	图像处理模块设计	10
3.4	本章小结	10
4	算法介绍	12
4.1	引言	12
4.2	YOLO v5	12
4.3	YOLO v5 展示效果与分析	13
4.4	本章小结	13
5	仿真实验测试与分析	15
5.1	仿真实验概述	15
5.2	机器人模型构建与仿真环境搭建	15
5.3	实验过程	16
5.4	实验结果分析	19
5.5	本章小结	19
	结束语	20
	致谢	21
	参考文献	22
	附录 A 源程序(摘要)	23
	附录 B 在学期间取得的成果	34

1 引言

1.1 研究背景及意义

视觉识别（Visual recognition）是计算机视觉领域的重要研究方向之一，其主要探究如何使计算机像人类一样通过视觉信息进行认知和理解。随着计算机视觉和深度学习技术的不断发展，视觉识别已广泛应用于许多领域，如安防、自动驾驶、医疗、智能家居等，具有重要的研究背景和意义。

视觉识别技术的应用范围非常广泛，如人脸识别、物体检测、场景理解等，在安全、智能化等领域的应用需求迫使视觉识别技术的不断发展^[1]。同时随着计算机处理能力的提升、传感器的发展、大数据的爆发性增长，越来越多的视觉信息得以产生和收集，推动了视觉识别技术的快速发展和广泛应用。视觉识别作为人工智能的重要组成部分，对人工智能的发展和进步发挥着关键性的作用，从而实现自动化、智能化等自主人工思维的发展。视觉识别领域是计算机视觉、模式识别、机器学习等多个学科的交叉领域，视觉识别技术研究涉及到图像处理、数学理论、机器学习、人工智能等领域，对于提高科学研究中的技术和理论水平，具有重要的意义。随着视觉识别技术的不断发展和应用，所带来的经济效益也越来越明显，如通行证监测、商品广告识别、人脸支付等等。这些新兴商业机会使得视觉识别成为了一个非常热门的领域。

1.2 国内外研究现状

1.2.1 视觉识别发展现状

目前，视觉识别技术已经得到了广泛应用和快速发展，取得了一系列重要进展。首先，深度学习算法的出现推动了视觉识别技术的革命性发展。卷积神经网络（CNN）等深度学习模型具有强大的特征提取和分类能力，可以有效地解决图像识别、目标检测、场景理解等问题。例如，ImageNet 比赛中使用的神经网络模型在 2012 年的错误率为 26.2%，而到了 2015 年，该错误率下降至 3.6%^[2]。其次，大规模数据集的建立也极大地促进了视觉识别技术的发展。开放图像数据库（Open Image）、COCO、ImageNet 等公开的数据集成为了深度学习算法训练和评估的基础，同时也促进了跨领域的研究合作和知识分享。

1.2.2 视觉识别算法研究现状

目前，视觉识别算法研究已经走过了漫长的历程，进展显著。深度学习算法是目前视觉识别算法中应用最广泛和最成功的方法。卷积神经网络（CNN）是其中最具有代表性的方法之一^[3]，它可以有效地解决图像分类、目标检测、语义分割等问题，并在大型数据集上取得了优秀的性能^[4]。目标检测是计算机视觉领域中的重要问题之一。近年来，基于深度学习的目标检测算法不断涌现，如 Faster R-CNN、YOLO、SSD 等。这些算法不断提高检测速度和准确率，同时也拓宽了目标检测的应用领域。人脸识别是一种广泛应用于人身份验证和安全访问控制的技术。基于深度学习的人脸识别算法已经取得了很好的效果，如 Eacelet 和 DeepID，等算法，在关键数据集上的准确率已经超过了类。行为识别是指从视频或图像序列中自动检测和预测人的行为。近年来，基于深度学习的行为识别算法不断涌现，如 LRCN、LSTM 等^[5]，这些算法可以有效地识别复杂的人类行为，并在社交媒体分析、智能监控等领域具有广泛应用。

1.2.3 视觉识别硬件发展现状

随着视觉识别算法的发展，硬件设备的性能和能力不断提高。当前视觉识别硬件发展的现状包括 GPU、FPGA、ASIC 和 CPU。GPU 具有强大的并行计算能力，是广泛采用的硬件平台之一；FPGA 灵活性高、功耗低，被越来越多的研究人员应用于视觉识别领域；ASIC 具有高速处理和低能耗的特点，企业如谷歌和华为正开发适用于视觉识别应用的 ASIC 芯片^[6]；虽然 CPU 的并行计算能力不及 GPU 和 FPGA，但它具有通用性和灵活性，在一些复杂计算任务中具有实用价值。各种硬件平台都有其优点和适用场景，越来越多的企业和研究机构也开始将定制化的硬件加速引入到视觉识别领域，以满足不同应用场景的需求。

1.2.4 YOLO 算法发展现状

自 2016 年提出以来，YOLO (You Only Look Once) 算法已经得到了广泛应用和持续的改进。目前，YOLO 算法发展的一些现状包括以下几个版本：YOLOv2、YOLOv3、YOLOv4、YOLOv5 以及后续版本^[7]。YOLOv2 在 YOLOv1 的基础上引入了 Batch Normalization 和 Anchor Boxes 等技术来提高检测精度和稳定性；YOLOv3 在 YOLOv2 的基础上进一步加强了网络结构，使用了更多的卷积层和

跨层特征连接等技术，并采用了多尺度训练和测试策略来大幅提升检测精度和速度；YOLOv4 针对目标检测任务中的一系列挑战，采用了更加先进的技术来提高检测精度和速度，如 CSPDarknet53 骨干网络、SPP-block、SAM 和 PAN 等；YOLOv5 利用了 AutoML 技术，自动搜索并优化了检测网络结构和超参数，同时引入了 Swish 激活函数和 Bag of Freebies 等新技术，实现了更快的检测速度，而仍然保持高精度。YOLO 算法已成为目标检测领域的重要方法之一，广泛应用于自动驾驶、安防、人脸识别等领域。未来，随着深度学习技术的不断发展，YOLO 算法仍将有更大的发展空间和潜力。

1.3 本章小结

视觉识别算法已经取得了显著的进展和成果，其中以深度学习算法和 YOLO 算法为代表的技术，为视觉识别的快速发展和广泛应用奠定了基础，同时硬件设备的不断进步也为视觉识别技术提供了有力的支持和保障。未来，随着视觉识别技术的继续发展和完善，它将在更多领域展示出无限的潜力和价值。

2 硬件系统设计

2.1 引言

全向移动机器人因其卓越的运动性能和精确的运动精度而在室内场景中得到广泛应用。本章中，我们介绍了基于 YOLO v5 的服务机器人模型，构建了相应的系统框架，并详细介绍了系统中的硬件部分。硬件层主要由机器人机械平台、控制单元以及外围模块等部分构成。机械平台采用了麦克纳姆轮，具备全向移动能力。平台上搭载基于 STM32F407 的控制单元，与仿真机器人通过 Gazebo 平台进行通信。在电脑主机上安装 Ubuntu18.04 系统，在 ROS 操作系统下实现视觉信息的采集和处理。STM32F407 是控制机器人底盘运动的核心部件，接收上位机决策信息生成电机驱动信号，从而实现平台的运动。

2.2 机器人底盘硬件设计

2.2.1 底盘核心控制板

机器人底盘核心控制板采用大疆公司推出的 RoboMaster 开发板，这款开发板是一款高性能移动机器人开发板，搭载了 STM32F407 芯片，主频 168MHz，RAM 容量为 192KB，Flash 存储容量为 1MB^[8]。此外，该开发板还装配了 MPU6050 六轴陀螺仪加速度计，可实现机器人的精确姿态检测。此外，RoboMaster 开发板还提供了多种接口，如串口、I2C、SPI、CAN 等，支持各种外部设备的接入和数据交互。除此之外，该开发板还支持 11.1V 锂电池供电，并搭载电量传感器和充电管理系统，能够有效监测电池电量并进行充电管理。

2.2.2 底盘运动学模型

麦克纳姆轮最是一种特殊的轮子，常用于移动机器人和车辆。麦克纳姆轮的独特之处在于它由 4 个 45 度的钢制分片构成，这种构造让麦克纳姆轮可以在平面上自由移动，并且具备全向移动的特性。每个分片都由独立的电机控制，通过控制电机的旋转方向和速度，可以控制机器人的运动。相比于普通轮子，麦克纳姆轮在平面内可自由移动，实现类似球一样的运动并可向任意方向移动和旋转。同时，麦克纳姆轮的全向运动特性也可以大大提高机器人在某些应用场景中的工作效率。图 2.1 所示为麦克纳姆轮不同运动状态下的示意图

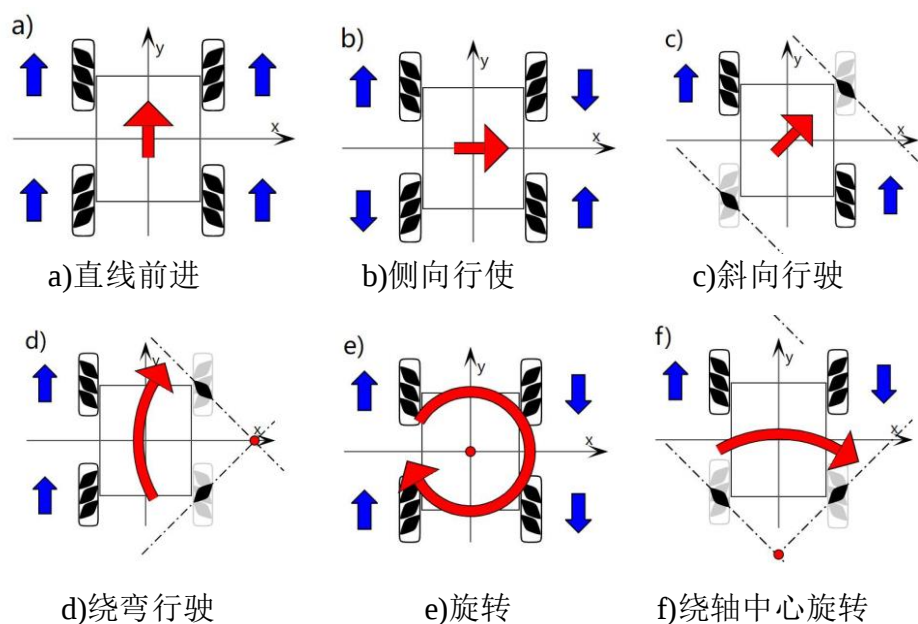


图 2.1 麦克纳姆轮不同运动状态下的示意图

2.2.3 计算机处理模块

该计算机搭载 INTEL I7-9750H 中央处理器、GTX1650Ti 图形处理器及 1TB 固态硬盘，为机器人产生的海量数据提供高效的计算平台，能够大幅提升数据处理和深度学习速度，从而更好地完成复杂任务。

2.3 传感器设计

2.3.1 Intel RealSense D435 深度摄像头

英特尔公司的 RealSense D435 摄像头是一款带深度感知功能的 RGB-D 相机，适用于许多应用领域，如 3D 扫描、虚拟现实、增强现实、机器人视觉等。它由一对立体红外传感器(Stereo IR Pair),一个红外激光发射(IRProjector) 和 RGB Camera 组成。RGB 摄像头分辨率达 2000 万像素，3D 传感器可以 30 帧/秒的速度提供分辨率高达 1280×720 , 或者以 90 帧/秒的速度提供 848×480 的较低分辨率。深度距离在 0.1 m~10 m 之间，视场角度为 85×58 度。D435 配备全局快门和宽视野，最小感测深度约为 0.11m，并具有远距离（10 米）功能以及高达 1280×720 的深度分辨率，非常适合用于在快速移动的 VR 或机器人应用中捕获深度数据。

原理介绍：该设备的左右两个红外相机接收到的红外光产生了两幅灰度图像，即左侧和右侧的 IR 灰度图像。中间的红外点阵发射器作为红外照射光源，

相当于一个补光灯，用以产生点云信息。该设备中的红外点阵发射器强度可调，可以控制是否发射，不打开时仍能测得深度，但效果略逊一筹。最右边的 RGB 相机用于采集彩色图像，最终可将彩色视频流与深度流进行对齐。该设备的测距原理与一般的双目相机相同，其原理基于左右图像的视差来计算距离。但与普通的彩色 RGB 相机不同的是，红外 IR 相机是用来接收目标返回的红外光信号的，从而得到左右两幅 IR 灰度图像。在室内环境中，即使关闭了灯光进入黑暗的环境，红外 IR 相机仍可以生成深度图像，但质量可能会有所下降。如图 2.2 所示，该设备正面的四个摄像头从左至右分别为：左红外相机、红外点阵投影仪、右红外相机、RGB 相机（前三个用于生成深度图像，最后一个用于生成彩色 RGB 图像）。



图 2.2 RealSense D435

RealSense D435 摄像头具备以下主要特点和优势：

高精度深度感知：RealSense D435 摄像头采用基于双目立体视觉技术的深度感知方案，可实现高精度的深度信息获取，并支持宽动态范围和高分辨率。**强大的自适应性能：**该摄像头支持自适应红外投射和环境光抑制等技术，能够在不同光照条件下保持稳定，并可在室内、室外等多种场景下使用。**灵活的软件支持：**RealSense D435 摄像头提供了丰富的软件开发工具(SDK)^[9]，包括 C++、Python、Unity 等多种编程语言的接口和示例程序，方便用户快速上手和开发应用。**友好的价格：**相比市面上其他深度相机，RealSense D435 摄像头的价格相对较低，可满足中小型项目的需求。

2.3.2 IMU

本设计使用了 MPU6050 六轴陀螺仪加速度计，如图 2.3 所示。IMU 即惯性测量单元（Inertial Measurement Unit），是一种将多个惯性传感器集成于一体的装置，它能够在三维空间内测量物体的加速度和角速度，从而计算物体的运动状态和姿态。

IMU 通常由加速度计、陀螺仪、磁力计等传感器组成，这些传感器能够利用

不同的物理原理来测量物体的加速度、角速度和磁场强度，进而反映物体的运动

状态和方向信息。



图 2.3 MPU6050

2.3.3 激光雷达

本设计使用了思岚 A1 雷达，如图 2.4 所示。激光雷达是一种利用激光光束来探测目标物体位置、距离和其他属性的设备。该设备通过发射脉冲激光光束，测量激光光束与目标物体之间的时间和空间信息，计算目标物体与激光雷达之间的距离和位置。

激光雷达可通过扫描方式或固定照射方式进行测量，通常使用旋转扫描或水平线扫描方式。旋转扫描可实现全方位的测量，而水平线扫描常用于垂直方向的测量。激光雷达通常包括光学器件、控制电路和激光器等组件。光学器件使用透镜或反射镜对光束进行处理，在激光器发射高强度的单色激光光束的同时，控制电路对光束进行精确的调节。



图 2.4 思岚 A1 雷达

2.4 本章小结

本章介绍了基于 YOLO v5 的服务机器人的建模和系统框架，以及系统中的硬件部分，包括运动底盘、控制单元和外围模块。针对机器人底盘硬件设计，介绍了底盘核心控制板和计算机处理模块。在传感器设计方面，详细介绍了采用的 Intel RealSense D435 深度摄像头、MPU6050 六轴陀螺仪加速度计和思岚 A1 激光雷达。这些硬件设备的选择和设计为机器人的运动性能和视觉感知能力提供了稳定可靠的支持。

3 软件系统设计

3.1 ROS 框架与工具介绍

ROS (Robot Operating System) 是一种用于编写机器人软件的开源操作系统。它提供了一组跨平台的工具、库和功能包，可以方便地构建机器人应用。ROS 最初由斯坦福大学人工智能实验室于 2007 年开发，现在由 Open Robotics 维护。ROS 广泛应用于机器人研究、开发和实践，包括视觉感知、运动控制、导航、SLAM 等多个领域。ROS 是以 C++ 和 Python 为主要编程语言的，同时也支持其他语言。

3.1.1 Gazebo

Gazebo 是一款开源的多机器人仿真器，它可以在三维环境中模拟机器人运动、感知、控制和交互等过程，软件界面如图 3.1 所示。Gazebo 最初由 USC 研究小组开发，现在已经成为 ROS 的官方仿真器，得到了广泛的应用。Gazebo 支持多种物理引擎，包括 ODE、Bullet、Simbody 等，可以模拟机器人在不同环境下的运动和行。此外，Gazebo 还支持 ROS 和 ROS2，提供了丰富的 ROS 接口，可以方便地与 ROS 应用集成。Gazebo 的特点是开源、灵活、可扩展和高度可定制化，适合用于机器人仿真、开发和测试等应用场景。

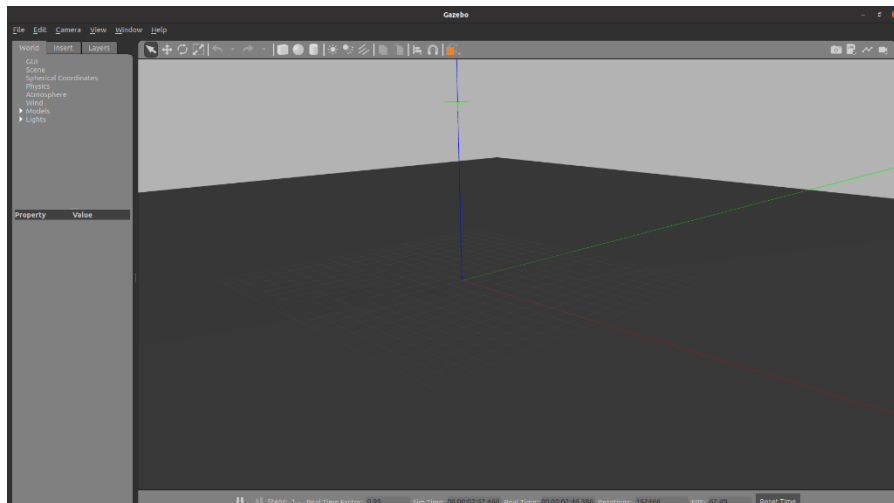


图 3.1 Gazebo

3.1.2 RViz

RViz 是 ROS 中的一款可视化工具，可以实时显示机器人的各种传感器数据、运动状态和环境信息等，软件界面如图 3.2 所示。RViz 支持多种可视化方式，

比如三维模型、图表、轨迹等，可以方便地观察机器人在空间中的运动、感知和交互行为。RViz 还支持 ROS 系统的各种消息类型，包括点云、激光雷达、摄像头等，可以轻松地将它们可视化。此外，RViz 还提供了一些交互工具，比如选择器、测距工具等，可以帮助用户更方便地与可视化对象进行交互。RViz 的特点是强大、灵活、可扩展和高度可定制化，适合用于机器人仿真、开发和测试等应用场景。

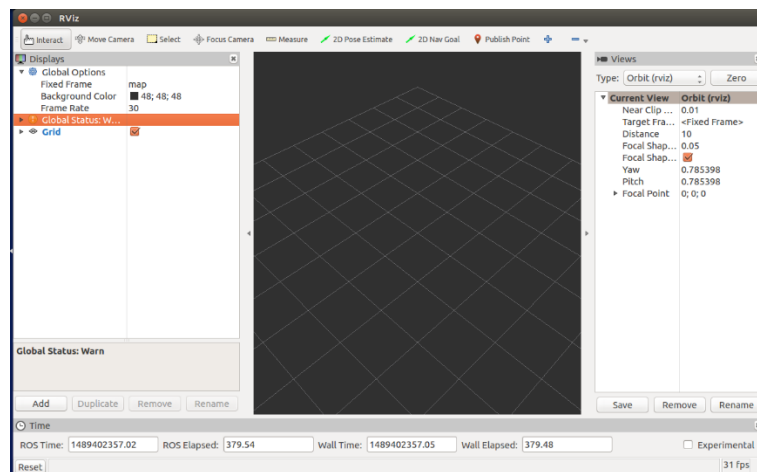


图 3.2 Rviz

3.2 视觉采集模块设计

视觉采集模块是机器人感知系统的重要组成部分，用于采集机器人所处环境的视觉信息。以下是视觉采集模块的设计步骤。

确定采集器类型：根据具体的应用场景和需求，选择合适的视觉采集器，包括普通相机、深度相机、红外相机、光学测距仪等。不同的采集器类型有不同的技术特点和采集性能，需要进行全面的分析和比较。**确定采集器参数^[10]：**针对不同的采集器类型，确定采集器的相关参数，包括分辨率、视野角度、光圈、曝光时间、帧率等。这些参数会直接影响到采集效果和速度，需要充分考虑应用需求和采集器性能。**设计采集器安装方式：**根据具体的机器人结构和应用需求，设计合适的采集器安装方式，包括直接安装、底座安装、云台安装、附加装置安装等。同时需要考虑采集器与机器人其他部件的协调性和空间布局。**设计采集器控制方案：**设计采集器的控制方案，包括硬件控制和软件控制两部分。硬件控制主要是指采集器与机器人控制板的连接，需要选择合适的接口和通信协议。软件控制主要是指驱动程序和数据采集程序的编写，需要充分利用机器人操作系统和相

关软件包的接口和工具。测试和调试：设计完成后需要进行实验测试和调试，包括采集效果测试、数据传输测试、控制程序测试等。通过测试结果可以对采集器的设计和调试进行优化和改进。

视觉采集模块的设计需要充分考虑应用需求和采集器性能，进行综合分析和比较，最终设计出合适的采集器和控制方案，同时进行实验测试和调试，确保采集效果和稳定性。

3.3 图像处理模块设计

图像处理模块是机器人视觉系统的重要组成部分，用于对采集到的视觉信息进行预处理和特征提取等操作。以下是图像处理模块的设计步骤：

确定图像处理算法：根据应用需求和采集器类型，选择适当的图像处理算法，包括基础的图像处理算法（比如滤波、边缘检测等）和高级的视觉算法（比如目标检测、物体跟踪等）。不同的应用场景需要针对性地选择合适的图像处理算法。设计图像处理流程：根据选择的图像处理算法，设计合适的图像处理流程，包括图像输入、图像预处理、特征提取等步骤。同时需要考虑图像处理流程与机器人其他组件之间的协调性和数据传输方式。选择图像处理工具和库：根据图像处理算法和流程的设计，选择适合的图像处理工具和库，包括 OpenCV、PCL、Matlab 等。这些工具和库可以方便地实现常见的图像处理算法和数据处理功能，同时也提供了一些工具函数和接口，方便与机器人操作系统和其他模块的集成。设计图像处理控制程序：根据图像处理流程和选择的工具和库，设计图像处理控制程序，包括图像输入控制、数据处理流程控制、输出结果控制等。程序应该具备良好的鲁棒性和性能，保证图像处理的效率和准确性。测试和调试：设计完成后需要进行实验测试和调试，包括图像处理效果测试、处理效率测试、程序稳定性测试等。通过测试结果可以对图像处理算法和流程进行优化和改进，保证系统的稳定性和性能。

图像处理模块的设计需要充分考虑应用需求和采集器性能，选择合适的图像处理算法和工具，设计合适的图像处理流程和控制程序，同时进行实验测试和调试，保证图像处理的效率和准确性。

3.4 本章小结

本章主要介绍了软件系统设计中的 ROS 框架和工具，包括 ROS 介绍、Gazebo

和 RViz。此外，本章还介绍了视觉采集模块和图像处理模块的设计步骤。其中，视觉采集模块主要包括选择采集器类型、确定采集器参数、设计采集器安装方式、设计采集器控制方案、测试和调试等步骤；而图像处理模块主要包括选择图像处理算法、设计图像处理流程、选择图像处理工具和库、设计图像处理控制程序、测试和调试等步骤。这些步骤都需要充分考虑具体应用场景和需求，选择合适的工具和算法，进行综合分析和比较，最终设计出稳定、高效、可靠的软件系统。

4 算法介绍

4.1 引言

目标检测是计算机视觉领域的重要问题之一，而 YOLO v5 (You Only Look Once, Version 5) 则是目前最先进的目标检测算法之一。该算法是由 Joseph Redmon 与 Alexey Bochkovskiy 共同开发，相比前一代算法有着显著的改进，具有更高的检测精度、更快的速度以及更小的模型体积。

YOLO v5 算法采用了许多领先的目标检测算法技术，并且提出了一些创新性的思想和方法。本章将详细介绍 YOLO v5 算法以及在图像检测中的应用。

4.2 YOLO v5

YOLO v5 是目标检测领域中的一种深度学习算法，也是 YOLO 系列的最新一代算法。相较于 YOLO v1、v2、v3，YOLO v5 在检测速度、精度和模型大小等方面都得到了显著的提升^[11]。以下是 YOLO v5 算法的一些特点：

YOLO v5 借鉴了 YOLO v4 的一些技术，例如 CSPResNeXt 网络结构、SPP 网络结构和 PANet 特征金字塔等，并吸收了其他 SOTA 算法的优秀思想和技术。YOLO v5 支持多尺度检测，可以更好地适应不同尺度和不同分辨率的输入图像。同时，它采用了卷积核大小可变的方法，提升了对小目标的检测能力。YOLO v5 相比于 YOLO v4 使用了更小的模型，且在加入剪枝技术后，模型大小仅为 4.4MB。这使得其在资源受限的环境下具有更好的可用性^[13]。YOLOv5 具有更高的检测速度，可达到每秒 140 帧。此外，相比于 YOLOv4，其权重文件数据量仅为其九分之一，结构更加小巧^[12]。如图 4.1 所示 YOLOv5 的网络结构包括输入端、Backbone、Neck 和 Prediction，这些组件共同协作完成目标检测任务。

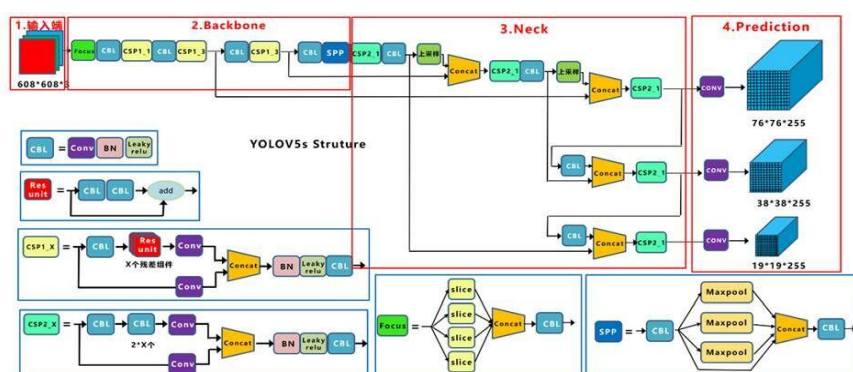


图 4.1 YOLO v5 网络结构

总之, YOLO v5 是目标检测领域的一款优秀算法, 具有诸多优点, 包括高精度、快速、轻量化、易用性等, 可以应用于多种场景下的目标检测任务。YOLO v5 是一种高效的目标检测算法, 具有比较好的性能和速度, 同时也拥有许多不同的模型配置和参数。

4.3 YOLOv5 效果展示与分析

YOLOv5s: 其具有最高的速度和模型大小, 适用于运行速度比准确性更重要的实时应用程序^[14]。

YOLOv5m: 在 s 和 l 之间权衡, 具有比 s 更好的精度, 同时还具有较快的速度和较小的模型大小。

YOLOv5l: 其具有最强大的精度, 并比 s 和 m 使用更广泛, 可用于精度要求较高的应用程序。

YOLOv5x: 其具有最高的准确性, 但速度较慢, 适用于精确性要求比速度更重要的场景。对比图如图 4.2 所示。

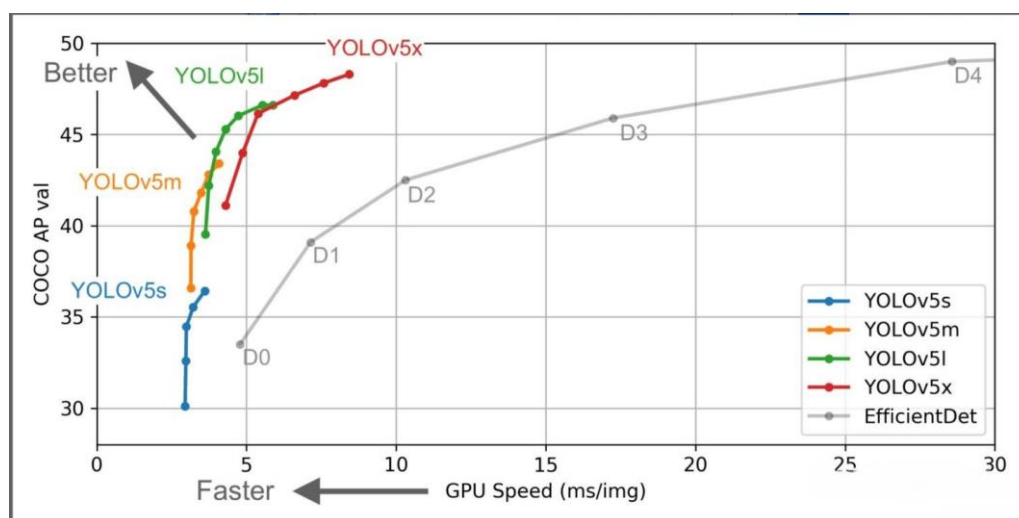


图 4.2 不同模型性能对比图

4.4 本章小结

本章主要介绍了目标检测算法中的 YOLO v5 算法, 包括其特点、网络结构和应用场景。YOLO v5 是深度学习领域的一个先进算法, 相比于前一代算法有着显著的提升, 包括更高的检测精度、更快的速度和更小的模型体积等。YOLO v5 采用了许多领先的目标检测算法技术, 包括 CSPResNeXt 网络结构、SPP 网络结构、PANet 特征金字塔等, 并在多尺度检测、卷积核大小可变、自适应锚框

计算与自适应图片缩放等方面进行了创新性的思考和方法^[15]。本章还展示了不同版本的 YOLO v5 的效果展示与分析，包括 YOLOv5s、YOLOv5m、YOLOv5l 和 YOLOv5x 四个版本，分别针对实时应用程序、精度和速度平衡、精度要求较高和精确性要求较高的场景进行适用。YOLO v5 是一种高效的检测算法，具有比较好的性能和速度，可以应用于多种场景下的目标检测任务。

5 仿真实验测试与分析

5.1 仿真实验概述

借助 ROS 机器人操作平台,利用其分布式架构,并结合其分布-订阅式通信框架,可以分别处理各个功能模块。且 ROS 提供大量的开源包以及工具,可以便携的完成我们所需任务。

5.2 机器人模型构建与仿真环境搭建

为实现机器人能够很好的还原现实环境状况,通过 URDF 描述文件构建出小车物理模型,并借助 GAZEBO 仿真模拟软件加载 URDF 小车模型,尽可能的提供较为真实的物理环境。如图 5.1 所示为仿真机器人。

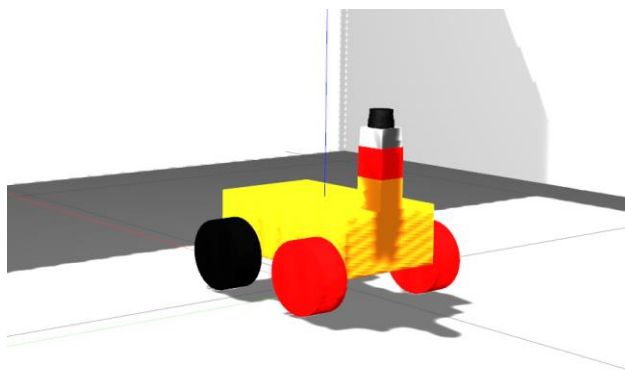


图 5.1 仿真机器人

5.2.1 机器人 URDF 描述文件

为了在 ROS 中描述小车的运动模型及传感器配置,需要使用 URDF 描述文件来描述。该文件可以描述小车的运动构建、关节联动、传感器配置等方面。本项目需要构建的是四轮全向轮运动模型,并搭载摄像头、IMU、激光雷达等传感器。

在 URDF 文件中,可以描述机器人的外观和物理属性。其中,机器人的尺寸和形状可以通过<link>元素来表示,<geometry>元素则用于描述部件的形状和大小,<color>元素则用于设置部件的颜色。

此外,机器人的物理参数也可通过 URDF 文件进行描述。例如,<collision>元素可用于描述机器人碰撞属性,<mass>元素则用于描述机器人的质量大小,<inertial>元素用于描述机器人的惯性特性。可以通过添加描述文件来加载各类传

传感器的相关信息，使机器人在 ROS 中具备更完善和智能化的功能。

5.2.2 RViz 与 gazebo 仿真环境

借助 RViz 可视化显示工具我们可以清晰地观察到车辆在静态环境中的各个模型以及传感器数据。然而，在动态物理场景下，机器人的表现是更具挑战性的，这时我们就需要利用 Gazebo 进行物理仿真了。Gazebo 是 ROS 操作系统下的一种三维物理仿真平台，它独具工程上的应用性和科学性，提供了强大的物理引擎和各种物理参数，例如地面、阳光以及各种物理模型。通过加载机器人 URDF 描述文件，我们可以在 Gazebo 中进行机器人物理仿真，如图 5.2 所示。

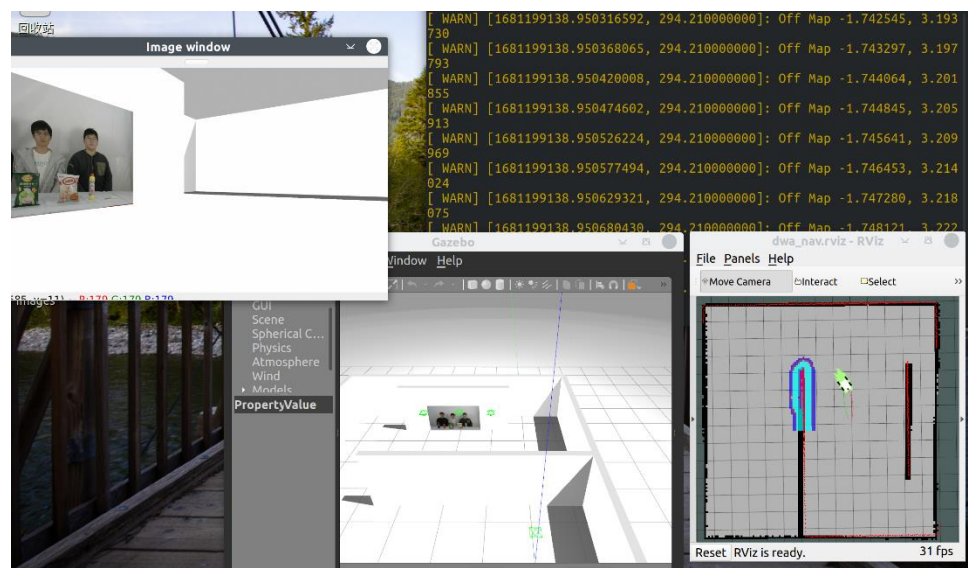


图 5.2 RViz 与 Gazebo 仿真环境

5.3 实验过程

根据摄像头摄取到的图像，使用 YOLOv5 网络模型对其进行目标检测，流程图如 5.3 所示。

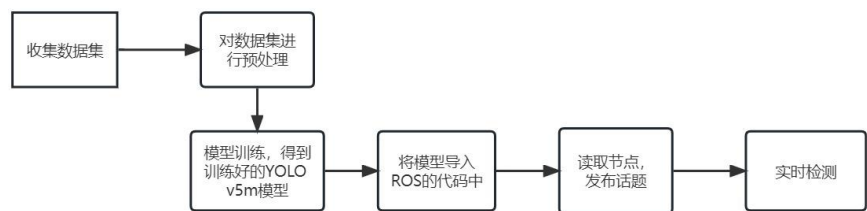


图 5.3 流程图

首先收集大量数据集，创建数据集目录如图 5.4 所示。YOLO v5 所要求的数据集目录结构包含了图片数据和对应的标签文件，各个目录应当严格按照要求

命名,以保证训练过程的正确进行。其中,“dataset/”是数据集所在的目录,“train/”和“val/”是用于训练和验证的样本集,“images/”目录下存储着图片数据,每张图片需要对应一个同名的 txt 标签文件,“labels/”目录下存储着 txt 格式的标签文件,每个 txt 文件对应一张同名的输入图片,标签文件的内容格式应符合 YOLOv5 算法的要求。

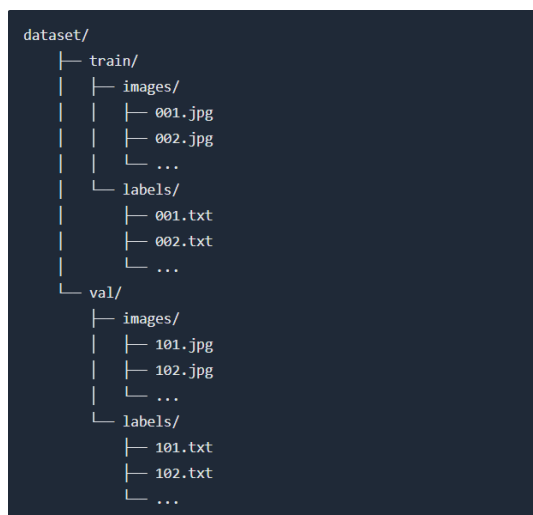


图 5.4 数据集目录

具体的,标签文件应当以以下格式呈现:每个物体一行,每行都有 5 个值,分别代表标签类别和 4 个 bbox 坐标 (x,y,w,h)。标签文件中的坐标值应当被缩放到 0-1 的范围之内,这么做是为了与图片在训练前统一尺寸。最后需要创建一个 dataset.yaml 文件,放在与 data 文件夹同一层。

制作好数据集之后,进行模型训练,模型训练过程如图 5.5 所示:

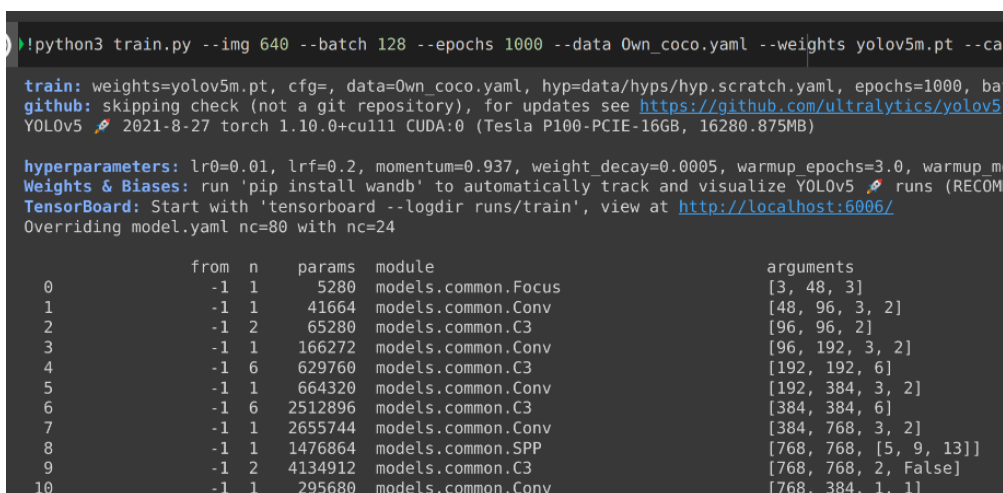


图 5.5 模型训练过程

其中`img-size`、`batch-size`、`epochs`需要与`dataset.yaml`中的定义保持

一致。`cfg`参数表示指定使用的 YOLOv5 模型，`weights`用于恢复之前训练的权重，如果没有预训练权重则填写空字符。`name`参数指定训练结果输出的文件夹名。训练过程可能需要几个小时甚至更长时间，视数据集大小和计算机效能而定。实验参数如表 5.1 所示。

表 5.1 主要参数配置表

参数	设置
nc（类别数）	9
names（类别名）	James, Steven, Paul, paper...
imgsz（图片大小）	640
Batch size	128
epoch（训练周期）	1000
weights（预训练模型）	YOLOv5m.pt
device（cuda 设备）	11.2

将训练好的 yolov5m.pt 模型导入到 ROS 的代码中,可以使用 pytorch 提供的 torch.load 函数进行模型的加载。接着，使用 ROS 的图像读取节点实时获取摄像头的数据，并在 ROS 的图像发布节点中配备预处理和目标检测脚本，实时地获取图像数据并对其中的目标进行检测。实现流程如下：

在 ROS 中新建一个图像读取节点，使用 ROS 库订阅摄像头数据流，提取图像中的像素信息。写一个 ROS 的发布节点，用于发送检测后的结果作为图像或 ROS 消息。在发布节点中引入预处理和目标检测的脚本，并根据需求对检测结果进行处理（如在图像中绘制目标检测框，输出 ROS 消息等）。将预处理和目标检测的脚本与模型文件组合起来，编写 ROS 服务节点，用于调用目标检测算法进行检测。

在服务节点中使用 Pytorch 的 load 函数加载要使用的模型（如 yolov5m.pt）。在服务节点中编写使用模型进行检测和返回结果的代码。例如，可以使用模型来预测输入图像中的目标位置、类别等信息，然后将这些信息封装成为 ROS 消息或者绘制到图像中并发送到发布节点。通过以上步骤，可以实现在 ROS 机器人系统中的目标检测，并通过可视化方式展示检测结果如图 5.6 所示。由图 5.6 分析可

知本次实验的检测效果较好，精度较高。

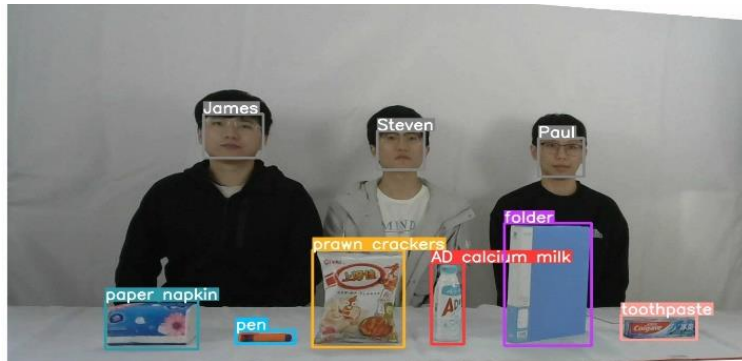


图 5.6 检测结果

5.4 实验结果分析

精确率 (precision) 和召回率 (recall) 的值都与测试集中预测结果为正确的正样本的数量有关，分别为预测结果为正确的正样本数量除以被模型预测为正样本的数量或实际为正样本的数量。训练周期为 1000 时，Precision、Recall 和 mAP 的变化曲线如图 5.7、5.8、5.9 所示。

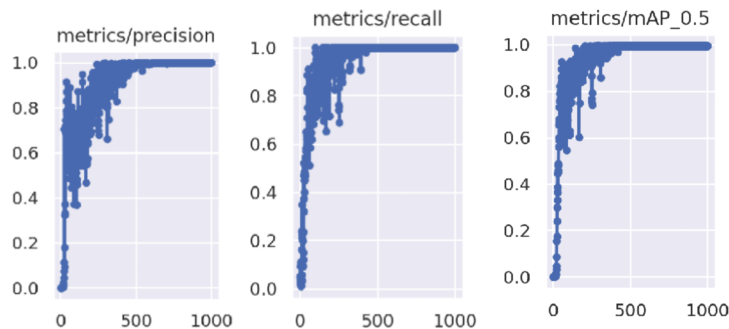


图 5.7 Precision 变化曲线 图 5.8 Recall 变化曲线 图 5.9 mAP 变化曲线

5.5 本章小结

本章介绍了机器人仿真实验测试与分析的流程。首先，通过 ROS 操作平台构建机器人模型，搭建机器人的物理环境。接着，通过 RViz 和 Gazebo 等工具进行仿真实验，模拟机器人的物理运动和传感器数据的采集，进行目标检测等实验。在实验过程中，收集了大量的数据集，并应用 YOLOv5 网络模型进行物体检测，得出了实验检测结果。最后通过精确率、召回率以及 mAP 等指标对实验结果进行了分析和评估。通过本章的实验，不仅验证了 ROS 平台的实用性和高效性，还验证了本设计的可靠性。

结束语

在本设计中，成功地构建了一个在 ROS 平台下基于麦克纳姆轮的仿真移动机器人，并在其中增加了摄像头、IMU 和激光雷达等多种传感器，实现了目标检测功能。在机器人模型构建阶段，我们通过 URDF 描述文件描述了机器人的外观和物理属性，然后通过 Gazebo 仿真环境模拟出机器人在物理场景下的表现。在实验过程中，我们使用 YOLOv5 网络模型对摄像头拍摄的图像进行目标检测，并通过 ROS 发布节点实时获取检测结果，再通过 RViz 工具将检测结果可视化展示。实验结果表明，我们的目标检测效果良好，精度较高，为后续机器人应用提供了一定的技术支持。此外，本实验还验证了在 ROS 机器人操作平台下结合开源包及工具的优越性，未来针对不同应用场景可进行更加深入的研究。

致谢

时间如水，岁月无情。转眼间，我即将结束四年的大学生涯。回首过去的三年，不仅有欢声笑语、汗水和收获，也有失落和不易。然而，感恩之情在心头盈满。

大一过后我便加入了学校无人机实验室，那里浓厚的科学研究与技术研发氛围让我深深喜爱科学研究与技术创新。有了学校的资源支持，我的个人能力得到了很大的提升。在此，我要衷心感谢实验室的刘小军老师、周军老师和刘增元老师，他们在机器人领域的软硬件技术方面拥有深入的研究，他们的创新思想让我们受益匪浅。

感谢实验室的同学，包括张富高、陈梓标、宋涛等，他们关心和帮助让我渡过了充实而快乐的时光。

对于我的家人，特别是那些默默支持我的人们，我想说一声谢谢。你们的爱是我永远的财富。

最后，我特别感谢那些在百忙之中审阅论文和参加答辩的各位老师。对于所有曾经帮助过我的老师、同学和朋友，我想表达最诚挚的谢意。

参 考 文 献

- [1] 牛洪超,白松,胡晓兵. 基于改进 YOLOv5 的视觉定位算法[J]. 计算机工程与设计, 2022.
- [2] 周传鑫,徐昊成,杜鹏飞,等. 基于视觉识别的自动采摘装置[J]. 智能城市, 2022.
- [3] 刘亮. 机器人视觉识别抓取物件[J]. 中小企业管理与科技, 2020.
- [4] 高云峰,吴秀芬. 服务机器人视觉系统模块化研究综述[J]. 机械设计与制造, 2010.
- [5] 朱木健. 基于服务机器人视觉系统的设计[D]. 重庆:重庆大学, 2006.
- [6] 郭磊,王邱龙,薛伟,等. 基于改进 YOLOv5 的小目标检测算法[J]. 电子科技大学学报, 2022.
- [7] 方永锋. 基于多轴无人机 RealSense 立体视觉技术空间测量的研究[J]. 电力系统装备, 2021.
- [8] 张月文,李松恒,张炜,等. 基于机器视觉的可回收垃圾智能分拣系统设计[J]. 实验室研究与探索, 2022.
- [9] 万旭,徐海黎,阮有兵,等. 基于 Intel Realsense 深度摄像头的巡逻机器人避障导航方法[J]. 电视技术, 2018.
- [10] 刘亚凡. 基于 RGB-D 相机的移动机器人视觉 SLAM 技术研究[D]. 北京:北京交通大学, 2020.
- [11] 蒲良,张学军. 基于深度学习的无人机视觉目标检测与跟踪[J]. 北京航空航天大学学报, 2022.
- [12] 张润,王永滨. 机器学习及其算法和发展研究[J]. 中国传媒大学学报（自然科学版）, 2016.
- [13] 杨观赐,胡丙齐,蒋亚汶,等. 智能家居服务机器人视觉和听觉行为隐私识别保护与度量技术[J]. 贵州大学学报（自然科学版）, 2021.
- [14] 汤辰,万衡,王凯凯. 移动机器人检测识别技术的研究[J]. 电气自动化, 2015.
- [15] YANG, KAILUN, WANG, KAIWEI, ZHAO, XIANGDONG, et al. IR stereo RealSense: Decreasing minimum range of navigational assistance for visually impaired individuals[J]. Journal of ambient intelligence and smart environments, 2017.

附录 A 源程序（摘要）

A.1 视觉识别启动文件

```
# YOLOv5  by Ultralytics, GPL-3.0 license
"""
Run inference on images, videos, directories, streams, etc.
Usage:
    $ python path/to/detect.py --source path/to/img.jpg --weights yolov5s.pt --img 640
"""

import argparse
import sys
import time
from pathlib import Path
import cv2
import numpy as np
import torch
import torch.backends.cudnn as cudnn
FILE = Path(__file__).absolute()
sys.path.append(FILE.parents[0].as_posix()) # add yolov5/ to path
from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, check_requirements, check_imshow, colorstr,
non_max_suppression, \
    apply_classifier, scale_coords, xyxy2xywh, strip_optimizer, set_logging,
increment_path, save_one_box
from utils.plots import colors, plot_one_box
```

```
from utils.torch_utils import select_device, load_classifier, time_sync
@torch.no_grad()
def run(weights='yolov5s.pt', # model.pt path(s)
        source='data/images', # file/dir/URL/glob, 0 for webcam
        imgsz=640, # inference size (pixels)
        conf_thres=0.25, # confidence threshold
        iou_thres=0.45, # NMS IOU threshold
        max_det=1000, # maximum detections per image
        device="", # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False, # show results
        save_txt=False, # save results to *.txt
        save_conf=False, # save confidences in --save-txt labels
        save_crop=False, # save cropped prediction boxes
        nosave=False, # do not save images/videos
        classes=None, # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False, # class-agnostic NMS
        augment=False, # augmented inference
        visualize=False, # visualize features
        update=False, # update all models
        project='runs/detect', # save results to project/name
        name='exp', # save results to project/name
        exist_ok=False, # existing project/name ok, do not increment
        line_thickness=3, # bounding box thickness (pixels)
        hide_labels=False, # hide labels
        hide_conf=False, # hide confidences
        half=False, # use FP16 half-precision inference
    ):
    save_img = not nosave and not source.endswith('.txt') # save inference images
    webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
        ('rtsp://', 'rtmp://', 'http://', 'https://'))
```

```
# Directories/路径

save_dir = increment_path(Path(project) / name, exist_ok=exist_ok) # increment
run/增量运行

(save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True) #
make dir/创建文件

# Initialize/初始化

set_logging()

device = select_device(device)

half &= device.type != 'cpu' # half precision only supported on CUDA/CUDA 仅支
持半精度

# Load model/导入模型

w = weights[0] if isinstance(weights, list) else weights

classify, suffix = False, Path(w).suffix.lower()

pt, onnx, tflite, pb, saved_model = (suffix == x for x in ['.pt', '.onnx', '.tflite', '.pb', ''])

# backend/后端

stride, names = 64, [f'class{i}' for i in range(1000)] # assign defaults/指定默认值

if pt:

    model = attempt_load(weights, map_location=device) # load FP32 model

    stride = int(model.stride.max()) # model stride

    names = model.module.names if hasattr(model, 'module') else model.names #
get class names

    if half:

        model.half() # to FP16

    if classify: # second-stage classifier

        modelc = load_classifier(name='resnet50', n=2) # initialize

        modelc.load_state_dict(torch.load('resnet50.pt',
map_location=device)['model']).to(device).eval()

    elif onnx:

        check_requirements(('onnx', 'onnxruntime'))

        import onnxruntime
```

```
session = onnxruntime.InferenceSession(w, None)

else: # TensorFlow models

    check_requirements(('tensorflow>=2.4.1',))

    import tensorflow as tf

    if pb: # https://www.tensorflow.org/guide/migrate#a_graphpb_or_graphpbtxt

        def wrap_frozen_graph(gd, inputs, outputs):

            x = tf.compat.v1.wrap_function(lambda:

tf.compat.v1.import_graph_def(gd, name=''), []) # wrapped import

            return x.prune(tf.nest.map_structure(x.graph.as_graph_element,

inputs),

                        tf.nest.map_structure(x.graph.as_graph_element,

outputs))

        graph_def = tf.Graph().as_graph_def()

        graph_def.ParseFromString(open(w, 'rb').read())

        frozen_func = wrap_frozen_graph(gd=graph_def, inputs="x:0",

outputs="Identity:0")

    elif saved_model:

        model = tf.keras.models.load_model(w)

    elif tflite:

        interpreter = tf.lite.Interpreter(model_path=w) # load TFLite model

        interpreter.allocate_tensors() # allocate

        input_details = interpreter.get_input_details() # inputs

        output_details = interpreter.get_output_details() # outputs

        int8 = input_details[0]['dtype'] == np.uint8 # is TFLite quantized uint8

model

imgsz = check_img_size(imgsz, s=stride) # check image size

# Dataloader/数据加载器

if webcam:

    view_img = check_imshow()
```

```
    cudnn.benchmark = True # set True to speed up constant image size inference/
    设置为 True 可加快恒定图像大小推断

    dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt)

    bs = len(dataset) # batch_size/批量大小

else:

    dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt)

    bs = 1 # batch_size/批量大小

vid_path, vid_writer = [None] * bs, [None] * bs

# Run inference/运行推理

if pt and device.type != 'cpu':

    model(torch.zeros(1, 3, *imgsz).to(device).type_as(next(model.parameters())))

# run once

t0 = time.time()

for path, img, im0s, vid_cap in dataset:

    if onnx:

        img = img.astype('float32')

    else:

        img = torch.from_numpy(img).to(device)

        img = img.half() if half else img.float() # uint8 to fp16/32

    img = img / 255.0 # 0 - 255 to 0.0 - 1.0

    if len(img.shape) == 3:

        img = img[None] # expand for batch dim

    # Inference

    t1 = time_sync()

    if pt:

        visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if

visualize else False

    pred = model(img, augment=augment, visualize=visualize)[0]

    elif onnx:

        pred = torch.tensor(session.run([session.get_outputs()[0].name],
```

```
{session.get_inputs()[0].name: img}))

else: # tensorflow model (tflite, pb, saved_model)

    imn = img.permute(0, 2, 3, 1).cpu().numpy() # image in numpy

    if pb:

        pred = frozen_func(x=tf.constant(imn)).numpy()

    elif saved_model:

        pred = model(imn, training=False).numpy()

    elif tflite:

        if int8:

            scale, zero_point = input_details[0]['quantization']

            imn = (imn / scale + zero_point).astype(np.uint8) # de-scale

            interpreter.set_tensor(input_details[0]['index'], imn)

            interpreter.invoke()

            pred = interpreter.get_tensor(output_details[0]['index'])

            if int8:

                scale, zero_point = output_details[0]['quantization']

                pred = (pred.astype(np.float32) - zero_point) * scale # re-scale

            pred[..., 0] *= imgsz[1] # x

            pred[..., 1] *= imgsz[0] # y

            pred[..., 2] *= imgsz[1] # w

            pred[..., 3] *= imgsz[0] # h

            pred = torch.tensor(pred)

        # NMS

        pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)

        t2 = time_sync()

        # Second-stage classifier (optional)

        if classify:

            pred = apply_classifier(pred, modelc, img, im0s)

    # Process predictions
```

```
for i, det in enumerate(pred): # detections per image
    if webcam: # batch_size >= 1
        p, s, im0, frame = path[i], f'{i}: ', im0s[i].copy(), dataset.count
    else:
        p, s, im0, frame = path, '', im0s.copy(), getattr(dataset, 'frame', 0)
    p = Path(p) # to Path
    save_path = str(save_dir / p.name) # img.jpg
    txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image'
else f'_{frame}') # img.txt
    s += '%gx%g ' % img.shape[2:] # print string
    gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh
    imc = im0.copy() if save_crop else im0 # for save_crop
    if len(det):
        # Rescale boxes from img_size to im0 size
        det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()
        # Print results
        for c in det[:, -1].unique():
            n = (det[:, -1] == c).sum() # detections per class
            s += f'{n} {names[int(c)]}' + ('s' * (n > 1)), " # add to string
        # Write results
        for *xyxy, conf, cls in reversed(det):
            if save_txt: # Write to file
                xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) /
gn).view(-1).tolist() # normalized xywh
                line = (cls, *xywh, conf) if save_conf else (cls, *xywh) #
label format
                with open(txt_path + '.txt', 'a') as f:
                    f.write((' %g ' * len(line)).rstrip() % line + '\n')
            if save_img or save_crop or view_img: # Add bbox to image
                c = int(cls) # integer class
```

```
label = None if hide_labels else (names[c] if hide_conf else f'{names[c]}
{conf:.2f}')

im0 = plot_one_box(xyxy, im0, label=label, color=colors(c,
True), line_width=line_thickness)

if save_crop:
    save_one_box(xyxy, imc, file=save_dir / 'crops' /
names[c] / f'{p.stem}.jpg', BGR=True)

# Print time (inference + NMS)
print(f'{s}Done. ({t2 - t1:.3f}s)')

# Stream results
if view_img:
    cv2.imshow(str(p), im0)
    cv2.waitKey(1) # 1 millisecond

# Save results (image with detections)
if save_img:
    if dataset.mode == 'image':
        cv2.imwrite(save_path, im0)
    else: # 'video' or 'stream'
        if vid_path[i] != save_path: # new video
            vid_path[i] = save_path
        if isinstance(vid_writer[i], cv2.VideoWriter):
            vid_writer[i].release() # release previous video writer
        if vid_cap: # video
            fps = vid_cap.get(cv2.CAP_PROP_FPS)
            w =
            =
            int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
            h =
            =
            int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        else: # stream
            fps, w, h = 30, im0.shape[1], im0.shape[0]
```

```
        save_path += '.mp4'
        vid_writer[i] = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
        vid_writer[i].write(im0)

    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir /
'labels'}" if save_txt else "
        print(f"Results saved to {colorstr('bold', save_dir)}{s}")

    if update:
        strip_optimizer(weights)  # update model (to fix SourceChangeWarning)

    print(f'Done. ({time.time() - t0:.3f}s)')

def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str,
default='/media/nonono/90DCB3C5DCB3A3BE/yolov5_zmh/yolov5-master/yolov5s.pt1',
help='model.pt path(s)')
    parser.add_argument('--source', type=str, default='0', help='file/dir/URL/glob, 0 for
webcam')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640],
help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence
threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU
threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections
per image')
    parser.add_argument('--device', default="", help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in
```

```
--save-txt labels')

    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction
boxes')

    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')

    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or
--class 0 2 3')

    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')

    parser.add_argument('--augment', action='store_true', help='augmented inference')

    parser.add_argument('--visualize', action='store_true', help='visualize features')

    parser.add_argument('--update', action='store_true', help='update all models')

    parser.add_argument('--project', default='runs/detect', help='save results to
project/name')

    parser.add_argument('--name', default='exp', help='save results to project/name')

    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok,
do not increment')

    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box
thickness (pixels)')

    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide
labels')

    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide
confidences')

    parser.add_argument('--half', action='store_true', help='use FP16 half-precision
inference')

    opt = parser.parse_args()

    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1 # expand

    return opt

def main(opt):

    print(colorstr('detect: ') + ', '.join(f'{k}={v}' for k, v in vars(opt).items()))

    check_requirements(exclude=('tensorboard', 'thop'))

    run(**vars(opt))
```

```
if __name__ == "__main__":
```

```
    opt = parse_opt()
```

```
    main(opt)
```

附录 B 在学期期间取得的成果

- (1) 2020 年 11 月，获第十七届江苏省高等数学竞赛本科二级 一等奖
- (2) 2020 年 12 月，获第十二届全国大学生数学竞赛（非数学类） 三等奖
- (3) 2021 年 6 月，获第十八届江苏省高等数学竞赛本科二级 二等奖
- (4) 2021 年 6 月，获第十六届全国大学生智能车竞赛 三等奖
- (5) 2021 年 6 月，获中国机器人大赛 无人机挑战赛-无人机实物赛 二等奖
- (6) 2021 年 11 月，获中国机器人技能大赛 机器人猜词项目 冠军（一等奖）
- (7) 2021 年 11 月，获第十二届江苏省大学生机器人大赛 视觉识别 二等奖（季军）