智能配货搬运车

崔芳铭；鲁金东

第一部分 设计概述

1.1 设计目的

伴随着年轻人生活习惯的改变以及大型超市的发展，外卖行业迎来了蓬勃发展，这对大型商超仓库的配货速度提出了新的挑战。传统的利用人工逐一拣配货的方式效率相对低下，逐渐成为影响制约配货速度的瓶颈。针对这一现象，参赛团队成员基于海思 Taurus & Pegasus 套件，设计了一款基于深度学习的全向移动自主抓取的智能配货搬运车。团队认为，在该机器的帮助下可大幅提升各种仓储配货速度，起到提升效率、节约成本的目的。

1.2 应用领域

本系统可以实现物料运输的自动化，提升内部物流速度，将员工从重复性和危险性的运输工作中解放出来，可广泛应用于物流仓储、特/危品运输、各种制造业及医疗/服务行业中。

1.3 主要技术特点

本系统可以在不依赖其他传感器，只使用惯性导航系统以及视觉识别系统的条件下，完成对目标商品识别、抓取以及放置到配货区域的功能。另外，本系统接入了华为云，实现远程操作与监控。从功能上划分，系统主要有以下几个方面的特点：

AI 智能识别与分类：我们使用 Taurus 套件集成的摄像头采集环境数据，并通过基于 YOLOv2 的卷积神经网络完成对商品的识别，并提取位置信息。进一步通过使用 RESNET18 卷积神经网络完成商品分类。为了提高运行速度，在视频采集过程使用 VPSS 帧进行预处理，并送至 NNIE 进行加速推理。

基于华为云端的远程操控：利用 Pegasus 强大的物联网功能，接入华为云，实现远程可视化操作，以及历史信息的监控。

优化的惯性导航算法：在 Pegasus 上高频运行惯性导航算法，实现对位置的精确控制，配合视觉系统可以对累计误差进行消除，保证对抓取以及放置位置的精确定位。

高效的机械臂逆解算算法：在 Pegasus 上运行机械臂逆解算法，与底盘配合实现对机械臂位置的控制操作。

1.4 关键性能指标

目标检测部分： 通过训练，模型在测试数据集下，拥有超过 95%的识别准确率。受单图商品数量的影响，模型召回率在 65% 到 80% 之间波动。在 3m*2m 的抓取区域内能有效识别出商品种类，并给出精确的定位坐标。

底盘部分：底盘尺寸 300mm*300mm，最大直线移动速度 4m/s，最大旋转速度 3rad/s。采用 gmr 高精度编码电机，mpu9250 陀螺仪。通过 Pegasus 对电机数据以及陀螺仪数据的读取，进行频率为 100hz 的积分，实现了惯性导航算法。

机械臂部分：底座采用闭环进电机控制，可以胜任高精度抓取任务。机械臂大臂小臂以及机械手采用 25kg 舵机，分辨率为 0.22 度，满足机械臂的精度要求。机械手采用可形变设计，使用限位开关检测形变程度，可以设定抓取力矩。使用电滑环实现无限圈数旋转，保证连续抓取的灵活性。

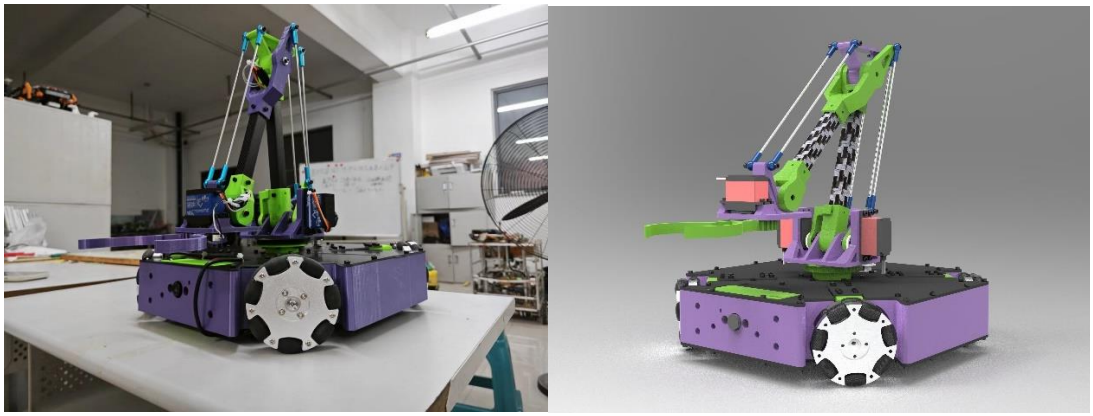


图 1-2 实物图以及渲染图

1.5 主要创新点

功能创新点：

（1）基于深度学习的单目测距算法

在 Taurus 套件的帮助下，图像信号通过 yolov2 卷积神经网络完成目标检测后，通过“相似三角形”法对目标检测框的面积进行计算，并对输出值进行距离映射后，得到目标距离，实现单目测距，系统误差小于 8mm。

（2）多数据融合的导航算法

单目测距结合惯性导航系统，完成目标相对待命区域的世界坐标结算，配合机械臂逆解算法实现对目标的精确抓取。Pegasus 以 100hz 的频率与电机控制模块进行通讯，实时获取陀螺仪数据与电机数据，进行角度以及位置的积分运算，经过卡尔曼滤波算法可以精确定位小车位置。

（3）优化的 YOLOv2 算法：

为了将高性能的卷积神经网络应用到低成本的边缘计算领域，使用海思嵌入式平台提供的神经网络加速器，部署了经过多次训练和优化的模型，实现了嵌入式平台级功耗下的高精度识别多层网络的部署。我们充分利用 Taurus 套件的硬件特性与功能，在板端完成本地的图像 捕获、处理、识别与判决，在 1GB 的有限运行存储中谨慎控制各程序的内存占用，并实现了抓取小车的定位 商品种类和位置识别。

（4）定制舵机驱动板以及滑环系统

为了使机械臂更具灵活性，采用定制的电滑环实现机械臂 360 度无圈数限制的转动。将 Pegasus 上的拓展板替换为我们自己绘制的 PCB，实现对舵机的额外供电，防止舵机抓取商品时电流过大损坏 Pegasus 开发板。

第二部分 系统组成及功能说明

2.1 整体介绍

本系统可以完成对目标商品识别、抓取以及放置到配货区域的功能。如图 2-1 所示，整个系统可分为五部分：Taurus 开发板、Pegasus 开发板、底盘控制驱动板、三轴机械臂、全向移动底盘。

Taurus 开发板对视频流进行抽帧并将得到的图像输入到优化的 YOLOv2 卷积神经网络完成对商品的识别，并提取商品位置信息。进一步通过使用 RESNET18 卷积神经网络完成商品分类。分类后通过基于深度学习的单目测距算法计算目标商品的坐标，通过串口发送给 Pegasus 开发板。

Pegasus 开发板接收到信息后对数据进行解析，将发送的坐标转化为原点为待命区的世界坐标。将解算后的速度数据发给底盘控制驱动板，同时对底盘驱动板的反馈数据进行处理，计算当前位置，控制全向移动底盘移动至目标坐标，到达指定坐标后控制机械臂进行自主抓取。整个系统的数据流程如图 2-2 所示。

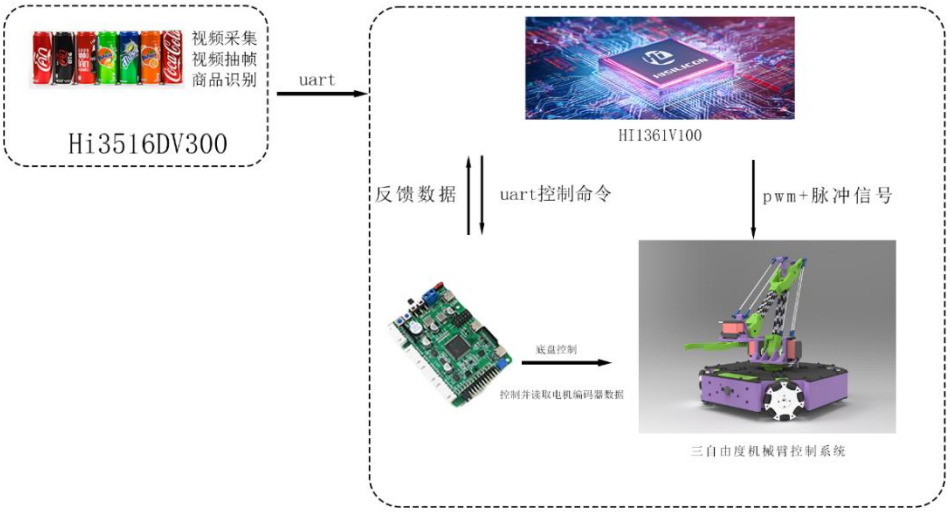


图 2-1 深度学习驱动的基于商品识别的智能配货搬运车结构示意图

2.2 各模块介绍

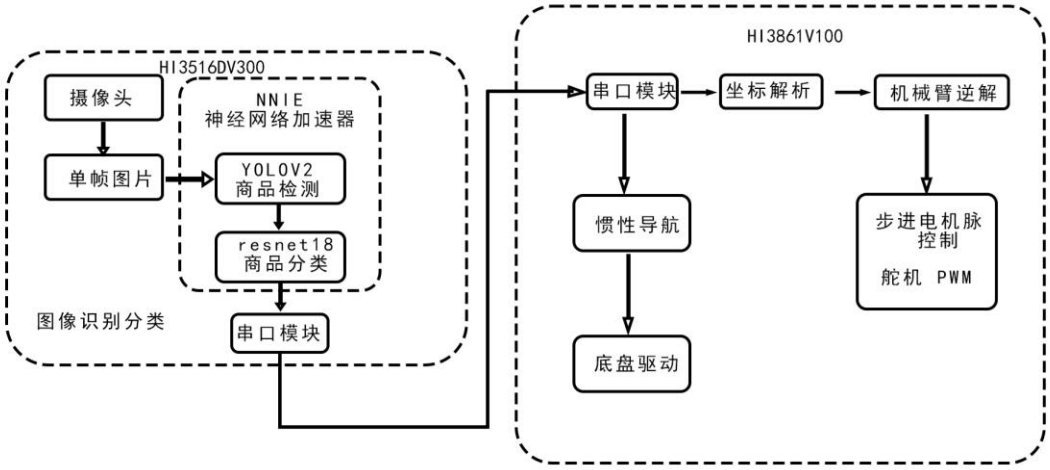


图 2-2 深度学习驱动的基于商品识别的智能配货搬运车系统工作流程

（1）Taurus 套件

Taurus 开发板通过 IMX335 摄像头采集外界视觉信息，从视频流中以接近 30fps 的速率采样出单帧图片，将图片传入由 NNIE 单元加速的 YOLOv2 商品检测网络中，完成对商品的识别以及坐标的计算，在 RESNET18 中完成对商品的分类。一方面将经过浅红色方框标注的图像输出至套件中的 LCD 显示屏，同时通过串口将坐标信息发送至 Pegasus 开发板。(由于空间受限，调试时使用 LCD 显示屏，实际整机运行时 LCD 显示屏未安装)

（2）Pegasus 套件

Pegasus 开发板通过 WIFI 模块接入华为云，从而接收用户的控制指令并反馈系统自身状态。收到控制指令后 Pegasus 开发板对指令进行解析，解析后把商品名称发送给 Taurus。Taurus 对相应的商品进行识别并进行坐标转换。Pegasus 收到信息后对坐标进行转换，转换后得到对底盘以及机械臂的控制信息。将底盘的控制信息转化成串口通讯数据发送到电机控制板上对电机进行控制。通过机械臂逆解算法得到每个舵机的角度以及底座步进电机的角度，最终转化为 PWM 信号控制舵机以及步进电机从而实现对商品的抓取操作。

（3）供电单元

舵机供电模块采用两块 DC-DC 降压模块，最大输出电流 3A，留出较大的冗余防止抓取较重的商品时降压模块过热。Pegasus 采用一块 5V DC-DC 降压模块供电，最大输出电流为 2A，可以满足 Pegasus 的稳定运行。Taurus 供电采用底盘

电机控制板的板载 type-c 接口，最大输出电流 3A。

（4）机械臂

机械臂采用 12 路电滑环以及自主设计的转接板连接旋转部分与固定部分的供电线和信号线。12 路电滑环，每路额定电流为 2A。取出 6 路为供电线，3 路为舵机信号线，以及三路拓展接口。使用其中一路拓展接口进行抓取力矩检测。

第三部分 完成情况及性能参数

深度学习驱动的基于商品识别以及坐标计算与校正系统可以与智能配货搬运车配合实现目标商品的抓取与配货功能。通过摄像头识别目标商品的位置，与配货小车进行通讯进行坐标的转换，实现对目标商品的自主抓取。以下是系统各部分的细节。

3.1 商品检测与分类模型训练部署过程

（1）卷积神经网络的训练与转换

我们使用海思提供的服务器完成对卷积神经网络参数的训练。考虑到在 NNIE 单元中的算法兼容性，我们选择 YOLOv2 网络实现目标检测模型的搭建。使用 labelme 进行数据标注，并使用 Python 脚本将标注好的数据转换成 Darknet 支持的数据格式。我们借助 Darknet2caffe 工具，将训练得到的.weights 权重文件首先转换成.caffemodel 与.prototxt 文件，并将.prototxt 文件中不适合 NNIE 加速的内容剔除。修改 input_dim 的格式，再通过华为海思提供的 Ruyi Studio 平台参照指导文档将 Caffe 的网络文件转换成 Taurus 板端可用的.wk 模型文件。悬链结果如图 3-1 所示。

```
Region Avg IOU: 0.896137, Class: 0.999983, Obj: 0.867855, No Obj: 0.007899, Avg Recall: 1.000000,  count: 19
19351: 0.797872, 0.803090 avg, 0.010000 rate, 0.158313 seconds, 928848 images
Loaded: 0.000173 seconds
Region Avg IOU: 0.833884, Class: 0.941167, Obj: 0.830406, No Obj: 0.007620, Avg Recall: 0.941176,  count: 17
Region Avg IOU: 0.908747, Class: 0.999860, Obj: 0.918357, No Obj: 0.009000, Avg Recall: 1.000000,  count: 19
Region Avg IOU: 0.912931, Class: 0.999991, Obj: 0.926015, No Obj: 0.006690, Avg Recall: 1.000000,  count: 16
Region Avg IOU: 0.910849, Class: 0.999993, Obj: 0.919557, No Obj: 0.006124, Avg Recall: 1.000000,  count: 14
Region Avg IOU: 0.898901, Class: 0.999975, Obj: 0.874561, No Obj: 0.008849, Avg Recall: 1.000000,  count: 18
Region Avg IOU: 0.905905, Class: 0.999991, Obj: 0.891686, No Obj: 0.006564, Avg Recall: 1.000000,  count: 14
Region Avg IOU: 0.899999, Class: 0.999997, Obj: 0.919059, No Obj: 0.006605, Avg Recall: 1.000000,  count: 16
Region Avg IOU: 0.919906, Class: 0.999994, Obj: 0.891865, No Obj: 0.006959, Avg Recall: 1.000000,  count: 16
19352: 0.841555, 0.806937 avg, 0.010000 rate, 0.161292 seconds, 928896 images
Loaded: 0.000215 seconds
Region Avg IOU: 0.901086, Class: 0.999952, Obj: 0.907765, No Obj: 0.006577, Avg Recall: 1.000000,  count: 11
Region Avg IOU: 0.902652, Class: 0.999968, Obj: 0.914267, No Obj: 0.008397, Avg Recall: 1.000000,  count: 18
Region Avg IOU: 0.887223, Class: 0.999988, Obj: 0.885201, No Obj: 0.006816, Avg Recall: 1.000000,  count: 17
Region Avg IOU: 0.864271, Class: 0.999946, Obj: 0.834550, No Obj: 0.008103, Avg Recall: 1.000000,  count: 21
Region Avg IOU: 0.912344, Class: 0.999967, Obj: 0.900572, No Obj: 0.008963, Avg Recall: 1.000000,  count: 21
Region Avg IOU: 0.907277, Class: 0.999995, Obj: 0.919992, No Obj: 0.005885, Avg Recall: 1.000000,  count: 13
Region Avg IOU: 0.917845, Class: 0.999966, Obj: 0.912938, No Obj: 0.008019, Avg Recall: 1.000000,  count: 16
Region Avg IOU: 0.897058, Class: 0.999978, Obj: 0.910482, No Obj: 0.006238, Avg Recall: 1.000000,  count: 14
19353: 0.958283, 0.822071 avg, 0.010000 rate, 0.157955 seconds, 928944 images
Loaded: 0.000234 seconds
```

图 3-1 用于目标检测的 YOLOv2 网络参数训练过程

（2）分类网络 RESNET18 的训练与转换

我们使用海思提供的服务器完成了分类网络参数的训练。首先参考官方文档进行数据集制作，清洗和训练之后通过华为海思提供的 Ruyi Studio 平台，参照指导文档将 Caffe 的网络文件转换成 Taurus 板端可用的.wk 模型文件。分类网络参数的训练结果如图 3-2 所示。

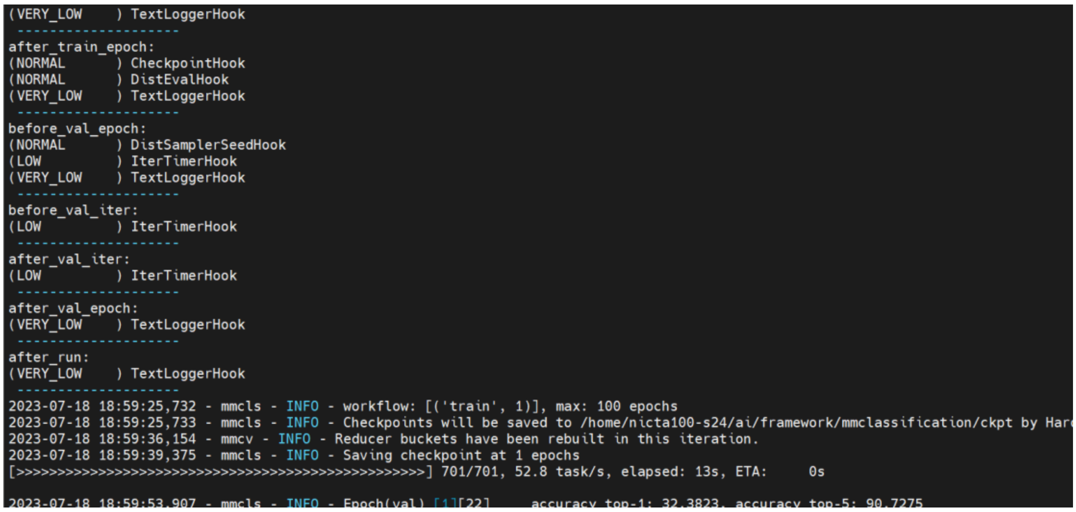


图 3-2 目标分类网络 RESNET18 的训练过程

3.2 Pegasus 拓展板与滑环转接板电路设计

机械臂抓取重物时峰值电流可达到 3.6A，单个舵机的堵转电流为 2.5A。为了驱动大功率的舵机，系统采用额外的的供电模块，确保 Pegasus 的安全以及稳定输出。我们设计了 Pegasus 拓展板实现机械臂的无干涉转动。为了更好的配合电滑环，我们还设计了滑环转接板，如图 3-3 及图 3-4 所示。

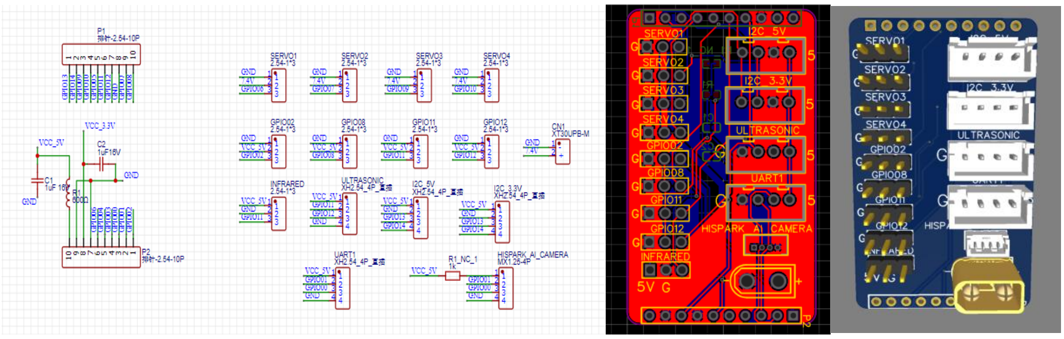


图 3-3 Pegasus 拓展板

根据图纸，我们进行了 Pegasus 拓展板 PCB 的打样，经过测试可以非常稳定的驱动舵机长时间运行，并且发热控制非常好，几乎感觉不到降压模块的升温。

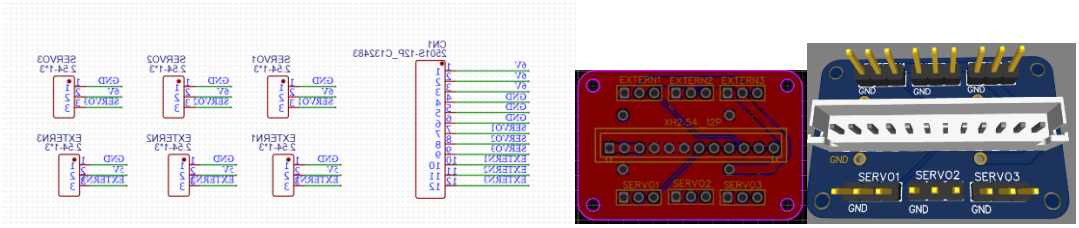


图 3-4 滑环转接板 PCB 设计

3.3 智能配货搬运车整体结构

我们按照功能将系统分为两个平台：物联网平台与视觉平台。Pegasus 作为物联网平台负责与网络通信以及控制各种模块；Taurus 作为视觉平台负责提供各信息给 Pegasus。

物联网平台可细分为 4 个主要部分：Pegasus 开发板、底盘控制驱动板、机械臂、复位系统。通过自主建模设计，并采用 3D 打印以及 CNC 加工组装，完成集成度较高的智能搬运配货车机械结构。

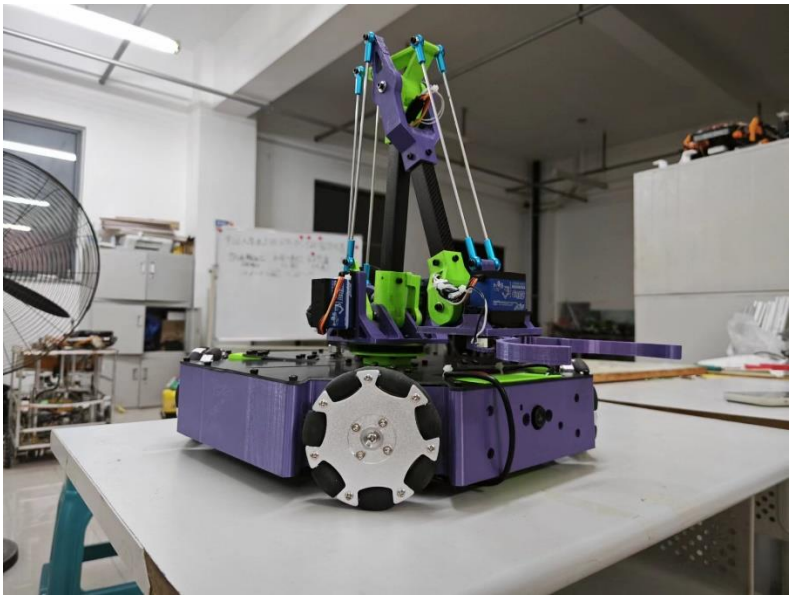


图 3-5 智能配货搬运车整体结构

图 3-5 给出智能搬运配货车的 45 度角实拍图。其中 Taurus 内嵌在前挡板中，整个系统电路内嵌在底盘，底盘拆解图如图 3-6 所示.

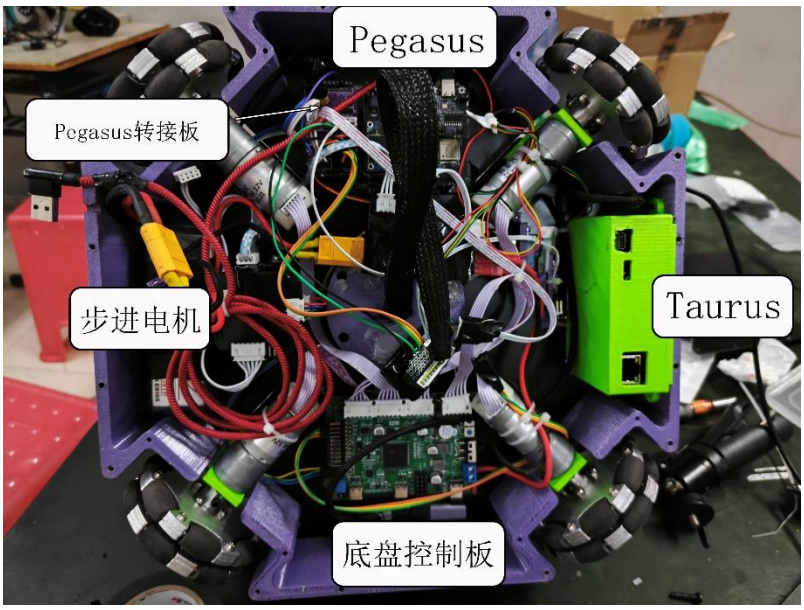


图 3-6 底盘拆解图

底盘内部结构如图 3-6 所示。所有供电系统集成在一张主板上，该主板集成了 5V、6V 以及防电流反涌电路。Taurus 布置在底盘前方，2200mah 航模电池以及闭环步进电机布置在底盘后方，Pegasus 和底盘控制板分别布置在左右两侧。中间布置了供电板和电滑环。整个底盘采用高度集成化的设计，充分利用底盘空间。

4、最终实现效果

通过各模块协同工作，我们成功实现了对商品的自主抓取与配货功能，并成功进行了系统测试。

测试流程如下。首先设置好 WIFI，给机器总开关上电，Pegasus 会自动连接华为云，收到下发的指令后，系统会将相应的商品名称发给 Taurus，并接收 Taurus 发来的坐标进行转化进而控制舵机自主完成抓取任务。

图 3-7 展示了华为云可视化平台的操作界面。通过该平台，可以实现对系统的远程操控。搬运车启动并完成自检后，可以实时接收华为云下发的指令并同时将自身的数据上报。同时云平台也可以备份搬运车的历史状态数据，方便操作者回溯。

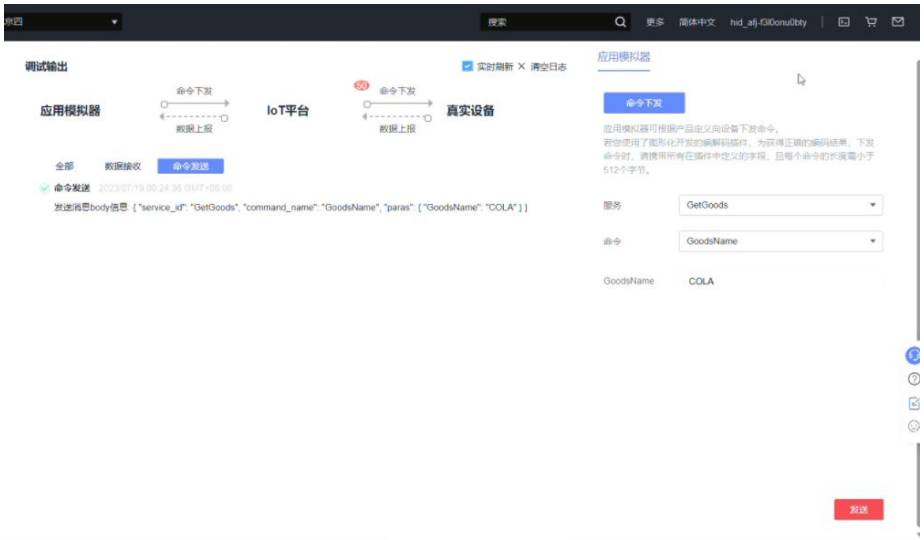


图 3-7 华为云指令下发

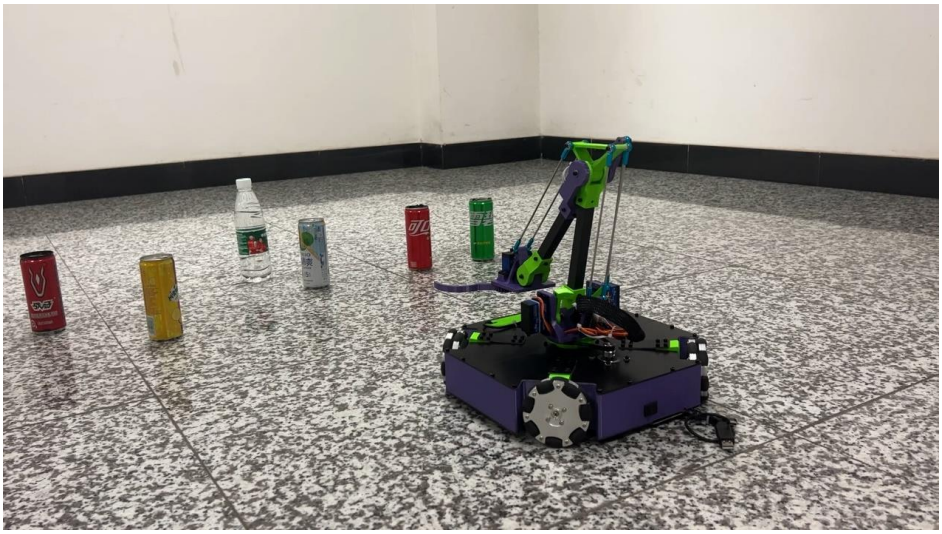


图 3-8 搬运车执行下发的指令

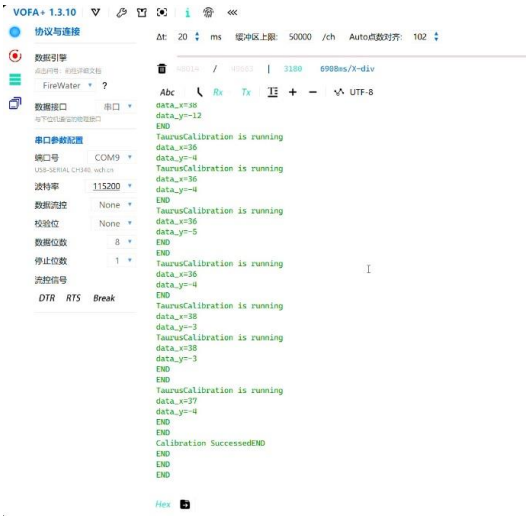


图 3-9 视觉校正打印数据

图 3-8 与图 3-9 展示了 Taurus 实时对目标商品进行识别并把数据发送给搬运车，图 3-9 为视觉校正时的真实距离数据，单位为厘米。

第四部分 总结

4.1 可扩展之处

我们接入了华为云平台但是没有对 Mobile App 和网页控制进行相应的开发，之后可以对手机 App 以及网页进行扩展。实现跨平台的可视化操作以及监视，可以实现识别信息的推流，在 App 上可以看到搬运车实时的摄像头画面。

在实际测试中我们发现轮子打滑比较严重，对地面要求较高，在光滑的地面容易发生较大的偏移。可以使用正交编码从动轮对里程进行记录，即使驱动轮打滑也不会影响到里程。

因成本原因我们的搬运车体积较小，机械臂的自由度也较少，不能胜任对较大商品的抓任务，未来我们可以对其进行重新设计，重新优化结构实现覆盖大部分商品的搬运车。

4.2 心得体会

本次使用 Taurus & Pegasus 套件进行开发，并且使用华为云进行物联网的控制，使我们对 Open Harmony 的设计理念有了更深层次的理解。通过本次项目的开发让我们对开发难度有了更加深刻的体会，“万物互联”这个理念让我对工程师们产生了深深的敬意，也更深刻理解了“万户互联”的理念。

在本次项目中，我们第一次实现了使用单摄像头深度学习对目标进行识别以及计算坐标，在 Taurus 上，以尽量高的速度进行目标物的检测以及坐标信息的转换。在 Pegasus 开发板上实现了多定位算法融合对目标物进行抓取以及接入云端。本次使用的开发套件为国产自研，让我深深感受到了祖国工程师们的强大，以及祖国的发展，同时也看见自研的不易，让我们对推动祖国发展的责任有了更加深刻的认知。

最后，感谢筹备此次大赛的全体工作人员，也祝愿此次大赛能圆满完成。

第五部分 参考文献

[1] Pegasus 开发套件实验指导手册
[2] Taurus & Pegasus 基础开发套件调试指导手册
[3] Taurus 计算机视觉基础开发套件操作指导
[4] 基于 Darknet 框架的 Yolo2 检测网进行 AI 应用开发与部署的指导手册
[5] 基于 Taurus 套件进行训练图片制作、模型训练、模型转换指导手册

第六部分 附录

6.1 视觉定位算法

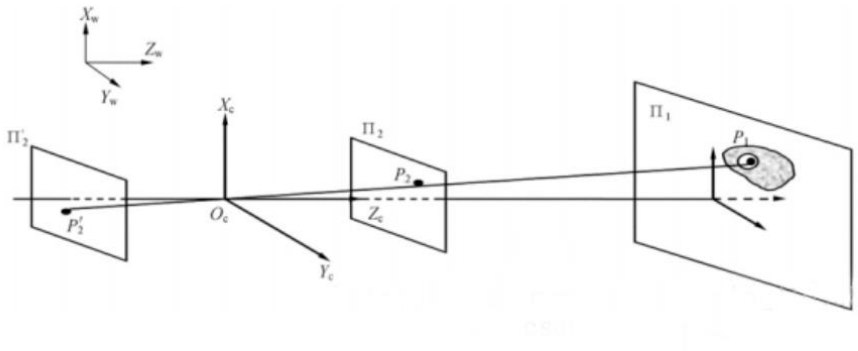
假设我们有一个宽度为 W 的目标或者物体。然后我们将这个目标放在距离我们的相机为 D 的位置。我们用相机对物体进行拍照并且测量物体的像素宽度 P 。这样我们就得出了相机焦距的公式：

$$F = \frac{(P \times D)}{W}$$

当我继续将我的相机移动靠近或者离远物体或者目标时，我可以用相似三角形来计算出物体离相机的距离：

$$D' = (W * F)/P$$

我们将使用相似三角形来计算相机到一个已知的物体或者目标的距离就可的出距离 D 与像素宽度 P，物体实际宽度 W 满足。



6-1 视觉定位算法原理图

以此为依据 代码如下（以 330 毫升 易拉罐为例 焦距为 3.6mm）

```
#define ACCURACY 0.0001
float Derivative(float (*func)(float),float position){
    float result = func(position+ACCURACY)-func(position);
    result /= ACCURACY;
    return result;
}

float function(float x){
    return (90*3.6*0.1)/x;
}
```

```
if (GoodsDetectFlag(numInfo[0]) == type)
{
    SAMPLE_PRT("{min:%d max:%d}\n", box->xmin, box->xmax);
    float x_max = (float)box->xmax;
    float x_min = (float)box->xmin;
     float area=(float)((box->xmax)-(box->xmin))*((box->ymax)-(box->ymin));
    int x=(int)Derivative(function,area);
    int y = (int)((x_max - x_min) / 2 + x_min) - 960;
    prInt(y,x, 0); //串口发送
}
```

6.2 底盘定位系统算法的设计

在底层电机控制方面采用增量式 pid 进行速度闭环，对轮子目标速度进行快速趋近。公式为：

$$\Delta UK = Kp(e(K) - e(K-1) + Kie(K) + KD[e(K) - 2e(K-1) + e(K-2)])$$

以下是部分代码。

```
Bias = Target - Encoder; //计算偏差
    Pwm += Velocity_KP * (Bias - Last_bias) + Velocity_KI * Bias;
    if (Pwm > 16700)
        Pwm = 16700;
    if (Pwm < -16700)
        Pwm = -16700;
    Last_bias = Bias; //保存上一次偏差
```

之后对整车的速度进行解算，通过四全向轮的运动学正解算法对整车的各轴速度进行计算分别是 x 、y、 z，下面是部分代码。

```
MOTOR_A.Target = +Vy + Vx - Vz * (Axle_spacing + Wheel_spacing);
        MOTOR_B.Target = -Vy + Vx - Vz * (Axle_spacing + Wheel_spacing);
        MOTOR_C.Target = +Vy + Vx + Vz * (Axle_spacing + Wheel_spacing);
        MOTOR_D.Target = -Vy + Vx + Vz * (Axle_spacing + Wheel_spacing);
```

在实现对整车速度的闭环之后对整车的位置进行闭环控制，使用位置式 pid 算法对输出速度数据进行计算。公式为

$$\Delta UK = KP \times e(K) + Ki \sum e(K) + Kd[e(K) - e(K-1)]$$

下面是 x 轴部分代码

```
pid_x.bias = target - position;
pid_x.integral += pid_x.bias;
pid_x.output_val = pid_x.Kp * pid_x.bias + pid_x.Ki * pid_x.integral + pid_x.Kd *
(pid_x.bias - pid_x.bias_last); //  位置式 pid
pid_x.bias_last = pid_x.bias;
```

实现对位置控制后对整车位置进行计算，采用陀螺仪角速度积分来计算整车的航向角，通过读取编码器的数据进行转换得到 x、y 轴的速度，之后进行位置的积分操作，以下是部分代码。

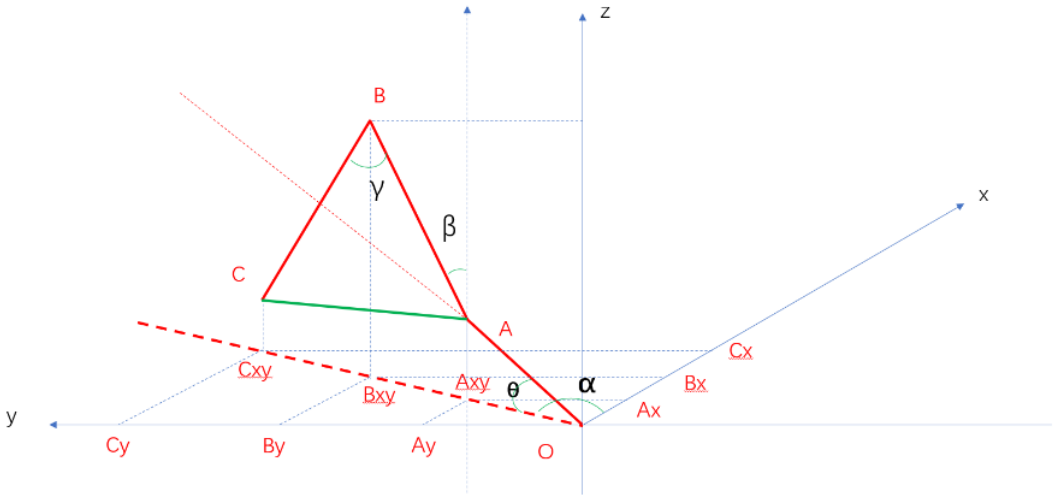
```
z_angle -= z_speed * 0.01; // yaw 角积分
  if (yaw > 0 && yaw < 90)
        {
x_world_speed = +d_x_encoder_speed * mycos(yaw) - d_y_encoder_speed *
mysin(yaw);
}
x_world_distance += x_world_speed * 1000 * 0.01; //*1000  转换为 mm
y_world_distance += y_world_speed * 1000 * 0.01; //*1000  转换为 mm
```

对位置进行计算后可对整体做闭环控制，下面是在航向角在 0 至 90 度之间时，x 轴目标速度输出的代码。

```
if (yaw > 0 && yaw < 90)
{
    x_car_speed_target = (+Position_PID_X(x_world_distance,
      X_DISTANCE_TARGET) * mycos(yaw) + Position_PID_Y(y_world_distance,
      Y_DISTANCE_TARGET) * mysin(yaw)) / 2;
}
```

6.3 机械臂逆解算法

机械臂的结构可抽象为



6.2 机械臂结构抽象图

$$\tan(\alpha) = \frac{Cy}{CX}$$

$$\cos(\gamma) = \frac{\frac{BC}{2} + \frac{BA}{2} + \frac{CA}{2}}{2 \times BC \times BA}$$

$$OC_{XY} = BC \times \sin(\gamma - \beta) + BA \times \sin(\beta) + OA \times \cos(\theta) = \sqrt{Cx^2 + Cy^2}$$

$$A_X = OA \times \cos(\theta) \times \cos(\alpha)$$

$$A_Y = OA \times \cos(\theta) \times \sin(\alpha)$$

$$A_Z = OA \times \sin(\theta)$$

$$AC = \sqrt{(Cx - Ax)^2 + (Cy - Ay)^2 + (Cz - Az)^2}$$

代码过长，附上图片，如图 6.2.

```
if (PedestalHeight > H)
{
    L2 = PedestalHeight - H;
    L1 = sqrt(L2 * L2 + L0 * L0);
    BigArmServoAngle = atan(L0 / L2) / pi * 180 + acos((BigArmLength * BigArmLength + L1 * L1 - SmallArmLength * SmallArmLength) / (2 * BigArmLength * L1)) / pi * 180 + BigArmServoCompensationValue-90;
    SmallArmServoAngle = 180 - acos((BigArmLength * BigArmLength + SmallArmLength * SmallArmLength - L1 * L1) / (2 * BigArmLength * SmallArmLength)) / pi * 180 - BigArmServoAngle + SmallArmServoCompoensationValue;
    ServoAngle[1] = BigArmServoAngle;
    ServoAngle[2] = SmallArmServoAngle;
}
else if (PedestalHeight < H)
{
    L2 = H - PedestalHeight;
    L1 = sqrt(L2 * L2 + L0 * L0);
    BigArmServoAngle = atan(L2 / L0) / pi * 180 + acos((BigArmLength * BigArmLength + L1 * L1 - SmallArmLength * SmallArmLength) / (2 * BigArmLength * L1)) / pi * 180 + BigArmServoCompensationValue ;
    SmallArmServoAngle = 180 - acos((BigArmLength * BigArmLength + SmallArmLength * SmallArmLength - L1 * L1) / (2 * BigArmLength * SmallArmLength)) / pi * 180 - BigArmServoAngle + SmallArmServoCompoensationValue;
    ServoAngle[1] = BigArmServoAngle;
    ServoAngle[2] = SmallArmServoAngle;
}
else if (PedestalHeight == H)
{
    L1 = L0;
    BigArmServoAngle = acos((BigArmLength * BigArmLength + L1 * L1 - SmallArmLength * SmallArmLength) / (2 * BigArmLength * L1)) / pi * 180 + BigArmServoCompensationValue;
    SmallArmServoAngle = 180 - acos((BigArmLength * BigArmLength + SmallArmLength * SmallArmLength - L1 * L1) / (2 * BigArmLength * SmallArmLength)) / pi * 180 - BigArmServoAngle + SmallArmServoCompoensationValue;
    ServoAngle[1] = BigArmServoAngle;
    ServoAngle[2] = SmallArmServoAngle;
}
```

6-2 机械臂逆解算法部分代码截图