浙江大学

SRTP 项目研究报告

基于轮式机器人的室内移动作业平台

姓名学号　刘海波 3180104391

叶晓桐 3180101949

曾亿诚 3180101990

指导教师　张宇

所在学院　控制科学与工程学院

目录

1. 项目背景

无人配送是近年来展现出强大成长潜力的新兴产业。无人配送包括智能快件箱、无人机、无人车等形式，是邮政快递末端服务的发展趋势之一。在疫情期间，订单量激增、劳动力不足、无接触配送需求旺盛，市场对于无人配送的需求有了显著提升。在办公大楼内，跨楼层的文件配送是常见事务，虽然单次工作量小，但是累积起来耗费人力。因此，我们小组基于轮式移动机器人搭建了一个室内作业平台，模拟配送场景，实现了机器人从起点出发、按动电梯按钮、搭乘电梯上楼的自主移动过程。

2. 技术路线

下面进行任务描述。任务的目标是使机器人平台可以在楼层之间穿梭并完成在指定地点之间传送文件。为达成该目标，我们将任务进行了更细一步的划分：根据初始定位与目标位姿的导航、基于视觉的按钮识别、机械臂按下按钮、切换楼层/进入电梯后的重定位等四项任务。每项任务的目标实现分别需要不同的传感器、控制器与执行器，在项目的前期，我们对平台的各个任务进行了单独调试，在能成功实现各个目标后，我们对平台进行了整合与集成，进行整体系统的联调。

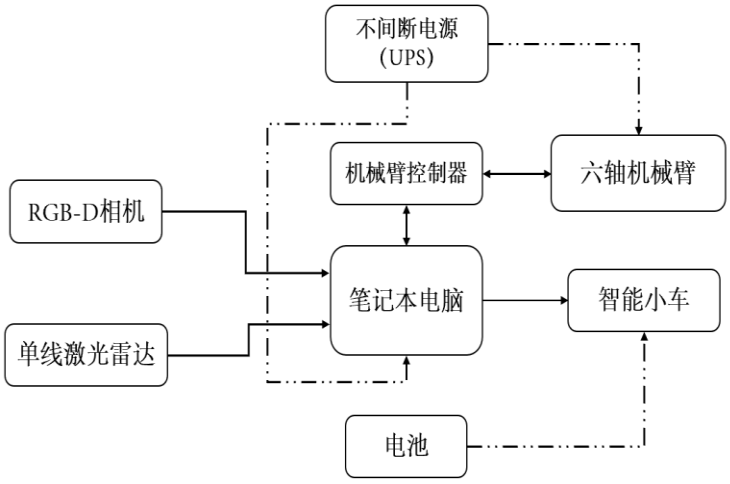

图 1-系统集成框图

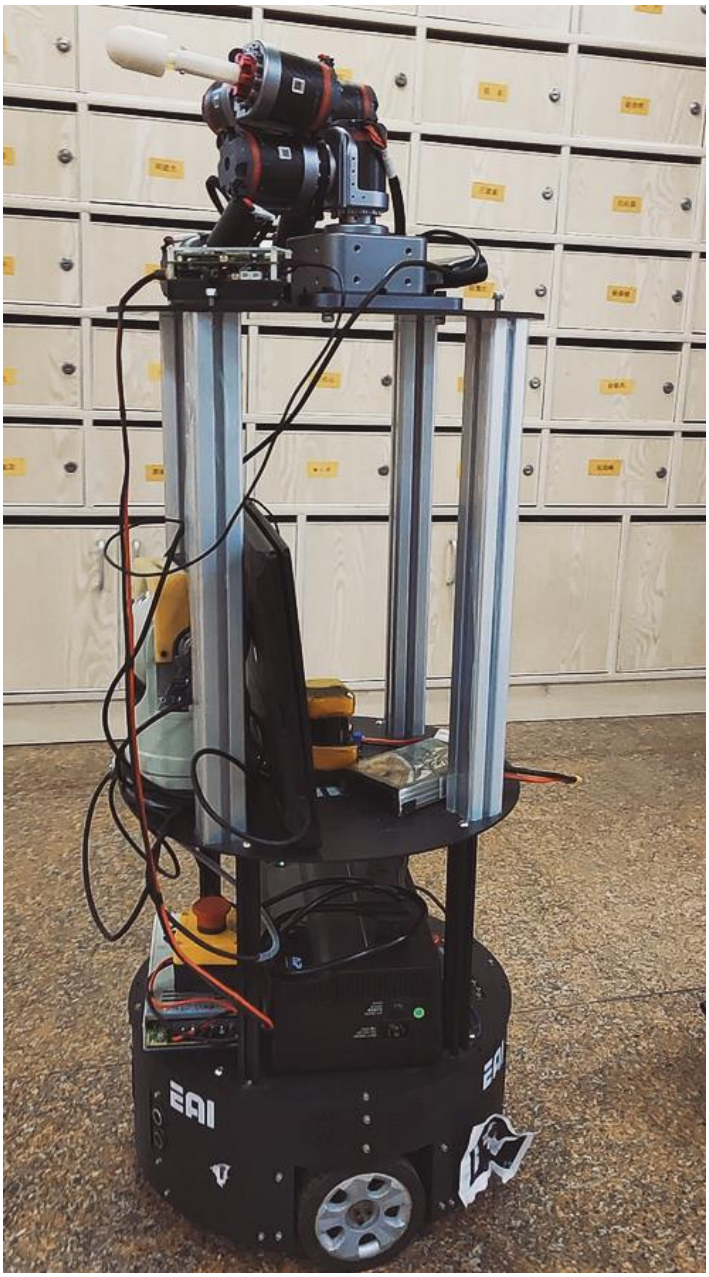

图 2-系统组成实物图

系统的完整框图如图 1 所示，系统的控制信号由笔记本电脑统一发送，利用 RGB-D 相机与激光雷达实现小车的定位与按钮识别功能。六轴机械臂的主要功能为按下电梯按钮。系统的供电由小型 UPS（不间断电源）完成，集成所有功能时的机器人系统如图 2 所示，通过向笔记本电脑发送启动指令后，系统将自动完成任务。

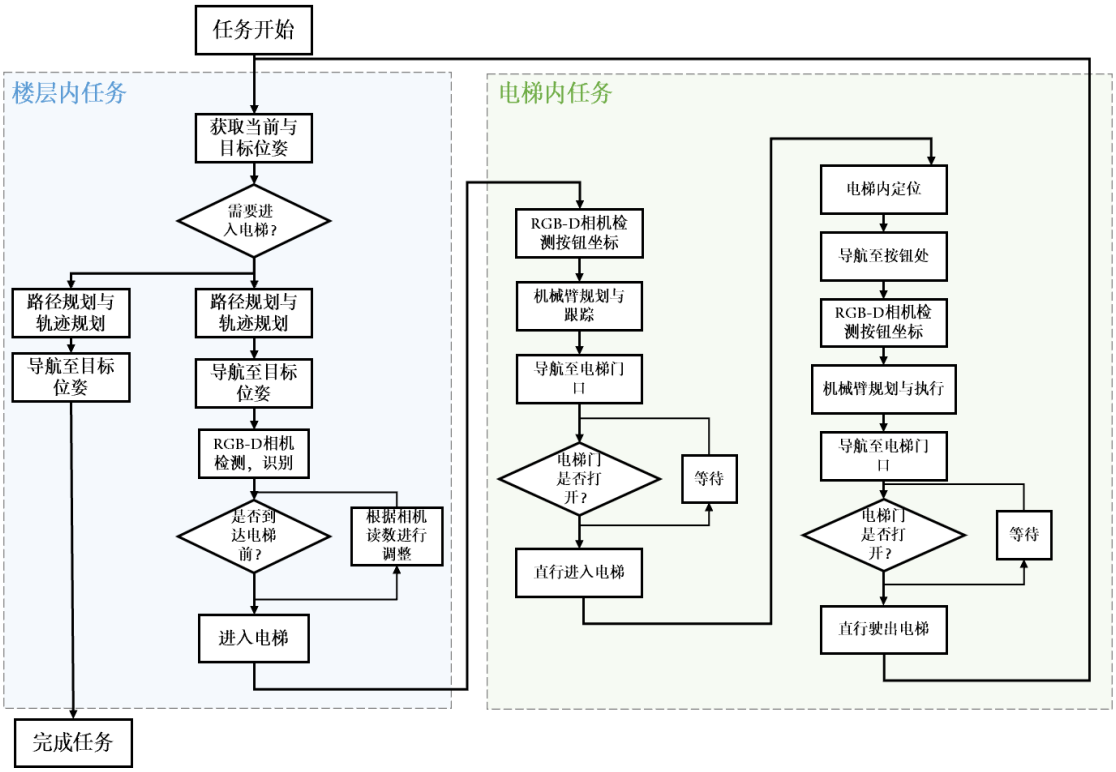

图 3-系统逻辑框图

任务的实现逻辑如图 3 所示，在软件层面，任务被进一步简化至楼层中任务与电梯内任务，通过 ROS 整合后，系统将自动完成上述逻辑，具体的话题与话题间通讯如图 4 所示。

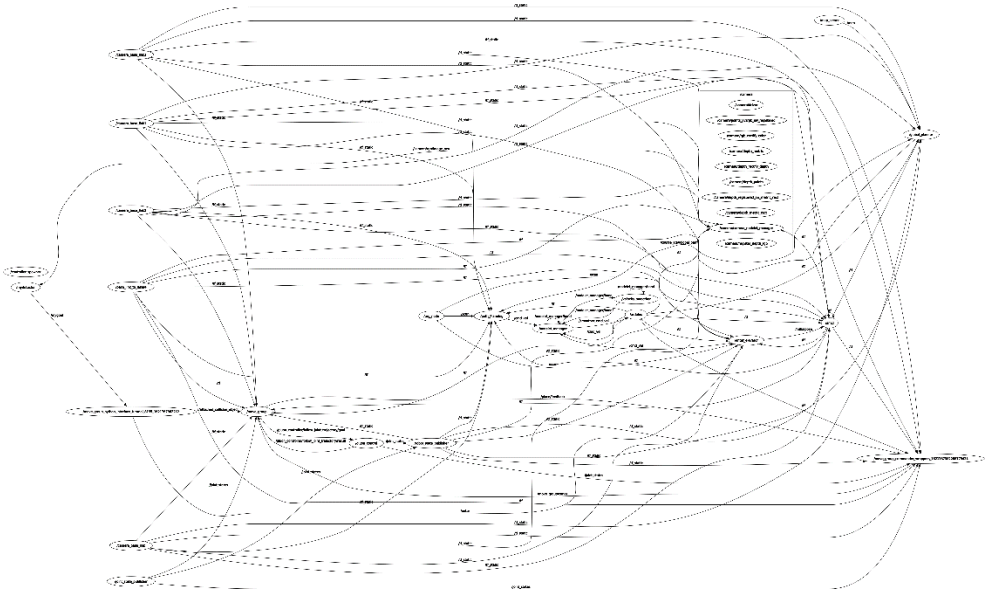

图 4-通信架构

3. 系统的硬件集成

3.1 坐标系集成

坐标系是机器人的核心概念，在整个机器人系统中，我们涉及了多个硬件的多个坐标系，如机器人平台坐标系、相机坐标系、机械臂坐标系、雷达坐标系、地图坐标系等，主要坐标系之间依赖关系如图 5 所示。

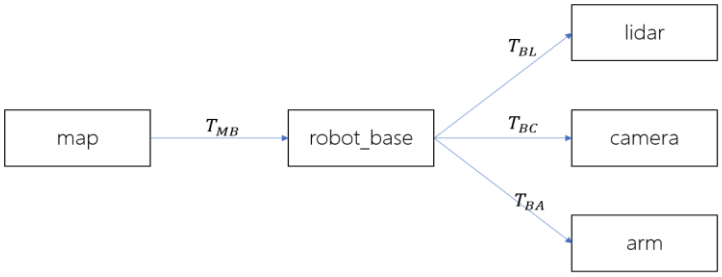

图 5-主要坐标系关系图

相机坐标系、机械臂坐标系、雷达坐标系、机器人平台坐标系之间的变换是固定的，并且在硬件集成的时候我们尽量保持坐标轴相互平行，因此可以简单地进行手动标定。而通过定位算法可以实时维护机器人平台与地图坐标系之间的变换。

坐标系变换的使用以按电梯按钮为例，相机检测到按钮坐标 ${}^{C}P$，将其变换到机械臂坐标系得到 ${}^{A}P = T_{AC}\,{}^{C}P = T_{BA}^{T} \cdot T_{BC} \cdot {}^{C}P$，机械臂再依此进行规划。

3.2 硬件集成

在硬件方面，项目需要兼顾多任务的实现，因此在融合多传感器与执行器后，需要根据工作环境的物理限制，进一步调整系统的硬件组成，优化部分平台本身的问题。机器人所用的视觉传感器为 ASUS 生产的 Xtion Pro RGB-D 相机，导航定位传感器为 velodyne 十六线激光雷达，搭载系统的轮式平台为 EAI 基于 ROS 的智能小车，执行器为 innfos 六轴机械臂。另外，轮式平台本身搭载超声传感器与近红外传感器，但由于精度不足，只能用于避障，因此未集成于系统中。

图 6-左为小车平台初始图，右为小车平台拓展后示意图

由于机械臂所需要触摸的按钮高度在[0.9,1.3](单位:m)的区间内，机械臂的工作空间半径为 0.6m，且激光雷达与 RGB-D 相机的运行均需要无近端障碍物遮挡的环境，轮式平台本身的空间不足以容纳供电系统与传感器-执行器。为此，我们利用铝型材与碳纤维板制作了拓展平台，用于搭载相机、机械臂及其控制器。拓展后，机械臂底盘的高度为 0.9m，在导航位置与按钮距离 0.2-0.3m 时，可以确保在末端水平的约束下，机械臂可以规划出合理路径以触摸按钮。同时，激光雷达与相机的检测范围不会受到遮挡（激光雷达被型材遮挡的点可以通过软件方式剔除）。

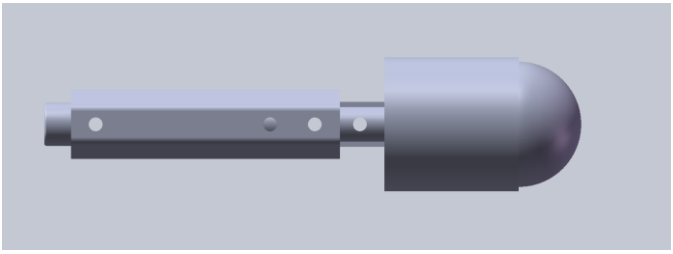

图 7-机械臂末端结构

由于机械臂本身并没有末端执行器，我们设计并制作了一个 3D 光固化打印件以满足机械臂触摸末端的要求，且可以通过螺栓位置进行长度的预调整。由于目标环境的按钮最大为 20mm*20mm，按下深度粗测约为 3mm。末端半球的直径

为 28mm，整体伸长量相较末端电机为[80,110](单位:mm)。初步实验结果显示，机械臂可以在由指定的目标坐标与约束提供的轨迹下，可以较为稳定地完成电梯按钮的触发，且不会受到机械臂本身重复定位精度较低的影响。

在导航的调试中，由于小车里程计存在一定误差以及基于单线雷达的定位算法精度可能不足，轨迹跟踪的常会有较大的误差累积，在导航结束时，累计误差与相机焦距的限制可能影响小车微调位置的成功率。项目进行时曾考虑通过更换驱动轮与电机、或设法建立更高精度地图以提高轨迹跟踪精度，但难度与时间消耗较大。随后，我们发现驱动轮与地面间的滑动可能由从动轮导致，轮式平台原装脚轮为塑料制，离地高度(38mm)略大于驱动轮自然触地时底盘高度(37mm)，小车运行时，驱动轮受压力较小，发生打滑。随后，我们将从动轮改为 1 英寸的聚氨酯万向轮，底座离地高度为 34mm，显著改善了驱动轮打滑的情况。

图 8 展示了系统硬件集成的最终结果。

图 8-集成整体图

4. 导航

4.1 建图

为了实现导航功能，首先建立环境地图，技术路线如下：

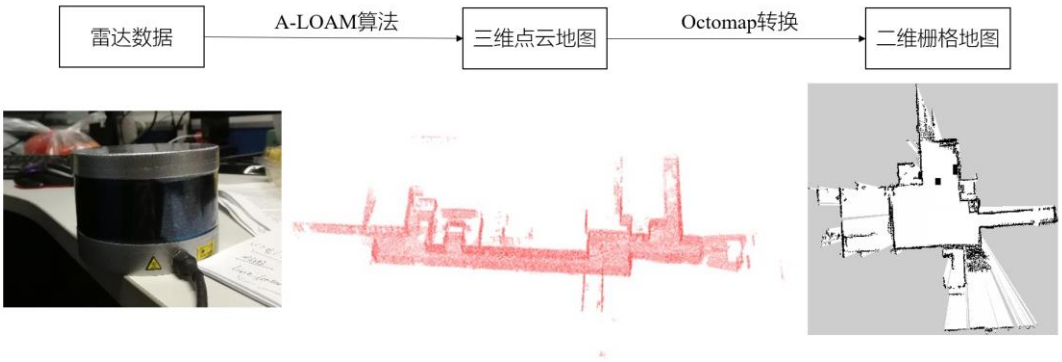

图 9-导航技术路线

① 使用 Velodyne VLP-16 十六线激光雷达在环境中采集数据，本项目的实验环境为工控新楼，所用雷达如图 9.a)所示。

② 使用 A-LOAM 算法作为里程计，A-LOAM 算法结合 scan-to-scan match 和 scan-to-map match 方法得到每帧点云的位姿，进而将点云变换到同一坐标系下，得到环境的三维点云地图，如图 9.b)所示。

③ 将环境划分为网格，通过点云数据得到每个网格被障碍物占据的概率并用灰度图表示，得到了环境的二维栅格地图，如图 9.c)所示。

4.2 路径规划

在建立好环境地图的基础上，给定起点和目标点，需要规划出一条合理路径。本项目选择使用 RRT*算法完成路径规划，并自己实现了该算法。

（1）RRT*算法实现

① 以运动规划初始状态q_{init}为根节点，建立搜索树。

② 进行以下循环：

在状态空间中，随机采样一个状态，用于引导搜索树的扩张，称为q_{rand}。

在现有的搜索树上查找与q_{rand}距离最近的节点q_{near}，以q_{near}和q_{rand}构建新的输入u，以q_{near}作为当前状态x，根据系统状态方程$\dot{x} = f(x,u)$，得到下一个状态即搜索树的扩张节点q_{new}。

对q_{new}进行碰撞检测，如果无冲突，则将q_{new}加入到搜索树，实现扩张。

对树中新节点邻域内节点进行判断，如果从新节点到该节点形成的路径优于现有树中路径，则将该节点父节点修改为新节点。

当q_{new}与终点足够接近或搜索树规模足够大时，结束循环。

③ 如果规划成功，生成从q_{init}到终点的路径。

Algorithm 6: RRT*.

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}(V)) / \text{card}(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\}$;
    $x_{\min} \leftarrow x_{\text{nearest}}$; $c_{\min} \leftarrow \texttt{Cost}(x_{\text{nearest}}) + c(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Connect along a minimum-cost path
      if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\min}$ **then**
        $x_{\min} \leftarrow x_{\text{near}}$; $c_{\min} \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$
    $E \leftarrow E \cup \{(x_{\min}, x_{\text{new}})\}$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Rewire the tree
      if $\texttt{CollisionFree}(x_{\text{new}}, x_{\text{near}}) \wedge \texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}})) < \texttt{Cost}(x_{\text{near}})$ **then** $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
        $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

图 10-RRT*算法

（2）路径规划效果展示

图 11-路径规划效果展示

4.3 轨迹规划及避障规划

在路径规划的基础上，进行轨迹规划和避障规划，得到每时刻下发给机器人的指令$[v,\omega]$。本项目选择使用 DWA 算法完成规划，并自己实现了该算法。

（1）DWA 算法实现

① 基于速度控制运动模型，构建可行的速度空间，选择可以让机器人停止不与障碍物相碰的可行速度集合V_a。

$$V_a=\left\{(v,\omega)\,|\,v\le\sqrt{2\cdot dist(v,\omega)\cdot\dot{v}_b}\wedge\omega\le\sqrt{2\cdot dist(v,\omega)\cdot\dot{\omega}_b}\right\}$$

② 考虑到机器人在运动过程中最大加速度的约束，在当前速度配置处以固定的小时间间隔开一个速度窗口空间V_d。

$$V_d=\{(v,\omega)\,|\,v\in[v_l,v_h]\wedge\omega\in[\omega_l,\omega_h]\}$$

$$\begin{cases}v_l=v_a-a_{v\max}\times\Delta t\\ v_h=v_a+a_{v\max}\times\Delta t\\ \omega_l=\omega_a-a_{\omega\max}\times\Delta t\\ \omega_l=\omega_a-a_{\omega\max}\times\Delta t\end{cases}$$

③ 考虑机器人实际最大速度约束，得到速度集合V_s。

$$V_s = \{(v,\omega) \mid v \in [-v_{\max}, v_{\max}] \wedge \omega \in [-\omega_{\max}, \omega_{\max}]\}$$

④ 综合以上三个约束，得到最终可选速度集合$V = V_a \cap V_d \cap V_s$，在速度集合中进行采样，对于每一个样本点$[v,\omega]$，使用评价函数$evaluation(x,\omega)$，选择出最优速度指令。

$$evaluation(v,\omega) = \alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega)$$

$$\alpha + \beta + \gamma = 1 (\alpha \geq 0, \beta \geq 0, \gamma \geq 0)$$

$heading(v,\omega)$ 朝向目标点 ：保证机器人朝目标点运动

$dist(v,\omega)$ 远离障碍物 ：保证机器人避开障碍物，安全不碰撞

$velocity(v,\omega)$ 速度最大化 ：保证机器人以最大速度运动

（2）效果展示

图 12-DWA 效果展示

4.4 定位

在机器人导航全过程中，定位功能至关重要，本项目主要使用了两种定位方法：在楼层内定位使用 AMCL 算法定位，在电梯内部使用雷达及环境约束进行定位。

（1）AMCL 定位

AMCL 即自适应蒙特卡洛定位，算法采用粒子数代表定位的可能性，首先在全局均匀撒一些粒子，通过每时刻的雷达点云信息、轮式里程计信息，改变粒子的分布情况，最终输出机器人的概率定位。

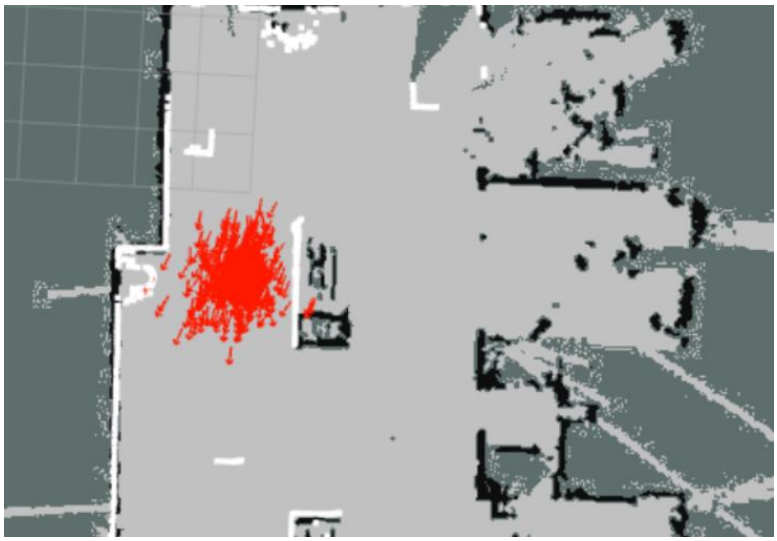

图 13-粒子滤波定位效果

（2）雷达和环境约束定位

如上所述，AMCL 定位使用了轮式里程计的信息，而机器人在出入电梯的时候，由于楼梯口存在夹缝，轮子容易出现打滑悬空等问题，引发轮式里程计的误差，进而造成定位误差。因此在进入电梯后，必须重新修正机器人的定位，项目采用雷达数据和电梯墙面的约束实现了机器人重定位。

① 在机器人进入电梯后，点云数据如下图所示，可以明显看出三面墙壁。

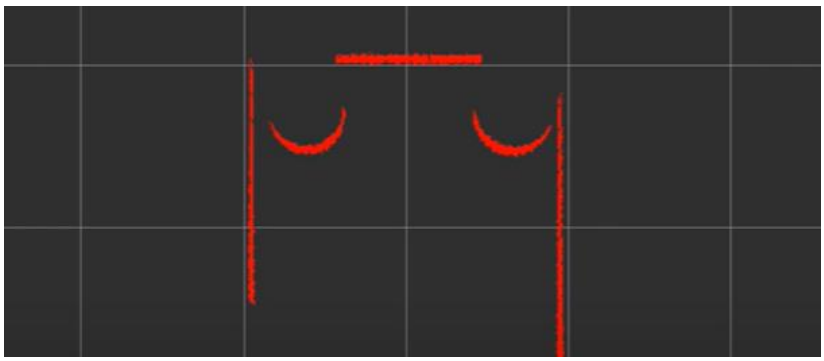

图 14-电梯内点云数据图

② 对点云数据进行直线拟合，假设其中一条直线为$y = kx + t$，如下图所示，则机器人到墙壁面的距离为$\frac{|t|}{\sqrt{1+k^2}}$，机器人正方向与墙壁面所成锐角为$\arctan(|k|)$，从而可以通过电梯墙壁推算出机器人的定位。

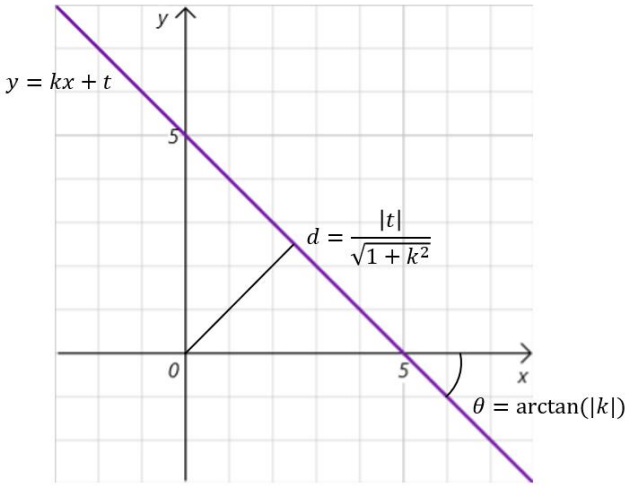

图 15-直线拟合结果

5. 识别

5.1 数字识别

我们原定的任务场景中包含了小车对写在配送物件上的数字进行识别，以确定文件配送的终点房间号。

在 mnist 数据集上训练一个识别手写数字的 CNN 网络，训练轮数 100 轮。

```
Epoch:  0 batch_idx:  9450 | train_loss: 0.1089
Epoch:  0 batch_idx:  9500 | train_loss: 0.1324
Epoch:  0 batch_idx:  9550 | train_loss: 0.1228
Epoch:  0 batch_idx:  9600 | train_loss: 0.1691
Epoch:  0 batch_idx:  9650 | train_loss: 0.1293
Epoch:  0 batch_idx:  9700 | train_loss: 0.1600
Epoch:  0 batch_idx:  9750 | train_loss: 0.1112
Epoch:  0 batch_idx:  9800 | train_loss: 0.1553
Epoch:  0 batch_idx:  9850 | train_loss: 0.1320
Epoch:  0 batch_idx:  9900 | train_loss: 0.1366
Epoch:  0 batch_idx:  9950 | train_loss: 0.1146
dinaelin@dinaelin-WRT-WX9:~/PycharmProjects/pythonProje
```

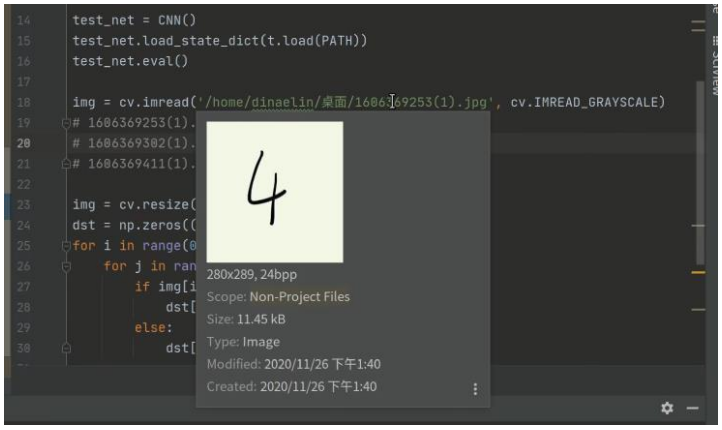

图 16-数字识别训练结果

由于输入的数字图片是白底黑字，而 mnist 数据集上的图片是黑底白字，对图像反转取二值化。输入到网络中，输出 10 个概率，概率最大一项就是识别结果。

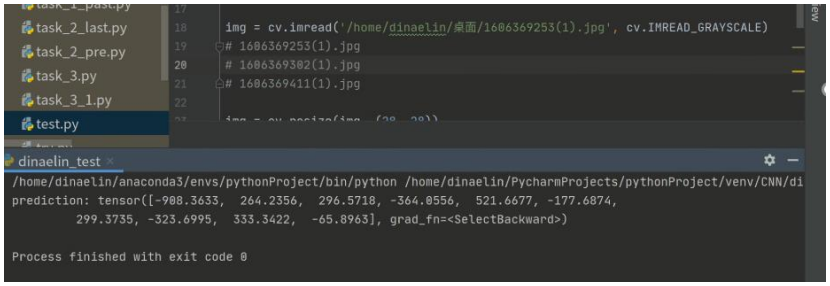

图 17-数字识别训练结果

5.2 按钮的识别

我们使用两种方法进行按钮位置的识别。

第一，采用 YOLO-v3 网络，在自己构建的按钮数据集上实现上下按钮的目标分割。以下是训练需要用到的文件夹。

第一个文件夹用于存放训练代码.，包括 YOLO 的网络架构、xml 格式的标签文件和 txt 格式的类别文件。

第二个文件夹用于存放训练权重。由于笔记本自带显卡不支持如此大数据量的网络训练，我们在 google colab 上训练好了网络，再把权重拷贝到本地。

第三个文件夹用于存放数据集。数据集中所有图片都是通过 RGBD 文件拍摄的，并通过 labelImg 打上“上下按钮”的标签。

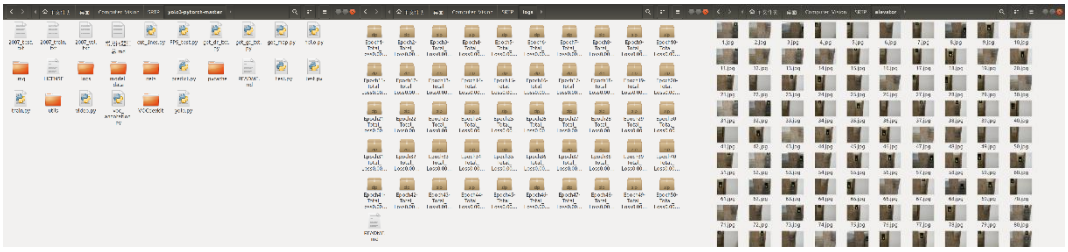

图 18-按钮识别数据集和权重文件

yolov3 的网络架构如图 13 所示：

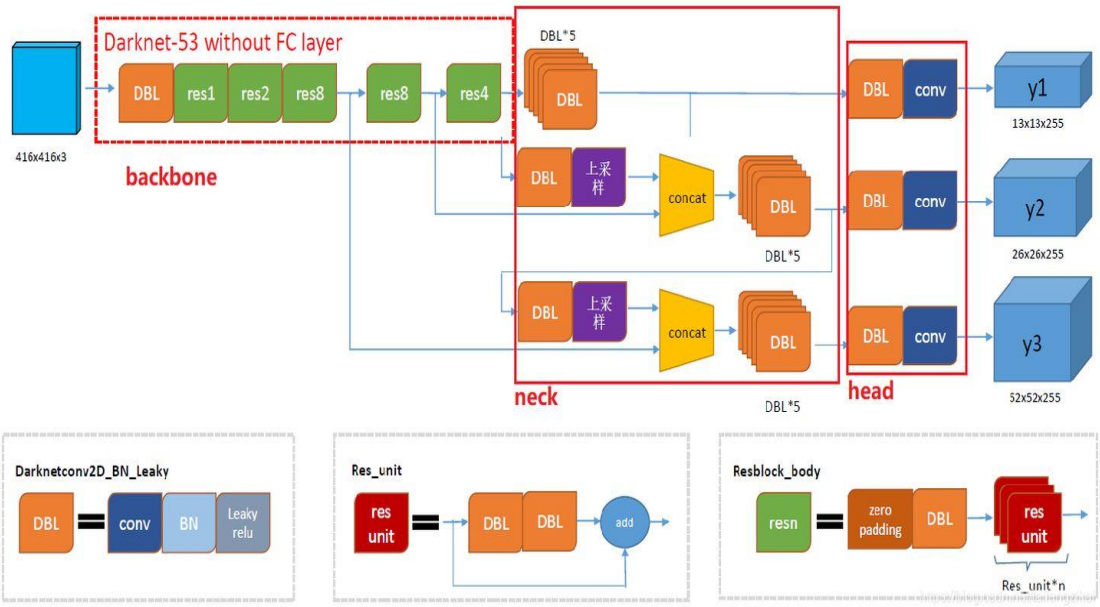

图 19-YOLO-v3 网络结构

以下是训练结果，将三张图片输入网络，得到如下检测框和置信度。

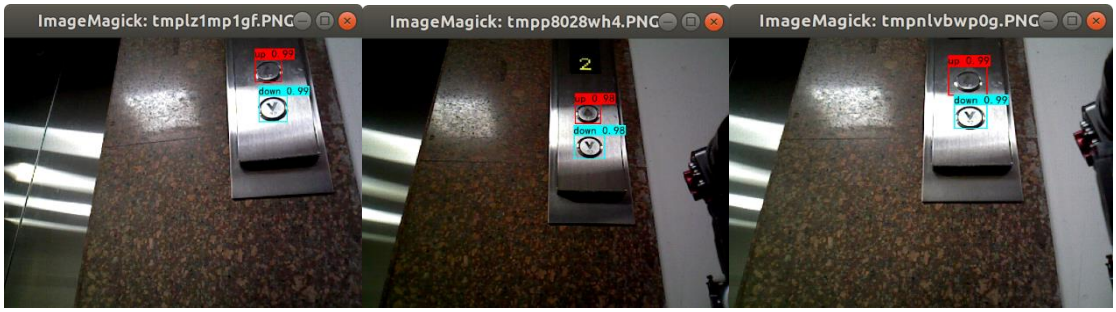

图 20-按钮识别训练结果

第二，利用 OpenCV 提供的 ArUco 库，通过相机检测粘贴在按钮下方的二维码位置，推算出按钮所在位置。

考虑到实验室的笔记本性能不足以支持按钮的实时目标检测，我们最终采用了这种方案。我们事先在按钮下方平整粘贴好二维码的打印图，调用库函数检测二维码，得到旋转矩阵和平移向量，进而推算出按钮的位姿。

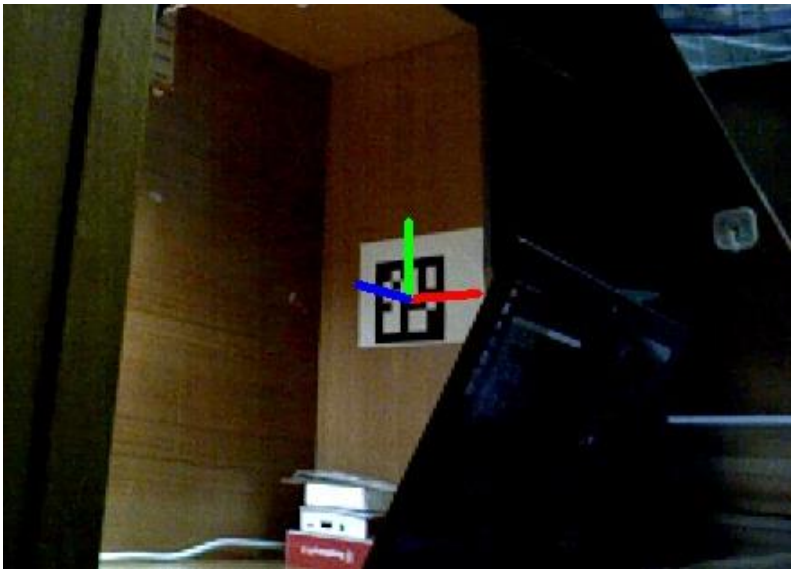

图 21-ArUco 识别结果

5.3 RGBD 相机标定

在导航过程中，小车的视觉识别任务是通过 RGBD 相机识别按钮位置，驱动机械臂执行按下的操作。

在上一步，我们已经得到了二维码在机械臂空间中的三维坐标。下一步，根据二维码和按钮的纵向距离，得到按钮的坐标。最后，驱动机械臂执行若干次按动操作，观察机械臂末端到达位置和按钮位置的差分，再做进一步的数据微调。

5.4 OCR-RCNN 识别

我们还找到了一种基于 RCNN 网络的两步目标检测方法，理论上来说它比 yolo 拥有更高的检测精度。

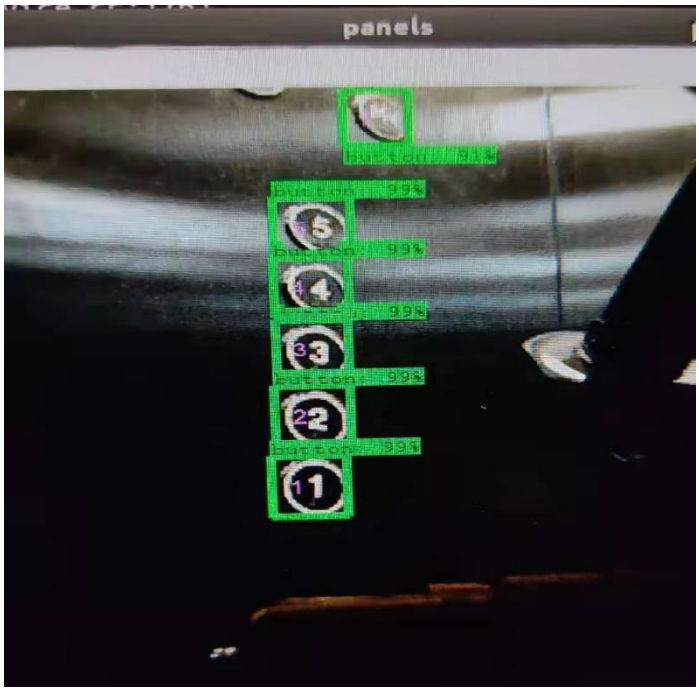

图 22-OCR-RCNN 识别结果

6. 机械臂控制

给定机械臂坐标系下一个目标位姿，控制机械臂从初始位姿移动到目标位姿。本项目采用 Moveit 开源框架实现机械臂控制。

① 在关节角度空间指定机械臂初始位姿如图-23 所示；

② 收到目标位姿后，调用 Moveit 提供的接口程序进行规划并执行，结果如图-24 所示。

图 23, 24- 机械臂规划实验结果

7. 遇到的问题

7.1 硬件

回首整个实验过程，我们遇到的最大问题，是硬件问题；硬件问题中的最大问题，是轮子的问题。出于实验需要，我们在 Dashgo 小车底盘上搭建了三层平台，导致底盘承重超出一定限度。Dashgo 的轮子有四个，横向的两个主动轮和纵向的两个辅助轮。在几次小车过电梯的过程中，由于电梯缝隙有较大的摩擦，小车辅助轮断裂。在网购了差不多相同尺寸的车轮后，我们又发现一些新辅助轮会导致小车拐弯过程中的滑动摩擦，影响拐弯精度。

第二个问题是电池。电池的供电对小车的电力驱动有很大影响。即使参数输入相同，如果电池电力不足，也会造成小车驱动效果下降。比如，小车前进速度下降，导致辅助轮卡在电梯缝隙中间；再如，小车拐弯速度下降，导致拐弯后没有面向电梯门口。

7.2 软件

小车在运行过程中要执行许多由 ros 发布的命令。我们发现，由于命令发布和执行由一定的时间偏差，连续发布命令会造成小车的部分命令没有执行完。比如，连续发布小车转弯 90°和直行的命令，会导致小车转弯 60°左右就开始直行。解决方法是，利用 time 库中的 sleep 函数，强制令下一个命令在上一个命令发布完毕后若干秒再执行。

我们在软件上遇到的最大问题是定位精度。当使用激光雷达第一次构建的地图中实验，我们发现小车从实验室门口运行到电梯按钮前的位置会有不确定的偏差，这是地图构建操作不当导致的。第二次构建后，我们仍然发现小车的转弯存在严重偏差。为了解决这个问题，我们利用雷达识别到的点云斜率，帮助小车判断自己是否正对障碍物和电梯门口。如果斜率在 0 附近，就说明是。如此一来，我们基本解决了转弯精度的问题。

8. 应用前景与后续研究方向

本项目的研究背景为物流行业与有电梯的大楼内的跨楼层文件运输。通过研究与实验，实现了轮式-机械臂平台的跨楼层导航定位与自主出/入电梯功能，且进行了实际情况的测试。因此，在完成目标环境的二维栅格地图的建图与电梯按钮的参数标定情况下，我们的实验平台可以达到楼内任一已知的指定目标。若进行一些交互功能的完善，可以应用于简单的物品输送任务。

但是，系统本身依然存在一些由于时间限制、知识限制与经费限制导致的功能不完善与难以泛化应用的问题。后续的深入研究与开发，可以从如下的几个方面展开：

1.通过标记过的电梯内/外图像，我们训练出了可以识别电梯按钮数字与位置的神经网络，但在坐标读取上，我们的识别精度并不理想，提供给机械臂后很难启动电梯开关。最终，项目所使用的识别方法为通过获取 aruco 二维码的坐标，

并通过预先获取的按钮与二维码的坐标转换关系计算出目标坐标。如果进行进一步的开发，可以着眼于提高数字识别的精度，实现更高鲁棒性的目标坐标获取。

2.当前的机械臂末端机构可以实现对某个目标点的触摸，但并没有夹取功能。若需要实现更进一步的物品夹取和运送，可以在机械臂末端加入伺服电机驱动的单自由度夹取机构，在合紧状态下，同时可完成电梯按钮的触摸。若能实现夹取，则实验平台可以实现完整的文件/物品运送功能。

3.机器人系统的集成度较低，需要通过笔记本电脑上基于 Linux 系统下的 ROS 实现各个功能的整合与话题间的通讯。这在前期提升了我们的调试效率，但对于产品而言存在性能冗余。且笔记本与机械臂的供电对平台有很大的限制。若要对系统进行更进一步的深入开发，可以考虑将所有功能集成于片上系统中，利用电脑端进行远程调试。

4.轮式平台的性能限制较大：自身的传感器精度不足，无法辅助实验效率的提升；运动能力不能适应实验要求，如进入电梯时颠簸严重，运动时存在打滑等。在后续的开发中，考虑使用基于麦克纳姆轮的实验平台，并引入弹簧式的悬挂系统。自主研发的全向轮平台应能实现二维地图中的完整移动，且能适应更复杂的室内地形。且由于机械结构可以主动调整，平台的拓展性会更强。可以集成更多适用的传感器。

9. 心得与体会

历时一年的 SRTP 项目中，我们体验了一个项目的研发全过程，从背景调研到具体任务的确定与分解。根据已有的平台进行分别调试，根据实验结果进行软硬件的迭代升级，机器人系统的整合与联调，直至最后的总结。为实现某个要求，我们查阅资料并吸取了优秀工程的经验；为提升某个任务的实现精度，我们不断发现已有的软、硬件上的问题并尝试解决；为提升系统的鲁棒性，一次次否定已有的方案并提出新的方案。这个过程中有实验成功的喜悦和满足，但更多的是艰难而煎熬的调试，我们经历了许多因实验失败而不眠不休的夜晚，也体验了因为

失误而导致几个月成功付诸东流的难过。对我而言，相较于成果，经验是我更需要品尝的收获。

回顾整个项目的历程，我感觉在后续的科研路上，最需要学习的几个经验有：

1.在项目开始前，确定一个清晰，明确的目标，一个好的目标可以使研发的方案更加明确。本次的项目中，我们并没有在开始时将任务的明确和分解做好，导致开始时的一段时间并不知道应作何知识上的准备，真正接触实物时花费了较多时间熟悉它们的使用，导致前期的进度偏慢，直到我们进一步明确任务并开始进行分别调试时，项目进度才步入正轨。

2.将每次的计划制定在能力和知识可行的范围内，避免太多、太复杂的任务导致停滞不前。项目开始时，我们希望实验的平台可以在结束时可以完成将文件从此处夹起，实现与人的柔性交互，并自主完成导航算法、识别算法与机械臂的控制算法，但开始时发现预计的目标高于目前的能力，导致项目中期处于焦虑和沮丧中。

3.需要不断积累知识，以应对项目中的不确定性。实现分立任务的功能实际上并没有花费太多时间，但联调却并不顺利，因为真正在面临实际的工作环境时，我们的实验平台暴露出了许多细小，但足以影响实验结果的问题。譬如导航的累计误差使平台无法通过相机信息进行位姿校正、驱动轮在进入电梯时卡住等若干问题。在之前的学习过程中，我们更专注算法的实现，但硬件知识，包括机械问题与嵌入式系统的开发，都能有效解决实物系统在运行过程中遇到的许多问题。