摘　要

随着机器视觉与 SLAM 技术的迅速发展，机器人智能化程度越来越高。扫地机器人是服务型机器人中成功落地的代表，与早期市面上随机碰撞式的扫地机器人相比，具有地图构建能力的扫地机器人更加智能化，可以利用已知的地图进行导航和路径规划。

本文基于 TurtleBot 机器人平台和 ROS 系统进行扫地机器人系统的设计，Kinect 深度相机为本设计的主要传感器，根据投影变换将采集到的深度图数据转换为激光扫描数据，利用 ROS 中的 GMapping SLAM 算法包进行环境地图的构建，其中里程计数据来自 Kobuki 底盘的车轮编码器。本文调研了移动机器人的路径规划算法，包括点到点路径规划与全覆盖路径规划。点到点路径规划算法包括迪杰斯特拉算法和 A*算法，并将启发式搜索的 A*算法应用于实际的导航应用。全覆盖路径规划采用子区域分解的方法，在每一个子区域内，通过简单的往复式运动实现。

我们在实际的会议室场景完成了地图构建、导航和路径规划实验，地图的栅格尺寸为 5cm，点到点的路径规划算法的实验结果表明，我们所有的算法可以有效地规划从起始点到目标点的路径；全覆盖路径规划算法在较小重复率的前提下可实现区域的完全覆盖。

关键词：路径规划；SLAM；全覆盖；A*算法

Abstract

With the rapid development of machine vision and SLAM, the degree of intelligence of robots is becoming higher. The mobile cleaning robot is the representative of the successful landing of the service robot. Compared with the random collision type sweeping robot in the early market, the sweeping robot with mapping capability is more intelligent, and the constructed map can be used for path planning and navigation.

Our work is based on the TurtleBot robot platform and Robot Operating System for the design of the mobile cleaning robot. The main sensor is the Kinect depth camera produced by Microsoft Corporation. The depth image data collected is converted into laser scan data by the projection transformation. Simultaneous positioning and map construction using the GMapping SLAM algorithm package in ROS, where the odometer data comes from the wheel encoder of the Kobuki mobile base.

We completed the mapping, navigation and path planning experiments in the actual conference room scene. The grid size of the map is 5cm. From the experiment of point-to-point path planning, we can see that the sweeping robot can avoid obstacles and reach the navigation goal. Full coverage path planning algorithm can achieve complete coverage of the area with a small repetition rate.

Key Words：path planning；Simultaneous Localization and Mapping；complete coverage path planning；A star algorithm

目录

第 1 章 绪论

1.1 课题的研究意义

随着社会的发展和科学技术的不断进步，机器人领域的相关技术也得到了迅速的发展。机器人根据应用场景的不同可分为工业机器人、服务型机器人和危险作业机器人等。和人们日常生活息息相关的是服务型机器人，常见的服务型机器人包括医疗服务机器人、图书馆/餐厅服务机器人和家庭清扫机器人等[1]，如图 1.1 所示。

根据调查数据表明，国内扫地机器人市场规模逐渐扩大，市场销售额自 2013 年到 2018 年连续多年持续增长。复合增长率高达 61%，远高于全球扫地机器人市场 22%的复合增长率，可见国内扫地机器人市场之大。在这种情况下，国内各路厂商纷纷开始进行扫地机器人的研发与生产，比如科沃斯、小米和海尔等。

扫地机器人行业门槛较低，但把产品做好却十分不容易。其核心技术是环境地图构建和路径规划，不具备建图功能的扫地机器人只能在房间内以随机遍历的方式进行清扫，假如清扫时间足够长，那么在理论上可以实现扫地机器人对房间的完全覆盖，但考虑到机器人电量的消耗，这种工作方式在实际工作中效果较差。

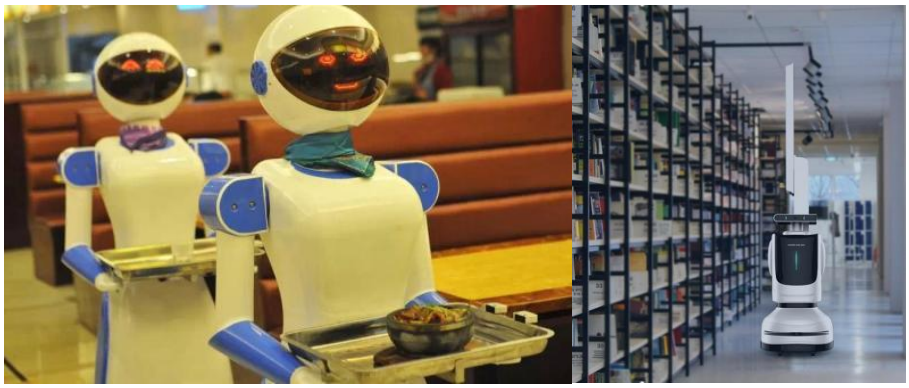


图 1.1 各种服务型机器人

具有环境地图构建能力的扫地机器人在第一次工作时会根据自身携带的传感器记录环境信息，已知机器人自身的位姿和环境信息就可以增量式构建环境地图。在已知环境地图的条件下，根据规划算法找出一条可以遍历整个地图的路径。因为此种类型的扫地机可以根据事先规划好了的路径进行工作，所以工作效率高，路线重复率小。

根据主要传感器的不同，扫地机器人可分为激光和视觉两种。国内外厂商对激光雷达扫地机器人的研究开始较早，因为可以应用到嵌入式开发平台，占用的计算资源相对较少，目前市面上已有不少此类产品。但是激光雷达的价格昂贵且长期处于旋转中的机械结构会给系统带来不稳定性，不利于普及应用。与激光雷达相比，视觉传感器的价格较低，可以

获得更为丰富的环境信息。但是视觉信息占用的计算资源大，在低成本的嵌入式平台实现较难。国外企业较早地开始对视觉导航式扫地机的研发，目前市面上已有多款产品。

1.2 国内外扫地机器人研究现状

扫地机器人最开始出现在 20 世纪末，在 iRobot 提出三段式清扫方案解决了清扫的问题之后[2]，扫地机器人才真正具备了实际应用的价值，才开始正式商业化。扫地机器人也给 iRobot 带来了巨大的商机，在接近 20 年的时间中，实现了从数万台到百万台的跨越发展。但是随机碰撞式扫地机器人产品的用户体验较差，客户抱怨重复清扫、漏扫，以及经常出现卡住情况。在 2015 年以后，iRobot Roomba980、科沃斯 DN 系列、小米米家等产品相继发布，扫地机器人由功能型向智能化转型，由基本的清扫功能，到清扫效率、清扫质量的显著提升，路径规划类产品已经成为行业的标配，用户体验明显提升。

iRobot 公司是国外一家高新科技公司，该公司生产的 Roomba780 和 Roomba980 分别为随机遍历式和路径规划式扫地机器人的代表。Roomba980 如图 1.1(a)所示，采用单目视觉规划技术，配有朝向天花板的相机，采用红外传感器进行避障。Roomba 980 具有地图构建和路径规划的能力，工作效率高，规划的路线重复率低，性能优良，但是价格昂贵，大约 7500 元人民币[3]。

(a)iRobot Roomba 980

(b)小米米家

(c)科沃斯 DN55

图 1.2 扫地机器人产品

小米米家扫地机器人如图 1.1(b)所示，通过机身的 LDS 激光测距传感器来感知环境信息，对所在房间进行扫描以获取距离信息，当激光投射到障碍物上，会在眼睛里形成光斑，图像传感器会根据光斑计算到激光测距传感器的中心距离[4]。此外通过 SLAM 算法可以实时构建房间地图，并准确计算自己在地图上所处的位置，为全面的路径规划提供保障。

科沃斯是国内一家专注于做家庭服务机器人的公司，该公司生产的地宝系列扫地机器人在国内市场占有较高的份额，科沃斯 DN55 产品如图 1.1(c)所示。该公司生产最近的研发成果融入了目标检测技术，可以识别室内场景中的拖鞋、垃圾桶等物品，从而进一步提高了扫地机的智能化水平。

1.3 研究内容与章节安排

本文主要研究适用于扫地机器人的 SLAM 建图算法和路径规划算法，并基于 Turtlebot 机器人平台进行扫地机器人的设计，在实际的会议室场景内完成地图构建、导航与路径规划等实验。

第一章首先说明了本课题的研究意义，调研了国内外的扫地机器人的研究现状。

第二章进行 SLAM 传感器的对比选择、SLAM 方案的选择与论证等，对适用于机器人导航的地图类型进行选择。

第三章对 TurtleBot 机器人平台进行介绍，包括 Kinect 相机和 Kobuki 移动底盘。以针孔相机模型说明相机的成像原理，并详细阐述了深度图到激光数据的转换。详细介绍了扫地机器人的运动模型，为机器人的导航、路径规划奠定了基础。介绍了路径规划领域常用的 Dijkstra 算法和 A*算法。全覆盖路径规划算法采用子区域分解的方法。

第四章进行实验测试，基于 Kobuki 移动底盘和 Kinect 深度相机进行扫地机器人环境建图、定位导航与路径规划实验。

第五章是对本文进行总结，并展望下一步的研究方向。

第 2 章 扫地机器人系统方案选择

2.1 SLAM 传感器选择

SLAM 常用的传感器有相机、激光雷达，本小节对这两种传感器进行详细介绍并选择适用于本文的方案。

2.1.1 相机

单目相机 常见的单目相机有针孔模型相机、鱼眼相机等。单目 SLAM 顾名思义就是只是用一个相机作为传感器进行 SLAM。因为单张图像是现实的三维世界向二维平面的投影，所以说仅通过单幅图像是无法感知距离信息的。因此为了得到距离信息，我们必须移动镜头改变位姿，通过不同角度的差异来计算深度信息，这种做法成为三角化。与 RGB-D SLAM 相比，单目 SLAM 需要初始化：即通过一段位移来确定尺度的大小[5]。

单目 SLAM 的优点是设备简单、成本低。缺点是相机无法适应快速运动的情况，比如说由于快速运动导致的运动模糊问题，因此存在建图精度较差的问题。

(a)单目相机　　(b)双目相机　　(c)深度相机

图 2.1 单目、双目和深度相机

双目相机 双目相机由两个固定的单目相机组成，这两个相机水平放置，而且两相机之间的距离（称之为基线）是已知的。可以通过计算两视图的视差来获取图像像素的深度信息，所以双目相机不存在单目相机的尺度不确定性问题。

双目相机的优点在于可以获取图像的深度图，同时也存在一些不足：相机的安装配置与标定过程比单目相机复杂，而且距离的量程与基线大小有关，基线越大，可以测量的范围越大。而且视差的计算十分耗时，在一般 CPU 上很难实时处理，一般需要并行设备加速后才可得到整张图像的深度信息。

深度相机 与单目、双目相机相比，深度相机可通过物理方法直接获取图像的深度数据。深度图的获取方法有红外结构光和飞行时间法等，即深度相机同时可以得到同一场景的彩色图和深度图，深度图一般为灰度图像，颜色越深表示距离越近。

通常深度相机有测量距离的限制，距离过大或过小都无法准确获取深度值数据。深度相机的应用场景一般在室内，在室外环境效果较差。

2.1.2 激光雷达

激光雷达通过发射激光光束探测物体，返回的一系列包含距离信息的数据点。搭载激光雷达的机器人通过对比两不同时刻的点云，来计算机器人位置和变化与朝向的改变。

激光雷达模型相对简单，距离测量相对准确，在阳光直射以外的情况下操作稳定，点云数据量小，可以低成本嵌入式设备中实时处理。同时，点云数据包含直接几何关系，使机器人路径规划和导航更加直观。

一般而言，激光雷达的价格比视觉传感器价格要贵。比如，应用于室外场景的激光雷达 Hokuyo 和 SICK，其价格在数万元人民币左右。广泛应用于室内场景的中低端激光雷达的价格在数千元左右，其价格与高端的工业级 CCD 相机相当，激光雷达的外形如图 2.2 所示。

图 2.2 激光雷达

综上所述，激光 SLAM 方案更加成熟，落地产品相对更多；但是相对于视觉传感器，激光雷达价格较为昂贵，长期工作会存在不稳定的隐患。更加经济可行方案是使用 Kinect 和 ASUS Xtion 摄像头，通过摄像头获取的三维点云来生成模拟激光扫描仪是更经济的 SLAM 实现方法。本文选用主要传感器为微软公司生产的 Kinect 相机。

2.2 SLAM 算法选择

SLAM（Simultaneous Localization and Mapping，同时定位与导航）是机器人搭载特定传感器（激光雷达或视觉传感器），在没有环境先验信息的前提下，在未知的环境中运动，并在运动中估计姿态变换与构建环境地图。一个 SLAM 流程可以这样理解：1）首先对未

知环境进行一次扫描，构建环境地图；2）保存环境地图；3）利用已知的环境地图进行定位、导航及路径规划。

2.2.1 Hector SLAM

Hector SLAM 算法[6]通过对比当前获得的点云数据与已得到的地图进行对比优化，估计占据栅格的概率和当前激光扫描数据在地图中的表达。该过程的扫描匹配过程是通过高斯-牛顿法实现进行求解计算的，用以获取激光数据映射到地图的变换(x, y, theta)。

优点：该算法不需要里程计数据的输入，应用场景包括空中无人机和非平坦区域的小车等。为避免出现局部最优而非全局最优的情况，Hector SLAM 的地图采用多分辨率的策略。

缺点：Hector SLAM 算法需要高精度的激光雷达数据作为输入。在实际的建图过程中，机器人的运动速度不能过快，将机器人的速度控制在较低的情况下建图效果才会比较理想。当我们的机器人有里程计数据时，不能够有效地应用。

2.2.2 GMapping SLAM

GMapping SLAM 算法[7]是应用极为广泛的激光 SLAM 方案，此采用了 Rao-Blackwellized 粒子滤波的方式。粒子滤波的方式需要大量的粒子来获取较好的实验结果，但这会引入计算量的增大。粒子是一个依据过程的观测逐步更新权重与收敛的过程，这种重采样的过程必然会代入粒子耗散问题，权重小的粒子会消失，权重大的粒子显著[8]。自适应重采样技术的引入会减少粒子耗散问题，从而减少机器人位姿在粒子滤波过程中的不确定性。缺点：需要里程计数据；没有回环检测的能力；无法适应无人机及地面小车不平坦区域。

表 2.1 激光 SLAM 方案对比

	Hector SLAM	GMapping SLAM
特征场景建图	效果差	效果较好
计算复杂度	较高	较低
缺点	传感器要求高	要求地面平坦

2.3 路径规划算法的地图类型选择

具有地图构建能力的扫地机器人更加智能化，根据构建的地图进行路径规划并用于导航，因此地图类型的选择是一个必须要考虑的问题。考虑到扫地机器人在 2 维平面上运动，所以只需考虑二维地图即可。适用于扫地机器人导航的地图一般有栅格地图和拓扑地图。

2.3.1 栅格地图

二维栅格地图由许多个小栅格组成，通过传感器获得的信息，判断当前栅格存在障碍物的概率，用此概率值给对应的栅格赋值，用来表示地图中哪里可以通过，哪里存在障碍物。一般而言，每个栅格有可通过、不可通过与未知三种状态，以表示该栅格内是否存在障碍物。栅格地图可用于 A*、D*等路径规划算法，对于一个确定的栅格，我们可以查询该栅格的状态来判断是否可以通过。

栅格地图的分辨率：即每个栅格代表的实际尺寸（m/cell）是一个重要的参数。如果分辨率过高，则会占用较大的存储空间，而且路径计算比较耗时；如果分辨率较低，则无法精确地描述环境信息。因此环境地图的分辨率要根据实际情况而定。2D 栅格地图如图 2.3(a)所示。

(a)二维栅格地图　　(b)拓扑地图

图 2.3 不同种类的地图

2.3.2 拓扑地图

拓扑地图是一种强调地图元素之间关系的地图，一种拓扑地图如图 2.3(b)所示。拓扑地图是一个图论中的图，由节点和边构成，其中节点表示地图中不同位置的子区域，边表示不同子区域之间的连通性关系。与 2D 栅格地图相比，拓扑地图的组织形式更为紧凑，同时导致了环境信息的极大缺失。一般而言，拓扑地图不能直接用于路径规划。

综上所示，我们选用 2D 栅格地图作为地图导航。在进行路径规划时，只需查询地图中栅格是否可以通过即可用于机器人的导航应用。

第 3 章 扫地机器人系统设计

为了验证 GMapping 算法的效果，我们基于 TurtleBot 移动机器人平台进行扫地机器人系统的设计。TurtleBot 是 Willow Garage 公司生产的一款具有优良性能的机器人开发平台。所用相机为微软公司的 Kinect 相机，移动底盘为 Yujin 公司的 Kobuki 底盘。

3.1 Kinect 深度相机

3.1.1 Kinect 深度相机简介

如图 3.1 所示，本系统选用的主要传感器为 Microsoft Kinect 深度相机。作为机器人的“眼睛”，除了 RGB 相机外，还有红外结构光发射器和红外结构光接收器，用于深度图像的采集。RGB-D 相机同时输出彩色图和深度图。

图 3. 1 Kinect 深度相机

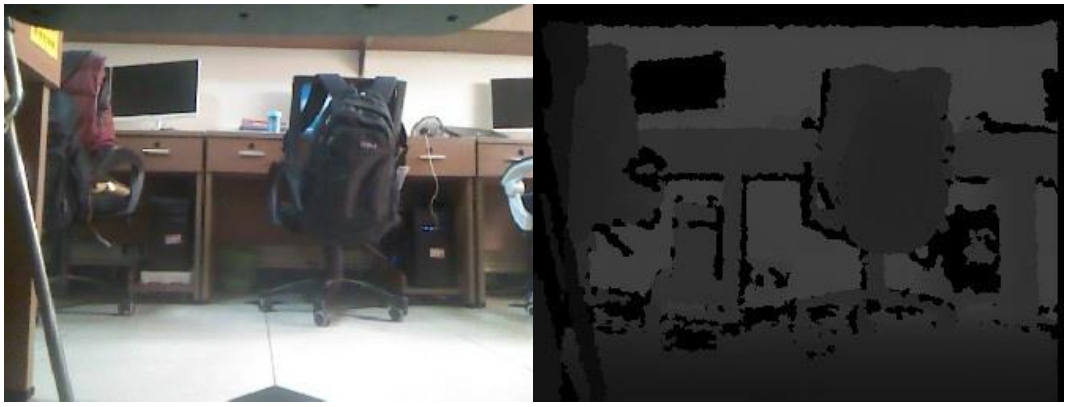
相机的 RGB 摄像机与红外结构光发射器、接收器以水平的方式排列。相机的详细参数见表 3.1，彩色图分辨率为 640×480，深度图分辨率为 320×240。

表 3. 1 Kinect 深度相机参数

颜色(Color)	分辨率(Resolution)	640×480
	帧率(FPS)	30FPS
深度(Depth)	分辨率(Resolution)	320×240
	帧率(FPS)	30FPS
检测范围(Range of Detection)		0.8~6.0m
深度误差(Depth Uncertainty)		2~30mm
角度(Angle)	水平(Horizontal)	57°
	垂直(Vertical)	43°

在 Ubuntu 系统中打开终端并启动 OpenNI 驱动程序，ROS 中使用 image_view 来查看

RGB 图像和深度图像，如图 3.2 所示。在 RViz 可视化软件中查看三维点云如图 3.3 所示。



(a)彩色图　　(b)深度图

图 3. 2 Kinect 相机采集的彩色图和深度图

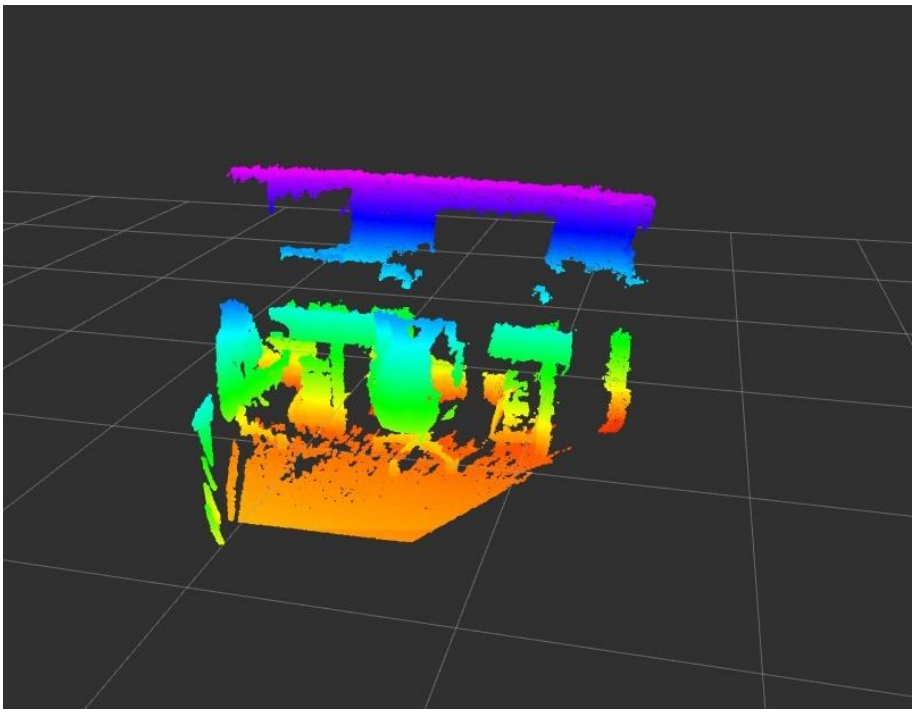


图 3. 3 Kinect 相机获得的点云图

3. 1. 2 针孔相机模型

针孔相机成像模型如图 3.4 所示，设$O-x-y-z$为相机坐标系，其中 O 为相机的光心所在位置。世界坐标中的一点P，其坐标为$[X,Y,Z]^T$，经过相机光心 O 投影后，落在相机的成像平面$O-x'-y'$上，成像点为P'，其坐标为$[X',Y',Z']^T$。已知成像平面到相机光心的距离（焦距）为f，根据相似三角形原理，可得下式：

$$\frac{Z}{f}=\frac{X}{X'}=\frac{Y}{Y'} \tag{3.1}$$

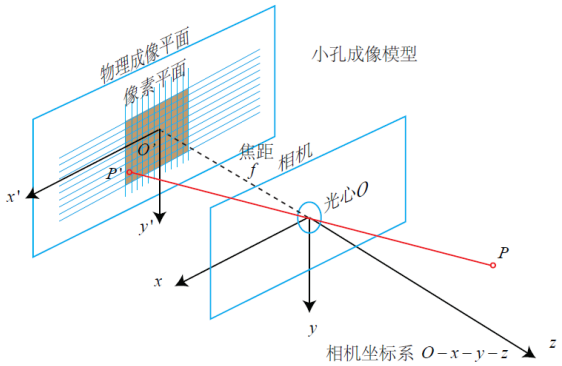


图 3.4 针孔相机模型

由式(3.1)可得：

$$\begin{cases} X' = f\dfrac{X}{Z} \\ Y' = f\dfrac{Y}{Z} \end{cases} \tag{3.2}$$

式(3.2)描述了现实世界中点 P 及其成像点之间的数学关系。图像中是一系列整齐排列的离散像素点，这需要在成像平面上对像进行抽样与量化。

为描述相机将现实世界中的坐标点映射到成像平面上的点的过程，我们设在物理成像平面上固定一个像素平面o－u－v。我们在像素平面得到了P'的像素坐标：$[u,v]^T$。

P'的坐标与像素坐标：$[u,v]^T$的关系如式(3.3)：

$$\begin{cases} u = \alpha X' + c_x \\ v = \beta Y' + c_y \end{cases} \tag{3.3}$$

将式(3.2)代入式(3.3)，可得：

$$\begin{cases} u = f_x\dfrac{X}{Z} + c_x \\ v = f_y\dfrac{Y}{Z} + c_y \end{cases} \tag{3.4}$$

为了更加直观地表示，我们将式(3.4)表示为矩阵的形式：

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{Z}\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \triangleq \frac{1}{Z}\boldsymbol{KP} \tag{3.5}$$

安装传统的习惯，把 Z 挪到左侧，得到式(3.6)：

$$Z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \triangleq \boldsymbol{KP} \tag{3.6}$$

式(3.5)与(3.6)中矩阵 K 为相机内参。其中f_x和f_y为 x 方向和 y 方向的焦距，通常有$f_x \approx f_y$成立。

3.1.3 深度图转激光

如图 3.5 所示为深度图转激光的示意图，对任意给定的深度图像素点$[u,v]^T$，对应的深度值为d，其转换为激光的步骤为：

1）对于深度图中的像素点$[u,v]^T$，其深度值为 d，由式(3.6)反推对应的相机坐标系中点$p_c=[x,y,z]^T$。

$$\begin{cases} x=\dfrac{u-c_x}{f_x}z \\ y=\dfrac{v-c_y}{f_y}z \\ z=d \end{cases} \tag{3.7}$$

2）计算直线AO和CO之间的夹角∠AOC，计算公式如下：

$$\theta=\operatorname{atan}\frac{x}{z} \tag{3.8}$$

3）将∠AOC映射到相应的激光数据槽中。激光的角度最小为α，最大值为β，激光束共计 N 份，我们用laser[N]来表示激光数据。那么点相机坐标系中的点p_c投影到数组 laser 中的索引值n可通过下式计算：

$$n=(\theta-\alpha)/((\beta-\alpha)/N)=\frac{N(\theta-\alpha)}{(\beta-\alpha)} \tag{3.9}$$

laser[n]的值即为p_c在 x 轴上的投影点 C 到相机光心 O 的距离 r，即

$$laser[n]=r=OC=\sqrt{z^2+x^2} \tag{3.10}$$

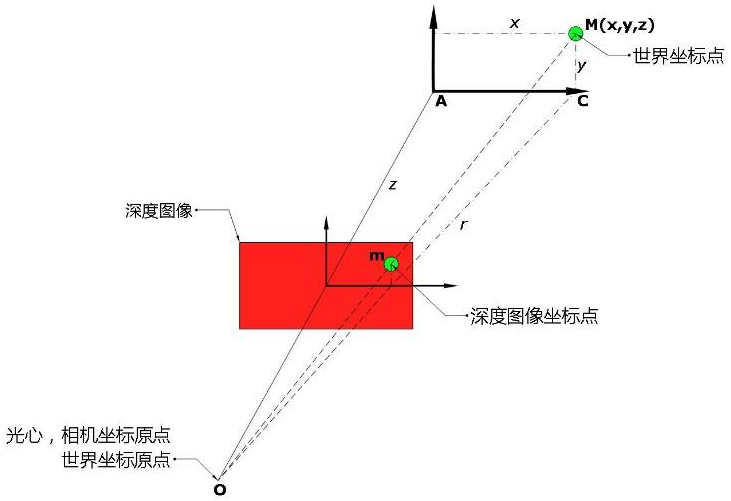


图 3. 5 深度图转激光示意图

3.2 Kobuki 底盘及其控制

3.2.1 Kobuki 底盘简介

Kobuki 是韩国 Yujin Robot 公司生产的一款机器人底盘，被广泛用作移动机器人的研究平台，其外观如图 3.6 所示。一些常用参数如表 3.2 所示，最大平移速度为 0.7m/s，一般实验过程中设为 0.5m/s 即可。控制 Turtlebot 运动，我们使用已安装 ROS 和 Ubuntu 系统的笔记本电脑，用 USB 线连接移动底盘和笔记本电脑。

图 3. 6 Kobuki 底盘

表 3. 2 Kobuki 底盘参数表

项目	参数值	说明
最大平移速度	70cm/s	无
最大旋转速度	180deg/s	大于 110deg/s 时 陀螺仪性能会下降
嵌入式处理器	单片机 STM32F103VC	ARM Cortex-M3 处理器
通信方式	USB 接口或串口 RX/TX	无

3.2.2 扫地机器人运动模型

路径规划和导航都需要对机器人的运动位姿进行改变，所以机器人的运动姿态控制是一个基本问题，下面是对机器人运动模型的简介。

扫地机器人的形状大多为圆饼式，其模型如图 3.7 所示，万向轮可以向任意方向旋转；两个驱动轮负责机器人的直行、后退和左右转。图中 C 为扫地机器的中心，D 表示扫地机器的直径。

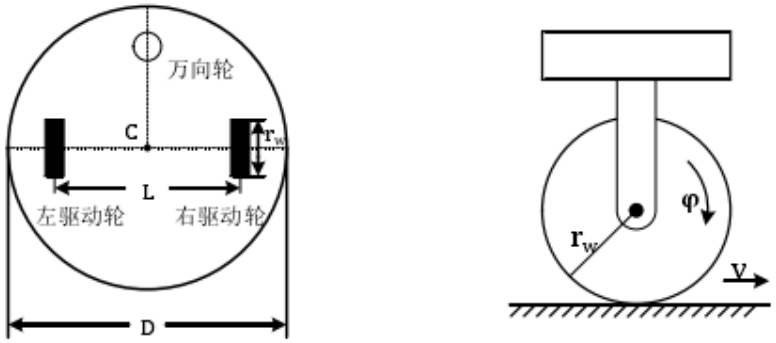


图 3. 7 扫地机器人运动结构图

扫地机器人的车轮运动结构如图 3.7 所示，驱动轮的半径为r_w，若转动的角速度为φ，那么驱动轮的线速度为$v = r_w\varphi$。

将扫地机器人左右驱动轮的角速度记为φ_L和φ_R，则左右驱动轮的速度分别为$v_L = r_w\varphi_L$和$v_R = r_w\varphi_R$。扫地机器人的运动模型如图 所示，通过调节两个轮子的速度控制扫地机器人前进、后退和左右转弯等操作。

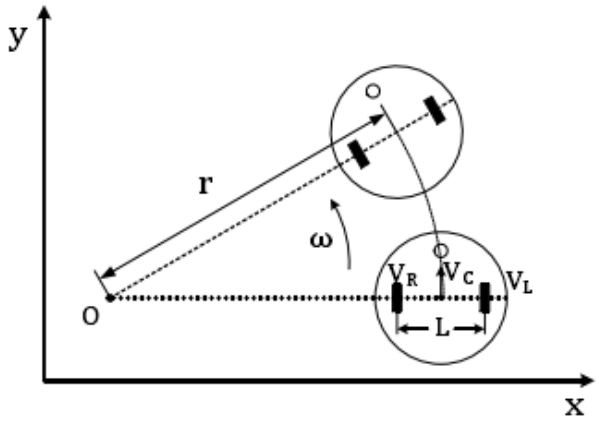


图 3. 8 机器人转弯示意图

考虑到扫地机器人的移动速度较慢（一般不超过 0.5m/s），因此可以忽略横滑影响，根据运动学的知识，扫地机器人中心点的速度v_C可表示为：

$$v_C = \frac{v_L + v_R}{2} \tag{3.11}$$

扫地机器人做半径为 r 的圆弧运动，O 为扫地机器人的瞬时转动中心，由刚体力学知识可知 O 与两个驱动轮的中心在一条直线上。L 为左右两轮，左右两驱动轮相对于中心点 O 点的角速度相等，可得：

$$\frac{v_L}{r - \frac{L}{2}} = \frac{v_R}{r + \frac{L}{2}} \tag{3.12}$$

经计算可得，扫地机器人做圆弧运动的半径 r 为：

$$r = \frac{L(v_R + v_L)}{2(v_R - v_L)} \tag{3.13}$$

圆弧运动的角速度ω为：

$$\omega = \frac{v_L}{r - L/2} = \frac{v_R - v_L}{L} \tag{3.14}$$

3.3 路径规划点到点路径规划

前面提到，具有自主导航能力的扫地机器人是智能化水平比较高的，更具体地，路径规划算法的性能直接影响机器人智能化程度的高低。路径规划是在已知地图的环境中，计算出从起始点到目标点的一条最优路径。本节主要考虑两种算法：迪杰斯特拉算法和 A*算法。

3.3.1 迪杰斯特拉算法

迪杰斯特拉（Dijkstra）算法[9]由荷兰计算机科学家迪杰斯特拉于 1956 年提出。迪杰斯特拉算法用于求解有向赋权图的单源最短路径问题，此算法采用了广度优先搜索的策略，即距离源点的顶点先被访问到，距离源点远的顶点后被访问。迪杰斯特拉算法在每一次寻找最近点时采用贪心策略，即每次查找距离源点最近的顶点。采用贪心策略也决定了此算法不能用于存在负权重的图。

表 3.3 迪杰斯特拉算法流程

输入：有向赋权图$G = (V, E)$; 源顶点V_S
过程：集合 Visited 记录未访问节点的集合；Unvisited 记录未访问节点集合
初始：Visited={}, Unvisited=V；
设置源顶点V_S到V_S距离为 0，其他顶点到V_S距离为∞（正无穷）
1: Repeat
2: 将集合 Unvisited 中选取距离V_S最近的点设为当前节点V_C
判断V_C是否为V_t，如果是则表示已找到最短路径 ,否则执行 3
3: 检查当前节点V_C未访问的邻居顶点，如果没有则执行 6
4: 计算从顶点开始到每个邻居顶点的距离
5: 如果计算所得距离小于已知距离，那么更新最短距离（与上一个顶点）
6: 将当前顶点添加至 Visit 列表中
7: 直至所有顶点全部被访问

迪杰斯特拉算法步骤如表 3.3 所示，通过 Dijktra 算法求解图G中的最短路径时，需要指定起点V_S。此外，引入两个集合Visited和Unvisited。集合Visited中存放已求出最短路径

的顶点，集合Unvisited中存放还没有求出最短路径的顶点。初始时，集合Visited中只有起点V_S；除V_S外的其他顶点都在集合Unvisited中，并且Unvisited中顶点的路径是“起点 Vs 到该顶点的路径”。然后，从Unvisited中找出路径最短的顶点，并将其加入到Visited中；接着，更新Unvisited中顶点和顶点对应的路径。然后，再从Unvisited中出路径最短的顶点，并将其加入到Visited中；接着，更新Unvisited中的顶点和顶点对应的路径。重复上述操作，直至集合Visited中包含所有的顶点。

3.3.2 A*算法

在机器人领域与计算机领域中，A*算法[10]被广泛应用于路径搜索与图遍历。A*算法可以看做是迪杰斯特拉算法的一种扩展，算法采用的启发式搜索可以在保证找到一条最优路径的同时，还提高了算法效率。

表 3.4 A*算法流程

OPEN 列表：搜寻节点的过程中保存未检测的节点列表
CLOSE 列表：已被检测的节点列表
comeFrom 列表：保存子节点和父节点，最终生成的路径在此列表中
把起点 S 加入到 OPEN 列表中
do{
搜寻 OPEN 列表中f(n)值最小的点设为当前节点N，并添加至 CLOSE 列表
遍历当前节点N相邻的所有候选节点 V
if (V 是障碍物节点 或 已在 CLOSE 列表中)
继续搜寻下一相邻候选节点
if (V 不在 OPEN 列表中)
将此候选节点加入 OPEN 列表，更新父节点，
计算其g(n)、h(n)和f(n)距离值
if (V 已经在 OPEN 列表中)
使用g(n)作为评判标准，来检测新的路径，判断是否有更低的g(n) 将它的父节点换为当前节点，计算当前节点的h(n)和f(n)值
使用 comeFrom 保存父节点
}while(目标节点在 OPEN 列表中，表示已找到起点到目标点的路径)
若 OPEN 列表为空，表示没有可以寻找的路径
最后把 comeFrom 保存的所有父节点从起点开始以此连接起来，此路线即为 A*算法所求路径

在此算法中，以$g(\mathrm{n})$表示从起点到中间顶点 n 的实际距离，$h(\mathrm{n})$表示从中间顶点 n 到目标顶点的估算距离，$h(\mathrm{n})$的选取视不同应用而定，常见的评估函数有：曼哈顿距离或欧几里得距离。A*算法的估计函数为：

$$f(\mathrm{n}) = g(\mathrm{n}) + h(\mathrm{n}) \tag{3.15}$$

在已构建的地图中，将起始点记为$\mathrm{S}(S_x, S_y)$，将中间点记为$\mathrm{N}(N_x, N_y)$，目标点记为$\mathrm{T}(T_x, T_y)$，若$g(\mathrm{n})$选用欧式距离，$h(\mathrm{n})$选用曼哈顿距离，则$f(\mathrm{n})$可表示为：

$$f(\mathrm{n}) = \sqrt{(S_x - N_x)^2 + (S_y - N_y)^2} + |T_x - N_x| + |T_y - N_y| \tag{3.16}$$

3.4 全覆盖路径规划算法

对于扫地机器人而言，对区域进行完全遍历是正常的清洁方式，本节对全覆盖路径规划算法进行介绍，重点介绍 Boustrophedon 全覆盖算法。

3.4.1 全覆盖路径规划算法概述

全覆盖路径规划是对给定的地图进行完全遍历的路径规划，一些相关的应用包括移动地板清洁机器人，割草机和自动作业的农业机械等[11]。环境地图的建模有多种类型，例如几何法，可视图法，单元分解法和梯形分解法和栅格法。本文进行全覆盖路径规划算法选用的地图建模为单元分解法[12]。

为使扫地机器人对整个区域进行全覆盖，基于单元分解法的全覆盖路径规划算法首先对环境地图进行子区域划分，然后对子区域内部通过简单运动来实现全覆盖，最后将所有子区域有序连接起来，从而得到完整的全覆盖路径。

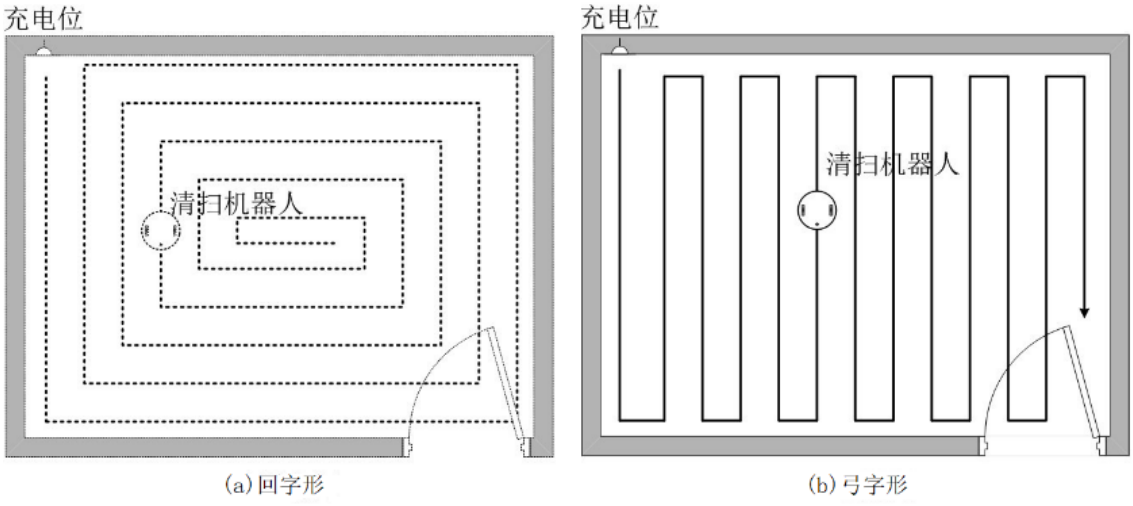


图 3.9 子区域覆盖方法

单元分解得到的子区域通常不含独立的障碍物，所以子区域的完全遍历可以通过简单规划方式实现，常见的两种方式：回字形清扫和弓字形清扫，其示意图如图 3.9 所示。回字形清扫的起点在子区域的一角，扫地机器人沿区域的边界按顺时针或逆时针的方向不断往内做螺旋运动，直到整个区域被完全覆盖；工字形清扫的起点和回字形清扫的起点相同，都是在区域的一角，扫地机在指定区域内做往复运动，终点在区域对角线或与起点在一条边上。

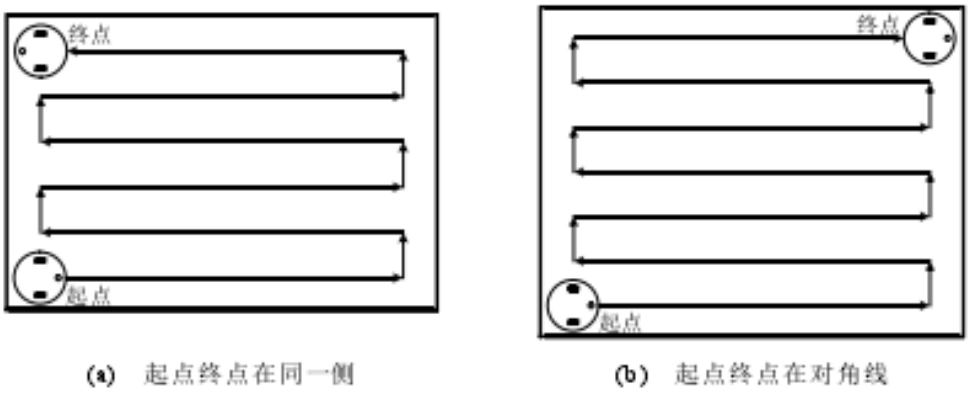


图 3. 10 沿长边的往复式路径

在弓字形清扫方式中，有两种方法进行遍历，一种是沿着较短的边行进，如图 3.10(b)所示；另外一种是沿着较长的边行进，如图 3.10 所示。这两种方式都可以完成子区域的全覆盖遍历，但是考虑到直行和转弯耗时不同，一般而言，机器人转弯占用的时间更多。沿着短边行进的方式转弯次数更多，因此更加耗时。所以我们选用沿子区域的长边方向行进的方式。

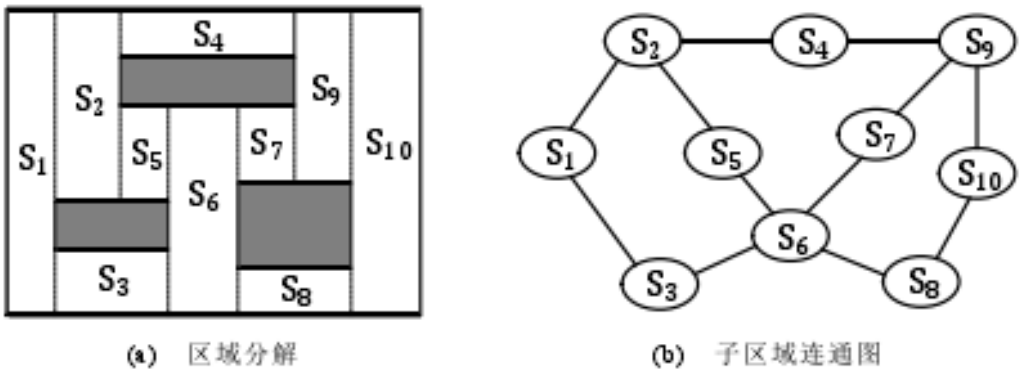


图 3. 11 子区域分解与连通

子区域之间的关系常用图论的模型进行表示，将每个子区域称为一个节点，根据每个节点与周围节点的连接关系可以构造出一张连通图。子区域的连通顺序通过图论算法进行确定，为了使所有子区域全部被遍历到，往往会出现某些区域重复遍历，增加了清扫的重复率。由于往复式路径的终点不确定，在进行子区域的连接时还要考虑当前区域的终点与

下一区域的起点之间的连接，如果算法设计不好，则会出现大量的重复路线；内螺旋路径同样也存在这一问题，由于当前区域的终点在区域中心，机器人移动到下一区域的起点必定会出现重复路线。

基于单元分解法的全覆盖路径规划需要综合考虑地图分解方法、子区域内覆盖和子区域连接这三个问题。子区域的复杂程度影响到区域内全覆盖的路径设计，子区域的数量直接影响到连接算法的复杂程度，子区域的起点终点的确定和连接算法会对机器人的重复率产生影响，所以该方法的难点在于如何平衡这三者之间的关系。

3.4.2 Boustrophedon 覆盖算法

该方法将给定的地图划分为若干个子区域，在每一个子区域内都是没有障碍物的，因此可以使用简单路径规划方式完成全覆盖。对于每一个子区域，规划的路径是向上，向下并平行于子区域的上边界和下边界。Boustrophedon 算法[13]流程如下所示：

1. 使用 Sobel 算子计算每个像素的梯度方向。利用该信息，可以找到最适合计算子区域的方向，比如说：当长的子区域出现时，可以通过此方式旋转子区域。

2. 通过给定的地图扫描切片（莫尔斯函数）并检查该线的连通性，即有多少个连接的段。如果连通性增加，即更多的段产生，则一个 IN 事件发生，打开新的独立单元格；如果联通性减小，说明段合并，一个 OUT 事件发生，将两个子区域合并。如果有事件发生，算法将沿当前路径检查关键点，即触发事件的点。从这些开始绘制单元格的边界，从 CP 开始并向左或向右移动直至碰撞黑色像素点。

3. 通过扫描切片确定所有的子区域之后，通过使用 OpenCV[14]中cv :: findContours() 找到这些轮廓。这为每个子区域提供一组点，用于从每个子区域中创建多边形。

4. 在所有的多边形被创建后，规划穿过所有子区域的路径使得所有的区域被覆盖到。为了实现此目标，首先规划出一条穿过所有子区域的全局路径，使用 TSP 问题的公式。这产生子区域的最优访问顺序。接下来对于每个子区域确定一个 boustrophedon 路径，该路径沿着单元来回穿过，沿着子区域边界的水平路径之间，确保所有子区域被覆盖。找到每个子区域的长边，使此边与 x 轴对齐。这将产生较长但是比较少的边，这对面积小但是较长的子区域是有利的。子区域路径的起点是由上一子区域的终点决定的，这种做法使得两子区域间距离最小。以旋转的方式确定子区域路径，因此在最后一步中，将子区域路径转换为最初转换的子区域，然后将其嵌入至全局路径中。

5.上述步骤产生关于视野的路径，欲将产生的路径与机器人的路径关联起来，需要做一些变换。如果机器人的位姿不在自由空间中，通过在视野周围的半径上找到它来生成另一个可访问点，使得到最后机器人位置的距离最小。如果不希望这样，则必须将相应的布尔值设置为 false 即可。

第 4 章 实验结果

4.1 扫地机器人实验环境搭建

根据第三章所述的系统设计方案，搭建如图 4.1 所示的移动机器人实验平台。选用的计算平台为戴尔灵越 5559 笔记本电脑，配有 4 核 Intel Core i5-6200U 处理器。传感器为 Kinect 一代深度相机，里程计数据来自移动底盘的车轮编码器。

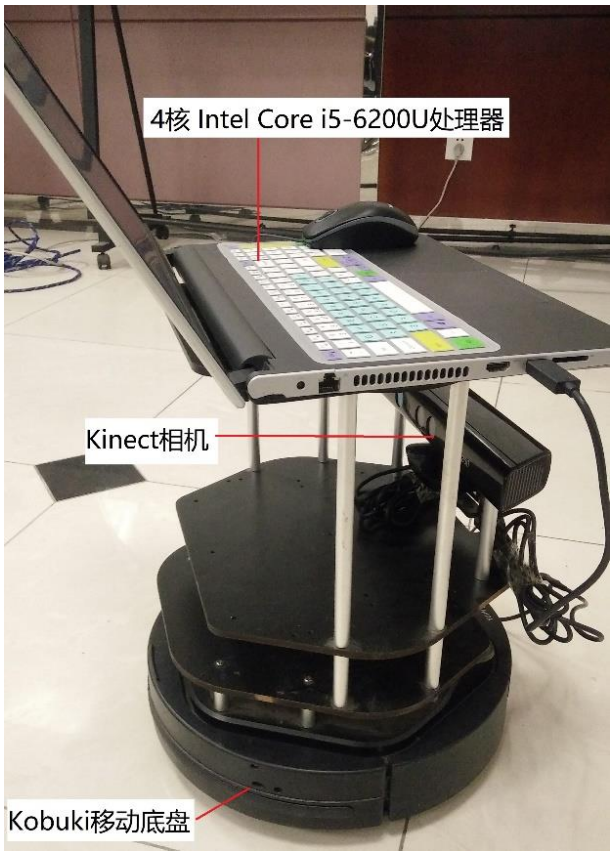


图 4.1 移动机器人实验平台

笔记本电脑上运行安装了 ROS Kinetic[15]的 Ubuntu16.04（Linux）操作系统，编程语言为广泛使用的面向对象的编程语言 C++，以及脚本语言 Python。

为了验证所用地图构建算法的实际性能，本文的实验环境为华中科技大学逸夫科技楼 903 会议室。具体环境如图 4.2 所示，会议室中间有一张会议桌，靠墙的位置摆放有桌子和若干把椅子，构成了一个封闭的实验环境。图中的白色线条为实验环境的边界。

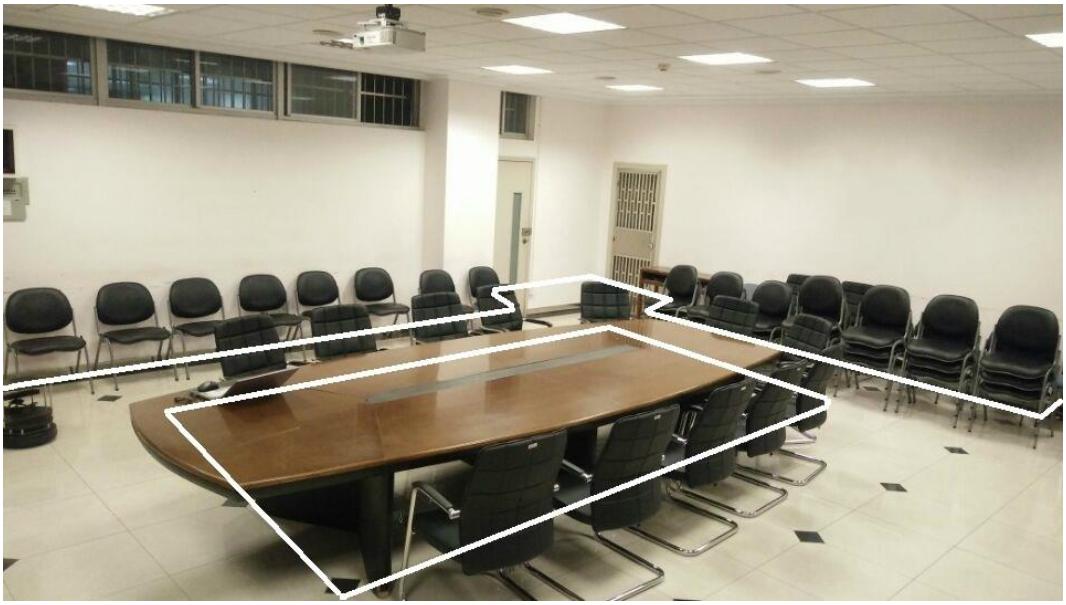


图 4.2 实验环境场景

4.2 环境地图构建实验

首先控制机器人在如 4.2 所示的实验环境内进行运动，在实验环境的不同位置，摄像头获取不同的深度图数据，把深度图转换为激光扫描数据，

在环境地图构建阶段，我们需手动控制机器人在实验环境中遍历一周，通过 Kinect 相机获得实验环境的三维点云来生产模拟激光扫描数据，激光扫描数据作为 SLAM 算法的输入，完成实验环境的地图构建。

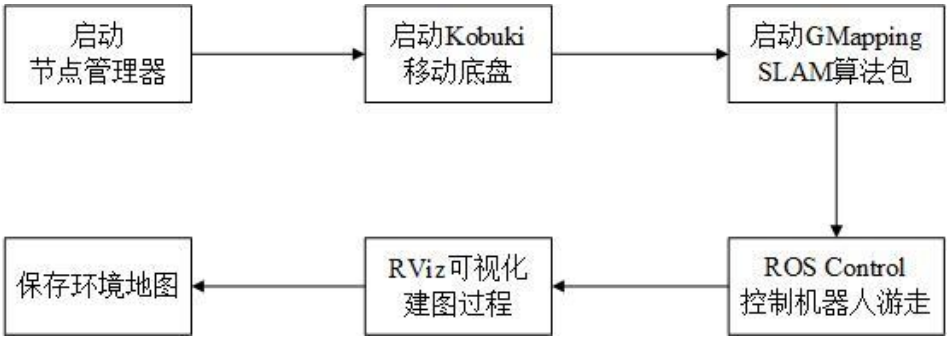
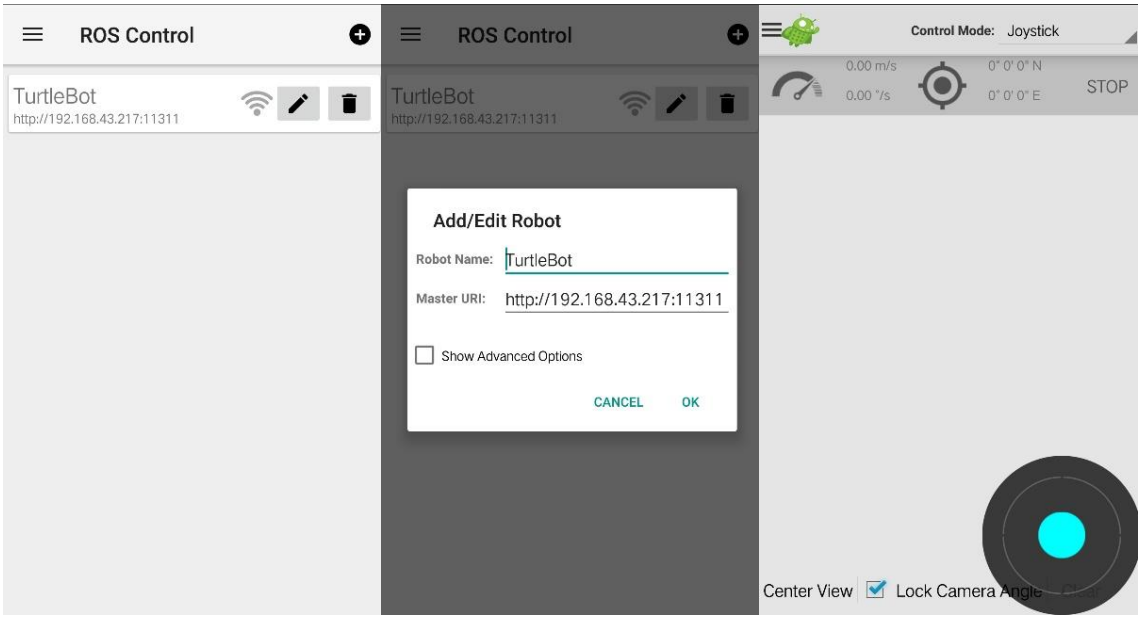


图 4.3 地图构建流程图

因为本实验基于 ROS 系统进行设计，首先启动节点管理器，然后启动 Kobuki 移动底盘，再者启动 GMapping SLAM 对应的 launch 文件，此 launch 文件会同时启动 Kinect 的 OpenNI 驱动、深度图转激光数据等。图 4.4(a)为 ROS Control APP 的机器人管理界面，图 4.4(b)为 IP 地址与端口号的配置界面，需保证移动端与机器人处于同一局域网中，APP 向机器人发送即可控制机器人运动，图 4.4(c)为操纵杆的控制界面，相比于直接键盘控制，这种控制方式很友好。



(a)ROS control　　(b)IP 地址及端口配置　　(c)操纵杆控制界面

图 4. 4 ROS Control APP 界面

通过 ROS 的可视化工具 RViz 观测 GMapping SLAM 算法的建图过程，如图 4.5 所示，图中的黑点为 Turtlebot 机器人的所在位置。随着扫地机器人在实验环境的不断游走，环境地图不断构建，最好将构建地图进行保存。

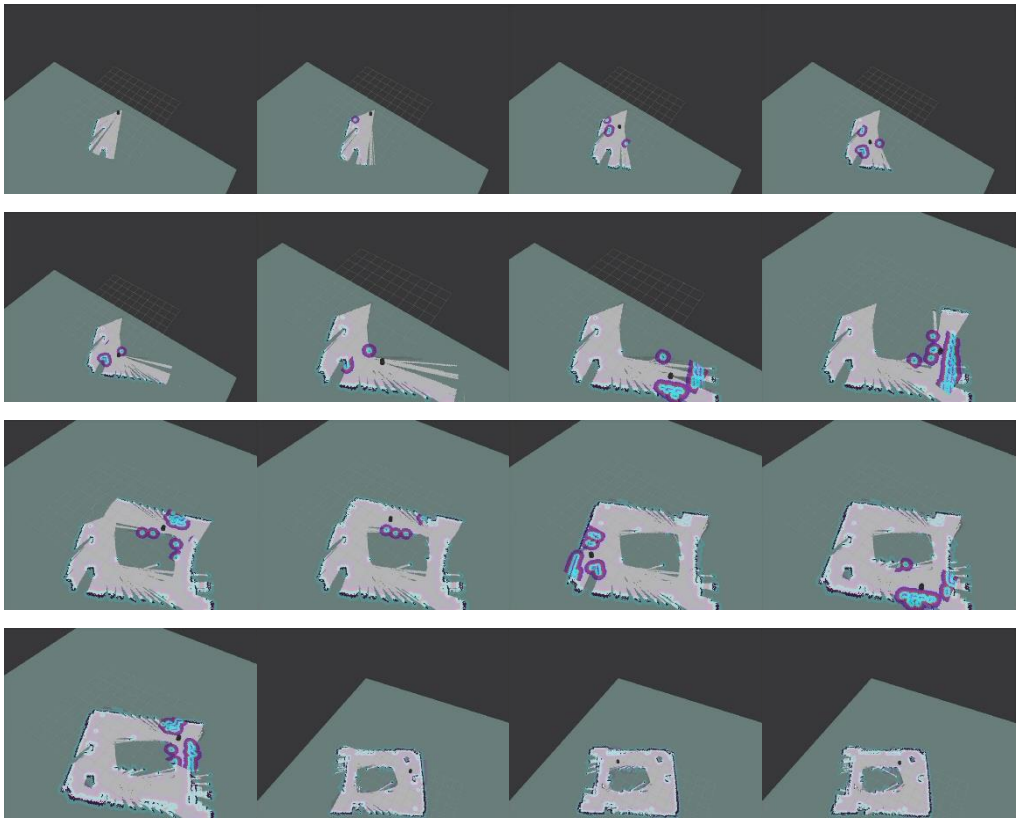


图 4. 5 实验环境的 GMapping 建图过程

如图 4.6 所示为 GMapping 算法构建的环境地图。图中灰色区域为未知区域，黑色区域为不可通过区域，即此处存在障碍物，白色区区域为可通行区域。图中红色箭头所指方向即为实验环境中照片的拍摄方位。

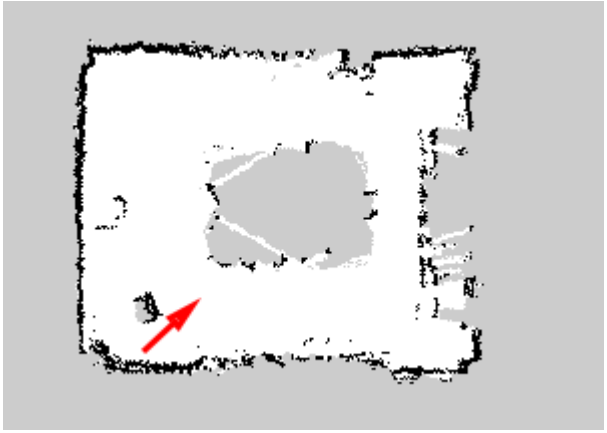


图 4. 6 实验环境的 GMapping 建图结果

可以看出机器人的全局地图与实际的实验环境轮廓基本符合，但是在角落或细节描述上还有缺陷，主要原因是 Kinect 的可视角度小和噪声点多造成的，如果使用高精度、宽视角的激光雷达可有效解决此问题。

4. 3 点到点路径规划实验

4. 3. 1 算法仿真

基于图 4.5 的建图结果，对地图进行二值化处理（灰度值大于 250 记为可通行区域，否则记为不可通行），白色区域为可通行区域，黑色区域存在障碍物，指定起始点（100，50）与目标点（200，150），A*算法绘制的规划路径如图 4.7 所示。

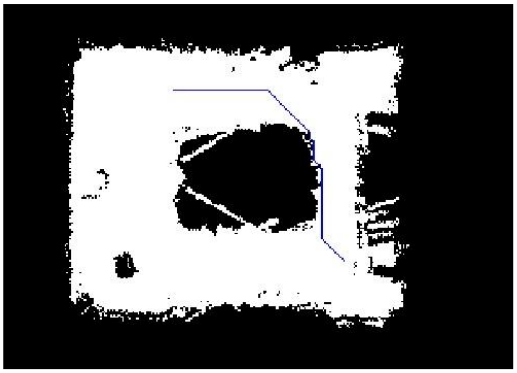


图 4. 7 A*算法仿真图

图 4.6 中产生的路径紧贴障碍物边缘，在实际的导航应用中，考虑到机器人本身的尺寸半径，将地图中障碍物进行扩张，避免与地图边缘或障碍物发生碰撞。

4.3.2 实际效果

扫地机器人的导航示意图如图 4.8 所示，图中右上角的绿色箭头为给定的目标终点，机器人根据已知的环境地图和当前所在位置，通过 A*算法计算出一条最优路径，如图 4.9 所示。

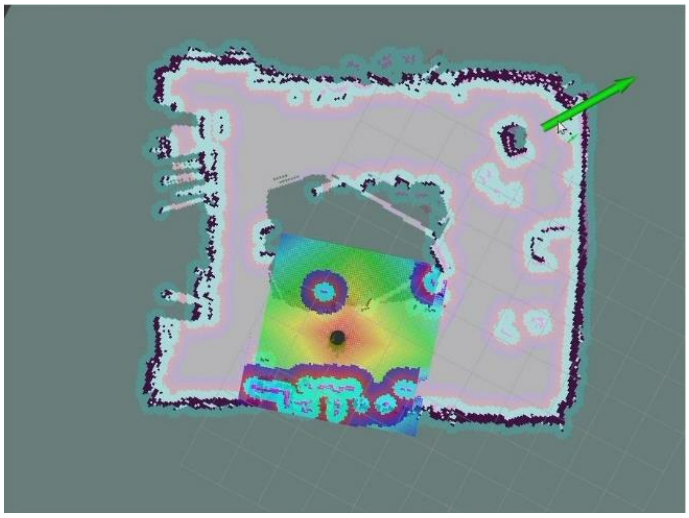


图 4.8 机器人导航图

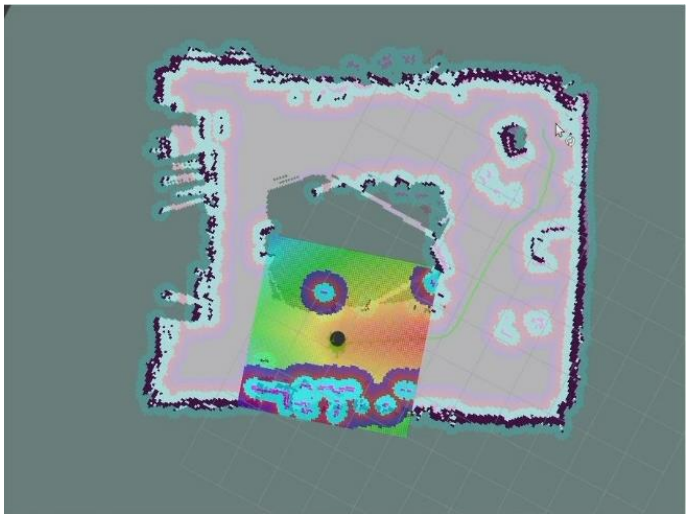


图 4.9 机器人路径规划图

实验过程中，我们发现机器人对动态障碍物的处理能力有限，本身的一个原因是所用 Kinect 相机的检测范围为 0.8~6.0m，即距离 0.8m 以内的范围处于视野盲区；第二个原因是地图分辨率大小的选取，如果栅格数量过多，则会导致算法的计算量增大，系统的

实时性不好。因此对环境中动态障碍物的躲避和快速到达目的地，二者要进行权衡。

4.4 全覆盖路径规划实验

在实际应用中，相对于覆盖率最优但是相对复杂的路径，扫地机器人的使用者更倾向于次优但是易于实现的路径。本文所采用的 Boustrophedon 算法在设计过程中尽可能少地进行转弯，以提高清扫的效率[16]。

全覆盖路径规划是扫地机器人与其他移动机器人相区别的规划方式。在已知环境地图的前提下，给定一个起始点，根据全覆盖路径规划算法规划的路径将整个区域覆盖。为验证我们在 3.4.2 小节所描述算法的性能，在实验环境地图上进行实验。

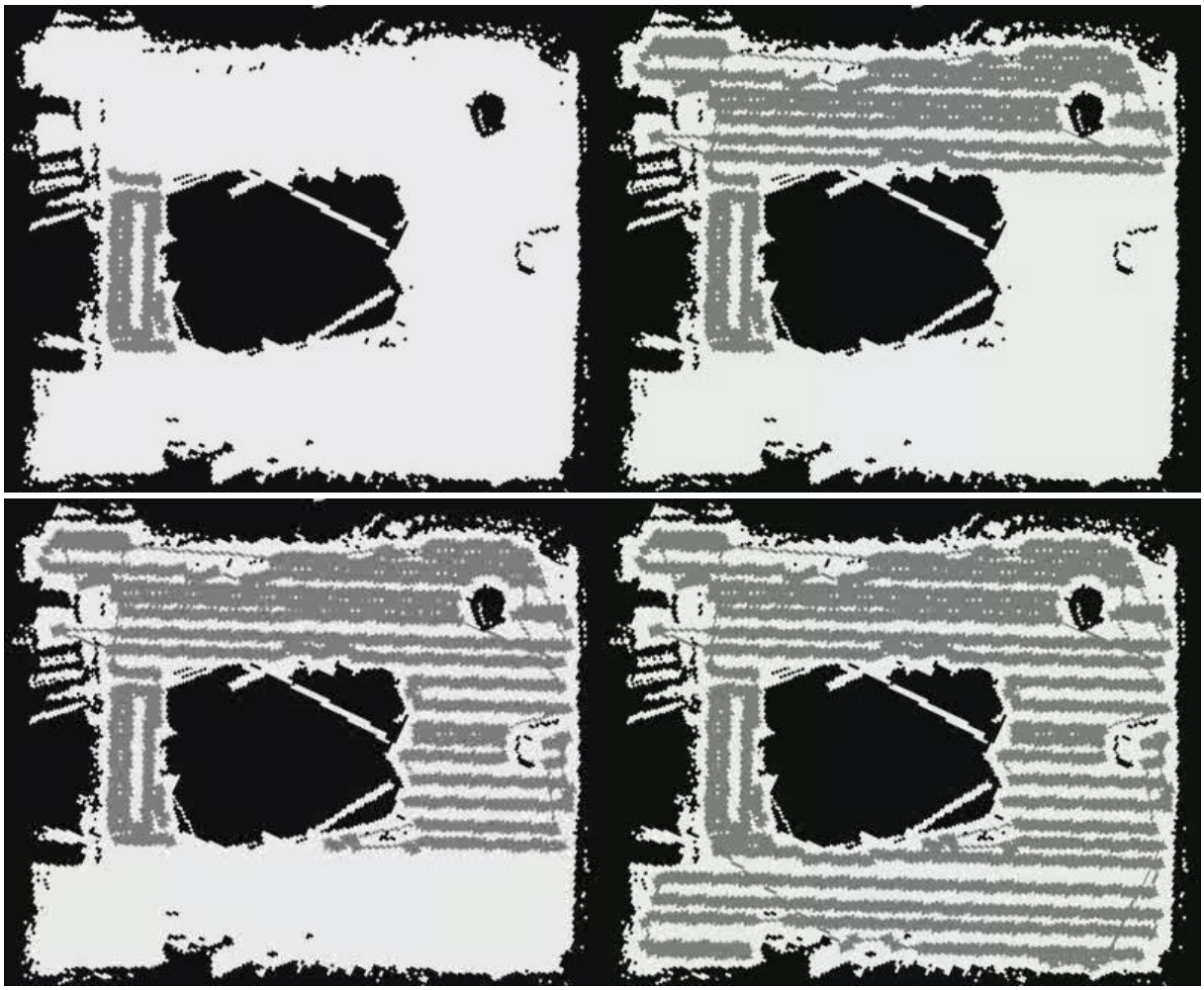


图 4.10 全覆盖路径规划结果图

全覆盖路径规划的结果如图 4.10 所示，首先将整个遍历区域分解为 4 个子区域，每个子区域的边界基本与环境边缘平行，在每一个子区域内，通过往返式的规划方式完成覆盖。全覆盖路径规划算法在较小重复率的前提下可实现区域的完全覆盖。

第 5 章 总结与展望

5.1 全文总结

本为首先基于 Turtlebot 机器人平台和 ROS 中 GMapping 算法包进行环境地图构建实验，得到构建的地图之后，利用 A*最短路径算法进行路径规划并用于导航，对于全覆盖路径规划，我们使用的算法在数据集和自己构建的地图上效果良好。对机器人运动学模型和 SLAM 算法、路径规划算法进行详细的阐述和方案对比。

本文首先对扫地机器人的国内外发展现状进行调研，调研了移动机器人路径规划领域的相关算法。

第二章是传感器与 SLAM 方案的对比选择，综合考虑选用 Kinect 深度相机；考虑到 Hector SLAM 依赖于高精度数据且不能利用里程计数据，我们选用 GMapping SLAM 算法。

第三章对机器人实验平台中 Kinect 相机和移动底盘进行详细阐述，包括深度图数据转激光数据的原理介绍，机器人底盘的运动模型介绍。也包括最短路径算法：Dijkstra 算法和 A*算法的算法流程。

第四章在实际的会议室场景完成环境地图构建、路径规划与导航实验。

5.2 工作展望

考虑到智能化是家用扫地机器人必然的发展方向，且智能化不仅仅包含导航和路径规划，也包括对家庭环境的识别等。考虑到本方案采用 RGB-D 相机作为主要传感器，因此考虑实现扫地机器人对家庭环境的识别，可能的改进有：识别宠物的排泄物；识别特定的家具环境，如拖鞋等；识别室内清洁容器，如垃圾桶；识别特定形态的物品，如各类线材等[17]。另外识别的物体信息如何辅助机器人的导航避障，也是需要思考的问题。

因为本文只是对 SLAM 算法和路径规划算法进行验证性实验，所以计算平台选用的是自己的笔记本电脑。后期可考虑选用支持 ROS 和 Ubuntu 系统的树莓派开发板作为控制中心和计算中心。

参考文献

[1] 陈玉. 基于 SLAM 的扫地机器人控制系统研究[D].哈尔滨工业大学,2017.

[2] 高斌. 室内智能扫地机器人的关键技术研究[D].重庆大学,2017.

[3] 李敏. 基于视觉的扫地机器人导航系统设计与实现[D].中国科学技术大学,2018.

[4] 王福安. 轮式移动机器人路径规划算法实现与优化[D].东南大学,2018.

[5] 高翔，张涛，颜沁睿，等. 视觉 SLAM 十四讲：从理论到实践[M]. 北京：电子工业出版社，2017.

[6] S. Kohlbrecher, O. von Stryk, J. Meyer and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, 2011, pp. 155-160.

[7] G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," in IEEE Transactions on Robotics, vol. 23, no. 1, pp. 34-46, Feb. 2007.

[8] 余彤. LiDAR/INS 组合的室内定位与制图（SLAM）算法改进[D].武汉大学,2017.

[9] Mark Allen Weiss. 数据结构与算法分析——C 语言描述[M]. 冯舜玺，译.北京：机械工业出版社，2016:224-228.

[10] 陆皖麟,雷景森,邵炎.基于改进 A~*算法的移动机器人路径规划[J].兵器装备工程学报,2019,40(04):197-201.

[11] R. Bormann, F. Jordan, J. Hampp and M. Hägele, "Indoor Coverage Path Planning: Survey, Implementation, Analysis," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 1718-1725.

[12] 任云天. 扫地机器人避障全覆盖路径规划算法的研究与设计[D].华中科技大学,2019.

[13] H. Choset, E. Acar, A. A. Rizzi and J. Luntz, "Exact cellular decompositions in terms of critical points of Morse functions," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 2270-2277 vol.3.

[14] 毛星云. OpenCV3 编程入门[M]. 北京：电子工业出版社，2015.

[15] 奥凯恩 M. 杰森. 机器人操作系统浅析[M]. 肖军浩. 北京：国防工业出版社，2016.

[16] R. Bormann, F. Jordan, W. Li, J. Hampp and M. Hägele, "Room segmentation: Survey, implementation, and analysis," 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, 2016, pp. 1019-1026.

[17] J. Huang et al., "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 3296-3297.