基于ROS移动机器人SLAM算法的研究

赵贺杨，褚慧慧，周宏胭，张裕磊
（南阳理工学院智能制造学院，河南 南阳 473000）

摘　要： 随着科技的发展，导航技术应用得越来越广泛。深入探索移动机器人在自主导航领域中如何实现快速且精确的导航，并针对地图构建核心问题提出解决方案。基于ROS创建URDF文件构建机器人模型，并在Gazebo中搭建仿真环境，实现Gmapping、Hector-SLAM、Cartographer 3种SLAM算法，构建高质量地图，通过对比得出不同算法的适用场景。

关键词： ROS移动机器人；Gazebo仿真；SLAM算法
中图分类号： TP242　　**文献标识码：** A　　DOI：10.19769/j.zdhy.2025.12.010

Research on SLAM Algorithms for ROS-Based Mobile Robots

ZHAO Heyang, CHU Huihui, ZHOU Hongyan, ZHANG Yulei
(College of Intelligent Manufacturing, Nanyang Institute of Technology, Nanyang, Henan 473000, China)

Abstract: With the development of technology, the application of navigation technology is becoming increasingly widespread. This paper deeply explores how mobile robots can achieve fast and accurate navigation in the field of autonomous navigation, and proposes solutions to the core problem of map construction. This paper creates a URDF file based on ROS to construct a robot model, and sets up a simulation environment in Gazebo to implement three SLAM algorithms: Gmapping, Hector-SLAM, and Cartographer. This paper builds high-quality maps and compares the applicable scenarios of different algorithms.

Key words: ROS mobile robots; Gazebo simulation; SLAM algorithm

0 引言

SLAM技术是人工智能与机器人领域中备受瞩目的研究方向，在自动驾驶和机器人领域内具有广泛的应用。SLAM技术凭借多样化的算法，为各项关键功能的实现提供了丰富的选项。依据不同的核心算法、应用场景以及传感器的使用，SLAM技术可以被进一步细分，如利用激光雷达实现的2D和3D SLAM、基于粒子滤波的Gmapping算法、借助深度摄像机技术的RGBD SLAM[1]等。

研究涵盖环境感知、SLAM地图构建、导航控制多个方面，针对2D激光SLAM中的Gmapping、Hector-SLAM和Cartographer建图方法进行仿真实现和优化。通过对比3种算法的精度、运算量和建图时间，探究出3种算法分别适用的场景，为后续研究提供参考。

基金项目： 2024年度南阳市科技发展计划项目“室内移动机器人自身定位与避障问题研究”（24KJGG051）；2024年度南阳市科技发展计划项目“基于三维空间实时综合感知的掘进准直技术研究”（24KJGG082）

作者简介： 赵贺杨，男，2003年生，研究方向为机器人SLAM及自主导航。

1 实验平台的搭建

机器人采用了一个由NUC和STM32F1驱动板构成的双系统架构，以实现两者之间的通信框架。配置RPLIDAR A2激光雷达[2]、LCD显示屏、麦克纳姆轮、MPU6050姿态传感器、大功率有刷直流伺服电机。Intel NUC运行Ubuntu 18.04操作系统，搭载ROS系统Melodic版本。底层驱动板搭载STM32F1支持多种控制方式，与上层主控之间采用ROSSERIAL协议进行通信，利用节点通过ROS提供的通信机制互动实现数据交换和任务协调。通过软硬件结合实现机器人自动导航的仿真。

1.1 硬件系统

机器人的硬件架构如图1所示。

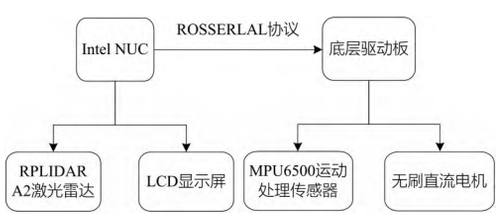

图1 机器人系统的硬件架构

ROS控制器采用Intel NUC，负责实时控制、传感器数据处理以及与外部设备通信，其控制了RPLIDAR A2激光雷达以及LCD显示屏模块。RPLIDAR A2激光雷达融合无线供电和光通信技术，实时获取环境高精度轮廓。激光雷达具体参数如表1所列。LCD显示屏负责显示系统的操作画面。

表1 激光雷达相关参数

参数	参数值	参数	参数值
采样频率	8K	供电电流	100 mA
扫描频率	5.5 Hz	供电电压	5 V
角度分辨率	≤1°	输出	UART串口
功耗	0.5 W	扫描范围	360°
工作温度	0~40 ℃	测量半径	0.15~12 m

底层驱动板的微控制器（MCU）搭载了STM32F103~RCT6高性能芯片。芯片负责控制集成3轴陀螺仪和3轴加速度计的MPU6050六轴运动处理传感器，并支持多种量程和数字滤波器，同时也负责无刷直流电机的闭环驱动控制。

1.2 软件架构

ROS机器人操作系统是一个功能卓越的机器人软件框架，具有节点单元的通信方式，保证了数据的一致性和运算的正确性。ROS也涵盖了Gazebo和RVIZ等工具。实验通过ROS实现软件架构的构建，首先构建URDF模型并与Gazebo环境集成，其次在RVIZ中显示，最后再调用建图功能包，构建地图。SLAM建图的软件架构如图2所示。

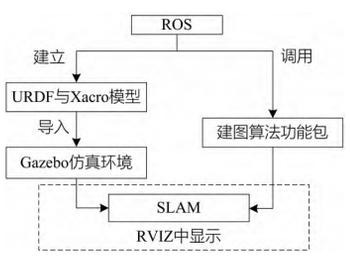

图2 软件架构

2 机器人建模与仿真

因现实环境中物理因素可变性大，场景搭建较为复杂难以实现，所以选择在ROS系统中利用Gazebo进行仿真，可模拟各种现实环境和突发情况。

2.1 仿真环境搭建

设计使用Gazebo对机器人导航环境进行搭建。设计墙高为5 m、墙厚为0.15 m、边长为20.5 m的正方形围墙，以模拟现实厂房，如图3所示。

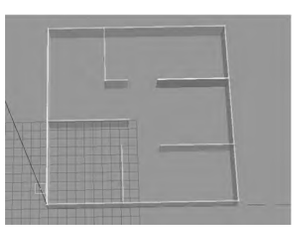

图3 仿真环境效果图

随后创建一个worlds的文件夹用来保存创建好的环境，以便于随时调用。

2.2 机器人建模

此次研究构建了URDF模型，建模主要步骤分为主体设计、轮子配置、雷达模型的配置3个步骤。

2.2.1 机器人主体设计参数

（1）<link name="base_link">；为link名称。

（2）<box size="0.30 0.25 0.15">；表示主体为长方体，后方参数为长宽高。

（3）<origin xyz="0 0 0" rpy="0 0 0">；*xyz*代表三轴的原点坐标，*rpy*通过欧拉角表示主体的旋转。

（4）<material name="yellow">；材料的颜色。

<color rgba="1 0.4 0 1">；颜色的透明度。

2.2.2 机器人轮子参数配置

左前轮配置如下：

（1）<link name="left_front_link">；左前轮连接。

（2）<origin xyz="0 0 0" rpy="1.5707 0 0"/>；origin *xyz*为原点坐标；*rpy*为绕轴心旋转的欧拉角，此处表示轮子绕着*X*轴旋转90°。

（3）<material name="white">；<color rgba="1 1 1 0.9"/>；颜色命名与颜色透明度。

（4）<joint name="left_front_joint"type="continu-ous ">；关节的命名和关节的持续运动。参考上方参数设计其他3个轮子同理。

2.2.3 添加激光雷达

激光雷达主体为圆柱体，引入了Xacro描述格式，且需要调整参数，参考表1。其部分主要代码如下：

（1）<gazebo reference="${base_link">：表明雷达被连接到‘base_link’的链接上。

（2）<xacro：property name="laser_length"value ="0.05" />表示圆柱体的高。

（3）<xacro：property name="laser_radius"value ="0.03" />表示圆柱体的半径。

（4）<pose>0 0 0 0 0 0</pose>：定义了雷达的原点和旋转角。

（5）<min_range>0.15</min_range>；<max_range>12.0</max_range>表示雷达探测范围为0.15~12.0 m。

（6）<update_rate>5.5</update_rate>：表示雷达扫描频率为5.5 Hz。

完成上述步骤后，机器人模型构建完毕，RVIZ中的机器人模型如图4所示。

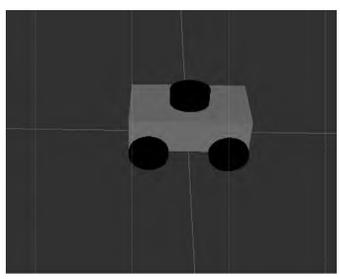

图4 RVIZ中的机器人模型

2.3 Gazebo仿真

Gazebo仿真首先要添加3个部分的Gazebo特征与属性，以完善机器人模型，随后配置launch文件启动节点显示机器人模型，最后再逐步实现激光雷达的仿真和机器人运动仿真。Gazebo仿真实现流程如图5所示。

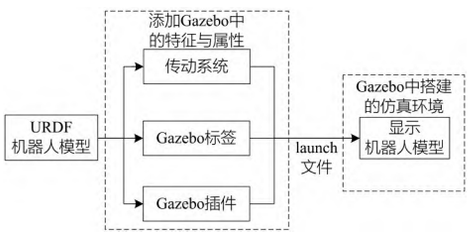

图5 Gazebo仿真实现流程

（1）添加动力传动系统关键代码如下：

<transmission name="wheel ${left_joint_trans">

定义了左轮传动装置。

< type > transmission interface / Differential Transmission < / type >定义了传动装置类型为差动类型。

（2）添加Gazebo标签关键代码如下：

<gazebo reference="base_link">

<material>Gazebo/Yellow</material>

</gazebo>

本段代码定义了Gazebo中机器人主体的颜色为黄色，其他属性同理。

（3）添加Gazebo插件关键代码如下：

<plugin name="unique_name" filename =" libgazebo_ros_diff_drive.so>定义了要加载到Gazebo中的差速插件。

（4）显示机器人模型部分，通过launch文件启动，其关键代码如下：

<node name= "urdf _spawner" pkg= "gazebo_ros" type= "spawn_model"respawn= "false"output= "screen" args = " -urdf -model mrobot param robot description "/>

代码实现在Gazebo中加载机器人模型。

<node name="joint state publisher"pkg="joint state publisher" type="jointstate publisher" />

代码运行joint state publisher节点，其他节点的运行同理。运行完以上步骤后，小车被添加到Gazebo中，如图6所示。

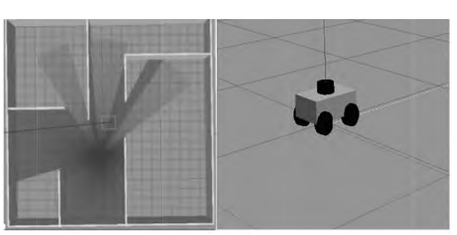

图6 Gazebo中的机器人模型

3 SLAM建图仿真

SLAM中主要有3种构建地图的算法：Gmapping、Hector-SLAM、Cartographer，接下来对其进行逐一的介绍，并对3种构建地图的方法和效果进行研究分析，研究出其分别适用的场景。

3.1 Gmapping地图构建

Gmapping是一种SLAM算法，用于接收深度信息、IMU信息、里程计信息，其外部框架如图7所示。

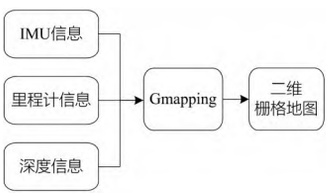

图7 Gmapping功能包外部框架

Gmapping算法依赖于里程计信息，通过控制数据和观测数据来联合求解位姿和实时建图，使用基于粒子滤波的定位建图的RBPF方法，定位部分采用与粒子滤波相似的过程，通过执行状态预测、测量数据获取、状态更新及重采样等一系列步骤，来完成对二维栅格地图的构建[3]。Gmapping内部运行框架如图8所示。

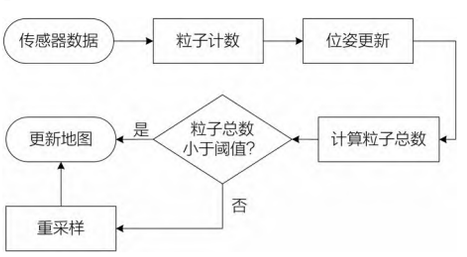

图8 Gmapping内部框架

要用Gmapping实现地图构建，首先需要创建一个gmapping.launch文件，用于启动节点并设置一些节点参数，主要设置了里程计数据的参考坐标系，地图更新时间间隔。其部分代码如图9所示。

```
<launch>
<arg name="scan_topic" default="scan" />
 <node pkg="gmapping" type="slam_gmapping"
name="slam_gmapping" output="screen" clear_params="tru
<param name="odom_frame" value="odom"/>
 <param name="map_update_interval" value="2.0"/>
```

图9 launch文件部分代码

首先启动Gmapping功能包和Gazebo仿真环境，其次将模型导入后在RVIZ中显示，最后启动键盘控制，让机器人围绕仿真环境运动1周进行建图。Gmapping建图过程如图10所示。

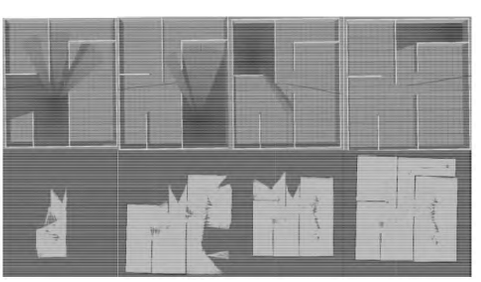

图10 Gmapping建图流程

最后再控制机器人绕场1圈后，输入指令：$ rosru-nmap_server map_saver-f catkin_ws/src/abot_slam/maps/map，地图会保存在当前终端的目录下命名为map，其构建的地图如图11所示。

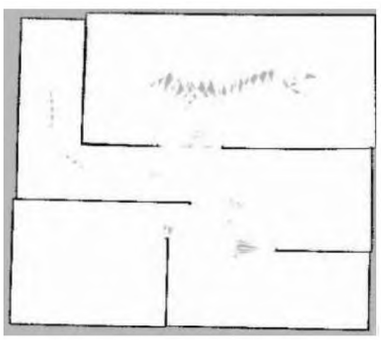

图11 Gmapping构建的地图

Gmapping建图算法建图的速度较快，相同情况下算法运算量小，算法结构简单。由于过度依赖里程计，在里程计性能不足或地图较大的情况下，建图精度较低，会出现如图11所示的错位现象，更换更高性能的里程计可改善此问题。

3.2 Hector-SLAM地图构建

Hector-SLAM接收激光雷达信息构建地图，其外部总体框架如图12所示。

图12 Hector-SLAM总体框架

Hector-SLAM是一种无需依赖里程计的实时2D SLAM解决方案。核心组件Hector-Mapping节点，负责从“/scan”话题接收激光扫描数据以支持SLAM过程。该节点不仅通过发布“map”话题来提供构建完成的地图信息，还额外发布“slam_out_pose”和“pose_update”两个话题，以实时更新和共享当前机器人位置的估计。Hector-Mapping的核心策略在于将获取的激光扫描点与已构建的地图进行精确匹配，该过程也被称为扫描匹配，以实现对机器人当前位置的精确估计。

启动功能包前，Hector-SLAM节点需要进行参数配置，主要配置地图更新距离阈值、地图更新角度阈值以及更新因子，从而达到高精度建图的目的。Hector节点参数如图13所示。

```
<!-- Map update parameters -->
<param name="update_factor_free" value="0.4"/>
<param name="update_factor_occupied" value="0.7" />
<param name="map_update_distance_thresh" value="0.2"
<param name="map_update_angle_thresh" value="0.06" /
```

图13 Hector节点参数

首先启动Hector-SLAM功能包，再配置好Gazebo仿真环境并导入机器人模型，将模型与地图在RVIZ中显示，最后启动键盘控制，操控机器人绕场地1周，建图过程如图14所示。

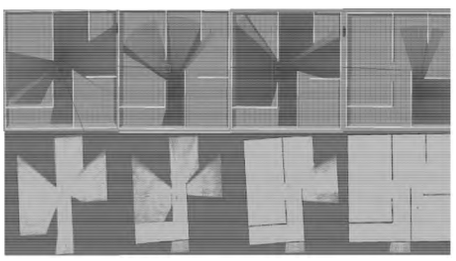

图14 Hector-SLAM功能包地图构建过程

建图完毕后，输入$ rosrun map_server map_saver catkin_ws/src/abot_slam/maps/hector保存地图。Hector-SLAM功能包构建的地图如图15所示。

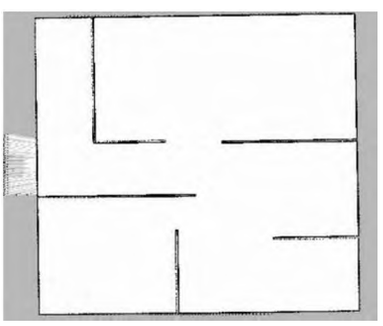

图15 Hector-SLAM功能包构建的地图

在研究过程中，一旦机器人的速度过快，会发生打滑现象，从而导致机器人构建的地图产生较大偏差。图16为机器人打滑现象，为避免该现象的发生，可以更换高性能的激光雷达或降低机器人的运行速度。

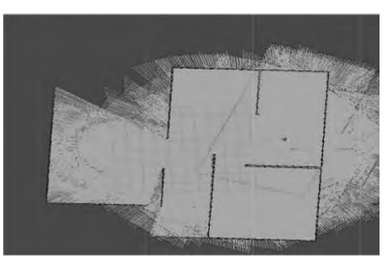

图16 机器人打滑现象

3.3 Cartographer地图构建

Cartographer是一种被广泛用于SLAM的算法。此算法综合了激光雷达、里程计等传感器数据来实现高效的地图构建和定位。Cartographer总体框架如图17所示。

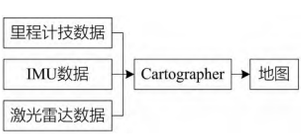

图17 Cartographer总体框架

Cartographer是基于图优化算法的SLAM。首先收集传感器数据进行数据处理，其次将数据传输至局部SLAM进行子图更新并反馈位姿优化值至整合数据，最后插入子图结果进行全局SLAM实现位姿调整[4-5]。Cartographer内部

框架如图18所示。

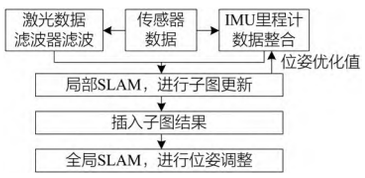

图18 Cartographer内部框架

在Cartographer功能包中，需使用launch启动文件来调用Cartographer功能包和启动RVIZ，其代码如图19所示。

```
include "map_builder.lua"
include "trajectory_builder.lua"
        options = {
                map_builder = MAP_BUILDER,trajectory_builder = TRAJECTORY_BUILDER,
                map_frame = "map",
                        tracking_frame = "base_footprint",
                published_frame = "odom",
                odom_frame = "odom",
                provide_odom_frame = false,
                publish_frame_projected_to_2d = false,
                use_odometry = true,
                use_nav_sat = false,
                use_landmarks = false,
```

图19 launch文件部分代码

首先启动Cartographer功能包，配置Gazebo仿真环境，其次导入机器人模型，启动键盘控制，最后控制机器人在地图中移动进行建图。Cartographer算法建图过程如图20所示。

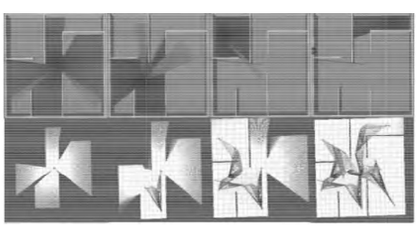

图20 Cartographer算法建图过程

建图完毕后，先停止接收指令，地图保存pbstream格式，此文件是Cartographer输出的二进制格式，它包含地图数据和机器人的轨迹信息，这种格式紧凑且高效。Cartographer构建的地图如图21所示。

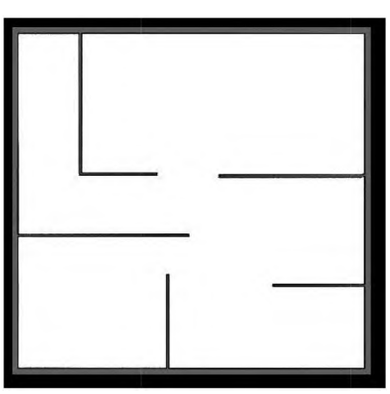

图21 Cartographer构建的地图

Cartographer算法建图精度极高，全局定位能力强，但对CPU等硬件设施要求较高。

4 数据分析与结论

本文详细研究了移动机器人自主导航设计中的构建地图算法。首先深入探讨了构建地图算法的基本原理，包括SLAM算法的实现过程和关键技术，通过传感器数据的获取和处理，机器人能实时构建周围环境的地图。其次对3种地图构建的算法进行了仿真实现，并通过对比3张地图，研究出3种算法所适用的场景。

统一设置机器人速度，运用3种算法进行8次建图，利用秒表记录地图构建时间表格，如表2所列。

表2 地图构建时间表格

建图次数/次	Gmapping算法建图时间/min	Hector-SLAM算法建图时间/min	Cartographer算法建图时间/min
1	1.5	3.2	4.2
2	2.2	3.4	5.1
3	1.3	2.7	4.7
4	1.9	2.7	4.2
5	1.5	2.9	4.5
6	2.1	3.1	4.9
7	1.7	3.3	4.2
8	1.8	2.6	4.4

根据表2中的参数绘制出柱状图，如图22所示。

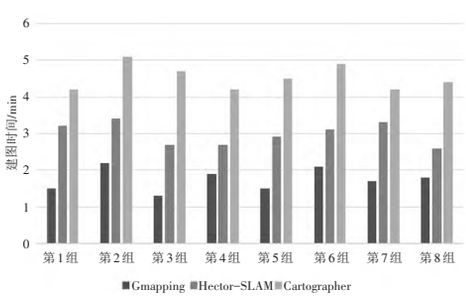

图22 3种算法建图时间柱状图

通过NAVIDIA检测工具检测出CPU占用率均值，如图23所示。CPU占用率不仅反映了建图算法的运算量，也间接反映了算法的复杂程度及其对外设硬件的性能要求。

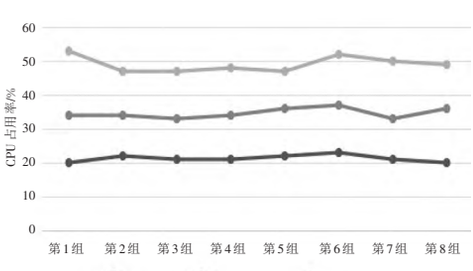

图23 3种算法CPU占用折线图

由图23可知，Gmapping算法建图的精度最差，但是其运算量小，且对硬件设备性能要求不高，相同速度下建图时间较短。Hector-SLAM算法精度较高，运算量适中，对硬件设备没有高性能要求，其实时性最好，但是速度过快时会出现机器人打滑现象，严重影响精度。Cartographer所需运算量较大，建图所需时间较长，且刚需高性能硬件设备，但其精度很高，稳定性最好。

（下转第42页）

aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization [J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2013, 43(6): 1451-1465.
[12]Roberge V, Tarbouchi M, Labonté G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning[J]. IEEE Transactions on Industrial Informatics, 2012, 9(1): 132-141.
[13]Duan H, Yu Y, Zhang X, et al. Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm[J]. Simulation Modelling Practice and Theory, 2010, 18(8): 1104-1115.
[14]Yang X S. Firefly algorithm, stochastic test functions and design optimisation [J]. International Journal of Bio-Inspired Computation, 2010, 2(2): 78-84.
[15]Eberhart R, Kennedy J. A new optimizer using particle swarm theory [C]//MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995: 39-43.
[16]Liu Y, Zhang X, Guan X, et al. Adaptive sensitivity decision based path planning algorithm for unmanned aerial vehicle with improved particle swarm optimization [J]. Aerospace Science and Technology, 2016, 58: 92-102.
[17]Suganthan P N. Particle swarm optimiser with neighbourhood operator[C]//Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 1999, 3: 1958-1962.
[18]Huang C, Fei J. UAV path planning based on particle swarm optimization with global best path competition[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2018, 32(6): 1859008.
[19]Higashi N, Iba H. Particle swarm optimization with Gaussian mutation[C]//Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), 2003: 72-79.
[20]高立群,吴沛锋,邹德旋.基于变异策略的粒子群算法[J].东北大学学报(自然科学版),2010,31(11):1530.
[21]Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1): 58-73.
[22]Qi Z, Shao Z, Ping Y S, et al. An improved heuristic algorithm for UAV path planning in 3D environment[C]//2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics, 2010, 2: 258-261.

(上接第37页)

通过对Gmapping、Hector-SLAM、Cartographer这3种算法实验结论的分析,可得出各SLAM算法各自的特性与所适用的场所,具体如表3所列。

表3 3种SLAM算法的总结

SLAM算法	精度	建图速度	运算量	原理	适用场景
Gmapping	较低	较快	较低	滤波器	地图结构简单,精度要求较低,没有高性能设备的场所
Hector-SLAM	适中	较快	较高	扫描匹配	实时性强,无高速要求,精度要求适中的场所
Cartographer	较高	较慢	极高	图优化	对精度要求高,并且有高性能设备的场所

5 结语

本文从SLAM建图算法入手,利用ROS框架以及其中的Gazebo与RVIZ工具,设计软件架构,对Gmapping、Hector-SLAM、Cartographer这3种SLAM算法进行仿真实验,通过传感器数据的获取和处理,机器人能实时构建周围环境的地图,详细研究了移动机器人自主导航核心领域之一的构建地图算法。研究深入探讨了构建地图算法的基本原理,包括SLAM算法的实现过程和关键技术,并对3张地图进行对比得出3种建图分别适用的场所,为后续实现机器人精确导航提供了坚实基础。

参考文献

[1]李少伟,钟勇,杨华山,等.SLAM算法建图对比研究[J].内燃机与配件,2024(3):36-38.
[3]吴晓锋.基于激光雷达的定位算法与模块实现[D].福州:福州大学,2021.
[3]王海瑶,安天洋.基于Gmapping和A*算法的运输机器人系统的设计[J].工业控制计算机,2024,37(1):19-21.
[4]马志艳,邵长松.移动机器人2D激光SLAM算法仿真与实现[J].湖北工业大学学报,2024,39(2):5-9.
[5]张宸.基于视觉的室内SLAM算法关键技术研究[D].沈阳:沈阳航空航天大学,2022.