**包头师范学院**

**本科毕业论文**

**题 目：基于2D激光测距模块的2Dslam算法**

**学 号： 1514860024**

**学生姓名： 任冠雄**

**学 院： 信息科学与技术学院**

**专 业： 电子信息科学与技术**

**班 级： 15电子**

**指导教师： 米红波**

**二 〇 年 月**

摘 要

本设计是根据2D激光测距模块、里程计和陀螺仪等机器人传感器的数据实现机器人的slam系统（即时定位与地图构建）的算法。系统会采集2D激光测距模块数据并解析，使用Bresenham算法转换成单帧地图，然后使用里程计推算定位算法集合底盘上传的里程计和陀螺仪数据进行机器人的定位，最后通过socket通信结合单帧地图算法和机器人定位推算算法实现地图实时更新。本文会详细介绍slam系统的硬件、软件设计方法，说明系统所使用传感器测距模块、里程计和陀螺仪的工作原理及性能特点，重点阐述系统所用算法的原理以及系统所用通信方式。slam系统能准确建立机器人环境地图并确定其位置，对机器人的路线规划有重要意义，系统有很好的实用价值。

关键词： slam；Bresenham算法；里程计推算定位算法；2D激光测距模块

Abstract

This design is an algorithm that is based on the data of 2D DEEP CAMERA, odometer, gyroscope and other robot sensors to realize the robot slam system (simultaneous localization and mapping) .System will collect the 2D DEEP CAMERA and data analysis, using Bresenham algorithm build a single frame map, then use the odometer reckoning positioning algorithm and data of the odometer and gyroscope to calculate the positioning of the robot, at last, through the socket communication combined with single frame calculation algorithm and a robot localization algorithm to build the real time map .In this paper, the hardware and software design methods of slam system will be introduced in detail, and the working principles and performance of the odometer and gyroscope will be explained, with emphasis on the principle of the algorithm used in the system and the communication mode used in the system.The slam system can accurately establish the robot environment map and determine its location, it is a great significance for the path planning of the robot, and the system has a good practical value.

Key words：slam; Bresenham algorithm; odometer reckoning positioning algorithm；2D DEEP CAMERA

目 录

[1 引言 1](#_Toc6675999)

[2 slam系统传感器选择 2](#_Toc6676000)

[2.1 2D激光测距模块 2](#_Toc6676001)

[2.1.1 模块性能 2](#_Toc6676002)

[2.1.2 模块系统组成 3](#_Toc6676003)

[2.1.3 数据采样及返回数据格式 4](#_Toc6676004)

[2.2 里程计和陀螺仪 4](#_Toc6676005)

[2.2.1 里程计传感器的选取 4](#_Toc6676006)

[2.2.2 陀螺仪的选取 5](#_Toc6676007)

[3 绘图程序设计 7](#_Toc6676008)

[3.1 2D激光测距模块数据解析 7](#_Toc6676009)

[3.2单帧地图创建 8](#_Toc6676010)

[3.2.1 Bresenham画直线算法 8](#_Toc6676011)

[3.2.2 构建单帧地图 10](#_Toc6676012)

[3.3机器人全图绘制及实时更新 12](#_Toc6676013)

[4 底盘程序设计 14](#_Toc6676014)

[4.1 底盘数据解析 14](#_Toc6676015)

[4.2 里程计推算定位算法 15](#_Toc6676016)

[4.3 机器人移动控制 17](#_Toc6676017)

[5 系统通信方式 18](#_Toc6676018)

[5.1 上位机与底盘控制板间通信 18](#_Toc6676019)

[5.2 绘图程序与底盘程序进程间通信 18](#_Toc6676020)

[5.2.1 socket通信 19](#_Toc6676021)

[5.2.2 绘图程序与底盘程序的数据传输 20](#_Toc6676022)

[结 论 21](#_Toc6676023)

[参考文献 22](#_Toc6676024)

[附 录1 23](#_Toc6676025)

[附 录2 24](#_Toc6676026)

[致 谢 25](#_Toc6676027)

# 1 引言

随着室内移动机器人应用的兴起，移动机器人的定位就成了非常重要的研发内容，常用的定位方式有正交码盘加陀螺仪、摄像头定位、激光雷达定位等，每种定位方式各有优劣。在这样的环境下，就出现了slam算法（simultaneous localizati and mapping），中文名称为即时定位与地图构建，使用该算法可以让机器人在自身位置不确定的情况下，创建陌生环境地图并确定自身位置。通俗的讲，slam就回答了两个问题，即“我在那？”，“我周围是什么样的？”。

slam算法包括机器人定位和建图算法以及通过子图匹配纠正定位算法两部分。前者主要涉及算法有栅格直线算法、里程计推算定位算法和二轮差动模型的建立，后者主要涉及各种粒子滤波和图像匹配算法，如卡尔曼滤波，Gmapping算法和基于贝叶斯滤波器的概率模型算法等。本文主要介绍机器人的定位与建图算法，不对子图匹配定位纠正进行论述。

利用激光雷达作slam传感器，精度很高，速度快，计算量也不大，再结合里程计和陀螺仪即可以做成实时slam，又可以弥补单个传感器slam系统累计误差太大的不足。slam系统最重要的就是激光雷达，传统激光雷达多为机械旋转式单线激光雷达，需要有电机和皮带带动激光头转动，安装部署对结构有要求（要求扫描平面无遮挡），且价格比较昂贵（目前市面上比较便宜的机械旋转式单线激光雷达也得几千元）。本次设计所用的测距模块为2D激光测距模块，相较于机械旋转式单线激光雷达，2D激光测距模块价格较便宜，体积更小，并且是固态结构，结构更稳定，使用寿命更长，非常具有实用价值。

# 2 slam系统传感器选择

slam系统所用传感器包括2D激光测距模块、里程计和陀螺仪。2D激光测距模块可向激光头前方发射红外线，并接受物体反射红外线，通过发送和接收间隔计算出前方物体的距离和形状。里程计负责记录机器人驱动轮所转动的圈数。陀螺仪则是用来记录机器人转动角度。slam算法通过上述三个传感器采集的数据建立运动模型，然后计算自身位置和周围环境。下面将分别论述各传感器的功能特性及工作原理。

2.1 2D激光测距模块

在slam系统中，最核心的传感器就是2D激光测距模块，测距模块的特性直接决定了slam算法的建图效果和定位纠正的效果。传统的机械旋转式单线雷达视角能达到360°，探测距离可达8米，数据刷新较快，缺点是价格比较昂贵，且需要电机和皮带带动激光头进行360°转动，旋转扫描的测量方式导致电机和皮带会经常损坏，导致其使用寿命不够长。针对机械旋转式单线雷达寿命短和价格居高不下的问题，本次设计选取了新型的固态激光雷达2D激光测距模块。

2D激光测距模块是北京雷动云合智能技术有限公司研发的一款激光测距模块，主要用于室内机器人的定位和建图，障碍物探测与规避。模块为全固态设计，无任何机械旋转部件，使用寿命较长，体积小巧，可直接嵌入机器外壳，价格相较旋转式激光雷达更便宜。

2.1.1 模块性能

2D激光测距模块视野广阔，数据采样频率较快，红外线较密集，适用与室内机器人定位和地图构建。2D激光测距模块各项性能参数如表2-1所示。

表 2-1 2D激光测距模块性能参数

|  |  |
| --- | --- |
| 性能参数 | 值 |
| 探测视角 | 111° |
| 探测距离 | 5米 |
| 角度分辨率 | 0.5° |
| 采样频率 | 3-5次/每秒 |

2D激光测距模块采用全固态设计，结构稳定，对工作环境要求不高，模块工作环境要求如表2-2所示。

表 2-2 2D测距模块工作环境要求

|  |  |
| --- | --- |
| 环境要求 | 值 |
| 工作温度 | 0℃ ~ 40℃ |
| 工作电压 | 5V (±0.5V) |
| 工作电流流 | 最大电流280mA，额定电流160mA |
| 模块重量 | 40.5g (±1g) |

2.1.2 模块系统组成

2D模块主要分为depth测距核心以及搭载核心的硬件部分组成。测距核心负责朝模块正前方扫描采样。硬件部分包括内部的主控电路和外部的UART通信接口、激光发射头、激光接收摄像头和目标摄像头。模块外形构造如图2-1所示。

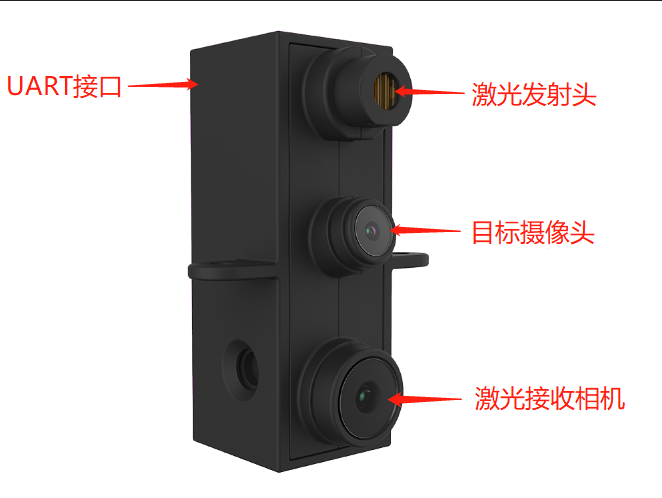


图 2-1 2D激光测距模块外形构造

模块各部分功能说明如表2-3所示。

表 2-3 2D激光测距模块各部分功能说明

|  |  |
| --- | --- |
| 硬件结构 | 功能说明 |
| 主控电路 | 包括主控单片机和驱动电路，是测距核心，控制激光收发传感器和摄像头工作，进行数据处理由UART串口发出 |
| UART接头 | 连接内部单片机串口，模块通过UART接口和其它设备进行数据传输 |
| 激光发射头 | 向前方111°范围发射红外线 |
| 激光接收摄像头 | 接收反射回来的红外线 |
| 目标摄像头 | 可以进行图像识别，本次设计未使用 |

2.1.3 数据采样及返回数据格式

2D激光测距模块通过UART串口与其他设备通信，接收到采样指令后，模块会测量正前方111°范围内前方障碍物距离并由串口返回。采样指令和返回数据都以十六进制数进行传输，串口传输波特率为115200 bps。

采样指令为：0xa5 0x07 0x01 0x36 0x00 0xe3。指令第1位为是数据帧头，固定为0x05，第2位为指令长度，第3位为设备号，后面4位为数据位。

每帧返回数据共446位，数据的第一位为数据帧头，固定为0xa5，第二位是设备号，后面的444位表示激光头前方-55°到+55°的距离值，每隔0.5°返回一个距离值，由相邻的两位十六进制数表示，前一位十六进制为高八位，后一位十六进制为底八位。为了能得到更精确的地图，本次设计只使用激光头前方90°3米以内的数据。2D激光测距模块数据格式如图2-2所示。

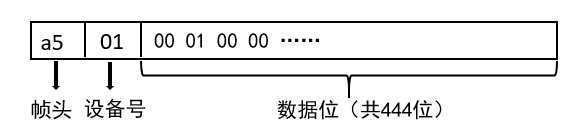


图 2-2 2D激光测距模块数据

2.2 里程计和陀螺仪

虽然通过2D激光测距模块距离变化值和时间变化可以实现slam的定位推算，但在某些特殊环境下（如前方为空旷地，激光扫描的有效点较少时），单靠激光模块建图和定位就会很困难。 因此，要在slam系统中加入里程计和陀螺仪，用来更精确的计算机器人的移动距离和转动角度。

2.2.1 里程计传感器的选取

里程计选取光电编码里程计，光电编码使用光电传感器的原理进行主动轮里程计算。光电编码里程计结构如图2-3所示，由发光二极管、光敏晶体管以及装置在速度表驱动轴上的遮光板构成。

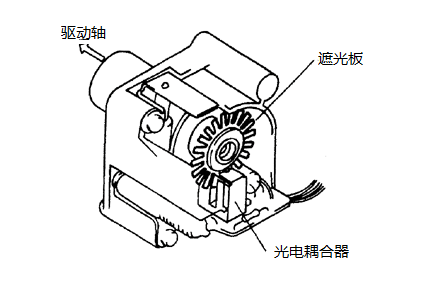


图 2-3 光电编码里程计结构

光电编码里程计工作原理如图2-4所示。当遮光板不能遮断发光二极管关照时，发光二极管的光会照到光敏晶体管上，光敏晶体管的集电极就会有电流通过，使光敏晶体管导通，这时三极管VT也导通，Si端就会产生一个高电平脉冲，主动轮旋转一周一共产生1620个高电平脉冲。根据底盘反馈的主动轮脉冲数就可以计算轮子的转动里程值。

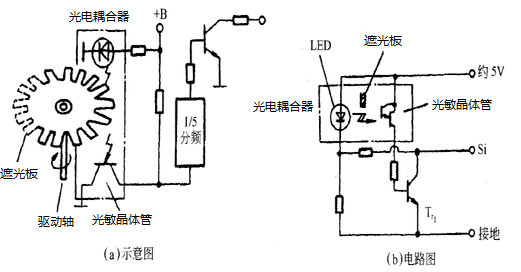


图 2-4 光电编码里程计工作原理

2.2.2 陀螺仪的选取

陀螺仪选取陀螺仪，如图2-5所示。FXAS21002C是一个小型、低功耗、偏航、俯仰和滚转角速率陀螺仪，具有16位ADC分辨率。同时具有I2C和SPI接口。工作电压1.95 V到3.6 V。

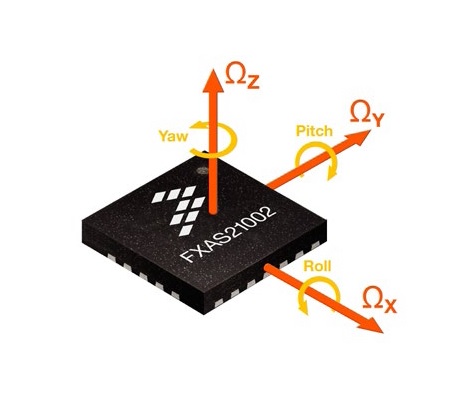


图 2-5 FXAS21002C三轴陀螺仪芯片

FXAS201002的数据传输由ST信号触发，主机（或MCU）向FXAS21002C发送一个ST，然后是从地址，R/W位设置为“0”进行写入，FXAS21002C发送一个确认。然后，单片机传输要读取的寄存器地址和FXAS21002C发送确认。MCU传输一个SR，然后是包含从地址的字节和R/W位设置为“1”，以便从以前选择的寄存器中读取。然后FXAS21002C确认并传输来自请求寄存器的数据。主进程传输一个NACK，然后是一个SP，表示传输结束。

陀螺仪与底盘控制板相连，底盘将解析后的数据上传到上位机供建图使用，陀螺仪数据形式如图2-6所示。机器人转动后朝向线在初始位置朝向线右边时，转动角θ值为正，范围为0°~180°，转动后朝向线在初始位置朝向线左边时，转动角θ值为负，范围为-180°~0°。

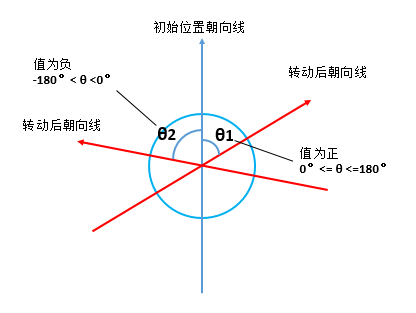


图 2-6 底盘上传陀螺仪数据模式

# 3 绘图程序设计

3.1 2D激光测距模块数据解析

2D激光模块可以由串口返回激光头前方111°范围内的测距值，为保证地图的精准，实际使用中只使用激光头正前方90°范围内3米以内的测距值。

绘图程序从串口将读取的原始数据存放到数组recvBuff中，从数组recvBuff[2]开始为测距模块测距值，且相邻的两位表示一个测距值，绘图程序会将相邻的两位进行移位相加转换成相应的十进制测距值，并存放到数组intbuf中，测距模块数据的读取及转换代码实现如图3-1所示。



图 3-1 2D激光测距模块数据读取及转换代码实现

由模块数据格式可知，intbuf数组中intbuf[0]存放激光头左侧55°的测距值，intbuf[1]存放左侧54.5°的测距值，intbuf[2]存放左侧54°的测距值，依次可得激光头左侧45°的测距值存放在intbuf[20]中，激光头右侧45°的测距值存放在intbuf[201]中，intbuf数组测距值及对应的测距角度如图3-2所示。

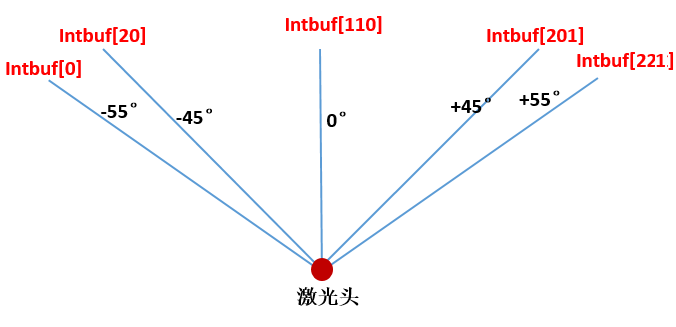


图 3-2 inbuf数组测距值及对应的测距角度

3.2单帧地图创建

通过Bresenham画直线算法和相应的几何关系，就可以将2D激光模块的数据转换成栅格地图，转换后的地图可以保存到二维数组中，供机器人路径规划使用，并可以使用opencv库转换成图片，显示到上位机界面。

3.2.1 Bresenham画直线算法

真实的直线是连续的，但计算机显示的精度有限，不能真正显示连续的直线。计算机中会用一系列离散化后的点（像素）来近似表现一条直线。Bresenham画直线算法是一种计算机图形学算法，Bresenham算法核心原理就是根据前一点坐标及直线斜率确定下一点坐标，依次确定所有像素点。

假设直线方程为y = kx + b，斜率k在0 ~ 1之间，并且。已知第i个像素点坐标为，直线的斜率小于等于1，所以下一个点的坐标只可能是或者，如图3-3所示。和表示直线实际点到上下两像素点的距离，通过比较和的大小来确定离实际点最近的像素点，然后用这一点表示直线实际点。

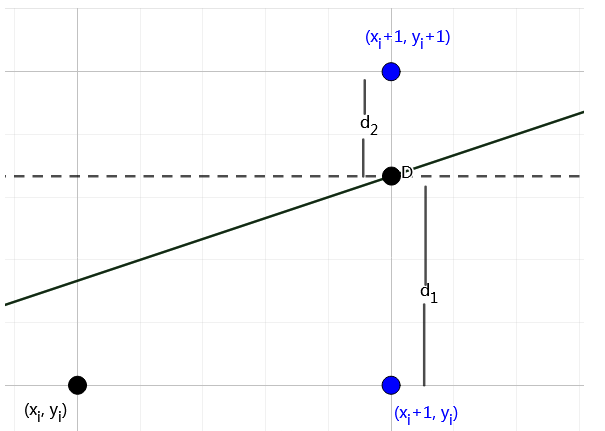


图 3-3 直线及附近像素点（网格相交点为像素点）

那么如何由第i个像素点确定第i+1个像素点呢？假设，则可得式（3-1）和式（3-2）

d1 = y – yi = k(xi + 1) + b - yi 式（3-1）

d2 = yi + 1 - y = yi + 1 - k(xi + 1) – b 式（3-2）

通过对d1和d2作差可得（式3-3），其中k = ，Δx = x2 - x1，Δy = y2 -y1。为避免除法运算，等式两边同乘Δx，得式（3-4）。

d1 ​− d2 ​= 2k( xi ​+ 1) − 2yi ​+ 2b – 1 式（3-3）

Δx (d1 ​− d2) ​= 2Δy (xi ​+ 1) − 2Δxyi ​+Δx (2b − 1) 式（3-4）

假设Pi = Δx (d1 ​− d2)，不变的部分用c代替，则有式（3-5），这样就可以通过判断Pi的正负来确定要选取的像素点，当Pi ⩾0，即d1⩾d2，选取右边的点，此时 yi+1 = yi+1; 当Pi < 0，即d1 < d2，选取右上的点，此时 yi+1 = yi。​

Pi ​= 2Δyxi − 2Δxyi ​+ c 式（3-5）

我们通过Pi的正负选出下个点，然后判断Pi和Pi+1的关系来求出Pi+1的值。根据Pi的表达式递推可得式（3-6），式（3-6）与式（3-5）相减即得Pi和Pi+1的关系，如式（3-7）所示。将式（3-7）展开后得到式（3-8），其中只有yi+1未知。

Pi+1 = 2Δyxi+1 − 2Δxyi+1 + c 式（3-6）

Pi+1 – Pi = 2Δy − 2Δx( yi+1 – yi) 式（3-7）

2Δyxi+1 − 2Δxyi+1 + c - 2Δyxi + 2Δxyi ​– c = 2Δy − 2Δx( yi+1 – yi) 式（3-8）

于是可得结论：当Pi ⩾0时，应取yi+1 = yi +1，得Pi+1​=Pi​+2(Δy − Δx)，选取右上的像素点表示直线实际点；当Pi < 0时，应取yi+1 = yi ，得Pi+1​=Pi​+2Δy，选取右边的像素点表示直线实际点。

求出Pi点和Pi+1的关系后，从起点开始递推就可推出表示直线的所有像素点。需要画斜率大于1的直线时可以将y轴看成x轴以同样的发式进行推导即可。代码实现见附录1。

3.2.2 构建单帧地图

地图存放于1024\*1024的二维数组xmap中，且认为机器人的位置在地图中心，初始朝向角为0°，则可建立如图3-4所示模型，其中（512, 512）表示数组xmap[512][512]，存放机器人位置，点P为红外线照射到的点，地图绘制时只选取激光头正前方90°的测距值，可得角α=135°。本章3.1节中介绍过intbuf数组中依次存放着激光头前方111°的测距值，每隔0.5°存放一个测距值，且intbuf[20]存放的激光头左侧45°的测距值，则红外线与x轴夹角β=α-(i-20)/2.0，式中i为当前红外线测距值保存在数组的位置下标。

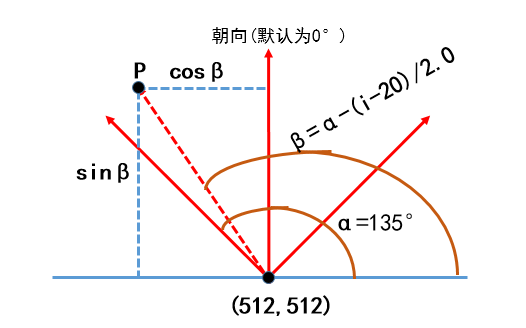


图 3-4 单帧地图几何模型建立

接下来我们来确定红外照射点在数组中的位置，首先算出红外线照射到的点相对机器人位置的距离，这个距离包括x轴方向的距离和y轴方向的距离，然后根据机器人当前位置确定红外线照射点在地图数组中的位置。

以机器人所在位置为原点建立二维坐标系，由三角关系可得红外线照射点P到机器人x轴方向的距离为cosβ,y轴方向的距离为sinβ。确定P点在数组的位置需要将二维坐标系和数组联系转换，在数组中（512,512）存放机器人位置，P点在数组中横纵下标如下：

P点的行下标 = 机器人行下标（512）+ cosβ \* 分辨率 \* 测距值

P点的列下标 = 机器人列下标（512）- sinβ \* 分辨率 \* 测距值

其中分辨率指的是坐标系中一个单位表示数组中的元素数，本次设计分辨率为1，即坐标系中每个到位表示数组一个元素。测距值取intbuf中所存放的值，如果测距值大于3米，测距值按3米计算。

确定红外照射点在数组中的为之后，使用Bresenham算法在机器人位置和P点画直线，所有的将障碍物点在数组中标记，这样就可以建立单帧地图，使用opencv按地图数组显示如图3-5所示，单帧地图代码实现如图3-6所示。

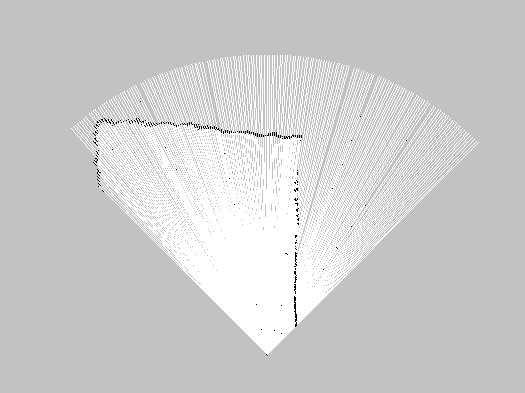


图 3-5 单帧地图效果呈现



图 3-6 单帧地图绘制代码实现

3.3机器人全图绘制及实时更新

机器人全图的绘制，需要在单帧地图的基础上不断更新机器人位置和朝向,机器人位置会每秒进行一次更新，全图绘制代码实现如图3-7所示，其中sxx为机器人当前位置行下标，syy为机器人当前位置列下表，theta为机器人当前位置朝向（机器人从初始位置到现在旋转的角度）。



图 3-7 全图绘制代码实现

让机器人在如图3-8环境下运动，全图绘制效果如图3-9所示，红色为机器人运动轨迹，白色部分为2D激光测距模块已探索区域，黑色部分为障碍物，灰色部分为2D激光测距模块为探索部分。



图 3-8 机器人运动环境

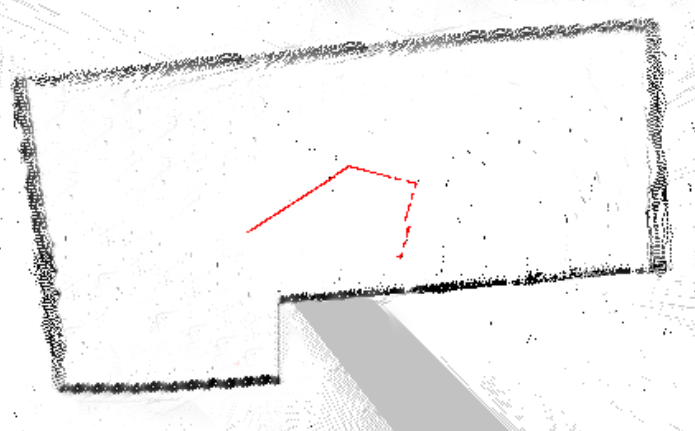


图 3-9 全图效果呈现

# 4 底盘程序设计

底盘程序主要包括解析底盘控制板数据、通过里程计数据进行定位推算，控制机器人进行移动三大部分组成。其中里程计推算定位算法是重点，这里将详细进行介绍。

4.1 底盘数据解析

上位机与底盘控制板的通信采用蓝牙通信，配对好的蓝牙会进入透传模式，相当于串口通信，数据通信协议会在第5章进行详细介绍。底盘上传数据内容包括时间戳、左右轮里程计脉冲、底盘请求、故障码、陀螺仪数据和红外信号等数据，本次设计只使用其中的时间戳、左右轮里程计脉冲，和陀螺仪数据。

每帧数据由帧头、帧类型，数据长度、数据内容、校验位及帧尾组成，其中数据长度占两位，表示数据内容和数据长度一共所占位数，数据长度和数据内容传输时高字节在前，校验位为两位数据长度位和所有数据部分异或所得。底盘部分数据内容如表4-1所示。

表 4-1 底盘部分数据格式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 帧类型 | length | Data | 含义 | 字节数 | 开始位 |
| 0x01 | 0x4d | 时间戳 | 从开机到现在运行总时间 | 4byte | 4 |
| 左轮脉冲 | 当前时刻的左轮脉冲总量，前进递增，后退递减 | 4byte | 11 |
| 右轮脉冲 | 当前时刻的右轮脉冲总量，前进递增，后退递减 | 4byte | 15 |
| 陀螺仪航向角 | 浮点数＊100取整上报 | 1byte | 43 |

上位机收到底盘数据后会将数据保存到数组中，然后通过移位相加的方式将十六进制数转换成浮点型数。针对陀螺仪的数据格式的特点，数据转换时需要进行特殊处理，陀螺仪的数据格式在2.2.1小节中详细介绍，如果陀螺仪数据为负，则需要转换成正数，底盘数据解析代码如图4-1所示，定位推算算法的实现所用数据为当前数据与上次数据的差值，即图中的delta\_time、delta\_lpulse和delta\_rpulse。



图 4-1 底盘数据解析代码

4.2 里程计推算定位算法

在推算机器人位置时，我们会选取相邻两个时刻机器人位置建立如图4-2所示运动模型，由于底盘更新数据间隔较短为20毫秒，运动距离短，在运动模型中，将相邻两时刻运动轨迹看成直线，θ角为机器人第i时刻朝向角，α角为从i时刻到i+1时刻机器人转动的角度。xrobot和yrobot是已第i时刻机器人位置为原点建立的坐标系。

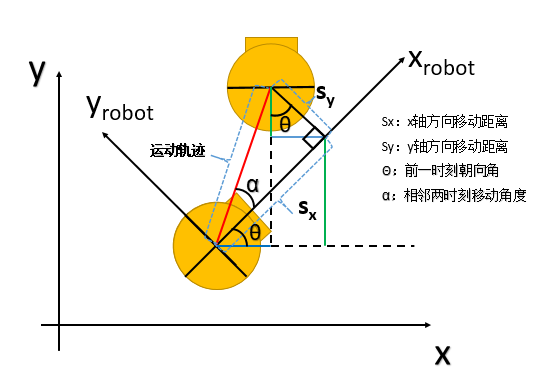


图 4-2 相邻两时刻运动模型建立

以第i时刻机器人位置为原点建立坐标系，且第i时刻机器人朝向为x轴方向，在知道x轴方向的移动距离Sx和y轴方向的移动距离Sy的情况下，由图4-2几何关系可得机器人从i时刻到i+1时刻在实际坐标系中的变化量如式4-1和式4-2所示。

Δx = Sx cosθ– Sy  sinθ 式（4-1）

Δy = Sx sinθ+ Sy  cosθ 式（4-2）

计算出相邻时刻坐标的变化量后，将变化量与前一时刻坐标相加即可得到当前时刻的坐标，如式4-3和式4-4所示。

xi+1 = xi + Sx cosθ– Sy  sinθ 式（4-3）

yi+1 = yi + Sx sinθ+ Sy  cosθ 式（4-4）

上面的定位推算是在Sx和Sy已知的情况下实现，现在我们就来根据里程计数据计算Sx和Sy的值。首先选取相邻两时刻机器人位置建立如图4-3所示模型。其中l为两驱动轮轴长。

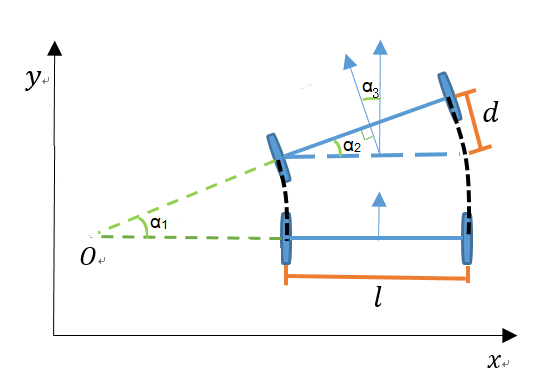


图 4-3 α角推算模型

图4-3中很明显，α3为相邻两时刻机器人朝向角变化量。d为左右轮移动距离的差值，由几何关系可得α2 = α3，应为两时刻时间间隔很短且移动距离很近，由可得α ≈ sinα，所以α3 = α2 ≈ sinα2 = dl。

已知两驱动轮周长为291.8mm，驱动轮旋转一周会产生1620个脉冲，两驱动轮中间的轴长237.5mm。根据相邻时刻移动角度α的值就可以计算出相邻时刻机器人在x轴方向和y轴方向的移动距离，进而实现机器人的定位推算，机器人定位推算代码实现如图4-4所示。

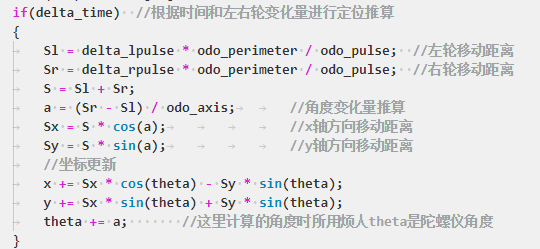


图 4-4 机器人定位推算代码实现

其中delta\_lpulse和delta\_lpulse分别为相邻时刻左右轮产生脉冲数，程序中相关常量宏定义如图4-5所示。



图 4-5 相关常量宏定义

4.3 机器人移动控制

机器人的移动由上位机发起，由底盘控制板控制，当底盘收到上位机下发的速度指令后，会控制两个轮已相应的速度进行转动，这样就实现了机器人的移动。下面将详细介绍上位机移动指令是如何发起的。

速度控制指令格式与底盘上传数据格式相同，由帧头、帧类型，数据长度、数据内容、校验位及帧尾组成。速度指令格式如图4-6所示。

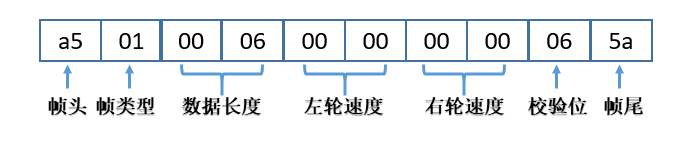


图 4-6 速度控制指令格式

左右轮速度由客户端定，上位机收到客户端速度值后，将十进制格式速度值转换成十六进制后存放到数组的4到7位，通过异或运算计算出校验位，将打包后的数据通过串口发送到底盘。代码实现见附录2。

# 5 系统通信方式

5.1 上位机与底盘控制板间通信

上位机与底盘控制板使用串口通信，电脑通过USB转串口接插件连接蓝牙模块，控制板串口连接一个蓝牙模块，两个蓝牙模块配对成功后会进入透传模式，上位机和底盘就可以进行无线传输数据。蓝牙模块的配置及使用这里将不做赘述，串口参数设置如表5-1所示。

表 5-1 串口参数设置

|  |  |
| --- | --- |
| 串口参数 | 值 |
| 波特率 | 115200 |
| 数据位 | 8 |
| 停止位 | 1 |
| 校验位 | N |
| 数据格式 | 十六进制 |

为保证及时对机器人进行控制，底盘控制板会每隔20毫秒上传一帧数据，上位机下发指令的频率为50毫秒每帧，数据帧格式如表5-2所示。上位机或下位机收到数据帧后会进行校验，如果数据校验不对，处理器就会丢弃这帧数据，应为指令传输频率足够快，所以这样即保证了数据传输的及时性又保证了数据传输的准确性。

表 5-2 通信数据帧格式

|  |  |  |  |
| --- | --- | --- | --- |
| header | 帧头 | 1byte | 0xa5 |
| command | 帧类型 | 1byte | com\_id |
| length | 帧长度 | 2byte | length+data总长度 |
| data | 数据内容 | variable |  |
| check | 校验位 | 1byte | length+data部分异或校验 |
| end | 帧尾 | 1byte | 0x5a |

5.2 绘图程序与底盘程序进程间通信

绘图程序和底盘程序同时运行在linux系统中，两者之间的通信属于进程间通信。进程间通信方式有管道通信、网络套接字通信、本地套接字通信及共享内存等多种通信方式，其中套接字通信是最简单且使用最常见的进程间通信方式，因为本次设计需要和客户端进行通信，所以选用网络套接字的方式进行通信。

5.2.1 socket通信

socket即网络套接字，是两个程序间实现数据传输的一个双向连接，这个双向的连接包括IP地址和端口号，通过对TCP/IP的封装实现网络通信。Socket通信流程如图5-1所示。

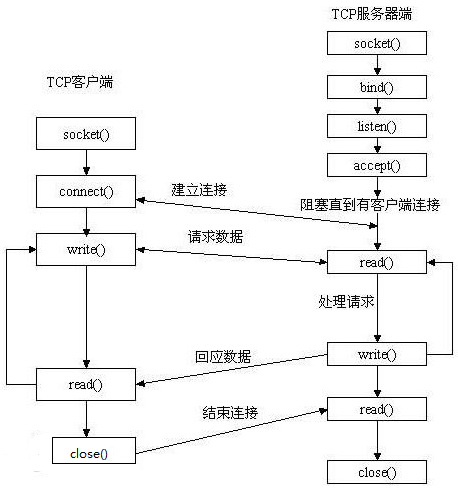


图 5-1 socket通信流程

通信过程分为这几个部分。第一部分，服务器和客户端创建socket句柄，由函数socket实现，函数返回为socket句柄，这个句柄有系统分配。

第二部分，服务器和客户端会创建一个sockaddr类型的结构体，并使用各自的IP、约定好的端口号和各自创建的socket句柄对结构体进行赋值，然后服务器端调用bind函数将IP地址、端口号和套接字句柄绑定在一起。

第三部分，服务器端调用listen函数对约定好的端口进行监听，并调用accept函数等待客户端的连接请求，这时服务端程序处于阻塞状态，如果有客户端发起连接请求，accept函数就会返回客户端套接字文件描述符，表示连接成功，上方就可以进行同信。

第四部分，客户端调用connect函数发起连接请求，如果函数返回为正则说明双方已建立连接，可以进行同信。

第五部分，服务端和客户端间的数据传输，使用函数send函数或write函数进行数据发送，使用recv函数或read函数进行数据接受。

5.2.2 绘图程序与底盘程序的数据传输

机器人各部分的功能都由不同的程序负责完成，slam算法的实现需要各部分程序使用socket通信的方式实现数据传输。底盘程序将位置信息发送给绘图程序，绘图程序才能绘制完整的及时的地图。机器人的地图和轨迹想要在客户端显示也需要绘图程序通过socket通信的方式将地图发送给客户端。如果客户端要实现对机器人的控制，则需要通过socket通信的方式向机器人发送速度指令。通过socket通信的进行数据交互的内容如表5-3所示，

表 5-3 使用socket通信的数据交互

|  |  |  |  |
| --- | --- | --- | --- |
| 发送端 | 接收端 | 通信端口 | 传输内容 |
| 底盘程序 | 绘图程序 | 15000 | 位置信息，用于绘图程序建图 |
| 底盘程序 | 客户端 | 13000 | 位置信息，用于客户端地图中机器人显示 |
| 绘图程序 | 客户端 | 14000 | 地图，实现客户端地图显示功能 |
| 客户端 | 底盘程序 | 13001 | 速度指令，经底盘程序转换后下发控制板 |

# 结 论

本次设计是只针对2D激光测距模块的slam算法。使用2D激光测距模块结合Bresenham算法进行地图绘制，使用里程计推算定位算法实现机器人的定位推算，使用socket通信的方式实现进程间的通信，完成全图的实时更新。

由于时间原因没有对所建地图进行去噪和美化，地图还不够美观。slam算法中没有作子图匹配定位纠正方面的研究。

slam算法的设计还需进一步完善，需要在目前基础上进行地图美化处理，加入子图匹配的算法，当里程计误差太大定位错误的时候可以通过图匹配算法对定位进行纠正。

使用2D激光测距模块、里程计和陀螺仪的多个传感器实现机器人的定位和建图，弥补了单传感器定位和建图误差较大的不足之处。虽然使用2D激光测距模块进行建图和定位没有使用机械旋转式激光雷达建图效果好，不能做到后者地图更新精确性和及时性，但2D激光测距模块相较于机械旋转式激光雷达优点突出，具有使用寿命长和价格便宜的特点，本次设计基本满足机器人应用中的要求，所以基于2D激光测距模块的slam算法的实现非常具有现实意义。

# 参考文献

[1] 洪洋，孙秀霞，王栋，等．基于矩形几何特性的小型无人机快速位姿估计方法［J］．中国激光，2016，43(5):226-238

[2] 杨海程，邓达强，等. 基于激光雷达和SLAM定位的麦克纳姆轮小车研究[J]. 机械工程师, 2018(11)

[3] 张建伟,张立伟,胡颖,等.开源机器人操作系统-ROS[M].北京:科学出版社,2012.

[4] 彭晟远．基于激光测距仪的室内机器人SLAM研究[D]．武汉：武汉科技大学,2012:1.

[5] 沈一鸣, 赵希宇. 基于激光ＳＬＡＭ 的移动机器人的改进实现[J]. 机械工程与自动化，2018(6)

[6] 王光庭,曹 凯,刘 豪. 基于激光雷达与视觉信息融合的SLAM方法[j]. 山东理工大学学报（自然科学报），2019(1)

[7] 赵希宇, 沈一鸣. 基于ICP的移动机器人同时定位与地图构建的研究[J]. 机械工程与自动化，2018(5)

[8] 寿佳鑫等. 基于ROS和激光雷达的室内移动机器人定位和导航系统设计与实现[J]. 智能工程, 2018(11)

[9] Bloesch M, Czarnowski J, Clark R, et al. CodeSLAM-Learning a Compact, Optimisable Representation for Dense Visual SLAM[J]. ArXiv preprint arXiv:1804.00874, 2018.

[10]Henriques J F, Vedaldi A. Mapnet: An allocentric spatial memory for mapping environments[C]//proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018

[11]Lan Z, Hsu D, Lee G H. Solving the Perspective-2-Point Problem for Flying-Camera Photo Composition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018

[12]Larsson V, Kukelova Z, Zheng Y. Camera Pose Estimation With Unknown Principal Point[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.2018

# 附 录1

void Bresenhamline(int x1,int y1,int x2,int y2,int color)

{

int dx, dy, yy, ix, iy, cx, cy, n2dy, n2dydx, d, temp;

dx = abs(x2 - x1);

dy = abs(y2 - y1),

yy = 0;

if (dx < dy) {

yy = 1;

int;

temp=x1; x1=y1; y1=temp;

temp=x2; x2=y2; y2=temp;

temp=dx; dx=dy; dy=temp;

}

ix=(x2 - x1) > 0 ? 1 : -1;

iy=(y2 - y1) > 0 ? 1 : -1;

cx=x1; cy=y1;

n2dy = dy \* 2;

n2dydx = (dy - dx) \*

if (yy) { // 如果直线与 x 轴的夹角大于 45 度

while (cx != x2) {

if (d < 0)

d+=n2dy;

else

{

cy += iy;

d += n2dydx;

}

if(xmap[cx][cy]!=4&&xmap[cx][cy]!=9)

xmap[cy][cx]=color; cx += ix;

}

} else { // 如果直线与 x 轴的夹角小于 45 度

while (cx != x2) {

if (d < 0)

d += n2dy;

else {

cy += iy; d += n2dydx;

}

if(xmap[cx][cy]!=4&&xmap[cx][cy]!=9)

xmap[cx][cy]=color; cx += ix;

}

}

}

# 附 录2

void move(int speedL, int speedR, int fd)

{

char one,two,thr,four;

int tempL, tempR;

char destBuffer[10]={0};

if(speedL<0) {

tempL = 0x8000 \* 2 + speedL;

one = tempL / 256;

two = tempL % 256;

} else if (speedL>=0) {

one = speedL / 256;

two = speedL % 256;

} if (speedR<0) {

tempR = 0x8000 \* 2 + speedR;

thr = tempR / 256;

four = tempR % 256;

} else if(speedR>=0) {

thr = speedR / 256;

four = speedR % 256;

}

destBuffer[0] = 0xa5;

destBuffer[1] = 0x01;

destBuffer[2] = 0x00;

destBuffer[3] = 0x06;

destBuffer[4] = one;

destBuffer[5] = two;

destBuffer[6] = thr;

destBuffer[7] = four;

destBuffer[9] = 0x5a;

destBuffer[8] = XoR(destBuffer, 3, 8, 10);

int i;

int count=5;

while(count--)

UART0\_Send(fd, destBuffer, 10); //串口发送

}

# 致 谢

本次毕业设计及毕业论文由米红波老师指导完成。感谢老师对我设计过程中的错误进行指正，并给出设计建议。本次设计内容是实习单位主要研究内容，很荣幸能够在北京雷动云合智能技术有限公司实习，感谢公司内实习导师的指导和帮助。通过本次设计我学到非常多以前没有接触过的内容，对以后的工作和学习有很大的帮助。