移动机器人自主回充技术理论与实践

1、前言

随着移动机器人的发展，工业界的移动机器人的产品层出不穷，在大家关注感知定位、路径规划、自主避障的同时，也有不少玩家注意到了机器人的充电问题。

移动机器人的充电？当然是要实现自主回充，不然的话还好意思说自己是机器人产品。

没错，我今天就要给大家提供一种移动机器人自主回充的实现方法：简单、可复现、提供源码。

2、常见的移动机器人自主回充技术

下面，我们简单介绍一下最常见的两种移动机器人的自主回充方案：

- 基于红外的自主回充

基于红外的自主回充，常见于扫地机器人，市面上常见的小米、石头、追觅等扫地机器人厂商均有使用该方案。

该方案移动一般仅有运动控制人员就可实现。

- 基于激光雷达的自主回充

基于激光雷达的自主回充，常见于大型移动机器人平台上使用，本篇主要介绍这种方案。

主要介绍的功能为**基于突变点检测**的方法对**特征充电座**进行识别。

3、基于2D激光雷达的回充方案

3.1、特征充电座的识别方案

3.1.1、说明

本篇充电座识别方案是参考如下论文的如下章节的，代码是复现论文的方法，论文会和代码一起打包给大家，这里感谢作者。

电子科技大学硕士学位论文

第五章 基于突变点检测的自动充电系统

当室内移动机器人在电量不足的情况下，需要有一定的低电量保护措施来让其自动转换到低电量模式，防止在后续服务过程中突然断电停机，影响使用感。在低电量保护模式下，机器人会开启自动充电远程导航模块抵达充电区域，接着采用近程对接技术完成精准对接后，开始充电。当机器人电量达到一定阈值后，重新开启远程导航模块，回到开始的地点，继续执行任务。目前看来，市面上大多数的移动机器人的供电系统都由机器人随身携带的蓄电池来提供支持，一般来说这种电池在不间断工作几个小时后便需要再次充电。因此为移动服务机器人设计一套自动充电系统[59]解决充电问题已经迫在眉睫。

3.1.2、基于突变点检测的充电桩识别算法

基于突变点检测的充电桩识别算法的基本思想：从集合 R 中筛选出符合设定的突变点变化规律的角度范围，对左右边界角度取平均作为中心角度，以该角度向前行驶，重复上述步骤，直到中心角度的绝对值小于5°，结束校正过程，向前行驶对接。具体算法步骤如下：

对于给定输入的激光数据数组 Rangs[N]，其中 N=360，即激光雷达绕过一度便产生一个距离信息，激光数据集合表示如下：

$$R=\{r_1,r_2,.....,r_i\},\ i\in[0,180)\cup[-180,0) \tag{5-1}$$

其中 r_i 表示的是激光雷达第 i 个角度的距离信息。

步骤 1：数据筛选，将在近程对接范围外或者无效的激光测距数据置零，筛选规则如下：

$$r_i=\begin{cases}0, & r_i=\text{Inf 或 } r_i>2.1\\ r_i, & \text{其他}\end{cases} \tag{5-2}$$

其中 Inf 表示无效的激光雷达数据，2.1 米是判断机器人进入近程对接范围的阈值。

步骤 2：数据补全，对于一些由于激光雷达本身误差导致的无效距离，采用人工补齐的方式将其复原，防止硬件误差带来突变点的错误标定，补全规则如下：

$$r_i=\begin{cases}(r_{i+2}+r_{i-1})/2, & r_i=0 \text{ 且 } r_{i+1}=0 \text{ 且 } |r_{i+2}-r_{i-1}|<0.01\\ (r_{i+1}-r_{i-1})/2, & r_i=0 \text{ 且 } |r_{i+1}-r_{i-1}|<0.01\\ r_i, & \text{其他}\end{cases} \tag{5-3}$$

本文采用的是跨越式补全法，如果无效距离的左右两个距离差的绝对值小于 0.01，则将该角度的距离赋值为左右两个角度距离的平均值。

步骤 3：突变点标定，标定集合 record 定义如下：

$$Record=\{rd_1,rd_2,.....,rd_i\},\ \ i\in[0,180)\cup[-180,0) \tag{5-4}$$

其中 rd_i 表示第 i 个角度的激光雷达测量距离是否发生突变，其初始化值如下：

$$rd_i=0,\ \ i\in[0,180)\cup[-180,0) \tag{5-5}$$

扫描激光数据集合，对突变点进行标定，标定结果存入 *Record* 中，具体标定过程如下：

$$rd_i=\begin{cases}1, & (r_i-r_{i-1})\geq 0.01 \text{ 且 } (r_i-r_{i-1})\leq 0.02\\ -1, & (r_i-r_{i-1})\geq -0.02 \text{ 且 } (r_i-r_{i-1})\leq -0.01\\ 0, & |r_i-r_{i-1}|<0.01\end{cases} \tag{5-6}$$

其中 0.01 和 0.02 是根据特殊形状标定物的突起部分和凹陷部分的高度差 0.012 给出的一个突变范围，在该范围内的突变都可以认为是检测到标定物的特征。

步骤 4：目标位置识别，根据上一步中得到的突变标定集合，判断连续突变点个数符合标定物特征的角度区间，区间左右边界分别表示标定物的左右边界角度。

$$\{m=i,n=j\},|rd_i,rd_j|=1 \text{ 且 } count\geq 3 \text{ 且 } count\leq 5 \tag{5-7}$$

其中 count 表示如下式 5-8 所示：

$$count=\sum_{k=i}^{j}|rd_k| \tag{5-8}$$

count 为[i,j]区间内突变点的个数，由于标定物一共包含 5 个突变点，其中中间三个突变点比较容易被标定，为了容许一定的误差存在，故若检测到三个突变点以上便可确定该标定物位置。

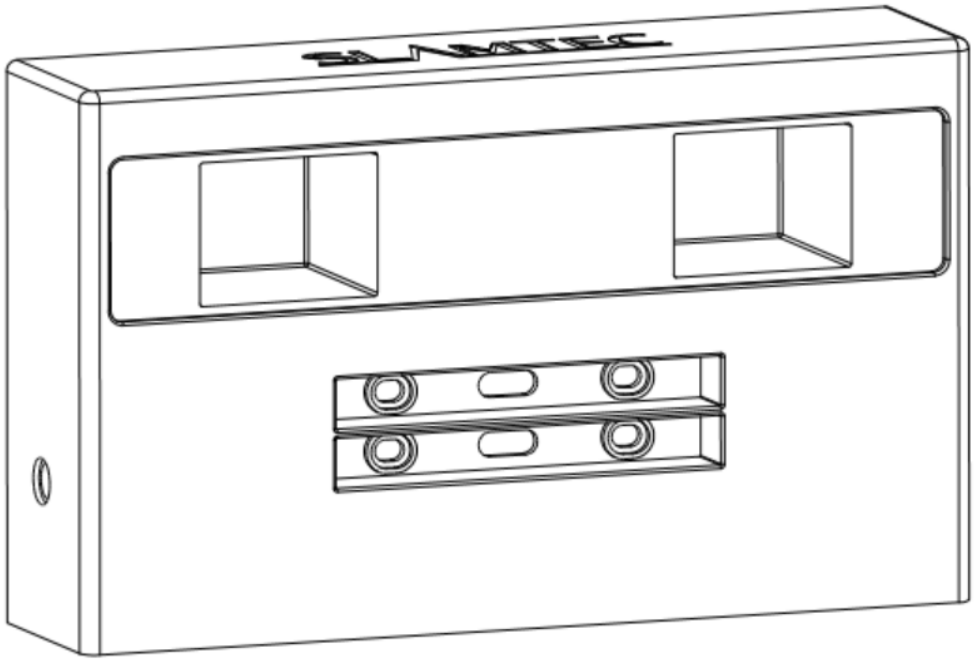
最后取两个边界角度的均值表示为充电桩的中心位置：

$$target=(m+n)/2 \tag{5-9}$$

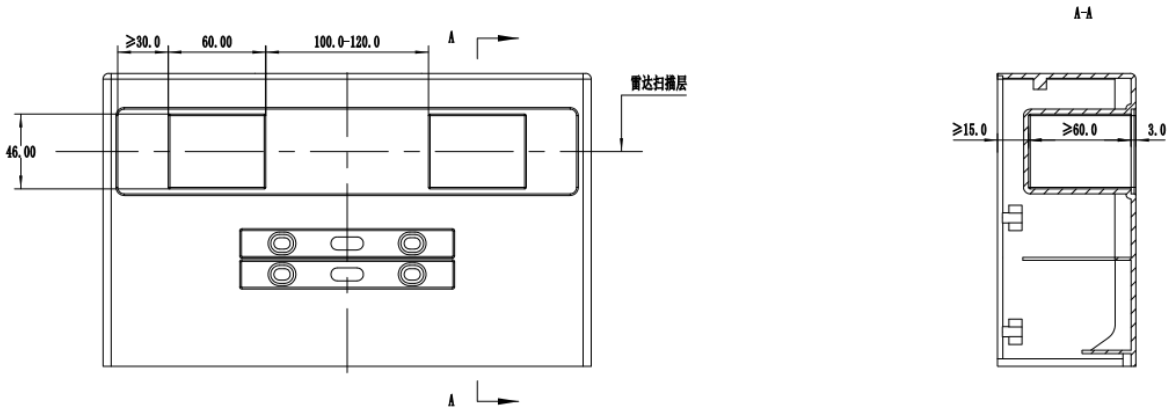
步骤 5：判断$|target|<5^{\circ}$，若成立，则退出识别算法，机器人向前移动完成对接；若不成立，则重复步骤 1-5。

3.2、充电座特征参数（参考）

- 充电座主题外观

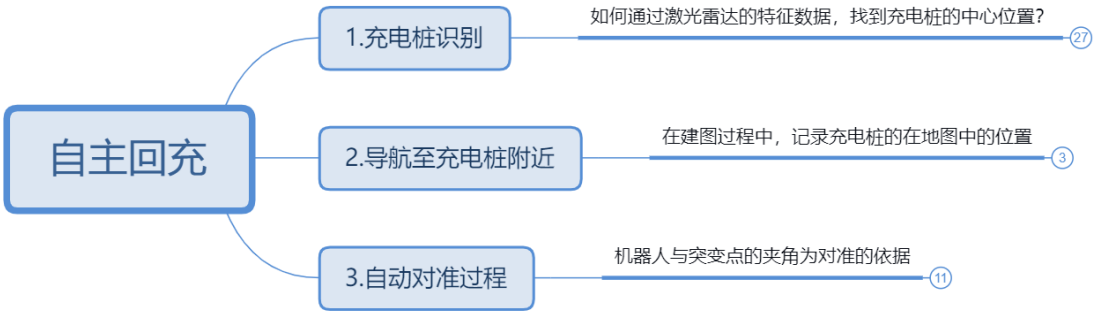


- 充电座具体参数



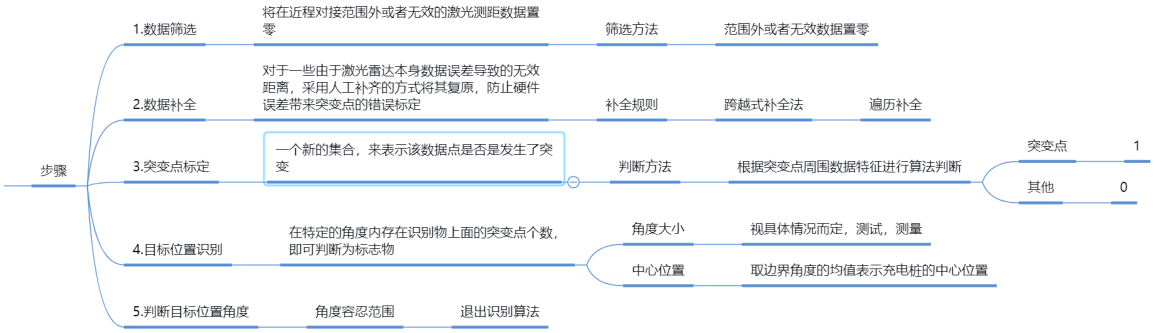
3.3、移动机器人自主回充方案实现

3.3.1、自主回充方案如下

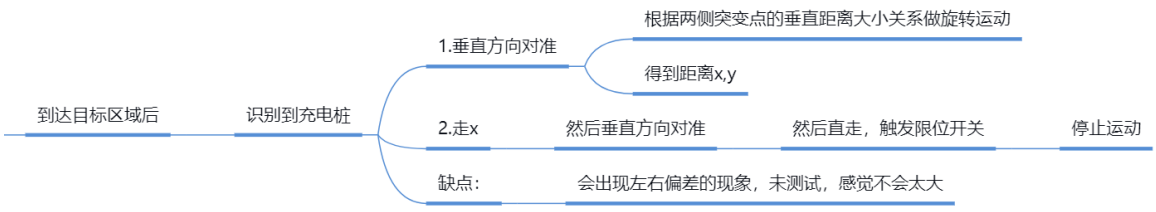


3.3.2、充电桩识别

和上面论文描述的过程一致，这里过一下算法步骤：



3.3.3、自动对准过程



3.3.4、充电座识别代码实现

该源码为我自己手写的一个基于ROS的特征充电座的识别程序，仅订阅Scan，发布充电座的相对于机器人的角度信息、甚至是姿态信息，和论文理论对应。

核心代码如下所示：

```
void laserScanCallback(const sensor_msgs::LaserScan &msg)
{
    // std::cout << "laser_ok" << std::endl;
    char mutation_point_count(0);
    std::vector<int> mutation_point_index;
    mutation_point_index.resize(4);

    laser_ranges.resize(MAX_RANGES_INDEX - MIN_RANGES_INDEX);

    //1.数据筛选
    for(size_t i = MIN_RANGES_INDEX ; i < MAX_RANGES_INDEX; i++) //在指定范围内筛选
数据
    {
        if(std::isinf(msg.ranges[i]) || msg.ranges[i] > MAX_RANGES_FROM_CHANGE)
            laser_ranges[i - MIN_RANGES_INDEX] = 0;
        else
            laser_ranges[i - MIN_RANGES_INDEX] = msg.ranges[i];
    }
    // std::cout<<"1.ok"<<std::endl;

    //2.数据补全
    for(size_t i = 1; i < laser_ranges.size()-2; i++)
    {
        if(laser_ranges[i] < 0.0000001 && (fabs(laser_ranges[i+1] -
laser_ranges[i-1]) < 0.02))//注意使用fabs，这里不要用abs
            laser_ranges[i] = (laser_ranges[i+1] + laser_ranges[i-1]) / 2.0;
        else if(laser_ranges[i] < 0.0000001 && laser_ranges[i+1] < 0.0000001 &&
(fabs(laser_ranges[i+2] - laser_ranges[i-1]) < 0.02))
            laser_ranges[i] = laser_ranges[i+1] = (laser_ranges[i+2] +
laser_ranges[i-1]) / 2.0;
    }
    // std::cout<<"2.ok"<<std::endl;

    //3.突变点标定，寻找突变点
    int mutation_point_num = 0;
    mutation_point.resize(laser_ranges.size());
    for(size_t i = 0; i < laser_ranges.size()-2; i++)
    {
        if((fabs(laser_ranges[i] - laser_ranges[i+1]) > 0.025) &&
(fabs(laser_ranges[i] - laser_ranges[i+1]) < 0.065)) //和自己标志物的形状有关
        {
            mutation_point_num++;
            mutation_point[i] = 1;
        }
        else
        {
            mutation_point[i] = 0;
        }
    }
    // std::cout<<"突变点个数："<<mutation_point_num<<std::endl;

    // std::cout<<"3.ok"<<std::endl;
    //4.确定充电桩中心位置
    //先找到前四个突变点
    for(size_t i = 0; i < mutation_point.size(); i++)
```

```
    {
        if(mutation_point[i] == 1)
        {
            mutation_point_index[mutation_point_count] = i;//把突变点代表的序号，放
入
            mutation_point_count++;
        }
        if(mutation_point_count == 4) break;
    }

    //依次遍历，直到找到符合要求的连续四个突变点
    if(mutation_point_count == 4)//前提是至少有四个突变点，否则没有继续计算的必要
    {
        static bool find_mutation_flag = 0;
        for(size_t i = 0; i < mutation_point.size(); i++)
        {
            //特征点的距离大小特征
            if(laser_ranges[mutation_point_index[0]] <
laser_ranges[mutation_point_index[1]] &&
                laser_ranges[mutation_point_index[2]] <
laser_ranges[mutation_point_index[3]] )
            {
                // std::cout<<" 满足特征限制0 "<<std::endl;
                //特征点的距离大小特征
                if(laser_ranges[mutation_point_index[0]] <
laser_ranges[mutation_point_index[0] - 5] &&
                    laser_ranges[mutation_point_index[3]] <
laser_ranges[mutation_point_index[3] + 5] )
                {
                    // std::cout<<" 满足特征限制1 "<<std::endl;
                    //特征点的间隔特征
                    if(abs(mutation_point_index[0] - mutation_point_index[1]) <
7 &&
                        abs(mutation_point_index[2] - mutation_point_index[3]) <
7 )
                    {
                        // std::cout<<" 满足特征限制2 "<<std::endl;
                        //特征点距离相差特征
                        if( fabs(laser_ranges[mutation_point_index[0]] -
laser_ranges[mutation_point_index[1]]) < 0.08 &&
                            fabs(laser_ranges[mutation_point_index[2]] -
laser_ranges[mutation_point_index[3]]) < 0.08 )
                        {
                            // std::cout<<" 满足特征限制3 "<<std::endl;
                            find_mutation_flag  = 1;//找到突变点

                            angle_left_charge   = MIN_RANGES_INDEX +
mutation_point_index[3];
                            angle_right_charge  = MIN_RANGES_INDEX +
mutation_point_index[0];

                            ranges_left_charge  =
laser_ranges[mutation_point_index[3]];
                            ranges_right_charge =
laser_ranges[mutation_point_index[0]] + 0.06;//因为突变点的厚度0.06m

                            //虽然理论上不在中间，但是可以这样简单的用
```

```
                            angle_centre_charge = (int)(0.5 + (angle_left_charge
+ angle_right_charge)/ 2.0);
                            ranges_centre_charge=
msg.ranges[angle_centre_charge];

                            charge_info.data.resize(6);
                            charge_info.data[0]=angle_left_charge;
                            charge_info.data[1]=ranges_left_charge;
                            charge_info.data[2]=angle_right_charge;
                            charge_info.data[3]=ranges_right_charge;
                            charge_info.data[4]=angle_centre_charge;
                            charge_info.data[5]=ranges_centre_charge;

                            break;
                        }
                    }
                }
            }

            //更新突变点序号
            mutation_point_index[0]=mutation_point_index[1];
            mutation_point_index[1]=mutation_point_index[2];
            mutation_point_index[2]=mutation_point_index[3];

            for(size_t j = mutation_point_index[3] + 1; j <
mutation_point.size(); j++)//这里要从下一个开始，而不是当前
            {
                if(mutation_point[j] == 1)
                {
                    mutation_point_index[3] = j;
                    break;
                }
            }
        }
        //找到突变点
        if(find_mutation_flag == 1)
        {
            find_mutation_flag = 0;
        }
        else
        {
            //不满足情况数据清零
            charge_info.data.clear();
    }
    else
    {
        //不满足情况数据清零
        charge_info.data.clear();
    }
    // std::cout<<"4.ok"<<std::endl;
}
```

4、分享

在公众号**小白学移动机器人**，发送：**自主回充**，即可获得源码以及论文等相关资料。


小白学移动机器人
公众号
微信群
专注于移动机器人领域技术分享，涉及机器人底层驱动、ROS、激光SLAM、运动规划……相关内容。
您的点赞就是对我最大的鼓励!


点个赞 再走吧！