机器人导航总结

参考视频：[1]B站-机器人导航《浅谈机器人导航与避障》薛午阳（睿慕课）

参考书籍：[2]《机器人学、机器视觉与控制》、[3]tutlebot3中文版书

1. 自己总结：

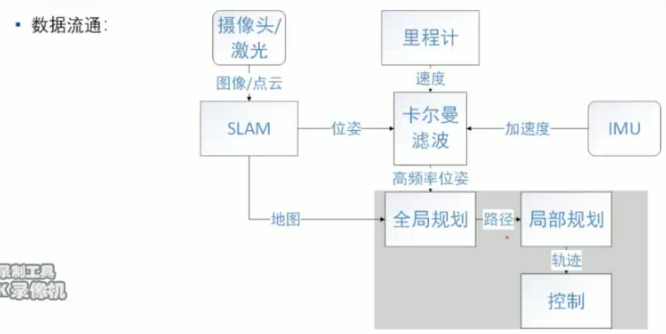
* 任务类型：找东西（先去某个特定位置找，没找到再扩大搜索范围）、到某个坐标点、遍历整个区域作路径规划
* 空间覆盖，扫地机器人会面临不同的地方，沿墙边，避障、大的自由空间等
* 参考网站：

<https://blog.csdn.net/hjhjhx26364/article/details/84100754>

* 1

《浅谈机器人导航与避障》

1. 机器人导航系统各个模块



1. 全局导航算法：给出通往目的地的路径

* Dijkstra：主要特点是以起始点为中心向外层层扩展，直到扩展到终点为止。
* A\*：与Dijkstra区别，启发函数，待处理队列排队时在已走过的距离上加入对未来预期要走的距离作为排队依据。Dijkstra均匀往外搜索，A\*会根据未来预期距离有目的的往外搜索，如下预期距离为不管障碍物和斜着走的距离

<http://theory.stanford.edu/~amitp/GameProgramming/>

<https://blog.csdn.net/denghecsdn/article/details/78778769>

<https://blog.csdn.net/tiandijun/article/details/62226163>

* Probabilistic road map（PRM）:算法流程：在无障碍物空间中随机采样n点；把n个点按照可连通性连接，获得概率路线图G；设定起点和终点，分别找出离起点和终点最近的图节点V属于G；以两个节点点为起点和终点使用A\*或Dijkstra算法搜索路径。

优点是：最简单，缺点是：线路非最短；狭长处较难采样

* Voronoi（沃罗诺伊）：取出地图中空白区域；对空白区域执行称为细化获得骨架图G；设定起点和终点，分别找出离起点和终点最近的骨架点V属于G;以两个骨架点为起点和终点使用A\*或Dijkstra算法搜索路径。

优点：完整的地图骨架网络；最安全的线路；缺点：线路非最短

* 可视图（visibility graph）：取出图中空白区域；将空白区域边界角点V取出两两间按连通性连成G；把起点和终点作为节点按连通性加入G；使用A\*或Dijkstra算法搜索路径。

优点：线路最短；缺点：不安全

* 快速扩展随机树（rapidly exploring tree，RRT）：以起点为根节点初始化树T；随机采样位姿，按可达条件加入到树T的叶子；重复上一步直到目的地加入为树T的叶子

优点:直接考虑了控制；缺点路径非最短；慢；路径会很奇怪；狭长处较难采样

1. 环境（地图）表示：网格地图和八叉树（octomap an efficient probabilistic 3D mapping framework based on octrees）
2. 局部避障算法：在不走错路的前提下，躲避行走中遇到的障碍物

* 人工势场法（artificial potential field，APF）：设计目的地和障碍物的势场公式；用势场梯度计算机器人的参考速度。

优点：不需要再计算轨迹，计算量小；缺点：易陷入局部最低点；无法保证绝对无碰撞；狭窄处无法通过；狭窄通道中摆动。

* 向量场直方图（vector field histogram，VFH）：根据不同角度方向障碍物距离计算直方图；根据直方图，阈值以下的空间即为安全空间

优点：计算量小；缺点：阈值选择对避障影响大（半径）；没考虑运动限制

* 动态窗口法（dynamic window approach，DWA）：根据当前速度、机器人加速度和时间步（将一条轨迹分成离散的很多步）限制划出采样空间，大量采样不同轨迹，抛弃有碰撞的轨迹；对无碰撞的轨迹进行评分，选取获得分数最高的轨迹

优点：考虑了轨迹和碰撞；缺点：计算量较大；调参困难；容易忽略狭窄通道

1. 轨迹与控制：论文（Minimum Snap Trajectory Generation and Control for Quadrotor）、《机器人学、机器视觉与控制》

《机器人学、机器视觉与控制》

1. 路径是空间中从初始位姿过渡到最终位姿的一个图形。轨迹是具有特定时间属性的一条路径。轨迹的一个重要特性是要平滑-位置和姿态随时间流畅的变化（连续可导）。通常速度和加速度必须是连续的，有时加速度的导数或加速度率也需要连续
2. 时间与运动（p39）总结有相关文献：

* 一维：即直线运动：
* 时间多项式函数：常用五次多项式，根据起始和终止位置、速度、加速度可以求出多项式系数，便可通过公式求出各个时刻的加速度和速度
* 混合曲线轨迹（因由匀速线段和两条抛物线组成而得名lspb）：梯形速度轨迹，速度光滑但加速度不光滑
* 多维：如二维将曲线分成多个时间步（多段）进行近似，求出时间步\*2的矩阵，每行对应一个时间步，每列对应一个轴
* 多段轨迹：
* 三维空间姿态插值：

1. 位形空间和任务空间：位形空间是所有可能的位置和形态的集合（流形），任务空间所有可能额机器人位姿的集合。通常任务空间自由度多余位形空间
2. P62页双轮自行车模型小车的运动学模型。转动速率、朝向转速或偏摆转速w可以由陀螺仪测量，也可以由左右轮的角速度值推算出来，左右轮转动半径不一样，所以其转速也不一样。到达点、跟踪直线、跟踪曲线（相当于跟踪动点）、到达某个固定姿态（自行车模型和两轮加万向轮不一样，自行车模型没有线速度不能转向。
3. 单纯靠航迹推算、激光slam和视觉slam这种递推式方法，难以避免误差的积累，要减小这种误差积累，可以加入已知准确位置的固定路标点
4. EKF框架允许采用多个以及多种传感器的数据来更新状态，这就是为什么该估计问题也被称作传感器融合。可以用多种传感器观测数据来更新状态。
5. 粒子滤波器是一种多假设估计器，可以解决误匹配或者路标被遮挡的情况，会对各种情况进行可能性建模
6. 反应式导航：现状是很多机器人在完成任务时并不依赖任何地图，如寻着光源前行、沿着地上白线走、沿着墙壁穿越迷宫和沿随机路线走
7. 基于地图的类人导航方法支持更复杂的机器人任务，需要地图和定位
8. 基于地图的路径规划常用算法：规划和查询，常只考虑距离，但有时需要考虑运动学和动力学产生的成本，如颠簸带来的磨损、能否控制电机沿着路径走。

假设条件是：机器人子网格中运动，且只占一格单元格（机器人可能不止占一个单元格）；机器人不含任何非完整约束，且可以移动到相邻单元格（运动学和动力学不满足全方向运动能力）；机器人可以确定它在平面上的位置（slam）；机器人可以使用地图计算它要走的路径（路径规划）

* 距离变换
* A\*
* D\*（A\*的升级版）：允许成本地图改变后重新计算路径，但他不支持改变目标。若B为目标，那以B为中心产生规划，然后根据不同A查询
* 沃罗诺伊（Voronoi）路线图法：细化或骨架化也与距离变换一样计算量很大的迭代算法，但它友好为我们展示了自由空间寻找路径的方法和原理。根据障碍物生成骨架，那么A、B都可以改变
* 概率路线图方法（PRM）：首先在自由空间中找N个随机点，然后用直线将每个点与其相近的邻点连接，不穿过障碍物，这样就形成点数最少且开环的网络图。查询时，从起点移动到离他最近的节点，然后按路线图走到离终点最近的节点，最后从节点走到终点
* 快速搜索随机树（rapidly-exploring random，RRT）:考虑了机器人的运动模型，也是一种概率算法

1. Dijkstra算法也即：单源最短路径算法。算法的基本思想是：每次找到离源点最近的一个顶点，然后以该顶点为中心进行扩展，最终得到源点到其余所有点的最短路径
2. 为了避免机器人与障碍物碰撞，可以使用闵可夫斯基和（minkowski sum）使障碍物膨胀
3. 导航时通过使用两种代价地图存储周围环境中的障碍信息：一种用于全局路径规划，一种用于本地路径规划和实时避障
4. DWA:<https://blog.csdn.net/heyijia0327/article/details/44983551>
5. 卡尔曼滤波：假设满足高斯分布
6. 蒙特卡罗定位：其也被称作粒子滤波器，无需对误差分布做任何假定。基本思路是：同时保存许多不同版本的机器人状态向量。当出现新的测量信息时，我们用每个版本的状态向量去解释这些数据，并给解释的好坏打分。我们从中挑选出解释最吻合的一组状态，然后给他们一个随机扰动以产生一组新的状态。多个这些可能的状态向量及其得分值汇总在一起，就能近似得出我们想要估计状态的概率密度函数。

[3]turtlebot3

1. 根据导航算法会有不同，机器人实现自主导航应该至少需要如下几种：

* 地图：P349
* 测量或估计机器人的姿态的功能
* 识别障碍物，如墙壁和物体的功能
* 能够计算出最优路线并行驶的功能:p370

1. 1