

自己动手写全套无人驾驶算法系列（一）机器人规划

一、概述

1.1 系列整体概述：

本人最近写完了整套无人驾驶的各种算法，近期重构完会一一传到 github，部分测试算法未传，至于为什么先写规划，主要是由于规划比较好重构，BUG 较少，但还是不少算法处于测试阶段，待测试完毕一一传上来再描述，本系列主要简略介绍各个算法的原理，因为比较忙写的会比较仓促，但之后有时间会慢慢详细描述。具体见我的 github：

<https://github.com/wujiazheng2020/>

欢迎大家对我的 github 项目提出 BUG，或者需要我更新新的算法，或者需要技术合作，都可以联系我：

163 邮箱：wujiazheng2020@163.com

gmail:wujiazheng2020@gmail.com

QQ 群：710805413

1.2 规划模块概述

自动驾驶规划包括全局规划和局部规划，无人车和扫地机器人都是如此。

在无人汽车中先由 map 根据 lattice(3.2)，三次样条插值(3.3)，MPC 等产生 map refer line，有些无人车也可以选择用百度地图产生全局规划，之后对于局部规划无人车采用 lattice 进行变道、停车、倒车入库、避障，也可以选择 Reeds-shepp 曲线(3.3)倒车入库，不过 Reeds-shepp 曲线本身是不考虑障碍物的。

总之，对于无人汽车，实际过程中效果比较好的正常行驶规划控制组合往往是三次样条 /Lattice + 模型预测路线 +MPC，这种算法实际测试有比较好的效果，而对于定点停车，倒车入库等，模型预测路线+PID 预瞄往往是比较好的组合，MPC 我们将在控制模块讲到，和控制模块一起讲是比较好的选择

而对于扫地机器人，迎宾机器人等往往采用 Dijkstra/A*(2.1,2.2) +(三次样条插值) + Dwa(3.1)，而对于扫地机器人的规划(2.6)，我的项目也作了实现可以参考。

原定 10 大全局规划+6 大局部规划，已经发布的有 6 大全局规划，4 大局部规划，剩下的之后调试完发布。

规划项目见我的 github：https://github.com/wujiazheng2020/WJZ_Planner

以下图片也均来自我的项目

1.3 目录

一、全局规划

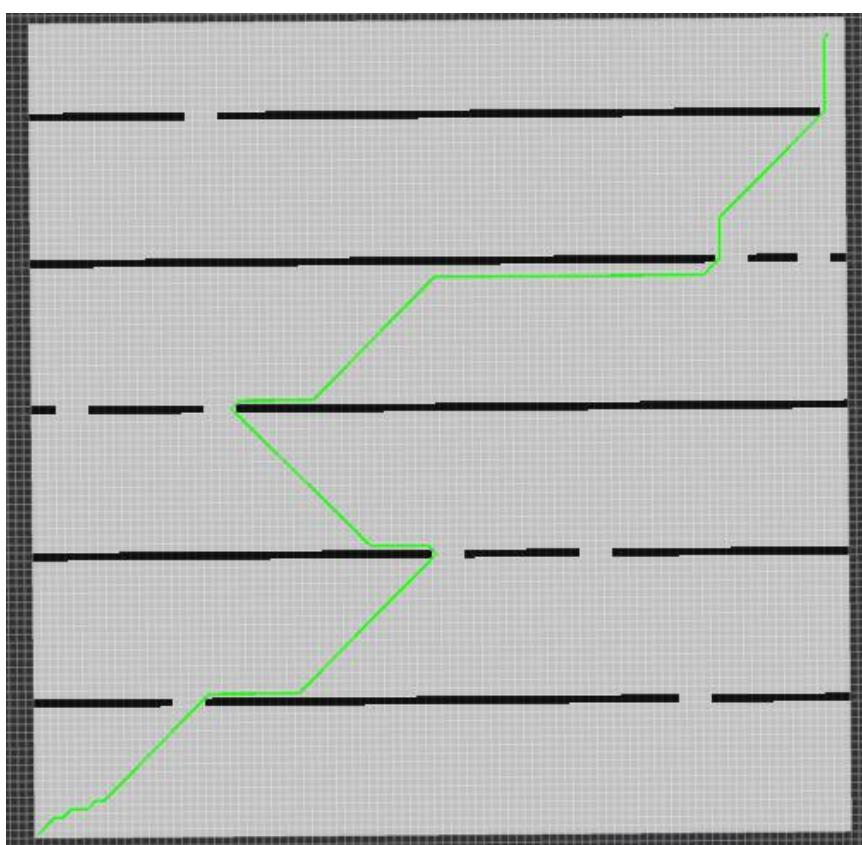
- 2.1 Dijkstra
- 2.2 A Star
- 2.3 RPM
- 2.4 RRT
- 2.5 RRT Star
- 2.6 Sweep robot planner

二、局部规划

- 3.1 Dwa
- 3.2 State Lattice (auto car only)
- 3.3 Reeds shepp (auto car only)
- 3.4 Cubic Spline

二、全局规划

- 2.1 Dijkstra

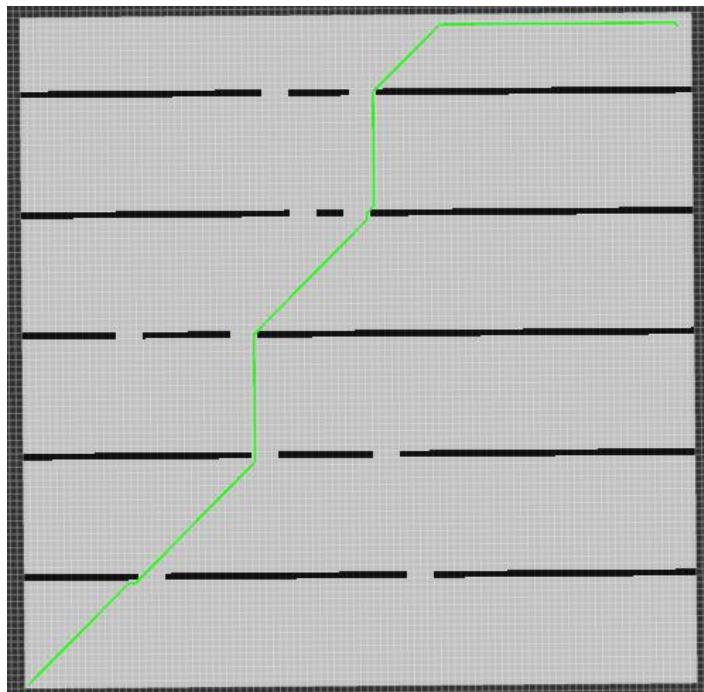


dijkstra 算法的目的是找寻两点间的最短路线，大家在数据结构的书上经常看到，就是有两个列表，一个开启列表一个关闭列表，每次从开启列表里找一个到起点的本轮距离最小的点，在开启列表中删除，在关闭列表中加入，然后对那个点计算他所有能到的并且在开启列

表中的点的距离，如果到那个点的距离 A 加上自己的本轮 B 距离小于那个点本轮的距离就更新那个点的距离为 A+B,就这么简单.而在栅格地图中对本轮距离最小的那个点只需要探索周围 8 个点就 OK 了，而为了输出路径，每次更新 A+B 的时候，让一个静态指向数组 $\text{pre}(\text{now_id}) = \text{before_id}$ 就可以 ,具体见我的 github

◦

2,2 A Star

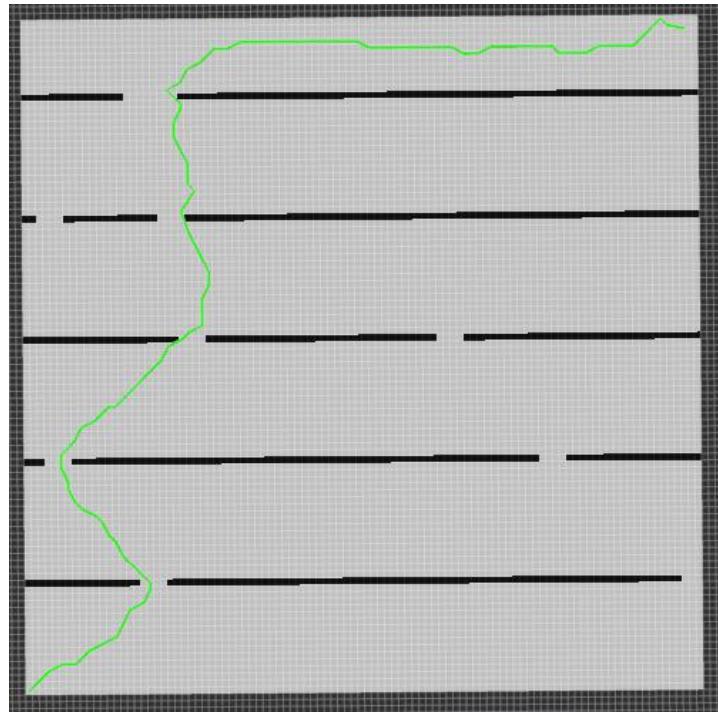


A Star 算法就是 dijkstra + 指示函数，把找本轮距离最小的那个点改为找本轮距离加上该点到终点的指示函数的和最小的点，可见 A star 算法介绍，为了算的更快，指示函数可以提前计算好，性能可以提升 4-5 倍。

A Star 算法目的是提高 dijkstra 的速度，但对于迷宫环境或者障碍物多的环境，dijsktra 优于 A Star，而对于无人车来说，前方障碍物很少，或者短路经时，A Star 就优于 dijsktra。

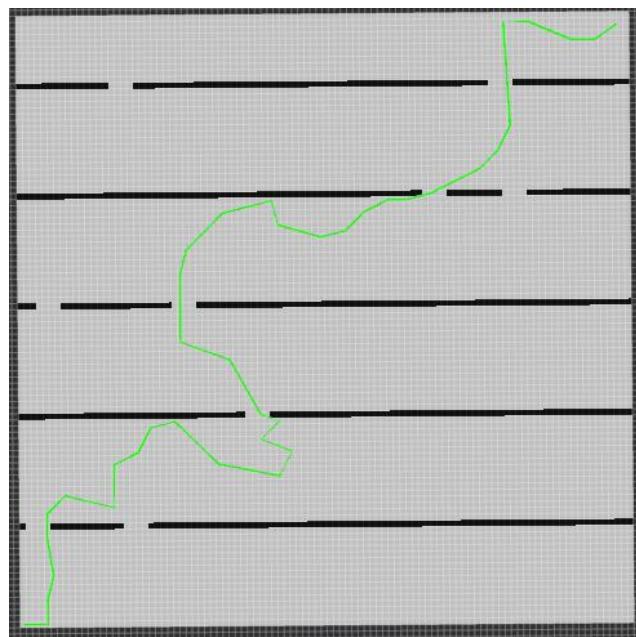
值得一提的是 A Star 在无人汽车中常与 Reeds shepp 组成 hybrid A star，这样做规划就符合运动学约束了，不过很耗计算量需要哦提前计算.

2,3 RPM



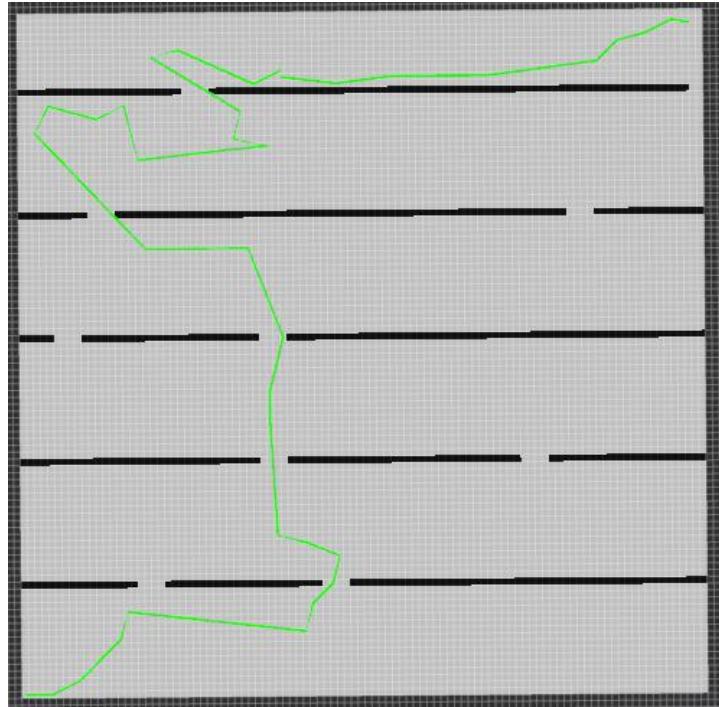
就是在地图中随机撒一些点，然后用 dijkstra 或者 A Star 寻找最短路径，如果靠终点距离小于某值就停止。主要是机械臂使用或者为了探测能否可达

2,4 RRT



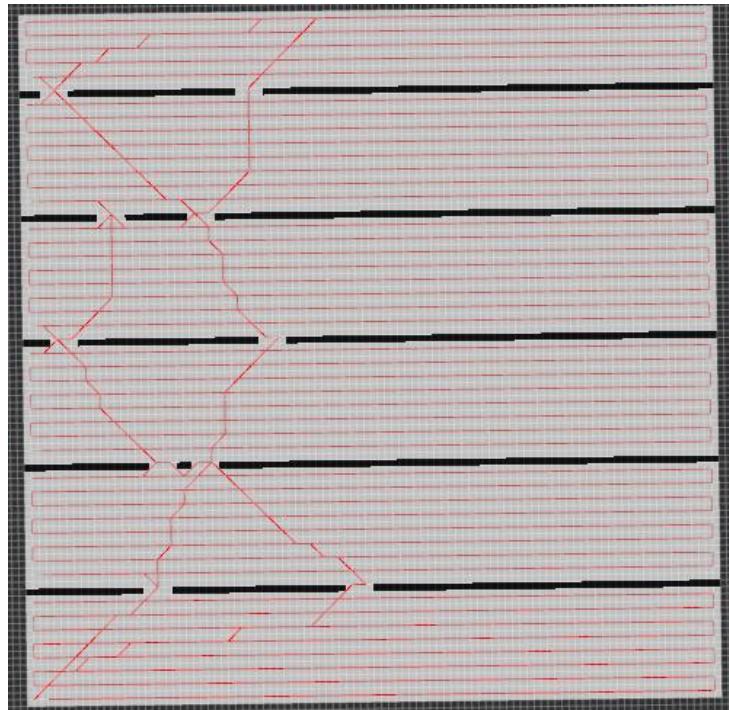
主要是机械臂使用或者为了探测，就是每次在地图中随机生成一个点，找到已生成树上最近的一个点（用 kd-tree），然后让这个点朝着那个随机点的方向生长，最大长度为 lmax，如果有障碍物就停止生长，同样利用 pre 数组记录路径，当树生长到离终点距离小于某个值的时候停止并输出。

2,5 RRT Star



就是每次对离随机点连线上最大距离那个点 P，搜索他周围离他距离小于 r_{min} 的已生成树上的点，对那些点回溯到起点，看哪个到起点最近并且没有障碍物阻拦就把哪个点连到 P。

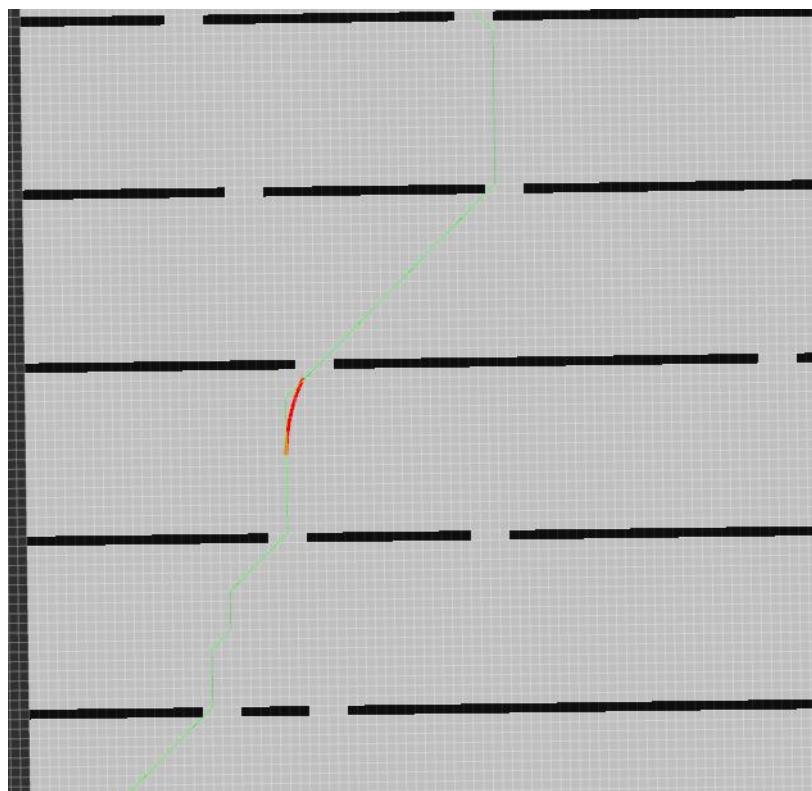
2,6 Sweep robot planner



扫地机器人的规划，先一直向左到障碍物，再一直向右到障碍物，再探测上有无障碍物，如无就上一格，如有就探测下一格，如无就下一格，如果四个方向都有障碍物，那就用 kd-tree 找到最近一个未探测的用 Dijkstra 或者 A star 走过去。重复，独创算法。

三、局部规划

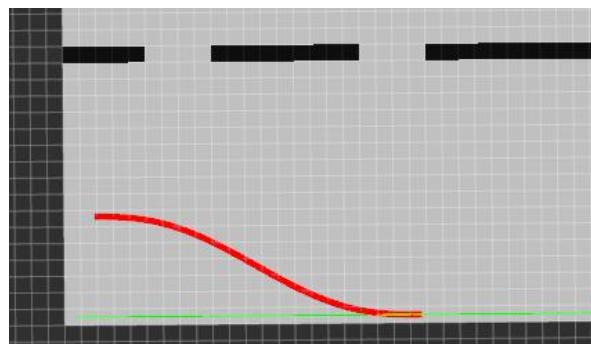
3.1 Dwa



图中红色的线就是 DWA 产生的轨迹，dwa 中文动态窗口法，就是比如当前速度为 v ，角速度为 w ，那么从 $(v-v_{min})$ 到 $(v+v_{max})$ 以某一间隔采样若干个速度 v ，同样按此方法采样若干个角速度 w ，那么把 v 和 w 两两组合，共 $v \times w$ 种组合，按时间间隔 dt 以差分驱动模型生成路径，然后评价每条路径的分数，选择分数最高的那个路径即可，分数的评价标准包括曲线上距离障碍物最近点的距离，终点距离全局路径朝向的差平方，终点距离全局路径中最近点的距离（也可以是垂直距离）等。

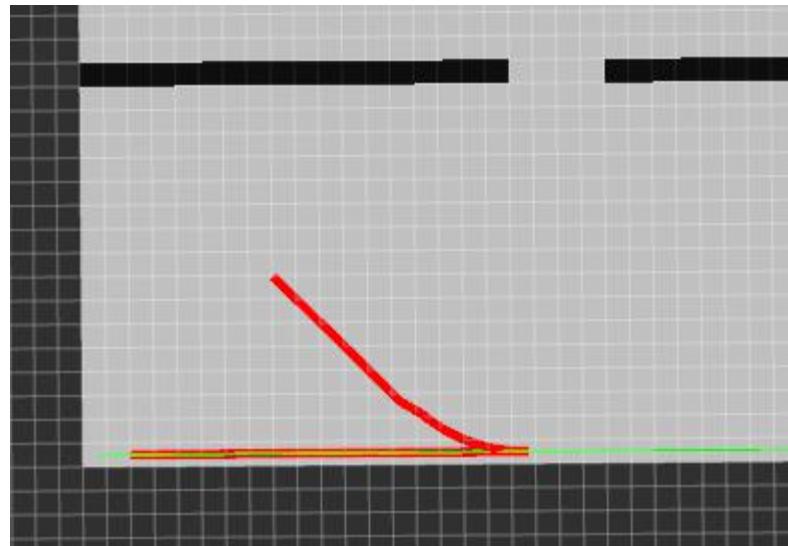
而对于差分驱动模型，它主要用于扫地机器人，无人汽车可见单车模型，但无人汽车一般不用 dwa.

3.2 State Lattice



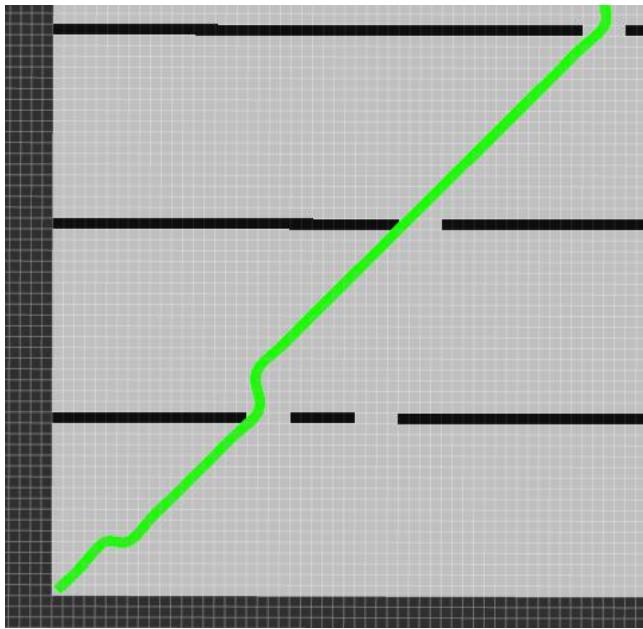
就先把当前位置转换到 frenet，frenet 介绍见[这里](#)，然后对横向进行 5 次拟合，纵向进行 4 次拟合即可，因为横向轨迹的终状态横向距离有 a 种，对应纵向距离有 b 种，故刚好有 ab 种组合，而纵向距离终状态的时间有 c 种组合，速度有 d 种，故共有 cd 种组合，我们可以先用 t 算出纵向距离 s ，再把 s 带入横向曲线得到对应横向距离，这样我们可以组合纵向和横向曲线得到多条曲线，对每条曲线按 ST 图障碍物，速度，加速度，终点与参考线方向角等进行评分，选出评分最大的那个即可。

3.3 Reeds Shepp



我在 [github](#) 中主要考虑最多出现的两种情况，与之相对的是 dubins 曲线，但由于一般车可以后退，故就不实现 dubins 了，这个算法中，我们采用的是无人汽车用的单车模型，主要用于停车，RS 曲线又被称为最大转向曲线。项目中我采用沿着曲线模拟搜索的方法一步步探测到距离重点航线线横向距离最近的转向点后用 RS 曲线到达终点。

3.4 Cubic spline



三次样条插值，即对每一段都进行插值，目的是让曲线变平滑，左右端点满足一阶，二阶，三阶条件，即可很容易推导得。

今天就这么多了，因为还有很重要的事，之后介绍会慢慢补上。