浙江大学 实验报告

课程名称：智能移动技术　　指导老师：　熊蓉，王越　　成绩：
实验名称：导航规划大作业　　组号：9

小组成员如下:

学生姓名：　王　威　　学号：　3210105001
学生姓名：　潘力豪　　学号：　3210100385
学生姓名：　杨安桐　　学号：　3210102194

一、实验内容

任务要求--分别在 simple 和 hard 场景下完成路径规划和速度规划两个任务，其中 simple 场景为静态障碍物场景，hard 场景为动态障碍物场景。（要求规划蓝队零号机器人，从(2400,1500)运动到坐标为 (-2400,-1500) 的位置）路径规划：通过 debug 信息绘制路径到 Client 面板，点击 simple 按钮进行场景重置，需要可以看到重新规划，要求每个场景进行至少 5 次场景重置，并均在面板看到所规划路径（将机器人放置于 (2400, 1500)，规划可以到(-2400,-1500)的路径）轨迹规划：通过 cmd 下发控制指令 ，控制机器人在两点之间往返运动 5 次以上。其中，单程到达目标点的判定标准 为：机器人停下的坐标与目标点之间距离小于 100（10cm），否则该运动不计入有效往返次数

二、实验思路

①路径规划算法(RRT*算法)设计思路：

初始化：设定起始点 start 和目标点 goal，并创建一个只包含 start 的 RRT 树 T
重复步骤直到找到路径或达到最大迭代次数：

a. 随机采样：在环境空间中随机采样一个点 rand_node

b. 扩展树：从树 T 中找到最近的节点 nearest_node，以 nearest_node 为起点，在方向上延伸一定的距离，得到新的节点 new_node（如果在目标点范围内则直接选择 goal 作为 new_node）

c. 碰撞检测：检测路径 new_node 到 nearest_node 之间是否与障碍物发生碰撞，如果发生碰撞，则返回步骤 2 继续下一次迭代

d.寻找最优连接：对于树 T 中与 new_node 距离在一定范围内多个节点 NEAR_NODES，计算通过 NEAR_NODES 连接到 new_node 的代价 cost，选择代价最小的连接方式，选择 min_node 和 min_cost

e. 更新树：将 new_node 加入树 T，并更新节点间的连接关系和代价信息

f. 判断终止条件：如果新加入的节点 x_new 接近目标点 goal，则找到了一条可行路径，算法结束

g 终止后得到最终的路径树

h. 优化路径：对树 T 中的部分节点，以目标点 goal 为目标进行优化，通过调整连接关系和代价信息，改善路径的质量(由于我们运动算法的思路是点到点，优化思路也是尽可能减少点的数量，同时避免障碍物过多的区域)

②DWA 算法设计思路

在 DWA 算法中，三个价值评估函数 heading_evaluation()， dist_evaluation()，velocity_evaluation() 分别返回方向耗散（朝向目标点越近 cost 越小），避障耗散和速度耗散（速度越大耗散越小），然后 在 velocity_planning()函数中调用以上三个函数，结合当前的速度和角速度，给出 predict_time 的时间 内可能的速度和角速度空间，并搜索其中总耗散最小的路径，输出对应的速度和角速度。

velocity_planning()中，调用了 move()函数来通过当前的速度和角速度指令，预测出 predict_time 的时间内小车的轨迹并计算 cost，实现避障。

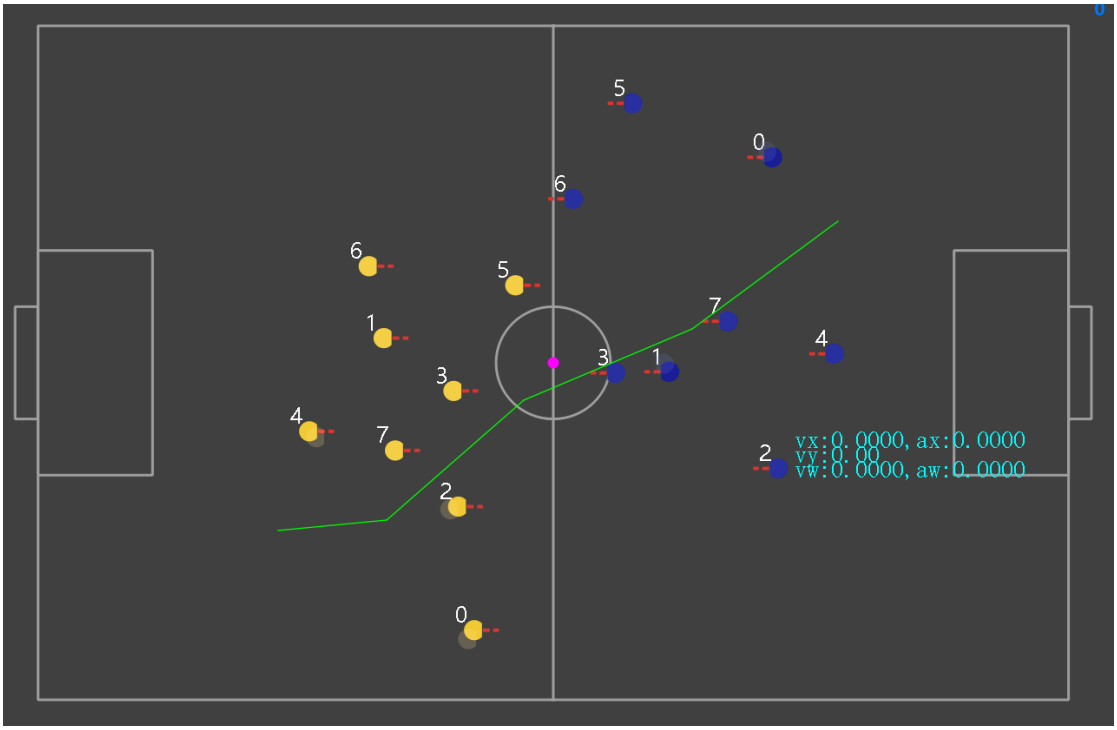
③RRT 和 DWA 结合思路

由于我们的 DWA 算法思路设计是从一个点到另一个点，而 RRT 是规划一个点从另一个点的路径，我们便将思路确定为，先生成包含几个点的路径（一般小于 8），然后分别采用从当前位置到下一个规划点的思路，由此将小车连贯起来，使得其不用考虑行走的过程，只用考虑一个点到另一个点，同时进一步优化 RRT*，保持生成点的数量在合理范围内，同时尽量避开障碍物多的区域，同时考虑时效性，从而完成 Task2 的任务。

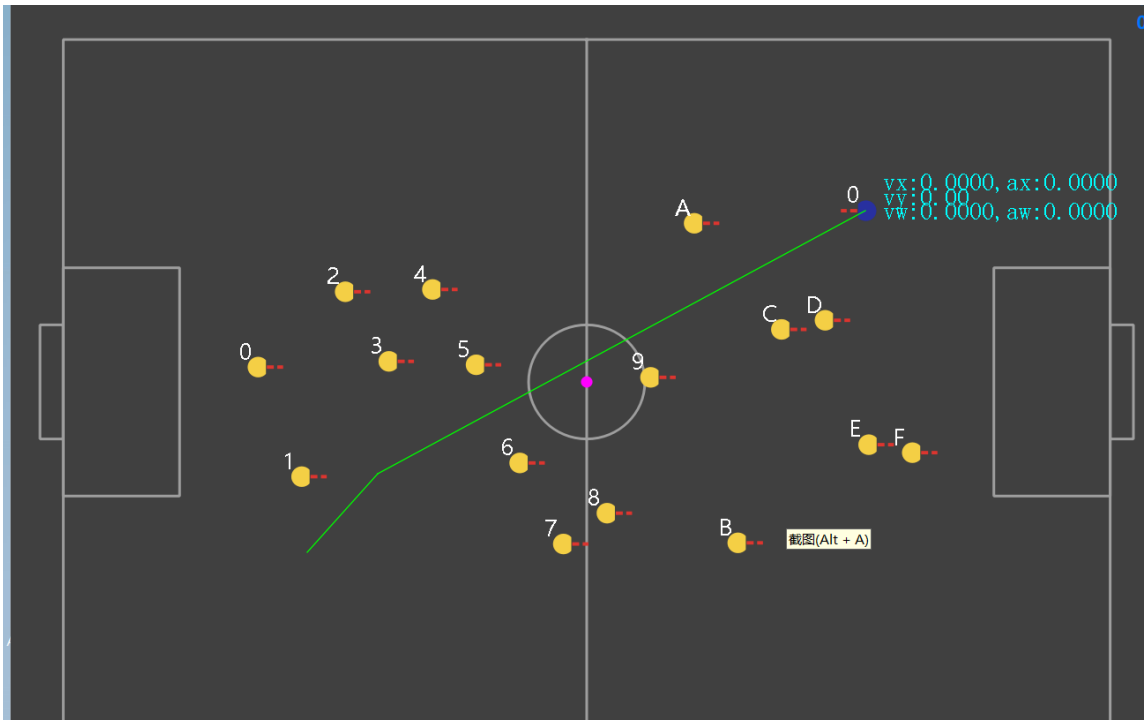
三、实验结果与分析讨论

开始的时候，生成路径跑的及其缓慢，往往需要几分钟，甚至可能导致时间过长而程序报错，为此，在 plan 函数中加入了判定在一定范围内便选择 goal 作为终点的代码，由此大大加快迭代成功率，同时为了进一步加快，在 sample 函数中加入可以直接将随机点作为 new_node 的代码（虽然在后面生成结果的过程发现，这样会使路径质量变得很差），使得迭代速度快了 2 倍，可以实现在 1s 中显示至少 4 种不同的路径。

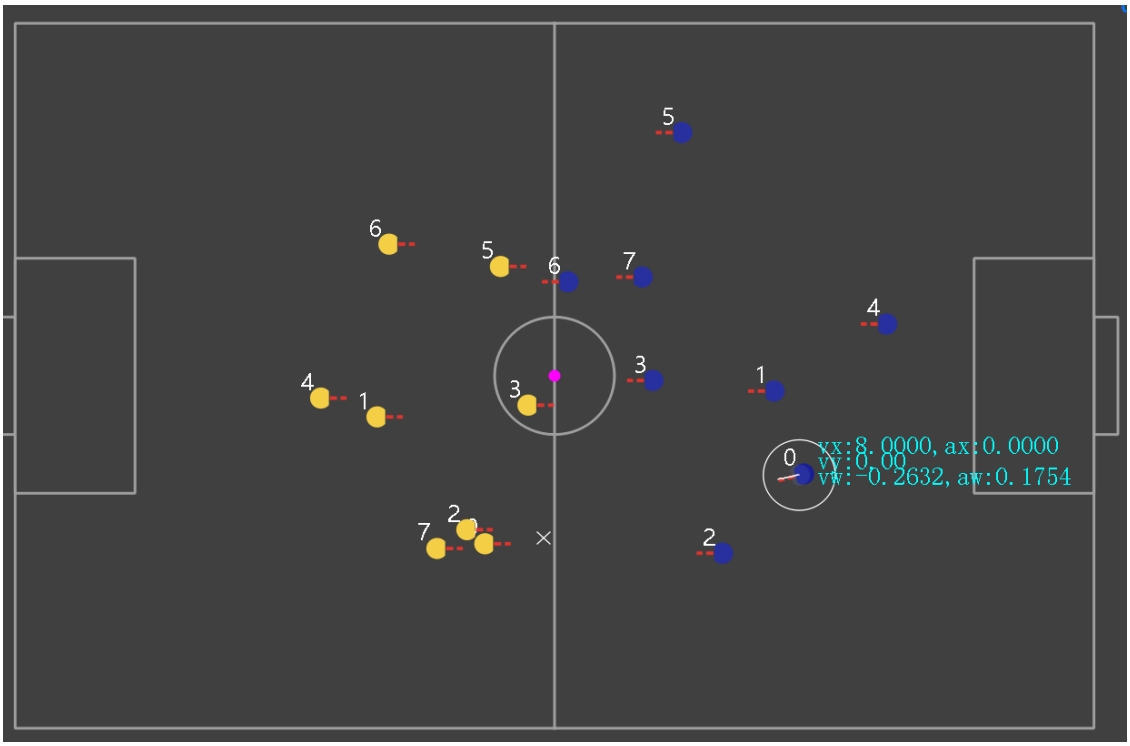
在做 task1_simple 部分的时候,开始经常遇到无限迭代导致无法生成路径的情况出现，通过修改迭代次数的判断标准和生成判定生成 new_node 的标准，使得代码以跑通，同时在由于 task1 属于 simple 类型。同时存在黄和蓝两种颜色的障碍物，我们并没有把蓝色加入障碍物的考虑中，导致出现会越过蓝色生成路径的情况，同样的情况也在我们的 dwa 代码中出现，我们在测试路径能否跑通时，出现会在规划的路径上撞不到黄色但是能撞到蓝色的情况，在加入判定蓝色的代码后，上述情况均得到解决，得到如下结果：



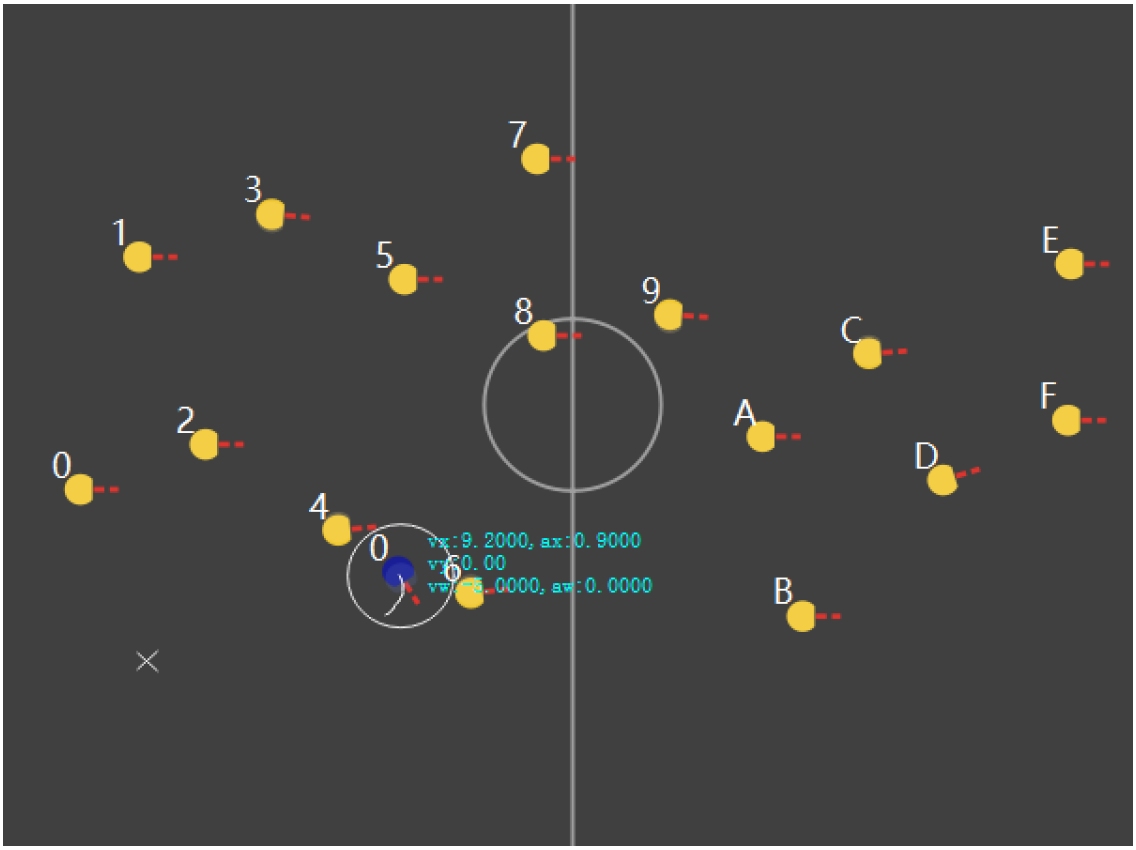
Task1_hard 部分的过程中，有时候路径生成会绕很大的弯，导致小车往返需要花费很多的时间，为此我们修改了 step 的范围，将其缩小，使得小车不至于绕那么大弯，但是接下来又出现了可能离的太近的情况，为此我们增大了 avoid_size 这个参数，使得避免碰撞，于是我们反复多次调整这两个参数，取中间值，使得得到一个较为完美的路径（综合时间和质量）。



Task2_simple 部分，由于前面已经完善了同时考虑红色和绿色的障碍物的代码部分，因此在本次调试中并没有出现碰到障碍物的情况，由于我们避障算法写的比较给力，就算路径出现比较曲折的情况，也能成功避开障碍物，但某些情况存在机器人原地转圈的情况，在修改避障某些没有必要的部分后，成功达成比较完美的效果。



Task2_hard 部分，对 task_2 hard 的动态避障，我们直接采用了 dwa 没有使用 RRT，因为生成的路径可能会被移动的障碍物阻挡。总体的避障效果非常好，小车能够很好的避开障碍物：



四、遇到的问题与优化

在调试 RRt*算法的过程中，我们开始的时候生成的点数很多，并且存在有可能碰到障碍物的情况，发现是生成随机点函数 sample()的问题，去掉其中为快速生成而直接取随机点为 new_node 的代码后，效果好了很多。

同时对参数 step 进行修改，当 step 过大时，路径很不好，会绕很大的弯，太小则会点过多，不利于小车移动的情况；最后则是路径优化函数的问题，两个优化函数，如果顺序颠倒，路径效果会受较大影响，确定先后顺序后，接着调整 near 函数中寻找近点的数量，最终得到效果较好的 RRT。

同时在调试 DWA 的过程中，heading_evaluation()一开始设定的是与目标方向的角度偏差，在 RRT+DWA 静态避障中也有较好的效果，但是在动态避障中，RRT 的路径可能会受到阻碍，因此我们采用了只用 DWA 的方式往返，而 heading_evaluation()的 cost 也改为了与目标点的欧式距离，这样效果比较好。

而在 dist_evaluation()的调试过程中，将离障碍物的距离转化成对应的耗散 cost 的函数是一大挑战。结合资料我们采用了五次项的反比例函数。而太小则小车无法通过较窄的通道；太大则小车容易刹不住车与障碍物发生碰撞。因此经历了很多调试。

而在确定各项 cost 前的系数和避障主函数 velocity_planning()中，由于最大角加速度和最大速度不能取太大，太大则小车轨迹可控性会变得很差，因此我们经过很久的调试，最终选择最大加速度为 1100，角加速度为 3。

五、心得与成员贡献：

我们作为代码能力比较弱的人，通过本次大作业进一步加深了 python 代码的理解，懂得了从原理到代码的转换过程，对于 RRT 和 DWA 算法的理解更加深刻，锻炼了自己的 debug 能力和代码能力，同时学会了如何根据自己的要求转换为实际的代码的过程。

成员贡献：

潘力豪：负责分工，DWA 避障代码的编写与调试，编写实验报告

王威：负责 RRT*算法的调试与编写，编写实验报告

杨安桐：负责代码的整合与实验报告的编写