机器人导论：RRT 和双向 RRT 算法

学号：18364066　姓名：卢彦作

一、RRT 和双向 RRT 算法概述

1. RRT 算法概述

RRT 算法又称快速扩展随机树算法。与 PRM 算法不同点在于，PRM 算法在一开始就通过抽样在地图上构建出完整的无向图，再进行图搜索，而 RRT 算法则是从某个点出发一边探索，一边抽样并建图。而与 PRM 算法相同点在于，RRT 算法也是概率完备的，只要路径存在，且规划的时间足够长，就一定能确保找到一条路径解，但如果规划器的参数设计不够合理，例如采样上限过低、步长过小等，就可能找不到解。

RRT 算法的运行大致可以分为四个部分：初始化、随机采样、扩展和终止条件。初始化部分需要我们将事先构建好的地图读取进来，对图进行灰度处理和二值化处理，目的是使地图矩阵只包含两种元素，一种是障碍物，另一种则是非障碍物。随机采样部分需要我们在每次采样时，有一定的概率向着目标点延伸，也有一定的概率在地图上随机选择一个方向延伸一段距离。扩展部分顾名思义就是对树进行扩展，最重要的就是做碰撞检测，找出当前树距离目标最近的顶点后，根据步长走一段距离就要判定路线内是否有障碍物阻挡。终止条件就是当我们对树扩展后，若当前点到终点的距离已经足够小，就可以终止算法的运行并输出。

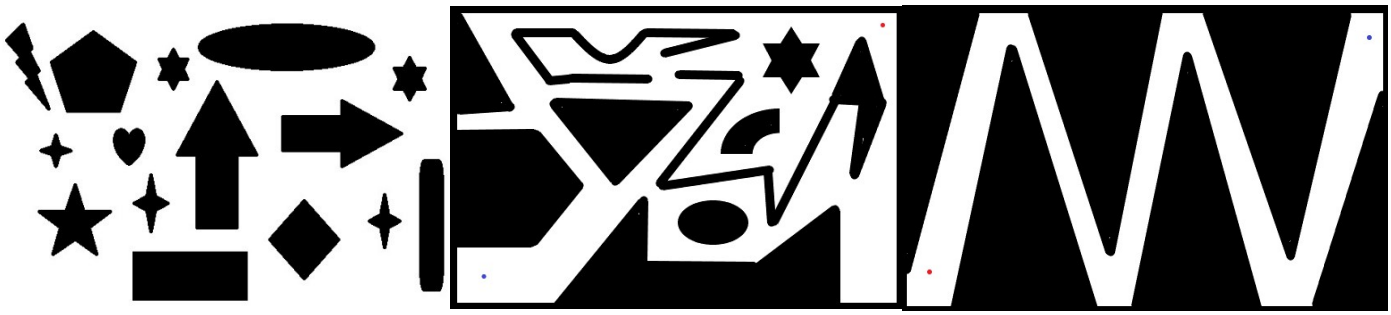
2. 双向 RRT 算法概述

双向 RRT 算法则是在 RRT 算法的基础上，为了解决 RRT 难以在狭窄通道环境找到路径的问题而提出的一种改进方案，具体做法其实十分简单，我们不仅在起始点开始建一棵树并对其进行扩展，同时也对终点建一棵树和起点树一起扩展，二者相互交替，哪棵树节点更少就优先对哪棵树进行扩展。

而当我们对一棵树例如起点树扩展了一个随机采样点后，我们立马对终点树开始扩展，并且扩展的目标正是起点树最新扩展的点，不仅如此，对终点树的扩展是不受一步限制的，需要一直朝着同一个方向进行扩展直到碰撞发生或者两者交合，这大大提高了算法的运行效率。

二、场景地图构建

本次实验中使用的场景地图构建方式比较简单，就是利用画图软件在空白画布上任意地放置一些几何图形，结合丁泓同学提供的地图共三张地图如下所示，其中第一张是我自己构建的，后两张是丁泓同学附件中的，尤其是第三张的狭窄通道能够帮助我们看出 RRT 和双向 RRT 两个算法的差距。

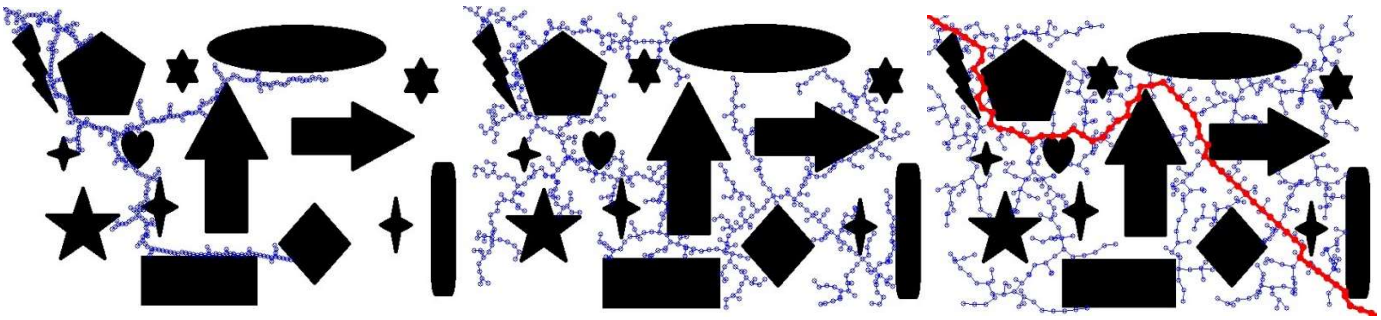


三、实验结果

本次实验代码全部由自己实现，没有参考或抄袭任何其他教程，代码已开源在 GitHub 仓库（链接在报告末尾），具体分为地图 Map、A 星算法 AStar、RRT 算法、双向 RRT 算法和主函数 main 五个部分，由于实验不要求代码讲解，这里不再赘述，下面展示不同参数对 RRT 算法和双向 RRT 算法的影响以及两个算法的对比。

1. RRT 算法的步长 pixel_step

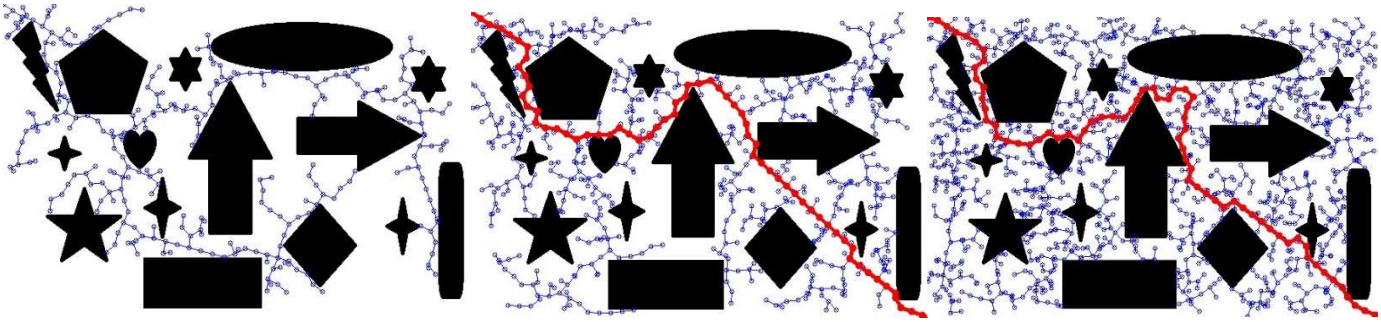
下面三张图的步长分别设置为 5、10、15 三档，随机概率和采样上限均为 0.5 和 3000，步长意味着每次沿着采样点方向前进的像素距离，可以看到随着步长的上升，地图被采样点覆盖的范围随着步长的提高是明显上升的，覆盖率的上升也意味着我们更容易在地图中发现终点，如图所示也只有步长设置为 15 时才能搜索出结果路径（加粗路线为 A*算法的搜索结果）。



（注:算法本身具有随机性，每次运行的结果都大不相同，报告只能选出那些比较具有代表性的展示出来，随机性的存在其实让很多种情况都有机会求出解）

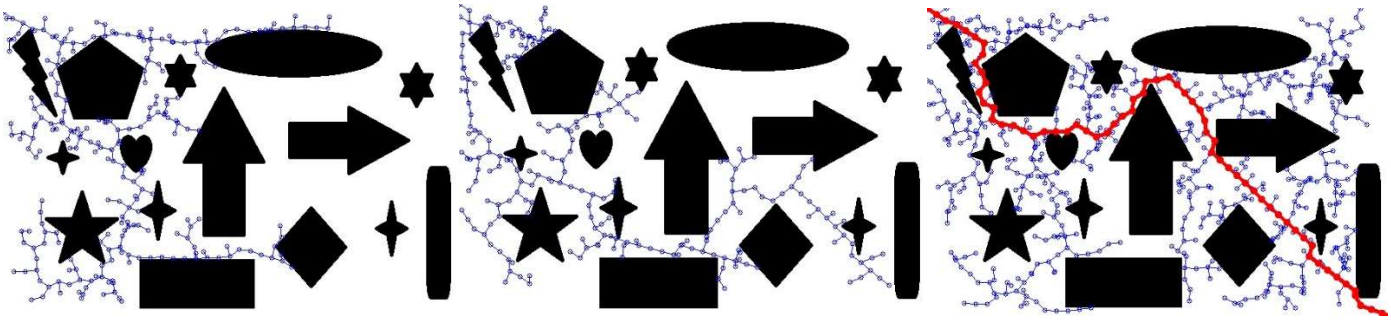
2. RRT 算法的随机概率

下面三张图的随机概率分别设置为 0.2、0.5、0.8 三档，步长和采样上限均为 15 和 3000，随机概率意味着每次采样在地图中随机采取点的概率。同样地可以看到，随着随机概率的上升，采样点在地图中的覆盖率也是逐步上升，但从理论上来说并不是越高越好，因为这建立在采样上限足够高的情况下才能成立，若采样上限降低一些，随着随机概率的上升很有可能出现无法求解的情况。



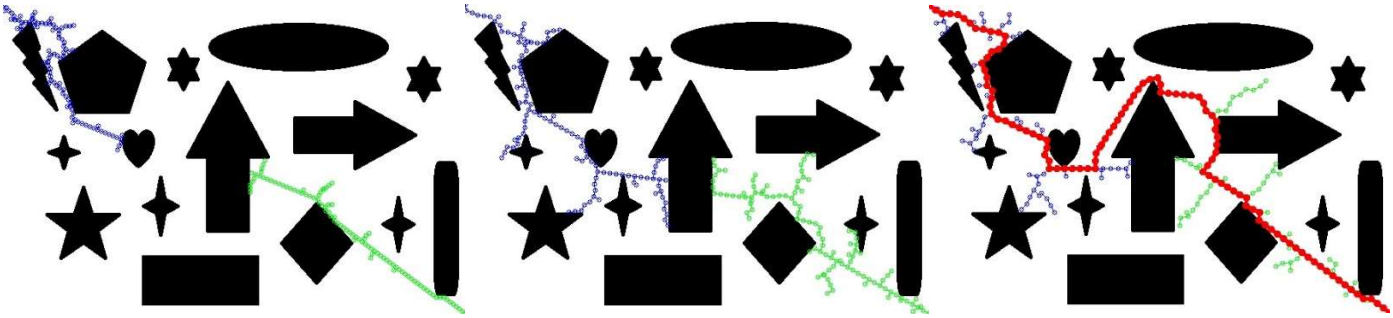
3. RRT 算法的采样上限

下面三张图的采样上限分别设置为 1000、2000、3000 三档，步长和随机概率均为 15 和 0.5，采样上限意味着最高的采样次数，若超过上限则算法自动终止。从图中可以看到，采样上限对算法的影响是比较大的，因为在相同随机概率下，采样上限越高，算法找到终点的概率也就越大。



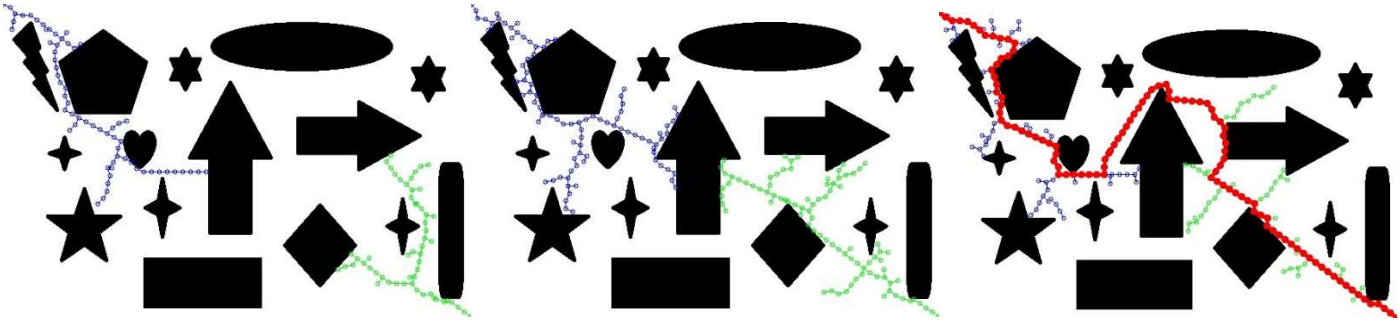
4. BiRRT 算法的步长

下面三张图的步长分别为 5、8、10 三档，BiRRT 算法中不存在随机概率的问题，每个采样点都是随机选取的，采样上限均为 800，原理同 RRT 算法相同，但 BiRRT 算法性能更强，需要的步长也更小。



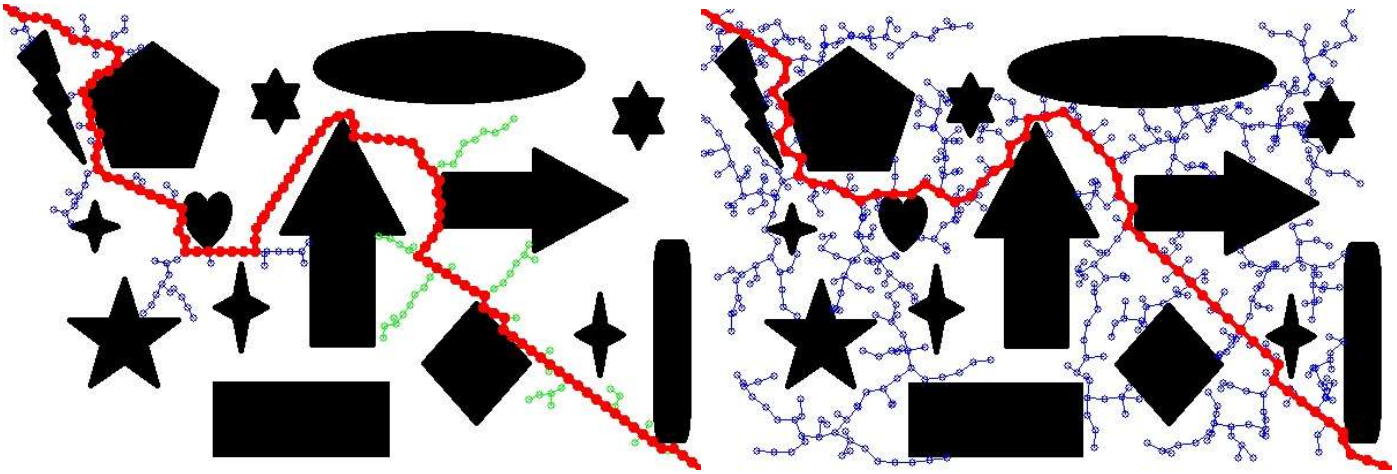
5. BiRRT 算法的采样上限

下面三张图的采样上限分别为 300、500、800 三档，步长均为 10，原理也和 RRT 算法类似，但在 BiRRT 算法中要求的采样数量是更小的，只要 800 个采样点就基本能确保找到解。

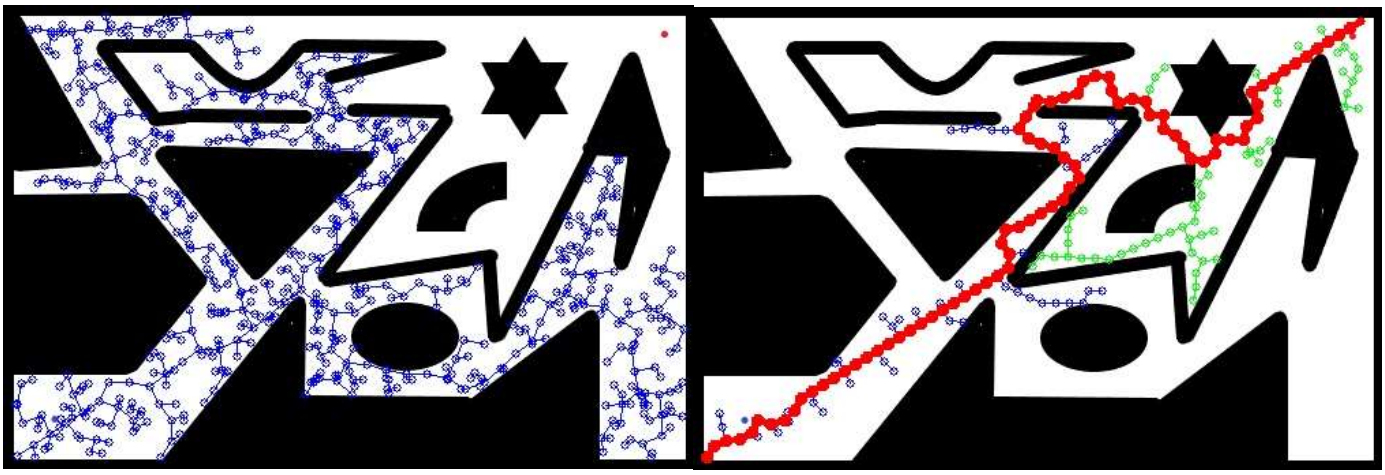


6. RRT 算法与 BiRRT 算法的对比

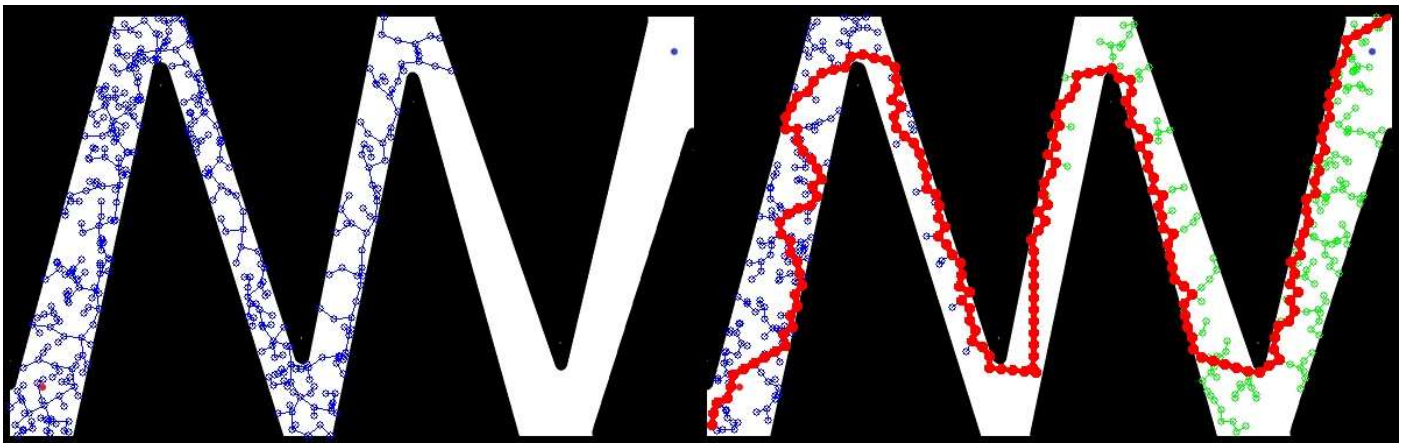
首先是第一张地图两个算法的比较（左 RRT，右 BiRRT）：



然后是第二张地图两个算法的比较：



最后是第三张地图两个算法的比较：



以上第一张和第二张地图的运行都是沿用此前的参数，RRT 步长 15、随机概率 0.5 和采样上限 3000，BiRRT 步长 10 和采样上限 800，可以非常明显地看出，即使 BiRRT 参数比 RRT 更加严格，但是表现依然更加优秀，甚至在第二张地图上 RRT 算法经常不能搜索出结果，BiRRT 算法在地图上覆盖率虽然比较小，但是它能够非常精准快速地求出解，并且更加逼近最佳路线。

而在第三张地图上就更有意思了，我将它们的采样上限提高到了相同的水平 4000，在这样的条件

下可以更容易地看出 BiRRT 算法的优势，在狭长的通道里，在两端同时向中间点搜索的效率是更加高的。

四、总结

总得来说，本次实验的收获是比较丰富的。一方面我对路径规划算法 RRT 和双向 RRT 进行了更深入的探讨和实现，不仅仅是停留在理论上，而是将算法应用到了具体的问题中，通过自己亲手构建地图，再自己实验一个算法去解决自己提出的问题，这整个过程具有挑战性的同时也给我带来了趣味感和成就感。另一方面，在编写和实现这些算法的时候，总会遇到大大小小的 bug 需要自己去发现和解决，比如说起初我使用 opencv 在地图上画线和画点时发现，搜索结果和打印结果完全对不上号，后来发现时横纵坐标 x 和 y 出现错位，imread 读取到的矩阵相应维度的意义和使用 cv2.line 和 cv2.circle 这两个绘画函数维度的意义是刚好相反的，所以在这个过程中我也总结了相关的一些小 tips 和经验。

回到算法本身的话，从实验结果上来看，两个算法的差异还是比较明显的，甚至出现了 BiRRT 参数比 RRT 参数更严格，表现却更好的情况，从单端出发的搜索到双端出发的搜索，这个改动的思想虽然很早就已经提出但还是极为有效的。

五、代码

详见 https://github.com/Yanzuo1019/SYSU-ROBOT（节约纸张，不再重复贴出）

打印报告若图不清晰也可在 repo 中查阅。