毕业论文 草稿

机器人发展综述

国内外情况

行业情况

Tesla Autopilot

Road star.ai

引言 DARPA Urban Challenge

整体规划综述 可以强调一下 Autopilot

着重 Planning

讲讲各种算法

细讲 Sampling-based algorithm

介绍 RRT 算法及 RRT variant

算法：

数据：

轨迹规划综述

随着软硬件质量与水平的提高，传感器与通信技术的发展，地面站与机载软件的质量都有了显著的提高，现有的无人机大多是遥控与跟踪预定的轨迹；但是高度自动化的无人机应该根据不同的环境制定相应的飞行轨迹，并且这些轨迹能够适应环境与机体自身动力学的约束，保证无人机能够顺利的按照轨迹飞行。

数学家与计算机学家等研究人员在经历了多年的实践与研究之后，诞生了很多轨迹生成的技术，并且迅速产生很多的衍生算法；如何将物体在不触碰到障碍物的前提下，从初始状态移动到目标状态成为了机器人轨迹规划的最基本问题之一。从 Lozano-Pérez [xx] 在 1979 年引入空间规划（spatial planning）这一方法开

始，不同的运动规划问题就转变为了在位形空间中寻找无碰撞路径的问题，并且使用了统一的数学方法进行了求解。之后，xxx [xx] 证明了运动规划问题是 NP 完全问题；现有的成熟的方法大多是一些经典算法的衍生版本，如路径图法（Roadmap），栅格法（Cell Decomposition），势场法（Potential Field）以及精确的数学规划法(Mathematical Programming)。理论上来讲，这些经典算法不是一定互斥的，在开发路径规划器的时候常常将其中一些结合在一起。

在路径图法中我们在可行域中抽取可行点集合将其组成一个可行点网络或路径，并且搜索路径就局限于该网络中。利用路径图法我们可以将路径规划问题简化为图论搜索问题。比较有名的路径图法是可视图法（VG），Voronoi 图法，Sihouette 和次目标（Subgoal）网络。其中可视图法在原理上是将凸障碍物的顶点作为特征点，并且将各个障碍物之间的特征点互相链接起来构成可视图；Voronoi 图法将空间切分成各个部分，其中每个部分包含靠近特定物体的点。

在栅格法中，我们将可行域分解为简单格子集合并且计算出每个格子的邻居格子关系。从初始域到终止域的非碰撞路线将由互相连接的栅格网络计算出来，其实也可以简化为图论搜索的方法，使用栅格法的想法在 [xxx] 中实现了出来。

势场法首先由 Oussama Khatib [xxx] 提出来，收到重力势场的启发，整个域可以整体建模为势场的互相叠加。整体势场是吸引势场域排斥势场的总和，其中吸引势场是目标域对机器人的吸引函数，排斥势场是障碍物对机器人的排斥函数。利用对域的整体建模生成了机器人由初始域向目标域前进的路径。

路径规划的启发式方法也有很多种，比如概率路径图法、快速搜索随机树法、模拟退火方法、蚁群算法和粒子群算法等。启发式算法不能保证能求到解，但是如果他们能求解路径的话，速度将是经典算法的无法比拟的。

我们还可以将启发式方法分为基于采样的方法和遗传算法；在基于采样的路径规划算法中，例如快速搜索随机树，没有必要对整个域进行建模，其在可行域中进行随机采样，并且将信息存入数据结构中，当目标域被采样得到之后，我们可以在已存储的数据结构中进行路径的搜索从而得到一条可行路径。基于采样的方法是概率完备但不是最优的方法，当然在不断的改进下还是得到了很大的进步。

以蚁群算法为代表的遗传算法也是最近几年科研人员研究与改进的对象。第一个将蚁群算法应用在路径规划的例子在 [xxx] 中给出。蚁群算法模拟了蚁群的一些特性，将整个域进行了路径图建模，并且在始末点初始化两个蚁群，分别

派出蚂蚁向始末点前进，并且留下信息素（信息素浓度会随着时间下降）；后面的蚂蚁根据自己周围信息素的浓度进行一定概率的选择；结果将在蚂蚁相遇或者分别找到目标点之后显示出来。实验结果表明遗传算法的结果能减少中间域的数量，当然是在可接受时间范围内。

最近提出的一类用来解决路径规划的算法在实施起来十分的成功。基于采样的方法拥有概率的特性，具有概率以及求解完备性，该论文采用了基于采样的方法进行了无人机的路径规划任务。

基于采样的方法

Xxxxxx 一些综述？

快速搜索随机树（Rapidly-exploring Random Tree）是基于采样的算法，在算法 x 中，本文提出了简单快速搜索随机树（sRRT）的计算过程。RRT 算法需要返回一颗由初始域q_{init}到目标域q_{goal}的搜索树。新的节点q_{rand}是在域中随机生成的，节点q_{nearst}是在搜索树中距离q_{rand}最近的节点，当下一个新节点将被添加到搜索树中时，q_{nearst}将被选择为父节点，最终新的节点q_{new}将按照一定方式生成：

$$q_{new} = q_{nearst} + \delta * \frac{q_{rand} - q_{nearst}}{||q_{rand} - q_{nearst}||} \tag{1}$$

其中 δ 是固定的步长，$||q_{rand} - q_{nearst}||$ 表示q_{rand}和q_{nearst}之间的距离，表示$||q_{rand} - q_{nearst}||$的方法有很多，一般使用欧式距离表示两点之间的距离。在一定迭代次数限制内，如果 sRRT 找到了路径会返回搜索树结果，如果在一定时间限制之内没有找到则会返回失败。

为了形象地表示 sRRT 的规划过程，本文在图 x 中展示了具体的 sRRT 的生长过程与寻路过程。图(a)展示了整个域（Configuration Space）的信息，其中黄色的点为初始域，绿色为目标域，灰色的为障碍物域，我们的目标是利用 sRRT 算法规划出一条非碰撞路径，使得机器人能顺利从初始域到达目标域；初始化时将初始点q_{init}加入搜索树 T 中；图(b)表示的是在 sRRT 算法循环中的第一步，以固定概率生成随机点q_{rand}；图(c)表示从搜索树 T 中找到距离q_{init}最近的点q_{nearst}，一般的衡量距离都是欧式距离；图(d)表示如果q_{nearst}和q_{new}组成的边在可行域中

的话，那么就将新生成的点加入到搜索树中，并且将新边也加入到搜索树的数据结构中；

图(e)～(f)表示了循环生成q_{new}的过程；特殊的，(g)～(h)图表示了当生成q_{new}之后$Edge(q_{nearst}, q_{new})$不在可行域中，而是被障碍物域阻隔，这种情况不符合路径要求，所以要舍弃，在算法 x 的 11 行中体现了这个特点；在重复算法 x 的 3～21 步之后，我们将得到搜索树的结果，如图(i)所示；

直观的，我们可以将路径从搜索树中抽象出来，规划出一条非碰撞路径使得机器人能顺利从初始域到达目标域，抽象出来的路径如图 x 所示；本文提出了一种减枝算法使得能从搜索树 T 中得到从初始域到目标域的唯一一条路径，减枝算法步骤见算法 x。

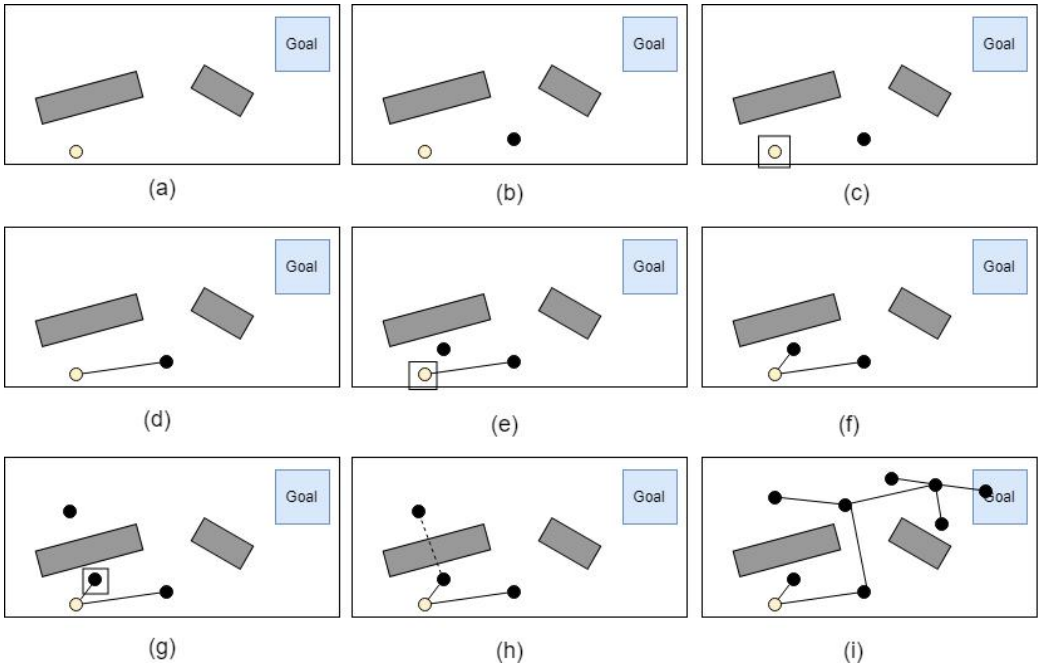

图 x.x sRRT 算法规划过程

算法：简单快速搜索随机树（Simple Rapidly-exploring Random Tree）

数据：T = 搜索树；初始域q_{init}；目标域q_{goal}

T.add(q_{init});
p = RandomProbability( );
for i = 1→k
以 p 概率将q_{goal}作为q_{rand}，以 1 - p 的概率在域中随机生成q_{rand} ；
计算得到搜索树 T 中距离q_{rand}距离最近的q_{nearst}；
计算$q_{new} = GenerateQNew(q_{rand}, q_{nearst}, \delta)$；
if q_{new} 不在域中
Continue；
End if
if q_{new}不在障碍物域中
if $Edge(q_{nearst}, q_{new})$在可行域中
Vertices ← q_{new}；
Edges ← $Edge(q_{nearst}, q_{new})$；
if q_{goal}在搜索树范围内
Vertices ← q_{goal}；
Edges ← $Edge(q_{touched}, q_{goal})$；
Return T
End if
End if
End if
End for
Return failure

算法 x xxx

算法：sRRT 减枝算法

数据：搜索树 = T；目标域 = q_{goal}

FirstVertices ← 数据结构 Edge 中的第一个顶点信息；

2 SecondVertices ← 数据结构 Edge 中的第二个顶点信息；

3 LeafVertices ← $FindLeafVertiecs(FirstVertices, SecondVertices)$；

4 将q_{goal}从 LeafVertices 中删除，保留主路径；

5 **while** LeafVertices 非空

6 更新 Vertices，从 Vertices 中删除 LeafVertices；

7 更新 Edge，从 Edge 中删除叶子边；

8 LeafVertices ← $FindLeafVertiecs(FirstVertices, SecondVertices)$；

9 **End while**

10 **Return** T

算法 x xxx

算法：RRT 路径抽象（skeleton）算法

数据：减枝后的搜索树 = T'

1 index ← 1;

2 AbstractedVertices ← Vertices(index);

3 **For** i = 2 → VerticesQuantity(T')

4 **If** $Edge(q_{index}, q_i)$ 与障碍物域冲突

5 index ← i - 1;

6 AbstractedVertices ← AbstractedVertices ∪ Vertices(index);

7 **End If**

8 **End For**

9 利用 AbstractedVertices 构建一条抽象出来的路径 P；

10 **Return** P;