基于 MB-RRT* 的无人机多点航迹规划算法研究

陈晋音 胡可科 李玉玮
(浙江工业大学信息工程学院 杭州 310023)

摘 要 随着小型无人机的广泛应用,无人机的自动巡航能力至关重要。多点航迹规划作为复杂的无人机航行任务之一,要求为无人机规划出一条最优航迹或次优航迹,如距离最短、速度最快或者时间最短,并保证其在不碰撞已知障碍物的条件下遍历所有特定的航点。针对无序的多点航迹规划问题,基于 MB-RRT* 算法并结合原本用于解决 TSP 问题的贪心策略提出了贪心 MB-RRT* 算法,其通过牺牲一定的航迹质量,来提高解决无人机多点航迹规划问题的速度,减少时间代价。最后在二维地图环境和三维环境下进行实验,验证了所提算法的可行性和有效性。

关键词 贪心算法,RRT,无人机,多点航迹规划,收敛速度
中图法分类号 TP242 **文献标识码** A

Research on UAV Multi-point Navigation Algorithm Based on MB-RRT*

CHEN Jin-yin HU Ke-ke LI Yu-wei
(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract With the wider application of UAV, automatic navigation capability of UAV plays a more important role. As one of the complex UAV missions, multi-point navigation algorithm requires an optimal path or sub-optimal path, such as the fastest, shortest distance or shortest time to traverse all specific destinations without collision with known obstructions. Aiming at the problem of disordered multi-point path planning, the greedy MB-RRT* algorithm was proposed which is based on MB-RRT* and combined with greedy strategy used to solve the TSP problem. The algorithm improves the speed of the multi-point navigation problem by sacrificing a certain path quality. Finally, the effectiveness of the algorithm was verified by simulation experiments in the two-dimensional and tree-dimensional environment.

Keywords Greedy algorithm, RRT, UAV, Multi-point navigation, Convergence rate

1 引言

无人机(Unmanned Aerial Vehicles, UAVs)作为一类自主飞行器,能够携带传感器、通信设备或者其他与任务相关的负载。随着相关技术的不断发展,无人机在现实世界中的应用越来越广泛[1]。其中,许多应用场合需要无人机在室内/室外能够进行自主导航飞行,并且能够在飞行过程中规避障碍以完成相应任务。航迹规划技术是实现无人机导航的关键技术之一,其可以细分为轨迹规划(Trajectory Planning)和路径规划(Path Planning),区别在于前者获得的航迹包含无人机的动力学描述,而后者不包含。作为自主机器人除了实时定位和导航以外的另一关键技术,航迹规划一直是研究热点,它用于在起点到其他任意点之间构建无碰撞路线。

无人机技术的发展与探索毫无疑问能够不断提升无人机本身的经济价值,并且复杂任务的实现能够更加满足当今社会对无人机应用的要求。因此,多点航迹规划对于无人机应用领域的拓宽有着十分巨大的帮助,例如在执行大面积区域内或者危险环境下的巡逻勘探以及多地补给货物运输等任务时,无人机多点航迹规划能够节省许多的人力和财力,并且使无人机更加高效与智能[2]。

从起点到终点之间简单的两点间航迹规划技术已经十分成熟,目前最主要的两种采样方法,即概率路线图算法(PRM)[3]和快速扩展随机数算法(RRT)[4],已被证明可以高效地解决航迹规划问题。后续针对这两种算法进行研究并提出的变种算法,如 MB-RRT* 算法等,使得在解决两点间航迹规划问题时的运行效率和结果质量有了极大的提高,同时也被逐渐应用于解决实际问题。

多点航迹规划作为复杂的无人机航行任务之一,因具有巨大的应用前景而受到了许多关注。它要求为无人机规划出一条最优航迹或次优航迹,如距离最短、速度最快或者时间最短,并保证其在不碰撞已知障碍物的条件下遍历所有特定的航点[5]。根据无人机访问航点的顺序是否已经事先给定,多点航迹规划可以分为定序和无序两种情况。定序,即无人机访问所有航点的顺序已经人为确定,只需要为无人机规划出从任意航点出发至其下一航迹点的可行航迹,并最终合并成一条遍历航迹即可。无序,则表示无人机访问所有航点的顺序并未确定,因此需要结合各航点之间可行航迹的长度来确定访问顺序,最后规划出一条最优或者次优的航迹。这决定了多点航迹规划问题的复杂性,其中无序的多点航迹规划尤为复杂。

目前,大多数航迹规划算法(例如上述几类算法)只解决了两点间航迹的生成问题,而不能满足复杂的无序多航迹点规划任务的需求。虽然在解决定序的多航迹点规划问题时,能够采用栅格地图建模下的可视图法(VG)[6]、概率地图法、

陈晋音(1982—),女,副教授,硕士生导师,主要研究方向为算法设计和分析、数据挖掘、机器学习等,E-mail:chenjinyin@zjut.edu.cn(通信作者);胡可科(1995—),男,硕士生,主要研究方向为算法设计和数据挖掘;李玉玮(1995—),男,硕士生,主要研究方向为算法优化与机器学习。

快速扩展随机树法、空间骨架化法(SS)或者拓扑地图建模下的逻辑时序语言法(LTL)[7],但是在面对无序的多航迹点规划问题时还需要增加为航点确定访问顺序的步骤。如果不先确定访问顺序,Trevor 等提出的基于基因算法的多点航迹规划算法虽能完成多航迹点任务,但由于基因算法的复杂性和计算任务的繁重性,其并不能很好地适用于高维环境。

因此,本文将结合原本用于解决 TSP 问题的贪心策略提出贪心 MB-RRT* 算法,用于无序的多点航迹规划任务,并在二维地图环境下通过实验验证其性能,进而扩展至三维环境下进行仿真实验来验证其有效性。

2 相关研究

2.1 TSP 动态规划算法

TSP 问题(Travelling Salesman Problem)[8],又称为旅行商问题、货郎担问题,其字面意思是:有一个推销员要到 n 个城市推销商品,他要找出一个包含所有 n 个城市的具有最短路程的环路。实际上,TSP 是一个多局部最优的优化组合问题。

动态规划(Dynamic Programming,DP)是运筹学的一个分支,是求解多阶段决策过程最优化的数学方法。1951 年,美国数学家 Bellman 等根据一类多阶段决策问题的特性,将多阶段过程转化为一系列单阶段问题并逐个求解,提出了解决这类问题的"最优性原理",同时研究了许多实际问题,从而创建了最优化问题的一种新方法——动态规划。

动态规划是一种将复杂的问题分解为更小的、相似的子问题,以解决最优化问题的算法策略。动态规划是以发展的观点来处理最优化问题的一种方法,它将问题按照发展过程的先后顺序分成若干阶段,并对每个阶段做出最优决策。

算法的迭代流程如算法 1 所示。

算法 1 $DP(S,x_k)$

```
θmin ← ∞
if C(S,xk) ≠ −1 then
    return C(S,xk)
end if
if S = Ø then
    return dko
end if
while i < n do
    if xi ∈ S then
        θ′ = DP(S−{xi},xi)+dki
        if θ′ < θmin then
            C(S,xk) = θmin = θ′
        end if
    end if
end while
return C(S,xk)
```

记 $k\in\{0,1,2,3,\cdots,n\}$,初次迭代时 x_k 为起点 x_o;S 为集合$\{x_0,x_1,x_2,x_3,\cdots,x_n\}$的子集,初次迭代时 S 为包含除 x_o 之外的所有元素的集合;d_{ij} 为节点 x_i 到 x_j 的距离;$C(S,k)$为从 x_k 出发遍历 S 中的节点并且终止于节点 x_o 的最短距离,默认初始值为 -1,其中 $k\notin S$。当集合 S 的个数为 0 时,$C(S,k)=d_{ko}$;当集合 S 的个数大于 0 时,根据最优性原理,可将 TSP 动态规划方程写成:

$$C(S,k)=\min_{i\in S,k\notin S}\{d_{ki}+C(S-\{x_i\},i)\}$$

通过动态规划算法可以得到该问题的最优解,但该算法是一个递归算法,它的时间复杂度为 $O(n^2 2^n)$,空间复杂度为 $O(n2^n)$。随着问题规模的扩大,其所需要的空间会急剧增加,故一般除了很小规模的问题外,几乎不采用该算法。

2.2 贪心算法

贪心算法[9],又称贪婪算法(Greedy Algorithm),可以有效地解决 TSP 动态规划算法复杂度较大的问题,但从结果上来说,得到的只是近似最优解,而非最优解。贪心算法在求解问题时采取了一种算法策略,总是做出当前情况下的最好选择,因此每次选择得到的都是局部最优解,如此反复,直到得到完整的解。选择的策略必须具备无后效应,即某个状态以前的过程不会影响以后的状态,其只与当前状态有关。

该算法的主要流程图如算法 2 所示。

算法 2 $GA(S,x_k)$

```
S ← {x0,x0,x0,…,xn} − {xo};
θ ← 0
while S ∉ Ø do
    xk ← NearstNode(xo,S)
    θ = θ + dok
    S = S − {xk}
    xo = xk
end while
return θ
```

NearstNode:给定一个节点 x_o 以及未被访问到的节点集合 S,该函数返回在集合 S 中与节点 x_o 距离最近的节点 x_k,k 为其下标。

贪心算法的时间复杂度为 $O(n^2)$,在节点数较多的情况下远小于 TSP 动态规划算法的复杂度,因此通过适当增加结果误差来提高运算效率的贪心算法被更多地应用于实际。

3 贪心 MB-RRT* 算法

3.1 主要思想

MB-RRT* 算法虽然能够提升两点间航迹规划的性能,但是仍不具备确定访问顺序的能力,并且如果简单利用解决 TSP 问题的算法来确定航点的具体访问顺序,不管是动态规划算法还是贪心算法,或者其他方法,都需要事先得到各航点间可行航迹的长度。那么,在有障碍物的环境下,势必首先需要利用路径规划算法来获得各航点间的可行航迹,以生成一张各航点的全连通图,最后才能利用 TSP 算法确定访问顺序。

这种思路进行操作的代价主要有:1)生成全连通图所需要的时间在复杂环境中将非常巨大;2)TSP 完全算法所需要的时间在航点数量增加之后将以指数级速度增长,而如果使用 TSP 近似算法,虽然确定航点访问顺序的时间将大大减少,但是获得的顺序并不是最佳的,这使得之前用于获得全连通图的巨大时间花费十分不划算。

针对这两个问题,本文提出了一种贪心 MB-RRT* 算法,即将 MB-RRT* 算法与贪心策略结合,通过牺牲一定的航迹质量,来提高解决无人机多点航迹规划问题的速度,降低时间代价。

3.2 算法步骤

本文提出的贪心 MB-RRT* 算法流程图如图 1 所示。

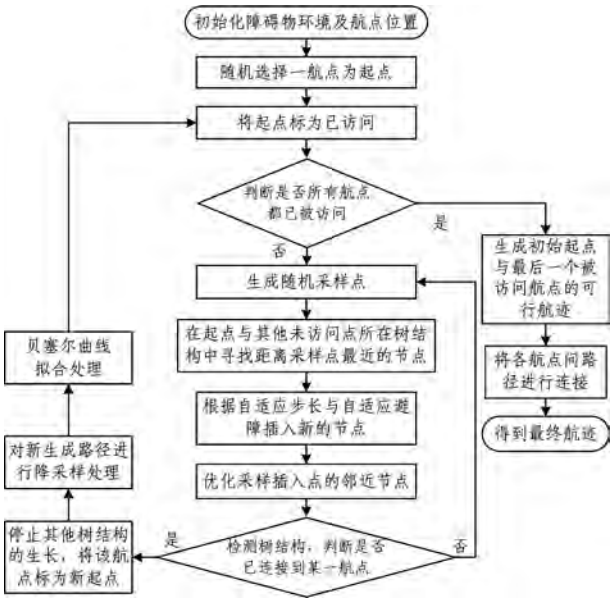


图 1　贪心 MB-RRT* 算法流程图

从流程图中可以看出，利用贪心 MB-RRT* 算法进行多点航迹规划时，需要输入环境空间及其障碍物的位置信息，如坐标、尺寸、起始航点与各个需要访问航点的位置等，经过规划，最终输出一条从起始航点出发并访问了所有目标航点而与障碍物无碰撞的航迹，该航迹由各点坐标表示。

规划过程中首先进行栅格化处理，即将环境空间均匀分解成栅格，栅格尺寸及权值可以预先设定，权值表示该栅格是否存在障碍物。然后将所有航点组成航点列表 $L_{\mathrm{All}}=\{X_i, i=1,2,\cdots,n\}$，并将起始航点（标记为 X_{Start}）放入已访问航点列表 $L_{\mathrm{Visted}}=\{X_i, i=1,2,\cdots,m\}$ 中。接着生成 $2*(n-m)$ 个随机采样点 $X_{\mathrm{Rand}}^i(i=1,2,\cdots,2n-2m)$，并为未访问航点列表 $L_{\mathrm{UnVisted}}(L_{\mathrm{UnVisted}}=L_{\mathrm{All}}-L_{\mathrm{Visted}})$ 中的所有航点（记为 $X_{\mathrm{Goal}}^i(i=1,2,\cdots,n-m)$）分配一个与各自对应的随机采样点，共 $(n-m)$ 个，同时为 X_{Start} 分配 $(n-m)$ 个随机采样点。通过调控随机采样点的采样概率，使得随机树既能够朝目标航点生长，又具有一定的随机性，以避免陷入局部最优。

接着，在起点 X_{Start} 与 X_{Goal}^i 各自所在树结构 $Tree_i(i=1,2,\cdots,2n-2m)$ 中寻找距离各自采样点最近的节点 $X_{\mathrm{Nearest}}^i(i=1,2,\cdots,2n-2m)$。之后，在 X_{Nearest}^i 到 X_{Rand}^i 的方向上进行判断，若这两点之间没有障碍物且两点距离小于最大搜索步长 l_{Max}，则连接这两点，并将 X_{Rand}^i 加入原 X_{Nearest}^i 所在树结构 $Tree_i$；否则首先在 X_{Nearest}^i 到 X_{Rand}^i 的方向上距离 X_{Nearest}^i 为 l_{Max} 的位置上生成一个点 X_{New}，判断 X_{New} 与 X_{Nearest}^i 之间是否有障碍物，若没有，则将 X_{New} 加入 $Tree_i$，否则重新计算新的步长 $l=\frac{D}{l_{\mathrm{Max}}}l_{\mathrm{Min}}$，其中 D 为 X_{Nearest}^i 至最近障碍物的距离，l_{Min} 与 l_{Max} 为预设的最小步长与最大步长。根据新步长 l 生成新的点 X_{New}，重新判断，直至 X_{New} 与 X_{Nearest}^i 之间没有障碍物后将 X_{New} 加入 $Tree_i$ 中，该步骤为自适应步长。为了避免 l 为 l_{Min} 时 X_{New} 与 X_{Nearest}^i 之间仍有障碍物而导致本次迭代无意义，在距离 X_{Nearest}^i 为 l_{Min} 的范围内随机选择一个无障碍物的栅格生成 X_{New} 以提高搜索效率，该步骤为自适应避障。经过自适应步长与自适应避障后，在 $2*(n-m)$ 条 $Tree_i$ 中各自插入一个新节点 X_{New}^i，此时需要对 X_{New}^i 的相邻节点 X_{Near} 进行优化，即在 $Tree_i$ 中寻找与 X_{New}^i 无障碍相连的新相连节点 X'_{Near}，使 X_{New}^i 到 X'_{Near} 的距离小于 X_{New}^i 到 X_{Near} 的距离。以上为树结构生长过程中的一次迭代。

接下来判断 X_{Start} 是否已连接到某一未访问航点，即判断当前 X_{New}^i 与目标航点所在树结构中是否有一点的距离小于当前步长且可以无障碍连接，若有，则说明这两点在同一树结构中，停止当前所有树结构的生长，将本次访问到的航点标为新起点，否则就继续进行当前树结构的生长。然后，对新生成的树结构即新一段航迹进行降采样处理，即在不与障碍物相碰的前提下对该树结构内的节点进行重新连接，舍弃冗余节点，使得路径最短；并采用贝塞尔拟合法对降采样后的航迹进行进一步优化，使得航迹更加光滑。以上为访问一个航点的过程。

将该航点标记为已访问后，判断所有航点是否都已被访问，如果没有，则继续访问其他航点，否则根据上述步骤生成最后一个被访问航点与起始航点的可行航迹。经过各航迹段的连接，得到最终巡回航迹。

为了更加形象地介绍此算法，本文设置了一个贪心 MB-RRT* 算法，在无障碍空间中进行多点航迹规划的小实验进行辅助说明。图 2 为该实验中算法运行过程的示意图，图 2(a)为一开始随意放置的 20 个需要访问的航点，选择左上角序号为 1 的点为起点去访问其他序号为 2－20 的 19 个点。图 2(b)到图 2(f)为算的法演变过程，从过程图可以看出，初始时起点向所有未访问航点方向进行双向树的生长，一旦达到其中一个航迹点，则停止其他树生长，并以该航迹点作为新起点向其他未访问点生长，直至访问到所有航点。

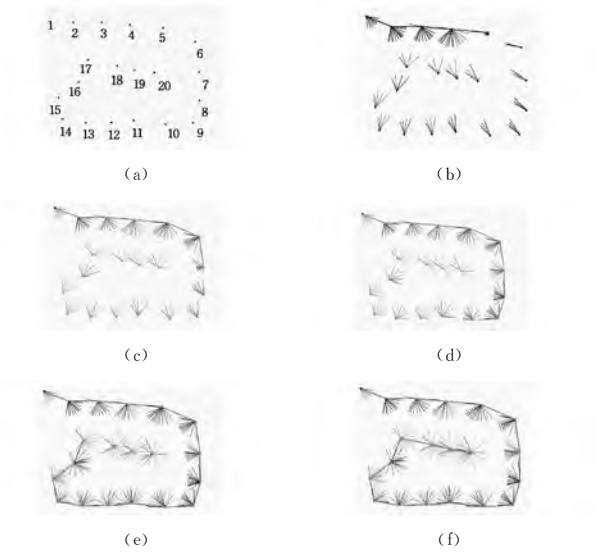


图 2　贪心 MB-RRT* 算法的二维运行过程图

4　仿真实验

本文将利用两种方法与本文所提的贪心 MB-RRT* 算法进行比较。首先介绍第一种方法，先利用 MB-RRT* 算法生成全连通图，再根据用于解决 TSP 问题的动态规划算法来得到访问所有航点的最佳顺序，最终生成一条巡航路径。该方法所得结果是最优解，可通过与次优解进行比较来评价次优解的质量。第二种方法是针对第一种方法中动态规划复杂度仍旧太高的缺陷，采用 TSP 问题中的贪心算法来减少降低确定访问顺序的时间，但所得结果为次优解。

由于动态规划算法的复杂性，点数过多时计算时间非常

长，导致无法为贪心 MB-RRT* 算法提供最优解进行对比实验。因此，在航点个数的选择上，本实验选用 5 个点来代表点数较少时的情况，选用 10 个点作为点数较多时的情况。

二维与三维环境下的航迹规划演示与优化均在 Ubuntu 系统下进行。其中，二维实验用 C++ 的跨平台开源应用程序框架 Qt 实现，并借助 Qt 的绘图功能对算法的运行过程进行展现；三维环境下的实验借助 ROS 系统进行，并利用其自带的可视化工具 rviz 来展示仿真结果。表 1 列出本文实验的硬件环境。

表 1 实验硬件环境

类型	参数
处理器	Intel(R) Core(TM) i5-3337U 1.80GHz
系统版本	Ubuntu 64-bit
内存	4GB

4.1 二维实验结果与分析

本节介绍动态规划算法、贪心算法和贪心 MB-RRT* 算法在不同类型的二维地图环境下的运行情况，采用的地图分别为自由空间地图、稀疏障碍物地图、密集障碍物地图以及带回廊型地图 4 种类型。地图尺寸为 600 * 800，栅格尺寸为 1 * 1，最大步长为 20，最小步长为 5。图 3－图 6 分别给出 3 种算法在 4 种地图上运行的结果，其中图(a)－图(f)分别为运行动态规划算法、贪心算法、贪心 MB-RRT* 算法后树的生长情况和路径结果。从树的生长情况来看，贪心 MB-RRT* 算法在求得可行解时所需的迭代次数要远少于动态规划算法和贪心算法。从路径结果来看，在自由空间地图和带回廊的地图上，3 种算法的路径质量相同；而在稀疏障碍地图和密集障碍物地图下，动态规划算法的路径质量最好，贪心算法其次，贪心 MB-RRT* 算法略差。以上说明，贪心 MB-RRT* 算法通过牺牲一定的路径质量来减少算法运行时间的处理方法是比较实际可行的。

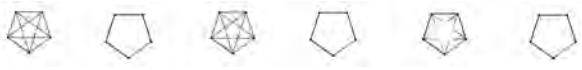


图 3 自由空间地图下的运行结果

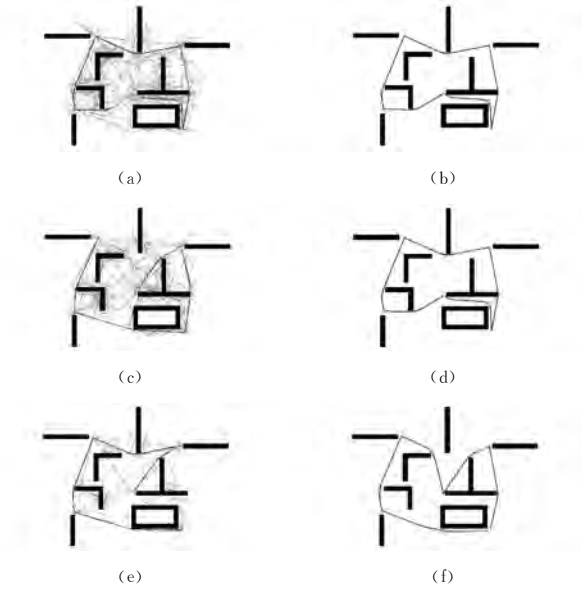


图 4 稀疏障碍物地图下的运行结果

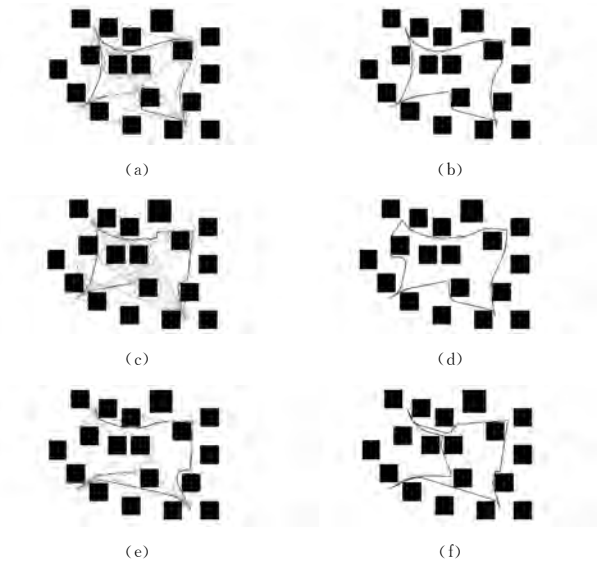


图 5 密集障碍物地图下的运行结果

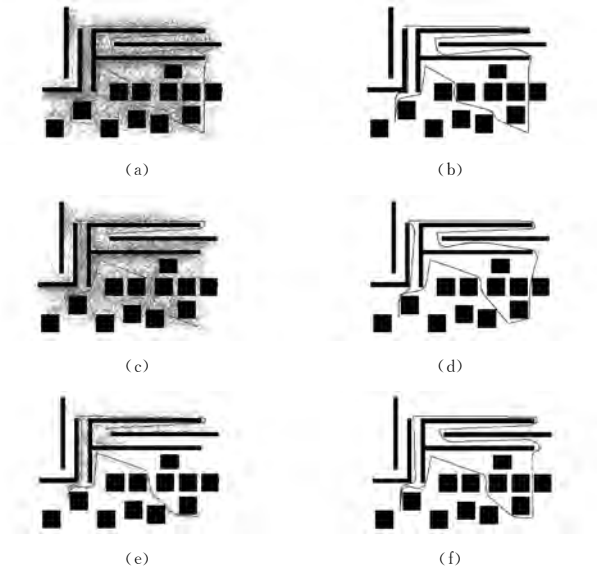


图 6 带回廊型地图下的运行结果

表 2 和表 3 列出 3 种算法在 4 类地图上运行的具体实验数据，表 2 为无序的 5 点航迹规划数据，表 3 为无序的 10 点航迹规划数据。其中，迭代次数与时间都为找到初始解后的数据，路径长度为降采样前的初始路径长度。由于 3 种算法都具有随机性，因此表中数据均为各自运行 10 次后所取的平均值。

由表 2 可以看出，为了获得最优解，将耗费较多的时间来生成全连通图并解决 TSP 问题。贪心算法虽能减少用于解决 TSP 问题的耗时，但是生成全连通图仍需要较多的时间，特别是在复杂环境中，全连通图的生成更是耗时惊人。贪心 MB-RRT* 算法虽然牺牲了一定的路径质量，导致路径长度比另外两种方法都要长，但是在复杂环境中寻找初始解时相较另外两种方法能节省非常多的时间，且路径长度也在可接受范围之内。结合图 3－图 6 可以看出，贪心 MB-RRT* 算法可以有效抑制冗余树结构的生长，减少一部分内存空间的占用，而且生成的路径也存在是最优解的可能性。表 3 是 10 点航迹规划时 3 种方法的比较，可以看出，随着点数的增加，另外两种方法用于生成全连通图的时间大大增加，而贪心 MB-RRT* 算法增加的耗时却相对很少，尤其是在复杂环境下尤为明显。

表 2　无序情况下二维 5 点航迹规划的比较

地图	算法	迭代数	路径长度	全连通时间/ms	TSP 解决时间/ms	总运行时间/ms
地图 1	动态规划法	5.0	585.8	36.4	620.0	656.4
	贪心算法	5.0	587.8	46.6	0.003	46.6
	贪心 MB-RRT* 算法	12	587.8	—	—	62.2
地图 2	动态规划法	203.0	1718.4	5114.5	1345.0	6459.5
	贪心算法	170.0	1846.6	5164.5	0.004	5164.5
	贪心 MB-RRT* 算法	102.1	1971.3	—	—	856.2
地图 3	动态规划法	127.0	2400.6	8724.7	823.4	9548.1
	贪心算法	127.0	2419.7	8617.5	0.004	8617.5
	贪心 MB-RRT* 算法	82.0	2505.1	—	—	1041.2
地图 4	动态规划法	548.5	3623.0	20320.0	639.0	20959.0
	贪心算法	548.3	3728.0	20290.9	0.004	20290.8
	贪心 MB-RRT* 算法	423.7	4103.1	—	—	16320.7

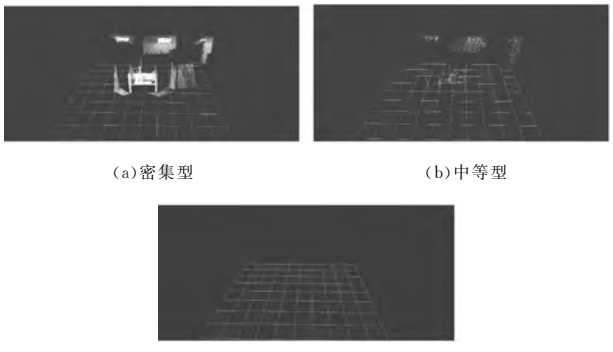
表 3　无序情况下二维 10 点航迹规划的比较

地图	算法	迭代数	路径长度	全连通时间/ms	TSP 解决时间/ms	总运行时间/ms
地图 1	动态规划法	5.0	585.8	36.4	620.0	656.4
	贪心算法	5.0	587.8	56.6	0.003	56.6
	贪心 MB-RRT* 算法	12	587.8	—	—	67.33
地图 2	动态规划法	203.0	1718.4	5114.5	1345.0	6459.5
	贪心算法	170.0	1846.6	5164.5	0.004	5164.5
	贪心 MB-RRT* 算法	102.1	1971.3	—	—	5435.2
地图 3	动态规划法	63.0	2400.6	8724.7	823.4	9548.1
	贪心算法	62.0	2419.7	2231.5	0.004	2231.5
	贪心 MB-RRT* 算法	153.0	2505.1	—	—	1921.2
地图 4	动态规划法	5109.5	4523.0	597512.0	639.0	598151.0
	贪心算法	5102.3	4728.0	6013.9	0.004	6013.9
	贪心 MB-RRT* 算法	1921.0	4917.6	—	—	4017.2

另外，在面对更多数量的航点时，利用动态规划算法获得最优解的时间将大大增加，11 点时需要 0.5 min，12 点时需要 3 min，13 点时需要 15 min，14 点时需要 90 min，而 20 点时计算量已经爆炸；同时，用于生成全连通图的时间也大大增加，环境复杂度的提高对其影响更甚。而贪心 MB-RRT* 算法在本文所设实验环境中的用时仍只需几秒钟，说明贪心 MB-RRT* 算法在解决无序多点航迹规划问题上占有相当大的时间优势。

4.2　三维实验结果与分析

本节介绍动态规划算法、贪心算法和贪心 MB-RRT* 算法在三维真实环境的点云地图上的运行情况，采用的地图为一幅办公室环境点云地图经过不同间距处理后获得的 3 幅实验地图，如图 7 所示。



(a)密集型　　(b)中等型

(c)稀疏型

图 7　不同密度的点云地图

其中图 7(a)为间距 0.01 m 采样后获得的密集型地图，共有 1504761 个障碍物点；图 7(b)为间距0.1 m采样后获得的中等型地图，共有 29793 个障碍物点；图 7(c)为间距 1 m 采样后获得的稀疏型地图，共有 667 个障碍物点。

为了展示贪心 MB-RRT* 算法在三维环境地图下的运行情况，图 8 给出该算法在密集型地图下巡回遍历 5 点生成的路径结果。由图中可以看出，贪心 MB-RRT* 算法在无序三维多点航迹规划任务的解决上具有有效性。

图 8　贪心 MB-RRT* 算法所得初始航迹图

对于三维环境下无序的多点航迹规划实验，本实验要求巡回遍历放置的 10 个航点，并与前文所提的动态规划算法以及贪心算法进行比较，得到表 4 所列结果，表中的数据均为每幅地图运行 10 次后的平均值。

本文将图 8 作为例子，对三维点云地图中的航迹进行展示。由于各算法仿真结果在形状上的差别仅在于航点访问顺序不同，而本文实验中各算法最终访问航点的顺序相同，因此仿真图相似。另外，各算法在其他性能上的区别也在表 4 中得到了体现，因此不再放置其他算法的仿真图。

从表 4 中的数据可得，贪心 MB-RRT* 算法的迭代次数和时间都明显小于其他两种方法，说明其在三维多点航迹规划中具有快速获得初始解的优良特性。另外，初始解的航迹长度虽然质量较差，与最优解相差较大，尤其是在较为稀疏的环境下，然而其在密集型环境中尚可接受，这说明贪心 MB-RRT* 算法在障碍物较为密集的环境中性能优势更大，更具有效性。

表 4 三维 10 点航迹规划比较实验

地图	算法	迭代次数	航迹长度	全连通时间/s	解决 TSP 时间/ms	总运行时间
稀疏型	动态规划法	280	1445.0	1.034	7240.0	8.274/s
	贪心算法	280	1789.5	1.068	0.267	1.068/s
	贪心 MB-RRT* 算法	71	1974.1	—	—	0.169/ms
中等型	动态规划法	350	1476.0	2.169	7300.2	9.469/s
	贪心算法	350	1628.8	1.596	0.315	1.596/s
	贪心 MB-RRT* 算法	80	1837.5	—	—	0.282/ms
密集型	动态规划法	500	1534.0	2.724	5989.0	8.718/s
	贪心算法	500	1574.5	3.395	0.188	3.395/s
	贪心 MB-RRT* 算法	107	1779.2	—	—	0.328/ms

结束语 本文通过将 MB-RRT* 算法与贪心策略结合设计了一种贪心 MB-RRT* 算法，用于解决无人机多点航迹规划问题，并利用二维实验证明了该算法在解决无人机多点航迹规划问题上的可行性；由于无人机真实作业环境为三维空间，因此继续利用三维实验来证明该算法对于无人机在真实环境中执行多点巡航任务时的航迹规划效率优势。

本文所提算法得到的航迹质量与最优航迹的差距仅在于航迹长度，其他质量（如光滑度、可行性等）是一致的；并且本文所展示的数据为贪心 MB-RRT* 算法在得到第一个可行解后的航迹长度，在此之后，可以通过后续降采样优化对该长度进行缩短，拉近所得航迹质量与最优航迹的差距，可以预见，该后续优化所需要的时间将少于构建全连通图所需时间，特别是在三维空间中的时间优势更明显。另外，由于无序多点航迹规划问题为 NP 复杂度的问题，对于更为复杂的环境以及所需访问航点数量更多的航迹规划任务，如果仍采用全连通图的方法来得到最优航迹，时间代价将是巨大的。因此，通过牺牲部分路径质量来提升航迹规划效率的做法是可行的。

除了可以继续优化航迹长度外，未来可以在全局静态规划的基础上加入局部动态规划方法，使得能够对动态障碍物进行有效避障。

参 考 文 献

[1] 何雨枫，曾庆化，王云舒，等. 室内微型飞行器实时路径规划算法研究[J]. 电子测量技术，2014，37(2)：23-27.

[2] KAN E M，SIEN H J，PING Y S，et al. An evolutionary algorithm for multiple waypoints planning with B-spline trajectory generation for Unmanned Aerial Vehicles (UAVs) [C]// International Conference on Computational Problem-Solving. IEEE，2011：77-81.

[3] LAVALLE S M，KUFFNER J J. Randomized Kinodynamic Planning[J]. IEEE International Conference on Robotics & Automation，1999，1(5)：473-479.

[4] MELCHIOR N A，SIMMONS R. Particle RRT for Path Planning with Uncertainty[C]// IEEE International Conference on Robotics & Automation. 2007：1617-1624.

[5] FAIGL J. On Self-Organizing Map and Rapidly-Exploring Random Graph in Multi-Goal Planning [M]. New York：Springer International Publishing，2016.

[6] TRIHARMINTO H H，PRABUWONO A S，ADJI T B，et al. UAV Dynamic Path Planning for Intercepting of a Moving Target：A Review [C]// RoboWorld Congress. Springer Berlin Heidelberg，2013：206-219.

[7] MARTIN S R，WRIGHT S E，SHEPPARD J W. Offline and Online Evolutionary Bi-Directional RRT Algorithms for Efficient Re-Planning in Dynamic Environments[C]// IEEE International Conference on Automation Science and Engineering. IEEE，2007：1131-1136.

[8] WANG J W，DAI G M，XIE B Q，et al. A Survey of Solving the Traveling Salesman Problem[J]. Computer Engineering & Science，2008，30(2)：72-74.

[9] 来学伟. 贪心算法在 TSP 问题中的应用[J]. 许昌学院学报，2017，36(2)：41-44.

（上接第 74 页）

参 考 文 献

[1] 赵红红. 汉语阅读理解问答题解答研究[D]. 太原：山西大学，2016.

[2] FERRUCCI D，LEVAS A，BAGCHI S，et al. Watson：Beyond Jeopardy![J]. Artificial Intelligence，2013，199-200(3)：93-105.

[3] PEÑAS A，HOVY E H，FORNER P，et al. Overview of QA4MRE at CLEF 2011：Question Answering for Machine Reading Evaluation[C]// Proceedings of the Cross Language Evaluation Forum 2011 Labs and Workshop，Notebook Papers. 2011：303-320.

[4] 卢志茂，刘挺，李生. 统计词义消歧的研究进展[J]. 电子学报，2006，34(2)：333-343.

[5] 宗成庆. 统计自然语言处理第二版[M]. 北京：清华大学出版社，2013.

[6] 杨陟卓. 基于上下文语境的词义消歧方法[J]. 计算机应用，2015，35(4)：1006-1008.

[7] CHAN Y S，NG H T. Scaling Up Word Sense Disambiguation via Parallel Texts[C]// The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference. Pittsburgh，Pennsylvania，USA，DBLP，2005：1037-1042.

[8] PILEHVAR M T，JURGENS D，NAVIGLI R. Align，Disambiguate and Walk：A Unified Approach for Measuring Semantic Similarity[C]// Meeting of the Association for Computational Linguistics. 2013.

[9] LU W P，HUANG H Y. Word Sense Disambiguation Based on Dependency Fitness with Automatic Knowledge Acquisition[J]. Journal of Software，2013，24(10)：2300-2311.

[10] AGIRRE E，SOROA A. Random walks for knowledge-basedword sense disambiguation[M]. Cambridge：The MIT Press，2014.

[11] MANNING C D，RAGHAVAN P，SCHUTZE H. Introduction to Information Retrieval [M]. 王斌，译. 北京：人民邮电出版社，2010.

[12] 吴军. 数学之美第二版[M]. 北京：人民邮电出版社，2014.

[13] WOODS A M. Exploiting Linguistic Features for Sentence Completion[C]// Meeting of the Association for Computational Linguistics. 2016：438-442.

[14] LIU T，CUI Y，YIN Q，et al. Generating and Exploiting Large-scale Pseudo Training Data for Zero Pronoun Resolution[J]. arXiv. org/abs/1606. 01603.