北京航空航天大學
BEIHANG UNIVERSITY

第二十九届“冯如杯”学生学术科技作品竞赛
项目论文

Inself 多机协同路径规划

——基于栅格化地图的启发式寻路算法

摘要

本文主要设计一种路径规划算法以及多机协控策略，针对二维和三维情景下的较长距离寻路问题，以栅格化地图方案进行规划、最后优化为非栅格化路径。主要瞄准无人机等运算时空开支小的路径导航，在牺牲小的长度优势、换取时间和空间上开支的大大减少，为了适应现实中模拟地图寻路，最后将栅格化路径再次优化得到方向角度任意的模拟路径；同时以十架无人机为例描述一种多机协控移动的策略。

在栅格路径规划方面，我们借鉴了 A*的启发式算法，将启发式搜索应用在时空消耗较小但长度没有保证的深度优先搜索（DFS）中得到第一级路径，再利用视线导引法对所得路径进行优化修正，提出 Inself 算法，能够快速、省空间地得到较优路径结果。并以游戏界常用的 A*算法以及 Dijkstra 算法甚至广度优先搜索（BFS）算法作为靶算法进行对比，分析并测试二维和三维栅格地图下路径规划的速度和空间优势。结果表明，在 600*600 地图下，Inself 的平均时间为 A*的 8.26%、BFS 算法的 0.025%，开拓点数为 A*算法的 69.18%、BFS 算法的 1.11%，而路径长度平均增长为最短路径的 0.26%。在低牺牲路径长度情景下，有效减少了运算时空开支。最后基于最大三角形的思想，将栅格化地图路径还原到模拟地图中。

而在多机协控方面，首先以二维和十架无人机为例，提出了一种弹性绕圈式的编队策略，在多种遇障和阻碍情境下给出解决的调度策略，尽可能保持队形移动；而三维时则仍采用二维的队形，在其基础上多增加一个维度且优先保持同一水平面的编队移动策略。

关键字：规划算法，栅格地图，多机协同。

Abstract

This paper mainly designs a path planning algorithm and multi-machine co-control strategy, The rasterized map scheme is used to plan and optimize the rasterized path for the problem of long distance pathfinding in 2d and 3d scenarios. In order to adapt to the reality of simulated map pathfinding, the rasterized path was optimized again to obtain the simulation path with arbitrary direction and Angle.At the same time, ten planes are taken as an example to describe a multi-machine co-control movement strategy.

In the planning of the grid path, we draw lessons from the A * heuristic algorithm, the heuristic search application in space and time consumption but smaller length, there is no guarantee that the depth first search (DFS) of the first level path, reusing the line of sight guidance method to optimize the path correction, Inself algorithm, was able to get A better path quickly and save space. After Analyzing and test the speed and spatial advantage of path planning in 2d and 3d raster maps, we found that under the 600*600 map, the average time of Inself is 8.26% of A*, 0.025% of BFS algorithm, 69.18% of A* algorithm and 1.11% of BFS algorithm, and the average growth of path length is 0.26% of the shortest path. In the case of low sacrifice path length, the computation time and space cost can be reduced effectively. Finally, based on the idea of maximum triangle, the rasterized map path is restored to the simulated map.

In the aspect of multi-aircraft co-control, firstly, taking the two dimensional and ten planes as an example, a flexible formation strategy is proposed, and the scheduling strategy is given under various obstacles and obstacles to keep the formation moving as far as possible. In the case of 3d, the formation is still in the form of 2d. The formation movement strategy is to add one more dimension on the basis of the formation and keep the same horizontal plane in priority.

Keywords

Planning algorithm, Raster map, Multiple cooperating

目录

第一章 项目简介

1.1 项目背景

无人机编队飞行通过多个无人机的协同运行可以使系统的综合性能和任务执行效率大幅提高。在军事侦察、目标打击等方面，无人机编队飞行可以大幅提高任务完成效率，因此这项技术也是未来军事领域的一项关键性技术。在近些年，一些无人机研发团队也在利用无人机编队飞行技术执行飞行表演任务，单次表演编队数量可达 1000 架，系统规模庞大。在这样的系统规模下，由于无人机体积小，计算能力弱，续航能力不强，因此对无人机集群的协同控制算法一直是人们研究的一个重点。

在无人机集群协同控制算法中，多无人机路径规划以及队形编制策略是比较重要的两个方面。其中路径规划是实现无人机自主导航的关键技术之一，而队形编制则是要解决无人机编队巡航过程中的队形紧密性、避障策略等问题。传统的路径规划算法有人工势场法、栅格法等，近年也出现了一些诸如蚁群算法、遗传算法等人工智能算法。而栅格法建模简单，计算方便，因此基于栅格法产生了许多著名的路径规划算法，如 Dijkstra 算法、A*算法等，这些算法已经被广泛地运用到了游戏与应用软件当中。但随着新技术的发展以及新需求的产生，这些传统算法已经在某种程度上显现出了弊端，例如 A*算法计算量大、内存占用大，导致消耗的计算资源太多，不适用于现在的无人机控制系统。

1.2 项目制作的目的与意义

面对大规模下的寻路问题，特别是对于无人机等计算资源重要的终端寻路，如何快速、低算力、低空间开支得到路径是最大难点。本文提出了一种改进的 A*算法，通过启发式深度优先算法(Inspire DFS)与视线导引自遍历优化算法(Self)结合，可以在降低资源消耗的同时，极大幅度提高寻路速度，该算法简称为 Inself 算法。通过与其他几个传统算法的对比，结果显示这样的算法保证了无人机在飞行过程中可以更加快速地找到最佳导航路径，并且相对于其他传统的较为复杂的算法来说路径长度没有明显提升，因此 Inself 算法具有良好的性能。

而对于现实中的地图寻路，栅格化后的路径存在最大的问题是不能任意角度移动，

只能按格移动。为了得到适合现实中模拟地图的路径，对 Inself 的前两级栅格化结果再次进行优化，利用最大三角形法得到移动方向角度、坐标非整数的可行路径，从而更适应现实中移动路径规划问题。

为了躲避障碍、改变编队飞行方向，无人机编队必须能够根据导航路径以及实际路况变换策略。为此，本文提出了一种动态可变长机的编队调度策略，能够在 Inself 算法规划的路径下，编队内各个无人机的飞行路径能根据飞行方向以及飞行障碍动态调整，提高了算法的实用性。

在无人机越来越多的明天，随着外卖、快递等市场的不断拓宽，无人机外卖、城市速递的趋势必然会出现，而对于长距离的、动态复杂的路径规划以及多机协同编队控制的问题，如果采用现有常用的 A*、D*等最短路径算法，空间开支、算力等需求较大，不适应无人机等对续航、算力和空间敏感的终端，故本文可以对未来大规模寻路提供一种基于栅格化的、算力和时空开支明显降低的规划方案。

1.3 项目创新点

在路径规划方面，针对大规模地图下的快速低算力的寻路问题，采用栅格化预处理+回归模拟地图的解决方案，形成了三级算法构成的 Inself 算法。其中栅格化预处理部分，借鉴 A*等成熟算法的启发式思想改进，对比 BFS 和常用的 A*、Dijkstra 等算法，在地图规模增大情况下有非常明显的时空开支的减少。

而在多机协同方面，我们谨提出一种编队方式，利于集群飞机协同前进，依靠 Inself 算法的路径进行动态协调，直至到达终点。

第二章 常见栅格地图算法简介

在路径规划问题中，为了简化处理，栅格化地图是常见的处理形式，而在栅格化地图的传统寻路方式中，大致分为深搜类、广搜类以及与启发式结合的各类变形。为了能够回归到模拟地图中，常常需要对栅格地图下的路径再次优化，得到更优的路径。

2.1 广度优先搜索（BFS）

在迷宫问题中，BFS 是求解最短路径的最常见也是最基本的解法。对于一个栅格地图，每个点都有父结点、坐标值以及自源点移动至当前点的最小距离消耗 G。实现

BFS 依赖于 open 队列和 closed 表。

算法流程为：从起点入队开始，每轮取队头元素作为探索点并出队、加入 closed 表内。探索点的探索过程为，对其周边可到达且非 closed 点，若未在 open 队列中，则计算从探索点到此相邻点后对应的 G 值：$G_{相邻点} = G_{探索点} + G_{移动消耗}$，并将探索点作为此点的父结点，最后入队；而对于已在队列中的相邻点，若新计算 G 值比相邻点之前的 G 值更小，则更新 G 值，并将队中此点的父结点也更新为当前探索点，不改变其在队中的顺序。按照此探索顺序，直至队为空或探索至终点。若存在通路，则从终点反向遍历父结点直至起点，即为最短路径。

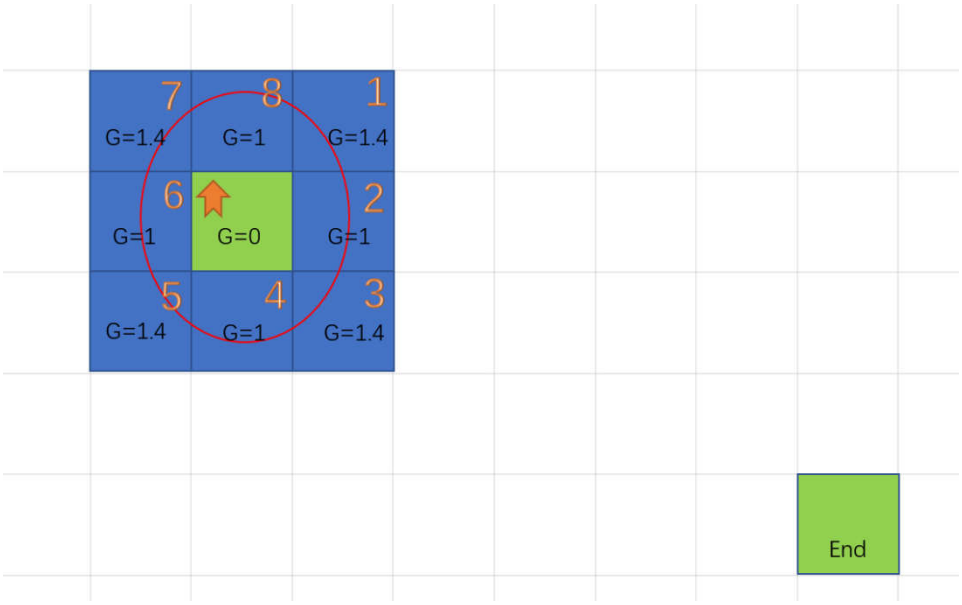

图 1　BFS 算法示意图（1）

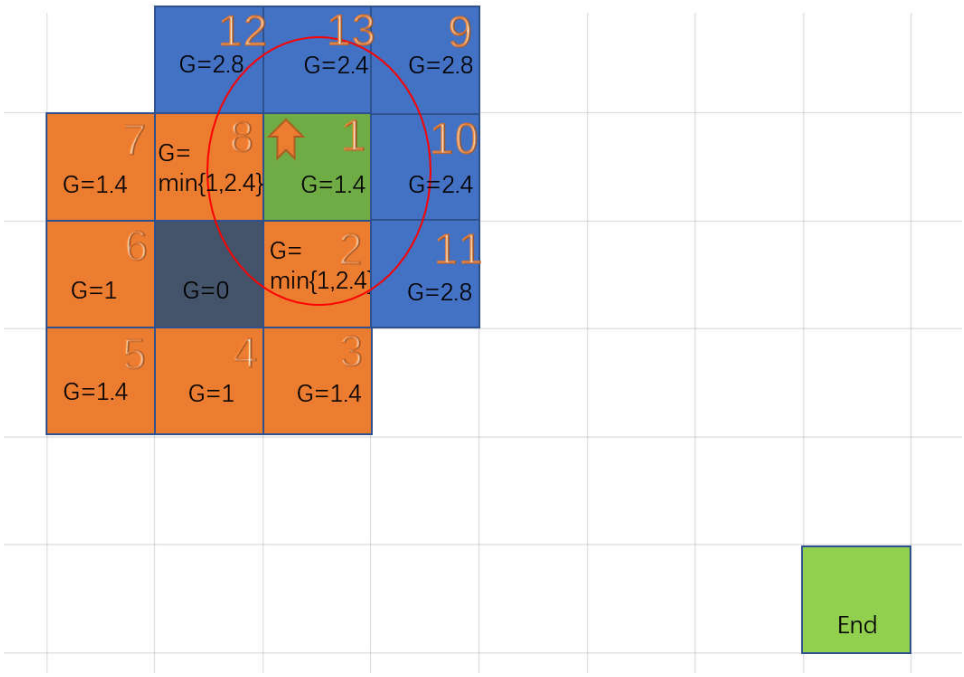

图 2　BFS 算法示意图（2）

2.2 Dijkstra 算法

Dijkstra 算法仍是类广搜的一种基本算法。由于 BFS 在开拓时严格按照队列顺序出队探索，故可能存在较多的 G 值更新、更换父结点的情况。为了减少这种情况的发生、提高“有效”入队效率，采用最小优先队列来代替 BFS 中的队列，即通过最小堆使得每次弹出的探索点均为 open 表中的 G 值最小点。这种最小堆的应用，使得探索顺序按照从源点“划圈”的方式向外探索，而不完全取决于入队的顺序。由于 G 值取值为源点到当前点移动消耗最小值，故这种以 G 值最小堆的 Dijkstra 算法能够在一定情况下提高 BFS 的速度，减少 G 值重复更新赋值，但开拓点数并不会有明显变化。

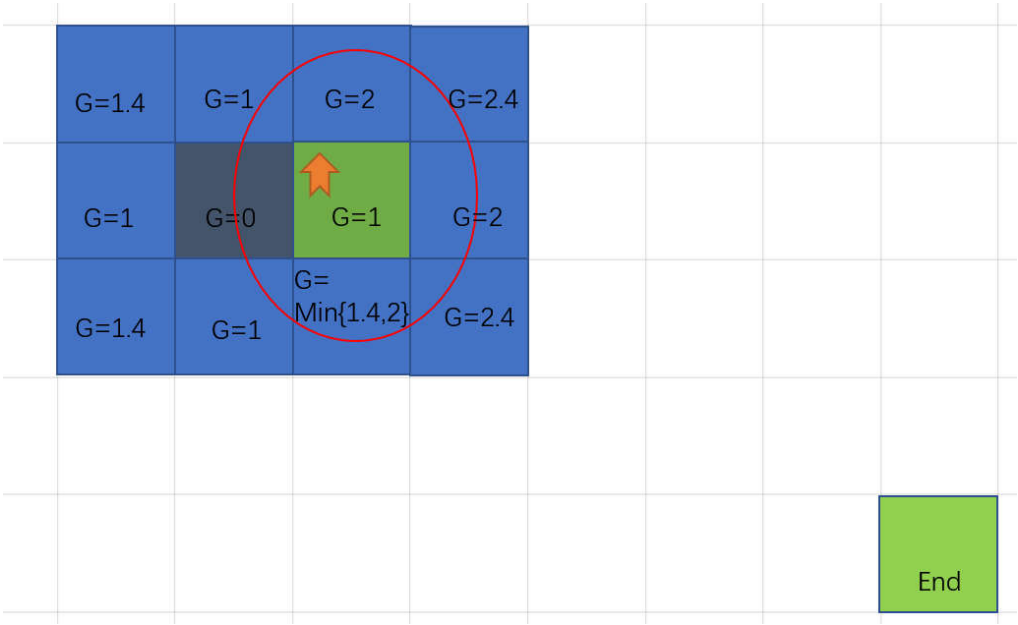

图 3　Dijkstra 算法示意图（1）

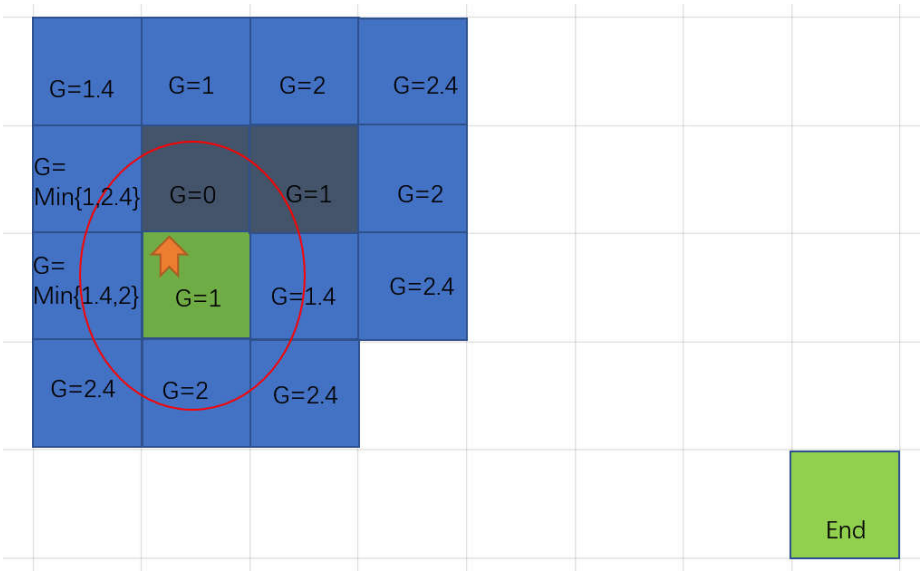

图 4　Dijkstra 算法示意图（2）

2.3A*算法

A*算法是目前游戏界常用的寻路算法，如很多类似 DOTA 的 2.5D 游戏中，常采用 A*算法进行寻路。

2.2 中的 Dijkstra 算法中，是利用最小堆，优化 BFS 队列的出队探索顺序。但本质上仍是从源点开始划圈，直至终点，也正因此能够保证是最短路径。而 A*算法为了进一步加快速度，可以减少探索、遍历过的点，每次在 open 表中弹出的元素按照启发函数值 F 最小弹出，即以 F 构造最小堆，这样每次弹出的都是 open 表中节省移动消耗

和接近终点的最佳点。可采用 F=G+H，其中 H 为当前点与终点的距离估算，由于栅格地图，故采用曼哈顿距离，如下图所示：

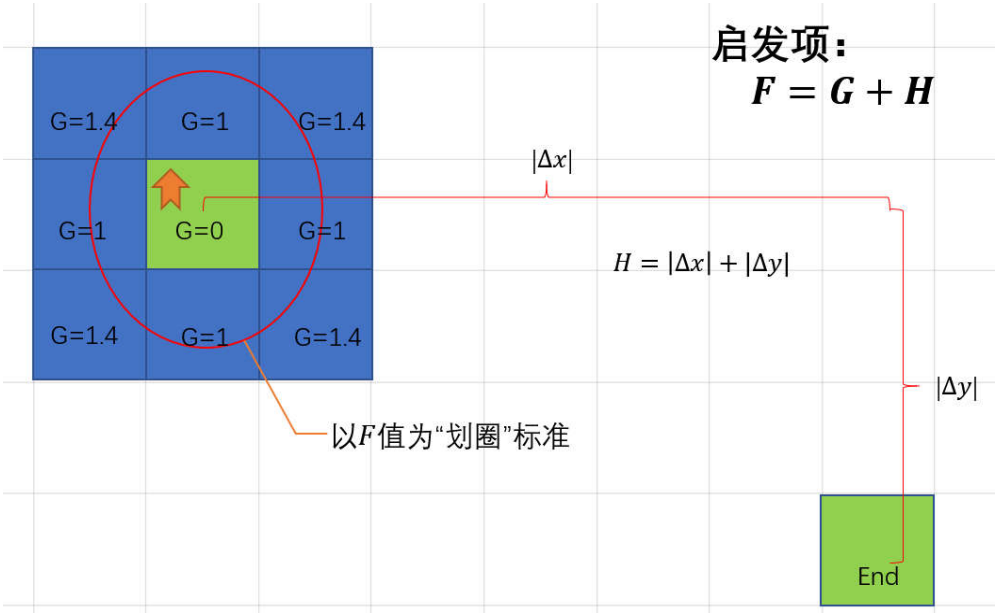

图 5　A*算法示意图

即 Dijkstra 和 A*都是启发函数的一个例子，只不过启发函数不同。而 A*通过这种方式，能够引导 open 表弹出更趋向于终点的探索点，在很多情况下能够减少探索点，从而提高速度。换个角度思考，A*的启发函数像给 BFS 加了一个“重力”，引导并加速了 open 表尝试出更近更快的路径，探索结点数有明显减少；但同时需要维护一个最小堆，相比于队列，在数据量较大时耗时也更长。

2.4 深度优先搜索（DFS）

BFS 能够保证最短路径，但耗时、空间都较大。而 DFS 利用一个栈结构，不断地尝试入栈、遇死路退栈，直至找到终点，在开拓节点数上通常远小于 BFS，且栈结构也较简单，故时间和空间消耗上通常小于 BFS；但 DFS 的缺点也同样明显，往往找出一条并非最短路径的结果，DFS 更适合快速寻路、但路径长度性能难以保证。

第三章　Inself 算法

3.1 栅格地图规划

Inself 算法的栅格地图寻路部分采用两级算法完成，先对 DFS 加入 A*算法中的启

发项进行引导修正，从而快速的得到一条更易趋向终点的通路，再对这条通路进行视线导引法优化，去掉多余的拐弯，从而较快的得到路径。

3.1.1 第一级：启发式深搜

借鉴 A*的思路，将启发式算法思想与 BFS 相结合，能够更快地趋向于最终的最短路径开拓元素，从而减少开拓点数。若在深度优先搜索中加入启发函数项，可以利用 DFS 的快速省空间的优势同时，利用启发式一定程度上弥补其长度性能的劣势，让启发的道路能够趋向目标、提高路径质量。

算法思路为：仍采用一个栈结构和 closed 表进行深搜，但同时每个结点具备启发函数值 F。对于每个探索点而言，探索过程为计算/更新不在 closed 表中的临结点 G 值以及父结点信息，并以 F 值最小点入栈；若为不存在，则退栈一次；再将探索过的结点填入 closed 表。每轮结束后再以栈顶元素作为探索点重复上述过程，直至栈为空或探索至终点。若存在路径，则为自终点逆向遍历父结点至起点的所得路径。

在启发式深搜中，需要维护的最小堆大小是恒定的（二维情况下≤8，三维情况下≤26），为周边相邻可到达且未在 closed 表中的点，数量较小；移动到新位置后堆元素全部更新，故不存在 A*等算法中堆不断扩展从而耗时较长的问题，且由于能够直奔终点，减少了很多额外探索，进一步节省运算空间。

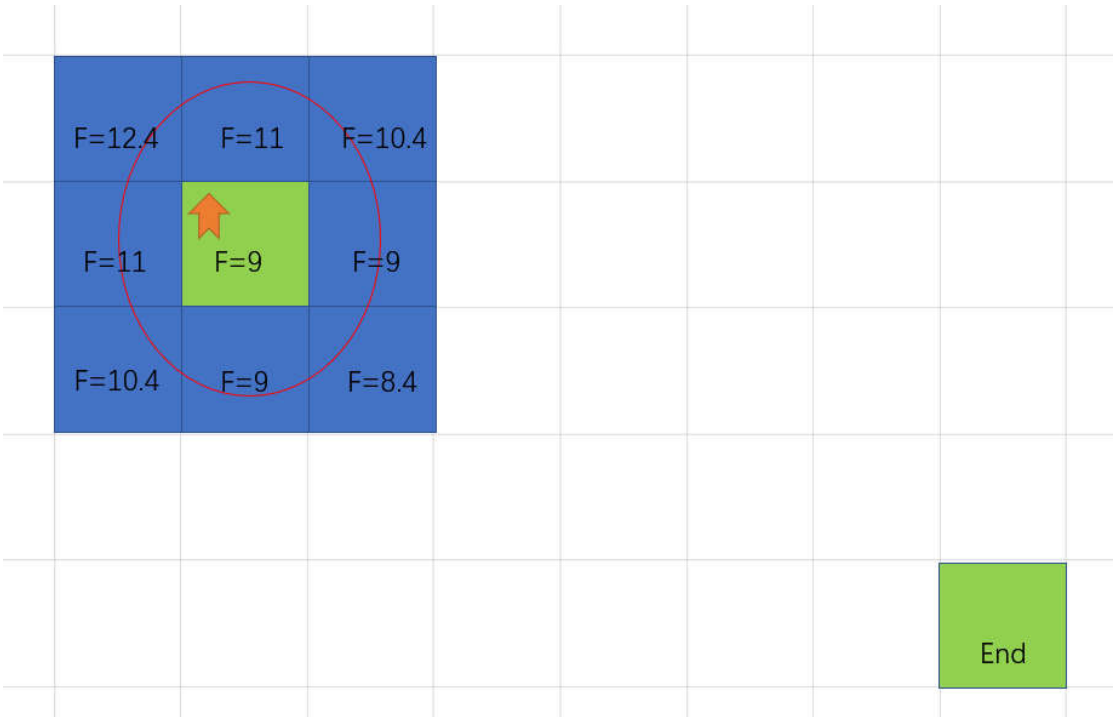

图 6　Inself 第一级算法示意图（1）

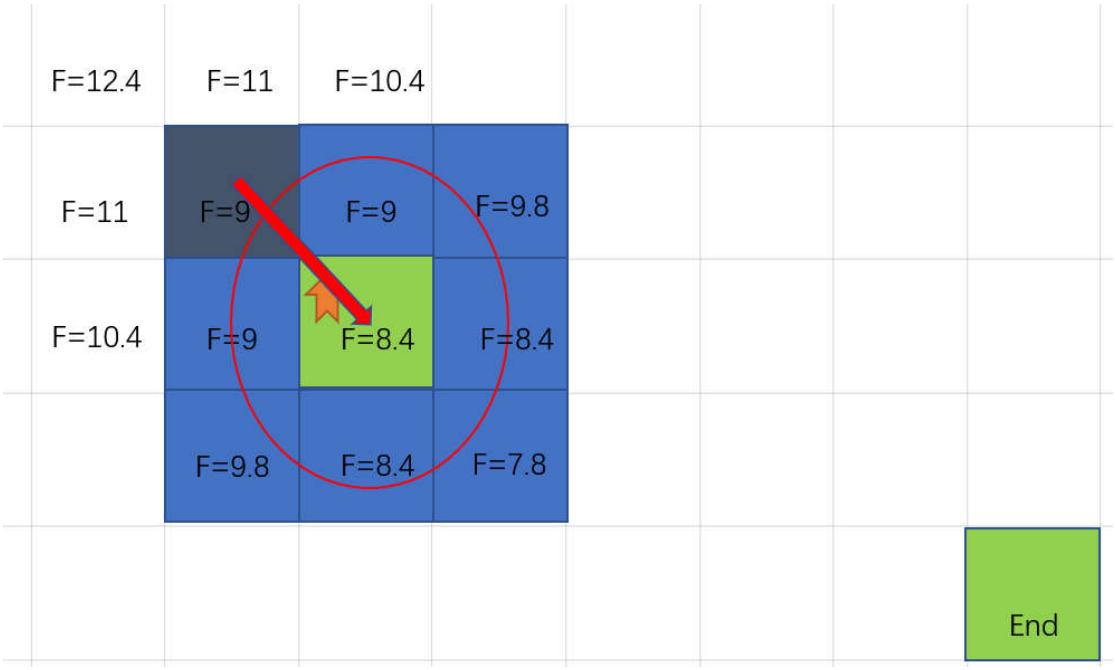

图 7　Inself 第一级算法示意图（2）

3.1.2 第二级：视线导引法

在启发式深搜后，虽然能够很快地得到一条较好的路径，但由于死路式障碍的遮挡，往往存在一些绕远、甚至是死路遍历的情况（如下图），经过测试发现，这是由于在 F 值计算结果相同时，邻点选择顺序不同会导致面对此类障碍时的反应不同。不管哪一种，都会比直接绕过障碍牺牲路径性能，因此，对路径进行适当的二次优化显得十分必要。

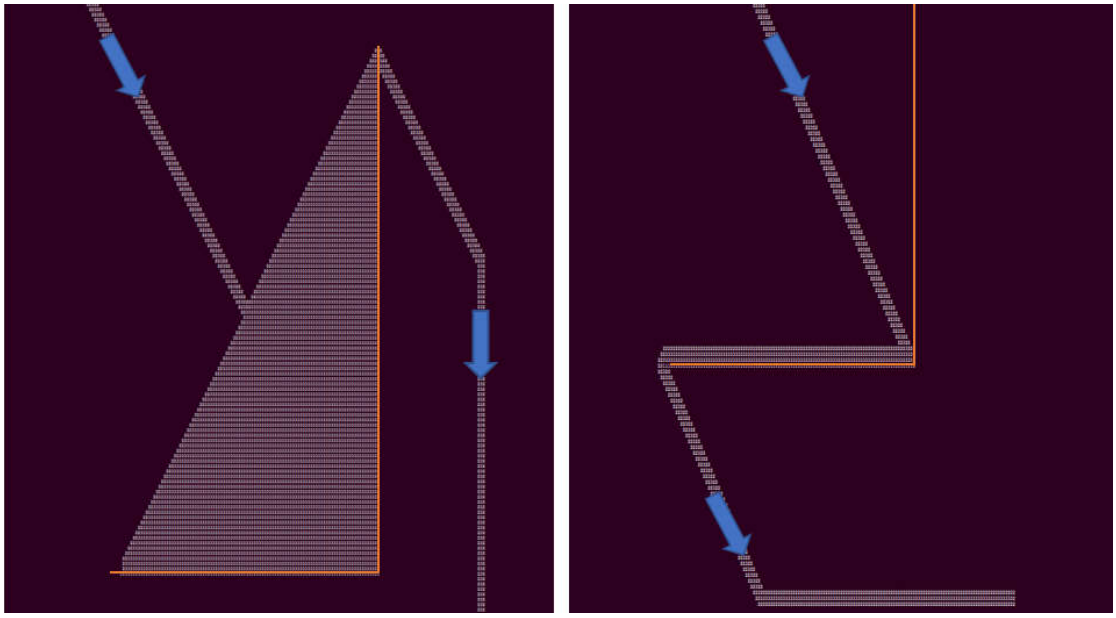

图 8　Inself 第二级算法要解决的问题

而路径再优化的方案也有很多，譬如多边形填充法、图形切割等等，但出于速度和空间的考量，我们采取自遍历的方式和视线导引法对路径绕远的问题进行优化，思路如下：

利用嵌套的两级 for 循环遍历路径链表结点，记外循环结点为 i，内循环结点为 j，若 i 和 j 在栅格地图中能够以直线连接且 i 和 j 直接没有障碍物，则将 i 和 j 直接的结点都删掉，并将其子结点、父结点填充入 i 和 j 之间线段的结点，从而删去了绕圈的路，优化路径。这种连直线优化的方式仅当 i 和 j 能够互相“看到”时才能够相连使用，故称为“视线导引法”，其时间复杂度为 O(n^2)，n 为第一级路径结点数，通过遍历的方式寻找其中可能存在的捷径，把非全局最优的解优化成为更好的解。

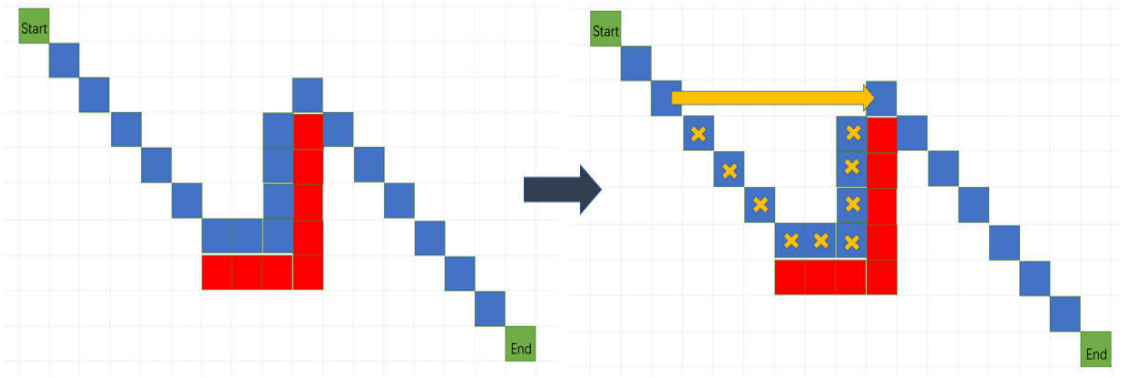

图 9　Inself 第二级算法效果

3.1.3Inself 算法栅格地图部分伪代码

```
//路径探索
curpoint = startpoint;
while(curpoint != endpoint)
    path.push(curpoint);
    curpoint = point in surroundpoints with minF & not opened;
path.push(endpoint);
//路径优化
while(point.parent != startpoint)
    tmp = point.parent;
    while(tmp = startpoint)
        if(( (abs(tmp.x - point.x) == abs(tmp.y - point.y))
        || (tmp.x == point.x) || (tmp.y == point.y) )
        && is not blocked )
            update subpath point -> tmp;
        tmp = tmp.parent;
```

3.2 回归模拟地图

模拟地图栅格化后虽然便于计算机处理，且量化大小不同，得到路径的模糊程度和计算耗费此消彼长，但存在一个很大的问题：无论量化多么精密，最终的路径只能是直线式平移或 45 度角斜向移动构成，难以使用现实寻路的任意角度方向可移动的场景下。故为了适应模拟地图，补充 Inself 的第三级算法，以最大三角形法的思想，将栅格化得到的路径进行优化。

在优化之前，我们首先分析一下前两级算法所保障的结果：由于第二级算法是对栅格地图下直/斜通连接的结果，保证了是栅格地图下不会绕远的路径，即路径中任意两点不可能不经路径而直连或斜连。保证这一点后，我们发现在三维地图下，或许只需对拐点进行优化即可，即让最终路径与水平的夹角为任意值，而非只有 0、45、90 度三种情况。但为了得到真正最优的路径，障碍可能存在于栅格路径附近的任何位置，故提出最大三角形法进行优化：

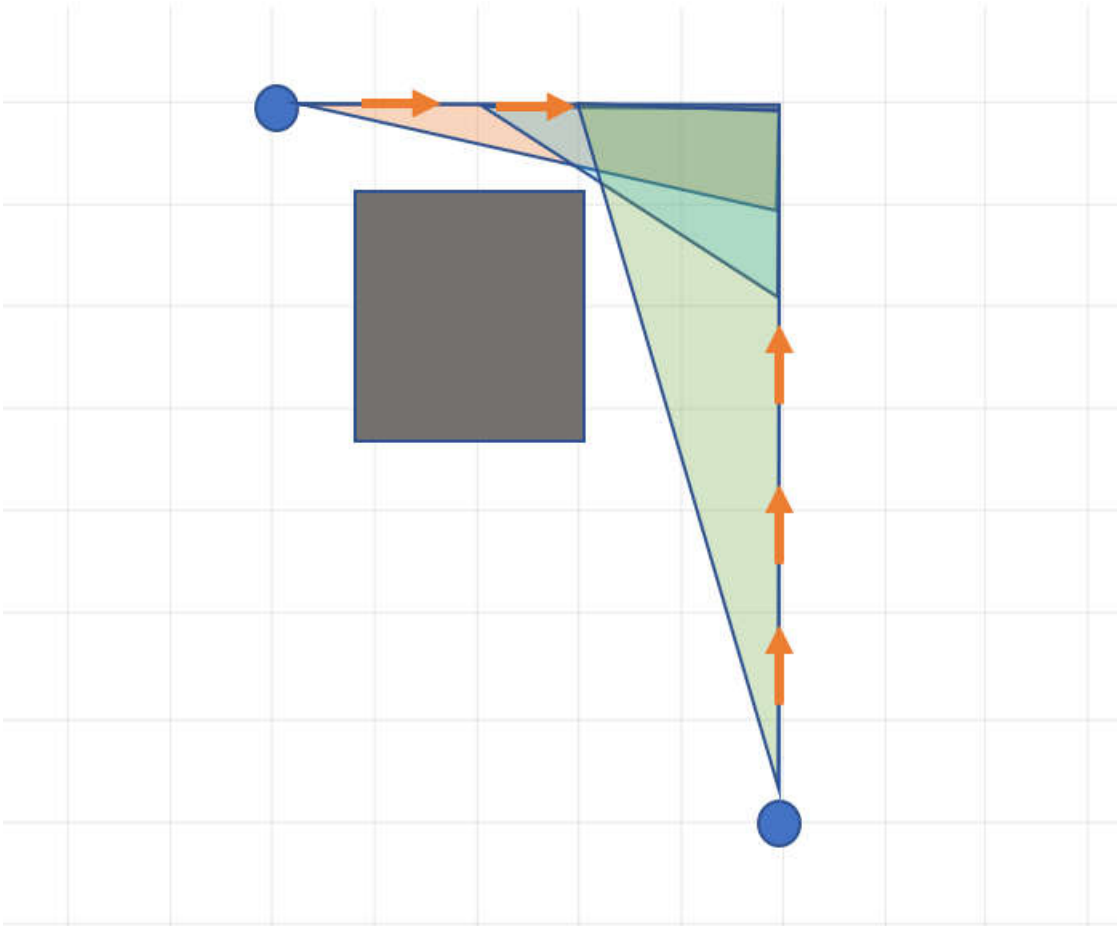

图 10　最大三角形法回归模拟地图示意图

由前两级算法可知，若要在栅格路径中得到模拟路径，则每段独立完整的优化路径必然在相邻的三个结点之内确定。故可采用最大三角形法，将优化路径的两点分别从相邻两条边进行遍历，测试能否直连，找到使得直连后三角形面积最大的边。对于整条路径而言，每相邻的三个点都经此方法进行优化，即可得到方向角度任意的基于栅格化 Inself 算法的最优模拟路径，从而回归到现实的寻路问题中，对目标的移动方向策略提供可执行的参考。

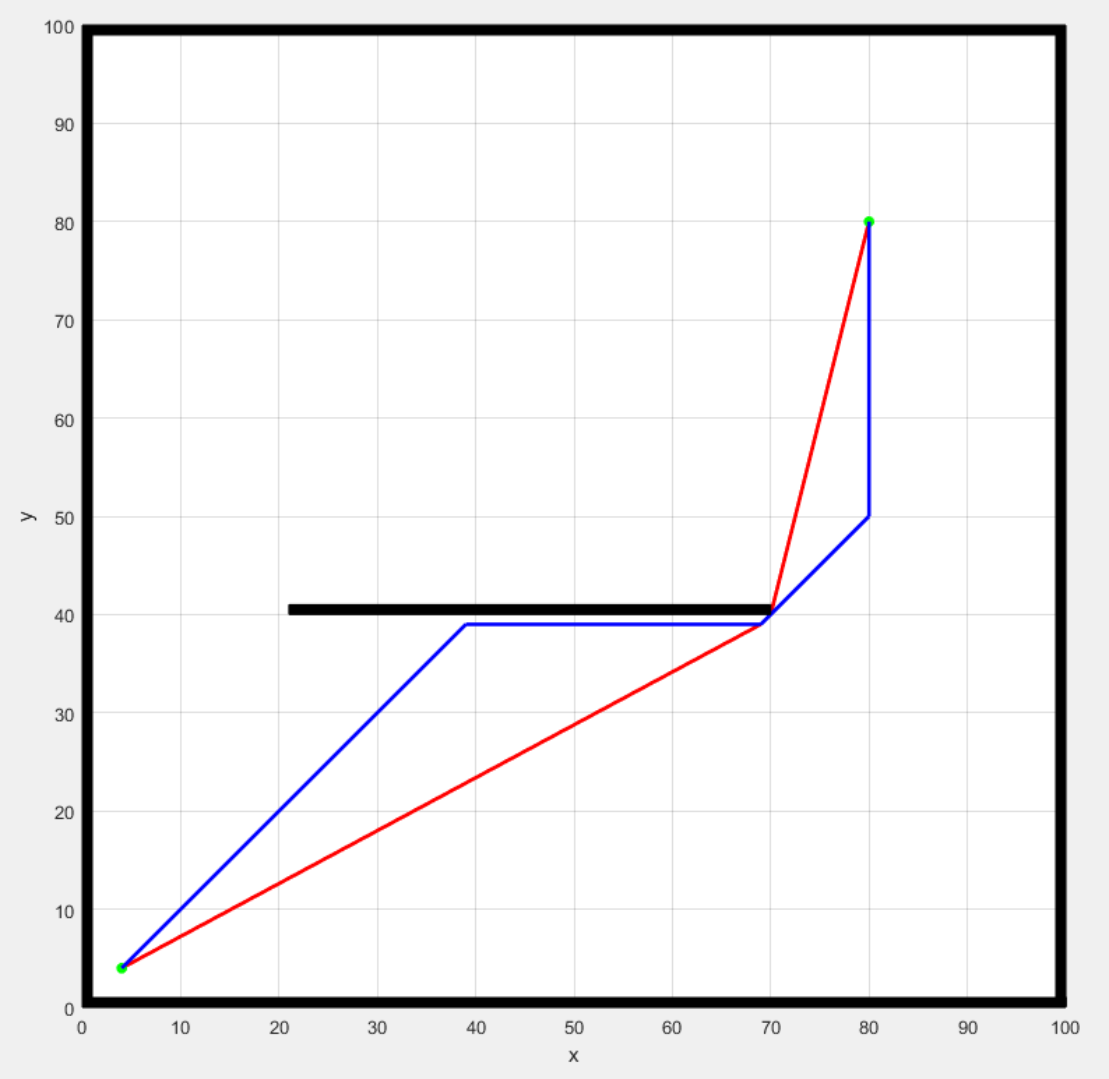

图 11　最大三角形法回归模拟地图效果

第四章 Inself 算法性能测试分析对比

为了统一对比，以 Inself 算法的栅格地图寻路部分与其他算法进行比较。二维三维的算法思想一致，故三维情况下只进行实测对比，不再重述复杂度分析。

4.1 复杂度分析对比

对于总网格数为 n 的栅格地图，不同算法的表现如下：

广度优先搜索（BFS）：在栅格地图上的广度优先搜索，类似于用邻接表保存的无向图，其中每个节点与其相邻的多个节点之间存在无向的通路，因此时间复杂度为 $O(n^2)$；

Dijkstra 算法：使用的仍然是广度优先搜索的思想，区别在于广度优先搜索是按照节点入队顺序进行探索的，而 Dijkstra 算法是按照距离源点的距离进行探索的，优先探索距离源点近的节点。所以，和 BFS 相比，Dijkstra 算法还需要维护一个以源点距离为判断元素的优先队列，因此 n 个节点的栅格地图下 Dijkstra 算法的时间复杂度为 $O(n^2logn)$。

A*算法：A*算法和 Dijkstra 算法相似，都维护了一个优先队列，区别在于 A*算法探索节点的顺序是由启发函数决定的。A*算法的时间复杂度为 $O(n^2logn)$，但是由于启发函数对探索方向有引导作用，因此实际上 A*算法探索的节点数会远远小于 n，实际的时间复杂度也会远小于 $O(n^2logn)$。

Inself 算法：Inself 算法同样使用了启发函数，和 A*的区别在于 Inself 算法选择相信启发函数对每一步选择的引导，直接根据当前节点和启发函数决定下一个要探索的节点，只在当前节点周围一圈的最多 8 个节点中进行选择，相比 A*不需维护一个优先队列。针对于节点数量庞大的栅格地图，这将是一个很可观的时间优化。Inself 算法在到达终点之后，再对得到的路径进行化简，所以整个算法的时间复杂度为 $O(n^2)+O(n^2) = O(n^2)$。

表 1　Inself 与几种常见算法的时间复杂度对比

栅格地图算法	BFS	Dijkstra	A*	Inself
时间复杂度	$O(n^2)$	$O(n^2logn)$	$O(n^2logn)$	$O(n^2)$

从上表及上述分析中可以看出，Inself 由于不需维护优先队列、普通队列等数据结构，只需采用定长比较，时间优化、特别是空间优化非常可观，时间复杂度为 $O(n^2)$。

4.2 蒙特卡洛仿真测试对比

4.2.1 二维情况测试对比

为了比较上述几种算法在实际应用中的性能表现，我们使用不同规模的随机障碍栅格地图进行测试，探究运算时间，开头点数以及路径长度的差距。

在不同规模的栅格地图中，进行大量随机性障碍设置重复实验，测试在每一次地图下，四种不同路径规划方案的运行时间以及开拓点数和路径长度，最后对同规模地图下每次数据的运算时间、开拓点数进行平均，且将每次 Inself 的路径长度较 BFS 最短路径长度增长比计算后再取平均，表征该规模下路径长度牺牲的平均程度，结果如

下：

表 2　Inself 与几种常见算法的仿真测试对比

指标 / 规模	运算时间/ms				开拓点数				路径长度牺牲
	BFS	Dijkstra	A*	Inself	BFS	Dijkstra	A*	Inself	
200*200	22908	31569	812	31	29938	30323	1009	1014	4.64%
400*400	171659	235296	8128	88	119198	120716	2805	2129	1.60%
600*600	597999	796816	18165	150	268557	272491	4296	2972	0.26%
1000*1000	-	-	30909	359	-	-	6170	5130	0.00%

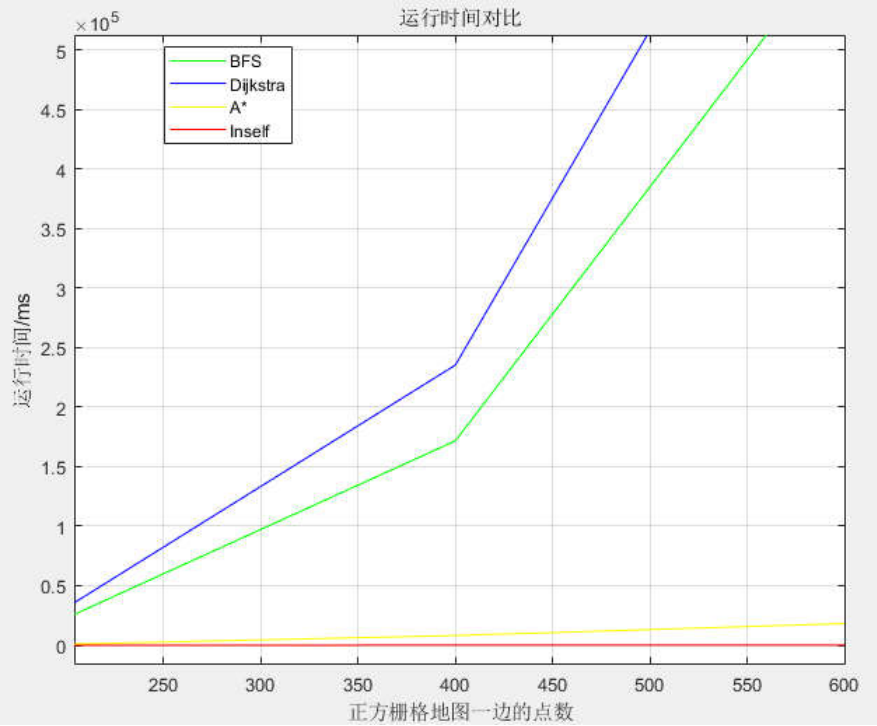

图 12　Inself 与其他算法时间测试对比（1）

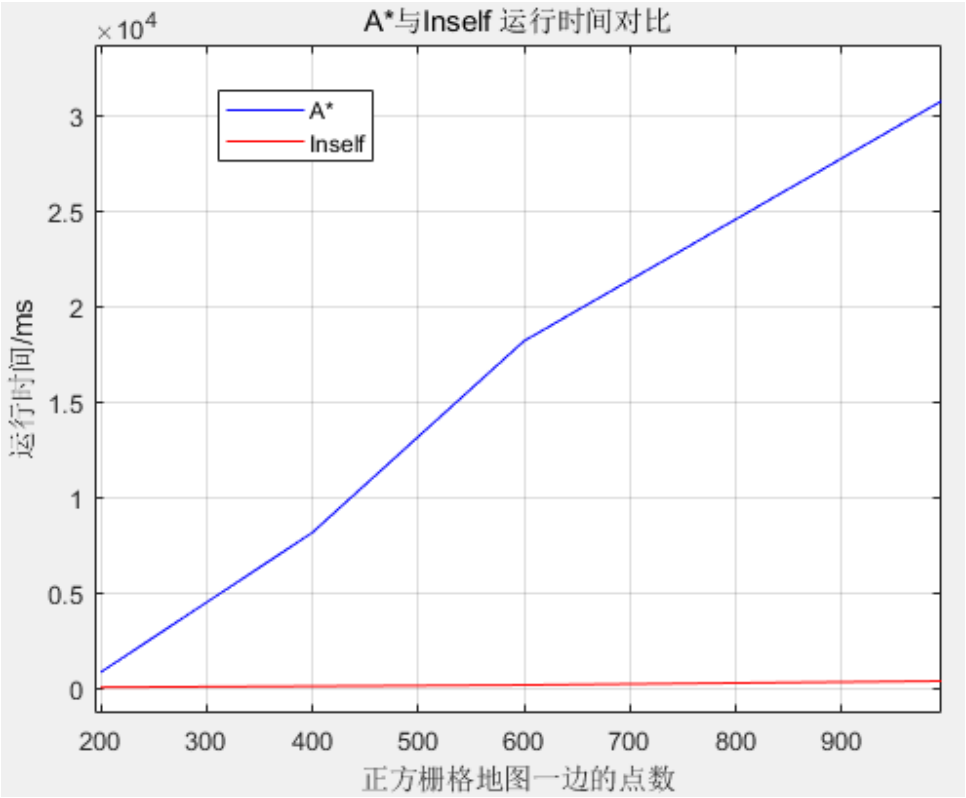

图 13　Inself 与其他算法时间测试对比（2）

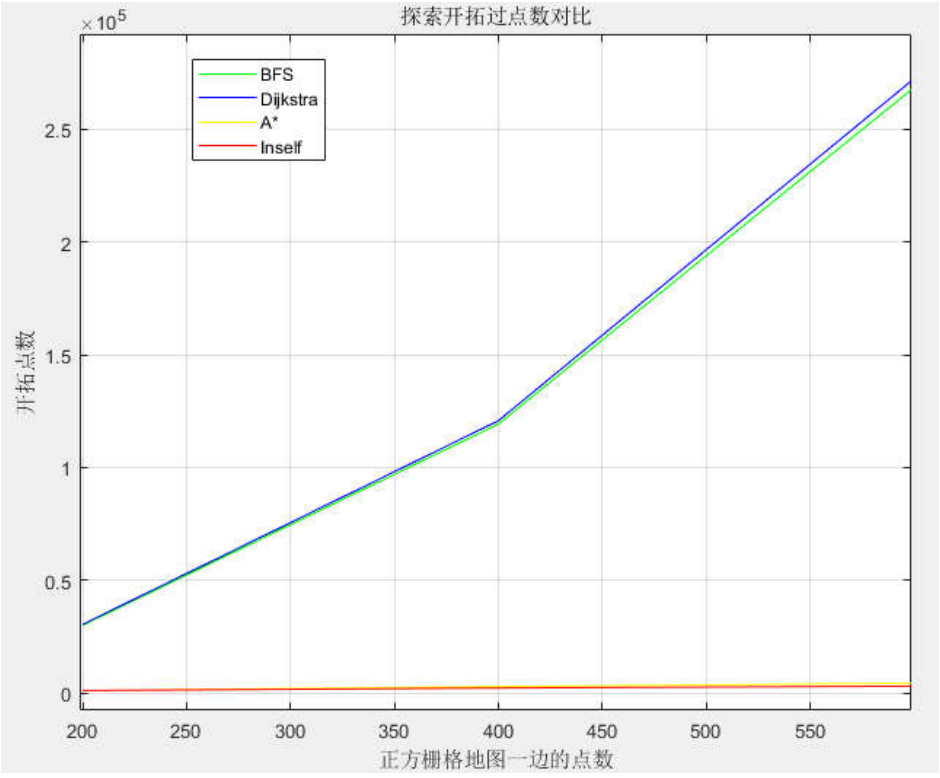

图 14　Inself 与其他算法开拓点数测试对比（1）

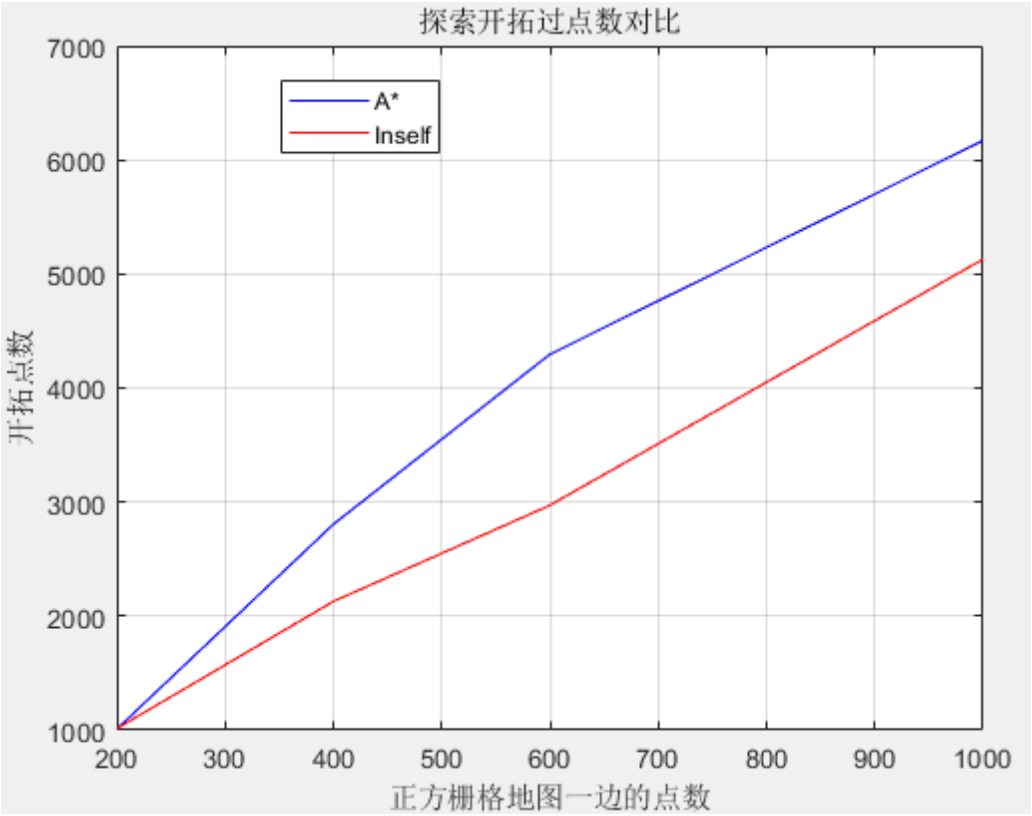

图 15　Inself 与其他算法开拓点数测试对比（2）

4.2.2 三维情况测试对比

在三维栅格地图下取不同规模：20*20*20、30*30*30、40*40*40 地图规模下，与上述相同的性能参数对比，结果如下：

表 3　Inself 与几种常见算法的仿真测试对比

指标 / 规模	运算时间/ms				开拓点数				路径长度牺牲
	BFS	Dijkstra	A*	Inself	BFS	Dijkstra	A*	Inself	
20*20*20	61562	59113	40	4.7	7953	7953	318	319	1.59%
30*30*30	690823	6202560	116	8	26823	26823	500	501	0.00%
100*100*100	-	-	1875	33.7	-	-	1703	1704	0.00%

可以看出，当地图增大时，BFS 和 Dijkstra 的运算时间和开拓点数迅速增加，对于较大规模的地图就已经不实用。A*算法的耗时相比前两种算法表现优秀，但是和 Inself 算法相比则显得十分缓慢。另外，Inself 算法在大大节省运算时间的同时，计算得到的路径与最短路径的差距很小，而且地图规模越大，Inself 算法得到的路径就越接近最短路径。由此可以看出，对于大规模地图，Inself 算法最为合适。

根据实验的出的结果分析，Inself 算法相比 A*等常见算法主要具有以下优点：

(1)运算时间快，开拓点数少，能有效节省无人机的计算和内存开销；

(2)利用 Inself 算法寻路时间短的优点，可以通过每隔一段时间重新调用一次算法，重新规划路径的方法，从而实现动态寻路，既不会出现卡顿，也不会影响无人机的其他性能。

(3)Inself 算法相比其他算法，可以处理同等精度下更大范围的地图，或者是同等范围下更高精度的地图（甚至可以以单个无人机大小为单个栅格的大小，从而找出比 Dijkstra 算法更短的路径）。

第五章 多机协同

5.1 协控调度策略

5.1.1 二维栅格地图情景

在路径规划得到一条可行的单机路径后，再通过长机带路的方式对多机路径进行协控规划、得到不同时刻飞机的位置坐标集合，即为机群的移动情况。借鉴连笔画的思路，二维机群可采取一笔画出的实心正方形方式提高队伍的紧凑程度，同时队伍也具有弹性可压缩的空位置，无论是多少架无人机，均可采取这个思路进行配置机群队形及编号，现以十架为例，调度策略如下：

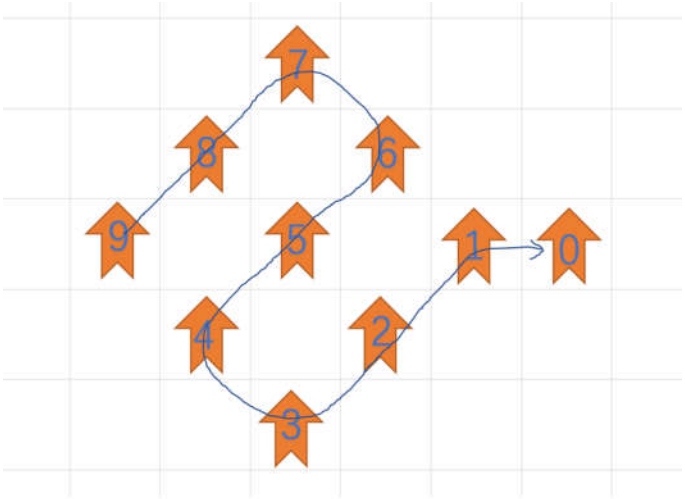

图 16 多机编队示意图

通过 0 号长机按照规划好的路径进行移动开始，每轮移动都要遍历十架飞机解决冲突并得到当前一组坐标，首先判断若长机能够按路线移动至下一点，再按编号从小到大依次判断分配移动情况：

① 若能够与长机同向移动且编号更小的均可通向移动，则此点也通向移动；

② 否则，尝试按照成非钝角坐标轴投影原则的方向移动，如 7 号所示：

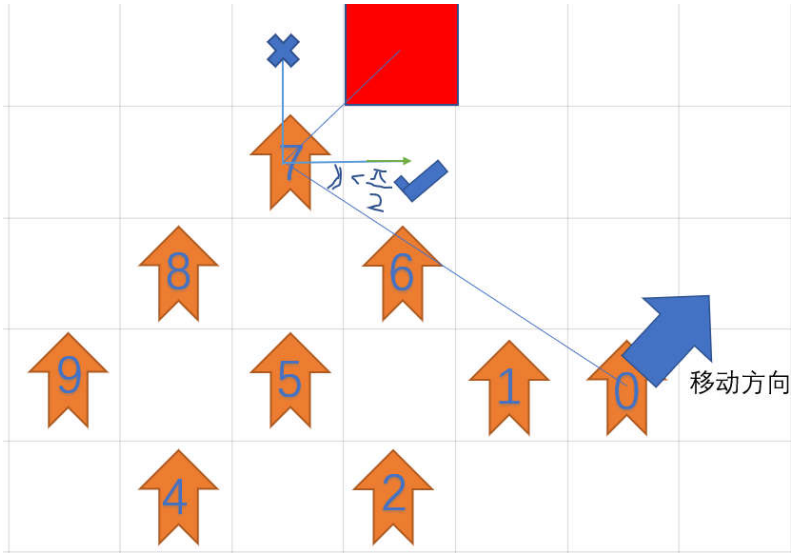

图 17　非钝角坐标轴投影原则移动

③若仍不可，则移动到其上一编号无人机的上一时刻位置处，即紧跟上一编号无人机。以上为正常情况的三种方式，但还有两种补充方案以防万一：

1. 防掉队的距离判断，若某无人机与长机距离超过起始时的两倍，认为掉队，大部队进行等待，掉队机重新进行路径规划至队伍中去。

2. 更换长机方案：如果一开始判断长机不能按照既定的路线移动至下一结点，即被其他飞机围死在角落中，则遍历其他无人机，找到能够与终点有可行路径且能够移动下一步的无人机作为长机，其他飞机重新编号，重复进行路径规划、协同过程。

5.1.2 三维栅格地图情景

三维情况与二维类似，且由于在停止时都是平放，高度相同，故机群仍按照二维的原则进行分布，相同高度地保持队形是最理想的情况，只是在路径规划和协控时多一个维度可以进行，即可以斜上、斜下或是上下地移动，其他冲突和解决策略与二维如出一辙，仍采取三个方案与两个补充策略相互保障的方式进行，最终输出十架无人机的三维同步移动路径，其他行为操作与二维情景相似，不再赘述。

5.2 多机演示效果

5.2.1 路径规划实测效果

通过路径规划得出的路径以及栅格地图信息，能够绘制出规划后的通路情况如下：

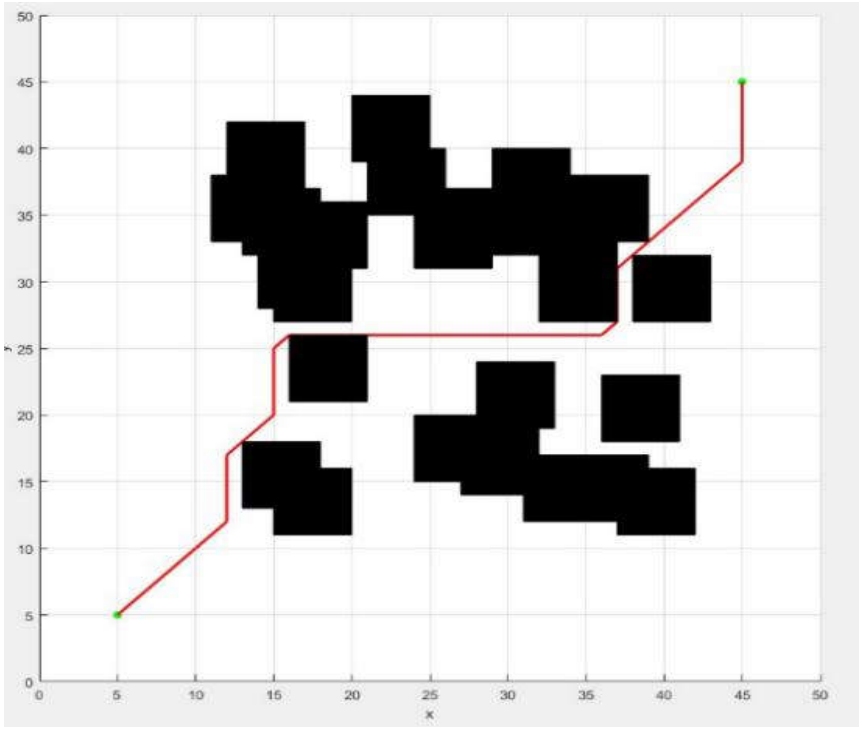

图 18　二维地图下规划结果测试（50*50 地图）

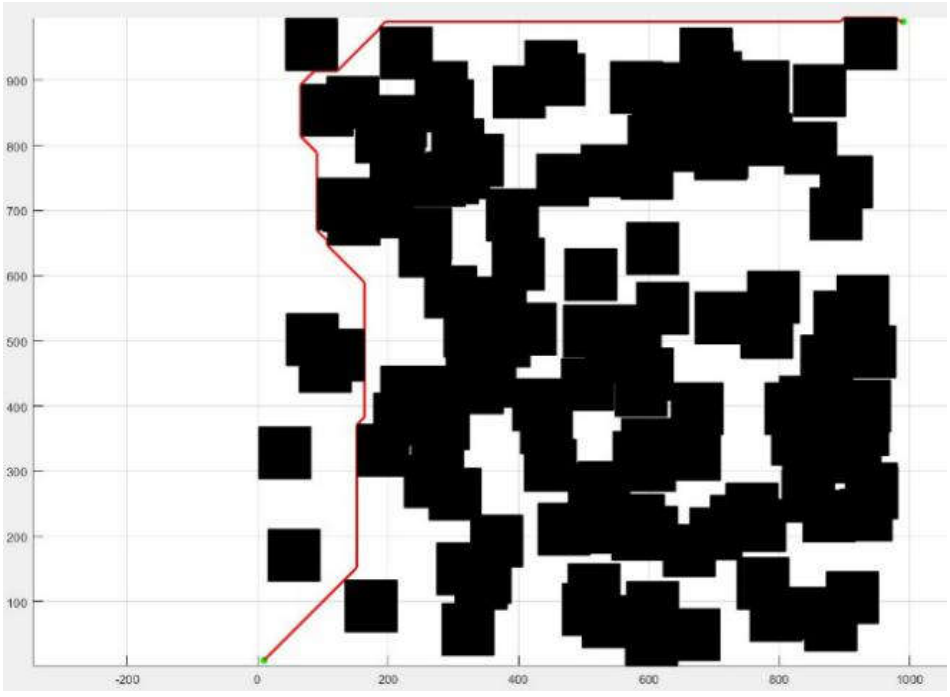

图 19　二维地图下规划结果测试（1000*1000 地图）

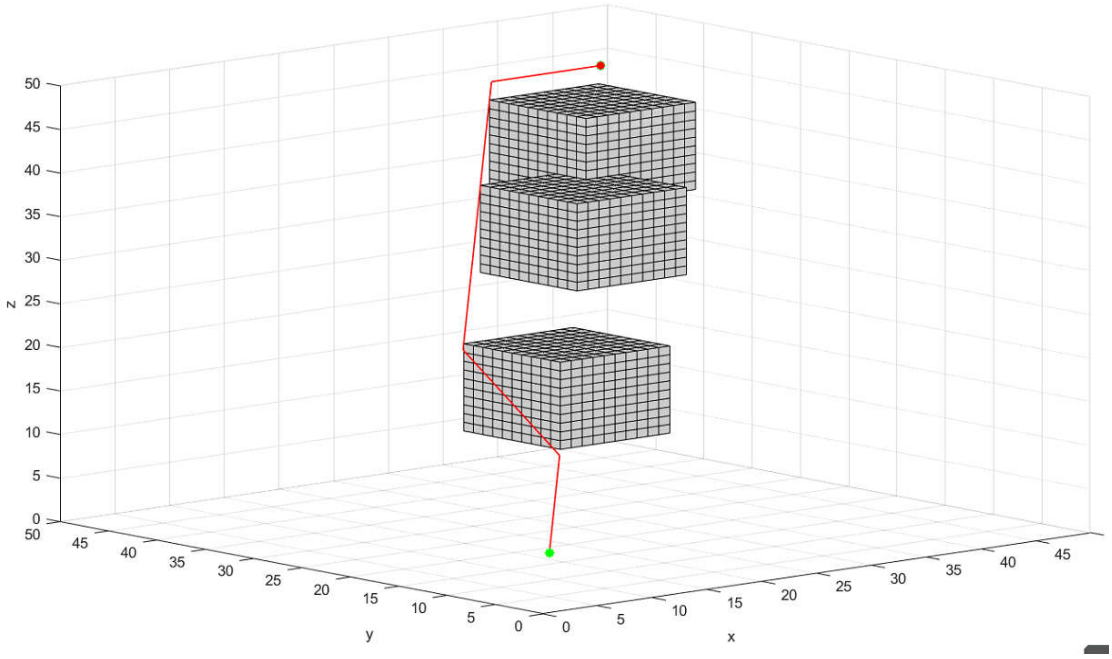

图 20　三维地图下规划结果测试（50*50*50 地图）

5.2.2 多机协同实测效果

通过多机协作策略和第一次规划好的策略，通过调度和冲突解决的方案后输出多机（10 个）每一时刻的坐标位置信息，即代表机群的移动情况：

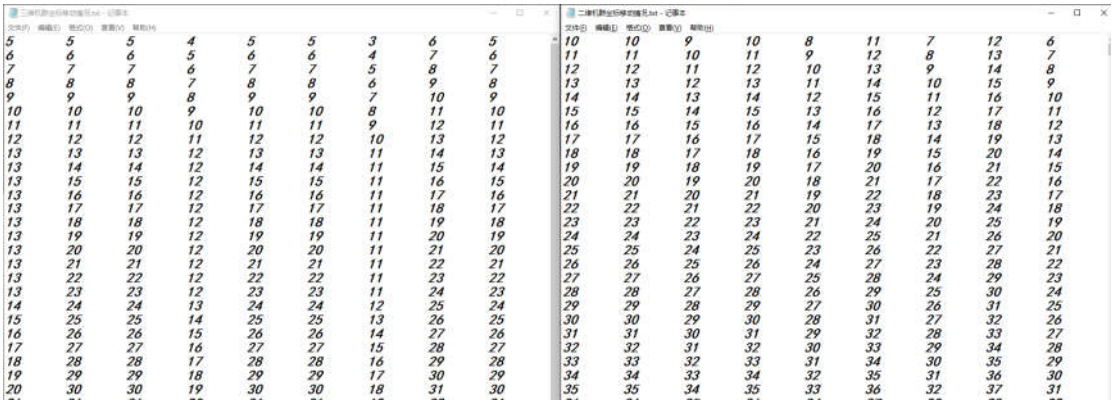

图 21　多机协同输出实时坐标变化

最后通过 Matlab 显示，效果如下：

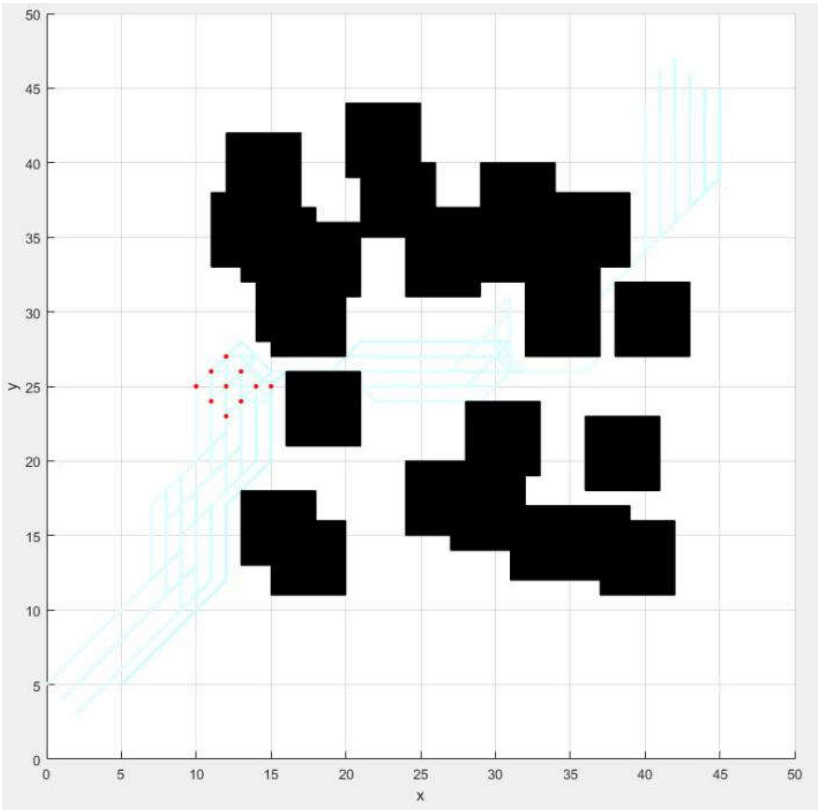

图 22　二维情景下多机协同仿真运行截图

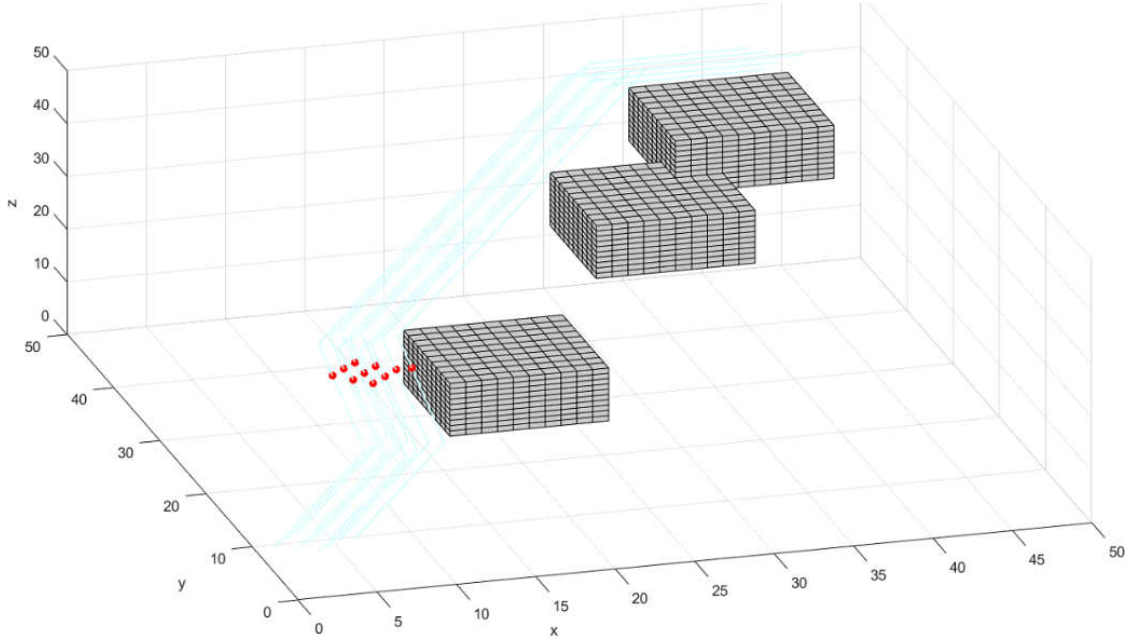

图 23　三维情景下多机协同仿真运行截图

结论

面对多机协同路径规划问题，首先对地图进行栅格化，并提出了 Inself 算法进行路径规划，将启发项与深搜结合，又利用视线导引法的自遍历二级路径优化，只需维护定长的最小优先队列，相较 BFS、Dijkstra 甚至 A*等算法有非常明显的时间优势以及空间优势，能够利用更小的内存消耗在更短时间内适当牺牲较小的路径长度为代价，得到一条较好的路径结果。经过分析，算法时间复杂度为 $O(n^2)$，优于 A*的 $O(n^2 \log n)$；后又经不同规模的二维、三维栅格随机地图测试，如在 600*600 栅格地图规模大量测试取平均后，平均运算时间为 A*方案的 8.26%、为 BFS 方案的 0.025%；平均开拓点数为 A*方案的 69.18%、BFS 方案的 1.11%；而相比于 BFS 的最短路径，Inself 的路径长度牺牲（平均增长率）为 0.26%。为了回归到模拟地图寻路，基于 Inself 的栅格部分，提出最大三角形法优化得到最终适合模拟地图的路径。

最后是多机协同策略，我们提出一种类似简笔画的可弹性压缩的编队方案，以 10 架无人机在二维及三维栅格地图为例，提供一种协同调度、编队飞行的方案，通过保持队形、非钝角坐标轴投影原则、牵引移动以及防掉队和更换长机策略，保证了多机的协同路径规划。最后通过 Matlab 将得到的路径结果以及多机协同实时坐标信息进行了演示。

参考文献

[1] 李井颂. 游戏中的智能路径搜索算法及其应用[D].昆明理工大学,2017.

[2] 房佳,杜震洪,张丰,曾志,刘仁义.应用于城市道路网的启发式深度优先有向搜索算法[J].浙江大学学报(理学版),2013,40(04):469-474.

[3] 杨鑫磊. 虚拟现实场景中的路径规划技术研究[D].哈尔滨工程大学,2016.

[4] 陈豪,李勇,罗靖迪.基于改进 A*算法优化的移动机器人路径规划研究[J].自动化与仪器仪表,2018(12):1-4.

[5] 赵晓,王铮,黄程侃,赵燕伟.基于改进 A*算法的移动机器人路径规划[J].机器人,2018,40(06):903-910.

[6] 张原,陈宇轩,魏璐璐.基于改进 A~*算法的 AGV 智能泊车算法[J].计算机系统应用,2019,28(01):216-221.