可支持跃层寻径的虚拟场景漫游方法

石敏[1]，魏育坤[1]，金相臣[1]，王素琴[1]，毛天露[2]
(1. 华北电力大学控制与计算机工程学院，北京 102206；2. 中国科学院计算技术研究所，北京 100190)

摘要：以三维虚拟场景中角色的自动寻径为研究背景，*提出了一种基于 A*算法的跃层寻径方法，采用改进的启发函数评估各个扩展节点的代价值以确定问题的最优解*。通过引入碰撞检测机制，对改进的 A*算法得到的路径进行修正以避免碰撞行为的发生，并对修正后路径的关键点进行插值，借助 Bézier 曲线构建平滑的导航路径。通过实验验证，在目标点可达的情况下，寻径结果能够良好地适配跃层的情形。

关键词：跃层寻径；A*算法；路径规划；Bézier 曲线；虚拟场景
中图分类号：TP391.9 文献标识码：A 文章编号：1004-731X (2019) 07-1358-09
DOI: 10.16182/j.issn1004731x.joss.18-VR0720

Virtual Scene Roaming Method Supporting Multi-layer Path Planning

Shi Min[1], *Wei Yukun*[1], *Jin Xiangchen*[1], *Wang Suqin*[1], *Mao Tianlu*[2]
(1. School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China;
2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Taking the automatic role pathfinding in three-dimensional virtual scene as the research background, *the author proposes a method of multi-layer pathfinding based on A-star algorithm, and uses advanced heuristic functions to evaluate the cost values of various expansion nodes to confirm the optimal solution to the problems.* By introducing collision detection mechanism, the navigation path generated by the improved A* algorithm is modified to avoid collision; and by interpolating the fixed path, the smooth navigation path is constructed using Bézier Curve. The experimental results show that the path finding results can well adapt to the multi-layer case when the target point is reachable.

Keywords: multi-layer path planning; A* algorithm; path planning; Bézier curve; virtual scene

引言

近年来，随着计算机技术的发展，城市的信息化建设得以迅速发展，而数字城市是城市信息化建设成果的重要体现。数字城市的重要功能之一便是虚拟寻径，虚拟寻径在一定程度上扩展了数字城市的功能、增强了数字城市的实用性。

在虚拟寻径中，常常会遇到路径规划的问题。传统数字城市中的虚拟寻径通常仅针对室外的情形，这类问题常常被简化为二维层面的平面寻径。而面对当代城市在对外宣传、城市规划、信息化管理等方面日益增长的需求，需要对三维层面的跃层寻径问题进行研究。考虑到适用于室内情形的跃层寻径具有层与层之间独立性强、复杂程度高等特点，故无法将平面寻径的方法直接应用于跃层寻径的情形。

此外，随着虚拟现实技术的逐渐发展及生产力

收稿日期：2018-07-10 修回日期：2018-10-29;
基金项目：国家十三五重点研发计划(2017YFC0804900), 装备预研重点基金(9140A15030115DZ08042);
作者简介：石敏(1975-)，女，山西，博士生，副教授，研究方向为虚拟现实；魏育坤(1996-)，男，内蒙，本科，研究方向为虚拟现实。

的不断提升，各种类型的虚拟现实设备不断涌现，如：VR 一体机、VR 眼镜等。这使得虚拟现实的应用不再局限于 PC 端，也意味着使用者可以从不同的角度认识虚拟世界。提高虚拟场景的真实感、可交互性及多设备兼容性，可有效弥补传统数字城市固有的不足，从而为数字城市的发展注入新的活力。

1 相关工作

目前，国内外对于路径规划问题已进行了大量的研究，现有的寻径算法主要可分为 2 大类：以 Dijkstra 算法为代表的解析算法及以 A*算法为代表的启发式算法。由于解析算法在复杂度、计算速度方面的局限性，启发式算法逐渐成为寻径算法研究的核心[1-6]。然而，利用启发式算法研究问题时往往局限于二维层面，未能考虑到跃层的情形。例如，文献[3]在对无人机进行航迹规划时采用了圆形节点扩展法动态搜索最佳扩展方向，弥补了传统节点扩展方法在方向、步长等方面的不足。在寻径过程中，文献[3]主要针对无人机在巡航阶段的航迹规划，即假设无人机在运行过程中高度保持不变，航迹规划被简化为二维平面的规划问题。然而，无人机的实际飞行过程还要经历起飞、爬升、着陆等阶段，处于不同运行阶段的无人机所处的环境差异明显，无法简单地视为二维平面规划问题；文献[4]针对移动机器人路径规划而设计的改进的 A*算法减少了路径冗余点，并缩短了路径长度及转折角度，以保证移动机器人在寻径过程中规划出一条无碰、安全的可行路径。但基于上述方法的移动机器人路径规划局限于二维层面，不同的应用场景是相互割裂的。

针对上述背景，为解决 A*算法在室内静态路网关系下的跃层寻径问题，本文提出了一种基于 A-Star (简称 A*)算法的寻径方法，以栅格化的方式构建地图模型，采用改进的启发函数评估各个扩展节点的代价值以确定问题的最优解。通过引入碰撞检测机制对改进 A*算法得到的路径进行修正以避免碰撞行为的发生，并对修正后路径的关键点进行插值，使用 Bézier 曲线构建平滑的导航路径，改进后的 A*算法能够良好地适配跃层的情形。基于本文方法的虚拟漫游支持自动漫游、平面及跃层寻径，并可部署于不同平台上，为城市导航、交通规划、智慧城市建设提供服务。

2 A*算法

2.1 环境模型描述

环境地图的建立是路径规划中非常重要的一环，地图建立的过程是将实际的环境信息抽象为地图模型的过程。以栅格为单位描述二维环境信息非常简单有效，故本文使用栅格法建立跃层寻径的环境模型。栅格法将环境分割成一系列具有相同尺寸的栅格，并将这些栅格分为 2 大类：可通过栅格和不可通过栅格，如图 1 右侧部分所示。

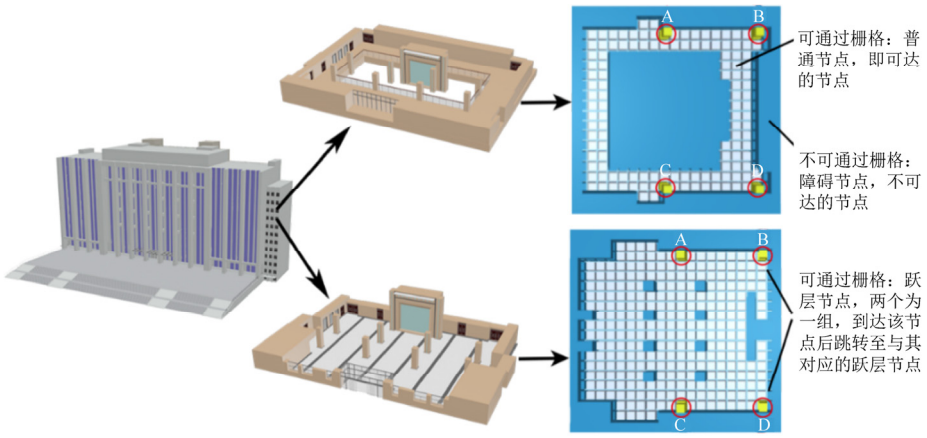

图 1 环境模型
Fig. 1 Environment model

对于可通过栅格，可进一步以节点的形式描述其状态：初始节点(originNode)、当前节点(currentNode)、目标节点(targetNode)、普通节点(normalNode)及障碍节点(obstacleNode)。originNode 为初始节点，是寻径的起点；currentNode 为当前节点，是寻径过程中当前搜索到的节点；targetNode 为目标节点，是寻径的终点；normalNode 为普通节点，表示该栅格可达。而对于不可通过栅格，则对应障碍节点(obstacleNode)，表示该栅格不可达。

2.2 传统的 A*算法

A*算法是静态路网中求解最短路径最有效的方法之一。A*算法采用了启发式搜索的方式，使用适当的估价函数确定搜索的方向，从起点开始向周围节点扩展并评估各个扩展节点的代价值，继而选取代价值最优的节点继续扩展，直至搜索到目标节点。在评估节点的代价值时，需要使用到估价函数 $f(n)$，其一般形式为：

$$f(n)=g(s,n)+h(n,e)$$

式中：s 为初始节点；n 为当前节点；e 为目标节点。$f(n)$为估价函数，为从寻径起点经过当前节点到达寻径终点的总代价，为了求得最短路径，需找到$f(n)$最小的路径。$f(n)$由 $g(s,n)$及 $h(n,e)$组成，$g(s,n)$是从初始节点出发到达当前节点的实际代价，$h(n,e)$是从当前节点到目标节点的估计代价，体现了搜索的启发式信息，被称为启发式函数。本文采用曼哈顿距离(Manhattan Distance)作为启发式函数度量两节点间的移动代价，其表示为：

$$h(n,e)=\left|X_i-X_j\right|+\left|Y_i-Y_j\right|$$

式中：(X_i,Y_i)，(X_j,Y_j)分别表示待评估节点 n_i 及其后继节点 n_j 的坐标。

当 $g(s,n)$=0 时，A*算法变成基于贪心策略的广度优先算法，虽可以快速寻得路径，但结果不一定是最优的；当 $h(n,e)$=0 时，A*算法则成为了 Dijkstra 算法，这样虽然可以保证找到最短路径(如果目标点可达)，但需要进行大量的运算，效率较低。若要在寻得最小代价路径的同时保证搜索效率，需将 $g(s,n)$及 $h(n,e)$结合。

A*算法在寻径过程中需不断地更新两个存放节点的表：开启列表(OpenList)与关闭列表(CloseList)。其中，OpenList 中存放着未展开遍历的节点，寻径过程中 A*算法会考察搜索方向上的每个节点，当搜索至某一节点时，该节点周围的节点会被添加至 OpenList，与此同时从中选取具有最小代价值的节点作为下一个扩展节点并加入 CloseList，当目标节点被加入 OpenList 则表示寻径成功；CloseList 中存放着已展开遍历的节点，用于最终的结果回溯。

可以看出，A*算法的搜索范围被局限于二维层面。如果直接将 A*算法应用于跃层情形，考虑到初始节点与目标节点不在同一个环境地图中，A*算法会遍历初始节点所在的环境地图中的所有节点直至没有新的节点可供搜索，最终在占用了大量计算资源的同时仍无法找到从初始节点到目标节点的可达路径。

3 基于改进 A*算法的跃层寻径

3.1 跃层节点

为解决 A*算法在室内静态路网关系下的跃层寻径问题，需要将三维的空间信息转换为二维平面信息处理。室内空间可通过不同的层区分，每一层均可抽象为二维环境信息并建立环境地图。

然而，各层之间是相互独立的，为了将各个二维环境地图联系起来，需要在各环境地图中引入跃层节点(transformNode)。跃层节点属于可通过栅格的一种，是不同层之间联系的纽带，它在环境地图中成组存在。寻径过程中若寻径的起点与终点不在同一层中，则通过跃层节点以一定的跃层代价跳转至与其相对应的跃层节点，代价值的计算将在下文中给出。

以图 1 为例，相邻的两个楼层通过跃层节点联系，每个楼层均预置有编号为 A、B、C、D 四个跃层节点，跃层节点之间按照编号两两对应。

3.2 可支持跃层的估价函数

寻径过程中，若当前节点与目标节点处于同一层，当前节点与目标节点的曼哈顿距离即为 $h(n,e)$ 的值；当前节点与目标节点不在同一层时，因路径当中须经过跃层节点，故需以跃层节点为分割点，将路径分为多段，再对每一段路径求解曼哈顿距离。对于涉及到跃层行为的层，计算相应跃层节点之间的跃层代价。若目标层数为 k，寻径起点所处层的层数为 i_0，$h(n,e)$的计算可表示为：

$$h(n,e)=\sum_{i=i_0}^{k}h_i(s_i,e_i)+\sum_{i=i_0}^{k-1}H_i$$

式中：$h(s_i,e_i)$为寻径过程中经过第 i 层的层内估算代价，H_i 为寻径过程中由第 i 层向下一层跃层的层间估算代价，即层与层之间的垂直距离。在虚拟场景中，能够到达目标节点的路径有很多条，故要对每一个可能的 $h_i(s_i,e_i)$值都进行求解，并选取最小的 $h(n,e)$值作为最终的估计代价。

在 A*算法中，$f(n)$一般以距离长度为单位，但考虑到跃层速度与平地行走的速度不同，以时间为单位可以更加准确地度量寻径过程的总代价。以时间为单位表示的启发函数可改写为：

$$h^*(n,e)=\sum_{i=i_0}^{k}\frac{\left|X_{s_i}-X_{e_i}\right|+\left|Y_{s_i}-Y_{e_i}\right|}{v}+\sum_{i=i_0}^{k-1}\frac{H_i}{v_i}$$

式中：S_i 表示第 i 层的寻径起点；e_i 表示第 i 层的寻径终点，(X_{s_i},Y_{s_i})、(X_{e_i},Y_{e_i}) 分别表示第 i 层的寻径起点及终点坐标，v 表示角色在平地上行走的速度，v_i 为由第 i 层向下一层跃层的跃层速度。

$g^*(s,n)$ 为以时间为衡量的实际消耗。用 $g^*(s,n)$ 表示从寻径起点到当前节点的实际消耗，$g^*(s,n+1)$ 表示从寻径起点到当前节点的后继节点的实际消耗，m 表示跃层代价。在栅格化的环境模型当中，角色可向栅格的正交方向(水平或垂直方向)或对角方向移动，也可以通过跃层节点到达其它层，每种移动行为均会产生相应的代价。若正交方向上相邻栅格的中心距离为 l，则对角方向上相邻栅格的中心距离为 $\sqrt{2}l$ 。

$g^*(s,n+1)$ 值的计算可表示为：

$$g^*(s,n+1)=\begin{cases}g^*(s,n)+\dfrac{l}{v}, & \text{正交方向移动}\\ g^*(s,n)+\dfrac{\sqrt{2}l}{v}, & \text{对角方向移动}\\ g^*(s,n)+\dfrac{H_i}{v_i}, & \text{通过跃层节点}\end{cases}$$

最终，将 $g^*(s,n)$ 与 $h^*(n,e)$ 合并，得到可支持跃层寻径的估价函数 $f^*(n)$：

$$f^*(n)=g^*(s,n)+h^*(n,e)$$

3.3 算法流程及效率分析

利用改进后的 A*算法进行跃层寻径的具体流程可表示为：

Step 1：初始化 OpenList 与 CloseList，并将 originNode 加入 OpenList；

Step 2：取出 OpenList 中 f 值最小的节点并放入 CloseList。若当前节点为 normalNode，则遍历当前节点周围尚未遍历的节点；若非 obstacleNode，则将其加入 OpenList 并计算出代价值；若当前节点为 transformNode，则将其对应的节点(如果对应节点已遍历则不加入)和周围未遍历过且不是 obstacleNode 的节点加入 OpenList，并计算出代价值。

Step 3：如果 OpenList 为空则结束寻径，未找到路径；

Step 4：将 OpenList 中的节点按照 f 值的大小从小到大排序；

Step 5：查看 OpenList 表中 f 值最小的节点，如果该节点是 targetNode 则结束寻径，并回溯得到完整的路径；否则，跳转至 Step2。

在目标节点可达的情况下，该方法的寻径结果总是最优的。此外，基于本文方法的跃层寻径亦支持平面寻径，只要将 transformNode 视为 normalNode 处理即可。

算法效率方面，假设起始楼层为 i_s，目标楼层为 i_e，寻径起点为 S，寻径终点为 E，每层拥有的跃层节点数为 $u_i(i_s\leqslant i\leqslant i_e)$，因要对每一条路径下的 $h_i(s_i,e_i)$ 值都进行求解，并选取最小的 $h(n,e)$ 值作为最终的估计代价，故需计算的路径条数为：

$\prod_{i=i_s}^{i_e} u_i$

可以看出，跃层路径规划的计算量随着场景规模及层数的增加而显著上升。

然而，寻径的过程可以拆分为 3 个部分：

(1) 从寻径起点 S 出发，搜索到达起始楼层 i_s 中的一个跃层节点的路径；

(2) 从 i_s 出发，通过跃层节点，到达目标楼层 i_e；

(3) 从 i_s 中的 1 个跃层节点出发，搜索到达寻径终点 E 的路径。

由于路网关系为静态的，寻径过程的第 2 部分可通过预先构建带权无向图提高搜索效率，图的各节点代表各组跃层节点的跃层代价，边则代表同一层跃层节点之间的层内移动代价。寻径过程由单纯的 A*算法搜索变为了首尾两段 A*算法搜索+图搜索，需求解的路径数减少为 $u_{i_s}u_{i_e}$ 条，且不随着跃层层数的增加而增加搜索路径条数。所以，借助图描述跃层节点关系可以有效地提高算法运行的效率。

4 无碰平滑路径生成

由于通过 A*寻径算法所得到的路径信息为离散的栅格点，需要将这些离散点转换为连续的路径以保证寻径过程的平滑过渡。引入了碰撞检测机制对改进的 A*算法得到的路径进行修正以避免碰撞行为的发生，并借助贝塞尔曲线(Bézier Curve)分段构建平滑的导航路径。

4.1 碰撞检测

改进 A*算法输出的原始路径在绕过障碍物时可朝着正交方向或斜角方向移动，若沿着斜角方向移动，很可能与场景中的障碍产生碰撞而影响导航行为的正常进行。图 2 给出了原始路径形成的折线路径。

针对上述问题，需对原始路径进行修正。路径修正的过程即是对可能发生的碰撞行为进行检测并处理的过程。碰撞检测以基于改进 A*算法的跃层寻径结果为输入，对结果中相邻的两个栅格点进行检测及处理，最终输出修正后的路径，其流程如图 3 所示。

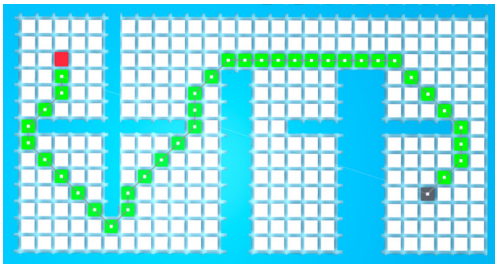

图 2 原始路径
Fig. 2 Original path

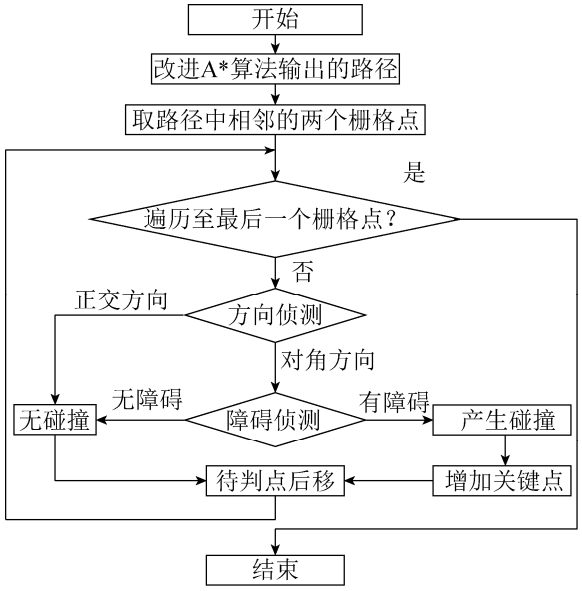

图 3 碰撞检测流程
Fig. 3 Sequence diagram of collision detection

部分操作的解释如下：

• 方向侦测：若沿着正交方向移动，则不会发生碰撞；若沿对角方向移动，需进行障碍侦测以判断是否可能产生碰撞；

• 障碍侦测：当前节点与下一个节点所在的四栅格区域内是否存在障碍节点，如存在障碍节点，则可能产生碰撞，需在障碍节点所在的四栅格的对角位置增加一个路径关键点。

借助上述碰撞检测方法，在保证改进 A*算法搜索方向多样性的同时有效地规避了碰撞行为的发生。图 4 给出了修正后的路径：

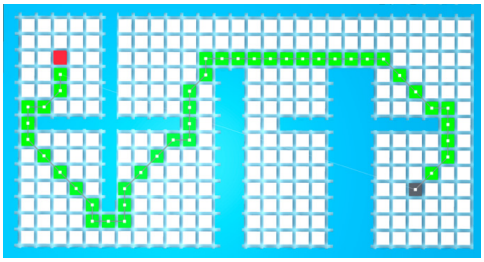

图 4　修正后的路径
Fig. 4　Fixed path

4.2 基于 Bézier 曲线的平滑导航路径

Bézier 曲线(Bézier Curve)是一种用于描述复杂几何图形的曲线构造方法，由于 Bézier 曲线具有连续性、凸包性、几何不变性等特征，近年来被广泛地应用于各种路径规划问题中[7]。

Bézier 曲线表示为：

$$P(t)=\sum_{i=0}^{n}P_iB_{i,n}(t),t\in[0,1]$$

式中：P_i 构成 Bézier 曲线的特征多边形，$B_{i,n}$ 是 n 次 Bernstein 基函数：

$$B_{i,n}(t)=C_n^it^i(1-t)^{n-i},t\in[0,n]$$

对于形成 Bézier 曲线的具有 n+1 个顶点的特征多边形中，第 1 个控制点 P_0 与最后 1 个控制点 P_n 总是通过曲线，其余顶点则影响着曲线的形状、阶次等。如直接将所有修正后路径点作为特征多边形的各顶点，生成的路径很可能再次与场景中的障碍发生碰撞，故应借助 Bézier 曲线，以修正后的路径为关键点进行插值，分段构建导航路径，并将其进行拼接以形成完整平滑的样条曲线。

若修正后的路径具有 n+1 个关键点：$P_0,P_1,P_2,\cdots,P_n$，则可划分出 n 个子段。在每 1 个分段区间$[P_i,P_{i+1}](i=0,1,\cdots,n)$ 分别构建 Bézier 曲线，第 i 条曲线应连接至第 i+1 条曲线。由于 3 次 Bézier 曲线在拼接处二阶连续，将其作为导航路径时其速度与加速度均是连续的，故选取具有 4 个控制点的 3 次 Bézier 曲线构建平滑的导航路径。

当曲线方程的次数为 3 次时，Bézier 曲线可表示为：

$$B(t)=(1-t)^3P_0+3(1-t)^2tP_1+3(1-t)t^2P_2+t^3P_3$$

$B(t)$的一阶、二阶导数分别表示为：

$$\begin{cases}B'(t)=-3(1-t)^2P_0+3(3t^2-4t+1)P_1+\\\qquad 3(2t-3t^2)P_2+3t^2P_3\\B''(t)=6(1-t)P_0+6(3t-2)P_1+6(1-3t)P_2+6tP_3\end{cases}$$

上述曲线的每 1 个分段区间内具有 4 个控制点，对于任意分段区间$[P_i,P_{i+1}]$，除去固有的 P_i、P_{i+1} 外，还需额外添加 $P_{i_{pre}}$、$P_{i_{post}}$ 两个控制节点，其中，$P_{i_{pre}}$ 为靠近 P_i 的控制点，$P_{i_{post}}$ 为靠近 P_{i+1} 的控制点。第 i 个分段区间$[P_i,P_{i+1}]$内的 Bézier 曲线方程表示为：

$$B_i(t)=(1-t)^3P_i+3(1-t)^2tP_{i_{pre}}+3(1-t)t^2P_{i_{post}}+t^3P_{i+1}$$

若要使得各分段区间构建的平滑曲线在连接点处二阶连续，需添加如下约束条件：

$$\begin{cases}B_i'(1)=B_{i+1}'(0)\\B_i'(1)=B_{i+1}'(0)\end{cases}$$

式中：$B_i'(t)$、$B_i''(t)$ 分别表示 $B_i(t)$ 的一阶、二阶导数：

$$\begin{cases}B_i'(t)=-3(1-t)^2P_i+3(3t^2-4t+1)P_{i_{pre}}+\\\qquad 3(2t-3t^2)P_{i_{post}}+3t^2P_{i+1}\\B_i'(t)=6(1-t)P_i+6(3t-2)P_{i_{pre}}+6(1-3t)P_{i_{post}}+6tP_{i+1}\end{cases}$$

最后，在整个样条曲线的首、尾两段添加以下约束，以确保样条曲线首尾两端的自然过渡：

$$\begin{cases}B_1'(0)=0\\B_n''(1)=0\end{cases}$$

利用上述表达式及约束条件，可求得各个分段区间$[P_i,P_{i+1}]$内 $P_{i_{pre}}$ 与 $P_{i_{post}}$ 的值，继而确定平滑的导航路径。由于上节中通过碰撞检测机制通过添加，且基于修正后路径的关键点构建的平滑导航路径通过了全部的关键点，故最终生成的导航路径是无碰且平滑的。图 5 给出了无碰平滑导航路径示例，与图 4 中的折线路径相比更加光顺。

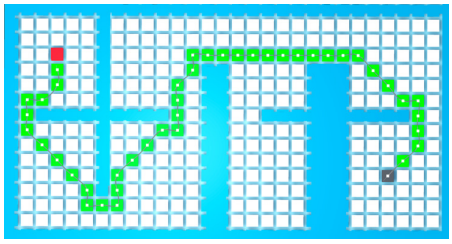

图 5　无碰平滑导航路径
Fig. 5　Smooth navigation path without collision

5 系统架构与实验结果

基于上述方法，本文以 3D Max 建模，Unity 为开发工具，构建出了可支持跃层寻径的虚拟场景。并将系统部署于搭载 Android 系统的“Deepoon M2”VR 一体机中。采用的开发及实验平台为 Intel(R) Core(TM) i7 CPU @ 2.5 GHz，16GB 内存，显卡型号为 AMD Radeon R9 M370X 2048 MB。

5.1 技术框架

本文技术框架主要由 3 部分组成：基于卫星地图及实地观测的三维虚拟场景的构建、交互逻辑的引入及多平台部署。如图 6 所示。

(1) 三维虚拟场景构建

要构建虚拟场景，需获取到场景内各个对象的参数信息并制作相应的三维模型。具体包括：

a. 参数信息获取

虚拟场景中构建有大量的建筑及其附属模型，要构建这些模型，首先需要获取模型的参数信息[8]，可采用卫星地图与实地观测相结合的方法获取模型的参数信息。

卫星地图中包含了模型的坐标及轮廓信息，利用这些信息可计算出指定对象的平面宽度参数。实地观测则可以获取到模型的高度参数及局部细节信息，拍摄的高清照片还可作为模型的贴图使用，弥补了卫星地图信息的缺失。将上述两种方法结合即可获取到三维模型的全部信息。

b. 三维模型构建

由于虚拟场景中的建筑模型种类繁多、形状各异，需要使用不同的方法构建相应的模型。利用模型的参数信息，可以将简单的几何模型变换成符合预期效果的建筑模型[9]。

模型构建完成后，需要为其赋予合适的材质贴图以增强其真实感，材质贴图的获取可以通过调制材质球、拍摄高清照片等方式获取。

(2) 交互逻辑的引入

为增强系统的实用性，需要进一步为其添加控制逻辑，大体分为：漫游逻辑、导航逻辑及交互逻辑。漫游逻辑允许使用者通过鼠标、键盘、虚拟准心为输入在系统中漫游；导航逻辑分为适用于室外情形的平面导航及适用于室内情形的跃层导航；交互逻辑包括使用者与系统之间的交互，如：主界面、菜单等。

(3) 多平台部署

虚拟场景支持部署在 PC 端及 VR 一体机上，PC 端的系统功能完备、界面友好、真实感强，VR 一体机端的系统摒弃了传统的交互方式，仅需简单的视角变换即可自由漫游于系统中。

5.2 跃层寻径

对于适用于室内情境的跃层寻径，基于改进的 A*算法进行寻径，并使用 Bézier 曲线构建出无碰平滑的导航路径。跃层寻径结果如图 7 所示。

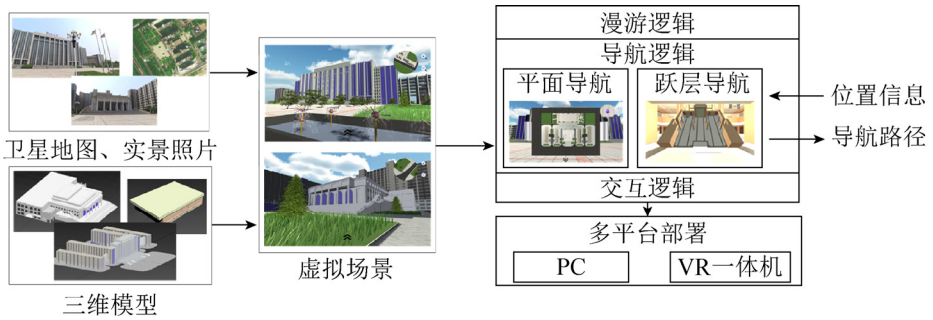

图 6 技术框架

Fig. 6 Technical framework

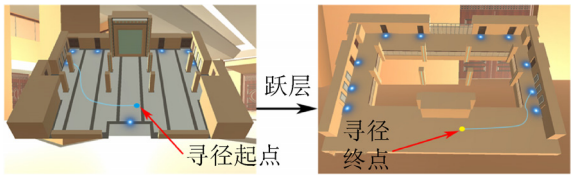

图 7　跃层寻径结果
Fig. 7　Result of multi-layer path planning

本文选取随机环境进行跃层寻径实验(障碍节点与跃层节点均随机放置于各层之中)，并与标准 A*算法寻径结果进行对照分析。实验的跃层层数为 3 层，环境信息被抽象化为 60*30 的栅格，测试 5 s 跃层寻径次数。对于跃层寻径算法，每层地图大小为 20×30，标准 A*算法的地图大小则为 60×30。表 1 给出了随机环境下标准 A*算法与跃层寻径算法的寻径次数，可以看出，在目标点可达的情况下，基于本文的寻径方法能够良好地适配跃层的情形，并且由于改进后的算法对各层的搜索范围进行了划分，当环境信息的总栅格数相同时，跃层寻径算法相较于标准 A*算法执行速度更快。

表 1　随机环境下标准 A*算法与跃层寻径算法的寻径次数比较
Tab. 1　Standard A* algorithm vs multi-layer path planning algorithm in random environment

随机环境	测试算法	路径长度	寻径次数/次			
			1	2	3	平均
A	标准 A*	13	98	98	97	98
	跃层寻径	19	636	633	634	634
B	标准 A*	12	96	95	89	93
	跃层寻径	20	574	570	567	570
C	标准 A*	17	79	79	80	79
	跃层寻径	14	997	954	997	983

6　结论

虚拟现实技术的发展为未来的生活带来了无限的可能。本文通过卫星地图与实地观测，构建了三维虚拟场景。将实际环境抽象为栅格地图，提出了一种基于改进的 A*算法的跃层寻径方法，并借助 Bézier 曲线构建无碰平滑的导航路径，解决了室内静态环境下的跃层寻径问题。

文本存在的不足之处体现在：

(1) 提出的跃层寻径方法仅支持静态环境下的跃层寻径，动态环境下的跃层寻径将是今后研究的方向。

(2) 跃层路径规划的计算量与场景中跃层节点的数量密切相关，若跃层节点过多则会影响算法的实时性，如何保证复杂场景下跃层寻径的高效进行有待进一步研究。

参考文献：

[1] 王防修，周康. 基于回溯法的 Dijkstra 算法改进及仿真[J]. 计算机仿真, 2013, 30(11): 352-355.
Wang Fangxiu, Zhou Kang. Improvement and Simulation of Dijkstra Algorithm Based on Backtracking Algorithm[J]. Computer Simulation, 2013, 30(11): 352-355.

[2] 郑辉. 一种基于谱分析的三维寻径启发式函数[D]. 杭州: 浙江大学, 2016.
Zheng Hui. A Spectral Based Heuristic Function for 3D path finding[D]. Hangzhou: Zhejiang University, 2016.

[3] 张帅，李学仁，张鹏，等. 基于改进 A*算法的无人机航迹规划[J]. 飞行力学, 2016, 34(3): 39-43.
Zhang Shuai, Li Xueren, Zhang Peng, et al. UAV path planning based on improved A* algorithm[J]. Flight Dynamics, 2016, 34(3): 39-43.

[4] 孙炜，吕云峰，唐宏伟，等. 基于一种改进 A*算法的移动机器人路径规划[J]. 湖南大学学报(自然科学版), 2017, 44(4): 94-101.
Sun Wei, Lü Yunfeng, Tang Hongwei, et al. Mobile Robot Path Planning Based on an Improved A* Algorithm[J]. Journal of Hunan University(Natural Science), 2017, 44(4): 94-101.

[5] 王道平，徐展，杨岑. 基于两阶段启发式算法的物流配送选址-路径问题研究[J]. 运筹与管理, 2017, 26(4): 70-75, 83.
Wang Daoping, Xu Zhan, Yang Cen. Study on Location-routing Problem of Logistics Distribution Based on Two-stage Heuristic Algorithm[J]. Operations Research and Management Science, 2017, 26(4): 70-75, 83.

[6] 孟珠李，焦俊，李郑涛，等. 基于 A*与 B 样条算法的农用机器人路径规划系统[J]. 安徽大学学报(自然科学版), 2018, 42(1): 45-53.
Meng Zhuli, Jiao Jun, Li Zhengtao, et al. Agricultural robot path planning system based on A* and B spline

algorithm[J]. Journal of Anhui University(Natural Science Edition), 2018, 42(1): 45-53.

[7] Yang L, Song D, Xiao J, et al. Generation of dynamically feasible and collision free trajectory by applying six-order Bezier curve and local optimal reshaping[C]// Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. IEEE. Hamburg, Germany: IEEE, 2015: 643-648.

[8] 汤圣君, 张叶廷, 许伟平, 等. 三维GIS中的参数化建模方法[J]. 武汉大学学报(信息科学版), 2014, 39(9): 1086-1090, 1097.

Tang Shengjun, Zhang Yeting, Xu Weipin, et al. Parametric Modeling Method in Three-Dimensional GIS[J]. Geomatics and Information Science of Wuhan University, 2014, 39(9): 1086-1090, 1097.

[9] 詹总谦, 林元培, 艾海滨. 基于 3ds Max 二次开发的建筑物快速三维重建[J]. 测绘通报, 2016(11): 22-25, 42.

Zhan Zongqian, Lin Yuanpei, Ai Haibin. Rapid 3D Reconstruction Based on Secondary Development of 3ds Max[J]. Bulletin of Surveying and Mapping, 2016(11): 22-25, 42.