基于路网的三维虚拟现实场景间接可视查询框架

孔德瀚 刘永山
(燕山大学信息科学与工程学院 河北 秦皇岛 066000)

摘 要 针对现有网络环境下的 B\S 模式下三维虚拟现实场景在加载过程中因加载大量的不可视的对象而引起资源消耗过高、加载速度过慢的问题,文中提出了一种基于路网的可视动态加载框架.该框架对三维虚拟现实场景内的静态对象和动态对象的可视查询分别进行处理.设计了框架内的路网、移动对象、静态对象可视关系表和动态对象可视关系表的数据模型,并给出了可视关系表的维护算法.对于场景内的静态对象的可视查询,给出基于静态对象可视关系表的静态可视查询框架及算法,避免了复杂的在线可视计算;对于场景内的动态对象的可视查询,则给出基于动态对象可视关系表的连续可视范围查询框架及算法.实验结果表明,在保持三维虚拟现实场景观测效果不变的情况下,可视动态加载框架能够大幅度降低虚拟现实场景中的静态和动态对象的加载数据量和更新数据量,降低对于网络带宽和客户端硬件的需求.

关键词 三维互联网;路网;连续可视范围查询;可视关系表
中图法分类号 TP392 **DOI 号** 10.11897/SP.J.1016.2016.02045

Visible Query Based on Road Network in Three-Dimension Scene

KONG De-Han LIU Yong-Shan
(*School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei* 066000)

Abstract To solve the problem that the existing loading modes in three-dimension scene cost too high resource because of too many invisible objects, a visible query framework based on road network is proposed. In the framework, the visible queries of static objects and dynamic objects are processed separately. The data structures of road network, mobile objects, static object visible relationship (SOVR) table and dynamic objects visible relationship (DOVR) table are designed and the maintenance algorithms of tables are given. The visibility relationships between static objects and segments in road network are calculated in offline preprocessing and saved in SOVR. Meanwhile the visibility relationships between segments in road network are saved in DOVR. The objects in the scene are indexed by segments in road network no matter static or dynamic. The visibility of the objects is linked with the static road. Based on the SOVR, static object visible query is transformed to a linear query to avoid the complexity of online calculation of visibility. Through the DOVR, a dynamic visible query named Continuous Visible Range Query (CVRQ) for mobile object is proposed. The visibility of the mobile objects to query point is transformed to the visibility of the segments which are visible to the segment which query point belongs. The experimental results showed that while maintaining the picture for observer unchanged, visible query framework reduce more than 45% of the static object, 72% of the dynamic objects and 70% of the update data.

Keywords three-dimension internet; road network; continuous visible range query; visible relationship table

收稿日期:2015-07-22;在线出版日期:2016-01-19.本课题得到国家自然科学基金(61572422)、河北省自然科学基金(F2013203187)资助.
孔德瀚,男,1985 年生,博士研究生,主要研究方向为空间数据库、移动对象数据库.E-mail: kdh0312@163.com. **刘永山**,男,1963 年生,博士,教授,博士生导师,主要研究领域为空间数据库.

1 引 言

随着“三维互联网”时代的来临,网络环境下大规模三维虚拟现实场景得到越来越广泛的应用,如虚拟旅游、虚拟教育、虚拟城市等.但是三维模型本身极为复杂,其在大规模场景或高精度模型等方面的应用受到了极大的限制.例如,现有的三维游戏其模型文件动辄十几 GB 乃至几十 GB,移动客户端很难满足其硬件需求.在大规模的 B\S 结构的三维虚拟现实场景中,客户端所需要加载的模型数量和质量的需求很难以整体加载的静态方式满足.因此,动态的空间查询技术广泛应用于“三维互联网”应用中.但是,在三维虚拟现实场景中,用户可能只对可视的对象感兴趣,不可视的对象对于观察者而言就是无效数据.因此基于可视性的三维虚拟现实场景动态加载框架也就成为在科研和商业领域都具有极高价值的热点研究.

在三维场景的加载过程中,现在通用的加载方式是以观察者为圆心,整体加载一定半径内的所有静态或动态对象.在这一过程中不考虑这些对象是否可视的问题,对他们的可视性计算是在读入客户端内存后进行的.这种先传输后计算的模式对于网络带宽和客户端硬件的要求都很高,不符合现在移动式互联网终端和云计算发展的趋势.因此,先计算后传输的方式成为网络环境下三维虚拟现实场景在低性能客户端加载的必然选择.在虚拟现实场景中,对客户端的化身所能看到的对象进行可视性查询后生成需加载对象列表,并结合 LOD 技术和渐进式传输技术向客户端传输模型,可以有效地减少单位时间内需要传输的模型数据量,降低对于网络和硬件的压力.

在三维虚拟现实场景中,决定一个对象是否需要被传输的关键就是这个对象是否是用户所能看到的.这就涉及到了对三维空间对象的可视性查询.在三维虚拟现实场景中,涉及到静态对象和动态对象这两类对象的可视性查询.这两种可视性查询随着用户的移动就会转变为对静态对象的连续可视查询和对动态对象的连续可视查询.这两种查询的复杂性会随着场景内三维空间对象数量的增加和查询区间的增加而增加,效率急剧恶化.一个 B\S 模式下的大规模三维虚拟现实场景,必然拥有众多的用户所代表的查询线程,采用现有的可视查询框架必然无法满足需求.

根据对以往的三维虚拟现实场景的应用观察,发现用户的移动轨迹并不像最初所设想的那样是在欧式空间内的自由移动,而是依托于路网的,其行动轨迹受限于路网.因此对于用户的可视查询请求,完全可以以其所在路网路段替换其真实移动轨迹进行可视查询计算.而路段和静态对象都是固定数据,这样就存在离线计算的可能.而对于动态对象,由于其移动轨迹也都限制于路网路段之中,因此完全可以以路段间的固态可视性替代移动对象间的动态可视性,从而有了离线计算的可能.基于这样的想法,本文提出一种基于路网的三维场景可视查询加载框架,将静态对象的可视查询与动态对象的可视查询分离处理、将可视计算与查询分离处理,实现对于三维场景的动态可视性加载.

本文的主要贡献如下:

(1)提出三维虚拟场景基于可视性查询的动态加载框架,以路网路段作为数据存储、查询结构的核心,以此为基础设计了包括离线预计算和在线查询处理算法两部分的可视性查询处理策略,实现对于静态对象和动态对象的间接可视查询,首次针对大规模三维场景的实时可视性查询加载问题提出可行的解决方案;

(2)提出三维场景内的静态对象与路网间的静态对象可视关系表,实现对于静态对象可视查询的离线预处理,避免现有可视查询模式的计算时间过长的缺陷,满足对于三维场景内静态对象的实时可视查询的时间要求;

(3)通过动态对象可视关系表建立路网路段间的可视关系,将三维场景内的移动对象间的可视查询转化为针对可视路段的范围查询,避免现有可视查询模式中计算开销随移动对象数量增加而急剧增加的缺陷.

本文第 2 节介绍可视性查询和移动对象路网连续查询工作的相关研究进展;第 3 节给出三维场景可视查询框架及数据模型;第 4 节给出静态对象可视关系表和动态对象可视关系表的原理、维护算法;第 5 节给出基于可视关系表的可视查询算法,并分别给出对静态对象和动态对象的可视查询算法;第 6 节利用实验对本文所提的框架、模型和算法进行验证;第 7 节总结全文并提出下一步的研究方向.

2 相关工作

对三维虚拟现实场景进行基于可视查询的动态

加载，主要的难点在于进行快速、准确的三维空间对象的可视查询. 由于三维虚拟现实场景的特殊性，因此其主要面对的就是静态对象的连续可视查询和移动对象的连续可视查询.

最初，三维可视计算方法在仿真领域得到广泛应用，如 JANUS 通视性算法[1]、DYNTACS 通视性算法[2]和 ModSAF 通视性算法[3]等，以及后期的改进算法如 Bresenham 通视性算法等. 这些算法都是在网格的基础上进行可视性计算，要求三维场景是一个整体，面对独立的三维空间对象模型时并不适用. 对于独立存储的空间对象的静态可视查询研究仍然集中于二维空间中. 早在 2004 年，Zhang 等人就在文献[4]中，针对空间中存在障碍的条件下的查询，给出了第一个综合性的处理方法，对于障碍空间下的范围查询、最近邻查询、最近对查询分别给出了解决算法. 到了 2007 年，Nutanong 等人在文献[5]中定义了一种不需要预先计算可视区域的 VkNN 查询，利用最近可视距离(MinViDist)进行可视最近邻查找. Gao 等人在可视反 k 近邻查询[6]和连续可视 k 近邻查询[7]上做了大量的工作. 在文献[6]中，提出 VRkNN 查询，将可视性引入反向 k 近邻查询中. 在 2011 年的文献[7]中，对连续可视 k 近邻进行研究，将连续查询的查询对象抽象为一条查询线段，在一次查询中完成整个 CVkNN 查询. Lu 等人在文献[8]中应用 Voronoi 图对潜在可视对象进行存储和查询，提高了查询的速度；但是在实际的三维虚拟现实场景中静态对象的连续可视查询中，随着查询区间的增大，参与可视计算的对象数量增加，可视查询的性能急剧恶化. 而且当出现查询轨迹接近的查询点，如同一条道路上顺序前进的用户时，近似的连续可视查询进程也浪费了大量的资源. 这些问题使得三维虚拟现实场景的应用规模和场景内用户数量受到极大的限制.

在对移动对象的连续查询研究方面，现有的研究主要分为欧式空间内的查询和网络空间内的查询两个分支. 在欧式空间的移动对象可视性查询方面，2007 年 Kusakari 等人在文献[9]中就进行了研究，给出了两种新的索引结构. 王艳秋等人在 2010 年的文献[10]中，给出了一种基于概率的不确定对象的可视 k 近邻查询机制. 2011 年 Guo 等人在文献[11]中定义了一种新的空间查询 DBS 并给出了 DBS 算法，在欧式空间及路网空间内都能给出更好的结果. 2012 年，Cheema 等人在文献[12]中提出在欧式空间和路网空间中的连续反向 k 近邻查询，该方法不仅提高了计算速度还降低了计算消耗. 2010 年 Güting 等人在文献[13]中对基于路径的移动对象 k 近邻查询做了研究，利用路径来简化查询在不同时间段内移动对象的 K 最近邻查询；2014 年，王艳秋等人在文献[14]中提出一种连续反 k 近邻查询，利用一种过滤和提纯的框架来处理移动对象的连续查询问题. 而在网络空间的查询方面，Papadias 等人早在 2003 年就在文献[15]中对移动对象在路网环境下的查询进行了详尽的研究，给出了范围查询、最近邻查询和 K 近邻查询的算法，但是对于路网内移动对象的可视性则完全没有涉及，将该方法应用于网络虚拟场景中时会产生大量的不可视的无效数据；赵亮等人在文献[16]中对于路网中移动对象的连续 K 近邻查询进行了研究，提出一种多线程的处理框架，将连续查询转化为周期性查询，并将数据更新与查询处理分开进行，该方法具有良好的路网扩展性能，但是同样会由于对路网内对象的可视性没有涉及而产生大量无效数据，本文借鉴了其数据更新的模式，并通过定义路网间可视性实现了对路网内动态对象的可视性查询. Jeung 等人在文献[17]中利用轨迹预测在路网中进行范围查询，该方法预测移动对象的行进轨迹，并利用该轨迹对查询结果进行筛选和排序. 对轨迹预测的准确率影响着查询的效率，而在人机交互性较高的虚拟场景内，动态对象的轨迹基本不可预测，因此该方法不可行. 2014 年 Lin 和 Wu 在文献[18]中分析路网中移动对象的运动关系，利用路径提出一种范围查询机制，该方法利用轨迹与移动速度确定最小更新时间间隔来降低查询的频率，同样当虚拟场景内的移动对象的运动轨迹不确定、不可预知时，该方法就失效了，而且同样，在该文章内对于路网内动态对象的可视性也没有涉及. 事实上，现有的对于移动对象的可视查询，都没有考虑过路网内移动对象间的可视性问题. 因此，当前对于三维虚拟现实场景内移动对象的可视查询问题存在明显的不足：在欧式空间内，移动对象的可视查询计算过于复杂且耗时，无法满足实时性的需求；而在路网空间中，各种扩张查询的方式使得查询的时间要求可以得到满足，但却完全没有考虑移动对象间的可视性关系，得到的无效数据过多，对带宽和客户端的硬件造成的压力过大，也不满足实际需求.

此前的研究认为观察者在三维场景中的移动是在欧式空间内进行的，所以采用了简单的范围查询来获取需加载的对象，因此产生了大量的无效数据. 采用可视查询对数据进行过滤时又因为计算复杂的

原因而需要限制移动对象的数量,使得三维场景的应用规模和范围受到极大的限制.在实际的 B\S 模式的三维虚拟现实场景的应用中可以观察到这样一个现象,观察者的行动受到路网的限制.因此可以采用用户所在的路段来代替用户实际的查询区间,使对静态对象的可视查询可以在离线情况下计算.而对于移动对象,路段上的移动对象间的可视关系也可以用路段间的可视关系近似表示.这样就将移动对象间的实时可视计算转化为路段间的可视关系这样的静态对象的可视关系计算,也可以在离线阶段进行.

3 基于路网的间接查询加载框架与模型

3.1 动态加载框架

因为在三维虚拟场景中,用户所控制的观察者必然是一个动态对象,而动态对象的运动轨迹又被路网所限制,因此将静态对象和动态对象以路网路段为主键分别建立索引是可行的.在现有的商业开源软件中,这一结构也得到了应用.但是现有的结构中距离因素是建立索引的条件,在加载时只考虑对象与路段间的距离而不考虑加载对象是否可视的问题.这样在生成索引时的速度较快,维护也较为方便.但缺点是在加载时会产生大量的无效数据,浪费带宽和客户端硬件资源.

因此提出一种基于可视性的动态加载框架,以路网路段代替动态对象的实际位置进行可视查询和加载.对于静态对象,可以计算固定的路段与固定的静态对象之间的可视关系,给出每个路段可视的静态对象列表;对于动态对象,因为所有的动态对象都是在路网路段上的,所以互相不可视的路段上的动态对象也一定互相不可视,所以计算固定的路段间的可视关系也可以代替实际的动态对象间的可视关系.这一框架的好处是,路网路段和静态对象都是固定的数据,因此其可视计算可以离线进行,从而避免复杂的在线计算.同时,对于三维虚拟现实场景内大量用户的并发可视查询请求可以根据路段属性进行线性查询的批处理,而不是进行近似的在线可视计算,这样极大减轻了服务器的压力.

根据这一思路在三维场景的可视性查询中,提出一种基于路网的可视动态加载框架,如图 1 所示.该框架的主要思想是,以路网作为索引,将三维场景内的对象分为静态对象和动态对象两部分分别进行处理;建立路网路段与对象间的可视性关系,将可视性计算与在线查询分离进行.

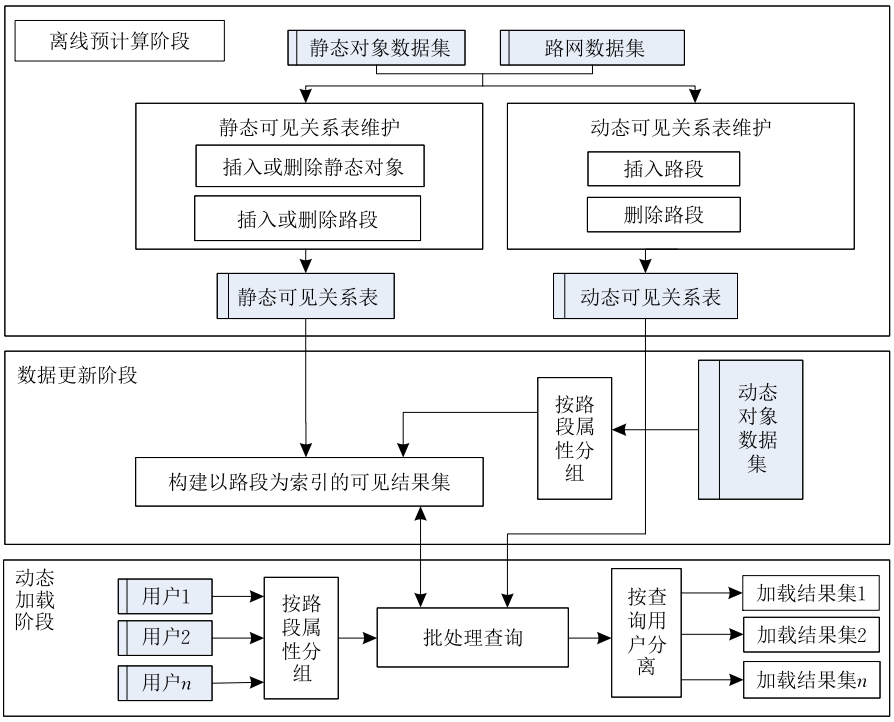


图 1 基于路网的可视动态加载框架

在现有的 GIS 系统中,基于路网路段的存储和动态加载的确有应用,但是主要仍然是基于距离条件的查询和加载,而在本文所提框架中,是针对三维虚拟现实场景的动态加载,可见性是首要的加载条

件. 为了避免复杂的三维可视计算，尤其是动态对象的可视计算耗时过长的问题，提出了静态对象可视关系表和动态对象可视关系表，将计算过程放在离线预计算阶段进行，从而满足了在线查询的时间要求.

在预处理阶段对可视性进行计算. 对于静态对象，建立其与路网中的路段之间的可视对应关系；对于动态对象，则通过计算路网中路段间的可视关系，并根据这一关系对移动对象进行可视范围查询.

在数据更新阶段，为每个路段建立一个列表保存该路段上的动态对象，对动态对象进行维护和更新. 生成一个以路段为索引的可视数据表.

在动态加载阶段，则根据用户所处的位置信息在可视数据表中进行线性查询，获取需要加载的静态对象和动态对象列表后进行数据的传输和加载. 不必进行在线的可视计算以节省时间.

下面将分别说明各个阶段在框架中的作用、处理方法和其中使用的数据结构.

3.2　数据模型

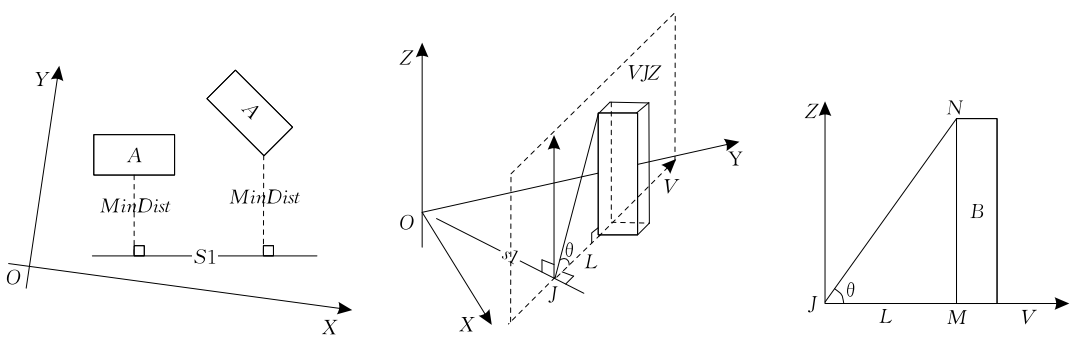
在对数据的预处理阶段，需要建立两个可视关系表，分别是路网与静态对象间的静态对象可视关系表和路网中路段之间的动态对象可视关系表. 通过这两个关系表，分别将三维场景内的静态对象和动态对象与路网路段间的可视关系固化. 将复杂的可视计算在预处理阶段完成，加快在线查询的反应速度. 因此下面先给出在本文中所涉及到的定义及相应的数据结构，以便更好地理解本文的思路.

定义 1(道路网).　一个道路网路可以表示为一个无向图 $G=(N,E)$，其中，N 是顶点(结点)的集合；E 是路段边的集合.

定义 2(静态对象).　一个静态对象在三维空间中可以表示为一个方向包围盒 OBB，$OBB=(p_1,p_2,\theta)$，其中 $p_1(x_{\min},y_{\min},z_{\min})$ 和 $p_2(x_{\max},y_{\max},z_{\max})$ 分别是包围盒的对称顶角，θ 为该 OBB 在水平面与坐标轴的夹角.

定义 3(动态对象).　在到路网中的一个动态对象位于网路对应的无向图中的路段边 $e(e\in E)$ 上，移动对象在图中的位置可以表示为一个四元组 $(n_1,n_2,speed,e)$，其中，n_1 和 n_2 是移动对象所在路段边的两个结点，$speed$ 是当前运动速度，$e(e\in E)$ 是移动对象当前所在的路段边.

定义 4(最小距离).　设路段在 XOY 平面上投影为 S_1，静态对象 O 在 XOY 平面上投影的 MBR 为 A，则最小距离 $MinDist$ 为 A 到 S_1 的最小欧氏距离，如图 2(a)所示.



(a) 最小距离$MinDist$　　(b) $MinDist$的垂直面　　(c) 投影角度θ

图 2　最小距离 $MinDist$ 和投影角度 θ

定义 5(投影角度).　设路段在 XOY 平面上的投影为 S_1，静态对象 O 到路段 S_1 的最小距离为线段 L，L 与 S_1 垂直相交于点 J，垂直投影面 VJZ 垂直平面 XOY 于 L，即 $\{VJZ\mid(VJZ\perp XOY)\cap(VJZ\cap XOY=L)\}$，如图 2(b)所示. 静态对象 O 在平面 VJZ 上的投影的 MBR 为 B，则投影角度 θ 为 $\angle MJN$，如图 2(c)所示.

定义 6(路段可视距离).　设两个路段 n_1 和 n_2，则路段可视距离 D：

$$D=\begin{cases}length(n_1,n_2), & n_1\text{与}n_2\text{互相可见}\\ \infty, & n_1\text{与}n_2\text{拥有共同结点}\end{cases},$$

其中 $length(n_1,n_2)$ 即为 n_1 和 n_2 的最小欧式距离.

根据静态对象和动态对象的定义，可以给出对象与路网的路段间的可视性的定义.

定义 7(可视性).　给定一个数据集 P、一个障碍集 O 和查询点 q，p 和 q 是可视的，当且仅当存在 p 和 q 之间的连线不穿过 O 的内部，即任意 $o\in O$，$[p,q]\cap o=\varnothing$.

由于在三维虚拟场景中，查询点必然是一个移动对象，因此根据定义 3，每个查询点必然在路网中的一个路段上，所以可以得出以下两个定理.

定理 1.　对于某查询点 q 可视的静态对象，则

一定对该查询点所在路段 n 可视;反之,当查询点 q 在路段 n 上任意点间移动时,对于 n 可视的静态对象也必将对 q 可视.

证明. 设静态对象 o_1 对 q 所见,则存在线 s 使 $[q,o_1]\cap O=\varnothing$,使 o_1 对 q 所在路段 n 可视;当 o_1 对路段 n 可视时,存在 o_1 与 n 之间的连线 s 交 n 于 a 点,因为 q 在 n 上任意行动,则 q 必通过 a 点,使 o_1 对 n 可视. 证毕.

定理 2. 当两个路段上的移动对象 $o_1(o_1\in n_1)$ 与 $o_2(o_2\in n_2)$ 互相可视时. n_1 和 n_2 同样互相可视;反之,n_1 和 n_2 两个路段互相可视,则在两个路段上任意移动的动态对象必然互相可视.

证明. o_1 与 o_2 可视,则存在一条线 o_1o_2 使 $[n_1,n_2]\cap O=\varnothing$,则 n_1 与 n_2 互相可视. 当 n_1 与 n_2 互相可视时,则至少存在一点 $a(a\in n_1)$ 和点 $b(b\in n_2)$,使 $ab\cap O=\varnothing$. 因为动态对象 $o_1(o_1\in n_1)$ 与 $o_2(o_2\in n_2)$ 在 n_1 与 n_2 上任意行动,则必有一时刻使 o_1o_2 与 ab 重合,则 o_1o_2 互相可视. 证毕.

根据定理 1,可以为路网中的每一个路段建立一个静态对象可视关系表(Static Object Visible Relationship,SOVR)记录对于该路段可视的静态对象. 从而将静态对象的可视关系通过路网固化下来,不必再进行在线的可视计算. 该表是一个 Hash 表,存储的内容包括:(1)静态对象 id;(2)静态对象与路段间的最小距离;(3)静态对象与路段间的投影角度. 通过查询该表,可以避免进行在线的可视计算并对在同一路段的多个查询进行批处理以降低服务器压力. 虽然降低了可视查询的精度,额外增加了一些静态对象结果,但是避免了在线计算,满足了可视查询的时间需求.

根据定理 2,可以为路网中每一个路段建立一个动态对象可视关系表(Dynamic Object Visible Relationship,DOVR)来描述各个路段间的可视关系,通过该表将移动对象间的复杂的可视计算,转化为对可视路段的范围查询. 该表包括如下信息:(1)路段 id;(2)路段间的可视距离 D. 通过动态对象可视关系表对移动对象进行可视查询,虽然精度有所下降,增加了一些无效数据,但却可以满足可视查询的时间需求.

3.3 数据存储与更新

对于三维场景中的相对不变的静态对象和路网数据,在离线预处理阶段进行复杂的可视计算,建立静态对象可视关系表及动态对象可视关系表,可以避免在线查询阶段进行长时间的计算. 而对于路网中的动态对象,首先按照路网路段为索引为每一个路段维护一个列表保存指向移动对象的指针. 当移动对象所在路段发生变化时,对该列表进行更新维护,确保在每一个路段的移动对象数据的实时性,以便于在线查询处理.

因此,对于一个三维场景,如路网中有 n 个路段,m 个静态对象,则需要建立 n 个静态对象可视关系表,每个表的长度较为稳定. 设建筑密度为 k,如在城市环境中,建筑密度 k 会较大,则单个路段的可视范围较小;而在野外环境中,虽然视野较为开阔,但是建筑密度 k 较小. 因此在一般情况下,单个路段的静态对象可视关系表的长度比较稳定.

而对于动态对象的存储,为每一个路段维护一个列表保存指向动态对象的指针,因此需要建立 n 个列表,列表长度与动态对象的数量及密度有关.

3.4 在线查询阶段

在在线查询阶段,首先获取查询点所在路段 id. 根据路段 id 分别对需加载的可视的静态对象和动态对象进行查询,获取结果集后再进行加载. 借助静态对象可视关系表进行的线性查询可以获取对于该路段可见的静态对象集;在动态对象可视关系表中的线性查询可以获取与查询点所在路段可见的路段的 id,并根据路段 id 检索移动对象,获取可视动态对象结果集. 将这两个结果集汇总便可以不进行复杂的可视计算而得出对于查询点而言可视的静态和动态对象,从而实现对于大规模三维虚拟场景的动态加载.

3.5 性能分析

在基于路网的可视性查询框架中,通过提取并抽象查询点的位置属性,将查询点个体的动态的可视性计算转化为所在路段的静态的可视性计算. 以静态的路段作为索引,将所有的数据的可视性计算放在离线预处理阶段,避免了在线的复杂计算. 将在线查询转变为一次线性的 Hash 查询. 原有的可视性查询模式,n 个不同位置的查询点需要进行 n 次复杂的可视性计算,需要对静态对象数据集及动态对象数据集进行重复的读取和计算. 而在本文提出的框架中,只需要在建立、维护可视关系表时进行一次可视性计算,n 个查询点就可以按照位置属性划分成 $m(m<n)$ 组,进行 m 次复杂度为 $O(1)$ 的线性查询,在查询阶段不需要进行可视性计算,也就不需要进行频繁的数据读取和计算,降低了查询数量对于服务器的压力,加快了在线查询的反应速度. 在面对大规模的三维虚拟场景时,基于路网的可视性查

询框架明显具有更好的性能.

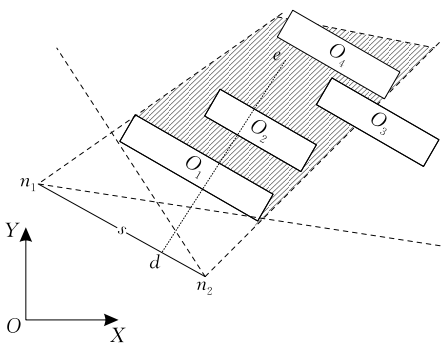
当三维虚拟场景的数据规模增加时，静态可见关系表和动态可见关系表的查询效率并不会发生显著地下降. 这是因为对于静态对象的可见性受到建筑密度和可视距离的约束，因此静态可见关系表的长度是存在上限的. 而动态对象需要再路网路段中行动，因此对于任意路段，对于动态对象的容纳数量也是有限的，而在线查询是线性查询，因此查询效率不会发生较大的改变.

4　基于路网的间接可视关系表

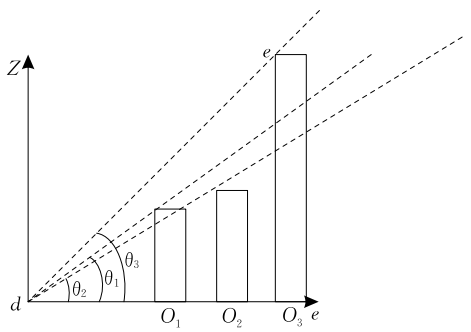
按照前文给出的定义和定理，可以通过构建以路网路段为检索核心的间接可视关系表. 将原有可视查询中最为耗时的可视计算部分剥离出来，放在离线计算阶段，以保证在线查询的反应速度. 在这一过程中，利用静态对象可视关系表固化静态对象与路段之间的可视关系，这样对于静态对象的可视查询就变成了对于该表的线性查询；利用动态对象可视关系表固化路段间的可视关系，这样对于动态对象的可视查询就变成获取可视路段后按路段范围查询的间接可视查询，同样避免了复杂的实时可视计算. 在本节中，将给出静态对象可视关系表和动态对象可视关系表的构建和维护方法.

4.1　三维场景中的可视计算

首先给出在三维场景中，对于三维空间对象进行可视计算的基本方法. 静态对象对路段的可视性计算与现有的二维场景不同，涉及到了高度的影响. 即对于三维空间数据，不仅要考虑到剪切角度的问题，也要考虑到投影角度的问题. 如图 3 所示，静态对象 O_1 和 O_2，O_2 与 O_3 都在 O_1 形成的不可视区域内，但是对应的投影角度 $\theta_2<\theta_1<\theta_3$，即 $\overline{de}\cap O_1=\varnothing$，所以 O_3 对于路段 s 可视. 因此可以得出结论：对于路段 s，静态对象 O_1，O_3 是可视的，O_2 是不可视的.



(a) xoy平面



(b) 投影角度对可见性的影响

图 3　三维空间内的可视性计算

在进行可视关系计算时，按照“远的对象不能影响近的对象”和“低的对象不能影响高的对象”的原则，按照最小距离和投影角度的顺序进行计算. 如果一个对象(无论是静态对象还是另一个路段)对路段是可视的，则将其加入可视关系表的同时加入障碍集对下一对象进行判断计算；一个对路段不可视的对象，必然已经被遮挡，因此不需要加入障碍集.

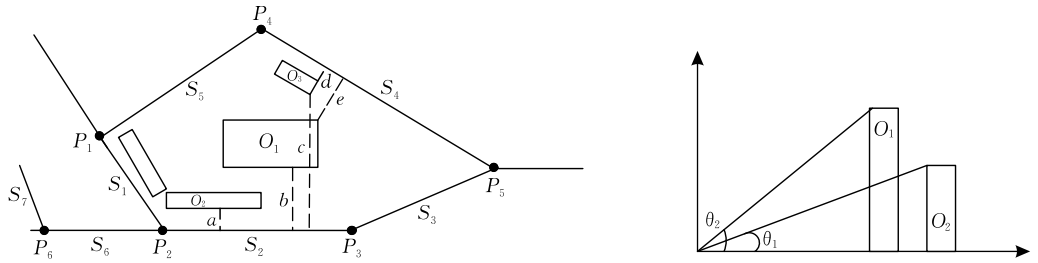
4.2　静态对象可视关系表构建与维护

静态对象可视关系表描述的是静态对象与路网路段之间的可视关系，即在某一路段上所能看到的静态对象. 根据这一关系表，在获取查询点所在路段 id 的情况下不必进行可视计算即可获取对其可见的静态对象. 在新增或删除静态对象时，需要对该表进行维护. 如果路段发生变化，也需要进行相应的维护.

4.2.1　插入静态对象

显而易见的是，一个新的对象加入到已有的地图中时，必然会对附近对象的可视关系产生影响，但该影响范围将只局限在离它最近的一组节点范围内. 如图 4(a)所示，在静态对象 O_1 加入到地图中后，其对节点 P_1，P_2，P_3，P_4，P_5 及相应的路段 S_1，S_2，S_3，S_4，S_5，S_6 产生了影响，而对节点 P_6 及对应的路段 S_7 并不产生影响.

因此，在插入一个新的静态对象时，应该首先找出其可能影响的路段列表集 $S_{set}=\{S_1,S_2,\cdots,S_n\}$，然后根据该列表计算新的静态对象对于各个路段 $S_1,S_2,\cdots,S_n$ 的可视关系，然后将该静态对象按照可视关系分别加入到 SOVR 中. 在插入新的数据后，需要更新插入数据对于原有数据的影响. 如图 4(a)中新插入对象 O_3 对于路段 S_2 的最小距离 c 大于原有对象 O_1 对于路段 S_2 的最小距离 b，或图 4(b)中新插入对象 O_2 对于路段 S_2 的投影角度 θ_1 小于原有对象 O_1 对路段 S_2 的投影角度 θ_2，则原有对象不受影



(a) 基于最小路径*MinDist*的影响 (b) 基于投影角θ的影响

图 4 插入静态对象对可视关系表的影响

响. 否则要重新计算该对象的可视关系. 在插入的过程中,如果由于 O_1 的原因使 O_3 变得相对于 S_2 不可视,则需要将 S_2 邻接表中的相应数据删除. 按照这样的步骤将所有受到影响的路段对应的邻接表的更新后,完成新的静态对象的插入处理过程. 对于每一条受到影响的路段 S,其插入算法如算法 1 描述.

算法 1. InsertStaticVR().

输入: InsertObject *IO*, effectedSegment *S*

输出: StaticVisibleRelation *SOVR*

1. get *MinDist* and θ from *IO*
2. get VisibleRelation from *S*
3. FOR each *element* in *VR*
4. IF(IO.MinDist<element.MinDist and IO.θ<element.θ)
5. THEN insert IO into VisibleRelation
6. recalculate the *VR* between *element* and *Segment*
7. IF Visible (Object,Segment)=true
8. THEN set *T* in *SOVR* as true
9. ELSE set *T* in *SOVR* as false
10. return *SOVR*

插入算法,受影响的应为离插入点距离最近的一组结点及路段. 这一组结点及路段应能围绕插入点行成包围结构. 在第 2 行对受到影响的静态对象进行更新静态对象可视关系表时,可以利用欧式空间内的反 k 近邻查询来确定受影响的点的集合,然后利用点集合确定受影响的路段的集合. 第 6 行对可视关系的计算,则是利用三维投影进行几何计算.

4.2.2 删除静态对象

当一个静态对象被从地图中删除时,不仅仅要删除对象本身,还要对其他受到该对象影响的静态对象状态进行更新. 如图 4 中对象 O_1 被删除后. 原本对 S_2 不可视的对象 O_3 重新变得对 S_2 可视. 因此在删除一个静态对象后,需要对该静态对象所能影响的路段的可视对象进行重新计算. 对于 *MinDist* 小于 O_1 或投影角度 θ 大于 O_1 的物体,其可视性不会随着 O_1 的消失而改变;对 *MinDist* 大于 O_1 的物体,原有可视的静态对象不会发生改变,原有不可视的对象则有可能变得可视,所以需要重新计算其可视性. 对于每一条受到影响的路段,对于该路段的 VR 表的删除算法如算法 2 所示.

算法 2. Delete().

输入: deleteObject *DO*, effectSegment *S*

输出: StaticVisibleRelation *SOVR*

1. delete *DO* from *S.SOVR*
2. FOR each *element* in *S.SOVR*
3. IF *DO.MinDist*<*element.MinDist* and *DO.*Θ>*element.*θ
4. THEN recalculate the visible(*element*,*segment*)
5. IF visible(*element.S*)=true
6. THEN set *element*.Flag=true;
7. ELSE set *element*.Flag=false;
8. return *SOVR*

与插入静态对象一样,在删除静态对象时,也需要对受影响路段进行处理. 通过反 K 近邻查询找出受影响路段并进行可视关系更新. 算法 2 描述了在确定了受影响路段列表后,对单一路段的可视关系表维护.

当插入一个新的路段时,就是一个构建静态对象可视关系表的新数据行的过程. 删除一个原有路段,就是在静态对象可视关系表中删除对应的数据行.

通过对静态对象可视关系表的维护,在三维场景内的静态对象与路网间建立可视关系. 将整体的三维场景,按照路网中的不同路段拆分. 将在线可视查询中的查询点的位置与静态对象的可视关系固定下来,将可视性计算转移到预处理阶段进行. 避免长时间的在线计算,加快静态可视查询的速度.

4.2.3 静态对象可视查询

利用静态对象可视关系表,可以批量的处理移动对象对静态对象的可视查询. 具体的做法就是首先确定查询者所在的路段;然后根据该路段在静态对象可视关系表中查询该路段所对应的静态对象.

既不需要进行在线的可视计算，又可以避免同一路段内的多个重复查询. 具体的算法将在后文给出.

4.3　动态对象关系表构建与维护

对于三维场景内的动态对象的可视查询，也是通过动态对象可视关系表进行的间接可视查询. 动态可视关系表存储的并不是动态对象本身的数据，而是路段与路段之间的可视关系，只在路段数据发生变化时进行维护，也是一个相对静态的表. 通过这一关系表，将动态对象间的间接可视关系固化.

4.3.1　动态对象数据存储

对于三维场景内的动态对象，以路段为主键为动态对象建立索引. 为每一个路段建立一个列表保存指向在该路段上的动态对象的指针，当动态对象的路段信息发生变化时更新该列表，确保对于动态对象的位置的实时监控.

4.3.2　动态对象可视关系表

动态对象可视关系表，存储的是路网路段间的可视关系，因此也是一个静态表. 在构建该表时，其实质为线段与线段之间的、在存在静态对象作为障碍集情况下的可视关系计算. 如图 5 所示，在一个城市环境中，路网路段间的可视性计算就转化为这样静态空间数据中的线段与线段间的可视性计算. 图中路段 S_8 受到静态建筑的影响，使路段 S_1，S_4，S_5，S_7，S_3 和 S_6 都对其不可视. S_{10} 对其可视的可视性受到距离的影响. 路段 S_2 和 S_9 由于与 S_8 共享一个端点，所以可以定义为绝对可视. 根据这一原理，给出绝对可视路段的定义 8.

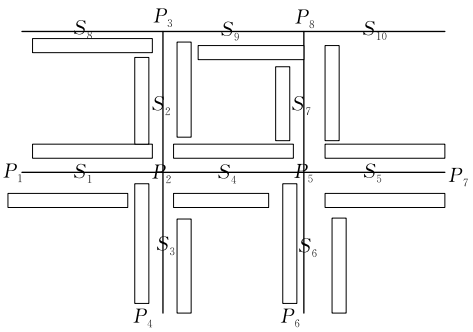


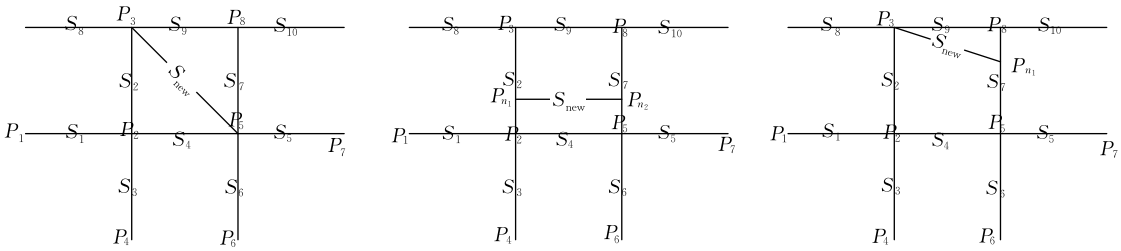
图 5　路段间的可视性

定义 8(绝对可视路段).　查询路段 $S_q=\{P_s, P_e\}$，P_s 和 P_e 分别是路段 S_q 的起始端点和终止端点，则对于与其共享这两个端点中的任意一点的线段，对于 S_q 可视. 即绝对可视路段集 $S_v=\{S(P_1,P_2)\mid \{P_1=P_s\cup P_1=P_e\cup P_2=P_s\cup P_2=P_e\}\cap S\neq S_q\}$.

除了绝对可视，还有一些路段间的关系属于条件可视. 如图 5 中路段 S_{10} 与 S_8 虽然不属于绝对可视，但进行可视计算却仍然可视. 只是这一可视性属于带有条件的可视. 当查询半径较小时，S_{10} 可能对 S_8 不可视. 因此在定义 6 中定义了路段可视距离 D 用于描述这一类条件可视，在图 5 中，S_{10} 与 S_8 之间的可视距离 $D=\mathrm{Length}(S_9)$. 对于动态对象可视关系表，当路段数据发生变化时需要进行维护.

4.3.3　新增路段

新增一个路段时，有 3 种情况：(1) 在原有端点上新增路段，如图 6(a)所示；(2) 在原有路段中间新增路段，如图 6(b)所示；(3) 两者结合，如图 6(c)所示.



(a) P_3 和 P_5 受影响　　(b) 新增两个端点　　(c) P_3, P_5, P_8 受影响

图 6　新增路段对于可视性的影响

在处理新增路段时，首先要确定其影响范围. 路段的新增不会凭空而来，必然与原有路网结合. 如图 6 所示，无论是哪种新增路段，都可以以端点来确定影响范围. 如图 6(a)中，受到影响的端点为 P_3 和 P_5，于是受到影响的路段为这两点所链接的所有路段，$\{S_8, S_9, S_2, S_7, S_4, S_5, S_6\}$，需要对这些路段的可视关系和可视距离进行重新计算；在图 6(b)中，新增了两个端点 P_{n1} 和 P_{n2}，这两个端点在路段 S_2 和 S_7 上，间接影响了端点 $\{P_3, P_8, P_2, P_5\}$，由此可以确定受影响的路段为 $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$，需要对这些路段的可视性关系和可视距离重新计算；而在图 6(c)中，受到影响的端点为 $\{P_3, P_5, P_8\}$，受到影响的路段为 $\{S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$.

4.3.4　删除路段

当删除一个路段时，不仅要删除与该路段有关

的数据，还要计算该路段消失对于其他路段可视距离的影响. 由于可视性是一种受距离影响的概念，因此路段的增加或删除，其影响范围是有限的. 如图 7 所示，路段 S_7 被删除，对 S_7 的端点 $\{P_5, P_8\}$ 产生影响，进而影响了 $\{S_4, S_5, S_6, S_8, S_9\}$ 的路段间的可视性和可视距离. 因此需要对这些路段间的可视性和可视距离进行重新计算.

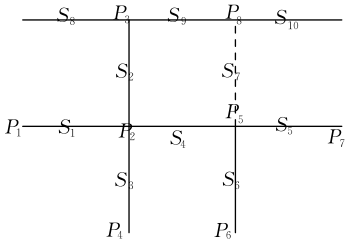


图 7 删除路段对于可视性的影响

通过对于动态对象可视关系表的维护，将移动对象间的在线的实时可视计算变成可以在预处理阶段完成的路段间可视关系计算，将复杂的计算过程在预处理阶段进行，避免长时间在线计算. 同时，对于同一路段的所有查询点的重复查询，可以通过批处理进行，降低了服务器压力.

4.3.5 动态对象可视查询

动态可视查询算法是基于动态对象可视关系表的线性查询. 在进行查询时，根据主函数传递来的查询点所在的路段 *id* 参数，在动态对象可视关系表中找到对于该路段可视的所有路段 *id*. 最后根据这些路段 *id* 再去动态对象数据中线性查找在这些路段上运动的所有动态对象. 这样对于动态对象的可视查询，就不必具体计算两个移动对象之间的可视性，而是直接进行基于路段的范围查询. 在对路网移动对象进行范围查询或扩展路径计算时，对路径的选择做出限制. 排除不可视路段，以此来排除范围查询中的不可视对象. 具体的查询算法将在后文给出.

4.4 性能分析

在现有的商业 GIS 系统中，也有其它的以路网为中心的动态加载模式，但是一般都是以路段为中心进行的范围加载. 建筑密度越高，其加载的不可视的、无效的静态对象比例就越高，整体性能就越差. 本文所提出静态对象可视关系表的意义在于：(1) 通过静态对象可视关系表，可以将三维场景内的静态对象与路网之间的可视关系固化，将移动的观察者对静态对象的连续可视查询转化为观察者所在路段的可视静态对象的查询；(2) 对于场景内的静态对象的可视性变化也可以通过静态对象可视关系表固定下来，如新建或拆除一栋建筑对周围建筑的可视性影响；(3) 原有的可视查询需要进行复杂的可视计算，响应时间较长，而通过静态对象可视关系表则将查询转变为简单的线性查询，处理速度更快；(4) 可以批处理在同一路段上的所有观察者的静态可视查询，降低服务器压力.

通过静态对象可视关系表，在对大规模的三维场景中的静态对象进行更新时，不再需要像现有模式一样强制所有用户进行更新，而是在后台完成更新，提高了场景的维护性能和灵活性. 而在用户进行在线查询时，只需要对静态对象可视关系表进行线性查询，提高了反应速度.

由于动态对象的运动轨迹是被限制于路网之上的，因此本文根据前文所提出并证明的定理 2，动态对象间的可视性可以由动态对象所在路段之间的可视性代替. 通过这一方式将动态对象间的动态可视性转化为静态的路段间的静态可视性. 在离线预计算阶段完成对于路网路段间的可视性计算，确定对于任意路段的可视路段列表，在在线查询阶段，不再计算具体的动态对象之间的可视关系，而是查询对于查询点所在路段可视的路段上的动态对象. 将动态对象的可视计算转变为路网内可视路段内的范围查询. 这一间接可视查询方法虽然降低了查询的精度，却满足了实时性的需求，同时使框架可以批处理动态对象的可视查询，对于动态对象的数量增加也有很好的适应能力.

主要的可视计算过程在离线预处理阶段，即生成静态对象可视关系表和动态对象可视关系表时进行. 生成静态对象可视关系表的计算只需要进行一次，其后静态对象再发生变化时只需要进行相应的更新维护即可. 对于单个路段，在生成静态对象可视关系表的过程中，首先进行一次范围查询，确定涉及到的 n 个静态对象并排序，然后对 n 个静态对象进行比较计算，如果可视，加入静态对象可视关系表并加入障碍集，直至完成循环. 因此对于拥有 m 个路段的整体地图而言，在生成静态对象可视关系表时时间复杂度为 $O(mn\log n + mn^2)$；在插入和删除对象阶段都需要遍历可视表和对影响到的静态对象进行可视计算，因此时间复杂度为 $O(\log T \times \log n)$. 动态对象可视关系表在本质上与静态对象可视关系表一样是静态表，只是存储的是路段之间的可视关系，是为处理动态对象而存在的. 因此其维护的时间复杂度与静态对象可视关系表一致.

5　基于路网的可视查询

5.1　基于路网的间接可视查询算法

在本文所提的框架中，对于三维场景内的静态对象和动态对象的可视查询都是在在线查询阶段进行了线性查询. 由于其是间接可视查询，因此避免了复杂的在线计算，从而满足框架的时间要求. 总体的基于路网的间接可视查询算法(Indirect Visible Query based on Road Section，IVQRS)如算法 3 所示.

算法 3.　IVQRS().

输入：QueryPoint p，$SOVR$，$DOVR$

输出：loadVisibleSet LVS

1.　$sectionId=p.sectionId$
2.　StaticSet S_1=staticVisibleQuery($sectionId$){
3.　$id=sectionId$,
4.　FOR id//
5.　　$RS+=SOVR$.get($sectionId$)
6.　　Return RS}
7.　DynamiSet S_2=dynamicVisibleQuery($sectionId$){
8.　　VisibleSegment VS=QO.DOVR($sectionId$,R)
9.　　rs=RangeQuery(VS) in MOS
10.　　FOR each element e in rs
11.　　IF e is not in LRS
12.　　e add to MO
13.　　Return MO}
14. $LOS=S_1+S_2$;

静态对象可视关系表 $SOVR$ 和动态对象可视关系表 $DOVR$ 作为全局变量在离线预处理阶段完成. 在第 2 行和第 7 行分别调用静态对象可视查询算法 staticVisibleQuery()和动态对象可视查询算法 dynamicVisibleQuery()，获取静态对象可视结果集 S_1 和动态对象可视结果集 S_2. 将这两个结果集汇总即得到对于查询点可视的、需要加载的对象集 LVS.

对于静态对象的可视查询主要是基于静态对象可视关系表进行，在 staticVisibleQuery()中，静态对象可视关系表作为全局变量，不需要从主函数传递. 第 3 行接受从主函数传递而来的查询点所在路段 id 信息. 从第 4 行开始，当从主函数传递而来的查询点所述路段 id 为多个路段时，如站在某个路口，进行一次 for 循环顺次获取所有路段所对应的静态可视对象. 在第 5 行，根据路段 id 在静态对象可视关系表中查询对应数据行. 最后返回可视静态对象结果集.

对于动态对象的可视查询时调用 dynamicVisibleQuery(). 首先，在第 8 行进行 Hash 查询，根据查询点所在的路段及查询半径，在动态对象可视关系表 $DOVR$ 中找出对于该路段可视的路段 id 集合，缩小查询范围；第 9 行是在每个路段所保存的移动对象数据表中进行线性查询，获取该路段中动态对象的指针，通过该指针确定动态对象候选集；在第 10 行开始进行判断过滤重复结果；第 13 行返回过滤后的移动对象结果集，客户端按照该结果集加载更新可视移动对象. 第 14 行汇总静态对象与动态对象作为可视查询结果集返回.

可以明显看出，在进行可视查询时，并不是以个体进行直接的可视计算并查询，而是以路段为单位进行间接可视查询，从而避免了长时间的在线可视计算，以精度为代价换取查询反应速度和低资源消耗.

5.2　性能分析

对静态对象的可视查询已经转变成为了根据路段 id 在一个静态表中时间复杂度为 $O(1)$的 Hash 查询. 因此根据传递而来的路段数量 n，单个查询点的在线静态可视查询的时间复杂度 $T_s=O(n)$.

在动态对象可视查询中，首先根据主函数传递而来的路段 id 在动态对象可视关系表中进行线性查询获取可视路段 id 集合，这一过程的时间复杂度为 $O(n)$. 然后根据 id 集合获取该路段中动态对象的指针来确定动态对象候选集，该部分的时间复杂度为 $O(n\times\log n)$. 在排除重复结果的过程中采用快速排序法的时间复杂度为 $O(n\times\log n)$. 因此整个动态对象可视查询的时间复杂度 $T_d=O(n+n\times\log n+n\times\log n)$.

基于路网的间接可视查询算法包括静态对象可视查询和动态对象可视查询两部分，所以其整体时间复杂度 $T=T_s+T_d$.

场景内路段数量为 n，场景内动态对象(观察者)数量为 m，$m\gg n$. 场景内动态对象，即观察者都是固定于路段范围内，因此他们的查询请求可以按照路网路段分组获取. 由于基于路段的可视查询是以路网路段为关键字进行的线性查询. 因此对于整个场景服务器而言，同时的查询请求进程数最大等于 n，所以本文提出的基于路网的可视查询框架能够有效处理动态对象数量增加所带来的压力.

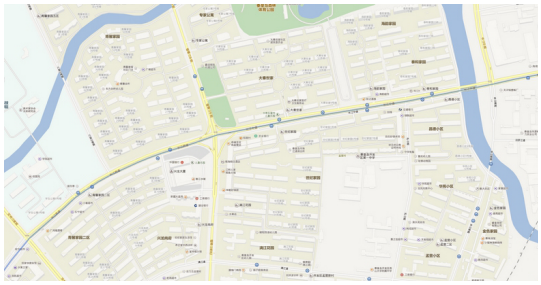
6　实验验证

实验在 Intel(R) Core(TM) i5-4200U CPU、

4 GB RAM、Windows7-64 bit OS、AMD Radeon HD 8670M 2GB 显存、网络传输速度 300 KB/s 环境下进行. 实验程序使用 C++语言在 Visual Studio.net 2010 开发环境下编写. 对本文提出框架的验证,将从静态对象和动态对象两个方面进行.

6.1 静态对象可视查询实验

对于三维虚拟现实场景内静态对象的可视查询实验,从单位时间加载模型数量、资源消耗情况、对于网络带宽等硬件压力方面做比较实验,以验证本文所提框架对静态对象的动态可视查询的有效性. 首先根据实际地图特意选取并制作了一份 5 km×3 km 的三维场景如图 8 所示. 该场景包括 9 个街区、462 栋建筑. 其中包括一个较为开阔的学校区、1 个建筑物密集的商业区、5 个建筑风格统一分布均匀的住宅区和 2 个视野开阔占地面积较大的工业区. 该模型的规模并不大,但是功能分区和建筑分布具有各种代表性,因此采用这一三维虚拟现实场景作为框架中对于静态对象可视查询的实验数据. 在预处理阶段建立静态对象可视关系表,包括 24 条道路边,462 栋建筑.



(a) 街区

(b) 建筑

图 8 静态对象试验图

当观察者进入场景后,整体式加载需要加载整个场景,范围加载需要以观察者为圆心无差别加载半径内的建筑,而本文提出框架中的可视加载则是获取观察者所在路段 ID 后,根据静态关系表对静态对象有选择的加载. 整体式加载不受观察者所在位置的影响,每次加载的数据量是一定的;范围加载和可视加载都受到观察者所在位置的影响,因此对它们在不同半径下的加载数量做比较实验. 对范围查询和可视查询在不同的半径下分别做 100 次随机查询点的实验后,取每次加载静态对象数量的平均值对比数据如图 9 所示.

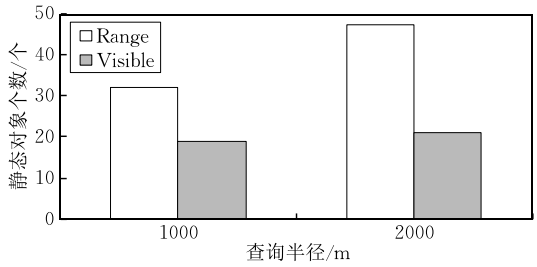


图 9 静态对象在不同查询半径下的加载数量对比

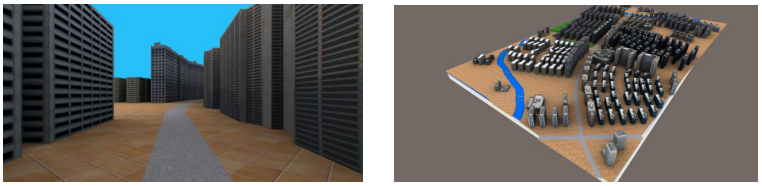
图 9 中能清楚地看出,加载的静态对象的数量随查询半径的增加而增加,但是可视查询的加载数量和增加速度都远远小于范围查询. 这是因为范围加载模式下会加载大量的不可视静态对象,查询半径越大则不可视对象越多. 而可视查询中可视对象与半径不是正比关系,而是存在一个极限. 因此当查询半径增加时,可视查询的结果数量远远小于范围查询的结果数量. 这是由可视性本身的特性所决定的. 这也就意味着,当观察者处于运动状态时,本文所提框架相较于现有的基于范围加载的商业软件在加载数据量优化方面具有更大的优势.

当观察者移动时,移动距离越远,则框架对于数据的加载量优化性能越好. 为了验证这一点,观察者选择如图 8(a)中灰线所示的移动轨迹,分别进行整体场景加载、范围查询加载和可视查询加载 3 种加载方式的对比实验. 对于观察者的效果对比如图 10 所示. 在 3 种不同加载模式下,对于场景内的观察者而言,其视觉感受并无区别;但这 3 种模式所需要传输和加载的数据量则存在极大的不同,对于客户端的硬件压力也完全不同,数据对比如表 1 所示.

表 1 不同加载方式的结果对比

加载模式	建筑数量/个	数据量/MB	传输时间/s	加载时间/s
整体加载	462	33.2	113	37
范围查询加载	410	29.4	101	33
可视查询	95	17.2	59	18

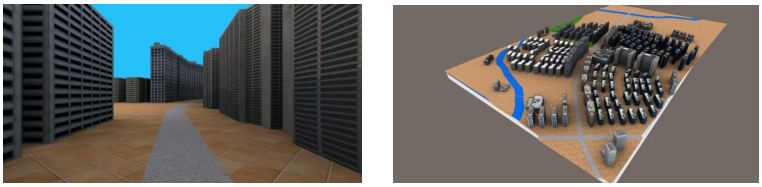
3 种不同加载方式中,整体式加载的速度最慢,数据从服务器端到客户端的传输时间就超过可视查询的传输时间 91.5%. 加载时间也超过了 105.6%. 而且,由于硬件显卡的限制. 当模型加载的数量越多,场景内的面片数越多,CPU 占用越多,场景的交互性能下降越严重. 在本实验中,在尽量使用简单的



(a) 整体加载下主视角、场景和数据

表i　整体式加载数据表

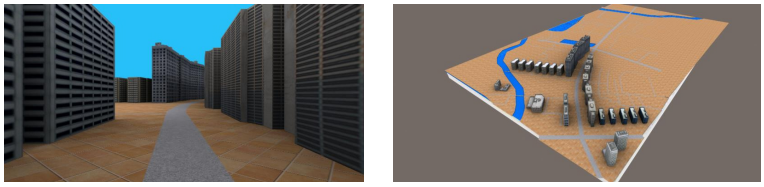
模型数量	场景规模/MB	CPU占用/%
462	33.2	65



(b) 范围查询加载下主视角、场景和数据

表ii　范围加载数据表

模型数量	场景规模/MB	CPU占用/%
207	15.7	54



(c) 可视查询加载下主视角、场景和数据

表iii　可视加载数据表

模型数量	场景规模/MB	CPU占用/%
28	2.4	38

图 10　不同加载方式视觉效果对比

三维模型以降低场景面片数的情况下，进行整体加载的场景仍然占用了 65%的客户端 CPU 资源. 而采用静态可视加载模式后，虽然在服务器端进行了长时间的离线预处理以生成静态对象可视关系表，但在线查询加载阶段，从服务器端需要传输的数据量只有整体加载模式的 51.8%，传输时间也相应地降至 52.2%，客户端 CPU 占用平均只有 35%. 在本文所提框架中，针对静态对象的可视查询加载模式被证明具有极为优良的性能. 从模型加载情况来看，在实际的应用中，对于不同的街区，静态可视加载的效率是不同的. 对于学校区和工业区，由于建筑低矮，存在大片的空旷区域，其加载量变化不大. 而对于高楼林立的住宅区和商业区，加载量变化极大. 表 1 中的数据差异主要也是来源于此. 在以后的研究中将根据这些情况做进一步的细分，以满足更大规模、更多模式下三维场景的实时可视查询加载.

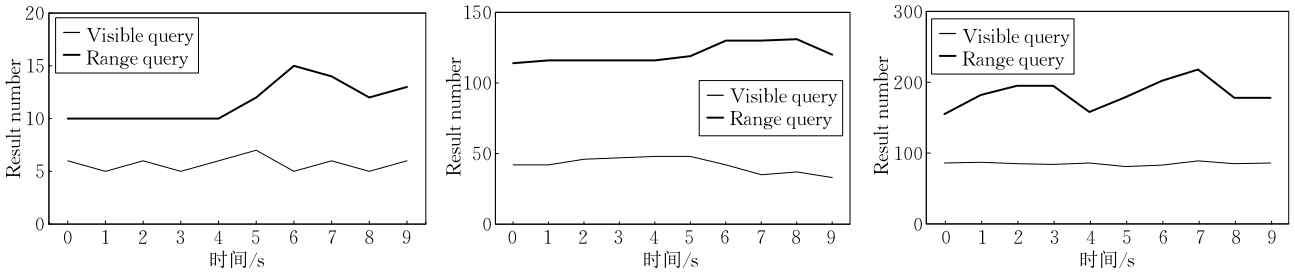
6.2　动态对象可视查询实验

对于三维虚拟场景内动态对象的可视查询实验，主要目的是验证采用动态对象可视关系表后，利用路段间的可视性对移动对象查询的结果数量与范围查询结果数量的对比. 因此可以采用二维的移动对象数据进行计算，在本文中路网移动对象数据采用德国 Oldenburg 的道路网及对应的移动对象生成器. 该路网拥有 6105 个道路节点、7035 条边、平均路段长度 184，是移动对象研究中广泛采用的测试数据集. 利用标准移动对象生成器生成10K、100K 和 200K 这 3 个不同规模的移动对象数据集. 范围查询的半径分别设为 200 m、500 m 和 1000 m. 由于没有具体的 Oldenburg 的建筑三维地图，无法真实确定 Oldenburg 路网中的动态可视表，因此默认只有相邻路段即拥有相同道路节点的路段间可视.

本文所提出的动态可视查询，是通过高速度、低精度的间接可视查询取代高精度、低速度的直接可视查询，其本质是一种范围查询，因此为了表现框架在进行移动对象查询时与现有模式下的范围查询在数据量方面的性能优势，采用本文间接可视查询与范围查询的对比实验. 首先进行在相同的查询半径、不同规模数据集中对范围查询与可视查询进行对比实验，重复实验 500 次取均值，结果随时间变化如图 11 所示.

从实验结果对比明显能够看出，随着数据集规模的增大，路网中的移动对象密度相应增加，范围查询和可视查询的查询结果数量都在增加. 但是可视查询的结果集在总数量上小于范围查询的结果集. 查询结果数量对比如图 12 所示.

平均查询时间对比如图 13 所示. 范围查询与可视查询的查询时间相差无几，但是仍然可以看到的趋势是，随着移动对象集规模的扩大，可视查询的增长率要低于范围查询. 这是因为移动对象的可视查询的本质就是在范围查询的基础上利用可视关系表对查询的路段做可见性方面的过滤，所以二者在查询机理上并不存在根本性的差别. 但是因为在进行



(a) 移动对象集规模:10K (b) 移动对象集规模:100K (c) 移动对象集规模:200K

图 11 相同半径下的范围查询与可视查询结果变化趋势对比

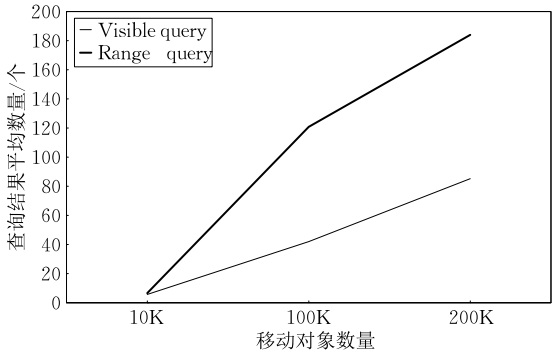


图 12 范围查询与可视查询结果对比

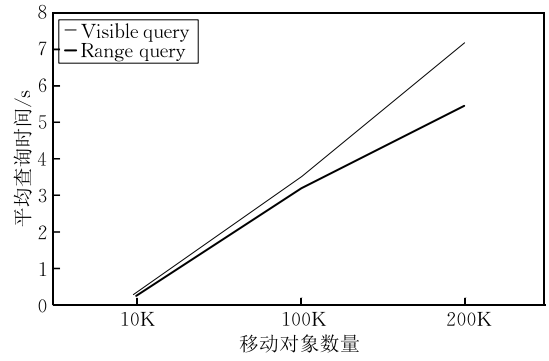


图 13 范围查询与可视查询时间对比

路网的范围查询时,可视查询首先排除了不可视的路段,因此其需要计算的路段数量小于等于正常范围查询的比较范围,所以可视查询的速度仍然要快于范围查询.

通过现有的数据可以看到,在不同规模的数据集中,可视查询的速度优于范围查询,结果集规模也小于范围查询.随着移动对象数据集规模的增长,可视查询的结果数量增加速度也小于范围查询结果数量的增加速度.

接下来我们验证不同的查询半径对于可视查询的影响.在移动对象为 200K 的数据集中,分别进行半径为 200 m、500 m 和 1000 m 的可视查询.实验结果如图 14 所示.由于在本次实验中设定只有共享某路段结点的路段间才可视,因此可视查询对于查询半径的变化没有反应,这是因为无论查询半径如何变化,对于查询点所在的路段可视的对象只有那么多,除非查询半径小于路段长度,否则结果数量不会随查询半径变化而发生变化.在真实的三维场景中,随着查询半径的增加,某些可视距离 D 小于查询半径的路段也会加入到可视查询结果中,因此数据量应会随着查询半径的增加而增加,但是由于可视路段本身的限制,其增加率将小于范围查询.

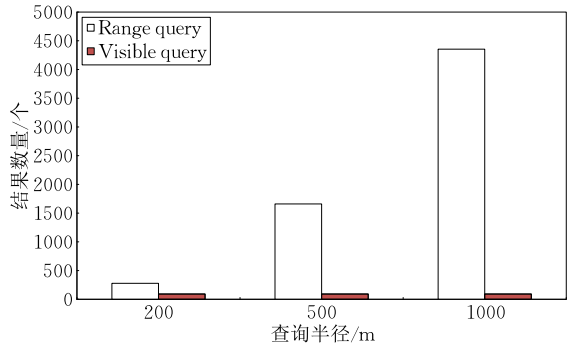
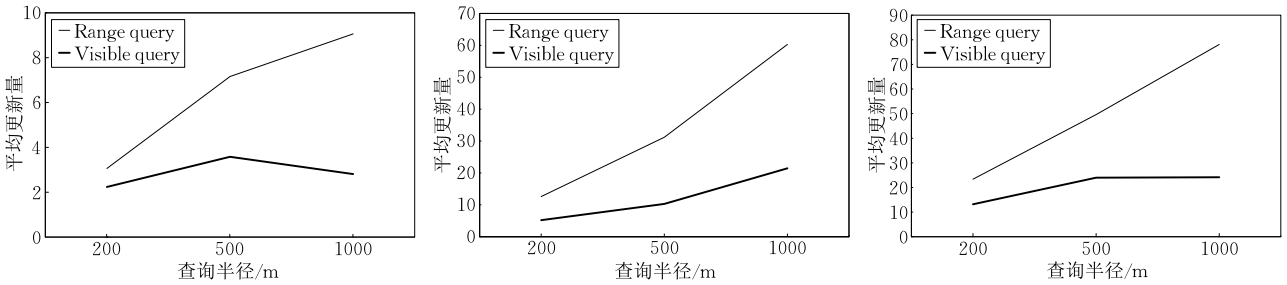


图 14 范围查询与可视查询在不同半径下结果数量对比

接下来对范围查询与可视查询的数据更新量进行对比实验.实验结果如图 15 所示.

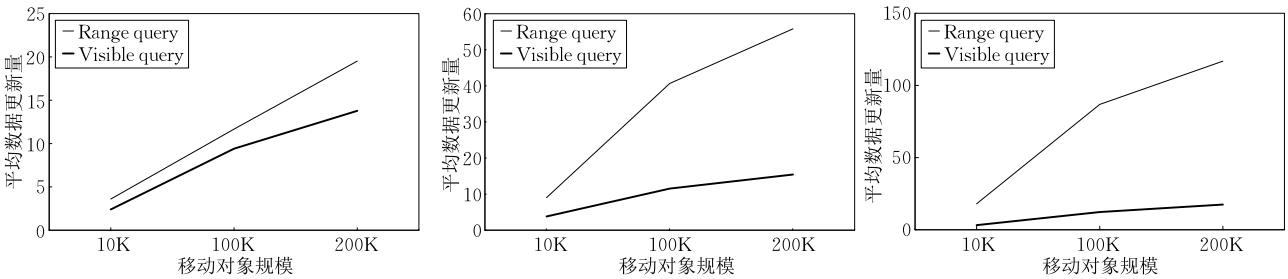
从图 15 中可以看出,随着数据规模的扩大和查询半径的增加,范围查询和可视查询在每一次查询周期的数据改变量也在增加.但是可视查询的数据改变量明显小于范围查询的改变量.这是因为在范围查询结果集中,有很多动态对象在不在可视路段上,所以在这些路段上的动态对象的变化情况不会在可视查询结果集中体现出来,于是可视查询的结果集在每一个查询周期内的变化量就远远小于范围查询结果集的变化量.在同一数据规模下,不同查询半径的查询结果集的数据改变量如图 16 所示.在同样规模的数据集中,随着查询半径的增加,范围查询和可视查询在每个查询周期内的数据更新量都在增加.但是非常明显的是,可视查询不仅仅在数据更新量上小于范围查询的数据更新量,而且其变化趋势也明显小于范围查询更新量的变化趋势.在三维虚拟场景的动态加载中,每一个动态对象都意味着一

组或多组模型，因此查询结果或更新对象数目的减少将直接减少更大规模的数据传输量，对于大规模三维虚拟场景的实时加载具有重要意义. 通过实验验证在本文所提的可视查询框架下，对于三维场景内的静态对象和动态对象的可视查询，可以有效避免复杂的在线可视计算，满足对于静态对象查询和动态对象查询的实时性需求，从而满足对于大规模三维虚拟场景的动态实时可视加载.



(a) 移动对象集规模10K　(b) 移动对象集规模100K　(c) 移动对象集规模200K

图 15　范围查询与可视查询更新结果量对比



(a) 查询半径200 m　(b) 查询半径500 m　(c) 查询半径1000 m

图 16　不同查询半径对于更新数据量的影响

7　总　结

在基于可视查询的大规模三维场景的实时加载中有两个难点：(1) 可视计算复杂且耗时，计算时间开销随着查询数量增多而急剧增加，服务器压力过大；(2) 对于动态对象的可视计算随着动态对象数量的增加而变得过于复杂，耗时过长，不能满足在线查询的时间要求. 本文针对这些困难，提出大规模三维场景的可视查询加载框架. 通过设计静态对象可视关系表和动态对象可视关系表将三维场景的可视性与路网路段建立对应关系，批量处理查询，降低服务器压力；同时，通过静态对象可视关系表与动态对象可视关系表将动态的在线可视计算查询转化为静态的线性查询，满足了在线查询的时间要求.

在大规模的三维场景实际应用中，该框架在不同类型的场景中的效果不同. 在城市类环境中，尤其是高楼林立的商业区中，基于可视性的查询性能比较优异. 但是在自然类环境中，尤其是在复杂地形环境中，现有模型仍存在较大的改进空间. 在进一步的研究中，对于城市道路环境下的静态对象的可视性推理和基于移动对象运动方向推理的查询算法以及脱离道路束缚的自由动态对象间的关系推理查询将是研究的重点.

致　谢　衷心感谢提供修改意见的审稿专家和编辑！

参　考　文　献

[1] Guan Meng, Zhang Yan, Bai Wen-Yang. Continous visible nearest neighbor queries on land surface. Journal of Computer Research and Development, 2010, 47(Supplement): 133-138 (in Chinese)
(管猛，张剡，柏文阳. 基于地表的连续可见最近邻查询方法. 计算机研究与发展，2010，47(Supplement)：133-138)

[2] Henderson D L. Modterrain: A Proposed Standard for Terrain Representation in Entity Level Simulation[Ph. D. dissertarion]. Naval Postgraduate School, Monterey, USA, 1999

[3] Schaufler G, Dorsey J, Decoret X, et al. Conservative volumetric visibility with occluder fusion//Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. New Orleans, USA, 2000: 229-238

[4] Zhang J, Papadias D, Mouratidis K, et al. Spatial queries in the presence of obstacles. Lecture Notes in Computer Science, 2004, 2992: 366-384

[5] Nutanong S, Tanin E, Zhang R. Visible Nearest Neighbor Queries. Berlin Heidelberg: Springer, 2007

[6] Gao Y, Zheng B, Chen G, et al. Visible reverse k-nearest neighbor query processing in spatial databases. IEEE Transactions on Knowledge & Data Engineering, 2009, 21(9): 1314-1327

[7] Gao Y, Zheng B, Chen G, et al. Continuous visible nearest neighbor query processing in spatial databases. The VLDB Journal, 2011, 20(3): 371-396

[8] Lu L, Yang C, Wang W, et al. Voronoi-based potentially visible set and visibility query algorithms//Proceedings of the 2011 Eighth International Symposium on Voronoi Diagrams in Science and Engineering (ISVD). Qingdao, China, 2011: 234-240

[9] Kusakari Y, Sugimoto Y, Notoya J, et al. Methods for searching mutual visible-intervals on moving object// Proceedings of the Workshop on Algorithms and Computation. Dhaka, Bangladesh. 2007: 13-27

[10] Wang Yan-Qiu, Xu Chuan-Fei, Yu Ge, et al. Visible k nearest neighbor queries over uncertain data. Chinese Journal of Computers, 2010, 33(10): 1943-1952(in Chinese)
(王艳秋, 徐传飞等. 一种面向不确定对象的可见近邻查询算法. 计算机学报, 2010, 33(10): 1943-1952)

[11] Guo Xi, Zheng Bai-Hua, Ishikawa Yoshiharu, Gao Yun-Jun. Direction-based surrounder queries for mobile recommendations. The VLDB Journal, 2011, 20(5): 743-766

[12] Cheema M A, Zhang W, Lin X, et al. Continuous reverse k nearest neighbors queries in Euclidean space and in spatial networks. The VLDB Journal, 2012, 21(1): 69-95

[13] Güting R H, Behr T, Xu J. Efficient k-nearest neighbor search on moving object trajectories. The VLDB Journal, 2010, 19(5): 687-714

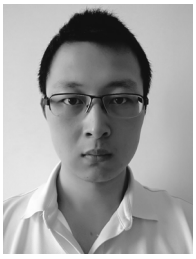
[14] Wang Y, Zhang R, Xu C, et al. Continuous visible k nearest neighbor query on moving objects. Information Systems, 2014, 44(8): 1-21

[15] Papadias D, Zhang J, Mamoulis N, et al. Query processing in spatial network databases. The VLDB Journal, 2003, 29: 802-813

[16] Zhao Liang, Chen Luo, Jing Ning, et al. Continuous K nearest neighbor queries of moving objects in road networks. Chinese Journal of Computers, 2010, 33(8): 1396-1404(in Chinese)
(赵亮, 陈荦, 景宁等. 道路网中的移动对象连续 K 近邻查询. 计算机学报, 2010, 33(8): 1396-1404)

[17] Jeung H, Man L Y, Zhou X, et al. Path prediction and predictive range querying in road network databases. The VLDB Journal, 2010, 19(4): 585-602

[18] Lin C S, Wu S Y. Processing directional continuous range queries for mobile objects on road networks//Proceedings of the 2014 IEEE 4th Annual International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER). Hong Kong, China, 2014: 330-335



KONG De-Han, born in 1985, Ph. D. candidate. His current research interests include spatial databases, moving objects databases.

LIU Yong-Shan, born in 1963, Ph. D., professor, Ph. D. supervisor. His research interests include spatial database.

Background

In recent years, there are more and more the applications of three-dimension virtual scene in internet. The application of large scale three-dimension (3D) scene makes the visible queries of the spatial database become the hotspot of research. The existing research on the visible query has two defects in the processing of large-scale 3D scene. First, the time of visibility calculation is too much to satisfy the requirement of on-line query of static objects. Second, there are too many invisible objects during the processing of mobile objects to cost so much resource.

In this paper, a visual query framework is proposed, which processes the visible query of static objects and dynamic objects separately. In this framework, the visibility of 3D scene is built and fixed on the road network by static object visible relationship (SOVR) table and dynamic object visible relationship (DOVR) table. The repeat visible queries in same segment will be processed by using batch mode to reduce the stress of server. Through the SOVR, the real-time visible query is transformed to a Hash query to satisfy the requirement of speed. Through the DOVR, the invisible mobile objects in road network are eliminated to save the resource. With the help of the framework and algorithms of visible query proposed in this paper, we can get far better performance than existing models while loading large-scale 3D scene.

This work is supported by the National Natural Science Foundation of China under Grant No. 61572422 and the Natural Science Foundation of Hebei Province China under Grant No. F2013203187.

The authors of this paper have been engaged in spatial-temporal query processing and moving object query processing. They have published some papers in this area.