大规模地形真三维可视化系统设计与实现 *

吕希奎, 易思蓉, 韩春华
(西南交通大学 土木工程学院, 成都 610031)

摘 要: 对已有算法进行了综合及改进,对地形绘制中的四叉树剖分控制、DEM 分块、纹理自动分块、地形数据的数据库存储、视见体剪裁、误差控制、裂缝拼接等方面进行了深入研究,提出了改进方法,并结合 OpenGL 立体显示技术,基于微机平台实现了一个简洁、快速的大规模真三维立体地形实时绘制原型系统,达到了很好的效果。实验结果表明,该算法实现简单、内存开销较少、CPU 耗费小,具有较低的时间、空间开销,适于大规模地形的真三维实时可视化。

关键词: 地形可视化; 连续细节层次; 真三维; 实时漫游; 视区剪裁
中图分类号: TP317.4 **文献标志码**: A **文章编号**: 1001-3695(2008)02-0603-04

Design and realization of really 3D visualization system for real-time rendering of large scale terrain

LV Xi-kui, YI Si-rong, HAN Chun-hua
(*School of Civil Engineering, Southwest Jiaotong University, Chengdu* 610031, *China*)

Abstract: Based on previous algorithms, this paper designed a new easy and fast terrain-rendering algorithm. Including the procedure of quad-tree generating, the texture automatic piecemeal, the terrain data database saves, view-culling, error-control and crack elimination conducted the thorough research and proposed a improvement method, unified the OpenGL stereo display technology, was realized succinctly based on the microcomputer platform, the fast large-scale really three dimensional terrain real-time plan prototype system, achieved a good visual effect and a high and steady FPS just on PC platforms. The exper-imental result indicates that, the algorithm realization is simple, the memory expenses are less, CPU consumed slightly, has a lower time and spatial expenses. It is suitable for the large-scale terrain really three dimensional real-time visible.

Key words: terrain visualization; continuously LOD(level of detail); really three dimensional; real-time wandering; view culling

0 引言

地形三维可视化技术近年来一直是相关领域的热点研究问题,在三维仿真和虚拟地理环境中占有十分重要的地位。随着遥感技术、卫星技术的发展,获取高分辨率的数字高程数据以及影像纹理数据成为可能,也使地形高程数据和纹理数据异常庞大,超出了一般图形系统的实时渲染和内存管理能力,对地形的实时绘制也提出了更高的要求。因此地形三维可视化技术研究的关键问题就是三维地形模型的管理、调度与实时渲染,以达到提高地形实时显示速度和最大程度提高视觉效果的最佳平衡。国内外在这方面进行了很多研究[1~11]。本文对前人算法进行了深入的研究和比较,提出了一种基于规则网格的、视点相关(view-dependent)的连续 LOD 地形模型的生成和实时绘制算法。在此算法中,采用地形小块(block)来组织数据,建立了视参数(视点位置、视线方向、地形起伏程度等)相关的误差评价函数,实现了视点相关的连续 LOD 地形模型的实时生成与绘制,并针对地形 LOD 算法中的几个关键技术进行了改进,提出的绘制算法在 PC 上达到了很好的效果。该算法的主要特点包括:a)给出了影像纹理自动分块原则和方法,实现纹理的自动分块,而不是以往研究的手工分块;b)实现了基于 Oracle OCI,解决了大规模地形数据(DEM 和影像纹理)数据库存储和调度问题,保证了数据的安全性,而不是以往研究的文件管理方式;c)提出的简化扇形视见体剪裁方法,使视见体剪裁计算简单而快速;d)DEM 子块单独构建 LOD,块内分辨率任意,提出向左向下原则,简化了 DEM 子块间裂缝拼接方法;e)基于立体透视投影原理,实现了地形真三维(3.0 维)可视化,而不是以往研究的伪三维(2.5 维)。

1 基于四叉树动态多分辨率三维地形的实现

1.1 基于四叉树结构的地形 LOD 模型算法

基于四叉树结构的 LOD 模型构造算法实质是通过递归的方法对地形进行从顶向下的四叉树划分[9]。在划分过程中实时计算节点误差。如果该节点误差大于限定的阈值,则对该节点所在区域继续进行划分,直到所有节点的误差小于给定的阈值。基于四叉树结构的 LOD 算法数据结构如图 1 所示。

收稿日期: 2006-12-18; **修回日期**: 2007-02-19 **基金项目**: 国家自然科学基金资助项目(50278082);西南交通大学博士生创新基金资助项目(200518)

作者简介:吕希奎(1976-),男,辽宁葫芦岛人,博士研究生,主要研究方向为虚拟环境与遥感铁路选线理论与方法(lvxikui@163.com);易思蓉(1957-),女,四川成都人,教授,博导,博士,主要研究方向为选线设计理论、城市轨道交通与磁悬浮交通技术;韩春华(1976-),男,云南大理人,博士研究生,主要研究方向为铁路选线智能专家系统.

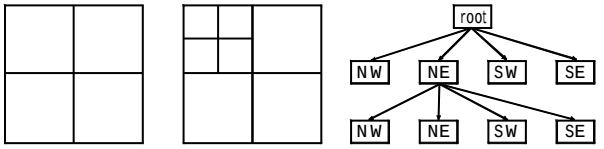


图 1 基于四叉树结 LOD 算法的数据结构

1.2 地形数据的分块

目前的计算机内存容量仍然有限,大规模的地形数据不可能一次性地调入内存。为此,笔者提出了一种基于数据分块、部分数据常驻内存、块内分层简化的处理策略。其基本思想是:首先将研究区地形数据划分为大小相同的 m 行 n 列,每个子块边长为 2^n+1,命名为 $\mathrm{Row}_i\mathrm{Col}_j$(i,j 分别为该子块所在的行和列),左下角为第 0 行第 0 列,对于最右侧和最上侧不满足大小相同要求的块,以无效数据(如 -9 999)填充,如图 2 中的灰色线填充部分;然后对每一个数据分块按照四叉树结构进行组织,以二进制格式存储到数据库中。考虑到 Intel 的 CPU 内存页大小是 4 KB,块的大小取 33 ×33 较好,这样在一定程度上提高了存储效率,降低了内存缺页的次数,同时数据调度也不会太频繁。

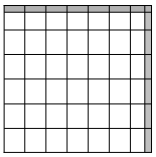


图 2 地形数据分块

1.3 影像纹理的自动分块

由于影像数据占用较大的存储空间,而且一般的计算机图形渲染设备限制了单次装载影像的大小,如目前的 OpenGL 渲染设备支持的单张影像的最大范围为 2 048 ×2 048,而在实际多数情况下地形影像的范围远远大于这个限制数量的大小。需要对大范围的影像纹理进行分块处理,即在平面空间上将影像分割成一系列规则的影像块。

影像分块分为规则分块(一个影像块对应一个或多个 DEM 子块)和不规则分块(一个影像块并不对应整数个 DEM 子块)。在实际应用效率上,不规则分块将造成大量纹理数据冗余,而且在纹理影像块调度时计算繁琐,因此,这里采用规则分块中的一个影像块对应一个 DEM 子块,漫游时可立即确定所应调入的纹理影像块。

通常情况下,影像纹理覆盖范围与地形并不是完全重合,因此在分块时要进行如下判断,如图 3 所示。

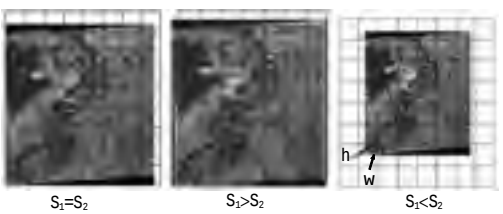


图 3 纹理分块示意图

设影像纹理覆盖范围为 S_1,地形范围为 S_2。

a)$S_1=S_2$,根据两者的范围坐标、影像分辨率和 DEM 子块大小,计算 DEM 子块对应的影像块大小,根据计算的尺寸对影像进行分块处理。

b)当 $S_1>S_2$ 时,先对原始影像进行剪裁处理。根据两者的范围坐标和影像分辨率,计算应剪裁掉的影像范围,对原始影像剪裁,使 $S_1=S_2$,然后按照 $S_1=S_2$ 时进行影像分块。

c)当 $S_1<S_2$ 时,根据影像左下角坐标,计算出影像起始分块对应的 DEM 子块号,并计算影像纹理起始行所对应的影像块高度 h 和宽度 w,每一行第一列宽度为 w,第一行影像块高度为 h,其余则按照 $S_1=S_2$ 时进行分块,此时存在部分 DEM 子块没有影像纹理。

对于分块后的影像纹理,为了便于以后的纹理映射和保证三维交互显示时纹理数据调度的高效性,设计了如表 1 所示的数据结构用于管理这些纹理数据。

表 1 数据库存储的纹理模型数据结构

序号	空间位置(角点坐标)	影像大小	影像数据
1	x_1,y_1,x_2,y_2	$w_1\times h_1$	01010101…
2	x_1,y_1,x_2,y_2	$w_2\times h_2$	01010101…
ú	ú	ú	ú
n	x_1,y_1,x_2,y_2	$w_n\times h_n$	01010101…

从表 1 中可以看出,纹理数据是由一些子块影像数据构成。其中,每一子块影像具有一定的属性特征。空间位置表示该影像被映射的区域范围,影像的大小表示该子块影像的高和宽,用于计算纹理坐标。影像数据用于存储该子块影像数据,以 BLOB 数据类型存储在 Oracle 数据库中。

1.4 LOD 选择

判断一个子块是否继续细分的标准是屏幕投影误差的大小。如图 4 所示,Lindstrom 屏幕误差算法将一个顶点带来的误差 h 投影到屏幕上,用得到的屏幕误差 δ 与阈值参数 τ 进行比较,控制该节点被引入或忽略。

Lindstrom 给出的顶点判断公式为

$$\{d^2\lambda^2\Delta h^2[(e_x-v_x)^2+(e_y-v_y)^2]\}/[(e_x-v_x)^2+(e_y-v_y)^2+(e_z-v_z)^2]\leqslant\tau^2 \quad (1)$$

其中:λ 为单位长度对应的像素个数;τ 为误差像素阈值。

采用屏幕投影误差来衡量一个 LOD 是否继续向下划分,而精确计算屏幕误差的公式比较复杂,且时间开销较大。根据式(1)可得简化误差判定公式。

首先将式(1)作简化,变为如下形式:

$$e_s=[h_s\times\cos(\theta)]/[2\tan(\alpha/2)]\times\mathrm{Nodeerror}(i)/d_e \quad (2)$$

其中:e_s 为屏幕空间投影误差估算值;h_s 为屏幕高度;θ 为视点的下视角;α 为视点的垂向张角;$\mathrm{Nodeerror}(i)$ 为对应节点的空间误差(即静态误差,为地形起伏程度,在预处理计算完成);d_e 为视点到节点的距离。

于是,当 $e_s>\tau$ 时,节点设置分割标志为 true,需要进一步分割;否则设置分割标志为 false,不需要再进行分割。

式(2)中的前半部分仅视参数(视线方向、视景体参数)发生变化时改变,在预处理的过程中进行计算。由式(2)可知,屏幕投影误差与视点的下视角 θ 成反比,与视点距误差计算点的距离 d_e 成反比。当 $\mathrm{Nodeerror}(i)$ 一定时,d_e 越大,e_s 也就越小,如用户指定的阈值为 δ,那么当 d_e 充分大时,其在屏幕上的投影误差 e_s 小于 δ,从而可以将其忽略[10]。

1.5 误差阈值负反馈控制

由于算法每帧的计算量与三角形数量是正相关的;同时,不同地形区域粗糙度相差较大时,三角形数量及帧速率相差也较大。为此,引入负反馈控制的思想[11]。通过三角形数量对误差阈值 τ 进行动态调整:当场景三角形数增加过多,就加大 τ;反之亦然。虽然对不同地形区域该系数是不同的,但由于视点是连续变化的,绘制的地形区域也是连续变化的。又由于只

需要控制三角形数在一个范围内,对 τ 进行一阶负反馈即可,伪代码为

if(结果三角形数 > maxTriNum)　　$\tau = \tau \times k_1$;

else if(结果三角形数 < minTriNum)　　$\tau = \tau / k_2$;

只要设定合适的参数 maxTriNum、minTriNum、k_1、k_2 使得反馈可以很好地跟上地形的变化,同时又不产生反馈振荡即可。实验结果证明,负反馈的方法很好地稳定了绘制帧速率,设置如图 5 所示。

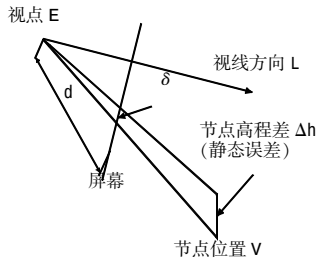


图 4　屏幕像素误差示意图

图 5　LOD 屏幕误差设置

1.6　LOD 模型构造中裂缝的处理

LOD 模型构造算法的另外一个重要特征是保证不同分辨率块之间模型的无缝隙拼接,从而保证在模型漫游或可视化过程中不会产生漏洞,保证 LOD 模型结构的有效性。地形是按照分块存储、调度和渲染的,这样就包括地形块内部和地形块之间的两种裂缝处理。

1)地形块内裂缝的处理　对于同一地形块内的区域而言,由于不同的地形子块间的分辨率不一样,在相邻精度等级相差一倍以上时,即在具有不同细节的地形间会出现裂缝现象。根据设计的四叉树数据结构,采用一定算法对缝隙进行缝合,在渲染地形时进行判断,如果有缝隙则进行一定的处理。这里采用垂直边线进行裂缝处理,对已经划分好的地形网格节点进行第二次遍历,遍历时对相邻节点精度进行比较。如果相邻的网格精度相差超过一倍,则在两个网格之间增加一条边线,如图 6 所示。

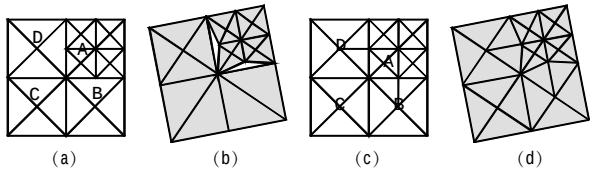


图 6　块内裂缝修正

2)相邻地形块之间裂缝的处理　在构建 LOD 模型中每个 DEM 子块都是单独进行的,这样可以有效提高 LOD 建模速度。当相邻两个 block 具有不同分辨率层次时,具有较高分辨率层次的 block 具有更多的高程点,两者共享边上就会出现 T 连接,引起裂缝,如图 7 所示。在对每个 block 构建 LOD 模型时,已经存储了每个节点的分割状态及高度,因此,算法采用后绘制的 block,其左共享边界向其前一列 block 看齐,下共享边界向其下一行 block 看齐,即向左向下原则,从而解决相邻地形块之间裂缝问题,如图 7 所示。

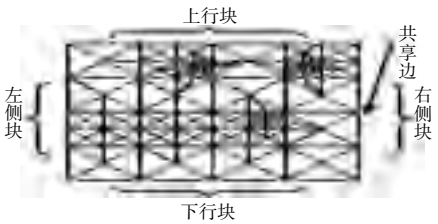


图 7　块间裂缝处理

2　系统优化关键技术

2.1　视锥体裁剪

剪裁是地形简化重要的一环,在特定视点下,地形大部分区域均位于视锥体之外,不需要绘制,应在预处理阶段剔除掉,因此,对视锥体外的数据进行剪裁是提高显示效率的关键之一。裁剪运算要占用较多的时间。传统的裁剪算法是解一个方程组,将每个三角形与视景体的六个裁剪面进行比较,判断是否可见;当处理的数据量很大时,会消耗大量的处理时间,导致渲染的速度非常慢。为了加快处理速度,忽略上、下剪裁面和近剪裁面,将视景体投影到 x-z 平面上,建立扇形剪裁区域,将节点(格网块)作为裁剪的最小单元进行裁剪,对数据块的节点进行可见性测试。

首先以每个地形块的几何中心为圆心,半径为 r,为每个地形块建立球形包围区域,以该球表示当前的地形块来完成各种裁减操作。由于所有的地形块大小相同,包围球的大小也相同。判断每个包围球的球心在 x-z 平面上的投影是否落于扇形内。如果是则绘制对应的地形块;否则分别计算球心到扇形两条边的距离,如果最小值小于 r,则绘制对应的地形块(图 8)。

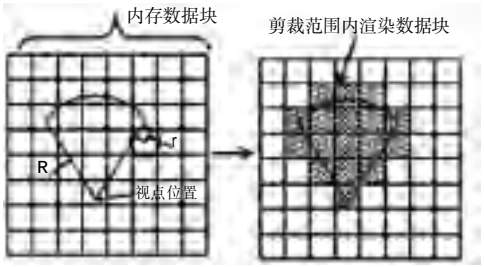


图 8　视锥体剪裁示意图

2.2　背面剔除

背面剔除是在地形绘制前,剔除掉视方向背离的多边形,只绘制面向视点的多边形,从而减少绘制的多边形数目。可通过计算视线方向和多边形表面法向量的夹角来确定是否剔除。

视线方向和多边形表面法向量的夹角可以转换为向量间夹角的计算问题。设向量 u 和 v 的夹角为 α(较小的那个角作为夹角),为此,构造角 α 的对边向量 $u-v$,从而 u、v 和 $u-v$ 向量构成了一个三角形,如图 9 所示。

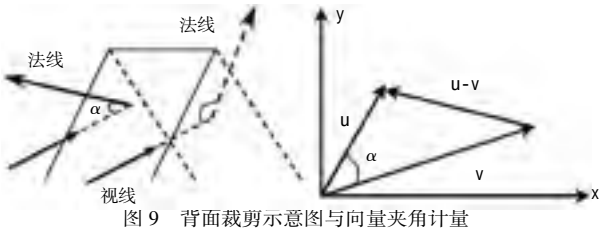


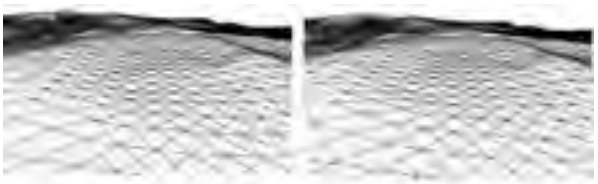
图 9　背面裁剪示意图与向量夹角计量

假设 $u=(u_x,u_y,u_z)$,$v=(v_x,v_y,v_z)$,从三角形的边角关系可以推导出

$$\cos\alpha=(u_xv_x+u_yv_y+u_zv_z)/(\|u\|\|v\|)=(u\cdot v)/(\|u\|\|v\|)\quad(3)$$

可见,当 $u\cdot v=0$ 时,向量 u 和 v 垂直($u\perp v$);当 $u\cdot v>0$ 时,u 和 v 的夹角为锐角(绘制三角形扇中对应的点);$u\cdot v<0$ 时,向量 u 和 v 的夹角为钝角(不绘制三角形扇中对应的点,即剔除该点)。

图 10 为同在屏幕误差 τ = 1 的情况下,分别进行背面剔除和未进行背面剔除的效果图。由所渲染的三角形数可以得出,由于进行了背面剔除,渲染的三角形数约为原来的 2/3,能够有效减少渲染的三角形。



(a) 背面剔除,TriNum=14 608　　(b) 不进行背面剔除,TriNum=23 614

图 10　背面剔除效果对比图(τ=1)

2.3　大规模数据的动态调度

大范围的地形数据通常非常庞大,难以全部显示,即使是利用一组 LOD 模型也难以做到。要实现形交互式渲染,每次只能读取其模型的一部分来进行;同时,随着视点和视线变化,参与计算的这一部分细节的详略程度也应动态地作相应改变[12]。维持这样一个与视点相关、视线相关的基于动态连续 LOD 模型的三维场景,需要一个动态视景更新机制来对数据进行有效的组织与管理。地形数据经过分块处理后,建立如下实时显示的分页机制:视点始终位于数据页的中点附近,在漫游过程中,随着视点的移动,不断更新数据页中的数据块,通过判断视点的当前位置(x_e,y_e)与数据页几何中心(x_c,y_c)间的两个方向的偏移量:

$$\begin{cases} \Delta x = x_e - x_c \\ \Delta y = y_e - y_c \end{cases} \tag{4}$$

当 Δx 为正时,视点向 x 的正轴方向移动;反之则向负方向移动。如果当 $\Delta y \geqslant$ cellwidth(数据块的宽度)时,将移动方向向上新的一行数据块读入数据页中,同时将反方向的一行数据块从数据页中删除,如图 11 所示。同理,根据 Δx 与 Δy 进行八个方向的移动,将移动方向上新的一行或一列数据读入数据页中,同时将反方向的另一行或另一列数据块从数据页中删除。这样,根据视点与数据页几何中心的偏移量大小情况,不断更新数据页中的内容,实现大范围的动态实时漫游。

3　地形真三维可视化的实现

3.1　立体透视投影原理的数学描述

将不同的图像分别放置入左右眼缓冲区,经透视投影,将产生一个无论在几何外观上还是感觉上都非常好的立体效果。一个高质量的立体图像包括两个立体部分,这两部分都是透视投影,且它们的投影中心(摄像机)在位置上平行对称,如图 12 所示。

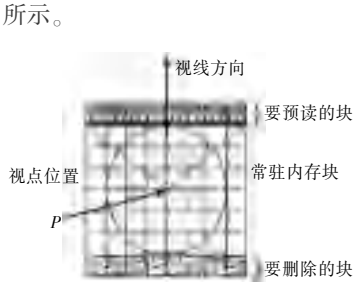


图 11　基于分块数据的动态数调度

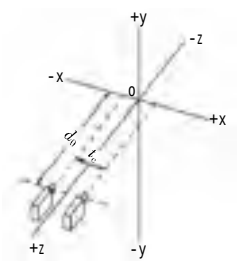


图 12　体图像的几何模型

假设摄像机位于 Z 轴正半轴上的点(0,0,d_0)处,d_0 表示摄像机距 x-y 平面的距离,点(x,y,z)投影到 x-y 平面的位置用式(5)(6)计算。

对左眼计算如下:

$$[(x + t_c/2) \times d_0/(d_0 - z) - t_c/2 \quad (y \times d_0)/(d_0 - z)] \tag{5}$$

对右眼计算如下:

$$[(x - t_c/2) \times d_0/(d_0 - z) + t_c/2 \quad (y \times d_0)/(d_0 - z)] \tag{6}$$

利用式(5)和(6)计算出来一对立体投影时,所有可能的三维点将投影到正视差区。左、右摄像机的透视投影能很好地平衡视差效果,使合成结果更具有立体感。

3.2　基于 OpenGL 真三维立体地形实现

在初始化每个显示窗口时加载立体缓存支持(两个前台缓存和两个后台缓存),分别设置左右眼的投影矩阵,并在后台缓存中绘制场景;最后将后台与前台缓存互换,这样后台缓存上的图像就被显示出来,而原来的前台缓存变为后台缓存用于新的绘图。整个绘图过程就是这样的循环往复过程。

将左右眼图像分别写入左右缓冲区中的伪代码如下:

```
glDrawBuffer (GL_BACK);//清空后台缓冲区
glClear (GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
  //显示左眼图像,准备向左后缓冲区写入场景内容
SetCamra(LEFT); //设置左照相机
glDrawBuffer (GL_BACK_LEFT);
DrawSecne(…);// 向左缓冲区写入场景
glClear (GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
  //显示右眼图像,即准备向右后缓冲区写入场景内容
SetCamra(RIGHT); //设置右照相机
glDrawBuffer(GL_BACK_RIGHT);
DrawSecne(…); //向右缓冲区写入场景
SwapBuffers (hdc);
```

4　结束语

根据上述算法,本文基于微机平台实现了一个简洁、快速的大规模真三维立体地形实时绘制原型系统,并结合 OpenGL 立体显示技术,实现了真三维立体地形,达到了很好的效果,如图 13 所示。实验运行环境:微机配置 CPU Pentium4 2.8 GHz、内存 512 MB、nVIDIA QuadroFX 4500 512 MB 立体图像卡、17 英寸显示器且分辨率设为 1 024 × 768 像素、显存 128 MB;DEM 数据原始采样点数目为 2 800 × 2 200,影像数据原始采样点数目为7 438 × 8 114(172 MB);在启动光照和纹理模型的条件下,平均刷新速率为 30 fps。

图 13　真三维　体地形效果图

参考文献:

[1]　万定生,龚汇丰. 一种基于四叉树的大规模地形实时生成算法[J]. 计算机工程与应用,2005,41(33):186-189.　(下转第 609 页)

3　实验结果分析

目前,上海市各类公交车为 18 186 辆,有 948 条线路,每天运送乘客量约达 750 万人次。现行的公交线路设置一般都是以一个居住区到市中心某一集散地,或到另一居住区为运营目的,中途又要经过某几个目的地。因此,上海市区大部分的公交线路走势呈现 S 形、C 形、Z 形等极不规则图形,造成重复线路多、绕道多和站点集中,加上私家车辆不断增多等原因,造成公交线路行驶时间过长、误点率高等问题,给市民出行带来很大的不便。缓解这些交通压力弥补轨道交通的不足,采用公交专用道是有关决策部门优先考虑的方法。根据前面所述的理论和算法,以上海城区及公交线路作为主要研究对象,本文在 VC 6.0、Mapx 5.0 和 Access 2003 环境中嵌入式开发本系统。系统完成各种 GIS 操作(如对各个图层进行控制、漫游、查询和相应的空间分析等),并能够在对公交线网评价的基础上对上海整个城市道路进行公交专用道设置选取工作。图 4 是实验系统的界面演示。图 5 为现有的公交线路层。

选取公交专用道考虑的因素有道路宽度、公交线路重复系数和高峰小时平均车速等。利用与或树对该知识进行表示后通过推理引擎对该问题进行求解,结果如图 6 所示。

图 4　辅助决策系统界面

图 5　上海公交线路层

图 6　上海公交专用道设置选取结果

根据《上海城市交通"十一五"规划纲要》,到 2010 年全市要建设 300 km 公交专用道,而在中心城区则要建设 100 km 以上的公交专用道,高峰时段公交专用道车辆运行时速要达到 18 ~20 km,为上班族乘车提速。从实验结果中可以看出,其完全涵盖了上海近期所建公交专用道(四平路——中山南路、虹桥路——肇嘉浜路——陆家浜路、中山南一路——中山南二路和山东路——云南路)。对实验结果分析可以看出,公交专用道多选取在人口出行量大的交通小区中,并与轨道交通相辅相承弥补了轨道交通产生的交通空白,缓解了一号和三号线轨道交通的压力。

在整个系统中交通领域相关知识表示录入界面如图 7 所示。在该界面中可以定义原则操作和符号表,并根据已知的符号表和原子操作构建求解问题的与或树。图 8 是问题求解界面,当用户在编辑框输入要解决问题的名称并点击按钮,就可以自动对已经构建好的与或树进行推理,同时对推理时所用到的知识进行解释。

图 7　基于与或树知识表示界面

图 8　问题求解界面

4　结束语

本文利用与或树结构对交通领域知识进行表示,用深度优先搜索和回溯技术实现推理引擎对快速公交线路的选取提供了一种新的思路,对其设置方法进行了有益探索。

将 GIS 与 ES 结合起来,是两者所研究的新方向。在构建系统中,GIS 的作用是存储显示空间数据并辅以必要的空间分析,相比传统方法能够更直观地显示给用户,且容易对线路进行编辑更新;专家系统的作用是对知识进行表示和推理后对线路的选取。在进行快速公交线路规划时要考虑的因素很多,然而主观上对这些因素重要性认识不同,直接会导致最后的结果不同。也就是说,知识的多少是决定本系统对线路进行选取时结果好坏的根本原因。对知识的精确表示和对问题求解高效的推理也是影响问题最终结果的重要因素。

参考文献:

[1] 王炜,杨新苗,陈学武. 城市公共交通系统规划方法与管理技术[M]. 北京:科学出版社,2002:76-77.

[2] 刘大勇,危辉,王增进. 基于地理信息系统的智能化应用及原型设计[J]. 计算机工程与应用,2003,39(28):224-226.

[3] 修文群,池天河. 城市地理信息系统[M]. 北京:北京希望电子出版社,2001:32-35.

[4] 龚建雅. 地理信息系统基础[M]. 北京:科学出版社,2001:3-5.

[5] 胡鹏,黄杏元,华一新. 地理信息系统教程[M]. 武汉:武汉大学出版社,2002:11-13.

[6] GIARRATANO J, RILEY G. Expert systems principles and programming[M]. Beijing: China Machine Press, 2000:220-227.

(上接第 606 页)

[2] LINDSTROM P, PASCUCCI V. Visualization of large terrains made easy[C]//Proc of IEEE Visualization. 2001:363-370.

[3] CLARK J H. Hierarchical geometric models for visible surface algorithms [J]. Communication of the ACM, 1996, 19(10): 547-554.

[4] GERSMER T. Top-down view-dependent terrain triangulation using the octagon metric[C]//Proc of Eurographics Symposium on Geometry Processing. 2003:1-11.

[5] EVANS W, KIRKPATRICK D, TOWNSEND G. Right-triangulated irregular networks[J]. Algorithmica, 2001, 30(2):264-286.

[6] BAO Xiao-hong, PAJAROLA R. LOD-based clustering techniques for efficient large-scale terrain storage and visualization[C]//Proc of SPIE Conference on Visualization and Data Analysis. 2003:225-235.

[7] 陈刚,夏青,万刚. 地形 RSG 模型的动态构网算法[J]. 测绘学报,2002,31(1):44-48.

[8] LINDSTROM P, KOLLER D, RIBARSKY W, *et al*. Real-time continuous level of detail rendering of height fields[C]//Proc of ACM SIGGRAPH. 1996:109-118.

[9] 殷宏,许继恒,周良伟,等. 基于限制四叉树的大规模地形可视化及其实现[J]. 计算机应用研究,2006,23(5):151-153.

[10] 苏虎,周美玉. 一种大规模地形的实时绘制算法[J]. 武汉大学学报:工学版,2003,36(3):81-85.

[11] 见英,叶榛. 一种实时视景仿真中高度场地形绘制算法[J]. 系统仿真学报,2005,17(1):83-86.

[12] 李清泉. 三维空间数据的实时获取、建模与可视化[M]. 武汉:武汉大学出版社,2003:236-237.